Simulating long-term food producing capacities in China using a Web-based land evaluation system

Langetermijnssimulaties van voedsel-productiecapaciteit in China door middel van webgebaseerde landevaluatie

Volume I. Text

Liming Ye

Thesis submitted in fulfillment of the requirements for the degree of Doctor (Ph.D.) in Geology

Proefschrift voorgedragen tot het bekomen van de graad van Doctor in de Geologie

Op gezag van

Rector: Prof. Dr. P. Van Cauwenberge

Decaan: Prof. Dr. H. Dejonghe

Promotor: Prof. Dr. E. Van Ranst
Acknowledgements

First of all I would like to express my profound gratitude to my promotor, Prof. Dr. E. Van Ranst, for his invaluable encouragement, supervision and useful suggestions throughout this research work. His moral support and continuous guidance enabled me to complete my dissertation successfully. I am also highly thankful to Prof. Dr. P. Finke and Dr. A. Verdoodt for their constructive suggestions and valuable discussions throughout the work.

I am grateful for the cooperation of many of my colleagues. The draft dissertation has been partly or entirely proofread by E.V.R., P.F. and A.V. J. Van de Wauw answered many of my statistical questions in a handy way. R. Moussadek shared his GIS expertise with me during the formation of the idea of grid-based simulation. D. Langouche translated the summary of my dissertation into Dutch. Prof. H. Tang of the Chinese Academy of Agricultural Sciences provided the Chinese soil survey report, which was extremely useful. I’d like also to thank all the administrative staffs at the Laboratory of Soil Science, where my research took place, for their help of many kinds.

I am especially indebted to my wife, Dr. Q. Yang and my little daughter J.Z. Ye. They both firmly supported me either mentally or materially. Most of my time and energy have been invested into my work during the past years. But your support is not in vain. I am equally indebted to my parents and to the whole family for their faith on me throughout my life.

I would also like to extend special thanks to the Ghent University for providing financial support in the name of the Special Research Fund (BOF). Without it, the work was totally impossible.

Liming Ye, April 2008
# Contents

Acknowledgements ........................................... i

List of Tables .................................................. vii

List of Figures ................................................... ix

Abbreviations ..................................................... xiii

List of Symbols .................................................... xv

Summary .......................................................... xvii

Samenvatting ...................................................... xxi

1 Introduction .................................................... 1

1.1 Problems .................................................. 1

1.1.1 Danger of food insecurity ................................ 1

1.1.2 Loss of cropland ......................................... 2

1.1.3 Soil degradation ......................................... 2

1.1.4 Need for long-term strategies ......................... 3

1.2 Objectives .................................................. 3

2 Agricultural Natural Resources Base in China .......... 5

2.1 Introduction ................................................ 5

2.2 Geographic subdivisions of China ....................... 6

2.2.1 Administrative subdivisions at the provincial level .... 7

2.2.2 Agro-ecological zones ................................. 7

2.2.3 Socio-economic belts .................................... 7

2.3 Grid representation of the study area .................. 9

2.4 Topography ................................................ 12

2.4.1 Data source ............................................. 12

2.4.2 Topographic map of China ............................ 12

2.5 Land cover types and agricultural cropland ........ 13

2.6 Climate ..................................................... 13

2.6.1 Climatic stations ...................................... 13

2.6.2 Spatial interpolation techniques ...................... 15

2.6.3 Grid generation with ordinary kriging ............... 21

2.6.4 Climatic grids ......................................... 28

2.6.5 Climate classification based on aridity index ........ 28
2.7 Soil .......................................................... 28
  2.7.1 Soil mapping unit .................................. 29
  2.7.2 FAO soil units in China ........................... 30
  2.7.3 DSMW rasterization ................................. 30
  2.7.4 Virtual soil profile ................................. 31
  2.7.5 Case study: SOC density and storage in cropland of China .... 33

3 Cropping Systems in Food Production .................. 38
  3.1 Introduction ........................................... 38
  3.2 Definition of food and grain crops .................. 38
  3.3 Crop phenology ...................................... 39
    3.3.1 Crop development stages ....................... 40
    3.3.2 Crop growth periods ........................... 42
  3.4 Crop rotations ...................................... 43
  3.5 Sown structure of food crops ........................ 45

4 Quantitative Land Evaluation Methodologies .......... 46
  4.1 Introduction .......................................... 46
  4.2 Reference evapotranspiration ....................... 46
    4.2.1 Combination equation ........................... 47
    4.2.2 Aerodynamic term ................................ 47
    4.2.3 Radiation term ................................... 48
    4.2.4 Annual $ET_0$ map of China ..................... 49
  4.3 Growing period ...................................... 49
    4.3.1 Concept ........................................... 49
    4.3.2 Determination ................................... 50
    4.3.3 Length of growing period inventory in China .... 53
  4.4 Radiation-thermal production potential ............ 53
    4.4.1 Net biomass production .......................... 53
    4.4.2 Respiration ...................................... 54
    4.4.3 Maximum gross biomass production rate ........... 55
    4.4.4 Total net biomass production .................... 55
  4.5 Water-limited production potential ................. 56
    4.5.1 Maximum crop evapotranspiration ................ 56
    4.5.2 Plant available soil water capacity ............. 59
    4.5.3 Effective rainfall ................................ 61
    4.5.4 Actual crop evapotranspiration .................. 63
    4.5.5 Anticipated rainfed crop yield ................... 66
  4.6 Land production potential .......................... 67
    4.6.1 Soil index ...................................... 67
    4.6.2 Management index .............................. 70

5 The Web-based Land Evaluation System ............... 76
  5.1 Introduction .......................................... 76
  5.2 System structure .................................... 76
    5.2.1 Standalone models ................................ 77
    5.2.2 Multi-tier structure of the WLES ................. 77
    5.2.3 Subsystems ...................................... 78
CONTENTS

8.5 Food production scenarios ............................................. 121
  8.5.1 Urbanization scenarios ........................................... 121
  8.5.2 Cropland scenarios ................................................ 123
  8.5.3 Cropping intensity scenarios .................................... 124
  8.5.4 Soil degradation scenarios ....................................... 124
8.6 Food producing capacities in 2030 and 2050 ....................... 126
  8.6.1 Soil degradation effect ........................................... 126
  8.6.2 Food producing capacities ....................................... 129
8.7 Long-term food security in China ................................... 131
  8.7.1 Per capita supply .................................................. 131
  8.7.2 Per capita demand .................................................. 132
  8.7.3 Food security index ............................................... 134

9 Conclusions and Recommendations ..................................... 136
  9.1 General conclusions .................................................. 136
  9.2 Methodological innovations ......................................... 137
    9.2.1 The WLES .......................................................... 138
    9.2.2 Applicability ...................................................... 138
  9.3 Uncertainties ............................................................ 138
  9.4 Policy options toward food security ............................... 139
    9.4.1 Soil management strategies .................................... 140
  9.5 Recommendations for future researches ........................... 142

A DEM-GRID Conversion ..................................................... 144
  A.1 About USGS GTOPO30 DEM .......................................... 144
    A.1.1 Geographic coverage ............................................. 144
    A.1.2 Grid spacing and resolution .................................... 144
    A.1.3 Need for projection ............................................. 144
    A.1.4 Applicability ...................................................... 145
  A.2 About Arc/Info GRID ................................................ 145
    A.2.1 Arc/Info binary grid ............................................. 145
    A.2.2 Arc/Info ASCII grid ............................................. 147
  A.3 USGS GTOPO30 DEM to Arc/Info grid conversion ................ 149
    A.3.1 Byte order ........................................................ 149
    A.3.2 Conversion and display ......................................... 149

B List of Climatic Stations ................................................. 151
  B.1 WMO meta data ....................................................... 151
  B.2 Data source and parameter completeness ........................ 160

C FAO Soil Units and Symbols ............................................. 169

D Records of Observed Yields and Factor Inputs ...................... 171

E Atlas ................................................................................. 173
  E.1 Topographic map of China .......................................... 174
  E.2 Land cover map of China in the year 2000 ....................... 175
  E.3 Map of cropland in China in the year 2000 ....................... 176
  E.4 Completeness and integrity of the climatic dataset .......... 177
List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td></td>
<td>List of provinces, metropolitans, autonomous regions and SARs in China</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td>List of agro-ecological zones in China</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td></td>
<td>Income and development disparity among provinces in 2000</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td></td>
<td>Definition of the socio-economic belts in China</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td></td>
<td>FAO LCCS classes for China in the year 2000</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td>The International Standard Atmosphere</td>
<td>21</td>
</tr>
<tr>
<td>2.7</td>
<td></td>
<td>Statistical parameters and tests for the measured and predicted $T_{max1}$ values</td>
<td>25</td>
</tr>
<tr>
<td>2.8</td>
<td></td>
<td>Aridity index-based climate classification</td>
<td>28</td>
</tr>
<tr>
<td>2.9</td>
<td></td>
<td>Composition rules for the propositions of dominant and component soils in a mapping unit</td>
<td>29</td>
</tr>
<tr>
<td>2.10</td>
<td></td>
<td>Full composition of the mapping unit $Bd35$</td>
<td>30</td>
</tr>
<tr>
<td>2.11</td>
<td></td>
<td>Composition of the mapping units of Cambisols in China</td>
<td>32</td>
</tr>
<tr>
<td>2.12</td>
<td></td>
<td>Zone-specific soil organic carbon storage (Tg) in cropland of China</td>
<td>36</td>
</tr>
<tr>
<td>2.13</td>
<td></td>
<td>Estimations of cropland SOC storage (Pg) in China</td>
<td>37</td>
</tr>
<tr>
<td>3.1</td>
<td></td>
<td>Sown area, production and productivity of food crops in China, 2005</td>
<td>39</td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td>Crop development stage durations (days) of food crops per zone</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td></td>
<td>Mapping crop development stages into crop growth periods</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td></td>
<td>Zone-specific cropping patterns</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td></td>
<td>Sown area of major food crops in China, 2005 ($\times 1,000$ ha)</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td></td>
<td>Components of a growing period</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td></td>
<td>Growing period in Chengdu, days</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>Soil requirement table of summer maize</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td></td>
<td>Number of equal sections and weighting factors for each section</td>
<td>69</td>
</tr>
<tr>
<td>4.5</td>
<td></td>
<td>Single factor production functions with the food productivity (t ha$^{-1}$) as the dependent and the factor input as the independent variable</td>
<td>72</td>
</tr>
<tr>
<td>4.6</td>
<td></td>
<td>Management index of individual food crops and crop groups at the high, intermediate and low input levels</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>Listing of evaluation phases in IM</td>
<td>81</td>
</tr>
<tr>
<td>5.2</td>
<td></td>
<td>High level overview of the WLES Class Library</td>
<td>84</td>
</tr>
<tr>
<td>6.1</td>
<td></td>
<td>Effective irrigation rates for typical agricultural systems</td>
<td>88</td>
</tr>
<tr>
<td>6.2</td>
<td></td>
<td>Province-specific effective irrigation rate, 2005</td>
<td>89</td>
</tr>
<tr>
<td>6.3</td>
<td></td>
<td>Province-specific factor inputs, 2005</td>
<td>89</td>
</tr>
<tr>
<td>6.4</td>
<td></td>
<td>Definition of the high, intermediate and low input levels</td>
<td>90</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Single factor production functions with the yield of an individual food crop $(\text{t ha}^{-1})$ as the dependent and the overall score of factor inputs as the independent variable</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>Simulated and observed yields of rice, wheat, maize, sorghum and millet</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>Simulated and observed yields of beans, tubers; and collective grain and food crops</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>Simulated versus observed yields of food crops: a summary</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>Results of the Shapiro-Wilk, F- and paired t-tests on simulated and observed yields of major food crops</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Summary of soil characteristic-crop yield regression</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>Summary of soil characteristic-crop yield variance analysis</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>Summary of soil index-crop yield regression</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>Summary of soil index-crop yield variance analysis</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>Summary of the management index-crop yield regression</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>Summary of management index-crop yield variance analysis</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>7.7</td>
<td>Summary of soil index-management index-crop yield variance analysis</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>7.8</td>
<td>Summary of soil index-management index-crop yield regression</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>8.1</td>
<td>Chinese population during the 20th century</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>8.2</td>
<td>Reconstructed time series of cropland area in China, in comparison to population size and urbanization rate on a yearly basis, 1949-2005</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>8.3</td>
<td>Impact class of soil degradation on crop yield adopted in ASSOD</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>8.4</td>
<td>Extent of soil degradation defined in ASSOD</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>8.5</td>
<td>Impact of soil degradation on crop yield classification by overall score</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>8.6</td>
<td>Reported effects of soil degradation on crop yield</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>8.7</td>
<td>Relative yield loss (%) due to soil degradation with regard to input levels</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>8.8</td>
<td>Relative yield loss (%) due to soil degradation at regional and national scales</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>8.9</td>
<td>Sown area, cropland and the multi-cropping index, 1978-2005</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>8.10</td>
<td>Scenario settings for 2030 and 2050</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>8.11</td>
<td>China’s food producing capacities in the years 2030 and 2050</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>8.12</td>
<td>Estimation of the per capita food consumption, 2003-2050</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>Ground distance covered by 30 arc-seconds at different latitudes</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>A.2</td>
<td>File structure and functions of Arc/Info Binary Grid</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>A.3</td>
<td>File structure and contents of an Arc/Info ASCII Grid</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>B.1</td>
<td>List of climatic stations: meta data</td>
<td>151</td>
<td></td>
</tr>
<tr>
<td>B.2</td>
<td>List of climatic stations: data source and parameter completeness</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>D.1</td>
<td>Records of observed yields of food crops in kg ha$^{-1}$, 1978-2005</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>D.2</td>
<td>Agricultural factor inputs, 1978-2005</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>E.1</td>
<td>List of maps in Appendix E</td>
<td>173</td>
<td></td>
</tr>
</tbody>
</table>
# List of Figures

1.1 *Per capita production and consumption of food, 1978-2005* ........................................ 2  
1.2 *Diminishing cropland area in contrast to growing population, 1979-2005* ................. 3  

2.1 *Geographic location and territory of China* .............................................................. 6  
2.2 *Provinces, metropolitans, autonomous regions and SARs in China* ......................... 8  
2.3 *Agro-ecological zones in China* ................................................................................. 9  
2.4 *Socio-economic belts in China* ................................................................................. 11  
2.5 *Spatial distribution of the point dataset of selected climatic stations* .................... 15  
2.6 *Length of station records (years) of the climatic datasets* ...................................... 16  
2.7 *Spatial distribution of the climatic stations on a per parameter basis* ..................... 16  
2.8 *Spatial lay-out of locations over a virtual terrain with both measured and unmeasured values* ........................................................................................................ 17  
2.9 *Some commonly used deterministic and geostatistical interpolators* ....................... 18  
2.10 *Temperature-elevation relationship in a standard atmosphere* .............................. 22  
2.11 *Pre-kriging data examination* .................................................................................. 23  
2.12 *Global trend identification for the maximal air temperature in January* ................ 24  
2.13 *Semivariogram modeling of $T'_{max1}$ in ArcGIS Geostatistical Analysis* .......... 25  
2.14 *Cross validation of the empirical semivariogram model* ........................................ 26  
2.15 *Standard error of the ordinary kriging interpolator for $T'_{max1}$* ............................ 27  
2.16 *Soil mapping units of Cambisols in China, identified on a $0.5\times0.5$ grid version of the DSMW* ................................................................................................. 31  
2.17 *Vector to raster (V2R) conversion of the soil map* .................................................. 33  
2.18 *Spatial patterns of cropland SOC density in China* ................................................ 34  
2.19 *Regional fluctuation and variability of cropland SOC density in China* ............... 35  

3.1 *China’s food production (1949-2005) and rank as a major food producer (1957-2005)* ................................................................. 39  
3.2 *Spatial distribution of food crops and species* .......................................................... 40  
3.3 *Cropping calendar of major food crops per zone* ...................................................... 44  

4.1 *Growing period determination using multi-year mean climatic data of Chengdu* .... 51  
4.2 *End of the growing period determination* ................................................................. 53  
4.3 *Length of growing period inventory, analysis of causes for absence of growing period and LGP-based agro-climatic zoning* ............................................................... 54  
4.4 *$K_c$ values of food crops sown in Chengdu, Southwest China* ............................. 58  
4.5 *Rooting depth dynamics during crop cycle of major food crops cultivated in Chengdu, Southwest China* ......................................................... 60  
4.6 *Average soil water depletion fraction (%) of food crops, 2005* ............................... 62
LIST OF FIGURES

4.7 Rainfall effectiveness in response to crop evapotranspiration and total rainfall 64
4.8 Yield response to soil water availability and actual crop evapotranspiration . 65
4.9 Relationships between the food productivity and the input factors of effective irrigation rate, applications of the nitrate, phosphorus, potassium and compound fertilizers, and the electricity consumption ........................... 71
4.10 Evolution of the overall score of factor inputs and the relationship between the average yield of food crops and the overall score of factor inputs .... 73
4.11 Wheat yield in response to the overall score of factor inputs, plotted on the logarithmic-logarithmic scales to demonstrate the determination and the meaning of the coefficients a and b of the fitted production function .... 74
4.12 Management index of food crops in China at homogeneously implemented high, intermediate and low input levels ........................................ 75

5.1 Block diagram of the WLES tier structure ................................. 77
5.2 Functional flow chart of the WLES at the subsystem level. Shaded entities are subsystems or groups of subsystems ................................. 78
5.3 WLES user authentication ...................................................... 79
5.4 WLES user authorization ........................................................ 80
5.5 Functional flow chart of the WLES-BM ..................................... 80
5.6 Annotated WLES Web interface of the WLES-Home .................... 81
5.7 Annotated WLES Web interface of the Effective Rainfall Phase ......... 82
5.8 Graphical presentation of the growing period, crop cycle determinations and the RPP result ................................................................. 85
5.9 Graphical presentation of the WBS result and the data download interface 86
5.10 Graphical presentation of the crop requirement table, the soil index and the LPP determinations ............................................................. 86

6.1 Yields of food crops (t ha⁻¹) in response to the overall score of factor inputs 91
6.2 Management index of food crops in 2005 with high, intermediate and low input levels for the east, middle and west economic development belts, respectively .......................................................... 93
6.3 Area-based irrigation rate and time-based irrigation probability ........ 94
6.4 Comparability of means and goodness of fit between simulated and observed yields of rice, wheat, maize, sorghum and millet ...................... 97
6.5 Comparability of means and goodness of fit between simulated and observed yields of beans, tubers, and collective grain and food crops .... 98

7.1 Soil characteristic (CaCO₃, gypsum, CEC and BS)-crop yield relationships 101
7.2 Soil characteristic (pH, SOC, ECe and ESP)-crop yield relationships . 102
7.3 Soil index-crop yield relationship for major crop species and groups .... 104
7.4 Management index-crop yield relationship for major crop species and groups 106
7.5 Improvement of the yield predictability with soil and management indices together over soil index alone as the predicting variable(s) ............ 109

8.1 China’s population growth: positive and normative analysis ............... 111
8.2 Relative rural-urban population composition in China, 1978-2005 .......... 112
8.3 Modeling annual growth rate of urbanization in China, 1978-2005 (a) against the generalized low-high-low urbanization growth pattern (b) .......... 113
8.4 Cropland area as explained by urbanization rate, 1980-2005 .............. 116
8.5 Impact class of common types of soil degradation on crop yield in China . 118
8.6 Overall impact class of soil degradation on crop yield in China ........... 119
8.7 Effect of soil degradation in terms of relative yield loss .................... 122
8.8 The urbanization scenarios 2030 and 2050 ................................. 123
8.9 Fluctuations and trend of cropland availability, 1949-2050 ................. 124
8.10 Trend and scenarios of the multi-cropping index in China, 1978-2050 .... 125
8.11 Effect of soil degradation on overall yield of food crops .................. 128
8.12 Compositions of H-I-L cropland classes in 2030 and 2050 ................. 129
8.13 China’s food producing capacities, 1949-2050 ............................ 130
8.14 China’s per capita food supply, 1949-2050 ................................. 132
8.15 China’s per capita food demand, 1949-2050 ............................... 133
8.16 Food security index, 1949-2050 .............................................. 134
9.1 Food security index under raised management levels for 2030 and 2050 141
9.2 Soil management strategies based on farmer’s response to soil degradation 142
E.1 Topographic map of China .................................................... 174
E.2 Land cover map of China in the year 2000 .................................. 175
E.3 Map of China’s agricultural cropland in the year 2000 ..................... 176
E.4 Completeness and integrity of the climatic dataset .......................... 177
E.5 Interpolated mean maximal air temperatures in January before and after elevation adjustment .................................................. 178
E.6 Monthly mean maximal air temperature class, January-December .... 179
E.7 Monthly precipitation class, January-December ............................. 180
E.8 Annual mean minimal air temperature class ................................ 181
E.9 Annual mean frequency of precipitation ...................................... 182
E.10 Annual mean relative air humidity .......................................... 183
E.11 Annual mean daily sunshine duration ....................................... 184
E.12 Annual mean wind velocity .................................................... 185
E.13 Aridity index-based climate classification ................................. 186
E.14 FAO soil groupings in China .................................................. 187
E.15 FAO soil units in Zone 5 ....................................................... 188
E.16 Annual reference evapotranspiration, mm .................................. 189
E.17 Net biomass production, harvest index and the RPP of food crops in 2005 190
E.18 Plant available soil water content and capacity for and average rooting depth of food crops in 2005 ........................................... 191
E.19 Combined rainfall effectiveness for food crops in 2005 .................... 192
E.20 Average $K_c$, $ET_m$ and $ET_a$ during crop cycle of food crops in 2005 193
E.21 $ET$ deficit, yield response factor, water stress coefficient and the WPP of food crops in 2005 ..................................................... 194
E.22 The most limiting soil characteristics for food crops in 2005 .............. 195
E.23 Soil index for food production in 2005 ..................................... 196
E.24 Cropland productivity change under the business-as-usual scenario by 2030 and 2050 .................................................. 197
E.25 Cropland productivity change under the double-degradation scenario by 2030 and 2050 ................................................ 198
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>aridity index</td>
</tr>
<tr>
<td>ALES</td>
<td>Automated Land Evaluation System</td>
</tr>
<tr>
<td>AML</td>
<td>Arc Macro Language</td>
</tr>
<tr>
<td>ANOVA</td>
<td>analysis of variance</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>a.s.l.</td>
<td>above sea level</td>
</tr>
<tr>
<td>ASSOD</td>
<td>Human Induced Soil Degradation in South and Southeast Asia</td>
</tr>
<tr>
<td>BAU</td>
<td>business as usual</td>
</tr>
<tr>
<td>BMC</td>
<td>biomass calculator</td>
</tr>
<tr>
<td>BS</td>
<td>base saturation</td>
</tr>
<tr>
<td>CAS</td>
<td>Chinese Academy of Sciences</td>
</tr>
<tr>
<td>CAAS</td>
<td>Chinese Academy of Agricultural Sciences</td>
</tr>
<tr>
<td>CCIA</td>
<td>climate change impact assessment</td>
</tr>
<tr>
<td>CDIAC</td>
<td>Carbon Dioxide Information Analysis Center, USDoE</td>
</tr>
<tr>
<td>CD</td>
<td>crop development</td>
</tr>
<tr>
<td>CDS</td>
<td>crop development stage</td>
</tr>
<tr>
<td>CEC</td>
<td>cation exchange capacity</td>
</tr>
<tr>
<td>CGP</td>
<td>crop growth period</td>
</tr>
<tr>
<td>DEM</td>
<td>digital elevation model, equivalent to digital terrain model (DTM)</td>
</tr>
<tr>
<td>DSMW</td>
<td>FAO/UNESCO Digital Soil Map of the World</td>
</tr>
<tr>
<td>DSSAT</td>
<td>Decision Support System for Agrotechnology Transfer</td>
</tr>
<tr>
<td>EC</td>
<td>electrical conductivity</td>
</tr>
<tr>
<td>EOF</td>
<td>end-of-file</td>
</tr>
<tr>
<td>ESP</td>
<td>exchangeable sodium percentage</td>
</tr>
<tr>
<td>ESRI</td>
<td>Environmental Systems Research Institute</td>
</tr>
<tr>
<td>FAO</td>
<td>Food and Agriculture Organization of the United Nations</td>
</tr>
<tr>
<td>FC</td>
<td>field capacity</td>
</tr>
<tr>
<td>FSI</td>
<td>food security index</td>
</tr>
<tr>
<td>GDP</td>
<td>gross domestic income</td>
</tr>
<tr>
<td>GIS</td>
<td>geographical information system</td>
</tr>
<tr>
<td>GHG</td>
<td>green house gas</td>
</tr>
<tr>
<td>GLASOD</td>
<td>Global Assessment of Human Induced Soil Degradation</td>
</tr>
<tr>
<td>GLF</td>
<td>Great Leap Forward</td>
</tr>
<tr>
<td>GP</td>
<td>growing period</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>hyper-text transfer protocol</td>
</tr>
<tr>
<td>IAP</td>
<td>Institute of Atmospheric Physics, CAS</td>
</tr>
<tr>
<td>ICAO</td>
<td>International Civil Aviation Organization</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>IDW</td>
<td>inverse distance weighting</td>
</tr>
<tr>
<td>IQR</td>
<td>interquartile range</td>
</tr>
<tr>
<td>IS</td>
<td>initial stage</td>
</tr>
<tr>
<td>ISA</td>
<td>International Standard Atmosphere</td>
</tr>
<tr>
<td>ISRIC</td>
<td>International Soil Reference and Information Center</td>
</tr>
<tr>
<td>ISSAS</td>
<td>Institute of Soil Science, CAS</td>
</tr>
<tr>
<td>JRC</td>
<td>European Commission Joint Research Center</td>
</tr>
<tr>
<td>LAI</td>
<td>leaf area index</td>
</tr>
<tr>
<td>LCCS</td>
<td>land cover classification system (FAO)</td>
</tr>
<tr>
<td>LE</td>
<td>land evaluation</td>
</tr>
<tr>
<td>LGP</td>
<td>length of growing period</td>
</tr>
<tr>
<td>LLE</td>
<td>land limitations evaluator</td>
</tr>
<tr>
<td>LOOCV</td>
<td>leave-one-out cross-validation</td>
</tr>
<tr>
<td>LPP</td>
<td>land production potential</td>
</tr>
<tr>
<td>LS</td>
<td>late season</td>
</tr>
<tr>
<td>MCI</td>
<td>multi-cropping index</td>
</tr>
<tr>
<td>MDG</td>
<td>Millennium Development Goal</td>
</tr>
<tr>
<td>MOA</td>
<td>Ministry of Agriculture, China</td>
</tr>
<tr>
<td>MOC</td>
<td>Ministry of Construction, China</td>
</tr>
<tr>
<td>MS</td>
<td>mid-season</td>
</tr>
<tr>
<td>MU</td>
<td>(FAO) mapping unit</td>
</tr>
<tr>
<td>NARPC</td>
<td>National Agricultural Regional Planning Council</td>
</tr>
<tr>
<td>NBSC</td>
<td>National Bureau of Statistics of China</td>
</tr>
<tr>
<td>NOARP</td>
<td>National Office of Agricultural Regional Planning</td>
</tr>
<tr>
<td>NSSO</td>
<td>National Soil Survey Office of China</td>
</tr>
<tr>
<td>PAW</td>
<td>plant available water</td>
</tr>
<tr>
<td>PWP</td>
<td>permanent wilting point</td>
</tr>
<tr>
<td>RPP</td>
<td>radiation-thermal production potential</td>
</tr>
<tr>
<td>SAR</td>
<td>special administrative region</td>
</tr>
<tr>
<td>SEB</td>
<td>socio-economic belt</td>
</tr>
<tr>
<td>SOC</td>
<td>soil organic carbon</td>
</tr>
<tr>
<td>SOCD</td>
<td>soil organic carbon density</td>
</tr>
<tr>
<td>SOCS</td>
<td>soil organic carbon storage</td>
</tr>
<tr>
<td>SPAC</td>
<td>soil-plant-atmosphere continuum</td>
</tr>
<tr>
<td>V2R</td>
<td>vector to raster conversion</td>
</tr>
<tr>
<td>UNESCO</td>
<td>United Nations Educational, Scientific and Cultural Organization</td>
</tr>
<tr>
<td>UNDP</td>
<td>United Nations Development Programme</td>
</tr>
<tr>
<td>UNEP</td>
<td>United Nations Environment Programme</td>
</tr>
<tr>
<td>UNFPA</td>
<td>United Nations Population Fund</td>
</tr>
<tr>
<td>USDA</td>
<td>United States Department of Agriculture</td>
</tr>
<tr>
<td>USGS</td>
<td>United States Geological Survey</td>
</tr>
<tr>
<td>WBS</td>
<td>water-balance simulator</td>
</tr>
<tr>
<td>WISE</td>
<td>world inventory of soil emission</td>
</tr>
<tr>
<td>WLES</td>
<td>Web-based Land Evaluation System</td>
</tr>
<tr>
<td>WMO</td>
<td>World Meteorological Organization</td>
</tr>
<tr>
<td>WOFOST</td>
<td>WOrld FOod STudies</td>
</tr>
<tr>
<td>WPP</td>
<td>water-limited production potential</td>
</tr>
<tr>
<td>WRI</td>
<td>World Resources Institute</td>
</tr>
</tbody>
</table>
List of Symbols

\( \alpha \) \hspace{1cm} \text{albedo or canopy reflection coefficient}
\( b_c \) \hspace{1cm} \text{maximum gross biomass production rate on clear days, kg CH}_2\text{O ha}^{-1} \text{ day}^{-1}
\( B_g \) \hspace{1cm} \text{gross biomass production, kg CH}_2\text{O ha}^{-1}
\( bgm \) \hspace{1cm} \text{maximum gross biomass production rate, kg CH}_2\text{O ha}^{-1} \text{ day}^{-1}
\( B_m \) \hspace{1cm} \text{accumulated net biomass, kg CH}_2\text{O ha}^{-1}
\( B_n \) \hspace{1cm} \text{net biomass production, kg CH}_2\text{O ha}^{-1}
\( bma \) \hspace{1cm} \text{average net biomass production rate, kg CH}_2\text{O ha}^{-1} \text{ day}^{-1}
\( bnm \) \hspace{1cm} \text{maximum net biomass production rate, kg CH}_2\text{O ha}^{-1} \text{ day}^{-1}
\( b_o \) \hspace{1cm} \text{maximum gross biomass production rate on overcast days, kg CH}_2\text{O ha}^{-1} \text{ day}^{-1}
\( cf \) \hspace{1cm} \text{coarse fragments, w\%}
\( ct \) \hspace{1cm} \text{respiration coefficient, \textit{dimensionless}}
\( \Delta \) \hspace{1cm} \text{slope of the vapor pressure curve, kPa °C}^{-1}
\( d_{ij} \) \hspace{1cm} \text{Euclidean distance between locations } S_i \text{ and } S_j
\( \epsilon(S) \) \hspace{1cm} \text{random, autocorrelated error in location } S
\( e_a \) \hspace{1cm} \text{actual vapor pressure, kPa}
\( e_s \) \hspace{1cm} \text{saturated vapor pressure, kPa}
\( \varepsilon' \) \hspace{1cm} \text{net emissivity}
\( ET_a \) \hspace{1cm} \text{actual evapotranspiration, mm day}^{-1}
\( ET_{aero} \) \hspace{1cm} \text{aerodynamic term of the reference evapotranspiration, mm day}^{-1}
\( ET_c \) \hspace{1cm} \text{maximal crop evapotranspiration, mm day}^{-1}. \text{Also see } ET_m
\( ET_m \) \hspace{1cm} \text{maximal crop evapotranspiration, mm day}^{-1}. \text{Also see } ET_c
\( ET_o \) \hspace{1cm} \text{potential or reference evapotranspiration, mm day}^{-1}
\( ET_{rad} \) \hspace{1cm} \text{radiation term of the reference evapotranspiration, mm day}^{-1}
\( f \) \hspace{1cm} \text{fraction of daytime that sky is overcast, \textit{dimensionless}}
\( f_W \) \hspace{1cm} \text{yield reduction coefficient due to water stress, \textit{dimensionless}}
\( \gamma_{ij} \) \hspace{1cm} \text{semivariance between locations } i \text{ and } j
\( h_c \) \hspace{1cm} \text{height of crop, m}
\( K \) \hspace{1cm} \text{absolute temperature}
\( K_c \) \hspace{1cm} \text{crop coefficient}
\( KLAI \) \hspace{1cm} \text{maximum growth rate ratio, \textit{dimensionless}}
\( k_y \) \hspace{1cm} \text{yield response factor, \textit{dimensionless}}
\( L \) \hspace{1cm} \text{length of crop cycle, day}
\( LAI \) \hspace{1cm} \text{leaf area index, m}^2 \text{ m}^{-2}
\( \lambda \) \hspace{1cm} \text{latent heat of evaporation, MJ kg}^{-1}
\( \lambda_i \) \hspace{1cm} \text{weighting factor assigned to location } i
\( \mu(S) \) \hspace{1cm} \text{deterministic trend in location } S
\( M_y \) \hspace{1cm} \text{management index, \textit{dimensionless}, } [0-1]
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>daily sunshine duration, hr</td>
</tr>
<tr>
<td>N</td>
<td>daylength, hr</td>
</tr>
<tr>
<td>P</td>
<td>rainfall, mm</td>
</tr>
<tr>
<td>p</td>
<td>soil water depletion fraction, dimensionless</td>
</tr>
<tr>
<td>P_{eff}</td>
<td>effective rainfall, mm</td>
</tr>
<tr>
<td>P_{freq}</td>
<td>rainfall frequency, rains month^{-1}</td>
</tr>
<tr>
<td>P_m</td>
<td>maximum leaf photosynthesis rate, kg CH_{2}O ha^{-1} hr^{-1}</td>
</tr>
<tr>
<td>r</td>
<td>effective irrigation rate, %</td>
</tr>
<tr>
<td>R</td>
<td>respiration lossess, kg CH_{2}O ha^{-1}</td>
</tr>
<tr>
<td>R_a</td>
<td>extra terrestrial radiation, MJ m^{-2} day^{-1}</td>
</tr>
<tr>
<td>R_a</td>
<td>aerodynamic resistance, s m^{-1}</td>
</tr>
<tr>
<td>R_b</td>
<td>net outgoing radiation, MJ m^{-2} day^{-1}</td>
</tr>
<tr>
<td>R_c</td>
<td>canopy resistance, s m^{-1}</td>
</tr>
<tr>
<td>RE</td>
<td>rainfall effectiveness, dimensionless</td>
</tr>
<tr>
<td>RH</td>
<td>relative air humidity, %</td>
</tr>
<tr>
<td>r_m</td>
<td>maximum respiration rate, kg CH_{2}O ha^{-1} day^{-1}</td>
</tr>
<tr>
<td>R_n</td>
<td>net radiation, MJ m^{-2} day^{-1}</td>
</tr>
<tr>
<td>R_{ns}</td>
<td>net incoming radiation, MJ m^{-2} day^{-1}</td>
</tr>
<tr>
<td>\rho</td>
<td>soil bulk density, kg M^{-3}</td>
</tr>
<tr>
<td>\sigma</td>
<td>Stefan Boltzmann constant, 4.903 \times 10^{-9} MJ m^{-2} K^{-4} day^{-1}</td>
</tr>
<tr>
<td>S_o</td>
<td>prediction location</td>
</tr>
<tr>
<td>S_a</td>
<td>plant available soil water, v/v% or cm m^{-1}</td>
</tr>
<tr>
<td>S_aD</td>
<td>plant available soil water over the root zone, mm (water depth)</td>
</tr>
<tr>
<td>pS_aD</td>
<td>easily available soil water over the root zone, mm (water depth)</td>
</tr>
<tr>
<td>S_i</td>
<td>the i^{th} location</td>
</tr>
<tr>
<td>S_{iD}</td>
<td>available soil water capacity during unit time t</td>
</tr>
<tr>
<td>S_T_e</td>
<td>soil water storage at the end of a period, mm</td>
</tr>
<tr>
<td>S_T_i</td>
<td>initial soil water storage, mm</td>
</tr>
<tr>
<td>S_T_p</td>
<td>provisional soil water storage, mm</td>
</tr>
<tr>
<td>S_T_t</td>
<td>soil water storage at time t, mm</td>
</tr>
<tr>
<td>S_y</td>
<td>soil index, dimensionless, [0-1]</td>
</tr>
<tr>
<td>\gamma</td>
<td>psychrometric constant, kPa °C^{-1}</td>
</tr>
<tr>
<td>T_d</td>
<td>mean daytime temperature, °C</td>
</tr>
<tr>
<td>T_{max}</td>
<td>maximal air temperature, °C</td>
</tr>
<tr>
<td>T_{min}</td>
<td>minimal air temperature, °C</td>
</tr>
<tr>
<td>T_n</td>
<td>mean nighttime temperature, °C</td>
</tr>
<tr>
<td>u</td>
<td>wind velocity, m s^{-1}</td>
</tr>
<tr>
<td>u_2</td>
<td>wind velocity measured at 2 m height, m s^{-1}</td>
</tr>
<tr>
<td>Y_{irri}</td>
<td>irrigated crop yield, kg ha^{-1}</td>
</tr>
<tr>
<td>Y_{rain}</td>
<td>rainfed crop yield, kg ha^{-1}</td>
</tr>
<tr>
<td>Z(S_i)</td>
<td>value at location i in spatial interpolation</td>
</tr>
<tr>
<td>\tilde{Z} (S_0)</td>
<td>predicted value at location 0</td>
</tr>
</tbody>
</table>
Summary

China is since long time the most populous country on earth. The enthusiasm to monitor its food security has never faded. Many efforts have been made in China in maintaining a relatively stable supply of food to its ever-growing population during the past few decades. However, the delicate equilibrium between per capita demand and supply of food has become more and more fragile. Primarily driven by rapid urbanization, China’s cropland has been lost at an average rate of 0.5 million hectares per year during the 1979-2005 period, and will continue to be lost in the foreseeable future. This, together with the observed (a) decrease in soil quality resulted from erosion, mismanagement and environmental pollution, (b) sensitivity of per capita production of food to climatic perturbations, and (c) inter-annual fluctuations of per capita food surplus during the post-1978 period, has casted tremendous doubts over China’s ability in fulfilling food self-sufficiency in the long run. From a global point of view, maintaining food security has been recognized as one of the prioritized Millennium Development Goals (MDGs) by the international community. Realizing it in China in the 21st century means the establishment of the livelihood and welfare of more than one-fifth of the entire population of the world.

This research presents a modeling approach to assess the long-term food producing capacities – and consequently food security – in China using a Web-based land evaluation system (WLES, http://weble.ugent.be) as the evaluation engine, a grid-based GIS as the scale regulator and a relational database as the data manipulator. The WLES implements a three-step, hierarchical, deterministic land evaluation model, based for specific crops on the radiation regime, and the water-limited and land production potentials. Functional components such as the biomass calculator (BMC), the water balance simulator (WBS) and the land limitations evaluator (LLE) compute these potentials, respectively. Homogeneous 5 km by 5 km grid data sets were created to store the spatial distribution of the climatic, crop and soil parameters. Using the cell number as the key, all data were stored in the relational database for fast and easy access. Food production reality in 2005 was simulated by considering, among others, the land use type and distribution, province-specific cropping system patterns, and the intensity of factor inputs, represented by (a) application of fertilizers and chemicals, (b) agro-machinery usage and electrical consumption, and (c) irrigation infrastructure investment. The simulation process was looped to iterate all the 778,104 grid cells, and output was stored in the database on a per cell basis. The simulation results were cross-analyzed against the historic and current yield records of the crop variety-input level combinations for each province in order to validate the model applied.

Food productions in 2030 and 2050 were simulated using production scenarios which involved population growth, urbanization rate, cropland area, cropping intensity, management level, and soil degradation. Three soil degradation scenarios – namely, “no-degradation” (0×SD), “business-as-usual” (BAU) and “double-degradation” (2×SD) –
were designed and included in the food production simulation. A food security index (FSI) – or the relative food surplus in per capita terms – was proposed to reveal the historical fluctuations and future trends of food security in China, and to help formulate policy options from the simulated effects of soil degradation and management level on FSI.

The model predicted that food crops may experience a 9.7% productivity loss by 2030 if the soil is degraded at the current rate (BAU scenario); productivity loss will increase to an unbearable level of 36.7% by 2050, should the soil be twice more degraded than it is now (2×SD scenario). China’s food producing capacity tends to decline in the long run if the general trend of soil degradation will not be reverted. China will be able to achieve a production of 428.6 million tons (i.e., a 11.5% decrease from the 2005 level or -11.5%, same below) from food crops in 2030 and 409.1 million tons (-15.5%) in 2050 under the BAU scenario. Under the 2×SD scenario, the production is projected to be 390.9 million tons (-19.2%) in 2030 and 326 million tons (-32.6%) in 2050, which are levels that China had surpassed in the late 1980s and the late 1970s, respectively. However, the population in 2050, for instance, will be 37% more than that in the late 1970s, suggesting a 37% decrease in per capita food availability during the time interval between the two. Under the 0×SD scenario, the food producing capacities in 2030 and 2050 will be 3.4% lower and 1.1% higher than the 2005 level, respectively, showing that the negative effect of the decreasing cropland area is nearly neutralized by the positive effect of the increasing cropping intensity on food production in the long run. This in turn suggests that the above-mentioned declining trend in food production can actually be recognized as the net effect of soil degradation on food production.

In per capita terms, the FSI value is predicted to drop from 12.7 in 2005 to -9.8 and -7.5 in 2030 and 2050, respectively, under the 0×SD scenario – meaning that the demand-supply equilibrium is changed from a surplus of 12.7% in 2005 to a deficit of 9.8% and 7.5% in 2030 and 2050, respectively, even under the ‘no-degradation’ scenario. This, together with the observation that the positive effect of a higher cropping intensity cancels the negative effect of a shrinking cropland area, suggests that the present-day producing capacity (2005 level) will not be able to sustain the long-term needs, which are associated with a higher per capita demand, despite no further soil degradation is assumed. Under the BAU scenario, the FSI value will be further decreased to -17.3 and -22.6 – suggesting that 17.3% and 22.6% of per capita food demand will not be met – in 2030 and 2050, respectively. Under the 2×SD scenario, the FSI value is predicted to be as low as -24.5 and -38.3 – suggesting that food shortage will be as high as 24.5% and 38.3% – in 2030 and 2050, respectively.

If the management level in 2030 is raised from high-intermediate-low in 2005 to high-high-intermediate for the east-middle-west economic development belts, and in 2050 to high-high-high, the FSI will increase from -17.3 to -10.9 in 2030 and from -22.6 to -2.1 in 2050 under the BAU scenario, and from -24.5 to -18.7 in 2030 and from -38.3 to -10.7 in 2050 under the 2×SD scenario, respectively.

Summary of major findings

The methodologies applied in this large-scale quantitative assessment of the long-term food producing capacities in China are proved to be successful and accurate. Data analysis reveals that (a) limiting soil characteristics should be collectively evaluated in order to derive a single soil index; and (b) soil and management indices should be used together in yield prediction. It is generally recommendable that the quantitative land evaluation
Methodologies should be used in its entirety in productivity assessment applications. The modeling results suggest that (a) soil degradation is the most significant single factor that adversely affects China’s food producing capacity in the long run; (b) the present-day producing capacity (2005 level) will not be able to sustain the long-term needs under the current management level even if soil degradation is not becoming more limiting; and (c) China is facing great challenges in safeguarding its food security by 2030/2050. The detrimental effect of soil degradation on food security is so evident that technical measures and policy levers must be activated today in order to avoid, or at least mitigate, the risks of food insecurity tomorrow.

Policy implications

Predicting food security outcomes has been part of the policy landscape since long. Major findings of this dissertation give rise to the following policy implications:

- Strong policy interventions must be immediately implemented in order to guide the technical countermeasures in safeguarding the national, or global, food security in the long run;

- Achieving food security needs institutional and investment reforms. Progressive policy actions must not only increase agricultural production, but also boost incomes and reduce poverty in rural areas where most of the poor live. Development programmes should be (re)focused on the integrated and efficient utilization of agricultural resources based on local ecological conditions;

- Early warning systems are needed in fighting food security especially in food-insecure regions;

- Agricultural investments in research and in field need to be strengthened. Such investments are essential yet efficient means not only in driving technical and institutional changes but also in mitigating soil degradation’s impact on food security;

- Environmental damages caused by production intensification must be controlled. Although a high potential of improvement exists for the long-term food security in China, the fulfillment of it is possible only if the intensive food production is maintained with managed environmental damages;

- Major breakthroughs are needed in basic plant physiology, agroecology, and soil science to achieve the long-term food security and environmental integrity.

Methodological innovations

The following innovations have been made regarding the methodologies of this dissertation:

- The characterization of input level creates a sound basis for the evaluation of the effect of management practices on crop yield; It helps identify the effect of agricultural investment on national food security, especially under the scenario of a decreasing soil quality;

- The introduction of the deterministic Cobb-Douglas production function into the otherwise empirical evaluation process of the management index;
The proposed food security index (FSI) attaches not only a magnitude but also a sign to the food security concept.
Samenvatting

China is van oudsher het meest bevolkte land op aarde. Er is steeds aandachtig toegezien op voedselzekerheid. De afgelopen tientallen jaren zijn in China vele inspanningen geleverd om voor de nog steeds groeiende bevolking de voedselvoorziening relatief stabiel te houden. Nochtans wordt het delicate evenwicht tussen per capita vraag en aanbod van voedsel steeds fragieler. In hoofdzaak door snelle verstedelijking is in de periode 1979-2005 een jaarlijks verlies aan landbouwland vastgesteld van 0.5 miljoen hectare, een trend die zich in de volgende jaren zal doorzetten. Samen met de waargenomen (a) vermindering van bodemkwaliteit ten gevolge van erosie, wanbeheer en milieuvvervuiling, (b) gevoeligheid van de per capita voedselopbrengst voor klimaatsstoornissen, en (c) jaarlijkse schommelingen van het per capita voedseloverschot gedurende de periode na 1978, heeft dit ernstige twijfels doen rijzen over de capaciteiten van China om op lange termijn de eigen voedselzekerheid te verzekeren. De internationale gemeenschap heeft het behoud van voedselzekerheid erkend als een van de prioritaire Millennium Ontwikkelingsdoelstellingen. Voor China betekent dit dat het in de 21

Deze thesis stelt een model voor waarmee de voedselproductiecapaciteit in China op lange termijn – en dus de voedselzekerheid – kan geschat worden, gebruikmakend van het Web-gebaseerd LandEvaluatieSysteem (WLES, http://weble.ugent.be) als evaluatiewerktuig, een raster-gebaseerde GIS om de schaal te regelen en een relationele databank om gegevens te bewerken. Het WLES implementeert een hiërarchisch deterministisch landevaluatiemodel in drie stappen, dat gebaseerd is op de begroting van het radiatiethermisch productiepotentieel (FAO methodologie), het water-gelimiteerd productiepotentieel en het landproductiepotentieel van de beschouwde landbouwgewassen. Deze potentiëlen worden berekend door functionele componenten zoals respectievelijk de biomassacalculator (BMVC), de waterbalanssimulator (WBS) en de evaluator van landbeperkingen (landlimitatie-evaluator of LLE). Er worden datasets gecreëerd volgens homogene rasters van 5 km op 5 km, om de ruimtelijke spreiding van klimaats-, gewas- en bodemparameters te slaan. De gegevens werden opgeslagen in de relationele databank, gebruik makend van het celnummer als sleutel voor vlugge en eenvoudige ontsluiting. De voedselproductie in 2005 werd gesimuleerd door rekening te houden met onder andere landgebruikstype en verspreiding ervan, landbouwproductiesystemen eigen aan de provincies, intensiteit van gebruik van productiemiddelen zoals (a) meststoffen en chemicaliën, (b) landbouwmachines en elektriciteitsverbruik, en (c) investering in infrastructuur voor irrigatie. Het proces werd gesimuleerd op alle 778.104 rastercellen, en de uitkomst werd per cel opgeslagen in de databank. Om het gebruikte model te valideren werden de simulatieresultaten in elke provincie getoetst aan de historische en reële gewasopbrengsten voor elke combinatie van gewasvariëteit en productiemiddelenniveau.

De voedselproductie voor 2030 en 2050 werd gesimuleerd gebruikmakend van produc-
SAMENVATTING

tiescenario’s steunend op bevolkingsgroei, toenemende verstedelijking, bebouwbare landbouwoppervlakte, teeltintensiteit, beheersniveau en bodemdegradatie. Drie bodemdegradatiescenario’s - namelijk “zero-degradatie” (0×SD), “gangbare degradatie” of “business-as-usual” (BAU) en “dubbele degradatie” (2×SD) werden uitgewerkt en gebruikt voor de simulaties. Om de historische schommelingen en de toekomstige tendensen in voedselzekerheid in China aan te tonen, werd een voedselzekerheidsindex (FSI) voorgesteld, die het relatieve voedseloverschot per capita uitdrukt. De studie van het gesimuleerde effect van bodemdegradatie en beheersniveau op deze FSI is een hulpmiddel voor het formuleren van mogelijke beleidsmaatregelen.

Het model voorspelde dat de voedselgewassen een productiviteitsverlies van 9,7% kunnen bereiken in 2030 indien de bodem verder degradeert aan het huidige tempo (BAU scenario); het productiviteitsverlies neemt toe tot een onaanvaardbaar niveau van 36,7% in 2050 indien de bodem verder degradeert aan een tempo dat tweemaal zo hoog is als het huidige (2×SD scenario). De voedselproductiecapaciteit in China daalt op langere indien geen halt wordt toegeroepen aan de algemene trend in bodemdegradatie. Onder het BAU scenario zal China in staat zijn om een productie te bereiken van 428,6 miljoen ton voedselgewassen in 2030 (d.i. een afname van 11,5% ten opzichte van het niveau in 2005, of -11,5%), en 409,1 miljoen ton (-15,5%) in 2050. Onder het 2×SD scenario wordt de productie in 2030 geschat op 390,9 miljoen ton (-19,2%) en op 326 miljoen ton (-32,6%) in 2050, productieniveaus die in China werden overschreden sinds respectievelijk de late jaren ’80 en ’70. Nochtans zal de bevolkingsdichtheid in 2050 ongeveer 37% hoger zijn dan in de late jaren ’70, hetgeen een 37% afname van de per capita voedselreserves betekent over dit tijdspanne. Onder het 0×SD-scenario zal de voedselproductiecapaciteit in 2030 en 2050 respectievelijk 3,4% lager en 1,1% hoger zijn dan in vergelijking met die van 2005, wat aantoont dat het negatieve effect op de voedselproductie tengevolge van de vermindering van het landbouwareaal op lange termijn geneutraliseerd wordt door het positieve effect van een verhoogde landbouwintensiteit. Dit suggerert dat de hoger vermelde neerwaartse trend van de voedselproductie kan gezien worden als het nettoresultaat van bodemdegradatie.

Per capita uitgedrukt, wordt voorspeld dat de FSI onder het 0×SD scenario zal dalen van een waarde van 12,7 in 2005 tot waarden van -9,8 in 2030 en -7,5 in 2050. Dit impliceert dat, zelfs wanneer de huidige bodemdegradatie kan worden gestopt, het evenwicht tussen voedselvraag en voedselproductie evolueert van een 12,7% surplus in 2005 naar een tekort van 9,8% in 2030 en 7,5% in 2050. Samen met de vaststelling dat het positieve effect van een hogere landbouwintensiteit het negatieve effect van een kleiner landbouwareaal teniet doet, suggerert dit dat de huidige productiecapaciteit (niveau 2005) niet in staat zal zijn om de behoeften op lange termijn, welke geassocieerd zijn met een hogere per capita vraag, te voldoen, alhoewel geen verdere bodemdegradatie verondersteld wordt. Onder het BAU-scenario neemt de FSI verder af tot -17,3 in 2030 en -22,6 in 2050 waardoor respectievelijk 17,3% en 22,6% van de per capita vraag naar voedsel niet zal beantwoord kunnen worden. Onder het 2×SD-scenario zal het voedseltekort in 2030 24,5% en in 2050 38,3% bedragen.

Indien het beheersniveau in de oost-mid-westelijke economische ontwikkelingsgordels stijgt van hoog-intermediair-laag in 2005 naar hoog-hoog-intermediair in 2030 en naar hoog-hoog-hoog in 2050, zal de FSI onder het BAU-scenario toenemen van -17,3 tot -10,9 in 2030 en van -22,6 tot -2,1 in 2050, en onder het 2×SD-scenario van -24,5 tot -18,7 in 2030 en van -38,3 tot -10,7 in 2050.
Samenvatting van de voornaamste resultaten

Er wordt aangetoond dat de methodologieën die in deze grootschalige kwantitatieve lange termijnschattingen van de voedselproductiecapaciteit van China gebruikt worden, succesvol en accuraat zijn. Uit het gegevensonderzoek blijkt dat (a) de limiterende bodemeigenschappen collectief moeten geëvalueerd worden om één enkele bodemindex te bekomen; en (b) bodem- en beheersindices gezamenlijk moeten gebruikt worden voor opbrengstvoorspellingen. Het is algemeen aanbevolen om, in toepassingen voor het schatten van de productiviteit, kwantitatieve landevaluatiemethodologieën in hun geheel te gebruiken.

De resultaten bekomen door modellering suggereren dat (a) bodemdegradatie de meest belangrijke enkelvoudige factor is die op lange termijn een negatief effect heeft op de voedselproductiecapaciteit in China; (b) de huidige productiecapaciteit (niveau 2005) onder het huidige beheersniveau niet in staat zal zijn om de voedselbehoefte op lange termijn te dekken, zelfs wanneer bodemdegradatie niet langer een beperkende factor is; (c) China voor grote uitdagingen staat voor wat betreft het veiligstellen van de voedselverzekering tegen 2030/2050. Het nefaste effect van bodemdegradatie op voedselzekerheid is zo evident dat technische maatregelen en stimulansen vanuit het beleid vandaag genomen moeten worden om het risico van voedselonzekerheid te verminderen, zo niet te voorkomen.

Implicaties voor het beleid

Voorspellingen rond voedselveiligheid maken sedert lang deel uit van het politiek landschap. De voornaamste resultaten van dit doctoraatsonderzoek leiden tot de volgende implicaties met betrekking tot het beleid:

- er moeten onmiddellijk sterke beleidsinterventies geïmplementeerd worden als ondersteuning voor technische maatregelen die de nationale, of wereldwijde, voedselzekerheid op lange termijn moeten veiligstellen;

- er zijn institutionele en investeringshervormingen nodig om voedselzekerheid te bereiken. Progressieve beleidsmaatregelen moeten niet alleen de landbouwproductie doen toenemen, maar ook inkomen genereren en armoede bestrijden in rurale gebieden, waar de meeste armen wonen. Ontwikkelingsprogramma’s moeten zich (her)oriënteren naar een geïntegreerd en efficiënt gebruik van landbouwproductiemiddelen, rekening houdend met lokale ecologische omstandigheden;

- vroege alarmsystemen zijn vereist om de voedselzekerheid te controleren in voedselonzekerere regio’s;

- op lange termijn is er in China een groot potentieel voor verbetering van voedselzekerheid, indien intensive productie gerealiseerd wordt onder een hoog investeringsniveau, en met aanpak van milieuschade;

- grote doorbraken zijn vereist op het gebied van plantenfysiologie, agroëcologie, en bodemkunde om op lange termijn zowel voedselzekerheid als milieu-integriteit te bereiken.
Methodologische vernieuwingen

Aangaande de methodologiën die in deze studie gebruikt werden, werden de volgende vernieuwingen gerealiseerd:

- de karakterisering van productiemiddelenniveau’s is een gezonde maar eenvoudige basis voor de evaluatie van het effect van beheerspraktijken op gewasopbrengsten; het helpt bij het identificeren van het effect van de landbouwinvesteringen op de nationale voedselzekerheid, in het bijzonder onder het scenario van een afnemende bodemkwaliteit;

- de introductie van de deterministische Cobb-Douglasproductiefunctie in het anders empirische evaluatieproces van de beheersindex;

- de voorgestelde voedselzekerheidsindex (FSI) geeft niet alleen een grootteorde, maar ook een richting aan het concept van voedselzekerheid.
Chapter 1

Introduction

Soil plays an irreplaceable role in realizing food security for humanity on earth. Before the dawn of agriculture, the hunter-gathering lifestyle supported ~4 million people globally (Cohen, 1995). Modern agriculture now feeds 6,000 million people. During the past 40 years, global cereal production has doubled, which reveals that a remarkable increase in soil productive capacity has been achieved from significant technological progress in food production (Tilman et al., 2002). The “Green Revolution”, for instance, realized by changes in plant architecture, improved harvest index and photoperiod insensitivity (Busch et al., 1984; Dyck et al., 2004), resulting in the growth rate in food production exceeding the growth rate in population (Conway, 1999). Recent deceleration in food production growth rates due to a combination of adverse meteorological, eco-environmental and marketing factors poses a new challenge to global food security (Swaminathan, 2007).

In China, traditional soil use and management techniques, which were accumulated and inherited from its permanent arable farming practices since ancient times (Chang et al., 2002; Yang, 2006), have played an important role in achieving food self-sufficiency (FAO, 2007) during the country’s long recorded history (Chang et al., 2002). China’s population size had been tripled during the 20th century alone, from 430 million in 1900 to 1,267 million in 2000 (NBSC, 2000). The total production of food largely kept up with the population growth (Qu and Li, 1994), although natural catastrophes and/or policy failures has caused major disturbances, as observed, for instance, during the Great Leap Forward period (1958-1960) (Aston et al., 1984; Smil, 1999b).

1.1 Problems

1.1.1 Danger of food insecurity

China has been the world’s largest grain producer since the late 1980s or early 1990s, shortly after the rural reform and the “open-door” policies had been adopted. China produced 484 million tons of grain in 2005 (NBSC, 2005), which was 10% higher than the second largest producer, the USA. In a broader context, China has supported 22% of the world’s population with only 10% of the world’s cropland. This is indeed remarkable achievement. However, the overall equilibrium between the production and consumption of food has been becoming more and more fragile during the past few decades. The per capita food production was observed to increase from 317 kg in 1978 to 370 kg in 2005, with an annual growth rate of 0.6%. The per capita consumption, on the other hand, was steadily increased from 291 kg in 1978 to 328 kg in 2005, with an annual growth rate of
0.5% (Figure 1.1). Although the average growth rate of the former is still higher than that of the latter during the post-1978 period, the decreasing trend of the per capita food surplus, together with the sensitivity of the per capita production to climatic variations, signal a strong early warning over the possibility of food insecurity in China in the 21st century.

![Figure 1.1: Per capita production (squares) and consumption (triangles) of food, 1978-2005, based on total production, population (NBSC, 1999, 2005) and consuming patterns of the rural and urban residents (Gao, 2004; Jiang and Davis, 2007). Black squares indicate years in which per capita production is greater than or equal to 400 kg](image)

### 1.1.2 Loss of cropland

China has witnessed serious loss of cropland during the post-1978 period, coinciding with fast economic expansion and rapid urbanization. A reconstructed time series of cropland area in China, based on multiple data sources (Crook, 1993; Frolking et al., 1999; Smil, 1999a; Xiao et al., 2003; Feng et al., 2005; Liu et al., 2005a), shows that agricultural cropland was lost at a steady rate of 0.26 and 1.45 million hectares per year during the 1978-1999 and 2000-2005 period, respectively (Figure 1.2). Recognized as the primary driver for the shrinkage of cropland (Yang and Li, 2000; Chen, 2007), urbanization in China is currently in its accelerating phase (Zhao, 2005; Chen, 2007), suggesting that the adverse effect of urbanization on cropland availability – and consequently on food supply – will be strong and long-lasting.

### 1.1.3 Soil degradation

Continuous cultivation of cereal crops under the mounting pressure of an ever increasing population has resulted in various forms of soil degradation in China, including water and
wind erosion, fertility depletion, physical deterioration, salinisation, etc. (Yang, 2006). Soil degradation leads to soil quality decrease (Sun et al., 2003), which in turn undermines the productive capacity of soil resources.

1.1.4 Need for long-term strategies

Maintaining soil quality is vital for the resilience and productivity of soils, especially in resource-scarce countries, like China. While food shortage and malnutrition are still seen in many parts of the world (FAO, 2007), tackling the problem requires not only short-term remedies but also long-term solutions that are ecologically sound and compatible with local natural and socio-economic configurations (Rosegrant and Cline, 2003). A larger population size and demand for a richer diet are believed to considerably raise the threshold of food security in the long run (Tilman et al., 2002).

1.2 Objectives

Therefore, the objectives of this dissertation are:

1. To conduct a systematic inventory of the agricultural natural resources in China;

2. To create a Web-based Land Evaluation System (WLES), which implements a three-step, hierarchical, deterministic land evaluation model; and to apply it in production simulations using climatic, soil, crop and management data;
(3) To simulate China’s food production reality in 2005 and to validate the WLES by comparing the simulated and observed yields using statistical analysis;

(4) To assess the long-term food producing capacities in China using production scenarios involving population growth, urbanization rate, cropland area, cropping intensity, management level, and, in particular, soil degradation; and

(5) To formulate soil management strategies and policy options toward food security in China in the long run.
Chapter 2

Agricultural Natural Resources Base in China

2.1 Introduction

China is one of the largest countries of the world, with a rich and varied physical environment which has been shaped by the interaction and integration of many factors, including vast area, mid-latitudinal location, mountainous topography, complex geological history, significant impact of human activities, and so forth.

China is located at the east coast of the largest continent (Eurasia) and the western margin of the largest ocean (Pacific) on earth. It has a land area of about 9.6 million km$^2$, occupying 6.5% of the world’s total. It spans 5,200 km from the confluence of the Heilong River and its tributary, the Wusuli River, in the east to the Pamir Plateau in the west, and 5,500 km from the midstream of the Heilong River in the north to Zengmu Shoal of the Nansha Islands in the south (Figure 2.1). The country covers the geographic coordinates of 3°50′ to 53°31′ N of the latitudes and 73°40′ to 135°05′ E of the longitudes (Zhao, 1994).

Temperate to subtropical continental climates dominate in the country.

During the long historical period of the Chinese civilization, agriculture had been in the center stage and enormous efforts had been made to help it develop. The living of the *Homo erectus*, or the “upright man”, tribes in China can be dated back to as early as 1.6 million years ago (Tattersall and Schwartz, 2001). Agricultural activities in the Loess Plateau, for instance, began at the Banpo neolithic village near Xi’an at about 5,000 BC to 4,000 BC (Chang et al., 2002; Liu, 2005b). Archeological evidences of the Hemudu culture, which were discovered in places to the south of the Hangzhou Bay in Zhejiang, revealed that rice cultivation flourished there during the period of 5,000 BC to 4,500 BC (Chang et al., 2002). Although formed much later, China’s traditional agriculture is believed to have uninterruptedly survived for 7 to 8 centuries. Its techniques are still being used in agricultural production in modern times.

At present, China has about 130 million hectares of cropland (WRI, 2005) and a population of more than 1.3 billion people (NBSC, 2005). Practically no virgin land still exists due to the very fact that the human inhabitants have made their imprints nearly everywhere across the country. Humanity is obviously one of the most active driving forces in the formation and evolution of agricultural systems. However, its effects on agricultural productivity will only be discussed in later chapters (4, 6 and 8).

Because food is overwhelmingly produced on the mainland and on major off-shore
2.2 Geographic subdivisions of China

The most natural way to manipulate a large dataset is to divide it into smaller subsets which can then be linked to geographic subdivisions of the study area. The administrative regions (§2.2.1), agro-ecological zones (AEZ) (§2.2.2) and socio-economic belts (SEB)
2.2 Geographic subdivisions of China

§2.2.3) are three ways applied in this research to construct the geographic subdivisions in China.

### 2.2.1 Administrative subdivisions at the provincial level

There are 33 administrative subdivisions in China, including 23 provinces, 3 metropolitans, 5 autonomous regions and 2 special administrative regions (SARs), as listed in Table 2.1 and shown in Figure 2.2.

One remark has to be made on the exclusion of Chongqing in the list. Chongqing had been promoted as a new metropolitan in 1997 and thus administratively and statistically detached from the Sichuan province thereafter. But to keep data consistency and comparability within a time series, for instance, of population, before and after 1997, Chongqing is intentionally viewed as part of Sichuan.

#### Table 2.1: List of provinces, metropolitans, autonomous regions and SARs in China

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Type†</th>
<th>No.</th>
<th>Name</th>
<th>Type†</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Beijing</td>
<td>M</td>
<td>18</td>
<td>Hunan</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>Tianjin</td>
<td>M</td>
<td>19</td>
<td>Guangdong</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>Hebei</td>
<td>P</td>
<td>20</td>
<td>Guangxi</td>
<td>AR</td>
</tr>
<tr>
<td>4</td>
<td>Shanxi</td>
<td>P</td>
<td>21</td>
<td>Hainan</td>
<td>P</td>
</tr>
<tr>
<td>5</td>
<td>Nei Mongol</td>
<td>AR</td>
<td>22</td>
<td>Sichuan‡</td>
<td>P</td>
</tr>
<tr>
<td>6</td>
<td>Liaoning</td>
<td>P</td>
<td>23</td>
<td>Guizhou</td>
<td>P</td>
</tr>
<tr>
<td>7</td>
<td>Jiín</td>
<td>P</td>
<td>24</td>
<td>Yunnan</td>
<td>P</td>
</tr>
<tr>
<td>8</td>
<td>Heilongjiang</td>
<td>P</td>
<td>25</td>
<td>Xizang</td>
<td>AR</td>
</tr>
<tr>
<td>9</td>
<td>Shanghai</td>
<td>M</td>
<td>26</td>
<td>Shaanxi</td>
<td>P</td>
</tr>
<tr>
<td>10</td>
<td>Jiangsu</td>
<td>P</td>
<td>27</td>
<td>Gansu</td>
<td>P</td>
</tr>
<tr>
<td>11</td>
<td>Zhejiang</td>
<td>P</td>
<td>28</td>
<td>Qinghai</td>
<td>P</td>
</tr>
<tr>
<td>12</td>
<td>Anhui</td>
<td>P</td>
<td>29</td>
<td>Ningxia</td>
<td>AR</td>
</tr>
<tr>
<td>13</td>
<td>Fujian</td>
<td>P</td>
<td>30</td>
<td>Xinjiang</td>
<td>AR</td>
</tr>
<tr>
<td>14</td>
<td>Jiangxi</td>
<td>P</td>
<td>31</td>
<td>Taiwan</td>
<td>P</td>
</tr>
<tr>
<td>15</td>
<td>Shangdong</td>
<td>P</td>
<td>32</td>
<td>Hongkong</td>
<td>SAR</td>
</tr>
<tr>
<td>16</td>
<td>Henan</td>
<td>P</td>
<td>33</td>
<td>Macao</td>
<td>SAR</td>
</tr>
<tr>
<td>17</td>
<td>Hubei</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

† M=metropolitan, P=province, AR=autonomous region, SAR=special administrative region

‡ Chongqing is counted as part of Sichuan

### 2.2.2 Agro-ecological zones

The country has been divided into 9 agro-ecological zones (NARPC, 1981) in terms of climate, landform and soils, and/or land cover, with a specific range of potentials and constraints for land use (FAO, 1996), as listed in Table 2.2 and shown in Figure 2.3.

### 2.2.3 Socio-economic belts

Recently economic growth and development in China have been extraordinary. However, regional disparities of development in China have been observed as an intrinsic character at
2.2 Geographic subdivisions of China

Figure 2.2: Provinces, metropolitans, autonomous regions and SARs in China, with subdivision number (Table 2.1) annotated on map

Table 2.2: List of agro-ecological zones in China

<table>
<thead>
<tr>
<th>No.</th>
<th>Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Northeast</td>
</tr>
<tr>
<td>2</td>
<td>Nei Mongol &amp; Great-Wall Adjacency</td>
</tr>
<tr>
<td>3</td>
<td>Huang-Huai-Hai Plain</td>
</tr>
<tr>
<td>4</td>
<td>Loess Plateau</td>
</tr>
<tr>
<td>5</td>
<td>Middle &amp; Lower Reaches of the Yangtze River</td>
</tr>
<tr>
<td>6</td>
<td>Southwest</td>
</tr>
<tr>
<td>7</td>
<td>South</td>
</tr>
<tr>
<td>8</td>
<td>Gansu &amp; Xinjiang</td>
</tr>
<tr>
<td>9</td>
<td>Qinghai-Tibetan Plateau</td>
</tr>
</tbody>
</table>

Source: NARPC (1981)

the current stage of economic expansion (Chen, 2002). The eastern and coastal provinces and municipalities have developed at a faster pace than its interior counterparts, leading to an obvious disparity. As shown in Table 2.3, the top 10 provinces and municipalities
with the highest per capita gross domestic income (GDP) are mostly from the eastern and coastal regions; and the 10 least developed provinces are mostly geographically located in the west.

As such, the east, middle and west socio-economic belts (SEBs) have been defined (NBSC, 2003) on the basis of the similarities of the socio-economic development status and the neighborhood topological relationships among the administrative subdivisions, as listed in Table 2.4 and shown in Figure 2.4. Because rapid economic expansion and subsequently social restructuring processes have been ongoing for years in China, initiatives had been put forward to reshaping the current SEBs since the national economic and development policies are differentiated for different SEBs (Liu, 2005a,c). Nevertheless, the SEBs discussed here are still officially adopted.

2.3 Grid representation of the study area

In order to keep compatibility among datasets of different sources and to facilitate subsequent processes of data processing, the study area was divided into 808 rows by 963 columns of grid cells, each of 25 km$^2$. A unique serial number, running from 1 at the...
Table 2.3: *Income and development disparity among provinces in 2000*

<table>
<thead>
<tr>
<th>Prov†</th>
<th>Per Capita GDP (¥)</th>
<th>Urban:Rural Income Ratio</th>
<th>City and Town Population (%)</th>
<th>Non-agricultural Employment (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17,936</td>
<td>2.25</td>
<td>77.54</td>
<td>88.30</td>
</tr>
<tr>
<td>2</td>
<td>16,377</td>
<td>2.25</td>
<td>71.99</td>
<td>80.10</td>
</tr>
<tr>
<td>3</td>
<td>7,545</td>
<td>2.28</td>
<td>26.08</td>
<td>51.20</td>
</tr>
<tr>
<td>4</td>
<td>4,985</td>
<td>2.48</td>
<td>34.91</td>
<td>53.30</td>
</tr>
<tr>
<td>5</td>
<td>5,896</td>
<td>2.51</td>
<td>42.68</td>
<td>45.50</td>
</tr>
<tr>
<td>6</td>
<td>11,017</td>
<td>2.27</td>
<td>54.24</td>
<td>62.30</td>
</tr>
<tr>
<td>7</td>
<td>6,675</td>
<td>2.38</td>
<td>49.68</td>
<td>49.80</td>
</tr>
<tr>
<td>8</td>
<td>8,818</td>
<td>2.29</td>
<td>51.54</td>
<td>50.50</td>
</tr>
<tr>
<td>9</td>
<td>27,178</td>
<td>2.09</td>
<td>88.34</td>
<td>86.90</td>
</tr>
<tr>
<td>10</td>
<td>11,539</td>
<td>1.89</td>
<td>41.49</td>
<td>57.80</td>
</tr>
<tr>
<td>11</td>
<td>12,906</td>
<td>2.18</td>
<td>48.67</td>
<td>62.20</td>
</tr>
<tr>
<td>12</td>
<td>5,075</td>
<td>2.74</td>
<td>27.81</td>
<td>40.20</td>
</tr>
<tr>
<td>13</td>
<td>11,293</td>
<td>2.30</td>
<td>41.57</td>
<td>53.10</td>
</tr>
<tr>
<td>14</td>
<td>4,838</td>
<td>2.39</td>
<td>27.67</td>
<td>48.10</td>
</tr>
<tr>
<td>15</td>
<td>9,409</td>
<td>2.44</td>
<td>38.00</td>
<td>46.90</td>
</tr>
<tr>
<td>16</td>
<td>5,550</td>
<td>2.40</td>
<td>23.20</td>
<td>35.90</td>
</tr>
<tr>
<td>17</td>
<td>7,757</td>
<td>2.43</td>
<td>40.22</td>
<td>52.00</td>
</tr>
<tr>
<td>18</td>
<td>5,732</td>
<td>2.83</td>
<td>29.75</td>
<td>39.20</td>
</tr>
<tr>
<td>19</td>
<td>11,180</td>
<td>2.67</td>
<td>55.00</td>
<td>58.90</td>
</tr>
<tr>
<td>20</td>
<td>4,500</td>
<td>3.13</td>
<td>28.15</td>
<td>37.80</td>
</tr>
<tr>
<td>21</td>
<td>6,587</td>
<td>2.46</td>
<td>40.11</td>
<td>39.70</td>
</tr>
<tr>
<td>22</td>
<td>4,979</td>
<td>3.21</td>
<td>29.89</td>
<td>41.95</td>
</tr>
<tr>
<td>23</td>
<td>2,818</td>
<td>3.73</td>
<td>23.87</td>
<td>32.70</td>
</tr>
<tr>
<td>24</td>
<td>4,559</td>
<td>4.28</td>
<td>18.93</td>
<td>26.10</td>
</tr>
<tr>
<td>25</td>
<td>4,483</td>
<td>4.84</td>
<td>23.36</td>
<td>26.20</td>
</tr>
<tr>
<td>26</td>
<td>4,607</td>
<td>3.55</td>
<td>32.26</td>
<td>41.30</td>
</tr>
<tr>
<td>27</td>
<td>3,838</td>
<td>3.44</td>
<td>24.01</td>
<td>40.30</td>
</tr>
<tr>
<td>28</td>
<td>5,088</td>
<td>3.47</td>
<td>34.76</td>
<td>39.10</td>
</tr>
<tr>
<td>29</td>
<td>4,725</td>
<td>2.58</td>
<td>32.43</td>
<td>42.20</td>
</tr>
<tr>
<td>30</td>
<td>7,087</td>
<td>3.60</td>
<td>33.82</td>
<td>42.30</td>
</tr>
<tr>
<td>Country</td>
<td>7,039</td>
<td>2.79</td>
<td>36.22</td>
<td>50.00</td>
</tr>
</tbody>
</table>

† See Table 2.1 for names

Source: (NBSC, 2000)
2.3 Grid representation of the study area

Table 2.4: Definition of the socio-economic belts in China

<table>
<thead>
<tr>
<th>SEB</th>
<th>Provinces†</th>
<th>Area (%)</th>
<th>Topography, climate</th>
<th>Agricultural suitability</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>1, 2, 3, 6, 9, 10, 11, 13, 15, 19, 20, 21, 31, 32, 33</td>
<td>14.3</td>
<td>flat, humid</td>
<td>highly suitable</td>
</tr>
<tr>
<td>Middle</td>
<td>4, 5, 7, 8, 12, 14, 16, 17, 18</td>
<td>29.3</td>
<td>hilly, humid to semi-arid</td>
<td>suitable</td>
</tr>
<tr>
<td>West</td>
<td>22, 23, 24, 25, 26, 27, 28, 29, 30</td>
<td>56.4</td>
<td>highlands, arid</td>
<td>marginal</td>
</tr>
</tbody>
</table>

† See Table 2.1

Figure 2.4: Socio-economic belts in China. Source: NBSC (2003)

upper-left to 778,104 at the lower-right corner of the coverage, was assigned to every cell. This cell number was used as an index in all subsequent map and database operations.

Suppose $i$ and $j$ represent the column and line number respectively. The cell at the junction of the $i^{th}$ column and the $j^{th}$ row is then solely identified by its cell number, $n$:

$$n = [\max (i)] \cdot (j - 1) + i$$  \hspace{1cm} (2.1)

where $i = 1 \cdots 963$ and $j = 1 \cdots 808$. 
2.4 Topography

Thematic layers of DEM (§2.4), land cover (§2.5), climatic (§2.6), soil (§2.7), crop (chapter 3) and management (chapter 6) parameters were first geo-referenced, projected (§A.2.1) and transformed into the Lambert Azimuthal Equal Area (§A.2.1) coordinate system (ESRI, 2004) with 105° as the central meridian and 44° as the original latitude, and then rasterized to Arc/Info GRID format with 5 km × 5 km as the resolution. Like other binary grids, the exported ASCII format (§A.2.2) of the Arc/Info GRID can easily be read, processed or generated by third-party programs and utilities. More technical details of the Arc/Info Grid and the conversion from DEM to Grid are documented in Appendix A.

2.4 Topography

2.4.1 Data source

The USGS 30 arc-second global elevation dataset (GTOPO30 DEM), resulted from a collaborative effort led by the U.S. Geological Survey’s EROS (Earth Resources Observation and Science) Data Center, was used as the data source to create a homogeneous topographic coverage of China. Elevations in the original GTOPO30 dataset are regularly spaced at 30 arc-seconds (approximately 1 km at the equator), but irregularly spaced in terms of distance between two neighboring cells at different latitudes (Table A.1). Nevertheless, the data set provides sufficient topographic details at the national scale of China (USGS, 1996).

2.4.2 Topographic map of China

Four tiles of the USGS GTOPO30 DEM were firstly converted to the Arc/Info Grid format and then merged, clipped and lastly projected to the Lambert Azimuthal Equal Area coordinate system. The resulting topographic map of China with a homogeneous 5 km × 5 km grid resolution (Figure E.1 in Appendix E), reveals the following topographical and landform characteristics at the national scale:

- China is a mountainous country, with mountains, hills, and plateaus occupying 65% of the total land area. Land surfaces at lower than 500 m a.s.l. account for only 25.2% of the total, while those above 3,000 m a.s.l. take 25.9%. Seven of the twelve peaks of more than 8,000 m a.s.l. in the world are located in China, including the highest, Mount Qomolangma (8,848 m a.s.l.), standing majestically on the Chinese-Nepalese border;

- Agricultural important regions are recognized as: the Northeast China Plain, the North China Plain, the Loess Plateau, the Sichuan Basin, the Plains of the Middle-and-Lower-Reaches of the Yangtze River, and a batch of smaller plains scattered in the hilly south and on oases in the northwest;

- Two great rivers run through the country: the Yellow River (or “Huanghe”) in the north, and the Yangtze River (or “Changjiang”) to the south. Both rivers originate in the Tibetan Plateau of the far west and flow eastward into the sea. The Loess Plateau, created since two million years ago by the deposition of wind-blown dust and by glacial till, is located in the upper reaches of the Yellow River and provides good agricultural soils. The highly erodible loess not only makes the alluvial plains...
of the middle and lower reaches of the Yellow River in North China fertile, but is also responsible for the distinctive color of the Yellow River and subsequently the Yellow Sea. In fact, most of the agricultural important regions benefit from the irrigation systems driven by these two rivers. Although the drainage basin of the Yangtze River is hillier, it is also warmer and more humid. Due to its high agricultural suitability, the region has been one of the most prosperous in China for the last millennium and is commonly known as the “country of rice and fish”.

2.5 Land cover types and agricultural cropland

The inventory of the agricultural cropland was carried out on the basis of the GLC2000 land cover map for China (Wu et al., 2003a). The original land cover map was projected from the WGS84 to the Lambert Azimuthal Equal Area coordinate system. The spatial resolution of the source map, which was 1 km at the equator, was upscaled to homogeneous 5 km for the target land cover map (Figure E.2 in Appendix E). The land cover was classified with data acquired in year 2000 with the SPOT-VEGETATION instrument (Maisongrande et al., 2004), whose legend was kept compatible with the FAO land cover classification system (LCCS) (FAO, 2000). There were 24 LCCS classes identified in China (Table 2.5). Among them, the “farmland” (grid cell value 21) and the “mosaic cropping” (value 23) classes were extracted into the map of agricultural cropland (Figure E.3 in Appendix E).

The above-mentioned (§2.4.2) spatial patterns of the agricultural important regions are now reaffirmed on both maps of the land cover types and of the agricultural cropland. In the cropland-scarce south-western highlands, grain crops were commonly inter-planted with perennials, shrubs and trees – hence the name “mosaic cropping”. The total area of croplands in China was accounted for 170 million hectares in 2000. The figure was 30% larger than the ground-based census result of the same period (NBSC, 2000). The following reasons were found responsible for this estimation discrepancy:

1. A continuous practice of under-reporting of cropland was observed in China since 1949 (Crook, 1993). Incentives to under-report were obvious (Smil, 1999a): less reported land meant lower state production quota and, more significantly, reduced taxes;

2. Widespread use of non-standard  mu  for area measurements was another source of uncertainty in census data. One  mu  equals 1/15 hectare. But in reality 1  mu  might be 25 - 50% larger than what it should be (Frolking et al., 1999);

3. Methodological differences in cropland survey also contributed to the estimation discrepancy. Field infrastructures, for example, which included irrigation ditches, canals, working paths, boundary ridges, etc., were excluded in ground-based census but included in remote sensing-based investigations (Xiao et al., 2003).

2.6 Climate

2.6.1 Climatic stations

The multi-year monthly means of 7 climatic parameters, including the maximal and the minimal air temperature  \( T_{\text{max}} \) &  \( T_{\text{min}} \), rainfall  \( P \) and frequency  \( P_{freq} \), relative air
humidity \((RH)\), daily sunshine duration \((n)\) and wind velocity \((u)\), were collected from published datasets of 374 stations. Among them, 266 stations were selected from two long-term instrumental climate databases released by the Chinese Academy of Sciences (IAP and CDIAC, 1991). These two databases were originally used as the “baseline” stations in climate change research (referred to as “data source 1” hereafter); 24 stations were bound to the reference soil profiles in China (ISSAS and ISRIC, 1994), which were part of an international collection of reference soil profiles at the ISRIC (“data source 2”); and the rest 84 were chosen from the base stations which defined the climatological normals in China (WMO, 1996) (“data source 3”). These datasets represent a comprehensive, long-term instrumental climatic network in China which constitutes an evenly distributed coverage of the study area (Figure 2.5).

Quality assurance was one of the most important criteria applied in the station screening phase. Extensive reviews on examining the completeness, rationality and accuracy of the data were conducted in the first place. Although these reviews were often requiring intensive programming efforts and usually time-consuming, the quality of the resulting climatic grids confirmed the necessity of doing so. Length of climatic records, upon which the mean values were computed, is guaranteed to be “long-term” on a per station basis so that the datasets were statistically representative. For the 205-station network of the IAP and CDIAC (1991) dataset, for instance, length of the precipitation records varied between 25
and 95 years with a median of 35 or a mean of 40 years. The WMO (1996) climatological normals were obtained on the basis of consecutive records between 1961-1990 (Figure 2.6).

However, the point datasets could hardly produce homogeneous coverages of climatic parameters over the study area due to their multi-source nature. Out of the 374 selected stations, there were merely 22% of them possessing all 7 required parameters, whereas 34% of all the stations were chosen for only 1 parameter (Figure E.4 and its legend in Appendix E). Furthermore, the spatial resolution of the point datasets, which was considered as one of the scale issues (Goodchild, 1999), varied from parameter to parameter, ranging from fine, for precipitation and relative air humidity, to coarse, for daily sunshine duration and wind velocity, with air temperature and frequency of precipitation in between the two (Figure 2.7). Therefore, continuous and homogeneous surfaces of climatic parameters needed to be derived from the point datasets by using an appropriate interpolation technique, such as ordinary kriging (ESRI, 2001).

The list of the selected stations, their WMO-verified meta data, and the completeness of the required climatic parameters are documented in Appendix B on a per station basis.

### 2.6.2 Spatial interpolation techniques

It is common in spatial analysis to interpolate a spatially continuous variable from point samples. Suppose the value of the precipitation, for instance, at an unmeasured location

---

**Data Source**

- IAP & CDIAC (1991) : 266 stations
- ISSAS & ISRIC (1994) : 24 stations
- WMO (1996) : 84 stations

---

Figure 2.5: *Spatial distribution of the point dataset of selected climatic stations*
Figure 2.6: Length of station records (years) of the climatic datasets: (a) 205-station network of IAP and CDIAC (1991); (b) ISSAS and ISRIC (1994); (c) 60-station network of IAP and CDIAC (1991); and (d) WMO (1996)

Figure 2.7: Spatial distribution of the climatic stations on a per parameter basis: (a) maximal and minimal air temperatures; (b) precipitation; (c) frequency of precipitation; (d) relative air humidity; (e) daily sunshine duration; and (f) wind velocity
2.6 Climate

$S_0$ is to be predicted from values of its measured neighbors in the study area (Figure 2.8):

$$
\tilde{Z}(S_0) = \sum_{i=1}^{n} \{ \lambda_i \cdot Z(S_i) \} \tag{2.2}
$$

where $S$ stands for a location, $i$ is the serial number of a location, $n$ is the total number of locations with measured values, $Z(S_i)$ is the value at location $S_i$, $\tilde{Z}(S_0)$ is the predicted value at unmeasured location $S_0$ and $\lambda_i$ is the weighting factor assigned to location $S_i$.

Figure 2.8: Spatial layout of locations over a virtual terrain with both measured and unmeasured values. After ESRI (2001)

Depending on the algorithms and/or models applied to evaluate the weighting factor $\lambda_i$, the interpolation techniques are divided into two groups: deterministic and geostatistical. All methods rely on the similarity of nearby sample points to create a prediction surface. Deterministic interpolation techniques use mathematical functions for interpolation, while geostatistics rely on both statistical and mathematical methods, which can be used to create surfaces and assess the uncertainties of the prediction.

**Deterministic interpolation**

Generally speaking, locations that are closer together tend to be more alike than those that are further apart. This is a fundamental geographic principle and will be termed as “spatial autocorrelation” hereafter (Tobler, 1970). Thus in one of the widely used deterministic approaches, the inverse distance weighting (IDW), for example, the weight of a value decreases as the distance increases from the prediction location $S_0$:

$$
\lambda_i = a \cdot (d_{i,0})^{-b} \tag{2.3}
$$
where $d_{i,0}$ is the distance between the prediction location $S_0$ and the sample location $S_i$, and $a$ and $b$ are coefficients. Here, in IDW, $b$ would take the value of 1 and $a$ would be defined as $a = \left( \sum_{i=1}^{n} (d_{i,0})^{-b} \right)^{-1}$.

It is worth mentioning that the entire dataset of the study area is not always used at its entirety for interpolation. If $\tilde{Z}(S_0)$ is predicted on basis of all the 11 measured locations, i.e., $i = 1...11$, then the approach represented by equations 2.2 and 2.3 are categorized as a *global* interpolator. To the contrary, if only the neighboring locations, for instance, within a certain distance from $S_0$, i.e., $i = 1...5$, are used, then it is said to be a *local* interpolator. An interpolator is said to be *exact* if the predicted value is identical to the measured one at a sampled location (i.e., location 1, 2, ..., or 11). On the other hand, if the predicted value is different from the measured one, the interpolator is then said to be *inexact*. Some commonly used deterministic interpolators are listed and categorized in Figure 2.9(a).

(a)

![Interpolators Diagram](image)

(b)

![Interpolators Diagram](image)

Figure 2.9: *Some commonly used deterministic (a) and geostatistical (b) interpolators*
Deterministic techniques are indeed easy to understand and apply, and often produce satisfactory results. However, the algorithms engaged are somewhat simplistic and can be improved upon by taking a more statistically sound approach to modeling the weighting factor, $\lambda_i$.

**Geostatistical interpolation**

As the name implies, a geostatistical interpolator creates surfaces incorporating the statistical properties of the measured data. Many methods are associated with geostatistics and they are all in the kriging family (Figure 2.9(b)), which consists mostly of exact interpolators.

Kriging was developed in the 1960s by the French mathematician Georges Matheron in modeling gold deposits in a rock from a few random core samples. Kriging is similar to IDW in that it weights the surrounding measured values to derive a prediction for a target location. However, the weights are based not only on the distance between the measured and the prediction locations but also on the spatial autocorrelation among the measured points. Spatial autocorrelation is treated as a function of distance in kriging. To quantify its effects on prediction, an empirical semivariogram model has to be fitted. The weighting factor, $\lambda_i$, can only be assessed in a following step.

A semivariogram is simply half of the value difference squared (i.e., the semivariance), between a pair of locations $i$ and $j$, plotted on the $y$-axis and the distance between them on the $x$-axis:

$$\gamma_{ij} = \frac{1}{2} \cdot |Z(S_i) - Z(S_j)|^2$$

(2.4)

where $\gamma_{ij}$ is the semivariance between locations $i$ and $j$.

In theory, the value at a location is composed of a deterministic trend ($\mu$) and a random, autocorrelated error term ($\epsilon$),

$$Z(S) = \mu(S) + \epsilon(S)$$

(2.5)

where $\mu(S)$ and $\epsilon(S)$ are the trend and error terms in location $S$ respectively.

Variations of this equation form the basis of different kriging interpolators. For instance, if the trend term, $\mu(S)$, is simplified as a location irrelevant, unknown constant, then the new model

$$Z(S) = \mu + \epsilon(S)$$

(2.6)

forms the theoretical basis of ordinary kriging.

The autocorrelation between locations $i$ and $j$, i.e., $\Delta \epsilon_{ij} = \epsilon(S_i) - \epsilon(S_j)$, is assumed only to be dependent on the distance between the two – instead of on the actual – locations ($S_i$ and $S_j$):

$$\Delta \epsilon_{ij} \sim d_{ij}$$

(2.7)

and

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

(2.8)

where $x_i$ and $y_i$ are the x-y coordinates of location $S_i$, $x_j$ and $y_j$ are the x-y coordinates of $S_j$, and $d_{ij}$ is simply the Euclidean distance between locations $S_i$ and $S_j$.

The solution to the minimization of the statistical expectation of the squared difference between the true value, $Z(S_0)$, and the predictor, $\sum \lambda_i \cdot Z(S_i)$,

$$\left(Z(S_0) - \sum_{i=1}^{n} \lambda_i \cdot Z(S_i)\right)^2$$

(2.9)
constrained by the unbiasedness of the predictions, i.e., to make the sum of the prediction errors equal to zero, the sum of the weight, \( \lambda_i \), must equal one,

\[
\sum_{i=1}^{n} \lambda_i = 1 \quad (2.10)
\]
gives the following kriging equation:

\[
\begin{pmatrix}
\gamma_{11} & \cdots & \gamma_{1n} & 1 \\
\vdots & \ddots & \vdots & \vdots \\
\gamma_{n1} & \cdots & \gamma_{nn} & 1 \\
1 & \cdots & 1 & 0
\end{pmatrix}
\begin{pmatrix}
\lambda_1 \\
\vdots \\
\lambda_n \\
m
\end{pmatrix}
= 
\begin{pmatrix}
\gamma_{10} \\
\vdots \\
\gamma_{n0} \\
1
\end{pmatrix} \quad (2.11)
\]

where the matrix \( \Gamma \) contains the modeled semivariance values between all pairs of sample locations, with entry \( \gamma_{ij} \) denoting the modeled semivariance values based on the distance between the two samples identified as the \( i^{th} \) and \( j^{th} \) locations; the vector \( g \) contains the modeled semivariance values between each measured location and the prediction location, with entry \( \gamma_{i0} \) denoting the modeled semivariance values between the \( i^{th} \) sample and the prediction location.

In practice, a six-step approach is used in geostatistical interpolation in ArcGIS:

1. Analyze the measured dataset in trying to identify the outlier values and/or the global trend. If identified, the outliers are then rejected and a possible trend is subtracted from the data surface (“detrending”). The remainder between the measured values and the trend, i.e., the residuals, are to be modeled in the following steps. If no global trend is identified, the measured dataset itself is to be modeled;

2. Calculate the empirical semivariance, and plot the semivariogram cloud. The empirical semivariance, \( \Gamma_{ij}(h) \), is calculated as the average semivariance, \( \gamma_{ij} \), for all pairs of locations separated by distance \( h \)

\[
\Gamma_{ij}(h) = \frac{1}{n} \sum_{k=1}^{n} \gamma_{ij} = \frac{1}{2n} \sum_{k=1}^{n} [Z(S_i) - Z(S_j)]^2 \quad (2.12)
\]

where \( h = d_{ij} \) (Equation 2.8), \( n \) is the total number of location pairs which are separated by \( h \). If plotted on an x-y plane, \( \Gamma_{ij} \) and \( h \) will be represented by one dot. Consequently, the semivariance of all distance intervals will form a semivariogram cloud;

3. Fit a semivariance model. A predefined semivariance model needs to be fitted through the points in the empirical semivariogram cloud graph to quantify the spatial autocorrelation in the data;

4. Create the matrix \( \Gamma \) and the vector \( g \) as defined in Equation 2.11. The kriging weights, i.e., vector \( \lambda \), is known for each interpolation point by solving the equation;

5. Predict unknown values for each interpolation point using the predictor found in Equation 2.2;
6. Add the global trend back to the predicted dataset. The sum represents the resulting surface geostatistically interpolated from the measured dataset.

### 2.6.3 Grid generation with ordinary kriging

The point datasets of climatic parameters were interpolated using the ordinary kriging functionalities of the Geostatistical Analyst extension of the ArcGIS software package (ESRI, 2001). Semivariogram models were fitted for the monthly average values of 7 climatic parameters, and in total 84 continuous surfaces were generated. To avoid redundancy, only the mean maximal air temperature in January is presented here as a case study to demonstrate the use of ordinary kriging in an elevation-adjusted air temperature interpolation and surface derivation application.

#### Temperature-elevation relationship

According to the International Standard Atmosphere (ISA) model (ICAO, 1993), the air temperature changes at different rates against the elevation in different layers of the atmosphere (Table 2.6). The temperature not only decreases with elevation (in troposphere and mesosphere) but increases (in stratosphere) as well. However, the most rapid change happens in the bottom layer of the atmosphere. The temperature drops at a lapse rate of 6.5°C km$^{-1}$ upwards until 11 km a.s.l. (Figure 2.10).

<table>
<thead>
<tr>
<th>Layer</th>
<th>Level name</th>
<th>$H^a$</th>
<th>$R^b$</th>
<th>$T^c$</th>
<th>$P^d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Troposphere</td>
<td>0.0</td>
<td>-6.5</td>
<td>+15.0</td>
<td>101,325.00</td>
</tr>
<tr>
<td>1</td>
<td>Tropopause</td>
<td>11.0</td>
<td>+0.0</td>
<td>-56.5</td>
<td>22,632.00</td>
</tr>
<tr>
<td>2</td>
<td>Stratosphere</td>
<td>20.0</td>
<td>+1.0</td>
<td>-56.5</td>
<td>5,474.90</td>
</tr>
<tr>
<td>3</td>
<td>Stratosphere</td>
<td>32.0</td>
<td>+2.8</td>
<td>-44.5</td>
<td>868.02</td>
</tr>
<tr>
<td>4</td>
<td>Stratopause</td>
<td>47.0</td>
<td>+0.0</td>
<td>-2.5</td>
<td>110.91</td>
</tr>
<tr>
<td>5</td>
<td>Mesosphere</td>
<td>51.0</td>
<td>-2.8</td>
<td>-2.5</td>
<td>66.94</td>
</tr>
<tr>
<td>6</td>
<td>Mesosphere</td>
<td>71.0</td>
<td>-2.0</td>
<td>-58.5</td>
<td>3.96</td>
</tr>
<tr>
<td>7</td>
<td>Mesopause</td>
<td>84.9</td>
<td>-</td>
<td>-86.2</td>
<td>0.37</td>
</tr>
</tbody>
</table>

$^a$elevation (km); $^b$lapse rate (°C km$^{-1}$); $^c$base temperature (°C); $^d$base atmospheric pressure (Pa)

The point dataset of the monthly mean maximal air temperature in January, $T_{\text{max1}}$, was first projected to the horizontal plane at the sea level (i.e., the x-y plane in Figure 2.12), and data values of $T_{\text{max1}}$ were transformed from terrain elevation $H$ to the sea level ($T'_{\text{max1}}$). An ordinary kriging interpolator was then applied to the transformed dataset $T'_{\text{max1}}$. The elevation effect was lastly put back to the temperature surface resulted from the interpolation before proceeding further.

$$
T'_{\text{max1}} = T_{\text{max1}} + 0.0065 \cdot H \quad (2.13)
$$

$$
T_{\text{max1}} = T'_{\text{max1}} - 0.0065 \cdot H \quad (2.14)
$$

where $H$ is the elevation (m) of a climatic station, taken from the DEM (Figure E.1 in Appendix E).
Data analysis

The ordinary kriging method gives the best results if data is free of exceptional values, or outliers, and normally distributed. The outlier values should be excluded from the dataset beforehand and a transforming method, e.g., the logarithmic transformation for a skewed dataset, is needed to make a dataset normal. The values of maximal temperature in January were statistically analyzed and plotted (R Development Core Team, 2007) to identify possible outliers and test its statistical distribution (Figure 2.11).

Outliers are considered as extreme values which are not clustered with the rest of the dataset and lie more than 1.5 times of the inter-quantile range (IQR) from the 1st or the 3rd quantile (Tukey, 1977)

\[ P_o < Q_1 - 1.5 \cdot IQR \]
\[ or \quad P_o > Q_3 + 1.5 \cdot IQR \]

(2.15)

where \( P_o \) represents an outlier point; \( Q_1 \) and \( Q_3 \) are the 1st and 3rd quantiles respectively; \( IQR = |Q_1 - Q_3| \).

In the Box-and-Whisker plot in Figure 2.11(a), the IQR was represented by the width of the box, and 1.5 times of the IQR from both sides of the box (\( Q_1 \) and \( Q_3 \)) were represented by two vertical bars in both directions. Absence of dots drawn beyond these two vertical bars indicated that no outliers were presented in both the original or the elevation-adjusted datasets of \( T_{\text{max1}} \). It was not strange to observe that the elevation-adjusted temperature values (averaged at 8.58°C) were higher than the original ones (averaged at 3.70°C), due to the lapse rate defined in equation 2.13.

The elevation-adjusted maximal temperature in January dataset, \( T'_{\text{max1}} \), was statistically tested using the Shapiro-Wilk method (R Development Core Team, 2007) and a normal distribution was confirmed (\( W = 0.9858 \) with a \( p \)-value of 0.0667). The same conclusion was suggested by either the histogram and the superposed probability distribution.
2.6 Climate

Figure 2.11: Pre-kriging data examination with the Box-and-Whisker plot (a), the histogram (b) and the normal quantile-quantile plot (c)

curve (2.11(b)) or the conformity between the distributions of $T'_{\text{max1}}$ and a theoretically normally distributed dataset (2.11c).

Trend analysis

A trend is the deterministic component of a surface which should be separated from the autocorrelated component (Equation 2.5) and mathematically represented outside the scope of geostatistical modeling. For example, a valley can be modeled by a “U” shape which may in turn be represented by a second-order polynomial function. Once the global trend is removed, the geostatistical analysis will be performed on the short range variations, or the residuals, of the surface. The trend will need to be added back before an accurate and meaningful surface can be finally created. This last step is automatically and explicitly done by the ArcGIS Geostatistical Analyst software (ESRI, 2001) at the moment of surface generation.

To identify the possible global trend in $T'_{\text{max1}}$, the dataset was plotted in a 3-dimensional space with the x-, y- and z-axis representing the easting, the northing and the elevation above sea level respectively. The data points in the xyz space were then projected to three 2-dimensional planes including the horizontal plane at the sea level (i.e., the x-y plane), the east-west plane (i.e., the x-z plane) and the north-south plane (i.e., the y-z plane). While the projected points on the x-y plane represented the spatial configuration of the climatic stations in the longitude-latitude grid, the near-“U”-shaped curve fitted to the projected points on the east-west plane suggested the presence of a second-order trend in the northeast-southwest direction (“northeast-southwest”, instead of “east-west”, since the x-axis had been rotated 43.7° counterclockwise and now pointed to the northeast). To the contrary, the straight line fitted in the north-south plane indicated the absence of any global trend in the northwest-southeast direction (Figure 2.12).

The results of the trend analysis could be justified by the climatic-topographic realities in China. A high-low temperature pattern is indeed observed at the national scale (Figure E.5(b)), with high values in the southeast and lower ones in the northwest. The latitudinal differences are partly a reason; the elevational differences are another yet much stronger one. Since the elevational effect on temperature differences has been removed from the $T'_{\text{max1}}$ dataset, it is not odd to observe a stronger trend in the northeast-southwest direction, with lower values in the northeast and higher ones in the southwest.
Semivariogram analysis

The semivariogram cloud of the detrended $T_{\max}^1$ was plotted in a Cartesian plane with the average semivariance values at the y-axis and the distance between the station pairs at the x-axis (Figure 2.13). Circular bands of 150 km in width, i.e., the lag size, were used to group, or bin, the semivariogram points in an attempt to control the complexity of the generated semivariogram cloud. A spherical semivariogram model was fitted to the semivariogram cloud with the following parameters:

- **Range**: 1,426.5 km. Climate is a large-scale phenomenon. Stations that are close to each other are more autocorrelated (i.e., a smaller semivariance value) than those that are further apart. The fitted semivariogram curve rises with increasing distances; it then levels out when the range is reached. The flattening out of the semivariogram indicates that there is little autocorrelation beyond the range;

- **Sill**: 14.69, which is the value of the semivariogram model after the range is reached. Beyond the range, the stations are not autocorrelated, or in other words, the dissimilarity between stations becomes constant with increasing distance;

- **Nugget**: 1.716. The nugget represents measurement error.

The following semivariogram model was fitted:

$$14.69 \times Spherical (1426500, 1239800, 13.7) + 1.716 \times Nugget$$  \hspace{1cm} (2.16)

The semivariogram cloud can also be used in outlier identification. Outliers would appear as high semivariogram values between relatively close stations. The absence of such exceptional points in the neighborhood of the origin of the semivariogram cloud (Figure 2.13) confirms the absence of outliers which was discussed earlier in § 2.6.3.
Cross validation

The goodness of a fitted empirical semivariogram model (for example, Equation 2.16) was cross validated using the leave-one-out cross-validation (LOOCV) method (Geisser, 1993). Cross-validation is a model evaluation method which partitions a $N$-point dataset into $k$ subsets ($k \leq N$), with each subset (the validation set) used in turn to validate a model fitted to the remaining $k - 1$ subsets (the training set). The LOOCV method is no more than a special case in cross validation, with $k = N$. Applied in here, the LOOCV chose one station each time from the $T'_{\text{max}1}$ dataset, and used the remaining stations to train a model, with which a prediction was made for the chosen station. This process was repeated until each station was used once as the validation set. The resulting predicted temperature values were compared with the $T'_{\text{max}1}$ values, as shown in Figure 2.14.

The statistical parameters of both the measured and the predicted $T'_{\text{max}1}$ are not only presented in Figure 2.14(c), but also summarized in Table 2.7. The $t$-test ($t$-statistic = 0.007, $p$-value = 0.994, degree of freedom = 356) (R Development Core Team, 2007) concluded that the measured and predicted temperature values were statistically representative to each other.

<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>$Q_1$</th>
<th>Median</th>
<th>Mean</th>
<th>$Q_3$</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured</td>
<td>-20.98</td>
<td>0.36</td>
<td>8.80</td>
<td>8.58</td>
<td>17.79</td>
<td>35.00</td>
</tr>
<tr>
<td>Predicted</td>
<td>-20.71</td>
<td>-0.63</td>
<td>8.83</td>
<td>8.57</td>
<td>17.68</td>
<td>36.45</td>
</tr>
</tbody>
</table>
Figure 2.14: Cross validation of the empirical semivariogram model: (a) predicted versus measured temperatures against the 1:1 line; (b) prediction error for each measured value; and (c) comparison of statistical distributions of measured and predicted temperatures

Prediction error

One of the advantages of using a geostatistical approach over a deterministic one is that the prediction error can be statistically computed. The spatial distribution of the standard error in predicting $T'_{\text{max1}}$ values is shown in Figure 2.15. The errors were well explained by the density, as well as the topology, of the climatic stations on which the interpolation was based. The least errors were observed in the immediate vicinities of the climatic stations (or clusters of them), while the most significant prediction errors were found in regions like the Tibetan Plateau where the climatic stations were sparsely distributed. The low prediction errors in the coastal regions were partly benefitted from the inclusion of foreign
climatic stations in the interpolation. Nevertheless, the fitted empirical semivariogram model was accepted due to the fact that the croplands (Figure E.3) are overlapping the areas with low prediction errors.

Grid generation

A grid surface of $T_{\text{max}1}'$, at a spatial resolution of 5 km $\times$ 5 km, was generated using the aforementioned empirical semivariogram model (Equation 2.16). It was clear that the temperature values of this grid were virtually measured at the sea level. They had to be transformed back to values measured at elevations of the terrain (Figure E.1 in Appendix E), using Equation 2.14. The transformation can easily be accomplished with the following Arc/Info GRID command:

$$\text{Grid: tmax1}_5\text{k} = \text{tmax1prime}_5\text{k} - 0.0065 \times \text{dem5k}$$

The resulting surfaces of both $T_{\text{max}1}'$ and $T_{\text{max}1}$ are shown in Figure E.5 (Appendix E), side by side for comparison. Figure E.5(b) is obviously more promising in representing the climatic conditions in China. At the macro scale, temperatures are generally higher in the southeast than in the northwest. Minimum values are found either in high latitudes.
(for example, the northwest) or in high altitudes (for example, the Tibetan Plateau). At the micro scale, topographical patterns are clearly reflected in Figure E.5(b) other than in Figure E.5(a), especially in areas with high slope gradients (for example, the tiny southern Himalayan region bordering eastern India, the south-western highlands, the section of the Yangtze River at the Three Gorges, the Central Mountains and the surrounding coastal flats in Taiwan, and so forth).

2.6.4 Climatic grids

Similarly, empirical semivariogram models were fitted in situ per climatic parameter, per month and in total 84 grid surfaces (including $T_{max}$) were generated. The monthly mean maximal air temperature and rainfall classes, January through December, are shown in Figures E.6 and E.7 respectively. The rest five climatic parameters, including the minimal air temperature, precipitation frequency, relative air humidity, daily sunshine duration and wind velocity, are shown in Figures E.8 through E.12 respectively, given in Appendix E.

2.6.5 Climate classification based on aridity index

The climate had been classified in an attempt to differentiate the dry regions from the wet ones, on the basis of the aridity index (UNEP, 1997). The aridity index (AI) is defined as

$$AI = \frac{P}{ET_o}$$

(2.17)

where $P$ is the mean annual precipitation (mm) and $ET_o$, the mean annual reference evapotranspiration (mm). Both $P$ and $ET_o$ must be expressed in the same unit, but not necessarily in mm. The value of AI is dimensionless.

The climate was classified into 2 orders and 5 classes. The arid order was divided into 3 classes: hyperarid, arid and semi-arid, while the humid order, into 2: dry subhumid and humid (Table 2.8). Maps of the AI values, climatic classes and orders are shown in Figure E.13 (Appendix E).

<table>
<thead>
<tr>
<th>Order</th>
<th>Class</th>
<th>AI value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arid</td>
<td>Hyperarid</td>
<td>AI &lt; 0.05</td>
</tr>
<tr>
<td></td>
<td>Arid</td>
<td>0.05 ≤ AI &lt; 0.20</td>
</tr>
<tr>
<td></td>
<td>Semi-arid</td>
<td>0.20 ≤ AI &lt; 0.50</td>
</tr>
<tr>
<td>Humid</td>
<td>Dry subhumid</td>
<td>0.50 ≤ AI &lt; 0.65</td>
</tr>
<tr>
<td></td>
<td>Humid</td>
<td>0.65 ≤ AI</td>
</tr>
</tbody>
</table>

2.7 Soil

The soil dataset was extracted and converted from the ISRIC-WISE database of 5′ × 5′ (Batjes, 1997, 2002b), based on an inventory of the FAO soil units in China (FAO, 1999) correlated to the Revised Legend (FAO-UNESCO-ISRIC, 1990). Although WISE was originally developed for the purpose of inventory of soil green house gas (GHG) emissions, datasets derived from WISE were considered appropriate to be applied in fields
ranging from upscaling/downscaling of GHG emissions (Denier van der Gon et al., 2000; Bouwman et al., 2002) to environmental change modeling (Ganzeveld et al., 1998), from land evaluation (Batjes, 1997) to agro-ecological zoning (AEZ) (Fisher et al., 2001) applications, etc.

2.7 Soil mapping unit

The FAO-UNESCO (2003) Digital Soil Map of the World (DSMW) is comprised of mapping units (MUs), which bound areas of contained soil units or associations of soil units together. When a MU is not homogeneous, it is composed of a dominant soil unit, associated soil unit(s) and included soil unit(s). The area coverage, in a given MU, for each of these categories is as follows:

- the dominant soil unit occupies the largest area of the MU;
- an associated soil unit occupies more than 20% of the enclosing MU but less than the area of the dominant soil unit;
- an included soil unit occupies less than 20% of the area of the enclosing MU.

The proportion of the areas of the dominant and component soils, if not explicitly given, can be derived from the composition rules (FAO-UNESCO, 2003) found in Table 2.9.

<table>
<thead>
<tr>
<th>Dominant soil % area</th>
<th>Associated soil(s) no.† % area</th>
<th>Included soils no.‡ % area</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>90</td>
<td>0 0</td>
<td>1 10</td>
</tr>
<tr>
<td>80</td>
<td>0 0</td>
<td>2 10 + 10</td>
</tr>
<tr>
<td>70</td>
<td>0 0</td>
<td>3 10 + 10 + 10</td>
</tr>
<tr>
<td>70</td>
<td>1 30</td>
<td>0 0</td>
</tr>
<tr>
<td>60</td>
<td>1 30</td>
<td>1 10</td>
</tr>
<tr>
<td>60</td>
<td>2 20 + 20</td>
<td>0 0</td>
</tr>
<tr>
<td>50</td>
<td>2 20 + 20</td>
<td>1 10</td>
</tr>
<tr>
<td>50</td>
<td>1 30</td>
<td>2 10 + 10</td>
</tr>
<tr>
<td>50</td>
<td>1 30</td>
<td>4 5 + 5 + 5 + 5</td>
</tr>
<tr>
<td>40</td>
<td>2 20 + 20</td>
<td>4 5 + 5 + 5 + 5</td>
</tr>
<tr>
<td>40</td>
<td>1 30</td>
<td>1 10</td>
</tr>
<tr>
<td>40</td>
<td>2 20 + 20</td>
<td>2 10 + 10</td>
</tr>
<tr>
<td>30</td>
<td>3 20 + 20 + 20</td>
<td>1 10</td>
</tr>
<tr>
<td>30</td>
<td>3 20 + 20 + 20</td>
<td>3 10 + 10 + 10</td>
</tr>
<tr>
<td>30</td>
<td>3 20 + 20 + 20</td>
<td>2 5 + 5</td>
</tr>
<tr>
<td>25</td>
<td>3 20 + 20 + 20</td>
<td>3 5 + 5 + 5</td>
</tr>
<tr>
<td>24</td>
<td>3 20 + 20 + 20</td>
<td>4 4 + 4 + 4 + 4 + 4</td>
</tr>
</tbody>
</table>

† Number of associated soils; ‡ Number of included soils

The soil associations were indicated on the original map (FAO-UNESCO, 1974) by the symbol of the dominant soil unit, followed by a number which referred to the descriptive
legend on the back of the map, where the full composition of the association was given. The symbol “Bd35”, for instance, represents the soil association of 1 dominant, 2 associated and 3 included soil units (Table 2.10).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Component</th>
<th>Key</th>
<th>Soil unit</th>
<th>% Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bd</td>
<td>Dominant</td>
<td>Bd</td>
<td>Dystric Cambisols</td>
<td>30</td>
</tr>
<tr>
<td>Bd35</td>
<td>Associated</td>
<td>Ao</td>
<td>Orthic Acrisols</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Associated</td>
<td>Jd</td>
<td>Dystric Fluvisols</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Included</td>
<td>Dd</td>
<td>Dystric Podzoluvisols</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Included</td>
<td>Ge</td>
<td>Eutric Gleysols</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Included</td>
<td>Po</td>
<td>Orthic Podzols</td>
<td>10</td>
</tr>
</tbody>
</table>

In the DSMW, the composition of soil associations were given in the expansion file which can be joined to the attribute table of the DSMW using the SNUM field. The SNUM is a global unique sequential number assigned to each MU of the DSMW, with value ranges of 4000-7100 and 7500-7512 for China. Each record inside the expansion file describes one MU, its components (dominant, associated and included soil units), the percentage occurrence of each soil unit within the MU, and so on. One MU may contain up to 8 component soil units. For example, the MUs of Cambisols in China, as identified on a $0.5^\circ \times 0.5^\circ$ grid version of the DSMW, and their compositions are given in Figure 2.16 and Table 2.11 respectively.

### 2.7.2 FAO soil units in China

The soil inventory was based on the vector coverage of the FAO soil units in China, extracted from the updated soil and physiographic database for North and Central Eurasia at 1:5 M scale (FAO, 1999). The coverage was cleaned to ensure the topographical integrity. Its coordinate system of Lambert Azimuthal Equal Area was kept intact. An overview of the FAO soil groupings in China is presented in Figure E.14 (Appendix E); the more detailed soil units in the middle and lower reaches of the Yangtze River (Zone 5, §2.2.2), for instance, is given in Figure E.15 (Appendix E), both correlated to the Revised Legend (FAO-UNESCO-ISRIC, 1990).

### 2.7.3 DSMW rasterization

The vector coverage of the soil units was rasterized to a $5 \text{ km} \times 5 \text{ km}$ grid by assigning the soil mapping unit number (SNUM) found at the center of each $5 \text{ km} \times 5 \text{ km}$ cell to that cell. Studies (FAO-UNESCO, 2003) revealed that this is a more accurate representation of the grid cell than the dominant MU approach, in which the SNUM of the biggest MU in a cell is assigned to the cell, since the latter is biased against the soil units that occur in smaller MUs. The rasterization procedure was as the following (Figure 2.17):

(a) The vector coverage of the soil units was cleaned; and the consistency of the SNUM field in the attribute database was checked using the SNUM value ranges for China (4000-7100 and 7500-7512), values at boundary conditions, for instance, the water bodies (6997), the glacial (6998) and the NODATA areas (0), as criteria, so that

(b) The thematic layer of the MUs was correctly representing the underlying soil units;
Figure 2.16: Soil mapping units of Cambisols in China, identified on a 0.5\degree\times\ 0.5\degree\ grid version of the Digital Soil Map of the World (DSMW)

(c) A blank 5 km \times 5 km grid was overlaid with the vector map of the MUs, and the SNUM of the mapping unit found at the center of each cell was assigned to that cell;

(d) A new thematic 5 km \times 5 km grid containing SNUMs was generated. The soil unit complexity within each cell is now represented by the expansion file which can be joined to the grid on the SNUM field.

2.7.4 Virtual soil profile

Each grid cell of 5 km \times 5 km was viewed as a virtual soil profile of 5 equal sections of 20 cm over 1 meter soil depth. The 80-100 cm section was extended to the rooting depth of a concerned crop, should the latter be deeper than 1 meter at any stage within its crop cycle. Horizon-specific soil parameters were converted from the ISRIC-WISE database of 5\degree \times 5\degree (Batjes, 1997, 2002b). A 5\degree \times 5\degree grid cell is roughly 9.2 km \times 9.2 km at the equator. The reasons why a finer 5 km \times 5 km grid was selected to represent the soil parameters were due to the following reasons:

- Compromise between representability and data redundancy. To represent the soil parameters in 5 km \times 5 km grids indeed created some data redundancy, which is
Table 2.11: Composition of the mapping units of Cambisols in China

<table>
<thead>
<tr>
<th>SNUM</th>
<th>Symbol</th>
<th>Soil¹</th>
<th>%</th>
<th>Soil²</th>
<th>%</th>
<th>Soil³</th>
<th>%</th>
<th>Soil⁴</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3663</td>
<td>Bd34</td>
<td>Bd</td>
<td>24</td>
<td>Ao</td>
<td>20</td>
<td>Bd</td>
<td>20</td>
<td>I</td>
<td>20</td>
</tr>
<tr>
<td>3664</td>
<td>Bd35</td>
<td>Bd</td>
<td>30</td>
<td>Ao</td>
<td>20</td>
<td>Jd</td>
<td>20</td>
<td>Dd</td>
<td>10</td>
</tr>
<tr>
<td>3679</td>
<td>Be78</td>
<td>Be</td>
<td>40</td>
<td>Hh</td>
<td>20</td>
<td>I</td>
<td>20</td>
<td>Bh</td>
<td>10</td>
</tr>
<tr>
<td>3683</td>
<td>Be82</td>
<td>Be</td>
<td>50</td>
<td>Lo</td>
<td>30</td>
<td>Ao</td>
<td>10</td>
<td>Lc</td>
<td>10</td>
</tr>
<tr>
<td>4140</td>
<td>Bd3</td>
<td>Bd</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4286</td>
<td>Be28</td>
<td>Be</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4287</td>
<td>Be49</td>
<td>Be</td>
<td>40</td>
<td>Bl</td>
<td>20</td>
<td>Lc</td>
<td>20</td>
<td>I</td>
<td>10</td>
</tr>
<tr>
<td>4288</td>
<td>Be50</td>
<td>Be</td>
<td>60</td>
<td>Lc</td>
<td>20</td>
<td>Re</td>
<td>20</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4291</td>
<td>Bd37</td>
<td>Bd</td>
<td>60</td>
<td>Be</td>
<td>20</td>
<td>Bh</td>
<td>20</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4298</td>
<td>Be87</td>
<td>Be</td>
<td>80</td>
<td>G‡</td>
<td>10</td>
<td>J</td>
<td>10</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4302</td>
<td>Bk36</td>
<td>Bk</td>
<td>50</td>
<td>Lo</td>
<td>20</td>
<td>Xk</td>
<td>20</td>
<td>Rc</td>
<td>10</td>
</tr>
<tr>
<td>4303</td>
<td>Bk42</td>
<td>Bk</td>
<td>70</td>
<td>Xh</td>
<td>30</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4305</td>
<td>Bk44</td>
<td>Bk</td>
<td>50</td>
<td>Xk</td>
<td>30</td>
<td>Gc</td>
<td>10</td>
<td>Jc</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNUM</th>
<th>Symbol</th>
<th>Soil₅</th>
<th>%</th>
<th>Soil₆</th>
<th>%</th>
<th>Soil₇</th>
<th>%</th>
<th>Soil₈</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3663</td>
<td>Bd34</td>
<td>Bc</td>
<td>4</td>
<td>Bh</td>
<td>4</td>
<td>G</td>
<td>4</td>
<td>O</td>
<td>4</td>
</tr>
<tr>
<td>3664</td>
<td>Bd35</td>
<td>G</td>
<td>10</td>
<td>P</td>
<td>10</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3679</td>
<td>Be78</td>
<td>Lo</td>
<td>10</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3683</td>
<td>Be82</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4140</td>
<td>Bd3</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4286</td>
<td>Be28</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4287</td>
<td>Be49</td>
<td>Re</td>
<td>10</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4288</td>
<td>Be50</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4291</td>
<td>Bd37</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4298</td>
<td>Be87</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4302</td>
<td>Bk36</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4303</td>
<td>Bk42</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>4305</td>
<td>Bk44</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

* Soil¹: dominant soil; Soil₂−₈: associated and included soils, see Table 2.9; † Soil unit, see Appendix C. ‡ See Table 2.10 remarks

2.7 Soil

a minor issue of disk storage space, but gained compatibility with other datasets including the climate, land use, etc. To the contrary, to avoid data redundancy in soil parameters would have forced all the other datasets to be represented in a coarser resolution which is a major issue of representability and accuracy.

- Ease of data manipulation and computation. It was practical and pragmatic to choose 5 km × 5 km, instead of 9.2 km × 9.2 km, since the latter was more difficult in spatial conversions and subsequent grid generation operations.

The WISE datasets are intrinsically profile-based. A locally representative soil profile was linked, in the very beginning by native experts, with each FAO soil unit within the scope of a 0.5°×0.5° cell of the grid version of the DSMW. Priorities had also been given to local experts on estimating the missing soil parameter values. Median of all the soil units of the same grouping was used as the last resort in filling parameter gaps, should the local knowledge be unavailable in this case (Batjes, 1997, 2002b).
2.7 Soil

Figure 2.17: Vector to raster (V2R) conversion of the soil map with the following thematic layers superposed from bottom to top: (a) soil unit (FAO, 1999); (b) mapping unit (SNUM); (c) central-MU V2R scheme and (d) resulting grid marked with representative SNUMs

2.7.5 Case study: SOC density and storage in cropland of China

Except those aforementioned applications, the WISE dataset of derived soil parameters has also been applied particularly in soil organic carbon (SOC) storage estimations at the global (Batjes, 1996) and continental (Europe) (Batjes, 2002a) scales. The WISE-based SOC inventories were recently conducted more at the national scale for a batch of countries including Kenya (Batjes, 2004), Brazil (Batjes, 2005), Jordan (Batjes, 2006), etc.

SOC is among a range of soil characteristics which are essential for crop growth. The importance of SOC comes from the fact that SOC not only improves soil structural stability but is also involved in adsorption of important crop nutrients. It can significantly influence soil water holding capacity which provides a water and nutrient reservoir for crops to take up. The SOC content (w%) (Ye et al., 2008) can be easily converted to the SOC density (SOCD, Mg ha\(^{-1}\)), which is frequently used for soil carbon accounting purposes (Smith, 1999; Penman et al., 2000; Marland et al., 2001), using the following equation:

\[
SOCD = 10^{-3} \times d \times \rho \times SOC \times (100 - cf)
\]  

(2.18)

where \(SOCD\) is the soil carbon density in Mg C ha\(^{-1}\), \(d\) is the depth of soil in meters, \(\rho\) is the soil bulk density in kg m\(^{-3}\), \(SOC\) is the soil organic carbon content in weight percentage and \(cf\) is the coarse fragment content (particle size > 2 mm) in weight percentage, all taken from the WISE dataset.

The cropland SOC density (Mg ha\(^{-1}\)) was mapped at the national scale to demonstrate its vertical and lateral distributions (Figure 2.18).
Vertical distribution

The understanding of the vertical distribution patterns of SOC, although not as frequently reported as the geographical ones, is key to predicting and simulating the influences of climate and human activities on the terrestrial carbon cycle (Wang et al., 2004). At the national scale, the cropland SOCD was averaged at 28.1, 13.3, 9.4, 7.1 and 6.1 Mg C ha$^{-1}$ for the 0-20, 20-40, 40-60, 60-80 and 80-100 cm sections respectively. The top-section contained 2 times more SOC than the first sub-section, and 3 times more than the average of all sub-sections. The decreasing rate of SOCD values over depth was 2.6, 1.5, 1.2 and 0.7 percent per meter between two neighboring sections, counted consecutively from soil surface downwards, respectively. On average, the SOCD decreased at a rate of 1 percent per meter over the 1 m depth range. More than half (i.e., 54%) of the cropland SOC is distributed in the top 30 cm layer ("topsoil" hereafter, Figure 2.18(a)), whereas the 70 cm layer ("subsoil", Figure 2.18(b)) underneath took the remaining 45%. The close match, between the vertical patterns found here and a published study on SOC density (Wang et al., 2004) based on the extrapolation of 2,473 soil profiles collected in the Second National Soil Survey in China (NSSO, 1998) during 1979-1994, partially validated the applicability and accuracy of the WISE dataset in revealing the soil realities at the national scale in China.

Regional distribution

At the provincial scale, the SOCD of the topsoil varied greatly from one province to another, with values ranging from 14.4 (in the loess dominant province of Gansu in the
northwest) to 37.7 Mg C ha\(^{-1}\) (in the “black soil” province of Hei Longjiang in the north-east). In general, high SOC values of the top-section were found in north-eastern provinces and lower ones in the southwest. This is similar to patterns observed elsewhere (Wu et al., 2003b; Song et al., 2005; Tang et al., 2006; Xie et al., 2007). Although the north-western provinces had the lowest values (14.8 Mg C ha\(^{-1}\)), they hardly influenced the general patterns of cropland SOCD in the national scale due to the scarcity of cropland in the region. The cropland SOCD tended to decline slightly from the northeast (34.8 Mg C ha\(^{-1}\)) to the southwest (21.2 Mg C ha\(^{-1}\)). The subsoil followed the same spatial patterns as those of the topsoil (Figures 2.18 and 2.19).

![Figure 2.19: Regional fluctuations (a) and variabilities, given in the boxplots (b), of cropland SOC density in China](image-url)
2.7 Soil

SOC storage

Much attention had been paid to estimating the carbon sequestration capacity of the topsoil (Pan et al., 2004; Song et al., 2005; Tang et al., 2006) while much less attention had been given to the subsoil, which contains also a substantial amount of carbon (Sombroek, 1993; Batjes, 1996; Batjes and Sombroek, 1997). More importantly, the turnover cycle of the SOC in an organic A horizon is of the decadal order or less, while that in the mineral B and C horizons is on the order of hundreds or thousands of years or even longer (Li et al., 1994; Schimel et al., 1994; Barrett et al., 2006). Therefore an effort has been made in assessing the SOC storage of both the topsoil and the subsoil of cropland in China, till 1 meter depth (Table 2.12).

Table 2.12: Zone-specific soil organic carbon storage (Tg) in cropland of China

<table>
<thead>
<tr>
<th>Zone</th>
<th>Cropland†</th>
<th>Topsoil</th>
<th>Subsoil</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11,618</td>
<td>1012.5</td>
<td>421.9</td>
<td>1434.4</td>
</tr>
<tr>
<td>2</td>
<td>5,589</td>
<td>365.6</td>
<td>118.6</td>
<td>484.2</td>
</tr>
<tr>
<td>3</td>
<td>13,855</td>
<td>878.0</td>
<td>322.6</td>
<td>1200.6</td>
</tr>
<tr>
<td>4</td>
<td>7,296</td>
<td>358.2</td>
<td>135.4</td>
<td>493.7</td>
</tr>
<tr>
<td>5</td>
<td>11,822</td>
<td>744.1</td>
<td>275.0</td>
<td>1019.1</td>
</tr>
<tr>
<td>6</td>
<td>10,132</td>
<td>524.9</td>
<td>166.4</td>
<td>691.4</td>
</tr>
<tr>
<td>7</td>
<td>4,892</td>
<td>266.7</td>
<td>79.0</td>
<td>345.7</td>
</tr>
<tr>
<td>8</td>
<td>2,573</td>
<td>109.9</td>
<td>57.6</td>
<td>167.5</td>
</tr>
<tr>
<td>9</td>
<td>178</td>
<td>6.1</td>
<td>1.8</td>
<td>7.8</td>
</tr>
<tr>
<td>Sum</td>
<td>67,955</td>
<td>4,266.0</td>
<td>1,578.4</td>
<td>5,844.3</td>
</tr>
<tr>
<td>%</td>
<td>-</td>
<td>73.0</td>
<td>27.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

† Refer to Table 2.2 for zone names
‡ Number of cells, each of 25 km²

Results showed that the Chinese cropland stored totally 5.8 Pg C over 1 meter soil depth, including 4.2 Pg in the topsoil and 1.6 Pg in the subsoil. The topsoil and subsoil maintained a 73:27 ratio in storing the SOC. A comparable yet slightly higher ratio of 77:23 was reported by Wu et al. (2003c) for both the cultivated and the uncultivated lands in China, suggesting that the topsoil may be more prone to lose SOC, than the subsoil, due to the change of land use from non-agriculture to agriculture (Mann, 1986; Li and Zhao, 2001). The topsoil SOC storage (SOCS) of 4.2 Pg is slightly higher (by 5%) than the simulation result of 4.0 Pg (Tang et al., 2006) based on the biogeochemical model of DeNitrification and DeComposition or DNDC (Li et al., 1994, 2003), but considerably lower (by 17.6%) than 5.1 Pg of another estimation (Song et al., 2005). In the latter case, the use of the 1980 value of the area of cultivated land in calculation may result in an overestimation of SOCS by 13.6%, against the SOCS estimated using the area of cultivated land in the publication year of 2005. The cultivated land in China has undergone a steady decrease since 1979. The annual decreasing rate was reported 0.26% before 1999 and 1.45% after that (WRI, 1987; NBSC, 1999; Feng et al., 2005; WRI, 2005) (see §8.5 for details). Consequently the area-corrected SOCS would be 4.5 Pg which is much closer to 4.2 Pg (7% difference). Wu et al. (2003c) presented yet another nation-wide estimation in an attempt to quantify the land use induced changes of SOCS in China. They found that the SOCD of uncultivated soils was 88 Mg ha⁻¹ while the combined SOCD
of both cultivated and uncultivated soils was 80 Mg ha\(^{-1}\), advising an unfavorable effect of agricultural cultivation on SOC density. Suppose \(x\) being the SOCD of the cultivated soils:

\[
\frac{88 \times (100 - 18) + 18 \times x}{100} = 80
\]  

(2.19)

where 88 and 80 are above-mentioned SOCD values in Mg ha\(^{-1}\), 18 is the % of cultivated over total land surface based on the 5 km \(\times\) 5 km grid map of cropland (Figure E.3).

The solution to Equation 2.19

\[x = 43.5\]

gives the SOCD of the cultivated soils in Mg ha\(^{-1}\). Consequently the SOCS of cultivated soils is deduced as 5.2 Pg based on the census data of area of cultivated land (NBSC, 2005), 77% of it, 4.0 Pg, is finally attributed to the topsoil.

The above four SOCS estimates are averaged 5.7 Pg with a min-max range of 5.2-6.2 Pg (17.5% difference against the mean). However, Xie et al. (2007) recently reported a significantly different estimate. The SOCS of the paddy soils was estimated 0.82 Pg and 2.09 Pg for the topsoil (0-15.2 cm) and the subsoil (15.2-90.5 cm) respectively, while SOCS for the upland soils was 3.07 Pg and 7.00 Pg for the topsoil (0-19.4 cm) and the subsoil (19.4-107.9 cm) respectively based on soil survey data in 1980 (NSSO, 1998). According to the farmland area and depth range, these values are adjusted, for the paddy soil, to 0.83 Pg and 3.26 Pg in the top (0-30 cm) and sub (30-100 cm) layers respectively and, for the upland soil, 1.31 Pg and 4.62 Pg in top- and sub-layers respectively. The cropland SOCS was therefore estimated as 4.09 Pg for the topsoil and 5.93 Pg for the subsoil, with 10 Pg as the total storage. To compare, the difference occurs mainly in the subsoil (5.9 Pg versus 1.5 Pg) than in the topsoil (5.9 Pg versus 4.2 Pg). This can partly be explained by the way in which both the cropland is defined and the top- and subsoil are differentiated.

The upland class may have counted non-cultivated land as cropland but this can not be verified without using a land-use map of proper scale.

For clarity, the aforementioned SOCS estimations are summarized for comparison in Table 2.13.

<table>
<thead>
<tr>
<th>Source</th>
<th>Topsoil</th>
<th>Subsoil</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>This dissertation</td>
<td>4.2</td>
<td>1.6</td>
<td>5.8</td>
</tr>
<tr>
<td>Xie et al. (2007)</td>
<td>4.1(^a)</td>
<td>5.9(^b)</td>
<td>10.0(^e)</td>
</tr>
<tr>
<td>Tang et al. (2006)</td>
<td>4.0</td>
<td>1.5(^d)</td>
<td>5.5(^c)</td>
</tr>
<tr>
<td>Song et al. (2005)</td>
<td>4.5(^c)</td>
<td>1.7(^d)</td>
<td>6.2(^c)</td>
</tr>
<tr>
<td>Wu et al. (2003c)</td>
<td>4.0(^f)</td>
<td>1.2(^g)</td>
<td>5.2(^h)</td>
</tr>
</tbody>
</table>

\(^a\) Area-and-depth-adjusted value from original 3.9 Pg  
\(^b\) Area-and-depth-adjusted value from original 9.1 Pg  
\(^c\) Topsoil + subsoil  
\(^d\) Calculated using top:sub ratio of 73:27  
\(^e\) Area-adjusted value from original 5.1 Pg  
\(^f\) Calculated as 77% of total  
\(^g\) Calculated using top:sub ratio of 77:23  
\(^h\) Calculated using deduced SOCD
Chapter 3
Cropping Systems in Food Production

3.1 Introduction

The production of food began in China with the domestication of wild species of crops, known as staples nowadays, early in the prehistoric times. The drought-resistant millet \((Setaria italica)\) and glutenous rice \((Oryza sativa)\), for instance, had been indigenously domesticated at latest by the 5th millennium BC. They were grown on the eolian and alluvial loess soils of the northwest and the north, and in the wetlands of the southeast, respectively (Chang et al., 2002; Liu, 2005b). Although introduced from the Middle East about 7,000 BP, the history of wheat \((Triticum aestivum)\) cultivation was only next to that of millet or rice. However, the introduction of another major food crop, maize \((Zea mays)\), into China happened much more recently in the 16th century (Ding, 1988).

Organized production of food had become controllable at the sub-regional or regional scale with application of coordinated development of farming techniques in Qin and Han Dynasties (2,000 BP), which was recognized afterwards as the beginning of the formation of the traditional agriculture in China. Since then crop rotations had been implemented with locally adapted cultivars and field management options. For instance, the technique of rice-wheat multiple cropping was introduced in a few of the most developed districts approximately during the early Tang Dynasty (618-906 AD), including the Yangtze Delta, the Chengdu Plain and along both banks of the Yangtze River (Hu and Chang, 1984). In practice, regionally optimized cropping systems produced more and helped the country feed its increasing population in modern times.

China has become the world’s largest grain producer since the late 1980s or early 1990s, shortly after the rural reform and the “open-door” policies had been adopted. China produced 484 million Mg of grain in 2005 (NBSC, 2005), which was 10% higher than that of the second largest producer, the USA (Figure 3.1).

3.2 Definition of food and grain crops

Grain crops in China, including rice, wheat, maize, millet, sorghum, and other miscellaneous cereals, took 78% of the sown area of all food crops \((i.e.,\) grain plus pulse, root and tuber crops) and produced 88% of the total production of food in 2005. The average unit yield of grains was 5,187 kg ha\(^{-1}\), which was 566 kg ha\(^{-1}\) higher than that of the food
3.3 Crop phenology

The phenology of 12 representative food crops or crop species was studied and, subsequently, the crop cycle of each crop was determined and its subdivisions distinguished. By definition (FAO, 1985), a crop cycle is the period required for an annual crop to complete its annual cycle of establishment, growth and production of the harvested part. Perennial crops have crop cycles of more than one year. In the simplest case, a crop cycle can be merely defined by its beginning and the ending dates (and therefore implicitly its length). A crop cycle is completely defined only after the following crop development stages (§3.3.1) have been defined: the initial (IS), crop development (CD), mid-season
(MS) and late season (LS). Technical details about the WLES implementation and use of the crop cycle concept can be found in §4.3 of Volume II.

Spatial distributions of food crop species in China are represented in Figure 3.2.

![Figure 3.2: Spatial distribution of food crops and species](image)

### 3.3.1 Crop development stages

As the crop develops, the ground cover, crop height and the leaf area change. The whole crop cycle can be divided into four distinct development stages (Allen et al., 1998):

1. **Initial (IS):** The initial stage runs from planting date to approximately 10% ground cover. The length of the initial stage is highly dependent on the crop, the crop variety, the planting date and the climate. The end of the initial stage is determined as the time when approximately 10% of the ground surface is covered by green vegetation. For perennial crops, the planting date is replaced by the “green-up” date, *i.e.*, the date when the initiation of new leaves occurs;

2. **Crop development (CD):** The crop development stage runs from 10% ground cover to effective full cover. Effective full cover for many crops occurs at the initiation of flowering. For row crops, such as beans, sugar beets, potatoes and corn, where rows commonly interlock leaves, effective cover can be defined as the time when some leaves of plants in adjacent rows begin to intermingle so that soil shading becomes nearly complete, or when plants reach nearly full size if no intermingling occurs. Because it is difficult to visually determine when densely sown vegetation, such as
winter and spring cereals and some grasses, reach effective full cover, the more easily detectable stage of heading (flowering) is generally used for these types of crops.

Another way to estimate the occurrence of effective full cover is when the leaf area index (LAI) reaches 3 m$^2$ m$^{-2}$. LAI is defined as the average total area of leaves (one side) per unit area of ground surface;

3. Mid-season (MS): The mid-season stage runs from effective full cover to the start of maturity. The start of maturity is often indicated by the beginning of the ageing, yellowing or senescence of leaves, leaf drop, or the browning of fruit. The mid-season stage is the longest stage for perennials and for many annuals, but it may be relatively short for vegetable crops that are harvested fresh for their green vegetation;

4. Late season (LS): The late season stage runs from the start of maturity to harvest or full senescence. For some perennial vegetation in frost-free climates, crops may grow year round so that the date of termination may be taken as the same as the date of “planting”.

The crop development stages of each of the food crops per zone are summarized in Table 3.2, based on local crop phenological observations (Zhang et al., 1987).

<table>
<thead>
<tr>
<th>Zone†</th>
<th>Crop ID</th>
<th>Crop</th>
<th>Sowing Date</th>
<th>IS</th>
<th>CD</th>
<th>MS</th>
<th>LS</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>northern rice</td>
<td>May 01</td>
<td>25</td>
<td>40</td>
<td>40</td>
<td>35</td>
<td>140</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>spring wheat</td>
<td>Apr 05</td>
<td>15</td>
<td>25</td>
<td>50</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>spring maize</td>
<td>May 01</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>30</td>
<td>150</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>soybean</td>
<td>May 01</td>
<td>25</td>
<td>35</td>
<td>55</td>
<td>25</td>
<td>140</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>millet</td>
<td>May 01</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>30</td>
<td>140</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>sorghum</td>
<td>May 01</td>
<td>25</td>
<td>40</td>
<td>45</td>
<td>30</td>
<td>140</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>potato</td>
<td>May 01</td>
<td>25</td>
<td>35</td>
<td>40</td>
<td>30</td>
<td>130</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>spring wheat</td>
<td>Mar 15</td>
<td>15</td>
<td>25</td>
<td>50</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>spring maize</td>
<td>Apr 25</td>
<td>30</td>
<td>45</td>
<td>45</td>
<td>30</td>
<td>150</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>soybean</td>
<td>May 01</td>
<td>25</td>
<td>35</td>
<td>55</td>
<td>25</td>
<td>140</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>millet</td>
<td>May 01</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>30</td>
<td>140</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>sorghum</td>
<td>May 01</td>
<td>25</td>
<td>40</td>
<td>45</td>
<td>30</td>
<td>140</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>potato</td>
<td>May 01</td>
<td>25</td>
<td>35</td>
<td>40</td>
<td>30</td>
<td>130</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>winter wheat</td>
<td>Oct 01</td>
<td>35</td>
<td>50</td>
<td>105</td>
<td>60</td>
<td>150</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>summer maize</td>
<td>Jun 10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>soybean</td>
<td>Jun 01</td>
<td>20</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>millet</td>
<td>May 15</td>
<td>20</td>
<td>40</td>
<td>40</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>sorghum</td>
<td>Apr 15</td>
<td>20</td>
<td>40</td>
<td>40</td>
<td>30</td>
<td>130</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>potato</td>
<td>Mar 15</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>100</td>
</tr>
</tbody>
</table>

To be continued...
### 3.3 Crop phenology

#### Table 3.2 continued

<table>
<thead>
<tr>
<th>Zone†</th>
<th>Crop ID</th>
<th>Crop</th>
<th>Sowing Date</th>
<th>IS</th>
<th>CD</th>
<th>MS</th>
<th>LS</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>winter wheat</td>
<td>Sep 20</td>
<td>40</td>
<td>80</td>
<td>115</td>
<td>65</td>
<td>280</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>intermediate rice</td>
<td>Jun 15</td>
<td>40</td>
<td>20</td>
<td>35</td>
<td>40</td>
<td>130</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>early rice</td>
<td>Apr 15</td>
<td>15</td>
<td>20</td>
<td>40</td>
<td>35</td>
<td>110</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>late rice</td>
<td>Jun 20</td>
<td>40</td>
<td>30</td>
<td>30</td>
<td>40</td>
<td>140</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>winter wheat</td>
<td>Oct 20</td>
<td>30</td>
<td>40</td>
<td>100</td>
<td>50</td>
<td>220</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>soybean</td>
<td>May 01</td>
<td>20</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>sorghum</td>
<td>May 01</td>
<td>20</td>
<td>35</td>
<td>40</td>
<td>25</td>
<td>120</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>potato</td>
<td>May 01</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>30</td>
<td>150</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>intermediate rice</td>
<td>Jun 15</td>
<td>40</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>130</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>early rice</td>
<td>Mar 01</td>
<td>30</td>
<td>25</td>
<td>55</td>
<td>20</td>
<td>130</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>late rice</td>
<td>Jun 15</td>
<td>30</td>
<td>20</td>
<td>60</td>
<td>35</td>
<td>145</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>spring wheat</td>
<td>Mar 15</td>
<td>40</td>
<td>20</td>
<td>60</td>
<td>30</td>
<td>150</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>winter wheat</td>
<td>Sep 15</td>
<td>40</td>
<td>75</td>
<td>120</td>
<td>65</td>
<td>300</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>spring maize</td>
<td>Mar 01</td>
<td>35</td>
<td>50</td>
<td>50</td>
<td>35</td>
<td>170</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>soybean</td>
<td>Apr 20</td>
<td>25</td>
<td>35</td>
<td>55</td>
<td>25</td>
<td>140</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>millet</td>
<td>May 15</td>
<td>15</td>
<td>25</td>
<td>40</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>sorghum</td>
<td>Apr 15</td>
<td>25</td>
<td>50</td>
<td>50</td>
<td>35</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>potato</td>
<td>Apr 15</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>30</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>spring wheat</td>
<td>Apr 15</td>
<td>20</td>
<td>40</td>
<td>65</td>
<td>35</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>winter wheat</td>
<td>Sep 15</td>
<td>40</td>
<td>90</td>
<td>150</td>
<td>70</td>
<td>350</td>
</tr>
</tbody>
</table>

† Refer to Table 2.2 for zone names

### 3.3.2 Crop growth periods

During the growth span, the crop passes through various phases and periods of growth. The growth rhythm of crops is slow during some periods and fast during some others. Hence the demand of water supply and, more importantly, response to water deficit varies during the whole crop cycle.

The crop cycle is divided into five distinct growth periods:

- Establishment(0);
3.4 Crop rotations

- Vegetative(1);
- Flowering(2);
- Yield formation(3);
- Ripening(4).

The crop growth periods and the crop development stages (§3.3.1) are defined on the basis of different criteria and serve different purposes. Among others, the crop development stages are used for determination of the crop coefficient \(k_c\), §4.4 in the process of estimating the maximum crop evapotranspiration \(ET_m\), whereas the crop growth periods are used to differentiate the crop sensitivity to water deficit in the process of evaluating the yield response factor \(k_y\), §4.5).

Due to lack of independent measurements, the durations of the crop growth periods of food crops in China were deduced from those of the crop development stages using the inter-mapping relationship found in Table 3.3.

<table>
<thead>
<tr>
<th>ID</th>
<th>CGP</th>
<th>ID</th>
<th>CDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>establishment</td>
<td>1</td>
<td>initial</td>
</tr>
<tr>
<td>1</td>
<td>vegetative</td>
<td>2</td>
<td>crop development</td>
</tr>
<tr>
<td>2</td>
<td>flowering</td>
<td>3</td>
<td>mid-season</td>
</tr>
<tr>
<td>3</td>
<td>yield formation</td>
<td>4</td>
<td>late season</td>
</tr>
<tr>
<td>4</td>
<td>ripening</td>
<td>sum</td>
<td>crop cycle</td>
</tr>
</tbody>
</table>

sum crop cycle

3.4 Crop rotations

The actual cropping patterns of major food crops on a per zone basis, which were constrained by the agro-natural resource conditions (chapter 2), were characterized in Table 3.4 and presented by the cropping calendar found in Figure 3.3.

<table>
<thead>
<tr>
<th>Zone</th>
<th>MCI (%)</th>
<th>Dominant cropping systems</th>
<th>Major crops or crop rotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85</td>
<td>1 crop per year</td>
<td>northern rice, spring wheat, spring maize, soybean</td>
</tr>
<tr>
<td>2</td>
<td>81</td>
<td>1 crop per year</td>
<td>spring wheat, spring maize, soybean</td>
</tr>
<tr>
<td>3</td>
<td>144</td>
<td>2 crops per year</td>
<td>winter wheat-summer maize, winter wheat-summer maize, cotton-wheat-maize</td>
</tr>
<tr>
<td>4</td>
<td>88</td>
<td>1 crop per year to 2 crops per year</td>
<td>winter wheat-summer maize, summer maize-soybean, winter wheat</td>
</tr>
<tr>
<td>5</td>
<td>157</td>
<td>2 crops per year to 3 crops per 2 years</td>
<td>winter wheat-intermediate rice, early rice-late rice</td>
</tr>
</tbody>
</table>

To be continued...
Table 3.4 continued

<table>
<thead>
<tr>
<th>Zone†</th>
<th>MCI‡</th>
<th>Dominant cropping systems</th>
<th>Major crops or crop rotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>127</td>
<td>2 crops per year to 3 crops per 2 years</td>
<td>winter wheat-intermediate rice, winter wheat-summer maize, winter wheat-soybean, early rice-late rice</td>
</tr>
<tr>
<td>7</td>
<td>123</td>
<td>3 crops per year</td>
<td>winter wheat-early rice-late rice, winter wheat-spring maize-late rice</td>
</tr>
<tr>
<td>8</td>
<td>87</td>
<td>1 crop per year</td>
<td>winter wheat, spring wheat, summer maize, soybean</td>
</tr>
<tr>
<td>9</td>
<td>93</td>
<td>1 crop per year</td>
<td>winter wheat, spring wheat, barley</td>
</tr>
</tbody>
</table>

† Refer to Table 2.2 for zone names
‡ Multi-cropping index; data source: NBSC (2005); national average: 118; refer to §8.5 for definition and further discussions

Figure 3.3: Cropping calendar of major food crops per zone. Calendar bars are annotated with crop IDs which were defined in Table 3.2; zone names were given in Table 2.2
3.5 Sown structure of food crops

An inventory of the sown areas of food crops in 2005 was made on a per province basis and the results are shown in Table 3.5. Although not explicitly shown in the table, the sown structure of an individual crop can be obtained as

$$\frac{S_{ij}}{F_j} \cdot 100 \quad (3.1)$$

where $S_{ij}$ and $F_j$ are sown areas of crop $i$ and food crops in total in province $j$.

Table 3.5: Sown area of major food crops in China, 2005 ($\times 1,000$ ha)

<table>
<thead>
<tr>
<th>Prov†</th>
<th>Agric‡</th>
<th>Food</th>
<th>Grain</th>
<th>Rice</th>
<th>Wheat</th>
<th>Maize</th>
<th>Beans</th>
<th>Tubers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>312.5</td>
<td>154.5</td>
<td>135.6</td>
<td>0.8</td>
<td>39.2</td>
<td>93.5</td>
<td>15.2</td>
<td>3.7</td>
</tr>
<tr>
<td>2</td>
<td>504.3</td>
<td>263.5</td>
<td>232.3</td>
<td>13.7</td>
<td>79.0</td>
<td>134.8</td>
<td>30.1</td>
<td>1.2</td>
</tr>
<tr>
<td>3</td>
<td>6,959.4</td>
<td>6,003.4</td>
<td>5,322.9</td>
<td>83.5</td>
<td>2,161.5</td>
<td>2,630.6</td>
<td>359.7</td>
<td>320.8</td>
</tr>
<tr>
<td>4</td>
<td>3,741.5</td>
<td>2,925.4</td>
<td>2,216.3</td>
<td>2.6</td>
<td>648.9</td>
<td>1,125.6</td>
<td>348.8</td>
<td>360.3</td>
</tr>
<tr>
<td>5</td>
<td>5,924.0</td>
<td>4,181.1</td>
<td>2,583.4</td>
<td>80.9</td>
<td>2,167.5</td>
<td>1,675.6</td>
<td>1,069.6</td>
<td>528.1</td>
</tr>
<tr>
<td>6</td>
<td>3,723.3</td>
<td>2,906.7</td>
<td>2,434.7</td>
<td>544.2</td>
<td>20.6</td>
<td>1,598.8</td>
<td>335.1</td>
<td>138.5</td>
</tr>
<tr>
<td>7</td>
<td>4,904.0</td>
<td>4,312.1</td>
<td>3,594.1</td>
<td>600.1</td>
<td>11.4</td>
<td>2,103.5</td>
<td>332.8</td>
<td>84.8</td>
</tr>
<tr>
<td>8</td>
<td>9,888.4</td>
<td>8,458.0</td>
<td>4,198.0</td>
<td>1,587.8</td>
<td>255.0</td>
<td>2,179.5</td>
<td>3,913.6</td>
<td>346.4</td>
</tr>
<tr>
<td>9</td>
<td>404.4</td>
<td>154.7</td>
<td>145.0</td>
<td>111.8</td>
<td>21.9</td>
<td>4.2</td>
<td>8.0</td>
<td>0.9</td>
</tr>
<tr>
<td>10</td>
<td>7,669.0</td>
<td>4,774.6</td>
<td>4,310.8</td>
<td>2,112.9</td>
<td>1,601.2</td>
<td>389.1</td>
<td>349.6</td>
<td>114.2</td>
</tr>
<tr>
<td>11</td>
<td>2,778.4</td>
<td>1,454.5</td>
<td>1,179.0</td>
<td>1,028.1</td>
<td>59.5</td>
<td>54.5</td>
<td>184.9</td>
<td>98.7</td>
</tr>
<tr>
<td>12</td>
<td>9,200.4</td>
<td>6,312.2</td>
<td>4,977.5</td>
<td>2,129.7</td>
<td>2,059.9</td>
<td>662.3</td>
<td>977.0</td>
<td>357.7</td>
</tr>
<tr>
<td>13</td>
<td>2,519.3</td>
<td>1,482.4</td>
<td>1,036.3</td>
<td>985.1</td>
<td>6.2</td>
<td>37.8</td>
<td>115.1</td>
<td>330.9</td>
</tr>
<tr>
<td>14</td>
<td>5,182.8</td>
<td>3,350.1</td>
<td>3,067.8</td>
<td>3,029.7</td>
<td>19.1</td>
<td>14.4</td>
<td>158.8</td>
<td>123.5</td>
</tr>
<tr>
<td>15</td>
<td>10,638.6</td>
<td>6,176.3</td>
<td>5,600.5</td>
<td>124.4</td>
<td>2,968.2</td>
<td>2,455.1</td>
<td>256.6</td>
<td>319.3</td>
</tr>
<tr>
<td>16</td>
<td>13,789.7</td>
<td>8,970.1</td>
<td>7,906.6</td>
<td>508.5</td>
<td>4,856.0</td>
<td>2,420.0</td>
<td>625.9</td>
<td>437.6</td>
</tr>
<tr>
<td>17</td>
<td>7,155.9</td>
<td>3,712.4</td>
<td>3,004.1</td>
<td>1,989.6</td>
<td>602.9</td>
<td>357.5</td>
<td>304.1</td>
<td>404.2</td>
</tr>
<tr>
<td>18</td>
<td>7,886.2</td>
<td>4,754.1</td>
<td>4,114.2</td>
<td>3,716.8</td>
<td>76.2</td>
<td>276.5</td>
<td>279.7</td>
<td>306.3</td>
</tr>
<tr>
<td>19</td>
<td>4,808.0</td>
<td>2,789.7</td>
<td>2,294.1</td>
<td>2,139.0</td>
<td>6.0</td>
<td>137.9</td>
<td>107.9</td>
<td>387.7</td>
</tr>
<tr>
<td>20</td>
<td>6,368.2</td>
<td>3,511.2</td>
<td>2,967.3</td>
<td>2,356.0</td>
<td>11.7</td>
<td>586.6</td>
<td>283.4</td>
<td>260.5</td>
</tr>
<tr>
<td>21</td>
<td>826.9</td>
<td>471.8</td>
<td>348.7</td>
<td>334.7</td>
<td>0.0</td>
<td>13.7</td>
<td>13.7</td>
<td>109.3</td>
</tr>
<tr>
<td>22</td>
<td>12,822.8</td>
<td>8,992.9</td>
<td>6,294.9</td>
<td>2,813.1</td>
<td>1,536.2</td>
<td>1,633.0</td>
<td>789.8</td>
<td>1,908.3</td>
</tr>
<tr>
<td>23</td>
<td>4,695.0</td>
<td>3,037.2</td>
<td>1,923.3</td>
<td>716.5</td>
<td>429.2</td>
<td>706.5</td>
<td>324.9</td>
<td>789.1</td>
</tr>
<tr>
<td>24</td>
<td>5,890.0</td>
<td>4,158.5</td>
<td>3,040.7</td>
<td>1,086.2</td>
<td>543.3</td>
<td>1,111.1</td>
<td>493.1</td>
<td>624.7</td>
</tr>
<tr>
<td>25</td>
<td>231.2</td>
<td>179.8</td>
<td>168.9</td>
<td>1.1</td>
<td>40.6</td>
<td>3.3</td>
<td>9.0</td>
<td>1.9</td>
</tr>
<tr>
<td>26</td>
<td>4,099.8</td>
<td>3,134.1</td>
<td>2,462.9</td>
<td>145.8</td>
<td>1,152.7</td>
<td>1,047.4</td>
<td>370.4</td>
<td>300.8</td>
</tr>
<tr>
<td>27</td>
<td>3,668.9</td>
<td>2,534.6</td>
<td>1,751.0</td>
<td>4.9</td>
<td>933.5</td>
<td>487.7</td>
<td>234.5</td>
<td>549.1</td>
</tr>
<tr>
<td>28</td>
<td>473.3</td>
<td>244.7</td>
<td>136.8</td>
<td>0.0</td>
<td>102.2</td>
<td>1.6</td>
<td>40.1</td>
<td>67.8</td>
</tr>
<tr>
<td>29</td>
<td>1,158.3</td>
<td>791.7</td>
<td>603.5</td>
<td>64.4</td>
<td>279.0</td>
<td>187.9</td>
<td>84.6</td>
<td>103.6</td>
</tr>
<tr>
<td>30</td>
<td>3,592.3</td>
<td>1,413.9</td>
<td>1,307.4</td>
<td>66.8</td>
<td>686.4</td>
<td>518.0</td>
<td>83.4</td>
<td>23.1</td>
</tr>
<tr>
<td>31</td>
<td>1,241.4</td>
<td>269.0</td>
<td>269.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Total</td>
<td>154,794.2</td>
<td>101,875.2</td>
<td>79,619.5</td>
<td>28,647.7</td>
<td>21,626.2</td>
<td>25,446.0</td>
<td>12,798.9</td>
<td>9,457.0</td>
</tr>
<tr>
<td>%</td>
<td>151.9</td>
<td>100.0</td>
<td>78.2</td>
<td>28.1</td>
<td>21.2</td>
<td>25.0</td>
<td>12.6</td>
<td>9.3</td>
</tr>
</tbody>
</table>

† Province. See Table 2.1 for names; ‡ Agricultural crops; ∗ Including 32 and 33
Chapter 4

Quantitative Land Evaluation Methodologies

4.1 Introduction

The quantitative assessment of food productivity was conducted using a three-step, hierarchical, deterministic land evaluation model (Ye and Van Ranst, 2002; Ye et al., 2007, 2008), based for specific crops on the quantification of the radiation-thermal (RPP) (FAO, 1984), the water-limited (WPP) and the land production potentials (LPP) (Sys et al., 1991).

The RPP (§4.4) represents the maximum yield a high-yielding cultivar can achieve under optimal irrigated conditions, *i.e.* no water and nutrients shortage, no pests and diseases and growing on a constraint-free soil. The economic yield is given by:

$$RPP = B_n \cdot HI$$

where $B_n$ is the net biomass production (§4.4.1), and $HI$ is the harvest index (§4.4.4).

The WPP (§4.5) approaches the maximum rainfed yield by incorporating the yield reduction effects of water stress during a crop cycle. Water stress occurs when crop can no longer evapotranspire at the maximum rate, or in other words, when the actual crop evapotranspiration ($ET_a$) becomes smaller than the maximum crop evapotranspiration ($ET_m$):

$$WPP = RPP \cdot \left[1 - k_y \left(1 - \frac{ET_a}{ET_m}\right)\right]$$

where $k_y$ is the yield response factor (§4.5.5). The $ET_m$ is obtained as the product of the reference evapotranspiration ($ET_o$) (§4.2) and the crop coefficient ($K_c$) (§4.5.1). The $ET_a$ can only be estimated after the soil water balance (§4.5.4) is simulated.

The LPP (§4.6) is the closest approximation of the real-world crop yield. It adopts the effects of a limiting soil fertility and of non-optimal management practices, represented by the soil ($S_y$) (§4.6.1) and management ($M_y$) (§4.6.2) indices, respectively:

$$LPP = WPP \cdot S_y \cdot M_y$$

4.2 Reference evapotranspiration

Reference evapotranspiration ($ET_o$) is defined as “the rate of evapotranspiration from a hypothetical reference crop with an assumed crop height of 0.12 m, a fixed surface
4.2 Reference evapotranspiration

...resistance of 69 s m$^{-1}$ and an albedo of 0.23, closely resembling the evapotranspiration from an extensive surface of green grass of uniform height, actively growing, well-watered, and completely shading the ground” (Doorenbos and Pruitt, 1977).

$ET_o$ can be estimated from a set of climatic parameters. Many algorithms had been developed since Penman (1948) published his widely cited calculation procedures, such as Blaney and Criddle (1950), which was further modified by Doorenbos and Kassam (1979); Monteith (1965); Doorenbos and Pruitt (1977); Frére and Popov (1979); and Smith (1991). Among them, the modified Penman-Monteith approach (Smith, 1991) was recommended by the expert consultation on procedures for revision of FAO guidelines for prediction of crop water requirements (Smith, 1991) as the best-performing combination equation, and therefore adopted in the quantitative land evaluation methodologies presented in this chapter.

4.2.1 Combination equation

The final equation of $ET_o$ consists of an aerodynamic ($ET_{aero}$) and a radiation term ($ET_{rad}$), all in mm day$^{-1}$:

$$ET_o = ET_{aero} + ET_{rad}$$ (4.1)

4.2.2 Aerodynamic term

The aerodynamic term is given by:

$$ET_{aero} = \frac{\gamma}{\Delta + \gamma^*} \cdot \frac{900}{t + 273} \cdot (e_s - e_a) \cdot u^2$$ (4.2)

where $\gamma$ is the psychrometric constant (kPa °C$^{-1}$); $\gamma^*$ the modified psychrometric constant (kPa °C$^{-1}$); $\Delta$ the slope of the vapor pressure curve (kPa °C$^{-1}$); $t$ the air temperature (°C); $(e_s - e_a)$ the vapor pressure deficit (kPa); and $u^2$ the wind velocity at 2 m height (m s$^{-1}$).

The psychrometric constant $\gamma$ is calculated by the atmospheric pressure $P$ (kPa) and the latent heat of evaporation $\lambda$ (MJ kg$^{-1}$):

$$\gamma = 1.6286 \times 10^{-3} P/\lambda$$ (4.3)

and

$$P = 101.3 \left( \frac{293 - 6.5 \times 10^{-3} Z}{293} \right)^{5.256}$$ (4.4)

where $Z$ is the altitude (m a.s.l.) of the place in concern.

The modified psychrometric constant $\gamma^*$ is a function of the crop canopy resistance $r_c$ and the aerodynamic resistance $r_a$:

$$\gamma^* = \gamma \left(1 + \frac{r_c}{r_a}\right)$$

$$= \gamma \left(1 + \frac{200/LAI}{208/u^2}\right)$$

$$= \gamma \left(1 + \frac{25u^2}{26LAI}\right)$$ (4.5)
4.2 Reference evapotranspiration

where LAI is the leaf area index. For the reference crop with a crop height \( h_c \) of 0.12 m, the LAI is calculated as \( LAI = 24 \cdot h_c = 2.88 \text{ m}^2 \text{ m}^{-2} \). The canopy resistance then becomes \( r_c = 200/2.88 = 69 \text{ s m}^{-1} \).

The slope of the vapor pressure curve, \( \Delta \), is governed by the saturated vapor pressure \( e_s \) (kPa) and the mean air temperature \( t_{\text{mean}} \) (°C):

\[
\Delta = 4098 \cdot \frac{e_s}{(t_{\text{mean}} + 237.3)^2}
\]

(4.6)

The saturated vapor pressure is the mere function of \( t_{\text{mean}} \):

\[
e_s = 0.6108 \cdot e^{\left(\frac{17.27 \cdot t_{\text{mean}}}{t_{\text{mean}} + 237.3}\right)}
\]

(4.7)

The actual vapor pressure \( e_a \) (kPa) is thus derived:

\[
e_a = e_s \cdot RH/100
\]

(4.8)

The wind velocity readings from instruments that are placed at a height different from 2 m have to be adjusted using the following equation:

\[
u_2 = 4.868 \cdot \frac{u_z}{\ln(67.75z - 5.42)}
\]

(4.9)

where \( u_z \) is the wind velocity (m s\(^{-1}\)) at measuring height \( z \) (m).

4.2.3 Radiation term

The radiation term is represented by:

\[
ET_{\text{rad}} = \frac{\Delta}{\Delta + \gamma^*} \cdot (R_n - G) \cdot \frac{1}{\lambda}
\]

(4.10)

where \( R_n \) is the net radiation (MJ m\(^{-2}\) day\(^{-1}\)); \( G \) the soil heat flux (MJ m\(^{-2}\) day\(^{-1}\)), which is negligible; and \( \lambda \) the latent heat of vaporization, which equals 2.45 MJ kg\(^{-1}\) (at 20°C). The net radiation, \( R_n \), is the difference between the net incoming shortwave radiation, \( R_{ns} \), and the net outgoing longwave radiation, \( R_{nl} \):

\[
R_n = R_{ns} - R_{nl}
\]

(4.11)

The net incoming shortwave radiation is given by:

\[
R_{ns} = (1 - \alpha)R_s
\]

(4.12)

where \( \alpha \) is the albedo, or in other words, the canopy reflection coefficient. The reference crop has an albedo of 0.23; \( R_s \) the incoming solar radiation (MJ m\(^{-2}\) day\(^{-1}\)), which can be measured by radiometers. In most cases, however, it is estimated from measured sunshine duration using the following relationship:

\[
R_s = \left(0.25 + 0.5 \cdot \frac{n}{N}\right)R_a
\]

(4.13)

where \( n \) is the daily bright sunshine duration (hr); \( N \) the daylength (hr); and \( R_a \) the extra terrestrial radiation (MJ m\(^{-2}\) day\(^{-1}\)). The latter is latitude and time of year dependent,
4.3 Growing period

whose value can be looked up, or interpolated, from a published table (Allen et al., 1998, Annex 2). Radiation expressed in mm day$^{-1}$ should be converted to MJ m$^{-2}$ day$^{-1}$ by multiplying with a factor of 2.45, if needed.

The net outgoing longwave (infrared) radiation depends on the black body radiation calculated according to the Stefan-Boltzmann law, the net emissivity of the atmosphere and the degree of cloudiness. The latter two elements act as absorbers of the emitted longwave radiation:

\[ R_{nl} = \sigma \cdot \frac{t_{\text{max}}^4 - t_{\text{min}}^4}{2} \cdot (0.34 - 0.14 \sqrt{e_a}) \cdot \left( 1.35 \frac{R_s}{R_{so}} - 0.35 \right) \]  

(4.14)

where \( \sigma \) is the Stefan-Boltzmann constant (4.903 \times 10^{-9} \text{ MJ m}^{-2} \text{ K}^{-4} \text{ day}^{-1} ), \( t_{\text{max}} \) and \( t_{\text{min}} \) are the maximum and the minimum air temperatures (K) respectively, \( e_a \) the actual vapor pressure (kPa), \( R_s \) the incoming solar radiation (MJ m$^{-2}$ day$^{-1}$) and \( R_{so} \) the solar radiation on clear-sky day (MJ m$^{-2}$ day$^{-1}$), i.e. when \( n = N \).

4.2.4 Annual ET\(_o\) map of China

Monthly ET\(_o\) values of each 5 km \times 5 km grid cell of China were first calculated from the climatic datasets presented in §2.6 using the modified Penman-Monteith approach, then summed to produce the annual total. An annual ET\(_o\) grid was finally generated and shown in Figure E.16 (Appendix E).

Annual ET\(_o\) values range from 303 to 1,705 mm, with low values (less than 1 mm day$^{-1}$) found either in high latitudes (the northeast) or at high altitudes (the Tibetan Plateau), and high values (between 4 and 5 mm day$^{-1}$) either in desert areas (the northwest) or in the southern Himalaya area, bordering eastern India, or in the southeast. At the national scale, the topographical patterns are clearly reflected by the high-low ET\(_o\) values. At the regional or sub-regional scale in areas with large elevation differences, especially in the southwest, the higher ET\(_o\) values, usually found in valleys, gradually decrease up-slope and finally defuse into surrounding low values. These results compare well to a recent monthly ET\(_o\) evaluation in China based on 580 stations using Thornthwaite, pan and Penman-Monteith methods (Chen et al., 2005).

4.3 Growing period

4.3.1 Concept

The growing period can be defined as the period of the year in which agricultural production is possible as a result of adequate moisture availability and absence of temperature limitations. To be more specific, the growing period is a continuous period in the year during which the precipitation is greater than half of the potential evapotranspiration, plus a number of days required to evaporate an assumed 100 mm of soil water, stored at the end of the rains (Kowal, 1978). The growing period for an annual crop is the duration within the year when soil moisture and air temperature permit crop growth and development. A temperature threshold value, for example, 6.5°C, is usually applied in growing period determination and the period with temperature below the threshold value is subtracted from the water availability period.

The concept of the growing period is different from that of the crop (growth) cycle (§3.3). The crop cycle is the duration required for a crop to complete its annual cycle of
establishment, growth and production of harvested part. Perennial crops have crop cycles of more than one year (FAO, 1984). Thus, a crop cycle is a property of the crop (i.e., a crop requirement) whereas a growing period is a condition of the land (i.e., a land quality or characteristic).

Table 4.1: Components of a growing period

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Definition</th>
<th>Relationship</th>
<th>Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>Humid Period</td>
<td>$P &gt; ET_o$</td>
<td>⊂ RP</td>
<td>Humid Season</td>
</tr>
<tr>
<td>RP</td>
<td>Rainy Period</td>
<td>$P &gt; \frac{1}{2} ET_o$</td>
<td>⊂ GP</td>
<td>Rainy Season</td>
</tr>
<tr>
<td>SP</td>
<td>Subhumid Period</td>
<td>Immediately after RP, but before 100 mm soil water evaporated</td>
<td>⊂ GP</td>
<td>Subhumid Season</td>
</tr>
<tr>
<td>DP</td>
<td>Dry Period</td>
<td>$P &lt; \frac{1}{2} ET_o$ but after 100 mm soil water depleted</td>
<td>YR – (RP + SP)</td>
<td>Dry Season</td>
</tr>
<tr>
<td>CP</td>
<td>Cold Period</td>
<td>$T_{mean}$ lower than threshold value</td>
<td>⊈ GP</td>
<td>Cold Season</td>
</tr>
<tr>
<td>GP</td>
<td>Growing Period</td>
<td>Period when moisture and temperature allow crops to grow</td>
<td>RP + SP – CP</td>
<td>Growing Season</td>
</tr>
</tbody>
</table>

† YR: length of year

A growing period may also be called a growing season. But there are subtle differences between them. If there is no ambiguity in the context, a growing season can be interpreted as the synonym of the growing period (Table 4.1). However, a growing season may be the superset of growing periods if discontinuity(-ies) exist(s).

The multi-year mean monthly values of the rainfall, the reference evapotranspiration and the half of it, and the mean air temperature of Chengdu, Southwest China (Figure E.1), are plotted in Figure 4.1 as an example to show the components of a growing period. The critical points, which will be discussed below in growing period determination, are annotated in the figure.

4.3.2 Determination

Decade values of $P$ and $ET_o$ are used in determining the critical dates of the growing period. It is necessary to interpolate the decade values from monthly ones beforehand using following algorithm (Gommes, 1983):

$$
D_1 = \frac{5M_1 + 26M_2 - 4M_3}{N}
$$
$$
D_2 = \frac{-M_1 + 29M_2 - M_3}{N}
$$
$$
D_3 = \frac{-4M_1 + 26M_2 + 5M_3}{N}
$$

(4.15)

where $M_1$, $M_2$ and $M_3$ are the normal values of the climatic parameter in three consecutive months, $D_1$, $D_2$ and $D_3$ the normal values in three consecutive decades of the second month, and $N = 27$ if the climatic parameter is an arithmetic mean (for example, $T$ and $u$), or 81 if the parameter constitutes a sum (for example, $P$, $ET$ and $n$).
4.3 Growing period

Figure 4.1: Growing period determination using multi-year mean climatic data of Chengdu. (a) Relationships between \( P \), \( ET_o \) and \( \frac{1}{2}ET_o \); (b) Mean air temperature and effective temperature threshold. Growing period components of HP, RP, etc., are defined in Table 4.1. Annotations: A: beginning of GP; B1: beginning of HP; B2: end of HP; C: end of RP; and D: end of GP

**Beginning of growing period (A)**

Assuming decade 1 is the last decade where \( P < \frac{1}{2}ET_o \), and decade 2 the first decade where \( P > \frac{1}{2}ET_o \), then the beginning of the growing period (point A in Figure 4.1(a)) is calculated as \( t \) days since mid-decade 1:

\[
t = \frac{P_1 - \frac{1}{2}ET_{o1}}{P_1 - P_2 - \frac{1}{2}(ET_{o1} - ET_{o2})} \cdot 10
\]

(4.16)

where \( P_1 \), \( ET_{o1} \), \( P_2 \) and \( ET_{o2} \) are rainfall and reference evapotranspiration of decade 1 and 2 respectively.
4.3 Growing period

Beginning of humid period (B1)
Assuming decade 1 is the last decade where \( P < ET_o \), and decade 2 the first decade where \( P > ET_o \), then the beginning of the humid period (point B1 in Figure 4.1(a)) is calculated as \( t \) days since mid-decade 1:

\[
t = \frac{P_1 - ET_{o1}}{P_1 - P_2 - (ET_{o1} - ET_{o2})} \cdot 10 \quad (4.17)
\]

End of humid period (B2)
Assuming decade 1 is the last decade where \( P > ET_o \), and decade 2 the first decade where \( P < ET_o \), then the end of the humid period (point B2 in Figure 4.1(a)) is calculated as \( t \) days since mid-decade 1:

\[
t = \frac{P_1 - ET_{o1}}{P_1 - P_2 - \frac{1}{2}(ET_{o1} - ET_{o2})} \cdot 10 \quad (4.18)
\]

End of rainy period (C)
Assuming decade 1 is the last decade where \( P > \frac{1}{2}ET_o \), and decade 2 the first decade where \( P < \frac{1}{2}ET_o \), then the end of the rainy period (point C in Figure 4.1(a)) is calculated as \( t \) days since mid-decade 1:

\[
t = \frac{P_1 - \frac{1}{2}ET_{o1}}{P_1 - P_2 - \frac{1}{2}(ET_{o1} - ET_{o2})} \cdot 10 \quad (4.19)
\]

End of growing period (D)
Assuming decade 0 is the ending decade of the rainy period, and the end of the rainy period (C) is \( r \) days since the beginning of decade 0, decade \( n \) the last decade in which the accumulated \( ET_o \) since C is still less than 100 mm, and decade \( n + 1 \) the first decade in which the accumulated \( ET_o \) exceeds 100 mm, then the end of the growing period (point D in Figure 4.1(a)) is calculated as \( t \) days since C:

\[
t = (10 - r) + 10n + \frac{100 - \left[ (10 - r)ET_{o0} + \sum_{i=1}^{n} ET_{oi} \right]}{ET_{o(n+1)} / 10}
\]

\[
= 10(n + 1) - r + \frac{1000 - \left[ (10 - r)ET_{o0} + 10 \sum_{i=1}^{n} ET_{oi} \right]}{ET_{o(n+1)} / 10} \quad (4.20)
\]

The determination of the end of the growing period is equivalent to that of the duration in which maximum 100 mm soil water is depleted by evaporation since the end of the rainy period, as depicted in Figure 4.2.

Cold period
A temperature threshold of +6.5°C was used in the Chengdu example (Figure 4.1(b)). A cold period existed in the first month of the year, which should be excluded from the growing period. As such, the growing season in Chengdu was finally concluded as 293 days, which was consisted of two periods of 274 (Apr 2-Dec 31) and 19 (Feb 1-19) days (Table 4.2).
4.4 Radiation-thermal production potential

\[ r \overset{C}{\leftarrow} t \overset{D}{\rightarrow} \text{decade} \]

Figure 4.2: *End of the growing period determination*

<table>
<thead>
<tr>
<th>Period</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humid period</td>
<td>Jun 11</td>
<td>Oct 09</td>
<td>121</td>
</tr>
<tr>
<td>Rainy period</td>
<td>Apr 02</td>
<td>Nov 10</td>
<td>223</td>
</tr>
<tr>
<td>Subhumid period</td>
<td>Nov 11</td>
<td>Feb 19</td>
<td>101</td>
</tr>
<tr>
<td>Dry period</td>
<td>Feb 20</td>
<td>Apr 01</td>
<td>41</td>
</tr>
<tr>
<td>Cold period</td>
<td>Jan 01</td>
<td>Jan 31</td>
<td>31</td>
</tr>
<tr>
<td>Growing period</td>
<td>Feb 01</td>
<td>Feb 19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Apr 02</td>
<td>Dec 31</td>
<td>274</td>
</tr>
</tbody>
</table>

Table 4.2: *Growing period in Chengdu, days*

4.3.3 Length of growing period inventory in China

An inventory of the length of the growing period (LGP) in China was made using the climatic datasets obtained with spatial interpolation (§2.6), and is shown in Figure 4.3(a). A vast area including mainly the southeast, but also extending westward to cover most of the Sichuan Basin, as well as northward till the Yangtze River, possesses a growing period of more than 300 days, if it is not year-round. To the other extreme, no growing period could be defined within a wide belt of deserts, high mountains and permafrost plateaus due to the fact that it is either too dry, too cold or both respectively (Figure 4.3b). A LGP-based agro-climatic zoning (FAO, 1996) was conducted in an attempt to produce a control (Figure 4.3c) to the AI-based dryness classification (§2.6.5).

4.4 Radiation-thermal production potential

4.4.1 Net biomass production

The net biomass production \( B_n \) is obtained from the difference between the gross biomass production \( B_g \) and the respiration losses \( R \), all in kg CH\(_2\)O ha\(^{-1}\):

\[ B_n = B_g - R \quad (4.21) \]

The relationship can be equally expressed in terms of the rate or the maximum rate of the processes:

\[ bnm = bgm - rm \quad (4.22) \]

where \( bnm \) is the maximum net biomass production rate, \( bgm \) the maximum gross biomass production rate and \( rm \) the maximum respiration rate, all in kg CH\(_2\)O ha\(^{-1}\) day\(^{-1}\).
4.4 Radiation-thermal production potential

Figure 4.3: Length of growing period inventory (a), analysis of causes for absence of growing period (b) and LGP-based agro-climatic zoning (c)

4.4.2 Respiration

The total respiration consists of two components: growth respiration to keep the photosynthesis process going and maintenance respiration to maintain the accumulated biomass.
4.4 Radiation-thermal production potential

in good shape. The following equation is written in the maximum rate form (FAO, 1984):

\[
\begin{align*}
rm &= 0.28 \, bgm + ct \cdot B_m \\
ct &= c30(0.044 + 0.0019t + 0.001t^2)
\end{align*}
\] (4.23)

where \(ct\) is the respiration coefficient, \(t\) the mean daily temperature (°C), and \(c30 = 0.00283\) for legumes and 0.00108 for non-legumes.

4.4.3 Maximum gross biomass production rate

The maximum gross biomass production rate can be calculated as the sum of the maximum gross biomass production rates on overcast (\(bo\)) and clear (\(bc\)) days:

\[
bgm = f \cdot bo + (1 - f) \cdot bc
\] (4.25)

\[
f = 1 - \frac{n}{N}
\] (4.26)

where \(f\) is the fraction of the daytime that the sky is overcast, \(bo\) and \(bc\) are functions of the latitude and middle of the month in the year. The values of the latter two can be looked up, or interpolated, from published tables (Sys et al., 1991, p.141), but such values are only valid at the maximum leaf photosynthesis rate \((P_m)\) of 20 kg CH\(_2\)O ha\(^{-1}\) hr\(^{-1}\). Corrections have to be made when \(P_m\) is different from 20:

\[
bgm = \begin{cases} 
  f \cdot bo \cdot (1 + 0.002y) + (1 - f) \cdot bc \cdot (1 + 0.005y), & P_m > 20 \\
  f \cdot bo \cdot (1 - 0.025y) + (1 - f) \cdot bc \cdot (1 - 0.01y), & P_m < 20 
\end{cases}
\] (4.27)

where \(y\) is the difference (%) between value of \(P_m\) and 20:

\[
y = \frac{P_m - 20}{20} \cdot 100
\] (4.28)

4.4.4 Total net biomass production

The combination through substitution of relevant equations leads to an expression for the total accumulated net biomass production as the following:

\[
B_n = \frac{0.36 \cdot bgm \cdot KLAI}{L^{-1} + 0.25 \cdot ct}
\] (4.29)

where \(KLAI\) is the maximum growth rate ratio, which is a function of the actual maximum LAI:

\[
KLAI = 0.35 \cdot LAI - 0.03 \cdot LAI^2
\] (4.30)

\(KLAI = 1\) when LAI equals 5 m\(^2\) m\(^{-2}\).

The radiation-thermal production potential (RPP), is finally written as:

\[
RPP = B_n \cdot HI = \frac{0.36 \cdot bgm \cdot (0.35 \cdot LAI - 0.03 \cdot LAI^2) \cdot HI}{L^{-1} + 0.25 \cdot ct}
\] (4.31)

where \(HI\) is the harvest index.
4.5 Water-limited production potential

4.5.1 Maximum crop evapotranspiration

Approaches

The maximum crop evapotranspiration ($ET_m$ or $ET_c$) is the maximal rate that a crop evapotranspires whereas the reference evapotranspiration ($ET_o$) is the maximal rate that the reference crop evapotranspires. $ET_m$ can either be directly calculated through the modified Penman-Monteith approach, which was detailed in §4.2, or through a two-step approach which involves the crop coefficient ($K_c$).

- In the modified Penman-Monteith approach: the factors of albedo ($\alpha$), crop height ($h_c$) and the maximum LAI of the reference crop can be replaced by those of the actual crop in the calculations of the modified psychrometric constant ($\gamma^*$) and the net incoming radiation ($R_{ns}$), which were defined by equations 4.5 and 4.12 respectively, in the following way:

$$\gamma^* = \gamma \left(1 + \frac{r_c}{r_a}\right) = \gamma \left(1 + \frac{200/LAI}{r_a}\right)$$

(4.32)

and

$$r_a = \ln \left(\frac{Z_w - d}{Z_{0w}}\right) \cdot \ln \left(\frac{Z_h - d}{Z_{0h}}\right) \cdot (k^2 u_z)^{-1}$$

(4.33)

where $Z_w$ is the height of the wind velocity measurement (m); $d$ the zero plane displacement of wind profile (m): $d = \frac{2}{3}h_c$ with $h_c$ being the height of the crop; $Z_{0w}$ and $Z_{0h}$ the roughness parameters: $Z_{0w} = 0.123\ h_c$, $Z_{0h} = 0.0123\ h_c$; $Z_h$ the height of the measurement of relative air humidity and air temperature (m); and $k$ von Karman constant: $k = 0.41$.

- In the traditional two-step approach: the maximum crop evapotranspiration is obtained by combining the values of the reference evapotranspiration and the crop coefficient in the following way:

$$ET_c = ET_m$$

$$ = K_c \cdot ET_o$$

(4.34)

The two-step approach was applied in this study.
4.5 Water-limited production potential

Crop coefficient

The crop coefficient \( (K_c) \) accounts for the effect of crop characteristics on crop water requirements: it relates the reference evapotranspiration to the crop evapotranspiration (Doorenbos and Pruitt, 1977). By definition, \( K_c \) relates to the evapotranspiration of a crop grown in large fields, optimally supplied with water and nutrients, free of diseases and achieving full production potential under the given growing environment. \( K_c \) is dimensionless and its value is empirically determined from the ratio \( ET_c/ET_o \). The crop coefficient is affected by crop characteristics, time of planting or sowing, stages of crop development and general climatic conditions such as wind velocity, relative air humidity, frequency of rainfall events, etc.

For most crops, the \( K_c \) value increases from a low value at the time of crop emergence (B) to a maximal value (C) during the period when the crop reaches full development (D), and then declines as the crop matures (E). Three \( K_c \) values are required to construct the \( K_c \) curve for an entire crop cycle: those during the initial (IS), the mid-season (MS) stages and at the time of harvest. The \( K_c \) values during the crop development (CD) and the late season (LS) are interpolated from values gained for other stages, assuming linear relationships. The dynamics of the \( K_c \) values of major food crops, as cultivated in Chengdu, Southwest China, are shown in Figure 4.4.

Typical \( K_c \) values during the IS, MS and at the time of harvest (E) can be found in published tables (Allen et al., 1998, Table 12; referred to as T12 hereafter in this section) for various agricultural crops. Such values represent average \( K_c \) values for non-stressed, well managed crops in sub-humid climates with an average daytime minimum relative humidity \( (RH_{min}) \) of about 45% and with moderate wind speeds averaging 2 m s\(^{-1}\). Modifications should be made if the actual climatic conditions are different from this.

The \( RH_{min} \) is calculated on a daily or average monthly basis as:

\[
RH_{min} = \frac{e_s(T_{dew})}{e_s(T_{max})} \cdot 100 \tag{4.35}
\]

where \( T_{dew} \) is mean dew-point temperature, \( T_{max} \) mean daily maximum air temperature, \( e_s(T) \) the saturated vapor pressure at temperature \( T \) (Equation 4.7). In case the dew-point temperature is not available or of questionable quality, it can be approximated by the mean daily minimum air temperature, \( T_{min} \):

\[
T_{dew} = \begin{cases} 
T_{min} - 2, & \text{(arid and semiarid climates)} \\
T_{min}, & \text{(other climates)} 
\end{cases} \tag{4.36}
\]

The crop coefficient for the initial stage can be derived from published figures (Allen et al., 1998, Figures 29 and 30; referred to as F29 and F30 respectively hereafter in this section), which provide estimates of \( K_c \) as a function of the average interval between wetting events, the evaporation power \( ET_o \), and the importance of the wetting event. F29 is used for all soil types when wetting events are light, i.e., the infiltration depths are of 10 mm or less; F30 is used for heavy wetting events when infiltration depths are greater than 40 mm, with a distinction between coarse textured and finer soils. When average infiltration depth is between 10 and 40 mm, the initial stage \( K_c \) value can be estimated using both of the figures:

\[
K_c\text{ ini} = K_c\text{ ini(F29)} + \frac{I - 10}{40 - 10} \left[K_c\text{ ini(F30)} - K_c\text{ ini(F29)}\right] \tag{4.37}
\]
Figure 4.4: $K_c$ values of local cultivars of early (a), intermediate (b), late rice (c), winter wheat (d), summer maize (e) and sorghum (f) in Chengdu, Southwest China. Annotations in (a): IS: initial, CD: crop development, MS: mid-season and LS: late season stages; A: sowing/planting date, B: start of CD, C: start of MS, D: start of LS and E: time of harvest.

where $K_{c \text{ ini}}$ is the $K_c$ value for the initial stage, $K_{c \text{ ini}(F29)}$ and $K_{c \text{ ini}(F30)}$ are $K_c$ values interpolated from F29 and F30 respectively, $I$ is the average infiltration depth (mm).

For the mid-season stage and time of harvest, tabulated $K_c$ values need to be adjusted for climates where $RH_{\text{min}}$ differs from 45% or $u_2$ is larger or smaller than 2.0 m s$^{-1}$ as the following:

$$K_{c \text{ adj}} = K_{c(T12)} + [0.04(u_2 - 2) - 0.004(RH_{\text{min}} - 45)] \left( \frac{h}{3} \right)^{0.3} \quad (4.38)$$
4.5 Water-limited production potential

where $K_{c\text{ adj}}$ is the adjusted $K_c$ value for mid-season or time of harvest; $K_c(T12)$ is the $K_c$ value taken from T12; $u_2$ mean daily wind speed at 2 m height over grass during the mid-season or at time of harvest (m s$^{-1}$), for 1 m s$^{-1}$ $\leq u_2$ $\leq$ 6 m s$^{-1}$; $RH_{min}$ mean daily minimum relative humidity during mid-season or time of harvest, for 20% $\leq RH$ $\leq$ 80%; and $h$ mean plant height (m) during mid-season or time of harvest (T12), for 0.1 m $\leq h$ $\leq$ 10 m.

Crop coefficient and maximum crop ET of food crops in China

The average $K_c$ and $ET_m$ values of food crops in China are shown in Figure E.20 (Appendix E). Food crops in most of the agricultural areas except South China (in green color on map) may evapotranspire at a rate between 75% to 100% of the reference evapotranspiration, should the soil moisture condition permit this. Under the same water conditions, however, the food crops in South China (in blue color) may evapotranspire at a rate higher than the reference evapotranspiration. In other words, the maximum evapotranspiration of food crops in North China and in the south-western highlands (in green color) ranges mostly between 2-3 mm day$^{-1}$; the $ET_m$ of food crops in Northeast China and in majority of the agricultural areas south of the Yangtze River (in yellow color) is between 3-4 mm day$^{-1}$. It is worth mentioning that the food crops in the northwest (in purple color) may evapotranspire at a maximal rate of 4-5 mm day$^{-1}$.

4.5.2 Plant available soil water capacity

Effective rooting depth dynamics

In order to take into account a changing rooting depth during the crop cycle, a linear increase of the rooting depth can be assumed from the crop emergence up to effective full ground cover (i.e., beginning of the MS stage). From this point onwards, the depth of the root remains at its maximal value. Indicative values of optimal rooting depth for different crops can be found from published tables (Sys et al., 1991, p.254). But such values have to be adapted to local conditions based on the rooting performance of the cultivars. Special attention needs to be paid to the presence of any root restricting layers in the soil, for example, a continuous stone layer, a fragipan or any other compacted layer or horizon that restricts root penetration.

The rooting depth dynamics of winter wheat, summer maize, intermediate rice and sorghum as sown in Chengdu of Southwest China are shown in Figure 4.5.

Plant available soil water

Water is held in the fine pores of the soil as a function of the relationship between the upward matric force and the downward gravitational force. Water is retained in the soil when the former is greater than the latter. The roots of the plant first have to overcome the retention force of the soil to water molecules before they can take up. At a tension of 1.5 MPa (pF 4.2), the water is tightly retained in the soil and not available to the plants. This is called the permanent wilting point (PWP) which draws a boundary between the available and the unavailable soil water. Soil holds the maximum amount of water at field capacity (FC) under a tension of 10-20 kPa (pF 2-2.3). In practice, field capacity is the moisture retention capacity 24-48 hours after a heavy rain or an irrigation event with free drainage.
The moisture content between field capacity and the permanent wilting point is defined as the plant available soil water (PAW) which is also called total available soil water ($S_a$).

\[
PAW = S_a = FC - PWP
\]  

(4.39)

where PAW (or $S_a$) is the plant available soil water (% v/v), $FC$ and $PWP$ are the moisture contents (% v/v) at field capacity and the permanent wilting point respectively.

The soil moisture content in volumetric percentage can also be expressed in terms of water depth over soil depth (mm m$^{-1}$):

\[
1 \, \text{% v/v} = 1 \, \text{cm m}^{-1} = 10 \, \text{mm m}^{-1}
\]  

(4.40)

Therefore the plant available soil water capacity is given by:

\[
S_aD = 10 \, (FC - PWP) \cdot d
\]  

(4.41)

where $S_aD$ is the plant available soil water capacity (mm), $FC$ and $PWP$ are the soil moisture contents (% v/v) at field capacity and the permanent wilting point, respectively, and $d$ is the effective rooting depth (m).

The total available soil water content of the cropland, the average rooting depth and the available soil water capacity for food crops in 2005 are shown in Figure E.18 (Appendix E).
4.5 Water-limited production potential

Easily versus scarcely available soil water capacities

The amount of energy that a crop consumes for water uptake is inversely proportional to the soil moisture content within the FC-PWP range (Cowan, 1965; Guswa et al., 2002). The crop consumes the minimum energy when the soil moisture content is at FC and the maximum energy when the soil moisture content is at PWP. Crop transpiration and photosynthesis are both adversely affected when the moisture content drops below a threshold value (van den Boogaard et al., 1996; Zhang and Oweis, 1999) which is defined as the soil water depletion fraction \( p \). The part of the total available soil water \( (S_a) \) that can be depleted without causing water stress is therefore determined as the *easily* available soil water \( (pS_a) \), and consequently \((1 - p)S_a\) is defined as the *scarcely* available soil water.

Expressed in terms of water depth:

\[
pS_a D = p \cdot (FC - PWP) \cdot d
\]

\[
(1 - p)S_a D = (1 - p) \cdot (FC - PWP) \cdot d
\]

where \( pS_a D \) is the easily available soil water capacity (mm), \((1 - p)S_a D \) is the scarcely available soil water capacity (mm) and \( d \) is the effective rooting depth (m).

Values of the \( p \) factor can be found in Allen et al. (1998, Table 22; referred to as T22 hereafter in this section). The factor \( p \) differs from one crop to another. Its value varies from 0.30 for shallow rooted plants at high rates of \( ET_c \) (> 8 mm d\(^{-1}\)) to 0.70 for deep rooted plants at low rates of \( ET_c \) (< 3 mm d\(^{-1}\)). A value of 0.50 for \( p \) is commonly used for many crops.

The fraction \( p \) is a function of the evaporation power of the atmosphere. At low rates of \( ET_c \), the \( p \) values listed in T22 are higher than at high rates of \( ET_c \). For hot dry weather conditions, where \( ET_c \) is high, \( p \) is 10-25% less than the values presented in T22, and the soil is still relatively wet when the stress starts to occur. The fraction \( p \) is up to 20% higher than the listed values when \( ET_c \) is low. The tabulated \( p \) values, \( p(T22) \), can be adjusted using the following equation:

\[
p = p(T22) + 0.04(5 - ET_c)
\]

where the adjusted \( p \) is limited to 0.1 \( \leq p \leq 0.8 \) and \( ET_c \) is in mm day\(^{-1}\).

As a certain soil matric potential corresponds in different soil types with different soil water contents, the value for \( p \) is also a function of the soil type. The \( p \) values listed in T22 can be reduced by 5-10% for fine textured soils. But for more coarse textured soils, the tabulated \( p \) values can be increased by 5-10%.

Soil water depletion fraction of food crops in China

The average \( p \) value, expressed as the \( pS_a D : S_a D \) ratio (%), of food crops in 2005 is shown in Figure 4.6. The food crops on most of the cropland (dark-gray shading on map) in China can *easily* take up 50-75% of the total plant available water storage. The percentage is increased beyond 75% for areas, represented by black shading on map, in the northeast, the north and the Shandong peninsula.

4.5.3 Effective rainfall

Effective rainfall is defined as the portion of total annual or seasonal rainfall which is useful directly and/or indirectly for crop production at the site where it falls but without
pumping. It therefore includes (Doorenbos and Pruitt, 1977; Dastane, 1978; Allen et al., 1998):

1. water intercepted by living or dry vegetation;
2. water lost by evaporation from the soil surface;
3. the precipitation lost by evapotranspiration during growth; and
4. that fraction which contributes to leaching and percolation.

Numerous methods for estimating effective rainfall have been proposed in the past including direct measurement techniques, empirical methods (e.g. the fixed percentage method, the FAO/AGLW formula and the USBR method (Smith, 1988), and the USDA Soil Conservation Service method (Dastane, 1978)) and soil water balance methods. The best estimates of effective rainfall can be obtained by conducting soil water balance computations (Patwardhan et al., 1990). There are also methods that use relatively simple procedures calibrated for a specific area and/or crop. The following methods, for example, are all specialized in estimating effective rainfall for the rice crop in East and South-east Asia (Mohan et al., 1996): evaporation-rainfall ratio method, Indian-1 method, Indian-2 method and the Vietnam method.
4.5 Water-limited production potential

The performance of the USDA Soil Conservation Service method (“USDA-SCS” hereafter) was found comparable to that of a physical water balance method (Patwardhan et al., 1990) and is usually used as a control in methodological benchmark tests (Mohan et al., 1996). It was therefore adopted as the effective rainfall estimator in this research.

Effective rainfall ($P_{\text{eff}}$) is computed from the monthly total rainfall ($P_{\text{tot}}$) and the monthly consumptive use ($ET_c$):

$$P_{\text{eff}} = \left( \frac{0.025}{ET_c} - 0.001 \right) P_{\text{tot}}^2 + (0.0016 \ ET_c + 0.6) \ P_{\text{tot}}$$ (4.45)

The obtained value is valid for an available soil moisture capacity at the root zone of 75 mm. For other values of available soil moisture capacity a correction (Sys et al., 1991, p.183) is required.

The rainfall effectiveness coefficient, $RE$, is subsequently computed as the ratio (%) of the effective rainfall over the total rainfall:

$$RE = \frac{P_{\text{eff}}}{P_{\text{tot}}} \cdot 100$$ (4.46)

Rainfall effectiveness for food crops in China

The effective rainfall was evaluated on a per crop basis using the USDA-SCS method. A combined rainfall effectiveness map for all food crops was generated based on the cropping system data (Figure E.19 in Appendix E). The map shows that the rainfall is more effective in a wide belt stretching from the northeast to the southwest and less effective in the southeast, whereas the highly effective rains ($RE > 80\%$) are falling in the outskirts of the Gobi Desert in North China and of the Takla Makan Desert in the northwest. This pattern can be explained by the relationships between the rainfall effectiveness and the total rainfall as well as the crop evapotranspiration (Figure 4.7).

Figure 4.7(a) reveals that the rainfall effectiveness is mostly proportional to the crop evapotranspiration with minor exceptions when the latter is less than 45 mm month$^{-1}$ or 1.5 mm day$^{-1}$ and the rainfall is greater than 120 mm month$^{-1}$. It also reveals that the rainfall effectiveness is inversely proportional to the total rainfall. Such simulated relationships are confirmed by observed rainfalls (Figure 4.7(b)) and $ET_c$ values (Figure 4.7(c)) evaluated using a water balance approach (§4.5.4).

4.5.4 Actual crop evapotranspiration

Actual crop evapotranspiration ($ET_a$) is the amount of water that is actually removed from the soil due to the processes of evaporation and transpiration. The crop evapotranspires at the maximal rate ($ET_m$ or $ET_c$) if the soil water storage over the root zone at time $t$ ($S_tD$) is higher than the scarcely available soil water capacity; it evapotranspires at a reduced rate ($ET_a$) when water stress occurs. The magnitude of $ET_a$ depends on the remaining soil water storage ($S_tD$) relative to the amount of the scarcely available water capacity and on $ET_m$. Since evapotranspiration implies a decrease of the actually available soil moisture over the rooting zone per unit time $t$ (Rijtema and Aboukhaled, 1975):

$$ET_a = \frac{S_tD}{(1-p)S_aD} \cdot ET_m$$

$$= - \frac{d(S_tD)}{dt}$$ (4.47)
Figure 4.7: Rainfall effectiveness in response (a) to crop evapotranspiration and total rainfall at $pS_a D$ of 75 mm, as simulated using Equation 4.45; (b) to observed rainfall; and (c) to evaluated crop evapotranspiration using a 2-step approach

where $S_t D < (1 - p) S_a D$.

The above equation can be rearranged into the following equation (Debaveye, 1986):

$$ET_a = \left[ 1 - e^{-\frac{ET_c}{(1-p)S_a D}} \right] \cdot S_t D$$  \hspace{1cm} (4.48)

where $S_t D$ is the available soil water storage at time $t$ over the rooting depth $D$ and $S_a D$ is the total available soil water capacity over the rooting depth $D$. This is the expression for the accumulated actual evapotranspiration ($ET_a$) during a period of $t$ days after a fraction ($p$) of the total available soil moisture ($S_a$) over the rooting depth ($D$) has been depleted. The actual available soil moisture content at the beginning of the period is $S_t D$ and the potential crop evapotranspiration during the period is $ET_c$.

The relationships among plant available soil water, soil water depletion fraction $p$ and actual crop evapotranspiration rate are shown in Figure 4.8.

**Actual crop ET of food crops in China**

The average actual evapotranspiration, in comparison with $K_c$ and $ET_m$, of food crops in China in the year 2005 is shown in Figure E.20 (Appendix E). The $ET_a$ values of food crops in the northeast and the north (in blue color on map) mostly vary between 1 and 2 mm day$^{-1}$, decreases from $ET_m$ values of 2-3 mm day$^{-1}$ for the north and 3-4 mm day$^{-1}$ for the northeast. In the arid northwest, although the $ET_m$ may be as high as 4-5 mm day$^{-1}$, the food crops there actually evapotranspire at a rate less than 1 mm day$^{-1}$.

**Soil water balance simulation**

Assuming:

1. No water stored in the beginning of the initial decade: $ST_{i,0} = 0$;
2. All rains fall in the beginning of a decade;
3. Soil water storage and effective rainfall are sufficient to allow crop to evapotranspire at the maximal rate ($ET_m$);
then the provisional soil water storage \((ST_p)\) at the end of decade \(i\) (after maximum evapotranspiration) is:

\[
ST_p = ST_i + P_{eff} - ET_m
\]  

(4.49)

where \(ST_i\) is the soil water storage in the beginning of the decade, \(P_{eff}\) is effective rainfall.

If the provisional storage is greater than the scarcely available water capacity, the crop evapotranspires at the maximal rate; otherwise it evapotranspires at a reduced rate:

\[
\begin{align*}
\begin{cases}
ET_a = ET_m & (ST_p > (1-p)SaD) \\
ST_e = ST_p & (4.50) \\
ET_a < ET_m & (ST_p \leq (1-p)SaD) \\
ST_e \neq ST_p & (4.51)
\end{cases}
\end{align*}
\]

where \(ST_e\) is the soil water storage at the end of the decade.

For situation (1), \(ET_a\) and \(ST_e\) are both known. The simulation proceeds to the next decade at step 7 below; for situation (2), extra steps are needed to obtain \(ET_a\) and \(ST_e\).

1. The number of days \(t\) that the crop can evapotranspire at the maximal rate:

\[
t = \frac{ST_i + P_{eff} - (1-p)SaD}{ET_m/10}
\]  

(4.52)

2. Actual accumulated maximal evapotranspiration during \(t\) days:

\[
ET_i = t \cdot \frac{ET_m}{10}
\]  

(4.53)
3. Water storage after $t$ days of maximal evapotranspiration:

$$ST_t = ST_i + P_{eff} - ET_t$$ (4.54)

4. Actual accumulated evapotranspiration during the remaining $k = 10 - t$ days:

$$ET_k = \left[1 - e^{-\frac{k\cdot ET_m}{10(1-p)S_a D}}\right] \cdot ST_t$$ (4.55)

5. Sum of the actual evapotranspiration of the decade:

$$ET_a = ET_t + ET_k$$ (4.56)

6. Soil water storage at the end of the decade:

$$ST_e = ST_i + P_{eff} - ET_a$$ (4.57)

7. Initial soil water storage of the next decade:

$$ST_{i(n+1)} = \begin{cases} 
ST_e(n), & ST_e(n) < S_a D_{(n+1)} \\
S_a D_{(n+1)}, & ST_e(n) \geq S_a D_{(n+1)} 
\end{cases}$$ (4.58)

where $n$ is the current decade and $n+1$ is the next decade. The initial water storage of decade $n + 1$ is guaranteed not exceeding the total plant available water capacity of the decade.

The procedure should be looped to traverse every decade within the crop cycle.

### 4.5.5 Anticipated rainfed crop yield

As aforementioned (§4.5.2), the crop transpiration and leaf photosynthesis are both adversely affected when actual soil water storage decreases below a threshold capacity defined by the scarcely available soil water fraction $(1-p)S_a D$. Yield reduction occurs accordingly if water stress develops during the crop cycle. The extent of yield reduction depends not only on the severity of water deficit but also on the crop’s sensitivity to it.

The aggregate effect of water stress on crop yield ($f_W$) can be evaluated from the relative evapotranspiration and the yield response factor ($k_y$); and the anticipated rainfed yield is then obtained as a product of the $f_W$ and the RPP.

$$WP = f_W \cdot RPP$$ (4.59)

$$\begin{align*}
W_P &= f_W \cdot R_{PP} \\
W_P &= 1 - k_y \left(1 - \frac{ET_a}{ET_m}\right)
\end{align*}$$ (4.60)

where $f_W$ is the yield reduction coefficient due to water stress, $k_y$ is the yield response factor.
Yield reduction coefficient due to water stress

The effect of water stress on crop growth and yield depends on crop species as well as the time and intensity of water stress occurrence during the crop cycle. In general, crops are more sensitive during the emergence, flowering and early yield formation periods and less sensitive during the vegetative and ripening periods. \( k_y \) values of popular crops, per crop growth period (CGP, §3.3.2), can be found from published tables (Sys et al., 1991, p.261). In practice, CGP-specific \( k_y \) values should be used in Equation 4.60 together with corresponding relative evapotranspiration deficit ratios to derive a single \( f_W \) for the entire crop cycle:

\[
f_W = \sum_{i=1}^{n} \left\{ \frac{L_i}{L} \left[ 1 - k_y(i) \left( 1 - \frac{ET_a(i)}{ET_m(i)} \right) \right] \right\}
\]

where \( i \) is the serial number of a CGP, \( n \) is the total number of CGPs, \( L_i \) is the length (day) of the \( i \)th CGP, \( L \) is the length (day) of the crop cycle, \( ET_a(i) \) is the actual crop evapotranspiration of the \( i \)th CGP, \( ET_m(i) \) is the maximal crop evapotranspiration of the \( i \)th CGP and \( k_y(i) \) is the yield response factor of the \( i \)th CGP.

Crop ET deficit, yield reduction and the WPP of food crops in China

The average values of (a) relative crop evapotranspiration deficit ratio, (b) yield response factor, (c) yield reduction coefficient due to water stress and (d) water-limited crop production potential of food crops in China in the year 2005 are shown side by side in Figure E.21 (Appendix E). As one of the four great basins in China, the Sichuan Basin (SB) is of great importance in grain production in China (Figure E.1) and thus taken as an example. The evapotranspiration deficit ratio, \( (ET_m - ET_a)/ET_m \), of food crops is averaged 20-40\% in SB. With an average yield response factor being greater than 1, the yield loss due to water stress is assessed as 40-60\%. The average potential yield is decreased from 7-8 ton ha\(^{-1}\) (RPP, Figure E.17) to 4-5 ton ha\(^{-1}\) (WPP).

4.6 Land production potential

The land production potential (LPP) is the last step in production potential determinations. It is the closest approximation of the realistic crop yield. Following the same philosophy that was used in obtaining the WPP from the RPP, the LPP is achieved by combining the soil index, accounting for the yield-reduction effects of limited soil fertility, and the management index, correlated to the management level in which crop production is carried out, with the WPP:

\[
LPP = S_y \cdot M_y \cdot WPP
\]

where \( S_y \) is the soil index and \( M_y \) is the management index.

4.6.1 Soil index

All soil parameters that have a direct impact on crop performance and had not been taken into account in WPP determination need to be assessed in order to derive the soil index. Soil parameters are divided into two groups, one for soils with fertility problems (or regions with an aridity index (§2.6.5) of 0.5 or greater); and the other for soils with salinity problems (or regions with an aridity index smaller than 0.50):
For humid climates ($AI \geq 0.5$) or soils with fertility problems:

- CEC-clay;
- SOC;
- the more limiting value of pH-H$_2$O and sum of exchangeable Ca$^{2+}$, Mg$^{2+}$ and K$^+$;

For arid climates ($AI < 0.5$) or soils with salinity problems:

- CaCO$_3$;
- gypsum;
- the more limiting value of the electrical conductivity (ECe) and the exchangeable sodium percentage (ESP).

Soil limitations evaluation

Crop-specific soil requirement tables (Sys et al., 1993) are used to obtain rating values correlated with the levels of severity of soil limitations on crop yield. Rating values of selected soil parameters are combined using the following formula (Storie, 1976) to derive the soil index:

$$S_y = \prod_{i=1}^{n} \left( \frac{R_i}{100} \right)$$

(4.63)

where $i$ is the serial number of a soil parameter, $n$ is the total number of selected soil parameters and $R_i$ is the rating value (0-100) of soil parameter $i$.

The soil requirements of summer maize, for example, are given in Table 4.3. Some data such as CEC-clay and ESP can be used as they are indicated in a soil analytical report, others have to be recalculated over a certain soil depth (sum of basic cations, pH-H$_2$O and SOC), sometimes by using weighting factors for different profile sections (CaCO$_3$, gypsum and ECe). In the latter case, a soil profile is subdivided into equal sections and to each section a weighting factor is attributed. Shallower sections have higher weights whereas deeper ones have lower weights (Table 4.4).

Values of limiting soil parameters are calculated in the following way:

- CaCO$_3$
  Weighted mean over the rooting depth using weighting factors for different profile sections given in Table 4.4.

- Gypsum
  Weighted mean over the rooting depth using weighting factors for different profile sections given in Table 4.4.

- CEC-clay
  CEC of the clay fraction in the B-horizon or at a depth of 50 cm in an A-C profile, without correction for organic matter.

- Sum of basic cations
  Weighted mean of the sum of the exchangeable Ca$^{2+}$, Mg$^{2+}$ and K$^+$ for the upper 25 cm of the mineral soil.
Table 4.3: Soil requirement table of summer maize

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rating</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CaCO₃ (%)</td>
<td>100</td>
<td>95</td>
<td>85</td>
<td>60</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>gypsum (%)</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>CEC-clay a</td>
<td>&gt;24</td>
<td>24</td>
<td>16</td>
<td>&lt;16(-)</td>
<td>&lt;16(+)</td>
<td>-</td>
</tr>
<tr>
<td>BS b (%)</td>
<td>&gt;80</td>
<td>80</td>
<td>50</td>
<td>35</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>SumBC c</td>
<td>&gt;8</td>
<td>8</td>
<td>5</td>
<td>3.5</td>
<td>2</td>
<td>&lt;2</td>
</tr>
<tr>
<td>pH-H₂O</td>
<td>6.6</td>
<td>6.2</td>
<td>5.8</td>
<td>5.5</td>
<td>5.2</td>
<td>&lt;5.2</td>
</tr>
<tr>
<td>SOC (%)</td>
<td>&gt;2.0</td>
<td>2.0</td>
<td>1.2</td>
<td>0.8</td>
<td>&lt;0.8</td>
<td>-</td>
</tr>
<tr>
<td>ECe (dS m⁻¹)</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>ESP (%)</td>
<td>0</td>
<td>8</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>-</td>
</tr>
</tbody>
</table>

* Parent material type: (1) kaolinitic; (2) non-kaolinitic & non-calcareous; (3) calcareous
- not applicable

a cmol(+) kg⁻¹ clay
b Base saturation
c Sum of exchangeable Ca²⁺, Mg²⁺ and K⁺: cmol(+) kg⁻¹ soil

Table 4.4: Number of equal sections and weighting factors for each section

<table>
<thead>
<tr>
<th>Depth Range(cm)</th>
<th>Sections</th>
<th>Weighting Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>125-150</td>
<td>6</td>
<td>2.00-1.50-1.00-0.75-0.50-0.25</td>
</tr>
<tr>
<td>100-125</td>
<td>5</td>
<td>1.75-1.50-1.00-0.50-0.25</td>
</tr>
<tr>
<td>75-100</td>
<td>4</td>
<td>1.75-1.25-0.75-0.25</td>
</tr>
<tr>
<td>50-75</td>
<td>3</td>
<td>1.50-1.00-0.50</td>
</tr>
<tr>
<td>25-50</td>
<td>2</td>
<td>1.25-0.75</td>
</tr>
<tr>
<td>0-25</td>
<td>1</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- Acidity (pH-H₂O)
  Weighted mean of the upper 25 cm.

- SOC
  Weighted mean of the upper 25 cm.

- ECe
  Weighted mean over the rooting depth using weighting factors for different profile sections given in Table 4.4.

- ESP
  The highest ESP value from soil horizons within a depth of 100 cm.
4.6 Land production potential

Soil limitations in food production in China

The rating values of the limiting soil parameters were obtained per crop (§3.4) per virtual soil profile (§2.7.4) for food production in China. The soil parameter with the lowest rating value was identified as the most limiting one and is shown in Figure E.22 (Appendix E). The soil index was computed from the selected limiting soil parameters (Figure E.23). The former indicates where the most efficient soil improvements can be made, while the latter suggests that soils in the north are in general more fertile, than soils in the south, for food production in China.

4.6.2 Management index

The management index is used to assess the effects of field management practices on crop performance using a crop-specific approach on the basis of input levels which are correlated to the levels of management practices.

The theory of production

One of the eternal problems of the theory of production is how to explain the growth of the output in terms of materials used. In agriculture, increasing concerns are raised about the relationship between the total production of food, in particular, and the levels of factor inputs. This is partly due to the more intensive use of fertilizers, chemicals, gasoline, electricity and so forth since the Green Revolution (Conway, 1999). The conventional approach to address the input-output relationship, classically exposed by Cobb and Douglas (1928), is to consider output $Y$ to be determined by input $I$:

$$ Y = A \cdot \prod_{i=1}^{n} I_i^{a_i} \quad (4.64) $$

where $I_i$ is the $i^{th}$ factor input, $A$ and $a_i$ are coefficients.

Equation 4.64 is the typical form of the Cobb-Douglas production function (Douglas, 1976; Pokrovski, 2003). It has been well applied in many fields including the agricultural input-output analysis (Canakci et al., 2005; Hatirli et al., 2006; Esengun et al., 2007). The function can further be expressed in the following form:

$$ \ln Y = \ln (A) + a_1 \cdot \ln (I_1) + \ldots + a_i \cdot \ln (I_i) + \ldots + a_n \cdot \ln (I_n) $$

$$ = a_0 + \sum_{i=1}^{n} [a_i \cdot \ln (I_i)] \quad (4.65) $$

where coefficient $a_0 = \ln (A)$.

The total number of coefficients in Equation 4.65 is $n + 1$. The coefficients can be statistically estimated using a multiple linear regression model.

A single factor variant of the production function is denoted as:

$$ Y = a \cdot I^b \quad (4.66) $$

where $a$ and $b$ are coefficients.
Relationship between the food productivity and the factor inputs in China

The average yield of food crops in a time series of 1978-2005 (NBSC, 1999, 2006) was cross analyzed against the effective irrigation rate (refer to §6.2 for definition), the application of nitrate, phosphorus, potassium and compound fertilizers, as well as the electricity consumption, both individually and collectively, in an attempt to reveal the input-output relationship in food production in China. The shape of the best fitting curves in Figure 4.9 suggests that a single factor production function is applicable in modeling the relationship between the food productivity and the factor inputs. Regression analysis was then conducted and production functions were successfully obtained at the highest significance levels (Table 4.5).

The time series data of yields and factor inputs are documented in Appendix D.

Figure 4.9: Relationships between the food productivity and the input factors of effective irrigation rate (a), applications of the nitrate (b), phosphorus (c), potassium (d) and compound (e) fertilizers, and the electricity consumption (f)
4.6 Land production potential

Table 4.5: Single factor production functions with the food productivity (t ha$^{-1}$) as the dependent and the factor input as the independent variable

<table>
<thead>
<tr>
<th>Input Variable</th>
<th>a</th>
<th>b</th>
<th>$R^2$</th>
<th>$p$-value</th>
<th>Significance†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Irrigation rate (%)</td>
<td>0.06</td>
<td>1.17</td>
<td>0.86</td>
<td>$5.17 \times 10^{-10}$</td>
<td>***</td>
</tr>
<tr>
<td>Nitrate fertilizers (kg ha$^{-1}$)</td>
<td>0.36</td>
<td>0.51</td>
<td>0.97</td>
<td>$2.20 \times 10^{-16}$</td>
<td>***</td>
</tr>
<tr>
<td>Phosphorus fertilizers (kg ha$^{-1}$)</td>
<td>0.87</td>
<td>0.43</td>
<td>0.94</td>
<td>$5.60 \times 10^{-14}$</td>
<td>***</td>
</tr>
<tr>
<td>Potassium fertilizers (kg ha$^{-1}$)</td>
<td>2.45</td>
<td>0.19</td>
<td>0.95</td>
<td>$3.12 \times 10^{-14}$</td>
<td>***</td>
</tr>
<tr>
<td>Compound fertilizers (kg ha$^{-1}$)</td>
<td>2.46</td>
<td>0.14</td>
<td>0.97</td>
<td>$2.20 \times 10^{-16}$</td>
<td>***</td>
</tr>
<tr>
<td>Electricity (kWh ha$^{-1}$)</td>
<td>1.04</td>
<td>0.20</td>
<td>0.86</td>
<td>$4.76 \times 10^{-10}$</td>
<td>***</td>
</tr>
</tbody>
</table>

† Significance levels: ≥ 0.1(•); > 0.05(•); ≤ 0.05(*); ≤ 0.01(**); ≤ 0.001(***)

Overall score of factor inputs

An overall score is obtained to represent all the input variables as a whole using the following formula:

$$S_i = \sum_{j=1}^{n} \frac{I_{ij}}{\min(I_j)}$$  (4.67)

where $S_i$ is the overall score of all input variables in the $i^{th}$ year, $I_{ij}$ is the absolute value of the $j^{th}$ input variable in the $i^{th}$ year, $I_j$ is the time series of the $j^{th}$ input variable, and “min” denotes the minimum function.

The overall score of the following three categories of factor inputs during 1978-2005 is shown as a bar chart in Figure 4.10(a): (1) application of fertilizers and chemicals, (2) agro-machinery usage and electrical consumption, and (3) irrigation infrastructure investment. The overall score is plotted against the average yield of food crops as a scatter plot in Figure 4.10(b). The shape of the best fitting curve in the latter suggests that the Cobb-Douglas production function is applicable in modeling the yield by using the overall score alone.

The following production function is thus obtained ($R^2 = 0.86$ and $p$-value = $1.95 \times 10^{-10}$):

$$Y = 2.10 \cdot S^{0.28}$$  (4.68)

where $Y$ is the food productivity in t ha$^{-1}$ and $S$ is the overall score of factor inputs.

Characterization of input levels

Based on the value ranges of the overall scores ($S$) correlated to the east-middle-west social-economic development belts (§2.2.3), three distinct input levels, namely the high, intermediate and low input levels, were established. The overall score of 3 with a value range of $0 \leq S < 4.5$ was defined as the low input level, while the overall scores of 6 with a range of $4.5 \leq S < 9$ and 12 with a range of $S \geq 9$ were defined as the intermediate and the high input levels, respectively. For a complete discussion of input levels based on overall scores, please refer to §6.3.
4.6 Land production potential

Figure 4.10: Evolution of the overall score of factor inputs (a) and the relationship between the average yield of food crops and the overall score of factor inputs (b)

Management index of food crops

The following production function, generalized from Equation 4.68, can be used to explain the yield \( Y \) of a food crop in terms of the overall score \( S \) of factor inputs:

\[
Y = a \cdot S^b
\]  

(4.69)

where \( a \) and \( b \) are coefficients.

Given \( Y_1, Y_2 \) and \( Y_3 \) as the yields of a particular food crop achieved at the high, intermediate and low input levels represented by the overall scores of \( S_1, S_2 \) and \( S_3 \), respectively:

\[
Y_1 = a \cdot S_1^b
\]  

(4.70)

\[
Y_2 = a \cdot S_2^b
\]  

(4.71)

\[
Y_3 = a \cdot S_3^b
\]  

(4.72)

where coefficients \( a \) and \( b \) are crop-specific.

The management index at the high, intermediate and low input levels, \( M_{y1}, M_{y2} \) and \( M_{y3} \), are obtained as the yield at the current input level over the yield at the high input level:

\[
M_{y1} = \frac{Y_1}{Y_1} = \frac{a \cdot S_1^b}{a \cdot S_1^b} = 1
\]  

(4.73)

\[
M_{y2} = \frac{Y_2}{Y_1} = \frac{a \cdot S_2^b}{a \cdot S_1^b} = \left(\frac{S_2}{S_1}\right)^b
\]  

(4.74)

\[
M_{y3} = \frac{Y_3}{Y_1} = \frac{a \cdot S_3^b}{a \cdot S_1^b} = \left(\frac{S_3}{S_1}\right)^b
\]  

(4.75)

The management index is therefore observed to be dependent on (1) the ratio of the overall score at the current input level over the overall score at the high input level, and
(2) the coefficient $b$ of production function 4.69. The management index is not, however, dependent on the coefficient $a$ of the same production function. Taking the relationship between the wheat yield and the overall score of factor inputs plotted on the logarithmic-logarithmic scales as an example (Figure 4.11), any change to the slope $b$ of the best-fit line will change the value of the management index of the wheat crop, but changes to the inception $a$ of the best-fit line will have no impact on the value of the management index.

Figure 4.11: Wheat yield in response to the overall score of factor inputs, plotted on the logarithmic-logarithmic scales to demonstrate the determination and the meaning of the coefficients $a$ and $b$ of the fitted production function.

The management index of individual food crops under the high, intermediate and low input levels, based on crop-specific production functions obtained from statistical analysis using input and yield data during 1978-2005, is given in Table 4.6. The creation of the production functions of food crops is detailed in §6.3.

The management index for food crops under the homogeneously implemented high, intermediate and low input scenarios in China is shown in Figure 4.12. The darker shading represents a higher index.

The food production realities in China are simulated in Chapter 6; the land limitations evaluation methodology in general, as well as the derived soil and management indices, are validated through statistical analysis in Chapter 7; and the design of the long-term food production scenarios are discussed in Chapter 8.
Table 4.6: Management index of individual food crops and crop groups at the high, intermediate and low input levels

<table>
<thead>
<tr>
<th>Crop</th>
<th>Input Level</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Intermediate</td>
<td>Low</td>
</tr>
<tr>
<td>rice</td>
<td>1.00</td>
<td>0.88</td>
<td>0.78</td>
</tr>
<tr>
<td>wheat</td>
<td>1.00</td>
<td>0.78</td>
<td>0.61</td>
</tr>
<tr>
<td>maize</td>
<td>1.00</td>
<td>0.81</td>
<td>0.65</td>
</tr>
<tr>
<td>sorghum</td>
<td>1.00</td>
<td>0.86</td>
<td>0.75</td>
</tr>
<tr>
<td>millet</td>
<td>1.00</td>
<td>0.70</td>
<td>0.49</td>
</tr>
<tr>
<td>grain</td>
<td>1.00</td>
<td>0.83</td>
<td>0.68</td>
</tr>
<tr>
<td>soybean</td>
<td>1.00</td>
<td>0.82</td>
<td>0.67</td>
</tr>
<tr>
<td>potato</td>
<td>1.00</td>
<td>0.91</td>
<td>0.83</td>
</tr>
<tr>
<td>food</td>
<td>1.00</td>
<td>0.82</td>
<td>0.68</td>
</tr>
</tbody>
</table>

Figure 4.12: Management index of food crops in China at homogeneously implemented high, intermediate and low input levels
Chapter 5

The Web-based Land Evaluation System

5.1 Introduction

A Web-based land evaluation system (WLES) was developed for quantitative land productivity assessments using climatic, crop, soil and management data. The WLES has a loosely coupled multi-tier structure which seamlessly integrates the domain knowledge of land evaluation (LE) and the natural resource databases (Ye and Van Ranst, 2004; Ye et al., 2004). The user-friendly front-end runs inside the client’s browsers, whereas the back-end database server defines reference crops and stores site-specific data collected. The LE knowledge engine sits in between and functions as the system core. It possesses a hierarchical structure with the biomass calculator (BMC), the soil-water balance simulator (WBS) and the land limitations evaluator (LLE) as its components. Ease-of-use features of the WLES range from database-driven parameter setting to graphical result visualization and simulated data downloading, and from topic-sensitive help to evaluation history tracking and management, and so forth. The WLES, together with its complete class documentation, is available at http://weble.ugent.be/. The implementation details are given in Volume II.

5.2 System structure

A model is a simplified representation of the reality in which outcomes can be computed without having to perform actual experiments. LE models are computer programs that predict the performance of a land use on a given land area, with information on that area’s land characteristics as inputs. Usually a LE model consists of three functional components: input, output, and the pluggable LE knowledge engine.

The input component deals with a wide range of data sources, which include reports (soil survey reports, land-use planning directives, development zoning regulations, etc.), spatial data in map formats (soil map, land-use map, infrastructure map, etc.) and legacy databases created by accomplished projects. The evaluation results are conveyed in the forms of reports, maps, and graphical presentations. The LE knowledge engine implements general or special purpose LE methodologies which can be plugged into the system upon client requests or system updates.
5.2 System structure

5.2.1 Standalone models

Standalone LE models are self-contained computer programs which apply the LE domain knowledge to local land-use conditions and produce qualitative or quantitative results. The term “self-contained” means the model is implemented with proprietary technologies, and the LE module itself is tightly coupled with a proprietary data engine. Most of the existing LE models, for instance the Automated Land Evaluation System (ALES) (Rossiter and Van Wambeke, 1997), the WOrld FOod STudies (WOFOST) (Supit et al., 1994; Boogaard et al., 1998), the Decision Support System for Agrotechnology Transfer (DSSAT) (Tsuji et al., 1994), etc., fall into this category.

Standalone models have maintainability, availability, reusability and modularity disadvantages. For example, ALES is usually released for running on DOS/Windows 9.x machines. A new version would have to be developed and released if the Linux (or Unix) operating system needs to be supported. A pluggable LE module would simply not work in ALES since the structure of the system is not compatible with this mechanism. Such disadvantages would obviously prevent the standalone models from being deployed in heterogeneous computation environments or being remotely accessed.

5.2.2 Multi-tier structure of the WLES

To the contrary, the LE knowledge engine of the WLES is decoupled with the data management and the user interface subsystems, and therefore the WLES has a 3-tier structure. The middle tier incorporates the LE domain knowledge, with two new standardized tiers, one in the front (frontend) and one in the back (backend). The frontend is a browser-based graphical user interface (GUI), which presents the system to the user, while the backend is the DBMS, usually a third party product, which keeps all data and results, collected or produced, by the middle tier. Figure 5.1 shows a block diagram of the tier structure of the WLES.

![Block diagram of the WLES tier structure](image-url)
5.2 System structure

The middle tier is built on top of a Web server so that the whole system communicates with the user using the hyper-text transfer protocol or HTTP (Fielding et al., 1997) to transport data and results across the network. On the other hand, separating the middle tier from data management provides greater scalability and higher performance for accessing land, soil and social-economic data. As such the middle tier concentrates on implementing the LE domain knowledge into, among others, a biomass calculator (BMC), a soil-water balance simulator (WBS), and a land limitations evaluator (LLE).

5.2.3 Subsystems

The WLES is made up of 7 subsystems which can be categorized in 3 groups, i.e., the user-related (User Management, User Authentication, and User Authorization), the evaluation-related (Interactive Mode and Batch Mode) and the help-related subsystems (WLES-Help and WLES-Doc). A functional flow chart of the WLES subsystems is shown in Figure 5.2.

![Functional flow chart of the WLES at the subsystem level](image)

Figure 5.2: Functional flow chart of the WLES at the subsystem level. Shaded entities are subsystems or groups of subsystems

The User Management Subsystem book-keeps the user profile records and manipulates the user roles; it redirects users to appropriate destinations, for instance, to the registration page for a new user and to the password retrieval page for returning users who forgot their passwords. It cooperates with the authentication and the authorization subsystems to help guard the system and to facilitate smooth evaluation pathways.

The User Authentication (Figure 5.3) and Authorization (Figure 5.4) Subsystems work together to identify users and to commit user roles. Users login with a username/password pair to gain admission to the LE service. Only pre-configured number of retrying times (§5.3.5) are allowed to attempt the credentials. Access to the login page will be denied if the limit is exceeded. Nevertheless guest login is always granted to guarantee the continuity of service. Registered users have significant advantages over guests firstly in the evaluation history management and review, and secondly in conducting regional-or-greater scale assessments (§5.3.3).

From the user’s perspective, the WLES works either in interactive mode (IM) or in batch mode (BM). The IM is handy for conducting a site scale evaluation while the BM is ideal for processing a much larger dataset quickly and silently (§5.2.4).
The WLES-Help and the WLES-Doc provide user assistance at the application and the programming level, respectively. WLES-Help is a topic-sensitive question answerer and problem solver concerning LE concepts, procedures, usage of the application, etc. WLES-Doc goes one step further in understanding the WLES: it documents all the programmatic entities and organizations in an object-oriented manner (§5.3.4). Both WLES-Help and WLES-Doc can be easily invoked by using system shortcuts (§5.3.1) during the entire evaluation life cycle.

5.2.4 Running modes

The WLES provides its core functionality of quantitative land productivity assessments through two distinctive running modes, i.e., the IM and the BM. An atomic sequence of method calls to the WLES class library forms an evaluation phase; multiple phases form a functional pathway. The BM evaluation pathway not only includes all the evaluation phases found in IM (see shaded “silent phases” in Figure 5.5), but extends to other, BM-only features as well. As an example, the IM pathway is summarized in Table 5.1.

Although the WLES is freely accessible in IM, it is license-protected in BM. Extra phases (Figure 5.5) were introduced in BM to add functionalities of license checking, source data uploading, error reporting, result downloading, etc. In practice, the BM finishes sooner than the IM does for a similar evaluation work-load: it is simply because the BM is computationally more task intensive per unit time.
5.3 Functional features

5.3.1 Web interface

The WLES Web interface is the client-side GUI that runs inside the user’s browsers. It guides the user to traverse the entire evaluation pathway. It accepts requests from, and feeds responses to, the user. Components of the Web interface include the access address bar, the system shortcuts and functional navigators, the user origin and client info display, the user profile updater, the suggestions and comments writer, and the workspace (Figures 5.6 and 5.7).
5.3 Functional features

Table 5.1: Listing of evaluation phases in IM

<table>
<thead>
<tr>
<th>Phase</th>
<th>Functions</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>StudyArea</td>
<td>Define the study area</td>
<td>1</td>
</tr>
<tr>
<td>ClimaticData</td>
<td>Collect climatic data: $T_{\text{min}}, T_{\text{max}}, RH, U, n, P$</td>
<td>2,3</td>
</tr>
<tr>
<td>GrowingPeriod</td>
<td>Determine the growing period and subperiods</td>
<td>G,R,4</td>
</tr>
<tr>
<td>CropData</td>
<td>Customize crop data, crop cycle and CDS</td>
<td>DB,G,1,5,6</td>
</tr>
<tr>
<td>RPP-Report</td>
<td>Radiation-thermal crop production potential</td>
<td>R,7</td>
</tr>
<tr>
<td>CropCoeff</td>
<td>Critical $k_c$ values; interpolates crop cycle</td>
<td>DB,G,R,1,4,5</td>
</tr>
<tr>
<td>SoilData1</td>
<td>Physical soil properties for all horizons</td>
<td>DB,R,1,2,3,7</td>
</tr>
<tr>
<td>RootingDepth</td>
<td>Rooting depth dynamics, limiting layer considered</td>
<td>DB,G,4,5,7</td>
</tr>
<tr>
<td>Depletion</td>
<td>Soil water depletion fraction ($p$) during crop cycle</td>
<td>DB,G,1,4,5,7</td>
</tr>
<tr>
<td>EffRain</td>
<td>Effective rainfall, user-defined method supported</td>
<td>DB,G,1,4,7</td>
</tr>
<tr>
<td>WaterBalance</td>
<td>Daily water balance simulation during crop cycle</td>
<td>G,R,4,7</td>
</tr>
<tr>
<td>YieldResponse</td>
<td>$k_y$ using mono, daily or CGP-specific scheme</td>
<td>DB,1,7</td>
</tr>
<tr>
<td>WPP-Report</td>
<td>Water-limited crop production potential</td>
<td>R,G,7</td>
</tr>
<tr>
<td>SoilChars1</td>
<td>(Auto) Selecting limiting soil characteristics</td>
<td>1,7,8</td>
</tr>
<tr>
<td>SoilData2</td>
<td>Chemical soil properties for all horizons</td>
<td>R,1,2,3,7,8</td>
</tr>
<tr>
<td>ReqTab</td>
<td>Builds the crop requirement table</td>
<td>DB,2,3,5,9</td>
</tr>
<tr>
<td>SoilIndex</td>
<td>$S_y$ based on the limiting soil characteristics</td>
<td>R,1,7</td>
</tr>
<tr>
<td>MgmtIndex</td>
<td>$M_y$ based on scheme used and input level</td>
<td>DB,R,1,7,9</td>
</tr>
<tr>
<td>LPP-Report</td>
<td>Land productional potential</td>
<td>R,G,4,5,9</td>
</tr>
</tbody>
</table>

1 assisted by drop-down list; 2 data well-formedness checker; 3 data completeness checker; 4 data download; 5 example available; 6 calendar assisted date selection; 7 hyperlinks to past phases; 8 assisted by “Auto” mode; 9 knowledge base rule in effect; DB database-driven default value setting; G graphical presentation; R textual report.

Access address
System shortcuts and function navigators
User origin and client info
Suggestions and comments
User profile
Workspace

Figure 5.6: Annotated WLES Web interface of the WLES-Home

- The address bar displays the http address to the current evaluation phase, usually
5.3 Functional features

The WLES adopted the comma-separated value format (.csv) as its interchanging data format for all data downloading/uploading tasks. The .csv format is well-supported and therefore the evaluation results generated by the WLES may be imported into 3rd-party spreadsheet, graphics and GIS software for post processing.
5.3 Functional features

The WLES-BM relies heavily on the .csv data format since the evaluation is conducted silently without user intervention. Be advised to consult the WLES-Help in advance to have a clear understanding of the specific data format.

5.3.3 Role-based user management

The WLES is running on the per user basis. Users are role players. The role is assigned by the user authentication and enforced by the user authorization subsystem. Four roles have been defined in WLES in an hierarchical order: system admin, user, guest and anonym.

- **System admin**
  The system admin is the top level role which is dedicated to the system administrative tasks, including system configuration (§5.3.5), log reading and error identification, communicating with users, database maintenance, etc. System admin is powerful indeed in many aspects, however, it has no authority to enter the running modes;

- **User and guest**
  The user and the guest are the only roles which are accepted by WLES for LE purposes. The user has to prove to the system who he is by using his username and password. So does the guest, although he doesn’t own a password. In other words, guest is a special, light-weight user with evaluation history review and BM capabilities trimmed;

- **anonym**
  The anonym is a group role which is automatically assigned to any client who has not proven his identity to the WLES. Special permissions are granted to anonym to visit the following subsystems: WLES Home, WLES-Help, WLES-Doc, User Registration, and Suggestions and Comments Report.

5.3.4 Topic-sensitive help and WLES-Doc

The WLES-Help Subsystem is a topic-sensitive help provider on the per page basis. It associates the help content with a topic keyword and stores them in the database. Special facilities are available for the system admin to conduct maintenance. Two keys (‘topic’ and ‘back’) are passed to the WLES-Help when the ‘Help’ hyper-link (Figures 5.6 and 5.7) is clicked from within the containing page. The following example shows the Web address (to the WLES-Help) triggered by the user registration page:

   http://weble.ugent.be/WLES/help/?topic=register&back=register

The WLES-Help use the ‘topic’ key to retrieve the help content from the database and ‘back’ key to redirect the client back to the invoking page (once the “Back” button in the Help page is clicked). The keys do not necessarily have the same value.

The WLES was implemented on top of the .NET Framework version 1.1 (Middleware Company, 2002; Richter, 2002; Microsoft Corporation, 2003) using 100% managed code of C#. The full MSDN-style documentation of the class library is available under the name of WLES-Doc. A top level overview of the namespaces, classes and structures is given in Table 5.2. It is not difficult to find the correlations between the UGent.WLES.RPP.Biomass class and the BMC, for instance, and between UGent.WLES.WPP.WaterBalance and the WBS, etc.
5.3 Functional features

Table 5.2: High level overview of the WLES Class Library

<table>
<thead>
<tr>
<th>Namespace†</th>
<th>Classes and structs‡</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP2Country</td>
<td>ClientInfo</td>
</tr>
<tr>
<td>WLES</td>
<td>Global, Logistics, RomanDate, Utilities, WlesDatabaseException, WlesException</td>
</tr>
<tr>
<td>WLES.Batch</td>
<td>Evaluation</td>
</tr>
<tr>
<td>WLES.Data</td>
<td>ClimaticData, CropData, Database, SoilData, SoilHorizon, SoilProfile, SoilTexture, WlesDataSet</td>
</tr>
<tr>
<td>WLES.ErrorHandler</td>
<td>GeneralError, PageNotFoundErr, ServerError</td>
</tr>
<tr>
<td>WLES.Help</td>
<td>Topic</td>
</tr>
<tr>
<td>WLES.Interactive</td>
<td>ClimaticDataUI, CropDataUI, Evaluation, SoilCharSet, SoilDataUI1, SoilDataUI2, StudyArea</td>
</tr>
<tr>
<td>WLES.LPP</td>
<td>CropRequirements, CropRequirementsUI, LPPReport, ManagementIndex, ManagementIndexUI, SoilIndex, SoilIndexUI, Yield</td>
</tr>
<tr>
<td>WLES.RPP</td>
<td>Biomass, CropCycle, GrowingPeriod, GrowingPeriodUI, RPPReport</td>
</tr>
<tr>
<td>WLES.Topo</td>
<td>Line, Point, Polygon, PolyLine</td>
</tr>
<tr>
<td>WLES.Users</td>
<td>Account, AuthCheck, Comments, CountryFlag, CustomImage, Filter, GuestLogin, ImageBroker, Login, Logout, MyAccount, MyProfile, Password, Register</td>
</tr>
<tr>
<td>WLES.WPP</td>
<td>CropCoefficient, CropCoefficientUI, DepletionFractionUI, EffectiveRainfall, EffectiveRainfallUI, PenmanMonteith, RootingDepth, RootingDepthUI, WaterBalance, WaterBalanceUI, WPPReport, YieldResponseToWater, YieldResponseToWaterUI, UserDefinedMethod</td>
</tr>
</tbody>
</table>

† Prefix “UGent.” omitted for neatness. Add upon use

5.3.5 XML-configurable system administration

The WLES was designed to have fundamental system administration tasks conducted by simply editing an XML-encoded text file named Web.config which is put at the application root. Two sections of the file, <appSettings> and <system.web> under the root <configuration> tag, are common to contain application-wide settings. Changes made to this file, once saved, take effect immediately, without program recompilation or application restart.

While the sub-tags of <authentication>, <authorization>, <globalization>, etc. in the <system.web> section are obviously used to modify the authentication/authorization mechanisms and the page encoding in a systematic way, the WLES replies more on the key-value pairs in the <appSettings> section for application maintenance. The following tag, for instance, specifies the Web server address of the WLES:

```
<add key = “WebServerURL” value = “http://weble.ugent.be/” />
```

The WLES takes advantage of this mechanism to avoid hard coding the application
settings, rules and physical URL values of all evaluation phases. For example, the key “MinLengthPwd” defines the minimum length of a valid password; “PwdRule” sets the basis for the system to judge whether a user password is valid (in the registration page or in profile update); and “WaterBalanceURL” stores the URL value of the water balance simulation phase.

5.3.6 Other ease-of-use features

The user-friendliness was one of the top requirements (together with performance and robustness) when the WLES was designed and implemented. Besides the aforementioned, the following features also contribute to the user-friendliness of the WLES:

- Build-in reference crop support. The WLES maintains a list of more than 70 crops and cultivars with known mean or range of physiological values;
- Data well-formedness, completeness and integrity checkers, for example for the climatic, crop (CDS, CGP) and soil (physical, chemical) data;
- Graphical presentation of results and source data download are supported whenever possible (Figures 5.8, 5.9 and 5.10);
- Database-driven default value settings and rule-based “auto” modes, for example the LAI at maximum growth rate value for a reference crop, and the CEC_{soil} and CEC_{clay} values when the clay content is known;
- Example-assisted value setting and calendar-assisted date setting, for example in the determination of the photosynthetical pathway of a customized crop, and in CDS/CGP definition.

![Graphical presentation of the growing period and crop cycle determinations](figure5_8.png)

Figure 5.8: Graphical presentation of the growing period (a) and crop cycle determinations, and the RPP result (b)
5.3 Functional features

Figure 5.9: Graphical presentation of the WBS result (a) and the data download interface (b) triggered by the “Data” link which is commonly found in WLES graphics

Figure 5.10: Graphical presentation of the crop requirement table and the soil index determinations (a) and the LPP results compared to those of RPP and WPP in a bar-chart (b)
Chapter 6

Yield Simulation and Model Validation

6.1 Introduction

The food production in China in 2005 was simulated using climatic, soil, crop and management data. The sown structure of food crops, the effective irrigation rate and the factor input data were collected and parameterized on a per province basis [NBSC, 2005]. The simulation was conducted per grid cell and the results were aggregated back to the provincial scale. The simulated yields were statistically tested against the observed ones, either individually (as rice, wheat, maize, etc.) or collectively (as grain or food crops), to validate the modeling process. The downscaling of parameters [Jarvis, 1995] was practically due to the lack of homogeneous measurements at the field-scale; and the upscaling of results was thus pragmatic yet appropriate in model validation [Hatfield, 2001; Anderson et al., 2003].

The average yields of rainfed and irrigated crops were assessed with the following formula, respectively [Ye et al., 2007]:

\[ Y_{\text{rain}} = \sum_{i=1}^{n} (B_n \cdot HI \cdot f_W \cdot S_y \cdot M_y)_i \]  
\[ Y_{\text{irri}} = \sum_{i=1}^{n} (B_n \cdot HI \cdot S_y \cdot M_y)_i \]

where \( Y_{\text{rain}} \) and \( Y_{\text{irri}} \) are the rainfed and the irrigated yield, respectively, \( i \) is the serial number of a rotating crop, \( n \) the total number of crops in the rotation, \( B_n \) the net biomass (§4.4.1), \( HI \) the harvest index (Figure E.17 in Appendix E), \( f_W \) the yield reduction coefficient due to water stress (Equations 4.60 and 4.61), \( S_y \) the soil index (§4.6.1) and \( M_y \) the management index (§4.6.2).

The actual crop yield (\( Y \)) was obtained by combining Equations 6.1 and 6.2 using the effective irrigation rate, \( r \) [Ye et al., 2008]:

\[ Y = \frac{r \cdot Y_{\text{irri}} + (100 - r) \cdot Y_{\text{rain}}}{100} \]

where \( r \) is expressed as a percentage (%) of the area of cropland with permanent irrigating infrastructures over the total area of cropland.
6.2 Effective irrigation rate

The effective irrigation rate \((r)\) is defined as the probability that the net crop water requirement, represented by the evapotranspiration deficit \((\Delta ET = |ET_a - ET_m|)\), is met by irrigation during a given observation period. It is usually expressed as a percentage (%) of the number of years \((m)\) that the equilibrium between the crop water requirement and the irrigation water supply is maintained over the total number of years \((n)\) of the observation period. The mathematical expectation of \(r\) is written as (MOC, 1999):

\[
r = \frac{m}{n + 1} \cdot 100
\]

where \(n \geq 30\). The denominator is deliberately set as \((n + 1)\) in order to raise the “effectiveness” criterium.

The median value of \(r\), from a multi-year time series where \(n \geq 30\), is given by the following empirical equation:

\[
\bar{r} = \frac{m - 0.3}{n + 0.4} \cdot 100
\]

The national standard GB50288-99: “Code for design of irrigation and drainage engineering” (MOC, 1999) adopts different effective irrigation rates to meet the average agricultural needs based on hydrological conditions in different agro-climatic zones (§2.6.5) in China (Table 6.1).

<table>
<thead>
<tr>
<th>Climate</th>
<th>Farming Systems</th>
<th>(r_{min}) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>humid</td>
<td>upland crops</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>paddy rice</td>
<td>85</td>
</tr>
<tr>
<td>arid</td>
<td>upland crops</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>paddy rice</td>
<td>75</td>
</tr>
<tr>
<td>others</td>
<td></td>
<td>70</td>
</tr>
</tbody>
</table>

† Refer to Table 2.8 for definition

In practice, the effective irrigation rate is also expressed as a percentage (%) of the area of cropland equipped with permanent irrigating infrastructures and supplied with sufficient irrigation water \(S_{irri}\) over the total area of cropland \(S_{tot}\):

\[
r = \frac{S_{irri}}{S_{tot}} \cdot 100
\]

Table 6.2 lists the areas of the irrigated and the total croplands and the effective irrigation rate for each province in China (NBSC, 2005).

6.3 Factor inputs and management index

Three categories of factors were selected to represent the intensity of the management practices. They include: (1) application of fertilizers and chemicals, (2) agro-machinery usage and electrical consumption, and (3) irrigation infrastructure investment (Table 6.2). The province-specific values of the former two are given in Table 6.3.
6.3 Factor inputs and management index

Table 6.2: Province-specific effective irrigation rate, 2005

<table>
<thead>
<tr>
<th>Prov†</th>
<th>Irrigated Arable EIR‡</th>
<th>Prov†</th>
<th>Irrigated Arable EIR‡</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1,000 ha</td>
<td></td>
<td>1,000 ha</td>
</tr>
<tr>
<td>1</td>
<td>186.7</td>
<td>343.9</td>
<td>54.3</td>
</tr>
<tr>
<td>2</td>
<td>353.4</td>
<td>485.6</td>
<td>72.8</td>
</tr>
<tr>
<td>3</td>
<td>4,459.8</td>
<td>6,883.3</td>
<td>64.8</td>
</tr>
<tr>
<td>4</td>
<td>1,088.2</td>
<td>4,588.6</td>
<td>23.7</td>
</tr>
<tr>
<td>5</td>
<td>2,635.9</td>
<td>8,201.0</td>
<td>32.1</td>
</tr>
<tr>
<td>6</td>
<td>1,520.1</td>
<td>4,174.8</td>
<td>36.4</td>
</tr>
<tr>
<td>7</td>
<td>1,595.2</td>
<td>5,578.4</td>
<td>28.6</td>
</tr>
<tr>
<td>8</td>
<td>2,282.1</td>
<td>11,773.0</td>
<td>19.4</td>
</tr>
<tr>
<td>9</td>
<td>245.7</td>
<td>315.1</td>
<td>78.0</td>
</tr>
<tr>
<td>10</td>
<td>3,839.0</td>
<td>5,061.7</td>
<td>75.8</td>
</tr>
<tr>
<td>11</td>
<td>1,406.9</td>
<td>2,125.3</td>
<td>66.2</td>
</tr>
<tr>
<td>12</td>
<td>3,304.6</td>
<td>5,971.7</td>
<td>55.3</td>
</tr>
<tr>
<td>13</td>
<td>941.5</td>
<td>1,434.7</td>
<td>65.6</td>
</tr>
<tr>
<td>14</td>
<td>1,841.6</td>
<td>2,993.4</td>
<td>61.5</td>
</tr>
<tr>
<td>15</td>
<td>4,766.8</td>
<td>7,689.3</td>
<td>62.0</td>
</tr>
<tr>
<td>16</td>
<td>4,829.1</td>
<td>8,110.3</td>
<td>59.5</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>54,904.3</td>
<td>130,872.5</td>
</tr>
</tbody>
</table>

† Province. Refer to Table 2.1 for names
‡ Effective irrigation rate

Table 6.3: Province-specific factor inputs, 2005

<table>
<thead>
<tr>
<th>Prov†</th>
<th>(1)</th>
<th>(2)</th>
<th>Prov†</th>
<th>(1)</th>
<th>(2)</th>
<th>Prov†</th>
<th>(1)</th>
<th>(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.5</td>
<td>38.4</td>
<td>12</td>
<td>277.6</td>
<td>59.4</td>
<td>23</td>
<td>74.3</td>
<td>30.3</td>
</tr>
<tr>
<td>2</td>
<td>22.9</td>
<td>48.4</td>
<td>13</td>
<td>121.7</td>
<td>137.6</td>
<td>24</td>
<td>137.2</td>
<td>40.3</td>
</tr>
<tr>
<td>3</td>
<td>289.9</td>
<td>310.8</td>
<td>14</td>
<td>123.5</td>
<td>42.3</td>
<td>25</td>
<td>4.0</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>93.4</td>
<td>63.2</td>
<td>15</td>
<td>451.0</td>
<td>304.1</td>
<td>26</td>
<td>143.1</td>
<td>84.0</td>
</tr>
<tr>
<td>5</td>
<td>104.4</td>
<td>26.7</td>
<td>16</td>
<td>493.2</td>
<td>157.7</td>
<td>27</td>
<td>72.4</td>
<td>41.4</td>
</tr>
<tr>
<td>6</td>
<td>117.9</td>
<td>158.0</td>
<td>17</td>
<td>281.9</td>
<td>64.8</td>
<td>28</td>
<td>6.6</td>
<td>2.9</td>
</tr>
<tr>
<td>7</td>
<td>159.1</td>
<td>23.7</td>
<td>18</td>
<td>203.2</td>
<td>57.5</td>
<td>29</td>
<td>27.6</td>
<td>9.0</td>
</tr>
<tr>
<td>8</td>
<td>143.8</td>
<td>33.5</td>
<td>19</td>
<td>201.3</td>
<td>748.2</td>
<td>30</td>
<td>99.2</td>
<td>32.6</td>
</tr>
<tr>
<td>9</td>
<td>15.0</td>
<td>114.3</td>
<td>20</td>
<td>195.2</td>
<td>32.4</td>
<td>31</td>
<td>114.1</td>
<td>500.0</td>
</tr>
<tr>
<td>10</td>
<td>336.8</td>
<td>679.8</td>
<td>21</td>
<td>41.1</td>
<td>3.3</td>
<td>32</td>
<td>146.3</td>
<td>442.1</td>
</tr>
<tr>
<td>11</td>
<td>93.3</td>
<td>441.2</td>
<td>22</td>
<td>291.7</td>
<td>146.3</td>
<td>Total</td>
<td>4,750.9</td>
<td>4,432.7</td>
</tr>
</tbody>
</table>

† Province. Refer to Table 2.1 for names
(1): Fertilizers and chemicals (10,000 ton)
(2): Machinery and electro (100 million kWh)

6.3.1 The high, intermediate and low input levels

The input values were firstly divided by the area of cropland and then summed per input category for each of the economic-development belts (§2.2.3), respectively. Relative scores were obtained per input category by assigning a score of 1.0 to the economic development belt with the lowest input value (Table 6.4). The resulting overall score of 12, which
was defined here as the high input level, was therefore attributed to the east belt in recognizing its leading role in agro-economic development. The overall scores of 6 and 3, of the middle and the west belts, were accordingly defined as the intermediate and the low input levels, respectively, reflecting the difference in agricultural investment intensity between the two. As such the management index can be assessed per input level using the crop-specific approach detailed in §4.6.2. But before the assessment is carried out, a production function has to be created for each crop.

### Table 6.4: Definition of the high, intermediate and low input levels

<table>
<thead>
<tr>
<th>Belt</th>
<th>Absolute Input (1)</th>
<th>Standardized (2)</th>
<th>Overall Score (3)</th>
<th>Input Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>405.8</td>
<td>7,082.9</td>
<td>56.0</td>
<td>12 High</td>
</tr>
<tr>
<td>Middle</td>
<td>335.0</td>
<td>2,085.2</td>
<td>39.8</td>
<td>6 Intermediate</td>
</tr>
<tr>
<td>West</td>
<td>193.4</td>
<td>875.2</td>
<td>25.8</td>
<td>3 Low</td>
</tr>
</tbody>
</table>

(1): Application of fertilizers and chemicals (ton ha$^{-1}$)
(2): Use of machinery and consumption of electricity (1,000 kWh ha$^{-1}$)
(3): Irrigation infrastructure, represented by irrigation rate (%)

The input levels defined here will also be used as a baseline in designing the long-term crop production scenarios in Chapter 8.

### 6.3.2 Crop-specific production functions

As already discussed in §4.6.2, the theory of production (Scanzieri, 1993; Hammond, 1994; O’Hara, 1997a,b), and in particular the Cobb-Douglas production function (Cobb and Douglas, 1928; Douglas, 1976; Pokrovski, 2003), as well as the overall score of factor inputs form the foundation of the crop-specific management index assessment approach proposed and used in this dissertation.

The average observed yields of individual food crops (rice, wheat, maize, etc.) and crop groups (grain and food) in a time series during 1978-2005 (NBSC, 1999, 2006), given in Appendix D, were drawn in scatter plots against the overall score of factor inputs of the same period (Figure 6.1). The overall score (refer to §4.6.2 for details) was calculated as the sum of the relative scores of three categories of factor inputs listed in the beginning of §6.3. Although the yields of food crops responded to factor inputs at different sensitivities, all crops and crop groups demonstrated the same pattern in their response to changing values of factor inputs. It was the pattern of a production function, as suggested by the best-fitting curves in Figure 6.1(a) through (i).

Coefficients $a$ and $b$ of the following production function (same as Equation 4.69):

$$Y = a \cdot S^b$$

were statistically determined on a per crop basis using input and yield data during 1978-2005. The values of $a$ and $b$, as well as the decisive coefficient ($R^2$) and the significance level of the fitted regression model are given in Table 6.5.

The production function of the wheat crop, for instance, can be written as:

$$Y = 1.47 \cdot S^{0.36}$$

where $Y$ is the wheat yield in t ha$^{-1}$ and $S$ is the overall score of factor inputs.
6.3 Factor inputs and management index

On average, 85% of the variations in wheat yield may be explained, at the highest significance level, by the overall score of factor inputs alone using this production function.

Table 6.5: Single factor production functions with the yield of an individual food crop (t ha\(^{-1}\)) as the dependent and the overall score of factor inputs as the independent variable

<table>
<thead>
<tr>
<th>Crop</th>
<th>a</th>
<th>b</th>
<th>(R^2)</th>
<th>p-value</th>
<th>Significance†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td>3.73</td>
<td>0.18</td>
<td>0.83</td>
<td>1.61\times10^{-09}</td>
<td>***</td>
</tr>
<tr>
<td>Wheat</td>
<td>1.47</td>
<td>0.36</td>
<td>0.85</td>
<td>2.98\times10^{-10}</td>
<td>***</td>
</tr>
<tr>
<td>Maize</td>
<td>2.20</td>
<td>0.31</td>
<td>0.78</td>
<td>3.04\times10^{-08}</td>
<td>***</td>
</tr>
<tr>
<td>Sorghum</td>
<td>2.10</td>
<td>0.21</td>
<td>0.76</td>
<td>2.06\times10^{-03}</td>
<td>**</td>
</tr>
<tr>
<td>Millet</td>
<td>0.71</td>
<td>0.52</td>
<td>0.85</td>
<td>4.09\times10^{-04}</td>
<td>***</td>
</tr>
<tr>
<td>Grain</td>
<td>2.36</td>
<td>0.27</td>
<td>0.86</td>
<td>1.91\times10^{-10}</td>
<td>***</td>
</tr>
<tr>
<td>Soybean</td>
<td>0.79</td>
<td>0.29</td>
<td>0.68</td>
<td>1.51\times10^{-06}</td>
<td>***</td>
</tr>
<tr>
<td>Potato</td>
<td>2.39</td>
<td>0.13</td>
<td>0.61</td>
<td>1.02\times10^{-05}</td>
<td>***</td>
</tr>
<tr>
<td>Food</td>
<td>2.10</td>
<td>0.28</td>
<td>0.86</td>
<td>1.95\times10^{-10}</td>
<td>***</td>
</tr>
</tbody>
</table>

† Significance levels: \(\geq 0.1(\ldots); > 0.05(\ldots); \leq 0.05(*)\); \(\leq 0.01(**)\); \(\leq 0.001(***))

Figure 6.1: Yields of food crops (t ha\(^{-1}\)) in response to the overall score of factor inputs
6.3.3 Management index of food crops in China

The management index, $M_y$, of a particular food crop under an input level represented by the overall score of factor inputs, $S$, is given by:

$$M_y = \left( \frac{S}{S_1} \right)^b$$  \hspace{1cm} (6.8)

where $S_1$ is the overall score of factor inputs at the high input level. According to Table 6.4, $S_1 = 12$ for food crops in China; $b$ is a crop-specific coefficient associated with production function (Table 6.5). Please refer to §4.6.2 for the derivation of this equation.

The management index of rice at the intermediate input level ($S = 6$), for instance, is easily obtained with Equation 6.8:

$$\left( \frac{6}{12} \right)^{0.18} = 0.88.$$  \hspace{1cm} (6.9)

The management index values of individual food crops and crop groups were calculated on a per input level basis and listed in Table 4.6 (Chapter 4). A map of management index of food crops, as sown in 2005 in China, was derived and shown in Figure 6.2. Higher indices are represented by darker shadings in the map. A general east-west decreasing trend of the management index values is observed, suggesting a similar trend for factor input levels across the east-west economic development belts. Local variations of index values are explained by the sown structure variations of food crops. Comparing to the hypothetical management index maps (Figure 4.12) of homogeneous input levels: the general trend found in this map was not presented in Figure 4.12, while local variations are identical in both cases.

Both the definition of input levels and the resulting management index of food crops will be validated in §7.4.

6.4 Grid-based yield simulation

6.4.1 The scale issue

The actual food production in 2005 was simulated on a per grid-cell basis using the climatic, soil, crop and management data following the approach outlined in §6.1. The simulation process was looped to iterate all the 778,104 grid cells with data retrieved from and results saved in the database. The potential productivities were then analyzed and mapped (Ye et al., 2007, 2008).

As discussed in earlier chapters, the climatic data was interpolated from point datasets and 5 km $\times$ 5 km grid surfaces were generated per climatic parameter per month (§2.6). The soil data was extracted and converted from the ISRIC-WISE database of 5' $\times$ 5', and downscaled to 5 km $\times$ 5 km per soil parameter per equal section of 20 cm (§2.7). The crop physiological and phenological data were linked to the representative crop rotations of each cropping system found in an agro-ecological zone (§3.3, §3.4), while the sown structure of food crops was given on a per province basis (§3.5). The input levels were defined according to the economic development belts (§6.3) while the province-specific effective irrigation rate will still be used to represent the irrigating probability in §6.4.2 below.

The downscaling of the crop and management data was pragmatic due to the lack of homogeneous data coverage at field scale over the study area. The error introduced at
downscaling by assuming parametric homogeneity is counter-balanced to a great deal, if not completely, when the results are upscaled back to the provincial scale. It is found common and acceptable in agro-ecological research to downscale part of the datasets from a larger scale (Denier van der Gon et al., 2000; Bouwman et al., 2002) as long as the results are aggregated back to the data scale for analysis and validation (Jarvis, 1995; Hatfield, 2001; Anderson et al., 2003).

6.4.2 Irrigating probability

The area-weighted average crop yield ($Y$) in a study area is calculated from the average irrigated ($Y_{irri}$) and rainfed ($Y_{rain}$) yields using the following formulas:

$$Y = \frac{Y_{irri} \cdot S_{irri} + Y_{rain} \cdot S_{rain}}{S_{tot}}$$  \hspace{1cm} (6.10)

and

$$S_{tot} = S_{irri} + S_{rain}$$  \hspace{1cm} (6.11)

where $S_{tot}$ is the total area of cropland in the study area, $S_{irri}$ and $S_{rain}$ are the areas of the irrigated and the rainfed croplands, respectively.
6.4 Grid-based yield simulation

The equation can be re-arranged in the following way:

\[
Y = \frac{Y_{irri} \cdot S_{irri} + Y_{rain} \cdot (S_{tot} - S_{irri})}{S_{tot}} = Y_{irri} \cdot \frac{S_{irri}}{S_{tot}} + Y_{rain} \cdot \left( 1 - \frac{S_{irri}}{S_{tot}} \right)
\]

where \( S_{irri}/S_{tot} \) was defined by Equation 6.6 as the effective irrigation rate \((r)\), which is actually equivalent to the irrigating probability \((p)\) which represents the likelihood that a particular tract piece of cropland is irrigated.

\[
Y = Y_{irri} \cdot p + Y_{rain} \cdot (1 - p)
\]

The meaning of this equation is different from that of Equation 6.3 or 6.10. The latter two calculate the area-weighted average yield of multiple crops while the former calculates the yield of a single crop, which is more precise in simulating the performance of individual crops in a rotation. The province-specific effective irrigation rate values were used as irrigating probabilities in the crop yield simulation, assuming homogeneity in water distribution over cropland surfaces and among rotating crops.

There are subtle semantic differences between the terms “irrigation rate” and “irrigating probability”. The former is usually area-based and can be determined by using Equation 6.6; the irrigated cropland is likely different from the non-irrigated one (Figure 6.3a). The latter is more time-based than area-based. The irrigation water is assumed to be evenly distributed (Figure 6.3b). Nevertheless, the calculation procedures for them are mathematically equivalent.

![Figure 6.3](image-url)

**Figure 6.3**: Conceptual sketches for area-based irrigation rate (a) and time-based irrigation probability (b). In (a): A and E are croplands with irrigation infrastructures while B, C and D are without; in (b): A-E are all irrigated at probability \( p \)
6.4 Grid-based yield simulation

6.4.3 Results

The simulated yields of all 12 food crop species (for example: the rice crop has 4 species including the northern, early, intermediate and late rice; refer to Table 3.2 for a complete list) were first combined to derive average yields of individual crops (for example: rice) per grid cell. The latter was then aggregated at the provincial scale. The weighted mean of food productivity was estimated by applying the actual sown structure of food crops in 2005. The resulting yields of food crops, both individually (as rice, wheat, maize, sorghum and millet) and collectively (as grain and food), are given in Tables 6.6 and 6.7.

Table 6.6: Simulated and observed yields of rice, wheat, maize, sorghum and millet, kg ha⁻¹

<table>
<thead>
<tr>
<th>Prov†</th>
<th>Rice O</th>
<th>S</th>
<th>Wheat O</th>
<th>S</th>
<th>Maize O</th>
<th>S</th>
<th>SM‡</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6,250</td>
<td>NA</td>
<td>5,179</td>
<td>6,168</td>
<td>4,652</td>
<td>4,154</td>
<td>2,381</td>
<td>2,460</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8,102</td>
<td>NA</td>
<td>4,785</td>
<td>5,192</td>
<td>5,059</td>
<td>5,546</td>
<td>833</td>
<td>951</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5,665</td>
<td>5,468</td>
<td>4,873</td>
<td>5,413</td>
<td>4,401</td>
<td>3,765</td>
<td>1,370</td>
<td>1,360</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4,231</td>
<td>NA</td>
<td>3,654</td>
<td>3,633</td>
<td>5,614</td>
<td>5,607</td>
<td>1,494</td>
<td>1,482</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6,737</td>
<td>6,867</td>
<td>2,639</td>
<td>1,118</td>
<td>5,658</td>
<td>5,951</td>
<td>1,651</td>
<td>1,839</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7,378</td>
<td>7,890</td>
<td>4,320</td>
<td>4,706</td>
<td>6,753</td>
<td>7,216</td>
<td>4,788</td>
<td>5,887</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7,292</td>
<td>7,658</td>
<td>2,982</td>
<td>2,452</td>
<td>6,238</td>
<td>7,067</td>
<td>4,242</td>
<td>5,286</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7,117</td>
<td>6,876</td>
<td>3,255</td>
<td>2,806</td>
<td>4,311</td>
<td>3,524</td>
<td>2,846</td>
<td>3,155</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8,005</td>
<td>8,628</td>
<td>3,653</td>
<td>3,888</td>
<td>6,190</td>
<td>NA</td>
<td>3,803</td>
<td>4,416</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>7,919</td>
<td>8,237</td>
<td>4,295</td>
<td>4,261</td>
<td>5,567</td>
<td>5,778</td>
<td>4,263</td>
<td>5,290</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6,681</td>
<td>6,873</td>
<td>3,193</td>
<td>2,585</td>
<td>4,128</td>
<td>NA</td>
<td>3,785</td>
<td>4,512</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>6,067</td>
<td>6,230</td>
<td>3,836</td>
<td>4,132</td>
<td>4,844</td>
<td>4,451</td>
<td>5,119</td>
<td>6,335</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>5,539</td>
<td>5,107</td>
<td>3,065</td>
<td>2,503</td>
<td>3,360</td>
<td>2,652</td>
<td>3,333</td>
<td>3,775</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>5,213</td>
<td>3,955</td>
<td>1,518</td>
<td>401</td>
<td>3,333</td>
<td>NA</td>
<td>2,609</td>
<td>2,816</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>7,283</td>
<td>NA</td>
<td>5,338</td>
<td>6,452</td>
<td>6,106</td>
<td>6,354</td>
<td>3,845</td>
<td>4,477</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>7,044</td>
<td>6,875</td>
<td>5,109</td>
<td>5,460</td>
<td>4,339</td>
<td>3,541</td>
<td>4,005</td>
<td>4,661</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>7,548</td>
<td>8,126</td>
<td>2,924</td>
<td>2,113</td>
<td>5,010</td>
<td>5,132</td>
<td>2,957</td>
<td>3,556</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>6,149</td>
<td>6,388</td>
<td>1,916</td>
<td>895</td>
<td>4,579</td>
<td>4,101</td>
<td>2,822</td>
<td>1,990</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>5,251</td>
<td>4,085</td>
<td>2,833</td>
<td>1,136</td>
<td>4,068</td>
<td>3,091</td>
<td>4,286</td>
<td>5,280</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>4,768</td>
<td>3,616</td>
<td>1,624</td>
<td>582</td>
<td>3,002</td>
<td>2,356</td>
<td>1,154</td>
<td>1,068</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>4,398</td>
<td>2,887</td>
<td>NA</td>
<td>1,872</td>
<td>3,431</td>
<td>2,846</td>
<td>NA</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>7,213</td>
<td>7,232</td>
<td>3,216</td>
<td>2,686</td>
<td>4,808</td>
<td>4,332</td>
<td>2,646</td>
<td>3,030</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>6,657</td>
<td>6,872</td>
<td>1,789</td>
<td>693</td>
<td>4,726</td>
<td>4,228</td>
<td>1,547</td>
<td>1,489</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>5,887</td>
<td>5,929</td>
<td>2,240</td>
<td>1,032</td>
<td>3,831</td>
<td>2,982</td>
<td>2,333</td>
<td>2,411</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>5,455</td>
<td>NA</td>
<td>6,429</td>
<td>NA</td>
<td>4,848</td>
<td>NA</td>
<td>4,939</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>5,967</td>
<td>6,042</td>
<td>3,559</td>
<td>3,158</td>
<td>3,886</td>
<td>3,062</td>
<td>1,171</td>
<td>1,076</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>7,959</td>
<td>8,626</td>
<td>2,917</td>
<td>1,895</td>
<td>5,024</td>
<td>5,292</td>
<td>2,333</td>
<td>2,406</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>NA</td>
<td>NA</td>
<td>3,630</td>
<td>3,481</td>
<td>7,500</td>
<td>8,100</td>
<td>3,152</td>
<td>3,670</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>8,152</td>
<td>NA</td>
<td>2,882</td>
<td>1,255</td>
<td>6,264</td>
<td>7,115</td>
<td>720</td>
<td>935</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>5,883</td>
<td>NA</td>
<td>5,138</td>
<td>5,560</td>
<td>6,979</td>
<td>7,811</td>
<td>3,619</td>
<td>4,314</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>4,974</td>
<td>3,576</td>
<td>NA</td>
<td>3,994</td>
<td>NA</td>
<td>6,659</td>
<td>NA</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>

† Province. Refer to Table 2.1 for names; ‡ SM = Sorghum + millet
b O = Observed; § S = Simulated; ¶ NA = Not a number
### Table 6.7: Simulated and observed yields of beans, tubers; and collective grain and food crops, kg ha$^{-1}$

<table>
<thead>
<tr>
<th>Prov†</th>
<th>Grain</th>
<th>Beans</th>
<th>Tubers</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O$^b$</td>
<td>S$^b$</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>4.779</td>
<td>4.291</td>
<td>2.098</td>
<td>2.471</td>
</tr>
<tr>
<td>2</td>
<td>5.958</td>
<td>5.288</td>
<td>1.528</td>
<td>1.578</td>
</tr>
<tr>
<td>4</td>
<td>4.222</td>
<td>3.117</td>
<td>1.399</td>
<td>1.078</td>
</tr>
<tr>
<td>5</td>
<td>4.369</td>
<td>4.085</td>
<td>1.263</td>
<td>0.757</td>
</tr>
<tr>
<td>6</td>
<td>6.653</td>
<td>8.185</td>
<td>1.667</td>
<td>1.871</td>
</tr>
<tr>
<td>8</td>
<td>5.247</td>
<td>5.626</td>
<td>1.772</td>
<td>2.011</td>
</tr>
<tr>
<td>11</td>
<td>6.314</td>
<td>7.589</td>
<td>2.218</td>
<td>2.930</td>
</tr>
<tr>
<td>13</td>
<td>5.429</td>
<td>5.745</td>
<td>2.142</td>
<td>2.580</td>
</tr>
<tr>
<td>14</td>
<td>5.177</td>
<td>5.573</td>
<td>1.497</td>
<td>1.450</td>
</tr>
<tr>
<td>15</td>
<td>5.704</td>
<td>5.767</td>
<td>2.970</td>
<td>3.841</td>
</tr>
<tr>
<td>16</td>
<td>4.981</td>
<td>4.811</td>
<td>1.886</td>
<td>2.051</td>
</tr>
<tr>
<td>18</td>
<td>5.923</td>
<td>6.538</td>
<td>2.016</td>
<td>2.371</td>
</tr>
<tr>
<td>20</td>
<td>4.391</td>
<td>3.715</td>
<td>1.320</td>
<td>1.060</td>
</tr>
<tr>
<td>22</td>
<td>5.387</td>
<td>5.576</td>
<td>2.022</td>
<td>2.364</td>
</tr>
<tr>
<td>23</td>
<td>4.673</td>
<td>4.261</td>
<td>1.250</td>
<td>752</td>
</tr>
<tr>
<td>24</td>
<td>4.133</td>
<td>2.755</td>
<td>1.243</td>
<td>604</td>
</tr>
<tr>
<td>25</td>
<td>5.299</td>
<td>NA$^z$</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>26</td>
<td>3.727</td>
<td>2.331</td>
<td>1.048</td>
<td>776</td>
</tr>
<tr>
<td>27</td>
<td>3.409</td>
<td>2.206</td>
<td>1.622</td>
<td>1.620</td>
</tr>
<tr>
<td>29</td>
<td>4.239</td>
<td>3.666</td>
<td>984</td>
<td>550</td>
</tr>
<tr>
<td>30</td>
<td>5.864</td>
<td>5.765</td>
<td>2.453</td>
<td>3.537</td>
</tr>
<tr>
<td>31</td>
<td>4.974</td>
<td>4.816</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

† Province. Refer to Table 2.1 for names

$^b$ O = Observed; $^z$ S = Simulated; $^z$ NA = Not a number

### 6.5 Model validation

#### 6.5.1 Yield difference analysis

A visual check of the mean observed and simulated yields of food crops, shown as box-and-whisker plots (a, c, e, g) in Figures 6.4 and 6.5, and the goodness of fit between the observed and simulated yields, represented by scattered circles against the 1:1 line (b, d, f, h in both figures) – both after outliers being rejected – reveals that the average
6.5 Model validation

Yields of rice (6.4a, b), wheat (6.4c, d), beans (6.5c, d) and tubers (6.5e, f) are rather closely simulated, whereas the average yield of maize (6.4e, f) is underestimated by 10% and that of sorghum/millet (6.4g, h) is overestimated by 12% (Table 6.8). As a matter of fact, systematic estimation errors were observed in either directions away from the average yield. The fitted lines suggest that overestimations probably occurred in high-yielding croplands, while underestimations occurred in low-yielding croplands. The use of average values at the provincial level for (a) effective irrigation rate (Equation 6.6) and (b) management index (Equation 6.8) may have largely contributed to the systematic under- and overestimations of crop yields. The use of the average effective irrigation rate at the provincial scale, for instance, erases the parametric heterogeneity among all 5 km × 5 km grid cells within the province. This simplification is pragmatic, due to unavailability of homogeneous field-level data, and still acceptable in predicting the average yields and total production. Its impact on the prediction of the long-term food producing capacities is quantifiable in a future research, as recommended earlier.

Nevertheless, the observed-simulated differences in yields of all grain (6.5a) or food (6.5g) crops taken together are still acceptable since the underestimation counterbalances the overestimation (6.5b, h), and vice versa, to a great deal. This assertion will be endorsed by the statistical analysis discussed next.

Figure 6.4: Comparability of means and goodness of fit between simulated and observed yields of rice, wheat, maize, sorghum and millet

6.5.2 Statistical analysis

The equality of the means of the simulated and observed yields of food crops was statistically tested using the paired t-test approach (R Development Core Team, 2006) with the
6.5 Model validation

![Figure 6.5: Comparability of means and goodness of fit between simulated and observed yields of beans, tubers, and collective grain and food crops](image)

![Table 6.8: Simulated versus observed yields of food crops: a summary](table)

<table>
<thead>
<tr>
<th></th>
<th>Rice</th>
<th>Wheat</th>
<th>Maize</th>
<th>S+M†</th>
<th>Grain</th>
<th>Beans</th>
<th>Tubers</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed (t ha⁻¹)</td>
<td>6.66</td>
<td>3.65</td>
<td>4.83</td>
<td>2.75</td>
<td>5.02</td>
<td>1.89</td>
<td>3.99</td>
<td>4.70</td>
</tr>
<tr>
<td>Simulated (t ha⁻¹)</td>
<td>6.87</td>
<td>3.76</td>
<td>4.39</td>
<td>3.09</td>
<td>5.05</td>
<td>2.05</td>
<td>4.19</td>
<td>4.92</td>
</tr>
<tr>
<td>Difference (%)</td>
<td>+3.15</td>
<td>+3.01</td>
<td>-10.02</td>
<td>+12.36</td>
<td>+0.60</td>
<td>+8.47</td>
<td>+5.01</td>
<td>+4.68</td>
</tr>
</tbody>
</table>

† Sorghum + millet

yield data grouped by province. As a pre-condition, the Shapiro-Wilk test was conducted to check the normality of both simulated and observed yields of all crops, individually and collectively. It is worth noticing that the classical t-test is only applicable if the simulated and the observed yields have the same variance. Otherwise the degree of freedom of yield samples has to be approximated using the Welch (or Satterthwaite) alternative (R Development Core Team, 2007, pp38-39). Different t-test algorithms were thus applied to calculate the t-statistic and the p-value depending on the equality of the variances, given by the F-test, of the simulated and the observed yields. The results (Table 6.9) showed that the observed and simulated yields of all food crops were indeed normally distributed. The same held true for the average yields of all grain and food crops taken together. It was confirmed that the simulated yields had the same means as the observed ones, either individually (as rice, wheat, maize, etc., but not sorghum/millet) or collectively (as grain and food).

However, the simulated yield of sorghum/millet was found statistically different from its observed counterpart. This can be explained by the residual effects in observed yield
data. The grain crops were defined in Chapter 3 as rice, wheat, maize and a batch of miscellaneous staple crops represented by sorghum/millet. The observed yield of sorghum/millet \(Y\) was indirectly calculated from the sown area and production of grain crops (Tables 3.1 and 3.5) using the following differential equation:

\[
Y = \frac{P_{\text{grain}} - (P_{\text{rice}} + P_{\text{wheat}} + P_{\text{maize}})}{A_{\text{grain}} - (A_{\text{rice}} + A_{\text{wheat}} + A_{\text{maize}})}
\] 

where \(P_{\text{rice}}, P_{\text{wheat}}, P_{\text{maize}}\) and \(P_{\text{grain}}\) are the production of rice, wheat, maize and grain crops in total, respectively; \(A_{\text{rice}}, A_{\text{wheat}}, A_{\text{maize}}\) and \(A_{\text{grain}}\) are the sown areas of rice, wheat, maize and grain crops in total, respectively. Errors that occurred in estimating and aggregating not only the production but also the sown area of grain crops are all propagated (to the provincial level) and attributed to the derived yield of sorghum/millet.

Nevertheless, due to the unimportance of sorghum and millet in food production (3.8% of total sown area and 2.2% of total production in 2005), the biased estimation of the yield of sorghum and millet did not contribute much to the overall difference between the observed and simulated yields of grain or food crops. In other words, the simulated yields are still statistically proven to be representative to the observed ones, and vice versa.

Table 6.9: Results of the Shapiro-Wilk, F- and paired t-tests on simulated and observed yields of major food crops

<table>
<thead>
<tr>
<th>Crop</th>
<th>Shapiro-Wilk Test (H(_0): normal distribution)</th>
<th>F-Test (H(_0): same variance)</th>
<th>Paired t-Test (H(_0): same mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Observed (W(p)) H(_0)</td>
<td>Simulated (W(p)) H(_0)</td>
<td>(F(p)) H(_0)</td>
</tr>
<tr>
<td>Rice</td>
<td>0.96 (..)† T</td>
<td>0.93 (..) T</td>
<td>0.40 (*) F</td>
</tr>
<tr>
<td>Wheat</td>
<td>0.93 (..) T</td>
<td>0.92 (..) T</td>
<td>0.36 (*) F</td>
</tr>
<tr>
<td>Maize</td>
<td>0.98 (..) T</td>
<td>0.95 (..) T</td>
<td>0.46 (..) T</td>
</tr>
<tr>
<td>SM♭</td>
<td>0.96 (..) T</td>
<td>0.94 (..) T</td>
<td>0.58 (..) T</td>
</tr>
<tr>
<td>Grain</td>
<td>0.98 (..) T</td>
<td>0.97 (..) T</td>
<td>0.29 (***) F</td>
</tr>
<tr>
<td>Beans</td>
<td>0.96 (..) T</td>
<td>0.96 (..) T</td>
<td>0.23 (***) F</td>
</tr>
<tr>
<td>Tubers</td>
<td>0.95 (..) T</td>
<td>0.97 (..) T</td>
<td>0.77 (..) T</td>
</tr>
<tr>
<td>Food</td>
<td>0.94 (..) T</td>
<td>0.96 (..) T</td>
<td>0.22 (***) F</td>
</tr>
</tbody>
</table>

† Significance levels: \(\geq 0.1(\cdot\cdot)\); \(> 0.05(.)\); \(\leq 0.05(*)\); \(\leq 0.01(**)\); \(\leq 0.001(***)\)
‡ T: true; F: false
♭ Sorghum + millet
Chapter 7

Soil Index-Management Index-Crop Yield Relationships

7.1 Introduction

Climate, crop, soil and management are the four most important factors that affect crop yield. Among them, the effects of soil characteristics and management practices can be assessed using the so-called “soil limitations evaluation” approach, which adopts the soil index (§4.6.1) to represent the limited soil fertilities and the management index (§4.6.2) to represent the factor input intensity and efficiency. Although the evaluation procedures (“how to do?”) have already been detailed in previous chapters, the question “why this way?” has not been answered. Therefore, the soil index-management index-crop yield relationships were statistically analyzed in this chapter in an attempt to reveal the rationale of the land limitations evaluation.

7.2 Soil characteristic-crop yield relationship

7.2.1 Regression analysis

An attempt was made to correlate individual soil characteristics (Ye et al., 2008), including CaCO$_3$, gypsum, CEC, BS, pH, SOC, ECe and ESP, to the observed crop yield using scatter plots (Figures 7.1 and 7.2). The dotted lines of the linear trend together with the best-fitting curves provide an overview of the relationship between a soil characteristic and the yield of a specific crop. Although some cases of close correlations are observed, for example, between CEC-soil and rice yield (Figure 7.1) and between ECe and maize yield (Figure 7.2), the scatter plots suggest a much weaker correlation between soil characteristics and crop yields, as confirmed by the regression analysis below.

Single factor linear regression models were applied to reveal the relationships between a soil characteristic as the factor and the yield of a specific crop as the dependent. Results (Table 7.1) show that the crop yields are barely explained by individual soil characteristics. In the best case only 21% of the variance of the rice yield can be explained by the base saturation; 41% of the variance of the wheat yield by the base saturation or the exchangeable sodium percentage; 60%, 41% and 35% of the variance of the maize yield by the electrical conductivity, the gypsum content and the exchangeable sodium percentage, respectively; slightly over 30% of the variance of the average yield of all gain or food crops by the organic carbon content. The rest of all the soil characteristic-crop yield relationships are
7.2 Soil-yield relationship

Figure 7.1: Soil characteristic (CaCO$_3$, gypsum, CEC and BS)-crop yield relationships. Yield is plotted on the y-axis in t ha$^{-1}$; soil characteristics are on the x-axis: CaCO$_3$ and gypsum: w%, CEC: cmol(+) kg$^{-1}$ soil, BS: %. Dotted lines are linear trends. Solid curves are fitted using a piecewise polynomial algorithm embedded with R’s smooth function.

either too weak in correlation or the correlation is statistically insignificant, or both. It is clear that soil interacts with crops as a whole and there is no single soil characteristic alone correlating well with crop yields.

7.2.2 Analysis of variance

The variances of the yields of food crops were analyzed against the variances of the soil characteristics in trying to relate the variations in yield to soil values. Analysis of variance (ANOVA) was used for this purpose. The results given in Table 7.2 show that the variation in values of an individual soil parameter is occasionally capable of explaining the variation in crop yield as in the cases of, for example, the base saturation for yield of rice, wheat or maize; and the gypsum content, CEC-soil or organic carbon content for the overall yield of grain. However, the soil characteristics are much more frequently observed to be incapable of explaining the yield variations. For example, although the variation in soil
organic carbon content does explain the variation in the overall yield of grain, it does not explain yield variations for any individual crop. This applies to all soil properties. But all soil characteristics taken together are capable of explaining variations in yields of any crop, individually or collectively. This suggests that the limiting soil characteristics should not be separated in the soil limitations evaluation (§4.6.1). Instead, they should be treated as a whole.

7.3 Soil index-crop yield relationship

7.3.1 Regression analysis

The crop yield is better correlated to soil index (Equation 4.63) (Figure 7.3) than to individual soil characteristics (Figures 7.1 and 7.2). This observation is statistically confirmed by the single factor regression analysis (Table 7.3) based on soil index and crop yield data.
7.3 Soil index-crop yield relationship

Table 7.1: Summary of soil characteristic-crop yield regression

<table>
<thead>
<tr>
<th>Soil char.</th>
<th>Rice ( R^2 )</th>
<th>Wheat ( R^2 )</th>
<th>Maize ( R^2 )</th>
<th>Grain ( R^2 )</th>
<th>Food ( R^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaCO(_3)</td>
<td>0.04 ( .)</td>
<td>0.18 ( * )</td>
<td>0.22 ( ** )</td>
<td>0.09 ( .)</td>
<td>0.10 ( .)</td>
</tr>
<tr>
<td>Gypsum</td>
<td>0.07 ( .)</td>
<td>0.14 ( .)</td>
<td>0.41 ( *** )</td>
<td>0.09 ( .)</td>
<td>0.08 ( .)</td>
</tr>
<tr>
<td>CEC(_{soil})</td>
<td>0.17 ( * )</td>
<td>0.04 ( .)</td>
<td>0.04 ( .)</td>
<td>0.14 ( * )</td>
<td>0.07 ( .)</td>
</tr>
<tr>
<td>BS</td>
<td>0.21 ( ** )</td>
<td>0.41 ( *** )</td>
<td>0.21 ( * )</td>
<td>0.07 ( .)</td>
<td>0.05 ( .)</td>
</tr>
<tr>
<td>pH-H(_2)O</td>
<td>0.14 ( .)</td>
<td>0.31 ( ** )</td>
<td>0.08 ( .)</td>
<td>0.09 ( .)</td>
<td>0.07 ( .)</td>
</tr>
<tr>
<td>SOC</td>
<td>0.02 ( .)</td>
<td>0.05 ( .)</td>
<td>9( \times 10^{-5} ) ( .)</td>
<td>0.32 ( ** )</td>
<td>0.31 ( ** )</td>
</tr>
<tr>
<td>ECE</td>
<td>0.13 ( .)</td>
<td>0.21 ( * )</td>
<td>0.60 ( *** )</td>
<td>0.04 ( .)</td>
<td>0.02 ( .)</td>
</tr>
<tr>
<td>ESP</td>
<td>0.02 ( .)</td>
<td>0.41 ( *** )</td>
<td>0.35 ( ** )</td>
<td>0.08 ( .)</td>
<td>0.09 ( .)</td>
</tr>
</tbody>
</table>

\( \dagger \) Significance levels: \( \geq 0.1(\ldots) ; > 0.05(\ldots) ; \leq 0.05(*) ; \leq 0.01(**) ; \leq 0.001(***) \)

Table 7.2: Summary of soil characteristic-crop yield variance analysis

<table>
<thead>
<tr>
<th>Soil char.</th>
<th>Rice</th>
<th>Wheat</th>
<th>Maize</th>
<th>Grain</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaCO(_3)</td>
<td>0.13 ( \dagger ) ( \ldots )</td>
<td>6.44 ( * )</td>
<td>8.69 ( ** )</td>
<td>1.56 ( \ldots )</td>
<td>0.95 ( \ldots )</td>
</tr>
<tr>
<td>Gypsum</td>
<td>0.31 ( \ldots )</td>
<td>1.56 ( .)</td>
<td>5.19 ( * )</td>
<td>5.22 ( * )</td>
<td>3.43 ( .)</td>
</tr>
<tr>
<td>CEC(_{soil})</td>
<td>5.71 ( * )</td>
<td>0.98 ( \ldots )</td>
<td>0.98 ( \ldots )</td>
<td>4.50 ( * )</td>
<td>2.03 ( \ldots )</td>
</tr>
<tr>
<td>BS</td>
<td>7.03 ( ** )</td>
<td>17.30 ( ** )</td>
<td>7.29 ( ** )</td>
<td>2.23 ( \ldots )</td>
<td>4.61 ( \ldots )</td>
</tr>
<tr>
<td>pH-H(_2)O</td>
<td>4.32 ( * )</td>
<td>11.41 ( ** )</td>
<td>2.44 ( .)</td>
<td>2.89 ( .)</td>
<td>1.96 ( .)</td>
</tr>
<tr>
<td>SOC</td>
<td>1.44 ( .)</td>
<td>0.08 ( .)</td>
<td>0.04 ( .)</td>
<td>7.59 ( ** )</td>
<td>2.31 ( .)</td>
</tr>
<tr>
<td>ECE</td>
<td>1.47 ( .)</td>
<td>4.12 ( * )</td>
<td>15.17 ( *** )</td>
<td>0.09 ( .)</td>
<td>0.03 ( .)</td>
</tr>
<tr>
<td>ESP</td>
<td>1.71 ( .)</td>
<td>8.38 ( .)</td>
<td>5.38 ( .)</td>
<td>2.51 ( .)</td>
<td>2.48 ( .)</td>
</tr>
</tbody>
</table>

\( \dagger \) Aggregated \( 4.89E29(*** \) \( 7.78E30(*** \) \( 1.87E30(*** \) \( 6.95E28(*** \) \( 9.44E28(*** \)

\( \dagger \) The F-statistic value. Notation 4.89\( \times 10^{29} \) stands for \( 4.89 \times 10^{29} \)

\( \ddagger \) Significance level: \( \geq 0.1(\ldots) ; > 0.05(\ldots) ; \leq 0.05(*) ; \leq 0.01(**) ; \leq 0.001(***) \)

aggregated at the provincial scale. The variance of the rice yield, for instance, was barely explained by any individual soil parameter (Table 7.1). In the best case, only 21% of the variance of the rice yield was explained by the base saturation. But 75% of the variances of the yields of the early, intermediate and late rice are significantly explained by the soil index alone. There are 6 out of 12 crop species or groups in which the variance of the crop yield is explained by the soil index with \( R^2 > 0.20 \) and \( p < 0.10 \), which gives an explainability of 50%. In the former case, there were 12 out of 40 occasions in which the variance of the crop yield was explained by a single soil characteristic with the same criterion of \( R^2 \) and \( p \) level. This corresponded to an explainability of 30%. In other words, the variance of the crop yield is 67% more predictable by the soil index than by a single soil characteristic.

7.3.2 Analysis of variance

ANOVA was applied to relate the variations in crop yield to variations in the soil index (Table 7.4). Similar results were obtained compared to the regression analysis discussed above.

Both the regression and the variance analysis confirmed that a significant improvement
Figure 7.3: Soil index-crop yield relationship for major crop species and groups. The degree of (dis)agreement between the dotted line of linear trend and the best-fitting curve indicates the extent of the (un)correlation between the soil index and the crop yield. The best-fitted curve is produced by a piecewise polynomial algorithm embedded with R’s smooth function in yield prediction was achieved using the soil index as the predicting variable. This suggests that the limiting soil characteristics should be collectively evaluated to give a single soil index, as already practised in §4.6.1.

Despite the observed improvement in yield prediction, the soil index alone does not completely explain the yields of food crops, individually or collectively. More factors should be included until the yields of all crop species and groups are statistically explained.

### 7.4 Management index-crop yield relationship

#### 7.4.1 Regression analysis

The crop-specific yield records were analyzed against the management index values using scatter plots (Figure 7.4). A close match between the best-fitting curves (black lines) and the linear trends (dotted lines) was observed, revealing the representability of the definition of the input levels over the food production reality across the east-middle-west economic development belts. This observation is confirmed by the statistical analysis given below.

Single factor linear regression models were fitted between the management index (as
the variable) and the crop yield (as the dependent). Close correlations are found between them, as suggested by the coefficient of determination ($R^2$) and the significance level (Table 7.5). On average, 25% of the variance of the yields of rice species are explained by the management index alone with a significance level of $p < 0.10$. But only 14% of the variance of the weighted average yield of rice is explained by the management index, suggesting that the management index, based on the characterization of input levels, performed better for individual crop species than for a crop group. The variance of the yields of all other crop species and crop groups are well represented by the management
7.4 Management index-crop yield relationship

Figure 7.4: Management index-crop yield relationship for major crop species and groups. The degree of (dis)agreement between the dotted line of linear trend and the best-fitting curve indicates the extent of the (un)correlation between the soil index and the crop yield. The best-fitted curve is produced by a piecewise polynomial algorithm embedded with R’s smooth function.

index: 68% of the variance of the yield of all grain crops taken together, for instance, is explained by the management index at a significance level of \( p < 0.001 \).

Table 7.5: Summary of the management index-crop yield regression

<table>
<thead>
<tr>
<th>Crop</th>
<th>( R^2 )</th>
<th>p-value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early rice</td>
<td>0.28</td>
<td>0.024</td>
<td>*†</td>
</tr>
<tr>
<td>Intermediate rice</td>
<td>0.20</td>
<td>0.080</td>
<td>.</td>
</tr>
<tr>
<td>Late rice</td>
<td>0.27</td>
<td>0.026</td>
<td>*</td>
</tr>
<tr>
<td>Rice</td>
<td>0.14</td>
<td>0.096</td>
<td>.</td>
</tr>
<tr>
<td>Spring wheat</td>
<td>0.39</td>
<td>0.018</td>
<td>*</td>
</tr>
<tr>
<td>Winter wheat</td>
<td>0.27</td>
<td>5.32\times10^{-3}</td>
<td>**</td>
</tr>
<tr>
<td>Wheat</td>
<td>0.26</td>
<td>4.17\times10^{-3}</td>
<td>**</td>
</tr>
<tr>
<td>Spring maize</td>
<td>0.63</td>
<td>4.07\times10^{-4}</td>
<td>***</td>
</tr>
<tr>
<td>Summer maize</td>
<td>0.56</td>
<td>1.42\times10^{-3}</td>
<td>***</td>
</tr>
<tr>
<td>Maize</td>
<td>0.48</td>
<td>6.47\times10^{-5}</td>
<td>***</td>
</tr>
<tr>
<td>Sorghum</td>
<td>0.60</td>
<td>1.20\times10^{-6}</td>
<td>***</td>
</tr>
<tr>
<td>Grain</td>
<td>0.68</td>
<td>4.56\times10^{-5}</td>
<td>***</td>
</tr>
<tr>
<td>Beans</td>
<td>0.39</td>
<td>2.53\times10^{-4}</td>
<td>***</td>
</tr>
<tr>
<td>Tubers</td>
<td>0.45</td>
<td>5.41\times10^{-5}</td>
<td>***</td>
</tr>
<tr>
<td>Food</td>
<td>0.57</td>
<td>1.32\times10^{-6}</td>
<td>***</td>
</tr>
</tbody>
</table>

† Significance level: \( \geq 0.1(\ldots); > 0.05(\ldots); \leq 0.05(*)\); \( \leq 0.01(**); \leq 0.001(***) \)
7.4.2 Analysis of variance

The results of the management index-crop yield ANOVA (Table 7.6) reveal that the variations in yields of food crops, individually or collectively, are all significantly \((p < 0.05)\) explained by the variations in the management index values. This, as well as the management index-crop yield regression, validates the definition of factor input levels (§6.3) according to the east-middle-west economic development belts and the crop-specific management index assessment scheme exercised in §4.6.2.

Table 7.6: Summary of management index-crop yield variance analysis

<table>
<thead>
<tr>
<th>Crop</th>
<th>F-statistic</th>
<th>p-value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early rice</td>
<td>7.39</td>
<td>1.59 × 10^{-2}</td>
<td>*†</td>
</tr>
<tr>
<td>Intermediate rice</td>
<td>7.87</td>
<td>1.49 × 10^{-2}</td>
<td>*</td>
</tr>
<tr>
<td>Late rice</td>
<td>7.14</td>
<td>1.74 × 10^{-2}</td>
<td>*</td>
</tr>
<tr>
<td>Rice</td>
<td>9.67</td>
<td>5.52 × 10^{-3}</td>
<td>**</td>
</tr>
<tr>
<td>Spring wheat</td>
<td>8.47</td>
<td>1.42 × 10^{-2}</td>
<td>*</td>
</tr>
<tr>
<td>Winter wheat</td>
<td>9.00</td>
<td>6.22 × 10^{-3}</td>
<td>**</td>
</tr>
<tr>
<td>Wheat</td>
<td>9.39</td>
<td>4.91 × 10^{-3}</td>
<td>**</td>
</tr>
<tr>
<td>Spring maize</td>
<td>21.22</td>
<td>6.36 × 10^{-4}</td>
<td>***</td>
</tr>
<tr>
<td>Summer maize</td>
<td>24.60</td>
<td>1.34 × 10^{-4}</td>
<td>***</td>
</tr>
<tr>
<td>Maize</td>
<td>22.03</td>
<td>9.05 × 10^{-5}</td>
<td>***</td>
</tr>
<tr>
<td>Sorghum</td>
<td>39.74</td>
<td>1.35 × 10^{-6}</td>
<td>***</td>
</tr>
<tr>
<td>Grain</td>
<td>43.04</td>
<td>4.90 × 10^{-7}</td>
<td>***</td>
</tr>
<tr>
<td>Beans</td>
<td>19.22</td>
<td>1.59 × 10^{-4}</td>
<td>***</td>
</tr>
<tr>
<td>Tubers</td>
<td>24.94</td>
<td>3.06 × 10^{-5}</td>
<td>***</td>
</tr>
<tr>
<td>Food</td>
<td>36.31</td>
<td>1.98 × 10^{-6}</td>
<td>***</td>
</tr>
</tbody>
</table>

† Significance level: \(≥ 0.1(\cdot); > 0.05(\cdot); ≤ 0.05(\ast); ≤ 0.01(\ast\ast); ≤ 0.001(\ast\ast\ast)\)

7.5 Soil index-management index-crop yield relationships

7.5.1 Analysis of variance

The variances of the yields of food crops were analyzed against the variances of both the soil index and the management index, as advised earlier (§7.3), in trying to gain explainability on crop yield over the soil index. Results (Table 7.7) show that the yield variations are explained more significantly by variations of the soil and management indices. Improvements in significance levels of nine crop species and groups (spring/winter/average wheat, spring/summer/average maize, sorghum, average grain and food) are observed. The significance level of the average grain, for instance, is raised from \(p > 0.1\) to \(p < 0.001\). However, the significance levels of two crops (average rice and beans) are adversely affected by the inclusion of the management index in the analysis, both one level downgraded from \(p < 0.001\) to \(p < 0.05\). The overall significance of the two factor ANOVA is mostly (60% occurrence) at \(p < 0.001\), with exceptions at \(p < 0.01\) (20%), \(p < 0.05\) (6.7%) and \(p < 0.1\) (13.3%).
Table 7.7: Summary of soil index-management index-crop yield variance analysis

<table>
<thead>
<tr>
<th>Crop</th>
<th>F-statistic</th>
<th>p-value</th>
<th>Significance</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early rice</td>
<td>31.37</td>
<td>3.13×10⁻⁶</td>
<td>***†</td>
<td>0†</td>
</tr>
<tr>
<td>Intermediate rice</td>
<td>24.48</td>
<td>2.94×10⁻⁵</td>
<td>***</td>
<td>0</td>
</tr>
<tr>
<td>Late rice</td>
<td>30.67</td>
<td>3.77×10⁻⁶</td>
<td>***</td>
<td>0</td>
</tr>
<tr>
<td>Rice</td>
<td>8.36</td>
<td>5.99×10⁻³</td>
<td>**</td>
<td>-1</td>
</tr>
<tr>
<td>Spring wheat</td>
<td>4.35</td>
<td>4.73×10⁻²</td>
<td>*</td>
<td>2</td>
</tr>
<tr>
<td>Winter wheat</td>
<td>2.80</td>
<td>0.10</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>Wheat</td>
<td>3.26</td>
<td>0.09</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>Spring maize</td>
<td>13.59</td>
<td>1.01×10⁻³</td>
<td>**</td>
<td>2</td>
</tr>
<tr>
<td>Summer maize</td>
<td>13.21</td>
<td>8.41×10⁻⁴</td>
<td>***</td>
<td>4</td>
</tr>
<tr>
<td>Maize</td>
<td>15.27</td>
<td>2.76×10⁻⁴</td>
<td>***</td>
<td>3</td>
</tr>
<tr>
<td>Sorghum</td>
<td>15.77</td>
<td>2.17×10⁻⁴</td>
<td>***</td>
<td>4</td>
</tr>
<tr>
<td>Grain</td>
<td>13.72</td>
<td>5.78×10⁻⁴</td>
<td>***</td>
<td>4</td>
</tr>
<tr>
<td>Beans</td>
<td>8.73</td>
<td>4.54×10⁻³</td>
<td>**</td>
<td>-1</td>
</tr>
<tr>
<td>Tubers</td>
<td>24.99</td>
<td>5.81×10⁻⁶</td>
<td>***</td>
<td>0</td>
</tr>
<tr>
<td>Food</td>
<td>14.71</td>
<td>3.16×10⁻⁴</td>
<td>***</td>
<td>4</td>
</tr>
</tbody>
</table>

† Significance level: ≥ 0.1(·); > 0.05(·); ≤ 0.05(*); ≤ 0.01(**); ≤ 0.001(***)
‡ Improvement of significance levels: > 0: upgraded; < 0: downgraded; 0: no improvement observed

7.5.2 Regression analysis

Two-factor linear regression models were fitted between the crop yield (as the dependent factor) and the soil and management indices (as the predicting factors). Very close correlations were found between the former and the latter, as suggested by the coefficient of determination (R²) and the significance level (Table 7.8). For grain crops, 68% of the variance of the overall yield is predicted by the soil and management indices together at a significance level of p < 0.001. For food crops, 58% of the variance of the overall yield is predicted at the same significance level. Furthermore, the predictability of crop yield is greatly improved (Figure 7.5) upon inclusion of the management index into the analysis. The percentage of the variance of the yield of grain and food crops, explained by the soil and management indices together, is 6.7 and 7.9 times higher (Table 7.8) than by the soil index alone (Table 7.3). Only one crop (spring wheat) is correlated to the soil and management indices at a significance level of p < 0.10. All the other crops are correlated at least with p < 0.05.

The implication of the improvement gained on yield predictability by joining the management and the soil indices is that the soil index and the management index should be used together in yield prediction. The performance is much lower if they are used separately. This positively validates Equation 4.62 using a large-scale yield dataset aggregated at the provincial scale in China (Ye et al., 2007).
Table 7.8: Summary of soil index-management index-crop yield regression

<table>
<thead>
<tr>
<th>Crop</th>
<th>( R^2 )</th>
<th>( p )-value</th>
<th>Significance</th>
<th>Improvement ( R^2 ) (%)</th>
<th>( p ) (grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early rice</td>
<td>0.79</td>
<td>(1.60 \times 10^{-5})</td>
<td>***†</td>
<td>5</td>
<td>0†</td>
</tr>
<tr>
<td>Intermediate rice</td>
<td>0.76</td>
<td>(1.70 \times 10^{-4})</td>
<td>***</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Late rice</td>
<td>0.78</td>
<td>(2.28 \times 10^{-5})</td>
<td>***</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Rice</td>
<td>0.41</td>
<td>(8.43 \times 10^{-3})</td>
<td>**</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>Spring wheat</td>
<td>0.44</td>
<td>(9.63 \times 10^{-2})</td>
<td>.</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>Winter wheat</td>
<td>0.28</td>
<td>(2.05 \times 10^{-2})</td>
<td>*</td>
<td>2,724</td>
<td>2</td>
</tr>
<tr>
<td>Wheat</td>
<td>0.26</td>
<td>(1.77 \times 10^{-2})</td>
<td>*</td>
<td>9,838</td>
<td>2</td>
</tr>
<tr>
<td>Spring maize</td>
<td>0.73</td>
<td>(3.84 \times 10^{-4})</td>
<td>**</td>
<td>225</td>
<td>3</td>
</tr>
<tr>
<td>Summer maize</td>
<td>0.47</td>
<td>(7.09 \times 10^{-4})</td>
<td>***</td>
<td>396</td>
<td>4</td>
</tr>
<tr>
<td>Maize</td>
<td>0.51</td>
<td>(1.87 \times 10^{-4})</td>
<td>***</td>
<td>393</td>
<td>4</td>
</tr>
<tr>
<td>Sorghum</td>
<td>0.61</td>
<td>(8.33 \times 10^{-6})</td>
<td>***</td>
<td>2,045</td>
<td>4</td>
</tr>
<tr>
<td>Grain</td>
<td>0.68</td>
<td>(2.05 \times 10^{-4})</td>
<td>***</td>
<td>671</td>
<td>4</td>
</tr>
<tr>
<td>Beans</td>
<td>0.53</td>
<td>(3.57 \times 10^{-5})</td>
<td>***</td>
<td>82</td>
<td>1</td>
</tr>
<tr>
<td>Tubers</td>
<td>0.62</td>
<td>(2.43 \times 10^{-6})</td>
<td>***</td>
<td>97</td>
<td>1</td>
</tr>
<tr>
<td>Food</td>
<td>0.58</td>
<td>(8.12 \times 10^{-6})</td>
<td>***</td>
<td>794</td>
<td>4</td>
</tr>
</tbody>
</table>

† Significance level: \( \geq 0.1(\cdot); > 0.05(\cdot); \leq 0.05(\cdot); \leq 0.01(\cdot); \leq 0.001(\cdot)\)
‡ Improvement of significance levels: > 0: upgraded; < 0: downgraded; 0: no improvement observed

Figure 7.5: Improvement of the yield predictability with soil and management indices together over soil index alone as the predicting variable(s)
Chapter 8

Long-term Food Producing Capacities

8.1 Introduction

China is a country in transition. In past few decades, China has undergone a fundamental shift from a centrally-planned economy to a market-oriented one, and from a largely rural-based society to an increasingly urbanized one. As a result of the increase in labor productivity, hundreds of millions of the rural labors have been released into the urban area and subsequently raised the food demand to a higher level. The recent disequilibrium between food supply and food demand during 1996-2003 signaled a loud alert, for yet another time, on the food security issue of China (Lal, 2007).

In this chapter, projections of China’s food producing capacities in 2030-2050 are made, based on trend analysis and scenario settings of the population growth, urbanization rate, cropland availability, cropping intensity and soil degradation. A food security index (FSI) is proposed at the end of this chapter to reveal the historical fluctuations and future trends of food security in China. Policy recommendations on food security in China will be discussed in Chapter 9.

8.2 Population

China is the world’s most populous country. The population size had been tripled during the 20th century. There were 430 million people living within the Chinese territory in 1900, 540 million in 1950, 1,267 million in 2000 and 1,308 million in 2005 (Table 8.1). Many projections, concerning the future trend of the population growth in China, have been made (Durand, 1961; Zeng and Vaupel, 1989; Wang, 1999). The widely accepted consensus about the growth pattern is that the population is about to grow at a decreasing rate before it reaches its apex in mid-21st century. The UNFPA (2006)’s projection is one of the most recent. According to it, China's population will reach the apex of 1.46 billion in 2033 – this apex comes years earlier than previously projected – before it starts declining. The population in 2050, for instance, will drop to 1.44 billion; and in 2060 to 1.42 billion (Figure 8.1).
Table 8.1: Chinese population during the 20th century

<table>
<thead>
<tr>
<th>Year</th>
<th>Pop†</th>
<th>Year</th>
<th>Pop</th>
<th>Year</th>
<th>Pop</th>
<th>Year</th>
<th>Pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>430</td>
<td>1978</td>
<td>963</td>
<td>1988</td>
<td>1110</td>
<td>1997</td>
<td>1236</td>
</tr>
<tr>
<td>1952</td>
<td>575</td>
<td>1982</td>
<td>1017</td>
<td>1991</td>
<td>1158</td>
<td>2000</td>
<td>1267</td>
</tr>
<tr>
<td>1957</td>
<td>647</td>
<td>1983</td>
<td>1030</td>
<td>1992</td>
<td>1172</td>
<td>2001</td>
<td>1276</td>
</tr>
<tr>
<td>1962</td>
<td>673</td>
<td>1984</td>
<td>1044</td>
<td>1993</td>
<td>1185</td>
<td>2002</td>
<td>1285</td>
</tr>
</tbody>
</table>

† Pop = Population (in million); Source: NBSC (2005)

8.2.1 Normative analysis

The census data closely follows a logistic growth curve. It is interesting to observe that the points of the UNFPA (2006) projection, as shown in Figure 8.1 as blank squares, fit into the curve as well, at least before the year 2033. The disagreement appears only after the apex point is passed. Some people believe that the population size stays at the peak level for a certain period of time before declining, as in the case of a logistic curve (Durand, 1961). Others believe that the population is about to decrease immediately after reaching the apex size (UNFPA, 2006). Which picture is more realistic? The effects of the population control policy – or the so-called “one child policy” – may provide a clue.
Population control has been implemented since the early 1980s. It caused the aggregate population growth rate to be abnormally lower than that of an uncontrolled, naturally growing population. The growth rate was 22% in 1950s and 19% in 1970s. The difference was only 3% over a 20-year period. But after the population control policy came into force, the growth rate declined as 3 times as quickly as in previous decades: the growth rate difference was 5% between 1970s and 1980s. For the first 5 years in the 21st century, the average aggregate growth rate of the population was as low as 6.6%: a 4.4% decrease from 11% in 1990s. The trend is so obvious that the population is likely to experience a negative growth soon after the balance point is passed. In other words, the logistic growth model misapplies after 2033 (Wang, 1999).

8.2.2 Rural-urban population composition

The image of population growth in China is not complete if the relative composition of the population is unknown (Gawthrop and Shen, 1995). Historically, the Chinese civilization has been an agricultural-centric one, with the majority of the population living in the countryside. In the very beginning of the rural reforms – which is still an on-going process to date – in 1978, 82% of the entire population were rural residents. By 2005, the figure had dropped to 57% (Figure 8.2). Accordingly, the urban population experienced a 390 million net increase during the same period (NBSC, 2005), compared with 315 million of the entire population in the Euro Zone (Eurostat, 2005). The mass migration of the rural population to the urban areas has been described as the largest of its kind ever observed in human history (Zhang and Song, 2003). The shrinkage of per capita cropland area and/or income disparities between the urban and the rural sectors, among others, are two examples of the driving forces (Zhao, 2005; Chen, 2006).

Figure 8.2: Relative rural-urban population composition in China, 1978-2005
### 8.2 Population

#### 8.2.3 Urbanization: 1978-2005

The issue of China’s urbanization has been of longstanding academic interest (Schnore, 1961; Barclay and Solomane, 1996; Zhao, 2005), especially when it is linked to soil protection and food security (Change and Brada, 2006; Chen, 2007). Urbanization, referring to a growth in the proportion of a population living in urban areas, is one of the major social changes interacting with the environmental change. The urbanization rate, or the share of urban population in the entire population, increased from 18% in 1978 to 43% in 2005, with an average acceleration of nearly 1% per year. The growth pattern of the urbanization rate in China during 1978-2005 can be better modeled by multiple linear relationships, each fitted to a distinct section of the entire period, than a single relationship.

The linear model \( y = 0.7691x - 1503 \) has been fitted to data points between 1978 and 1989 with \( R^2 = 0.9956 \) at significance level \( p < 0.01 \), where \( y \) represents the urbanization rate and \( x \) represents the year. The annual urbanization growth rate is given by \( x \)'s coefficient 0.7691, meaning that urbanization grows constantly at a rate of 7.7% during the section of 1978-1989. Similarly, another two linear relationships were fitted to the other two sections of 1989-1995 and 1995-2005. The annual urbanization growth rate during 1989-1995 and 1995-2005 are 4.9% and 14.3%, respectively (Figure 8.3a).

It is generally accepted that the urbanization process follows a low-high-low growth pattern (Button, 1976; Kojima, 1995; Antrop, 2004; Angel et al., 2005), which produces a S-shaped curve. Turning points A and B divide the curve into three phases: the first low phase, the high phase and the second low phase (Figure 8.3b). The urbanization rate in the first low phase is less than or equal to 10%, while in the second low phase the urbanization rate is greater than or equal to 60%. In fact, China has passed point
8.3 Loss of cropland

A in 1975 and is now approaching to point C within the high phase. Point C marks the entrance to an urbanized society, which by definition (UNFPA, 2006) possesses 45% of the population living in urban areas. Beyond point B, the urbanization rate is about to be eventually stabilized.

This generalized low-high-low urbanization pattern will be used to predict the future trends of per capita food demand in §8.7.

8.3 Loss of cropland

Given the population size, China’s food production has been constrained by the scarcity of per capita cropland area. This basic characteristic reveals the very fact that loss of cropland draws profound influence on food security in China for decades, if not centuries, to come (Yang and Li, 2000). Despite the preciousness of cropland in China, systematic studies of it were few before 1980s. Part of the reason was the inadequacy and inaccuracy of the statistical data, as observed and indicated by many authors (Ash and Edmonds, 1998; Froliking et al., 1999; Smil, 1999a; Xiao et al., 2003; Liu et al., 2005a). Although satellite-born remote sensing has been recognized as an effective alternative data source to ground-based census (Frolking et al., 1999), uncertainties remain the primary obstacle for it to be used in small-scale applications (Xiao et al., 2003). Using multiple data sources, including the official statistics (NBSC, 1999), independent third-party estimations (WRI, 1987, 2005), researches (Feng et al., 2005), and remote sensing (Liu et al., 2005a), the time series of the cropland area in China since 1949 onwards are reconstructed, as given in Table 8.2.

The data given in Table 8.2 shows that: (1) before the rural reforms and the subsequent economic expansion, a net increase of cropland at an annual rate of 1.22 million hectares was found during the period of 1949-1979, despite the significant losses of cropland during the Great Leap Forward (GLF, at 2.12 million hectares per year between 1950 and 1960) period to non-agricultural purposes. In general, reclamation prevails urbanization as the primary factor on cropland area change during 1949-1979; (2) a loss of 0.26 million hectares of cropland was observed annually during 1979-1999; and (3) the situation deteriorates significantly since the year 2000. The cropland is being lost at a rate of 1.45 million hectares per year. Converting the marginal croplands, which were mostly reclaimed for agricultural use during the pre-reform period, to pasture, forest and wetland, known as the “Grain for Green” project in China (Yang, 2004; Xu et al., 2006), may partially explain the shrinkage of the cropland but can not be responsible for the full magnitude of it.

Rapid urbanization since early 1980s (see §8.2.3) is found to be another, yet much stronger, factor that casts an influence on cropland availability. A negative feedback was revealed between the urbanization rate and the cropland area by a single-factor regression analysis (Figure 8.4). Urbanization means not only the growth of the urban population but also the expansion of the urban area. On average, every percent increase of urbanization rate will cost 0.52 million hectares of cropland.

This observation will be included in scenario building (§8.7) to predict future cropland availabilities at different urbanization levels in the long run.
8.4 Soil degradation

Soil degradation is a major environmental threat to the sustainability and productive capacity of world agriculture (Larson et al., 1983; Frye et al., 1985; Blaschke et al., 2000; Wiebe, 2003; Karlen, 2004; Stocking, 2006; Montgomery, 2007), and thus received an increasing amount of attention either internationally (Crosson and Anderson, 1992; Lal, 1998; Conway and Toenniessen, 1999; Scherr, 1999; Eswaran et al., 2001; Tilman et al., 2002; Rosegrant and Cline, 2003; Bakker et al., 2004; McNeill and Winiwarter, 2004; Tan et al., 2005; Bakker et al., 2007; Swaminathan, 2007) or domestically in China (Tieh, 1941; Rozelle et al., 1997a,b; Wang and Gong, 1998; Liu, 1999; Lindert, 1999, 2000; Sun et al., 2003; Jiang et al., 2005; Yang et al., 2005; Chen et al., 2007). Globally nearly one-third of the arable land has been lost by erosion during the past 40 years and continues to be lost at a rate of more than 10 million hectares per year (Pimentel et al., 1995; Trimble and Crosson, 2000; Lal, 2007). A recent study by Wilkinson and McElroy (2007) estimates that soil loss from global cropland is currently running at a rate of more than 6 t ha$^{-1}$ yr$^{-1}$, which is more than 15 times the estimated average rate (0.42 t ha$^{-1}$ yr$^{-1}$) during the geological history of the earth. In China, efforts have been made to link soil degradation to the interacting issues of population pressure (Qu and Li, 1994; Brown, 1995; Pimentel et al., 1997; Rozelle et al., 1997a; Neupert, 1999; Chinese Academy of Sciences, 2000; Jiang et al., 2005; Gilland, 2006; Rain et al., 2007), rapid urbanization (Chen, 2002; Liu et al., 2005b; Change and Brada, 2006; Chen, 2006, 2007), and food security (Prosterman et al., 1996; Anderson and Peng, 1998; Yang and Li, 2000; Yang, 2004; Xu et al., 2006; Xiong

Table 8.2: Reconstructed time series of cropland area in China, in comparison to population size and urbanization rate on a yearly basis, 1949-2005

<table>
<thead>
<tr>
<th>Year</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>Year</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>Year</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1949</td>
<td>106.3</td>
<td>7.6</td>
<td>530</td>
<td>1968</td>
<td>123.6</td>
<td>13.9</td>
<td>774</td>
<td>1987</td>
<td>140.7</td>
<td>25.3</td>
<td>1096</td>
</tr>
<tr>
<td>1950</td>
<td>108.0</td>
<td>7.9</td>
<td>538</td>
<td>1969</td>
<td>123.4</td>
<td>14.4</td>
<td>792</td>
<td>1988</td>
<td>140.4</td>
<td>26.2</td>
<td>1110</td>
</tr>
<tr>
<td>1951</td>
<td>109.7</td>
<td>8.1</td>
<td>547</td>
<td>1970</td>
<td>125.3</td>
<td>14.8</td>
<td>810</td>
<td>1989</td>
<td>140.2</td>
<td>26.2</td>
<td>1127</td>
</tr>
<tr>
<td>1952</td>
<td>111.5</td>
<td>8.4</td>
<td>555</td>
<td>1971</td>
<td>127.3</td>
<td>15.3</td>
<td>826</td>
<td>1990</td>
<td>139.9</td>
<td>26.4</td>
<td>1143</td>
</tr>
<tr>
<td>1953</td>
<td>113.2</td>
<td>8.6</td>
<td>565</td>
<td>1972</td>
<td>129.2</td>
<td>15.8</td>
<td>842</td>
<td>1991</td>
<td>139.7</td>
<td>26.9</td>
<td>1158</td>
</tr>
<tr>
<td>1954</td>
<td>115.0</td>
<td>8.9</td>
<td>576</td>
<td>1973</td>
<td>131.1</td>
<td>16.3</td>
<td>858</td>
<td>1992</td>
<td>139.4</td>
<td>27.5</td>
<td>1172</td>
</tr>
<tr>
<td>1955</td>
<td>116.7</td>
<td>9.2</td>
<td>586</td>
<td>1974</td>
<td>133.1</td>
<td>16.8</td>
<td>874</td>
<td>1993</td>
<td>139.1</td>
<td>28.0</td>
<td>1185</td>
</tr>
<tr>
<td>1956</td>
<td>118.4</td>
<td>9.5</td>
<td>596</td>
<td>1975</td>
<td>135.0</td>
<td>17.4</td>
<td>890</td>
<td>1994</td>
<td>138.9</td>
<td>28.5</td>
<td>1199</td>
</tr>
<tr>
<td>1957</td>
<td>120.2</td>
<td>9.8</td>
<td>607</td>
<td>1976</td>
<td>137.0</td>
<td>17.9</td>
<td>908</td>
<td>1995</td>
<td>138.6</td>
<td>29.0</td>
<td>1211</td>
</tr>
<tr>
<td>1958</td>
<td>118.1</td>
<td>10.1</td>
<td>620</td>
<td>1977</td>
<td>138.9</td>
<td>18.5</td>
<td>925</td>
<td>1996</td>
<td>138.4</td>
<td>30.5</td>
<td>1227</td>
</tr>
<tr>
<td>1959</td>
<td>115.9</td>
<td>10.5</td>
<td>633</td>
<td>1978</td>
<td>140.8</td>
<td>17.9</td>
<td>943</td>
<td>1997</td>
<td>138.1</td>
<td>32.0</td>
<td>1238</td>
</tr>
<tr>
<td>1960</td>
<td>113.8</td>
<td>10.8</td>
<td>646</td>
<td>1979</td>
<td>142.8</td>
<td>19.7</td>
<td>960</td>
<td>1998</td>
<td>137.8</td>
<td>33.4</td>
<td>1250</td>
</tr>
<tr>
<td>1961</td>
<td>111.7</td>
<td>11.2</td>
<td>660</td>
<td>1980</td>
<td>142.5</td>
<td>19.4</td>
<td>978</td>
<td>1999</td>
<td>137.6</td>
<td>34.8</td>
<td>1260</td>
</tr>
<tr>
<td>1962</td>
<td>114.2</td>
<td>11.5</td>
<td>673</td>
<td>1981</td>
<td>142.3</td>
<td>21.0</td>
<td>1001</td>
<td>2000</td>
<td>136.1</td>
<td>36.2</td>
<td>1273</td>
</tr>
<tr>
<td>1963</td>
<td>116.6</td>
<td>11.9</td>
<td>689</td>
<td>1982</td>
<td>142.0</td>
<td>21.7</td>
<td>1017</td>
<td>2001</td>
<td>134.7</td>
<td>37.7</td>
<td>1288</td>
</tr>
<tr>
<td>1964</td>
<td>119.1</td>
<td>12.3</td>
<td>705</td>
<td>1983</td>
<td>141.7</td>
<td>22.4</td>
<td>1030</td>
<td>2002</td>
<td>133.2</td>
<td>39.1</td>
<td>1298</td>
</tr>
<tr>
<td>1965</td>
<td>121.5</td>
<td>12.7</td>
<td>720</td>
<td>1984</td>
<td>141.5</td>
<td>23.1</td>
<td>1047</td>
<td>2003</td>
<td>131.8</td>
<td>40.5</td>
<td>1309</td>
</tr>
<tr>
<td>1966</td>
<td>124.0</td>
<td>13.1</td>
<td>738</td>
<td>1985</td>
<td>141.2</td>
<td>23.7</td>
<td>1063</td>
<td>2004</td>
<td>130.3</td>
<td>41.8</td>
<td>1319</td>
</tr>
<tr>
<td>1967</td>
<td>123.8</td>
<td>13.5</td>
<td>756</td>
<td>1986</td>
<td>141.0</td>
<td>24.6</td>
<td>1078</td>
<td>2005</td>
<td>128.9</td>
<td>44.5</td>
<td>1328</td>
</tr>
</tbody>
</table>

(1) Cropland in million ha.; (2) Urbanization rate in %; (3) Population in million
8.4 Soil degradation

Soil degradation

20 25 30 35 40 45
130 132 134 136 138 140 142
Cropland (million ha.)
Urbanization Rate (%)
y = 153.8 − 0.52x
\( R^2 = 0.95 \)

Figure 8.4: Cropland area as explained by urbanization rate, 1980-2005. Blank circles represent the observation points; the line represents the linear relation between cropland area and urbanization rate. Regression equation obtained: \( y = 153.8 - 0.52x \), with \( R^2 = 0.95 \) at the highest significance level.

et al., 2007).

A big step forward was achieved in the early 1990s in understanding the extent and degree of soil degradation at the global scale, marked by the publication of the “world map of the status of human-induced soil degradation” or GLASOD (Oldeman et al., 1991). GLASOD was successful not only in enhancing the public awareness of soil degradation but also in informing the urgency of soil conservation to policy makers. However, this non-digital map at a scale of 1:10 million can hardly be used in any quantitative assessments of yield effects of soil degradation at the national scale. Although a post-publication, digitized version of GLASOD was eventually made available, lack of database support still largely limits its scope of applications. In catering the needs of an improved map of soil degradation at the regional or bigger scales, massive efforts in international cooperation amongst soil scientists across South and Southeast Asia, led by ISRIC, have produced an Asian variant of GLASOD – the ASSOD (van Lynden and Oldeman, 1997). The inventory of the type and extent of soil degradation is given in ASSOD at a scale of 1:5 million, which is much more detailed than in GLASOD. Nevertheless, the impact of soil degradation on crop productivity is still expressed in qualitative rather than quantitative terms (Grierson, 2000). Therefore, the effects of soil degradation on crop yield have to be quantified before the factor of soil degradation can be included in building the long-term production scenarios in China.

ASSOD assesses the impact of soil degradation on crop productivity by comparing the current average productivity to the average productivity in the non-degraded (or non-improved, where applicable) situation, which was 10 to 15 years earlier in time, and
classifies the impact into five classes ranging from “negligible” to “extreme” (Table 8.3).

Table 8.3: Impact class of soil degradation on crop yield adopted in ASSOD

<table>
<thead>
<tr>
<th>Code</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Negligible</td>
</tr>
<tr>
<td>1</td>
<td>Light</td>
</tr>
<tr>
<td>2</td>
<td>Moderate</td>
</tr>
<tr>
<td>3</td>
<td>Strong</td>
</tr>
<tr>
<td>4</td>
<td>Extreme</td>
</tr>
</tbody>
</table>

The extent of degradation is defined in ASSOD as the area percentage of the entire mapping unit which is affected by a certain type of degradation, rounded to the nearest 5% (Table 8.4). If more than one type (or subtype) of degradation is present, overlaps may exist between different (sub)types. Furthermore, each map unit which does not show a 100% extent for degradation includes by definition some stable and/or wasteland. Clearly, overlap does not occur here.

Table 8.4: Extent of soil degradation defined in ASSOD

<table>
<thead>
<tr>
<th>Code</th>
<th>Extent (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>75</td>
</tr>
</tbody>
</table>

Figure 8.5 shows the impact class of common (sub)types of soil degradation on crop yield in China. At the national scale, water erosion occurs dominantly in the southeast with moderate to strong influences on crop yield. Water erosion may also cause strong to extreme impacts on crop yield in hilly areas in the northwestern outskirts of the North China Plain and in the southern Tibetan Plateau (a). Wind erosion is found to dominate in the northwest (b), while deterioration of soil physical properties is primarily observed in the North China Plain, which has strong adverse effects on crop yield (c). Two subtypes of soil chemical deterioration are differentiated: fertility decline moderately affects a large part of the southeast (d), while salinisation lightly to moderately affects the North China Plain, the northeast and the irrigated farmlands in the middle reaches of the Yellow River Basin (e).

In order to derive a combined map of soil degradation from its (sub)types, an overall score is computed using the following formula:

\[ d = \sum_{i=1}^{5} (E_i \cdot I_i) \]  

(8.1)

where \( d \) is the overall score of soil degradation with regard to the extent and degree of degradation occurring within an entire mapping unit; \( E_i \) is the extent of degradation type \( i \), expressed as area percentage (%) within the mapping unit; and \( I_i \) is the code of the impact class of degradation type \( i \), already given in Table 8.3.
Figure 8.5: Impact class of common types of soil degradation on crop yield in China for (a) water erosion, (b) wind erosion, (c) soil physical deterioration, (d) fertility decline, and (e) salinisation
The overall score $d$ is reclassified into five impact classes, ranging from “negligible” to “extreme” (Table 8.5) and mapped for China in Figure 8.6. The new map provides a high-level overview of the limitation on crop production imposed by the extent and degree of soil degradation, and will be used as the basis of the quantitative assessment of the effects of soil degradation on crop yield below.

<table>
<thead>
<tr>
<th>Score Interval</th>
<th>Impact Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>Negligible</td>
</tr>
<tr>
<td>(0.00 - 1.25]</td>
<td>Light</td>
</tr>
<tr>
<td>(1.25 - 2.50]</td>
<td>Moderate</td>
</tr>
<tr>
<td>(2.50 - 3.75]</td>
<td>Strong</td>
</tr>
<tr>
<td>(3.75 - 5.00]</td>
<td>Extreme</td>
</tr>
</tbody>
</table>

Figure 8.6: Overall impact class of soil degradation on crop yield in China

### 8.4.1 Effects on crop yield

Soil degradation has a detrimental effect on soil quality for agricultural production because erosion degrades soil functions for crop growth such as the supply of water, nutrients and rooting space (Bakker et al., 2007). These effects have been demonstrated through
Soil degradation 120

numerous experiments conducted on plots where erosion was either simulated by artificial desurfacing (Mbagwu et al., 1984; Gollany et al., 1992; Tanaka, 1995; Larney et al., 2000; Oyedele and Aina, 2006), or by comparing yield on strongly eroded areas with yield on less eroded areas (Mielke and Schepers, 1986; Olson and Carmer, 1990; Kosmas et al., 2001). The reported results, however, show a wide variability. Pimentel et al. (1993) estimate, based on available secondary data, that global production is 10-15% lower as a result of all the various effects of soil erosion. Estimates of Buringh and Dudal (1987) are even higher. They predicted that, among other regions, erosion-induced soil nutrient depletion would result in a 20% to 36% yield loss in Southwest and Southeast Asia, respectively.

Other figures for the effects on global or regional productivity, based more on empirical evidence, are much lower (Table 8.6). Dregne and Chou (1992) estimate that the irrigated and rainfed lands in Asia have experienced at least a 10% and 25% loss in potential productivity, respectively. Using GLASOD data, Crosson (1995) estimates an aggregated global loss of 11.9-13.4% of agricultural supply, assuming a 15%, 35%, and 75% yield decline, respectively, for lightly, moderately and strongly degraded arable soils. Using FAO (1995) data, Lal (1998) estimates a 10% or 15% reduction of grain yield worldwide or in Asia, respectively, due to soil erosion.

Soil degradation research in China is found to focus overwhelmingly on the northwest (Yang et al., 2005; Chen et al., 2007). A recent study by Chen et al. (2003) estimates a 1.0-3.9% yield decrease per 1 cm loss of surface soil in the Loess Plateau. Generally, soil degradation in this erosion-prone region causes a 22-80% yield reduction for rainfed crops (Chen et al., 2004). The effect in the Loess Plateau is believed to be considerably more detrimental than elsewhere (Crosson, 1995; Bakker et al., 2004). Yield reductions at the field scale in Europe and North America under intense, mechanized agriculture (Bakker et al., 2007), for example, are of the order of 4% for each 10 cm of soil loss, which is one-tenth of the magnitude in the Loess Plateau. At the national scale in China, the effect is estimated to be more conservative. Based on elasticity analysis, Rozelle et al. (1997b) estimate that soil erosion and salinity cause a 5.8% and 0.1% decrease in grain yield, respectively, during the period of 1975-1990. This is in accordance with Crosson and Anderson (1992), who suggest that “productivity loss due to soil degradation is hardly greater than 5%”.

Table 8.6: Reported effects of soil degradation on crop yield

<table>
<thead>
<tr>
<th>Region</th>
<th>Yield Loss (%)</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td>15 - 30</td>
<td>Pimentel et al. (1993)</td>
</tr>
<tr>
<td>World</td>
<td>10 (light) - 35 (moderate) - 75 (strong)</td>
<td>Crosson (1995)</td>
</tr>
<tr>
<td>Asia</td>
<td>20 (southwest) - 36 (southeast)</td>
<td>Buringh and Dudal (1987)</td>
</tr>
<tr>
<td>Asia</td>
<td>10 (irrigated) - 25 (rainfed)</td>
<td>Dregne and Chou (1992)</td>
</tr>
<tr>
<td>China</td>
<td>5.8 (1975-1990)a</td>
<td>Rozelle et al. (1997b)</td>
</tr>
<tr>
<td>China</td>
<td>1.0 - 3.9 per 1 cm loss of surface soilb</td>
<td>Chen et al. (2003)</td>
</tr>
<tr>
<td>China</td>
<td>21.9 - 80 (rainfed)b</td>
<td>Chen et al. (2004)</td>
</tr>
</tbody>
</table>

*a national; b Loess Plateau

Great care should be taken when interpreting the impact class of soil degradation, as already shown in Figure 8.6, in terms of relative yield loss. The magnitude of yield
response to the level of degradation is closely related to the level of field management practices (van Lynden and Oldeman, 1997). A moderately degraded soil, for instance, may cause 50% yield loss at the low level of management. However, half of the lost yield can be avoided should the field management be practised at the intermediate level. Under the high management level, the adverse effect of soil degradation may be largely controlled. Under the worst scenario, an extremely degraded soil may be totally unproductive if poorly managed (i.e., at the low management level). But up to 50% of this detrimental effect would be controllable should the management be practised at high level. Table 8.7 summarizes the quantitative effects of soil degradation on crop yield as a function of the impact class of soil degradation (Figure 8.6) and the level of factor inputs (§4.6.2).

Table 8.7: Relative yield loss (%) due to soil degradation with regard to input levels

<table>
<thead>
<tr>
<th>Impact</th>
<th>Input Level</th>
<th>High</th>
<th>Intermediate</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negligible</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td>0</td>
<td>10</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Moderate</td>
<td>10</td>
<td>25</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Strong</td>
<td>25</td>
<td>50</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Extreme</td>
<td>50</td>
<td>75</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Consequently, the effect of soil degradation on crop yield is evaluated at the national scale, with regard to the characterization of the input levels for the east, middle and west socio-economic belts (§2.2.3), as shown in Figure 8.7. The influence of the management level on yield is obviously observed against the physical degradation conditions given in Figure 8.6. On average, 1.1%, 9.4% and 25.3% of crop yields were lost in 2005 due to soil degradation in the eastern, the middle and the western parts of the country, respectively. The relative yield loss at the national scale is therefore evaluated to be 11.2% (Table 8.8).

Table 8.8: Relative yield loss (%) due to soil degradation at regional and national scales

<table>
<thead>
<tr>
<th>SEB†</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>0</td>
<td>25</td>
<td>1.1</td>
</tr>
<tr>
<td>Middle</td>
<td>0</td>
<td>50</td>
<td>9.4</td>
</tr>
<tr>
<td>West</td>
<td>10</td>
<td>75</td>
<td>25.3</td>
</tr>
<tr>
<td>Country</td>
<td>0</td>
<td>75</td>
<td>11.2</td>
</tr>
</tbody>
</table>

† Social-economic belt (Figure 2.4)

8.5 Food production scenarios

8.5.1 Urbanization scenarios

As already discussed in §8.2.3, China has been experiencing rapid urbanization in the post-1978 period. A net increase of 10% in urbanization rate was observed during the last decade only (Figure 8.3). Following the UNFPA (2006) standard, China is about to be entitled as an “urbanized” country in a few years time by having more than 45% of the entire population living in urban areas. This will, on one hand, propel further economic growth
Figure 8.7: Effect of soil degradation at national scale in terms of relative yield loss (%) as influenced by management levels in the east, middle and west socio-economic belts. Darker shades represent heavier yield loss.

since urbanization plays a central role in the national development strategy (Change and Brada, 2006). But, on the other hand, urban expansion will cause more cropland to be transferred to non-agricultural uses (Figure 8.4).

Based on the generalized low-high-low pattern (Button, 1976; Kojima, 1995; Antrop, 2004; Angel et al., 2005), urbanization in China is most likely to retain its momentum to grow until the urbanization rate is slightly over 60% by 2020 (i.e., the turning point B in Figure 8.3). The average annual growth in urbanization rate is projected to decrease from 14% during 1995-2020 to 8% during 2020-2030. It will be further decreased to 5% during 2030-2050. In 2030 and 2050, the urbanization rate will reach the levels of 70% and 80%, respectively (Figure 8.8).

The likelihood of the scenarios can be revealed by comparing with the realized urbanization rates of other countries and country groups. The Euro Zone, for example, had reached an urbanization level of 70% in 1980s, which leads China for at least half a century, should China reach the projected level in 2030. The United States, as another example, owns similar advantage over China. The urbanization rate in the USA was as high as 80% in 2004, which China is projected to achieve by 2050. As early as in 1990, countries like the United Kingdom and Belgium fulfilled an urbanization rate of 89% and 96%, respectively, which China will unlikely reach in the coming century.
8.5 Food production scenarios

8.5.2 Cropland scenarios

Although many reasons have contributed to the loss of cropland in China (Smil, 1999a), urbanization rate alone explains as much as 95% of the variations in cropland availability, based on observations in the post-1978 period (§8.3). As the urban areas in China are likely to expand – at a decreasing rate – by 2050, the shrinkage of cropland is unlikely to be avoided. But the amount of annual loss is bounded to decrease in the long run.

The cropland area is projected to decrease from 130 million hectares in 2005 to 118 million hectares in 2020, at an average losing rate of 0.73 million hectares per year. The government goal of cropland protection of 120 million hectares, or in Chinese unit, 1.8 billion MU, towards 2020 is unlikely to achieve, should the urbanization process tend to proceed in the current direction and at the current pace. The average loss of cropland is projected to decrease to 0.5 and 0.3 million hectares per year during the 2020-2030 and 2030-2050 period, respectively. Consequently, the cropland area will decrease to 113 and 107 million hectares by 2030 and 2050, respectively (Figure 8.9). Accumulatively, 12% of the cropland is lost by 2030, and 17% by 2050, since 2005. Nationally, the availability of agricultural cropland by 2050 is maintained at the same level as one century ago (i.e., in 1950), but in per-capita terms, the former is only 37% of the latter.
8.5 Cropping intensity scenarios

As the area of cropland in China is inevitably and continuously shrinking in the foreseeable future, cropping intensity becomes more and more important in safeguarding the national food security. The multiple cropping index (MCI) – which is defined as the ratio between the total sown area and the area of cropland, usually expressed in percentages – increased steadily during the past 20 years at an annual rate of 0.9% (Table 8.9). In China, 1% increase of the MCI is equivalent to a 1.33 million hectare increase of cropland (FAO, 2003). Although double, and even triple, cropping is common practice in south China, the nation-wide average of the MCI value is only 120% in 2005. A recent study (NOARP, 1997) suggests that the average MCI value can be increased to 160-170% at the national scale, with a theoretical potential to 190%.

In the most-likely scenario, the MCI is to be increased by 15%, 10% and 5% over its 2005 value in the east, middle and west socio-economic belt, respectively, in 2030. By 2050, the MCI is projected to have another 15%, 10% and 5% increase over its 2030 value in the east, middle and west socio-economic belt, respectively. The average MCI, at the national scale, will be 133% in 2030 and 147% in 2050, compared to 120% in 2005. No abrupt changes are observed between the projected values and the steady trend during the past 20 years (Figure 8.10).

8.5.4 Soil degradation scenarios

As discussed in §8.4.1, the relative yield loss due to soil degradation is evaluated at the national scale as 11% under production conditions in the year 2005. It is worth to reiterate
Table 8.9: Sown area, cropland and the multi-cropping index, 1978-2005

<table>
<thead>
<tr>
<th>Year</th>
<th>(1) Sown area (million ha.) of all crops: NBSC (2005)</th>
<th>(2) Area of cropland (million ha.); refer to §8.3</th>
<th>(3) MCI (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1978</td>
<td>150.1</td>
<td>140.8</td>
<td>107</td>
</tr>
<tr>
<td>1980</td>
<td>146.4</td>
<td>142.5</td>
<td>103</td>
</tr>
<tr>
<td>1985</td>
<td>143.6</td>
<td>141.2</td>
<td>102</td>
</tr>
<tr>
<td>1989</td>
<td>146.6</td>
<td>140.2</td>
<td>105</td>
</tr>
<tr>
<td>1990</td>
<td>148.4</td>
<td>139.9</td>
<td>106</td>
</tr>
<tr>
<td>1991</td>
<td>149.6</td>
<td>139.7</td>
<td>107</td>
</tr>
<tr>
<td>1992</td>
<td>149.0</td>
<td>139.4</td>
<td>107</td>
</tr>
<tr>
<td>1993</td>
<td>147.7</td>
<td>139.1</td>
<td>106</td>
</tr>
<tr>
<td>1994</td>
<td>148.2</td>
<td>138.9</td>
<td>107</td>
</tr>
<tr>
<td>1995</td>
<td>149.9</td>
<td>138.6</td>
<td>108</td>
</tr>
<tr>
<td>1996</td>
<td>152.4</td>
<td>138.4</td>
<td>110</td>
</tr>
<tr>
<td>1997</td>
<td>154.0</td>
<td>138.1</td>
<td>111</td>
</tr>
<tr>
<td>1998</td>
<td>155.7</td>
<td>137.8</td>
<td>113</td>
</tr>
<tr>
<td>1999</td>
<td>156.4</td>
<td>137.6</td>
<td>114</td>
</tr>
<tr>
<td>2000</td>
<td>156.3</td>
<td>136.1</td>
<td>115</td>
</tr>
<tr>
<td>2001</td>
<td>155.7</td>
<td>134.7</td>
<td>116</td>
</tr>
<tr>
<td>2002</td>
<td>154.6</td>
<td>133.2</td>
<td>116</td>
</tr>
<tr>
<td>2003</td>
<td>152.4</td>
<td>131.8</td>
<td>116</td>
</tr>
<tr>
<td>2004</td>
<td>153.6</td>
<td>130.3</td>
<td>118</td>
</tr>
<tr>
<td>2005</td>
<td>155.5</td>
<td>130.0</td>
<td>120</td>
</tr>
</tbody>
</table>

(1) Sown area (million ha.) of all crops: NBSC (2005)
(2) Area of cropland (million ha.); refer to §8.3
(3) MCI (%)

Figure 8.10: Trend and scenarios of the multi-cropping index in China, 1978-2050

that this 11% is not measured as an annual rate. Instead, it is measured as an accumulated rate of yield change during a period of 10 to 15 years (van Lynden and Oldeman, 1997). Three soil degradation scenarios are designed:

1. Business-as-usual (1×SD)
   Soil degradation occurs at the current degree and extent. The same amount of yield will be lost in the next 15 years as it was during the past 15 years.
   Given \( p \) as the percentage of yield which is lost due to soil degradation in year \( t_1 \), \( Y_1 \) as the average yield in year \( t_1 \), the average yield in year \( t_2 \), \( Y_2 \), is calculated using
8.6 Food producing capacities in 2030 and 2050

the following equation:

\[ Y_2 = \left( 1 - \frac{t_2 - t_1}{15} \cdot \frac{p}{100} \right) \cdot Y_1 \]  

(8.2)

under the assumption that the same management level is maintained during the entire duration of the time interval \([t_1, t_2]\).

2. Zero degradation (0×SD)

No adverse effect of soil degradation is observed in year \(t_2\). This may be achieved by (i) adopting (new) soil conservation measures, and/or (ii) raising management levels. Other yield-affecting factors are kept unchanged during \([t_1, t_2]\). The same yield is achieved at \(t_2\):

\[ Y_2 = Y_1 \]  

(8.3)

3. Double degradation (2×SD)

The soil degradation in year \(t_2\) is observed as twice more limiting as in year \(t_1\), which can be caused by (i) soil conservation failure, and/or (ii) mismanagement. The average yield in year \(t_2\) is given by:

\[ Y_2 = \left( 1 - 2 \cdot \frac{t_2 - t_1}{15} \cdot \frac{p}{100} \right) \cdot Y_1 \]  

(8.4)

Despite that the yield effect of soil degradation is closely related to field management levels (van Lynden and Oldeman, 1997), it is practically hard and unwise to introduce management options into soil degradation scenarios. The “complete” set of 27 combined scenarios – 3 soil degradation rates by 3 management levels in each of the 3 socio-economic belts – not only includes redundancies, but also introduces difficulties in result handling. It is over-complicated to try to attach policy significance to one of the 27 scenarios. The management level settings in 2005, as used in grid-based yield simulation (§6.3), are therefore kept unchanged for the years 2030 and 2050. By doing so, the impact of soil degradation on long-term food security is thus focused to reflect the grave reality of soil degradation in China (Wang and Gong, 1998).

A summary of the scenarios of population, urbanization, cropland area, cropping intensity and soil degradation is given in Table 8.10.

8.6 Food producing capacities in 2030 and 2050

8.6.1 Soil degradation effect

Soil degradation on food productivity

Food productions in 2030 and 2050 were simulated using the scenario settings. Overall yield was obtained from the yield of individual food crops, following the same procedure as detailed in Chapter 6. The overall yield of food crops in 2030 and 2050 was compared, per scenario per province, to that of 2005. The resulting difference of yield between the target and the baseline years was plotted in Figure 8.11 as the effect of soil degradation on yield of food crops.

Results show that on average food crops may experience a 9.7% productivity loss by 2030 if the soil is degraded at the current rate (“business as usual”); the productivity
loss will be increased to an unbearable level of 36.7% by 2050, should the soil be twice more degraded than it is now ("double degradation"). The yield loss will be accounted for 19.2% either under the double-degradation scenario by 2030 or under the business-as-usual scenario by 2050. Results also show that the yield effect of soil degradation is highly management level dependent. The adverse effect of soil degradation on food productivity is almost completely suppressed in east China, where the input level is high. However, the effect of soil degradation in west China, where the input level is low, is so detrimental that about half – to be more specific, 45.5% – of the yield is lost under the business-as-usual scenario; and the soil is nearly unproductive – i.e., 93% yield loss – under the double-degradation scenario (Figure 8.11b).

Spatial patterns of cropland productivity classes

Much attention has been paid to the spatial patterns of the high (H), intermediate (I) and low (L) cropland productivity classes in food production at the national scale in China. Monitoring the change of the spatial patterns of them is helpful in diverse fields ranging from production organization, land conservation to foreign trade of food products; or in a broader context, from sustainable land use planning to more balanced rural development.

The aggregate productivity of food crops is used as the criterium to classify the croplands (Shi et al., 1985). The H-class is defined as yield $\geq 6,536$ kg ha$^{-1}$; the I-class as $3,765$ kg ha$^{-1} \leq$ yield $< 6,536$ kg ha$^{-1}$, and the L-class as yield $\leq 3,765$ kg ha$^{-1}$. The resulting maps are shown in Figures E.24 and E.25 (Appendix E). Change in spatial patterns is observed for almost all of the agricultural important regions. Under the business-as-usual scenario, the area of the L-class cropland in the northeast, marked by a greenish color in Figure E.24, is shrinking significantly across 2005 and 2030. Shrinking is also observed between 2030 and 2050, although the extent is smaller. In the North China Plain, the country’s most important region in food production, the H-class cropland – marked as a blueish color in map – is not only smaller in size, but also more fragmented in the southern part of the region, in particular, across the entire 2005-2030-2050 period. The Sichuan Basin, the biggest food producing base in the southwest, is classified as low-yielding (L) in both 2030 and 2050, compared to intermediate-yielding (I) in 2005. On the other hand,
Figure 8.11: Effect of soil degradation on overall yield of food crops, expressed as relative yield loss against the 2005 baseline, by 2030 (a) and 2050 (b). Scenarios: $0 \times SD =$ no degradation, $1 \times SD =$ business-as-usual (BAU), and $2 \times SD =$ double degradation.
the L-class cropland – represented by a pinkish color on map – in the middle-and-lower reaches of the Yangtze River expands significantly across 2005-2030-2050.

The change of cropland productivity classes under the double-degradation scenario is observed to occur in similar patterns but more significantly. The middle-and-lower reaches of the Yangtze River, for instance, are dominated by the L-class croplands in 2050 (Figure E.24), compared to the I-class dominance during 2005-2030.

In summary, the area of the high- and intermediate-yielding croplands is decreasing across the 2005-2030-2050 period under both scenarios. Consequently, the low-yielding cropland is taking more and more surface areas during the same period and under the same scenarios (Figure 8.12).

![Figure 8.12: Change of the area composition of the high-, intermediate-, and low-yielding croplands by 2030 and 2050 under the business-as-usual (a) and the double-degradation (b) scenarios](image)

### 8.6.2 Food producing capacities

China has sown 104 million hectares of food crops and produced 484 million tons of food in 2005 (NBSC, 2005). Under the business-as-usual scenario, the average aggregate yield of food crops, for instance, will be 4.3 t ha$^{-1}$ by 2030 and 3.9 t ha$^{-1}$ by 2050. According to the “most-likely” scenarios (§8.5), the cropland area will decrease from 130 million hectares in 2005 to 113 and 107 million hectares in 2030 and 2050, respectively. The multi-cropping index is supposed to increase from 120% in 2005 to 133% and 147% in 2030 and 2050, respectively. The total sown area of food crops is therefore projected to be 101 and 105 million hectares in 2030 and 2050, respectively. The food producing capacities in 2030 and 2050 are obtained with regard to the soil degradation scenarios and given in Table 8.11.

China’s food producing capacity tends to decline in the long run if the general trend of soil degradation will not be reverted. China will be able to produce 428.6 million tons (i.e., a 11.5% decrease from the 2005 level or -11.5%, same below) of food in 2030 and 409.1 million tons (-15.5%) of food in 2050 under the business-as-usual scenario. Under the double-degradation scenario, China is projected to produce 390.9 million tons (-19.2%) of food in 2030 and 326 million tons (-32.6%) of food in 2050, which are levels that China had surpassed in the late 1980s and the late 1970s, respectively (Figure 8.13). However, the
### 8.6 Food producing capacities

#### Table 8.11: China’s food producing capacities in the years 2030 and 2050

<table>
<thead>
<tr>
<th>Year</th>
<th>Scenario†</th>
<th>Yield (kg ha⁻¹)</th>
<th>Cropland (million ha)</th>
<th>MCI‡</th>
<th>Sown area (million ha)</th>
<th>Capacity (million t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>Observed</td>
<td>4.642</td>
<td>130</td>
<td>120</td>
<td>104.3</td>
<td>484.1</td>
</tr>
<tr>
<td>2030</td>
<td>0×SD</td>
<td>4.642</td>
<td>113</td>
<td>133</td>
<td>100.7</td>
<td>467.4</td>
</tr>
<tr>
<td></td>
<td>1×SD</td>
<td>4.256</td>
<td>113</td>
<td>133</td>
<td>100.7</td>
<td>428.6</td>
</tr>
<tr>
<td></td>
<td>2×SD</td>
<td>3.882</td>
<td>113</td>
<td>133</td>
<td>100.7</td>
<td>390.9</td>
</tr>
<tr>
<td>2050</td>
<td>0×SD</td>
<td>4.642</td>
<td>107</td>
<td>147</td>
<td>105.4</td>
<td>489.2</td>
</tr>
<tr>
<td></td>
<td>1×SD</td>
<td>3.882</td>
<td>107</td>
<td>147</td>
<td>105.4</td>
<td>409.1</td>
</tr>
<tr>
<td></td>
<td>2×SD</td>
<td>3.094</td>
<td>107</td>
<td>147</td>
<td>105.4</td>
<td>326.1</td>
</tr>
</tbody>
</table>

† 0×SD: no-degradation; 1×SD: business-as-usual; 2×SD: double-degradation
‡ multi-cropping index (§8.5.3)

Population in 2050, for instance, will be 37% more than that in the late 1970s, suggesting a 37% decrease in per capita food availability during the time interval between the two. Under the no-degradation scenario, the food producing capacities in 2030 and 2050 will be 3.4% lower and 1.1% higher than the 2005 level, respectively, showing that the negative effect of the decreasing cropland area is nearly neutralized by the positive effect of the increasing cropping intensity on food production in the long run. This in turn suggests that the above-mentioned declining trend in food production can actually be recognized as the net effect of soil degradation on food production.

Figure 8.13: China’s food producing capacities, 1949–2050. Scenarios: 0×SD = no degradation, 1×SD = business-as-usual (BAU), and 2×SD = double degradation
8.7 Long-term food security in China

The concept of food security is by no means new. Malthus (1798) created wide awareness of the problem of food security in his *Essay on Population*. He proclaimed that “the power of population is infinitely greater than the power in the earth to produce subsistence for man.” Since then and up to date, food security has been a topic of considerable attention (Higgins et al., 1983; Brown and Kane, 1995; Cohen, 1995; Pimentel et al., 1997; Rosegrant and Cline, 2003; Gilland, 2006; Lobell et al., 2008). Under the anthropogenic (Chen, 2007; Lal, 2007) and the changing climatic (Parry et al., 1999; Lal et al., 2005; WRI, 2005; Brown and Funk, 2008) influences, the declination of the quality of the agricultural land resources (Beinroth et al., 2001; Wiebe, 2003) and, in particular, soil degradation (Scherr, 1999, 2003; Koning et al., 2001; Lal, 2003) have largely threatened the potential food security at the global scale.

China has long been the most populous country on earth. The enthusiasm in monitoring its food security has never faded (Brown, 1995; Prosterman et al., 1996; Anderson and Peng, 1998; Yang and Li, 2000; Ye and Van Ranst, 2002; Yang, 2004; Xu et al., 2006; Chen, 2007; Xiong et al., 2007). However, a clear understanding about the magnitude of the effect of soil degradation on food security at the national scale is still missing from literature and thus needed.

The effect of soil degradation on long-term food security in China over the 2005-2030/2050 period is assessed, from a supply-demand equilibrium point of view, in this section.

8.7.1 Per capita supply

The fluctuations of the observed per capita food production and the future trends of it, based on production scenarios, is given in Figure 8.14. Although the total food productions in 2030 and 2050 can be roughly maintained at the same level as that in 2005 under the no-degradation scenario (Figure 8.13), the per capita food supply decreases under the same scenario in 2030 and 2050, suggesting a negative effect of population size on food security. The combined effect of population growth and soil degradation, shown as dashed lines and marked as “1×SD” for the business-as-usual and “2×SD” for the double-degradation scenarios, is overwhelmingly significant. In 2030, the per capita food supply is decreased to just above and below the lowest observed level since 1949 – which was 288 kg in 1970 – under the 1×SD and 2×SD scenarios, respectively. The per capita supply continues to decrease till 285 kg and 227 kg under the 1×SD and 2×SD scenarios, respectively, in 2050, although the population has started declining slightly since 2033 according to the “most-likely” population scenario (Figure 8.1).

The per capita food supply in 2030 and 2050 is projected to be lower than the average observed value of 360 kg during 1949-2005. The gap is even wider if they are compared to the “well-known” threshold of 400 kg per capita, which is considered sufficient to sustain the economic growth (Xiong et al., 2007). But neither the exact value of this threshold nor the width of the gap is a measurement of food security. Food security can only be properly measured with an equilibrium approach. As a matter of fact, there were only a few years – all appeared in the mid-1990s – in which the per capita food supply was greater than 400 kg. The per capita supply in the rest of the years were, unsurprisingly, below 400 kg.
8.7 Long-term food security in China

8.7.2 Per capita demand

Although the calculation of per capita food supply was straightforward – by simply dividing the total supply by the total population – the per capita food demand is much more complicated to obtain. Based on the

(1) characterization of the distinctive food consuming patterns of the urban and rural residents (Fan et al., 1994; Gao, 2004; Yen et al., 2004; Wan, 2005);

(2) analysis of the recent trend in per capita food consumption during the post-1978 period (Shi and Chang, 2004); and

(3) identification of the fraction of the total consumption which is used as direct diet (Mei, 2003),

the per capita food demands in 2030 and 2050 are estimated as 356 kg and 368 kg, respectively, as illustrated in Table 8.12. The detailed reasoning in economic terms (Liu et al., 2005b; Jiang and Davis, 2007) will, however, not be discussed any further.

The fluctuations and future trends of the estimated per capita food demand in China during 1949-2050 are shown in Figure 8.15, in comparison with the averages of the developing countries and the world based on a recent FAO (2006) study. It is not a surprise to observe that China falls into the group of developing countries. The average per capita food consumption of developing countries has been 20% lower than the world average in recent years (FAO, 2006). No abrupt change to the general situation of per capita food demand of the world is likely to occur in 2030 and 2050. China’s per capita food demand, for instance, will be only slightly higher than the average of the developing countries but still 12% lower than the world average, either in 2030 or in 2050.
Table 8.12: *Estimation of the per capita food consumption, 2003-2050*

<table>
<thead>
<tr>
<th>Year</th>
<th>2003</th>
<th>2005</th>
<th>2010</th>
<th>2020</th>
<th>2030</th>
<th>2050</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Population composition (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>urban</td>
<td>40.5</td>
<td>41.8</td>
<td>50.3</td>
<td>64.6</td>
<td>72.6</td>
<td>82.6</td>
</tr>
<tr>
<td>rural</td>
<td>59.5</td>
<td>58.2</td>
<td>49.8</td>
<td>35.4</td>
<td>27.4</td>
<td>17.4</td>
</tr>
<tr>
<td>sum</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

(2) Grain as direct diet (kg)
- urban: 139.1, 137.1, 132.0, 122.2, 113.2, 96.4
- rural: 190.5, 187.7, 180.7, 167.3, 155.0, 132.0
- average\(^\dagger\): 169.7, 167.2, 161.0, 149.1, 138.0, 117.6

(3) Fraction of food as direct diet (%)
- percentage: 52.3, 50.7, 46.5, 40.0, 34.5, 27.5

(4) Total consumption (kg): (4) = (2) × (3)/100
- urban: 266.0, 270.4, 283.9, 305.6, 328.1, 350.7
- rural: 364.2, 370.1, 388.6, 418.4, 430.0, 450.0
- average\(^\dagger\): 324.4, 328.5, 336.0, 345.6, 356.1, 368.0

\(^\dagger\) Weighted average using (1) as the weighting factor

Figure 8.15: *Estimated and projected per capita food demand in China in comparison with the averages of the developing countries and of the world, 1949-2050*
8.7 Long-term food security in China

8.7.3 Food security index

The relative food surplus is hereby defined as the food security index, or FSI, in an attempt to quantitatively measure food security. The FSI is given, in per capita terms, by:

$$FSI = \frac{s - d}{d} \cdot 100$$ (8.5)

where $s$ is the per capita food supply, and $d$ is the per capita food demand.

The food security index of China, based on the observed and estimated data for the pre-2005 and on scenarios for the post-2005 periods, is shown in Figure 8.16. It shows that historical variations in food security are well captured by the FSI values. The food security situation was significantly improved soon after the long-lasting wars – Japanese invasion since 1937; WWII; and the civil war – ended in the late 1940s. At the end of the first 5-year plan (mid-1950s), the FSI increased from -5.4 in 1949 to 26.1 in 1957, meaning that the demand-supply equilibrium turned from a 5.4% per capita deficit to a 26.1% per capita surplus. The peak FSI value of 31.5 appeared in 1984, coinciding with the record harvest of 3.9 billion tons of food in the same year. Although higher productions (∼5 billion tons) were achieved consecutively during 1996-1999, FSI values in the same period were not greater than that of 1984, reflecting a raised living standard. Extreme climatic events and natural hazards, which caused notable production loss during 2000-2003, are responsible for the second largest drop of the FSI values, after the Great Leap Forward period, which will be discussed separately in a case study below.

![Figure 8.16: Food security index, 1949-2050. Three soil degradation scenarios are used to project the post-2005 FSI values: 0×SD = no degradation, 1×SD = business-as-usual (BAU), and 2×SD = double degradation. Extra points based on literature data are included to show the highest possible FSI’s under optimal conditions](image)

China faces great challenges in safeguarding its food security in the long run. The FSI value is seen to drop from 12.7 in 2005 to -9.8 and -7.5 in 2030 and 2050, respectively,
under the no-degradation scenario. This, together with the observation that the positive effect of a higher cropping intensity cancels the negative effect of a shrinking cropland area (§8.6.2), suggest that the present-day producing capacity (2005 level) will not be able to sustain the long-term needs, which are associated with a higher per capita demand, despite no further soil degradation is assumed. Under the business-as-usual scenario, 17.3% and 22.6% of per capita food demand will not be met in 2030 and 2050, respectively. Under the double-degradation scenario, food shortage in 2030 and 2050 will be as high as 24.5% and 38.3%, respectively.

FSIs converted from third-party projections of China’s medium- to long-term food producing capacities may provide extra insight. Chen and Chen (2000) forecast that, under optimal conditions, China may have the potential to produce 597 million tons of food in 2010. In an earlier study, CAS (1992) projects that China’s maximum long-term food producing capacity can be as high as 830 million tons. The former means a FSI value of 29 in 2010, and the latter, a FSI of 48 in 2050 (Figure 8.16). The wide gap between the literature line and the scenario lines suggests that there is high potential to improve the long-term food security in China, despite the above-mentioned challenges.

Case study: Food security during the Great Leap Forward period

Official statistics either leave the 1958-1960 period out, or are incomplete or unreliable due to inflated figures reported by local government officials. Cross checks with other sources of data have to be undergone with care. Aston et al. (1984) reported a sharp decline of per capita grain production by 30% in 1960 and 28% in 1961, as compared to the reference period of 1956/1957. Smil (1999b) estimated that the per capita food production during the same period was 120-150 kg, which corresponds to a 40-50% drop from 1957. The following discussion is based on the former estimation, since a more conservative estimation is more likely to happen.

Consequently the FSI dropped 45 units in one year (from 26.1 in 1957 to -18.8 in 1958). Abnormal weather conditions may adversely affect food production but cannot be blamed for the whole magnitude of the production decline. Rather, the agricultural policy – in service to the GLF – was the primary cause. The FSI in Figure 8.16 gradually increased to -5.9 in 1962. The estimation was reasonable since it fitted the rest of the years based on statistical data.

Reasons for the sharp decline in FSI include:

(1) Agricultural policy: much of the agricultural labor was diverted to non-agricultural activities, resulting in severe nation-wide shortage of labor during the harvest season in 1958;

(2) Natural catastrophe: serious droughts in the north and floods in the south during the period widened the supply-demand gap and exacerbated the suffering in a great deal. However, Smil (1999b) argued that abnormal weather conditions should not be blamed since the droughts and floods in the 1990s were believed to be the worst of its kind in China’s modern history and had only a marginal effect on the country’s adequate food supply; and

(3) Bureaucracy: biased decisions on the nation-wide food distribution, production and trade, which were largely based on fabricated statistics, and lack of reliable means of monitoring the food production in general helped the regional food shortage develop into a national one.
9.1 General conclusions

The methodologies applied in this large-scale quantitative assessment of the long-term food producing capacities in China are proved to be successful. Statistical validation procedures (§6.5) suggested a close match between simulated and observed yields of food crops. Regression and variance analysis of crop yields against soil characteristics revealed that limiting soil characteristics should be collectively evaluated in order to derive a single soil index (§7.2). Progressive statistical analysis (Chapter 7) indicated that the soil and the management indices should be used together in yield prediction. The performance is much lower if they are used separately. For the performance sake, it is generally recommendable that the quantitative land evaluation methodologies (Chapter 4) should be used in its entirety in productivity assessment applications.

As the Chinese society evolves toward a more urbanized one, the agricultural cropland in China will continue to be lost at an average rate of 0.5 million hectares per year. Until 2050, the total area of cropland will decrease to 107 million hectares. To keep up with the consumptive needs, the cropland will have to be utilized more intensively in food production. The negative effect of diminishing cropland area on food producing capacity can be compensated by the positive effect of the increasing cropping intensity by 2050, leaving soil degradation as the most significant single factor that adversely affects China’s food producing capacity in the long run.

Under the business-as-usual scenario, China will be able to produce 428.6 million tons of food in 2030, which is 11.5% lower than the 2005 level. In 2050, the food producing capacity will further decrease to 409 million tons, or a 15.5% drop from the 2005 level. Under the double-degradation scenario, China will only be able to produce 391 and 326 million tons of food in 2030 and 2050, respectively. This suggests that present-day producing capacity (2005 level) will not be able to sustain the long-term needs under the current management level even if soil degradation is not becoming more limiting. Food shortage will account for 17.3-22.6% and 24.5-38.3% of the total demand under the business-as-usual and the double-degradation scenario, respectively, during the 2030-2050 period.

The results reveal the very fact that China is facing great challenges in safeguarding its food security by 2030/2050. The detrimental effect of soil degradation on food security is so evident that technical measures and policy levers must be activated today in order
to avoid, or at least mitigate, the risks of tomorrow.

9.2 Methodological innovations

The following methodological approaches, among others, which have contributed to the aggregate performance of yield prediction of food crops in China, are observed to be innovative:

(1) **The grid-based data integration, production simulation and result presentation at the national scale of China:** (a) The heterogeneous data sets of climate, soil, crop and management have been properly handled in form of 5 km × 5 km grids. The resulting maps suggest that the grid resolution of 5 km is an appropriate compromise between the performance and the cost. The grid-based approach of data integration not only smooths the simulation process but also facilitates the result presentation; (b) The point climatic data sets have been interpolated to generate continuous surfaces covering the entire study area using geostatistics. Prediction error analysis reveals that the quality of large climatic data sets is largely dependent on reliability, consistency and configuration of the source points once a threshold number of input points is available; (c) Data sets collected at greater scales (provincial, zonal, etc.) should be downscaled to the predefined 5 km × 5 km scale at simulation time. At validation time, simulation results should be upscaled back to the provincial scale;

(2) **Input level characterization:** Instead of dealing with individual parameters of factor inputs, an overall score was computed and attributed to each of the economic development belts, and subsequently defined as the high, intermediate and low input levels. The characterization of input levels creates a sound yet simple basis for the evaluation of the effect of management practices on crop yields. Furthermore, it makes it possible to simulate the response of crop yield to soil degradation under varying management conditions, which in turn helps identify the effect of agricultural investment on national food security, especially under the scenario of a decreasing soil quality;

(3) **Management index evaluation using production functions:** The significance of the use of the one-factor variant of the Cobb-Douglas production function (Cobb and Douglas, 1928; Douglas, 1976; Pokrovski, 2003) comes from the introduction of a deterministic mechanism into the evaluation process. Statistical analysis procedures (§7.4) showed a high correlation between the calculated management index and the observed crop yield, suggesting a performance gain over the otherwise empirical approaches;

(4) **Food security index:** Food security is a dynamic concept from the demand-supply point of view. Different from other approaches (Beinroth et al., 2001), the proposed FSI, as demonstrated in §8.7, attached not only a magnitude but also a sign to the food security concept. The sign of the index shows immediately the status (surplus/shortage) of the food security while the magnitude of the index shows the degree (in percentage) of it. A FSI of -7.5, for instance, depicts a clear picture of food insecurity – food shortage of 7.5% – under the no-degradation scenario in 2050.
9.2.1 The WLES

The WLES is observed to be the first Web-based system of its kind. Compared to its stand-alone predecessors, the WLES has a loosely coupled multi-tier structure which seamlessly integrates the domain knowledge of land evaluation and the natural resource databases. The user-friendly front-end runs inside the client’s browsers, whereas the back-end database server defines reference crops and stores site-specific data collected. The system core possesses a hierarchical structure with the biomass calculator (BMC), the soil-water balance simulator (WBS) and the land limitations evaluator (LLE) as its components. Ease-of-use features of the WLES range from database-driven parameter setting to graphical result visualization and simulated data downloading, and from topic-sensitive help to evaluation history tracking and management, and so forth (Ye et al., 2004, 2007, 2008; Ye and Van Ranst, 2004).

9.2.2 Applicability

As a Web-based system, there is no technical barrier that restricts the application of the WLES. In most of the cases, the WLES provides real-time assistance in parameter settings, which is viewed as an innovative feature of it. However, errors can be introduced by simply accepting the default values suggested by the system. These defaults can be either mean values or values from a different agro-climatic zone. Physiological and phenological parameters, among others, of localized crop cultivars have to be collected prior to model runs. The same holds true for soil parameters.

9.3 Uncertainties

Uncertainties arise from the following sources:

1. **Prediction error** arisen from kriging interpolation of the climatic parameters (§2.6.3) has propagated and added to the prediction error of simulated yields. The following measures are recommendable to lower the kriging error in future studies: (a) increasing the number of climatic stations for the parameters of, in particular, daily sunshine duration and wind velocity (see Figure 2.7); and (b) improving the reliability, consistency and the spatial configuration of the source points;

2. The **mismatch** between the average climate (see Figure 2.6 for record length) and the transient observed yields (in 2005) may introduce extra errors into yield prediction. The magnitude of yield response to climatic fluctuations can be simulated using climatic and yield records covering the same time span, as recommended as one of the future research topics;

3. The characterization of the zone-specific **cropping systems** may differ, to some extent, from the reality of the Chinese rural economy which is dominated by the fragmented household farming. Improvements can be made by adopting, for instance, satellite imagery in cropping system identification on a yearly basis;

4. The use of **average values** at the provincial level, as discussed earlier, may have largely contributed to the systematic under- and overestimations of crop yields. The ‘unmeasured’ simulation error at grid scale propagated up to the provincial level at validation time and contributed to the overall estimation error;
The representability of **production scenarios** can moderately to strongly influence the accuracy of the long-term yield forecasts. However, scenario uncertainty can not be actually measured;

Difficulties in judging the **consuming patterns** of the rural and the urban residents based, in particular, on trend analysis of per capita grain consumption and the fraction of grain as direct food over the total grain consumption also introduce uncertainties into the FSI projections. Unfortunately, improvements in this field have to be made by economists;

A misjudgement of the future trend of the **urbanization process** affects the accuracy of the estimated per capita food demand, which in turn influences the credibility of the FSI. The urbanization process in China is, as claimed by a batch of economists (Ebanks and Cheng, 1990; Liu et al., 2003; Change and Brada, 2006), lagged behind countries at a comparable level of development. This assertion suggests that a higher urbanization rate would likely occur in the coming years, which has apparently an adverse effect on food security;

The assumed effect of **soil degradation on food productivity** was only partially validated at the national scale. Large disparities were observed at the regional scale, for example, in the erosion-prone northwest (Yang et al., 2005; Chen et al., 2007). As a matter of fact, no consensus has been reached and uncertainty remains up to date concerning the productivity effect of soil degradation even at the global scale (Lal, 1998). Yield reduction in Africa due to past soil erosion may range from 2 to 40%, with a mean loss of 8.2% for the continent. In South Asia, annual loss in productivity is estimated at 36 million tons of cereal by water and wind erosion (Eswaran et al., 2001). One study in China (Scherr, 1999) shows that despite significant differences in cumulative soil loss and water runoff, there were no differences in maize yield. Similar inferences can be drawn with regard to the impact of cumulative soil erosion on yield of rice in Thailand. Whereas soil loss ranged from 330 to 1,478 t ha$^{-1}$, the corresponding yield of rice ranged from 4.0 to 5.3 t ha$^{-1}$. The lowest yield was obtained from treatments causing the least soil loss (Scherr, 1999). Yield of sisal in Tanzania was reported correlated with pH, CEC, and Al saturation but not with SOC and N contents (Hartemink, 1995). Similar effects have also been commonly observed in soil parameter-crop yield relationship analysis in §7.2.1.

### 9.4 Policy options toward food security

The ability of agriculture to support a growing population has been a concern for generations and continues to be high on the global policy agenda (Rosegrant and Cline, 2003; Wiebe, 2003; Stocking, 2006). The eradication of poverty and hunger was included as one of the United Nations Millennium Development Goals adopted in 2000. One of the targets of the goals is to halve the proportion of people who suffer from hunger between 1990 and 2015 (World Bank, 2003). Meeting this food security goal will be a major challenge under the conditions of declining soil quality (Anderson and Peng, 1998; Lal, 2003; Swaminathan, 2007) and changing climate (Farquhar et al., 1980; Melillo et al., 1993; Braswell et al., 1997; Schimel et al., 2000; Brown and Funk, 2008).

Predicting food security outcomes has been part of the policy landscape since Malthus (1798). One of the major predictions this dissertation made is that soil degradation will
be the single most influencing factor for the long-term food security in China. Between now and 2050, soil degradation alone will cause the FSI to decrease at an average rate of 0.6 unit yr$^{-1}$ under the business-as-usual and 1.0 unit yr$^{-1}$ under the double-degradation scenarios. The accordance between this prediction and the long-standing concerns about the ability of agricultural production to keep up with global food demand (Meadows et al., 1979; Brown, 1995; Brown and Kane, 1995; Pimentel et al., 1995; Lal, 2003) suggests that strong policy interventions must be immediately implemented in order to guide the technical countermeasures in safeguarding the national, or global, food security in the long run.

Achieving food security needs institutional and investment reforms (Fan, 1997) on multiple fronts, including human resources, agricultural research, rural infrastructure, water resources, and farm- and community-level agricultural and natural resources management. Progressive policy actions must not only increase agricultural production, but also boost incomes and reduce poverty in rural areas where most of the poor live. Focus of development programmes should be reoriented to or strengthened on the integrated and efficient utilization of agricultural resources based on local ecological conditions (Robertson and Swinton, 2005; Yang, 2006). Furthermore, early warning systems are needed in fighting food security especially in food-insecure regions (Lobell et al., 2008).

Making substantial progress in improving food security is indeed a challenging task. However, innovations in agro-ecological approaches and crop breeding have brought some documented successes (Delmer, 2005; Huang et al., 2005). The rice hybrids, which is planted in more than half of the total rice-growing area in China (Wang et al., 2005; IRRI, 2007), for instance, have achieved a stable yield advantage of at least 15% over the best inbred cultivars (Yuan, 1994, 2001; Jauhar, 2006; Cheng et al., 2007), which had greatly contributed to food security in contemporary China. Agricultural investments in research and in field (Dyson, 1999) are seen as essential yet efficient means not only in driving technical and institutional changes but also in mitigating soil degradation’s impact on food security. If the management level in 2030 is raised from high-intermediate-low to high-high-intermediate for the east-middle-west economic development belts, and in 2050 to high-high-high, the FSI will increase from -17.3 to -10.9 in 2030 and from -22.6 to -2.1 in 2050 under the business-as-usual scenario, and from -24.5 to -18.7 in 2030 and from -38.3 to -10.7 in 2050 under the double-degradation scenario, respectively (Figure 9.1).

A comparison of this figure to Figure 8.16, together with the wide gap between the ascending literature line and the descending scenario lines in either figure, suggest that a high potential of improvement exists for the long-term food security in China, if intensive production is maintained under high investment levels with managed environmental damages, as being practised in the U.S., the E.U., and Japan (Tilman et al., 2002). In a broader sense, major breakthroughs must occur in basic plant physiology, agroecology, and soil science to achieve the long-term food security and environmental integrity (Cassman, 1999).

### 9.4.1 Soil management strategies

Soil degradation poses enormous threat to the long-term food security in China and elsewhere, as we have seen so far. Farmers will be the first to receive the impact of soil degradation since they depend on the soil for their livelihood. Therefore good soil resource management strategies should assist farmers in responding to soil degradation threats by following economic incentives and adopting appropriate technologies (Templeton and...
Figure 9.1: Food security index, 1949-2050. Management levels in 2030 and 2050 are raised from high-intermediate-low (for east-middle-west belts) in 2005 to high-high-intermediate and high-high-high, respectively. Three soil degradation scenarios are used to project the post-2005 FSI values: $0 \times SD = \text{no degradation}$, $1 \times SD = \text{business-as-usual (BAU)}$, and $2 \times SD = \text{double degradation}$. Extra points based on literature data are included to show the highest possible FSI’s under optimal conditions.

Scherr, 1997; Scherr, 1999). Pattern 1 in Figure 9.2 illustrates such a process of incentive-response interactions, in which increasing pressure on soil resource over time initially leads to soil degradation. But farmers are economically “stimulated” to respond by eventually improving soil management practices and making investment to restore, maintain, or even ultimately improve the soil’s potential productive capacity.

More efficient incentives are desirable to accelerate farmer’s response in situations (a) when farmers delay to take action (pattern 3 in Figure 9.2), and (b) where social benefits are greater than farmer’s private benefits (pattern 4). In the latter case, farmers are encouraged to respond sooner so that degradation effects on soil’s productive capacity are largely avoided. Strong policy interventions are needed to slow or even reverse soil degradation when farmers fail to take action (pattern 2). It usually means that appropriate technologies are not in farmer’s possession, or productive factors such as labor, capital or other inputs are not in their disposal, or economic returns are believed to be marginal.

Knowledge about the following soil conservation options should be readily available to farmer’s enquiry: ridge-planting, no-till cultivation, crop rotation, strip cropping, grass strips and mulches, living mulches, agroforestry, terracing, contour planting, cover crops, windbreaks, and so forth (Pimentel et al., 1995). Although the specific processes vary, all conservation methods reduce erosion rates by maintaining a protective vegetative cover over the soil, which is often accompanied by the reduction in the frequency of ploughing. Reduced tillage, for instance, and no-till agriculture as a special case, produce erosion
9.5 Recommendations for future researches

Recommendations for future researches are given as the following:

(1) Application of error propagation and estimation techniques should be prioritized in future researches, so that overall estimation error of crop yield can be traced back to its sources with a ‘measured’ magnitude. Using existing computer tools (Brown and Heuvelink, 2007) in error propagation and spatial uncertainty investigation is preferably a feasible solution;

(2) Improvement of the integration between the WLES and a grid-based GIS. The functionalities of reading input data from and exporting evaluation results into the grid format need to be strengthened and integrated into the I/O module of the WLES, which will greatly improve the performance of the BM mode of the WLES, and simplify data I/O using the GRID format;

(3) Enhancement of the production function-based management index assessment which was initialized in Chapter 4. Although the approach was proved to be (a) valid in
representing the levels of factor inputs using overall scores based on characterization of input factors, and (b) applicable in modeling the effects of management practices on yields of food crops at the national scale in China, the applicability and validity of the approach still need to be tested either extensively with datasets from a broader geographical coverage, or intensively with datasets densely sampled at the catchment to ecosystem scales;

(4) Addition of new functionality of climate change impact assessment (CCIA), in which biogeochemical processes such as “CO₂ fertilization” (Farquhar et al., 1980; Melillo et al., 1993; Braswell et al., 1997; Schimel et al., 2000), nitrogen deposition (Vitousek et al., 1997; Schimel et al., 2000; Nemani et al., 2003) and temperature dependence of the autotrophic respiration rate (Kirschbaum, 1995, 2006; Cerovic et al., 1996; Rustad and Fernandez, 1998; Rustad et al., 2000), etc. need to be properly and quantitatively addressed;

(5) Applications of the WLES in, for instance, the assessment of climate change effects on long-term food security in China. Many studies on this topic have been conducted at the global scale (Parry et al., 1999; Rosegrant and Cline, 2003; Lal et al., 2005; Brown and Funk, 2008; Lobell et al., 2008), but few has been reported in China. Being the world’s most populous country, the fluctuation of China’s long-term food security will affect the livelihood and welfare of more than one-fifth of the entire population on earth.
Appendix A

DEM-GRID Conversion

A.1 About USGS GTOPO30 DEM

A.1.1 Geographic coverage

USGS GTOPO30 (USGS, 1996) is a global DEM dataset covering the full extent of latitude from 90°S to 90°N, and the full extent of longitude from 180°W to 180°E. The horizontal grid spacing is 30 arc-seconds (0.008333333333333 degrees), resulting in a DEM having dimensions of 21,600 rows and 43,200 columns. The horizontal coordinate system is decimal degrees of latitude and longitude referenced to WGS84. The vertical units represent elevation in meters above mean sea level. The elevation values range from -407 to 8,752 meters. In the DEM, ocean areas have been masked as “no data” and have been assigned a value of -9999. Lowland coastal areas have an elevation of at least 1 meter, so in the event that a user reassigns the ocean value from -9999 to 0 the land boundary portrayal will be maintained. Due to the nature of the raster structure of the DEM, small islands in the ocean less than approximately 1 square kilometer will not be represented.

A.1.2 Grid spacing and resolution

For any application, the horizontal grid spacing (which limits the resolution) and the vertical accuracy of GTOPO30 must be considered. The 30 arc-second grid spacing equates to about 1 km, although that number decreases in the east/west (longitudinal) direction as latitude increases. Table A.1 lists the approximate distance covered by 30 arc-seconds at different latitudes. Thus, at high latitudes there is an unavoidable redundancy of data in order to keep the 30 arc-second spacing consistent for the global dataset. This is particularly true for the geographic version of Antarctica where the ground distance for 30 arc-seconds of longitude converges to zero at the South Pole.

A.1.3 Need for projection

The variation in ground dimensions for one 30 arc-second cell should be especially considered for any application that measures area of or distance across a group of cells. Derivative products, such as slope maps, drainage basin areas, and stream channel length, will be more reliable if they are calculated from a DEM that has been first projected from geographic coordinates to an equal area projection, so that each cell, regardless of latitude, represents the same ground dimensions and area as every other cell.
A.2 About Arc/Info GRID

Table A.1: *Ground distance covered by 30 arc-seconds at different latitudes*

<table>
<thead>
<tr>
<th>Latitude (*)</th>
<th>Ground distance (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E/W</td>
</tr>
<tr>
<td>Equator</td>
<td>928</td>
</tr>
<tr>
<td>10</td>
<td>914</td>
</tr>
<tr>
<td>20</td>
<td>872</td>
</tr>
<tr>
<td>30</td>
<td>804</td>
</tr>
<tr>
<td>40</td>
<td>712</td>
</tr>
<tr>
<td>50</td>
<td>598</td>
</tr>
<tr>
<td>60</td>
<td>465</td>
</tr>
<tr>
<td>70</td>
<td>318</td>
</tr>
<tr>
<td>73</td>
<td>272</td>
</tr>
<tr>
<td>78</td>
<td>193</td>
</tr>
<tr>
<td>82</td>
<td>130</td>
</tr>
</tbody>
</table>

A.1.4 Applicability

Users should maintain the distinction between grid spacing and resolution. Even though the global data set has a consistent 30 arc-second grid spacing, not all topographic features that one would expect to be resolved at that spacing will be represented. The level of detail of the source data determines whether the 30-arc second sampling interval is truly appropriate for resolving the important topographic features represented in the source.

A.2 About Arc/Info GRID

A.2.1 Arc/Info binary grid

The Arc/Info Binary Grid format is the internal working format of the Arc/Info GRID product. It is also usable and creatable within the Spatial Analyst component of ArcView. It is a tiled (blocked) format with run length compression capable of holding raster data of up to 4 byte integers or 4 byte floating point data.

The Arc/Info Binary Grid is a proprietary format and thus undocumented by the ESRI, which means the technical details of the format are unknown. This format should not be confused with the Arc/Info ASCII Grid format which is the interchange format for grids. Files can be converted between binary and ASCII format with the GRIDASCII and ASCIIGRID commands in Arc/Info. This format is also different than the flat binary raster output of the GRIDFLOAT command. The Arc/Info binary float, and ASCII formats are also accessible from within ArcView.

File structure

A binary grid coverage is stored in a directory of the same name. For instance, a grid called farmland5k was used to represent the spatial distribution of the farmland in China at 5 km by 5 km resolution, which was resulted in from a remote-sensing-based inventory at the national scale (§ 2.5). The grid will have the following files under the directory “farmland5k”:
Table A.2: File structure and functions of Arc/Info Binary Grid

<table>
<thead>
<tr>
<th>File</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>dblbnd.adf</td>
<td>Bounds (LLX, LLY, URX, URY) of the utilized portion of the grid. LLX and LLY are the coordinates of the lower-left corner of the portion, and URX and URY, of the upper-right corner.</td>
</tr>
<tr>
<td>hdr.adf</td>
<td>Header file which contains information on the tile sizes, and number of tiles in the dataset</td>
</tr>
<tr>
<td>log</td>
<td>Arc/Info and/or GRID commands executed on this grid</td>
</tr>
<tr>
<td>prj.adf</td>
<td>Projection definition file</td>
</tr>
<tr>
<td>sta.adf</td>
<td>Raster statistics. In particular, the raster min, max, mean and standard deviation</td>
</tr>
<tr>
<td>vat.adf</td>
<td>Value attribute table</td>
</tr>
<tr>
<td>w001001.adf</td>
<td>Actual raster data</td>
</tr>
<tr>
<td>w001001x.adf</td>
<td>Index file containing pointers to each of the tiles in the w001001.adf raster file</td>
</tr>
</tbody>
</table>

Projection definition

The projection file prj.adf possesses the same file name and content as those of an Arc/Info vector coverage. The projection definition for the Lambert Azimuthal Equal Area projection, for instance, used in chapter 2 can be stored as plain ASCII texts in the file:

```
Projection LAMBERT_AZIMUTHAL
Datum NONE
Zunits NO
Units METERS
Xshift 0.0000000000
Yshift 0.0000000000
Parameters 6370997.0000000000 0.0000000000
6370997.00000 /* radius of the sphere of reference */
105 0 0.000 /* longitude of center of projection */
44 0 0.000 /* latitude of center of projection */
0.00000 /* false easting (meters) */
0.00000 /* false northing (meters) */
```
A.2.2 Arc/Info ASCII grid

The Arc/Info ASCII Grid format is a specific interchange format developed for Arc/Info GRID. An ASCII grid cannot be directly processed by Arc/Info, Arc/Info GRID or ArcView; it has to be converted to the Arc/Info Binary Grid format (A.2.1) first using the ASCIIGRID command at the Arc: or Grid: command prompt.

Although multiple binary grids can be combined to form multi-bands, for instance in the case of a false color grid using three binary grids as R-G-B channels, ASCII grid is a single-band format. Therefore a combined 3-band grid will need three ASCII grids to separately store the datasets of the component binary grids.

File structure

The format consists of a header that specifies the geographic parameters and resolution, followed by the actual grid cell values. Usually the file extension is .asc, but recent versions of ESRI software also recognize the extension .grd. The content of the file is organized in the following way:

```
ncols xxxx
nrows xxxx
xllcorner xxxx
yllcorner xxxx
cellsize xxxx
nodata_value xxxx
```

where “xxxxx” is to be replaced by real-world values as shown by the example in the end of § A.2.2.

Lines 1-6 contain the geographic header; all the rest lines contain the space-delimited data values (Table A.3).

The xllcorner and yllcorner in the header are given as the edges of the lower-left corner cell instead of the center of the cell. Arc/Info supports the use of xllcenter and yllcenter in the header to allow the coordinates of the center of the corner cell to be given. The origin of the grid is the upper left and terminates at the lower right.

Projection file

The Arc/Info ASCII Grid is incapable of telling the projection system of the grid. Therefore a separate plain text projection file, of the same file name but with the .prj extension, should be accompanying the grid file. Such a project file can be automatically created by the GRIDASCII command if the projection of the grid is known. Similarly the correct projection data will be incorporated in the generated grid by the ASCIIGRID command provided that a same name projection file exists in the same directory as the grid file.

For the content of the projection file, see § A.2.1.
Table A.3: *File structure and contents of an Arc/Info ASCII Grid*

<table>
<thead>
<tr>
<th>Line</th>
<th>Content</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ncols xxxxx</td>
<td>Number of columns in the grid and xxxxx is the numerical value</td>
</tr>
<tr>
<td>2</td>
<td>nrows xxxxx</td>
<td>Number of rows in the grid and xxxxx is the numerical value</td>
</tr>
<tr>
<td>3</td>
<td>xllcorner xxxxx</td>
<td>X-coordinate of the left edge of the cell in the lower-left corner of the grid; xxxxx is the numerical value</td>
</tr>
<tr>
<td>4</td>
<td>yllcorner xxxxx</td>
<td>Y-coordinate of the lower edge of the cell in the lower-left corner of the grid; xxxxx is the numerical value</td>
</tr>
<tr>
<td>5</td>
<td>cellsize xxxxx</td>
<td>Cell size or resolution of the grid; xxxxx is the numerical value</td>
</tr>
<tr>
<td>6</td>
<td>nodata_value xxxxx</td>
<td>Value that represents missing data and xxxxx is the numerical value; Optional; -9999 as ESRI default</td>
</tr>
<tr>
<td>7</td>
<td>EOF†</td>
<td>Data values, either integer or floating point numbers</td>
</tr>
</tbody>
</table>

† end-of-file

**Example**

To convert the `farmland5k` grid (§ A.2.1) to an ASCII grid, the following command is used

```
gridascii farmland5k farmland5k.asc
```

It will result in two files: the ASCII grid `farmland5k.asc` and its projection file `farmland5k.prj`. The contents of the latter has already been stated in § A.2.1; the contents of the former is given below.

```
ncols 963
nrows 808
xllcorner -2650000
yllcorner -2835000
cellsiz 5000
NODATA_value -9999
-9999 -9999 -9999 -9999 -9999 -9999 21 21 21 21 21 23 23 ...
```

The `farmland5k` grid is 963 cells wide and 808 cells high. The X-Y coordinates of the lower-left edge are (-2650000, -2835000) meters in Lambert Azimuthal Equal Area projection. Grid resolution is 5000 m, and -9999 means no data.

The first 7 cells in the first row of the grid have no data, or in other words, are out of the study area. The next 5 cells have an integer value of 21 which stands for cropland; and the following 2 cells have the value of 23 for “mosaic cropping”. There are 778,104 values in total to be given in the grid body before the end of the file.
A.3 USGS GTOPO30 DEM to Arc/Info grid conversion

A.3.1 Byte order

The DEM data are provided in Motorola byte order, which stores the most significant byte first (“big endian”). Systems such as Sun SPARC and Silicon Graphics workstations use the Motorola byte order. The Intel byte order, which stores the least significant byte first (“little endian”), is used on DEC Alpha systems and most PCs. Users with systems that address bytes in the Intel byte order may have to “swap bytes” of the DEM data unless their application software performs the conversion during ingest. The statistics file (.STX) provided for each tile gives the range of values in the DEM file, so users can check if they have the correct DEM values stored on their system.

A.3.2 Conversion and display

Arc/Info or ArcView can display the DEM data directly after simply renaming the file extension from .DEM to .BIL. However, if a user needs access to the actual elevation values for analysis in Arc/Info the DEM must be converted to an Arc/Info Grid with the command IMAGEGRID. IMAGEGRID does not support conversion of signed image data, therefore the negative 16-bit DEM values will not be interpreted correctly. After running IMAGEGRID, an easy fix can be accomplished using the following formula in Grid:

\[
\text{out\_grid} = \text{con}(\text{in\_grid} \geq 32768, \text{in\_grid} - 65536, \text{in\_grid})
\]

The converted grid will then have the negative values properly represented, and the statistics of the grid should match those listed in the .STX file. If desired, the -9999 ocean mask values in the grid could then be set to NODATA with the SETNULL function.

The procedure of converting DEM to GRID is detailed below. It is trivial to translate the procedure into an Arc/Info AML script.

Suppose the study area is consisted of 4 tiles, tile1.dem though tile4.dem:

1. Rename the .dem files to .bil:
   D:\Data> ren tile?.dem tile?.bil

2. Convert the .bil files to grids using the Arc imagegrid command:
   Arc: imagegrid tile1.bil topo1
   Arc: imagegrid tile2.bil topo2
   Arc: imagegrid tile3.bil topo3
   Arc: imagegrid tile4.bil topo4

3. Merge these 4 tiles into a single grid using the GRID merge function:
   Grid: temp1 = merge(topo1, topo2, topo3, topo4)

4. Use the study area boundary polygon to clip the merged grid:
   Arc: latticeclip(temp1, china, temp2)

5. Restore the negative elevations in the grid using the GRID con function:
   Grid: temp3 = con(temp2 >= 32768, temp2 - 65536, temp2)

6. Set the ocean mask value as NODATA using the GRID setnull function:
   Grid: topo = setnull(temp2 == -9999, temp2)
7. Project the grid to the Lambert Azimuthal Equal Area coordinate system (§ A.2.1) and upscale the original DEM of 30-arc seconds (approximately 1 km at the Equator) to a 5 km equal area grid:

Grid: \( \text{topo5k} = \text{project(topo, lambert.txt, 5000)} \)
Appendix B

List of Climatic Stations

B.1 WMO meta data

Table B.1: List of climatic stations: meta data

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Station</th>
<th>Latitude (°N)</th>
<th>Longitude (°E)</th>
<th>Elevation (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>319600</td>
<td>Vladivostok</td>
<td>43.12</td>
<td>131.90</td>
<td>138</td>
</tr>
<tr>
<td>02</td>
<td>366660</td>
<td>Zaysan</td>
<td>47.50</td>
<td>84.98</td>
<td>557</td>
</tr>
<tr>
<td>03</td>
<td>368700</td>
<td>Almaty</td>
<td>43.23</td>
<td>76.90</td>
<td>847</td>
</tr>
<tr>
<td>04</td>
<td>369740</td>
<td>Naryn</td>
<td>41.43</td>
<td>76.00</td>
<td>2,041</td>
</tr>
<tr>
<td>05</td>
<td>384570</td>
<td>Taskent</td>
<td>41.27</td>
<td>69.27</td>
<td>489</td>
</tr>
<tr>
<td>06</td>
<td>409480</td>
<td>Kabul</td>
<td>34.55</td>
<td>69.22</td>
<td>1,791</td>
</tr>
<tr>
<td>07</td>
<td>415710</td>
<td>Islamabad</td>
<td>33.62</td>
<td>73.10</td>
<td>508</td>
</tr>
<tr>
<td>08</td>
<td>417791</td>
<td>Karachi</td>
<td>24.83</td>
<td>67.05</td>
<td>5</td>
</tr>
<tr>
<td>09</td>
<td>419174</td>
<td>Dacca</td>
<td>23.77</td>
<td>90.38</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>420270</td>
<td>Srinagar</td>
<td>34.08</td>
<td>74.83</td>
<td>1,587</td>
</tr>
<tr>
<td>11</td>
<td>421820</td>
<td>Delhi</td>
<td>28.58</td>
<td>77.20</td>
<td>216</td>
</tr>
<tr>
<td>12</td>
<td>444540</td>
<td>Kathmandu</td>
<td>27.70</td>
<td>85.37</td>
<td>1,337</td>
</tr>
<tr>
<td>13</td>
<td>450050</td>
<td>Hong Kong</td>
<td>22.30</td>
<td>114.17</td>
<td>62</td>
</tr>
<tr>
<td>14</td>
<td>470350</td>
<td>Sinuju</td>
<td>40.10</td>
<td>124.38</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>470550</td>
<td>Wonsan</td>
<td>39.18</td>
<td>127.43</td>
<td>36</td>
</tr>
<tr>
<td>16</td>
<td>470580</td>
<td>Pyongyang</td>
<td>39.03</td>
<td>125.78</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>470690</td>
<td>Haeju</td>
<td>38.03</td>
<td>125.70</td>
<td>81</td>
</tr>
<tr>
<td>18</td>
<td>471080</td>
<td>Seoul</td>
<td>37.57</td>
<td>126.97</td>
<td>87</td>
</tr>
<tr>
<td>19</td>
<td>471530</td>
<td>Pusan</td>
<td>35.10</td>
<td>129.00</td>
<td>71</td>
</tr>
<tr>
<td>20</td>
<td>471840</td>
<td>Cheju</td>
<td>33.52</td>
<td>126.53</td>
<td>24</td>
</tr>
<tr>
<td>21</td>
<td>478270</td>
<td>Kagoshima</td>
<td>31.57</td>
<td>130.55</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>479300</td>
<td>Naha</td>
<td>26.18</td>
<td>127.65</td>
<td>8</td>
</tr>
<tr>
<td>23</td>
<td>480080</td>
<td>Myitkyina</td>
<td>25.37</td>
<td>97.40</td>
<td>147</td>
</tr>
<tr>
<td>24</td>
<td>480420</td>
<td>Mandalay</td>
<td>21.98</td>
<td>96.10</td>
<td>76</td>
</tr>
<tr>
<td>25</td>
<td>483270</td>
<td>Chiang Mai</td>
<td>18.78</td>
<td>98.98</td>
<td>314</td>
</tr>
<tr>
<td>26</td>
<td>488190</td>
<td>Hanoi</td>
<td>21.03</td>
<td>105.83</td>
<td>10</td>
</tr>
<tr>
<td>27</td>
<td>501360</td>
<td>Humamehe</td>
<td>53.47</td>
<td>122.37</td>
<td>296</td>
</tr>
<tr>
<td>Nr</td>
<td>WMO</td>
<td>Station</td>
<td>Latitude (°N)</td>
<td>Longitude (°E)</td>
<td>Elevation (m)</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>---------------------</td>
<td>---------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>28</td>
<td>503530</td>
<td>Huma</td>
<td>51.72</td>
<td>126.65</td>
<td>177</td>
</tr>
<tr>
<td>29</td>
<td>504340</td>
<td>Xuguitu Qi Tulihe</td>
<td>50.48</td>
<td>121.68</td>
<td>732</td>
</tr>
<tr>
<td>30</td>
<td>505140</td>
<td>Man Zhou Li</td>
<td>49.60</td>
<td>117.4</td>
<td>646</td>
</tr>
<tr>
<td>31</td>
<td>505270</td>
<td>Hailaer</td>
<td>49.22</td>
<td>119.75</td>
<td>612</td>
</tr>
<tr>
<td>32</td>
<td>505570</td>
<td>NenJiang</td>
<td>49.17</td>
<td>125.23</td>
<td>242</td>
</tr>
<tr>
<td>33</td>
<td>505640</td>
<td>Sunwu</td>
<td>49.43</td>
<td>127.35</td>
<td>234</td>
</tr>
<tr>
<td>34</td>
<td>506320</td>
<td>BoKeTu</td>
<td>48.77</td>
<td>121.92</td>
<td>739</td>
</tr>
<tr>
<td>35</td>
<td>506580</td>
<td>Keshan</td>
<td>48.05</td>
<td>125.88</td>
<td>236</td>
</tr>
<tr>
<td>36</td>
<td>507270</td>
<td>Horqing Youyi Qianq</td>
<td>47.17</td>
<td>119.95</td>
<td>1,027</td>
</tr>
<tr>
<td>37</td>
<td>507450</td>
<td>QiQiHaEr</td>
<td>47.38</td>
<td>123.92</td>
<td>145</td>
</tr>
<tr>
<td>38</td>
<td>507560</td>
<td>Hailun</td>
<td>47.43</td>
<td>126.97</td>
<td>239</td>
</tr>
<tr>
<td>39</td>
<td>507740</td>
<td>Yichun</td>
<td>47.72</td>
<td>128.90</td>
<td>231</td>
</tr>
<tr>
<td>40</td>
<td>507880</td>
<td>Fujin</td>
<td>47.23</td>
<td>131.98</td>
<td>64</td>
</tr>
<tr>
<td>41</td>
<td>508440</td>
<td>Tailai</td>
<td>46.40</td>
<td>123.42</td>
<td>150</td>
</tr>
<tr>
<td>42</td>
<td>508540</td>
<td>Anda</td>
<td>46.38</td>
<td>125.32</td>
<td>149</td>
</tr>
<tr>
<td>43</td>
<td>509150</td>
<td>Dong Ujimqin Qi</td>
<td>45.52</td>
<td>116.97</td>
<td>838</td>
</tr>
<tr>
<td>44</td>
<td>509490</td>
<td>Qian Gorlos</td>
<td>45.12</td>
<td>124.83</td>
<td>134</td>
</tr>
<tr>
<td>45</td>
<td>509530</td>
<td>HaErBin</td>
<td>45.68</td>
<td>126.62</td>
<td>171</td>
</tr>
<tr>
<td>46</td>
<td>509630</td>
<td>Tonghe</td>
<td>45.97</td>
<td>128.73</td>
<td>108</td>
</tr>
<tr>
<td>47</td>
<td>509860</td>
<td>Sangzhi</td>
<td>45.22</td>
<td>127.97</td>
<td>191</td>
</tr>
<tr>
<td>48</td>
<td>509780</td>
<td>Jixi</td>
<td>45.28</td>
<td>130.95</td>
<td>232</td>
</tr>
<tr>
<td>49</td>
<td>510760</td>
<td>Altay</td>
<td>47.73</td>
<td>88.08</td>
<td>735</td>
</tr>
<tr>
<td>50</td>
<td>511330</td>
<td>Tacheng</td>
<td>46.73</td>
<td>83.00</td>
<td>548</td>
</tr>
<tr>
<td>51</td>
<td>511560</td>
<td>Hobokasar</td>
<td>46.78</td>
<td>85.72</td>
<td>1,291</td>
</tr>
<tr>
<td>52</td>
<td>513340</td>
<td>Jinghe</td>
<td>44.62</td>
<td>82.90</td>
<td>320</td>
</tr>
<tr>
<td>53</td>
<td>513560</td>
<td>Shihezi</td>
<td>44.30</td>
<td>86.03</td>
<td>457</td>
</tr>
<tr>
<td>54</td>
<td>513790</td>
<td>Qitai</td>
<td>44.02</td>
<td>89.57</td>
<td>793</td>
</tr>
<tr>
<td>55</td>
<td>514310</td>
<td>Yining</td>
<td>43.95</td>
<td>81.33</td>
<td>662</td>
</tr>
<tr>
<td>56</td>
<td>514630</td>
<td>WuLuMuQi</td>
<td>43.78</td>
<td>87.62</td>
<td>917</td>
</tr>
<tr>
<td>57</td>
<td>514950</td>
<td>Hami Qijiaojing</td>
<td>43.48</td>
<td>91.63</td>
<td>873</td>
</tr>
<tr>
<td>58</td>
<td>515730</td>
<td>Turpan</td>
<td>42.93</td>
<td>89.20</td>
<td>34</td>
</tr>
<tr>
<td>59</td>
<td>516440</td>
<td>Kuqa</td>
<td>41.72</td>
<td>82.95</td>
<td>1,099</td>
</tr>
<tr>
<td>60</td>
<td>517090</td>
<td>Kashii</td>
<td>39.47</td>
<td>75.98</td>
<td>1,288</td>
</tr>
<tr>
<td>61</td>
<td>517160</td>
<td>Bachu</td>
<td>39.80</td>
<td>78.57</td>
<td>1,116</td>
</tr>
<tr>
<td>62</td>
<td>517770</td>
<td>Ruoqiang</td>
<td>39.03</td>
<td>88.17</td>
<td>888</td>
</tr>
<tr>
<td>63</td>
<td>518110</td>
<td>Shache</td>
<td>38.43</td>
<td>77.27</td>
<td>1,231</td>
</tr>
<tr>
<td>64</td>
<td>518280</td>
<td>Hotan</td>
<td>37.13</td>
<td>79.93</td>
<td>1,374</td>
</tr>
<tr>
<td>65</td>
<td>518550</td>
<td>Qiemo</td>
<td>38.15</td>
<td>85.55</td>
<td>1,247</td>
</tr>
<tr>
<td>66</td>
<td>522030</td>
<td>HaMi</td>
<td>42.82</td>
<td>93.52</td>
<td>737</td>
</tr>
<tr>
<td>67</td>
<td>522670</td>
<td>Ejin Qi</td>
<td>41.95</td>
<td>101.07</td>
<td>940</td>
</tr>
<tr>
<td>68</td>
<td>523230</td>
<td>Yemajie</td>
<td>41.58</td>
<td>96.88</td>
<td>1,962</td>
</tr>
<tr>
<td>69</td>
<td>524180</td>
<td>Dunhuang</td>
<td>40.15</td>
<td>94.68</td>
<td>1,138</td>
</tr>
<tr>
<td>70</td>
<td>524240</td>
<td>And</td>
<td>40.53</td>
<td>95.77</td>
<td>1,170</td>
</tr>
<tr>
<td>71</td>
<td>524360</td>
<td>Yumenzhen</td>
<td>40.27</td>
<td>97.03</td>
<td>1,526</td>
</tr>
<tr>
<td>Nr</td>
<td>WMO</td>
<td>Station</td>
<td>Latitude (°N)</td>
<td>Longitude (°E)</td>
<td>Elevation (m)</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>-----------------</td>
<td>---------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>72</td>
<td>524950</td>
<td>Bayan Mod</td>
<td>40.75</td>
<td>104.50</td>
<td>1,329</td>
</tr>
<tr>
<td>73</td>
<td>525330</td>
<td>Jiu Quan</td>
<td>39.77</td>
<td>98.52</td>
<td>1,477</td>
</tr>
<tr>
<td>74</td>
<td>526020</td>
<td>Lenghu</td>
<td>38.83</td>
<td>93.38</td>
<td>2,733</td>
</tr>
<tr>
<td>75</td>
<td>526330</td>
<td>Qilian Tuole</td>
<td>38.82</td>
<td>98.42</td>
<td>3,360</td>
</tr>
<tr>
<td>76</td>
<td>526520</td>
<td>ZhangYe</td>
<td>38.93</td>
<td>100.58</td>
<td>1,483</td>
</tr>
<tr>
<td>77</td>
<td>526810</td>
<td>Minqin</td>
<td>38.63</td>
<td>103.08</td>
<td>1,367</td>
</tr>
<tr>
<td>78</td>
<td>527130</td>
<td>Da Qaidam</td>
<td>37.85</td>
<td>95.37</td>
<td>3,173</td>
</tr>
<tr>
<td>79</td>
<td>527790</td>
<td>Zaoyang</td>
<td>32.15</td>
<td>112.67</td>
<td>127</td>
</tr>
<tr>
<td>80</td>
<td>527870</td>
<td>Tianzhu Wushaoling</td>
<td>37.20</td>
<td>102.87</td>
<td>3,045</td>
</tr>
<tr>
<td>81</td>
<td>528180</td>
<td>Golmud</td>
<td>36.42</td>
<td>94.90</td>
<td>2,807</td>
</tr>
<tr>
<td>82</td>
<td>528360</td>
<td>Dulan</td>
<td>36.30</td>
<td>98.10</td>
<td>3,191</td>
</tr>
<tr>
<td>83</td>
<td>528420</td>
<td>Uulan Caka</td>
<td>36.78</td>
<td>99.08</td>
<td>3,087</td>
</tr>
<tr>
<td>84</td>
<td>528560</td>
<td>Gonghe</td>
<td>36.27</td>
<td>100.62</td>
<td>2,835</td>
</tr>
<tr>
<td>85</td>
<td>528660</td>
<td>XiNing</td>
<td>36.62</td>
<td>101.77</td>
<td>2,261</td>
</tr>
<tr>
<td>86</td>
<td>528890</td>
<td>LanZhou</td>
<td>36.05</td>
<td>103.88</td>
<td>1,517</td>
</tr>
<tr>
<td>87</td>
<td>529570</td>
<td>Tongda</td>
<td>35.27</td>
<td>100.65</td>
<td>3,290</td>
</tr>
<tr>
<td>88</td>
<td>529840</td>
<td>Linxia</td>
<td>35.58</td>
<td>103.18</td>
<td>1,917</td>
</tr>
<tr>
<td>89</td>
<td>529960</td>
<td>Tongwei Huajialing</td>
<td>35.38</td>
<td>105.00</td>
<td>2,450</td>
</tr>
<tr>
<td>90</td>
<td>530680</td>
<td>Erenhot</td>
<td>43.65</td>
<td>111.97</td>
<td>964</td>
</tr>
<tr>
<td>91</td>
<td>531920</td>
<td>Abag Qi</td>
<td>44.02</td>
<td>114.95</td>
<td>1,126</td>
</tr>
<tr>
<td>92</td>
<td>532760</td>
<td>Sonid Youqi</td>
<td>42.40</td>
<td>112.90</td>
<td>1,150</td>
</tr>
<tr>
<td>93</td>
<td>533360</td>
<td>Urad Zhongqi</td>
<td>41.57</td>
<td>108.52</td>
<td>1,288</td>
</tr>
<tr>
<td>94</td>
<td>533520</td>
<td>Darhan Muminggan</td>
<td>41.70</td>
<td>110.43</td>
<td>1,375</td>
</tr>
<tr>
<td>95</td>
<td>533910</td>
<td>Huade</td>
<td>41.90</td>
<td>114.00</td>
<td>1,482</td>
</tr>
<tr>
<td>96</td>
<td>534630</td>
<td>HuHeHaoTe</td>
<td>40.80</td>
<td>111.63</td>
<td>1,063</td>
</tr>
<tr>
<td>97</td>
<td>534870</td>
<td>Datong</td>
<td>40.10</td>
<td>113.33</td>
<td>1,067</td>
</tr>
<tr>
<td>98</td>
<td>535020</td>
<td>Alxa Youqi Jartai</td>
<td>39.78</td>
<td>105.75</td>
<td>1,031</td>
</tr>
<tr>
<td>99</td>
<td>535290</td>
<td>Otog Qi</td>
<td>39.10</td>
<td>107.98</td>
<td>1,380</td>
</tr>
<tr>
<td>100</td>
<td>535880</td>
<td>Wutaishan</td>
<td>39.03</td>
<td>113.53</td>
<td>2,895</td>
</tr>
<tr>
<td>101</td>
<td>535930</td>
<td>Weixian</td>
<td>39.83</td>
<td>114.57</td>
<td>909</td>
</tr>
<tr>
<td>102</td>
<td>536020</td>
<td>Alxa Zuqiu</td>
<td>38.83</td>
<td>105.67</td>
<td>1,561</td>
</tr>
<tr>
<td>103</td>
<td>536140</td>
<td>YinChuan</td>
<td>38.48</td>
<td>106.22</td>
<td>1,111</td>
</tr>
<tr>
<td>104</td>
<td>536460</td>
<td>YuLin</td>
<td>38.23</td>
<td>109.70</td>
<td>1,057</td>
</tr>
<tr>
<td>105</td>
<td>536640</td>
<td>Xingxian</td>
<td>38.47</td>
<td>111.13</td>
<td>1,012</td>
</tr>
<tr>
<td>106</td>
<td>536730</td>
<td>Yuanping</td>
<td>38.75</td>
<td>112.70</td>
<td>838</td>
</tr>
<tr>
<td>107</td>
<td>536980</td>
<td>Shijiazhuang</td>
<td>38.03</td>
<td>114.42</td>
<td>81</td>
</tr>
<tr>
<td>108</td>
<td>537050</td>
<td>Zhongning</td>
<td>37.48</td>
<td>105.67</td>
<td>1,183</td>
</tr>
<tr>
<td>109</td>
<td>537230</td>
<td>Yanchi</td>
<td>37.78</td>
<td>107.40</td>
<td>1,347</td>
</tr>
<tr>
<td>110</td>
<td>537720</td>
<td>TaiYuan</td>
<td>37.78</td>
<td>112.55</td>
<td>777</td>
</tr>
<tr>
<td>111</td>
<td>537980</td>
<td>Xingtai</td>
<td>37.07</td>
<td>114.50</td>
<td>76</td>
</tr>
<tr>
<td>112</td>
<td>538450</td>
<td>Yanan</td>
<td>36.60</td>
<td>109.50</td>
<td>957</td>
</tr>
<tr>
<td>113</td>
<td>538630</td>
<td>Jiexiu</td>
<td>37.05</td>
<td>111.93</td>
<td>748</td>
</tr>
<tr>
<td>114</td>
<td>538980</td>
<td>Anyang</td>
<td>36.12</td>
<td>114.37</td>
<td>75</td>
</tr>
<tr>
<td>115</td>
<td>539150</td>
<td>Pingliang</td>
<td>35.55</td>
<td>106.67</td>
<td>1,346</td>
</tr>
<tr>
<td>Nr</td>
<td>WMO</td>
<td>Station</td>
<td>Latitude (°N)</td>
<td>Longitude (°E)</td>
<td>Elevation (m)</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>--------------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>116</td>
<td>539230</td>
<td>Qingyang Xifengzhen</td>
<td>35.73</td>
<td>107.63</td>
<td>1,421</td>
</tr>
<tr>
<td>117</td>
<td>539590</td>
<td>Yuncheng</td>
<td>35.03</td>
<td>111.02</td>
<td>376</td>
</tr>
<tr>
<td>118</td>
<td>540260</td>
<td>Jarud Qi</td>
<td>44.57</td>
<td>120.90</td>
<td>265</td>
</tr>
<tr>
<td>119</td>
<td>540270</td>
<td>Bairin Zuoqi</td>
<td>43.98</td>
<td>119.40</td>
<td>484</td>
</tr>
<tr>
<td>120</td>
<td>540940</td>
<td>MuDanJiang</td>
<td>44.57</td>
<td>129.60</td>
<td>241</td>
</tr>
<tr>
<td>121</td>
<td>540960</td>
<td>Suifenhe</td>
<td>44.38</td>
<td>131.15</td>
<td>496</td>
</tr>
<tr>
<td>122</td>
<td>541020</td>
<td>Xilin Hot</td>
<td>43.95</td>
<td>116.07</td>
<td>989</td>
</tr>
<tr>
<td>123</td>
<td>541150</td>
<td>Linxi</td>
<td>43.60</td>
<td>118.07</td>
<td>779</td>
</tr>
<tr>
<td>124</td>
<td>541350</td>
<td>Tonghao</td>
<td>43.60</td>
<td>122.27</td>
<td>178</td>
</tr>
<tr>
<td>125</td>
<td>541570</td>
<td>Siping</td>
<td>43.18</td>
<td>124.33</td>
<td>164</td>
</tr>
<tr>
<td>126</td>
<td>541610</td>
<td>ChangChun</td>
<td>43.88</td>
<td>125.33</td>
<td>236</td>
</tr>
<tr>
<td>127</td>
<td>541860</td>
<td>Dunhua</td>
<td>43.37</td>
<td>128.20</td>
<td>523</td>
</tr>
<tr>
<td>128</td>
<td>542080</td>
<td>Duolun</td>
<td>42.18</td>
<td>116.47</td>
<td>1,245</td>
</tr>
<tr>
<td>129</td>
<td>542180</td>
<td>Chifen</td>
<td>42.27</td>
<td>118.97</td>
<td>571</td>
</tr>
<tr>
<td>130</td>
<td>542370</td>
<td>Fuxin</td>
<td>42.03</td>
<td>121.65</td>
<td>144</td>
</tr>
<tr>
<td>131</td>
<td>542920</td>
<td>Yanji</td>
<td>42.88</td>
<td>129.47</td>
<td>176</td>
</tr>
<tr>
<td>132</td>
<td>543110</td>
<td>Weichang</td>
<td>41.93</td>
<td>117.75</td>
<td>842</td>
</tr>
<tr>
<td>133</td>
<td>543240</td>
<td>Chaoyang</td>
<td>41.55</td>
<td>120.45</td>
<td>169</td>
</tr>
<tr>
<td>134</td>
<td>543370</td>
<td>Jinzhou</td>
<td>41.13</td>
<td>121.12</td>
<td>65</td>
</tr>
<tr>
<td>135</td>
<td>543420</td>
<td>ShenYang</td>
<td>41.77</td>
<td>123.43</td>
<td>41</td>
</tr>
<tr>
<td>136</td>
<td>543630</td>
<td>Tonghua</td>
<td>41.68</td>
<td>125.90</td>
<td>402</td>
</tr>
<tr>
<td>137</td>
<td>543740</td>
<td>Linjiang</td>
<td>41.72</td>
<td>126.92</td>
<td>332</td>
</tr>
<tr>
<td>138</td>
<td>544010</td>
<td>Zhangjiakou</td>
<td>40.78</td>
<td>114.88</td>
<td>724</td>
</tr>
<tr>
<td>139</td>
<td>544230</td>
<td>Chengde</td>
<td>40.97</td>
<td>117.93</td>
<td>375</td>
</tr>
<tr>
<td>140</td>
<td>544550</td>
<td>Xingcheng</td>
<td>40.58</td>
<td>120.70</td>
<td>8</td>
</tr>
<tr>
<td>141</td>
<td>544710</td>
<td>Yingkou</td>
<td>40.67</td>
<td>122.20</td>
<td>4</td>
</tr>
<tr>
<td>142</td>
<td>544760</td>
<td>Gaixian Xiongyue</td>
<td>40.17</td>
<td>122.15</td>
<td>20</td>
</tr>
<tr>
<td>143</td>
<td>544930</td>
<td>Kuanding</td>
<td>40.72</td>
<td>124.78</td>
<td>260</td>
</tr>
<tr>
<td>144</td>
<td>544970</td>
<td>Dandong</td>
<td>40.05</td>
<td>124.32</td>
<td>15</td>
</tr>
<tr>
<td>145</td>
<td>545110</td>
<td>BeiJing</td>
<td>39.93</td>
<td>116.28</td>
<td>54</td>
</tr>
<tr>
<td>146</td>
<td>545270</td>
<td>TianJin</td>
<td>39.10</td>
<td>117.17</td>
<td>3</td>
</tr>
<tr>
<td>147</td>
<td>545340</td>
<td>Tangshan</td>
<td>39.67</td>
<td>118.15</td>
<td>29</td>
</tr>
<tr>
<td>148</td>
<td>546020</td>
<td>BaoDing</td>
<td>38.83</td>
<td>115.57</td>
<td>17</td>
</tr>
<tr>
<td>149</td>
<td>546160</td>
<td>Cangzhou</td>
<td>38.33</td>
<td>116.83</td>
<td>9</td>
</tr>
<tr>
<td>150</td>
<td>546620</td>
<td>DaLian</td>
<td>38.90</td>
<td>121.63</td>
<td>92</td>
</tr>
<tr>
<td>151</td>
<td>547140</td>
<td>Dezhou</td>
<td>37.43</td>
<td>116.32</td>
<td>21</td>
</tr>
<tr>
<td>152</td>
<td>547250</td>
<td>Huimin</td>
<td>37.50</td>
<td>117.53</td>
<td>11</td>
</tr>
<tr>
<td>153</td>
<td>547510</td>
<td>Chang Island</td>
<td>37.93</td>
<td>120.72</td>
<td>40</td>
</tr>
<tr>
<td>154</td>
<td>547650</td>
<td>YanTai</td>
<td>37.53</td>
<td>121.40</td>
<td>46</td>
</tr>
<tr>
<td>155</td>
<td>547760</td>
<td>Rongchengshanto</td>
<td>37.40</td>
<td>122.68</td>
<td>47</td>
</tr>
<tr>
<td>156</td>
<td>548230</td>
<td>JiNan</td>
<td>36.68</td>
<td>116.98</td>
<td>51</td>
</tr>
<tr>
<td>157</td>
<td>548430</td>
<td>Weifang</td>
<td>36.70</td>
<td>119.08</td>
<td>44</td>
</tr>
<tr>
<td>158</td>
<td>548520</td>
<td>Laiyang</td>
<td>36.93</td>
<td>120.70</td>
<td>30</td>
</tr>
<tr>
<td>159</td>
<td>548570</td>
<td>QingDao</td>
<td>36.07</td>
<td>120.33</td>
<td>76</td>
</tr>
<tr>
<td>160</td>
<td>549060</td>
<td>Heze</td>
<td>35.25</td>
<td>115.43</td>
<td>49</td>
</tr>
<tr>
<td>Nr</td>
<td>WMO</td>
<td>Station</td>
<td>Latitude (°N)</td>
<td>Longitude (°E)</td>
<td>Elevation (m)</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>161</td>
<td>549160</td>
<td>Yanzhou</td>
<td>35.57</td>
<td>116.85</td>
<td>51</td>
</tr>
<tr>
<td>162</td>
<td>549360</td>
<td>Juxian</td>
<td>35.58</td>
<td>118.83</td>
<td>107</td>
</tr>
<tr>
<td>163</td>
<td>549380</td>
<td>Linyi</td>
<td>35.05</td>
<td>118.35</td>
<td>87</td>
</tr>
<tr>
<td>164</td>
<td>552280</td>
<td>Geershiquanhe</td>
<td>32.50</td>
<td>80.08</td>
<td>4,728</td>
</tr>
<tr>
<td>165</td>
<td>552990</td>
<td>Nagqu</td>
<td>31.48</td>
<td>92.07</td>
<td>4,507</td>
</tr>
<tr>
<td>166</td>
<td>555780</td>
<td>Rikaze</td>
<td>29.25</td>
<td>88.88</td>
<td>3,836</td>
</tr>
<tr>
<td>167</td>
<td>555910</td>
<td>LaSa</td>
<td>29.70</td>
<td>91.13</td>
<td>3,658</td>
</tr>
<tr>
<td>168</td>
<td>556960</td>
<td>Lhunze</td>
<td>28.42</td>
<td>92.47</td>
<td>3,860</td>
</tr>
<tr>
<td>169</td>
<td>557730</td>
<td>Yadong</td>
<td>27.73</td>
<td>89.08</td>
<td>4,301</td>
</tr>
<tr>
<td>170</td>
<td>560210</td>
<td>Qumarleb</td>
<td>34.13</td>
<td>95.78</td>
<td>4,175</td>
</tr>
<tr>
<td>171</td>
<td>560290</td>
<td>Yushu</td>
<td>33.02</td>
<td>97.02</td>
<td>3,681</td>
</tr>
<tr>
<td>172</td>
<td>560330</td>
<td>Madoi</td>
<td>34.92</td>
<td>98.22</td>
<td>4,272</td>
</tr>
<tr>
<td>173</td>
<td>560800</td>
<td>Hezuo</td>
<td>35.00</td>
<td>102.90</td>
<td>2,910</td>
</tr>
<tr>
<td>174</td>
<td>560930</td>
<td>Minxian</td>
<td>34.43</td>
<td>104.02</td>
<td>2,314</td>
</tr>
<tr>
<td>175</td>
<td>560960</td>
<td>Wudu</td>
<td>33.40</td>
<td>104.92</td>
<td>1,079</td>
</tr>
<tr>
<td>176</td>
<td>561370</td>
<td>Qamdo</td>
<td>31.15</td>
<td>97.17</td>
<td>3,306</td>
</tr>
<tr>
<td>177</td>
<td>561460</td>
<td>Garze</td>
<td>31.62</td>
<td>100.00</td>
<td>3,393</td>
</tr>
<tr>
<td>178</td>
<td>561780</td>
<td>Xiaojin</td>
<td>31.00</td>
<td>102.35</td>
<td>2,369</td>
</tr>
<tr>
<td>179</td>
<td>561820</td>
<td>Songpan</td>
<td>32.65</td>
<td>103.57</td>
<td>2,850</td>
</tr>
<tr>
<td>180</td>
<td>561930</td>
<td>Pingwu</td>
<td>32.42</td>
<td>104.52</td>
<td>876</td>
</tr>
<tr>
<td>181</td>
<td>561960</td>
<td>Mianyang</td>
<td>31.47</td>
<td>104.68</td>
<td>470</td>
</tr>
<tr>
<td>182</td>
<td>562570</td>
<td>Litang</td>
<td>30.00</td>
<td>100.27</td>
<td>3,948</td>
</tr>
<tr>
<td>183</td>
<td>562870</td>
<td>Yaan</td>
<td>29.98</td>
<td>103.00</td>
<td>627</td>
</tr>
<tr>
<td>184</td>
<td>562940</td>
<td>ChengDu</td>
<td>30.67</td>
<td>104.02</td>
<td>506</td>
</tr>
<tr>
<td>185</td>
<td>563120</td>
<td>Nyingchi</td>
<td>29.57</td>
<td>94.47</td>
<td>3,000</td>
</tr>
<tr>
<td>186</td>
<td>563740</td>
<td>Kangding</td>
<td>30.05</td>
<td>101.97</td>
<td>2,617</td>
</tr>
<tr>
<td>187</td>
<td>563860</td>
<td>Leshan</td>
<td>29.57</td>
<td>103.75</td>
<td>424</td>
</tr>
<tr>
<td>188</td>
<td>564440</td>
<td>Deqen</td>
<td>28.50</td>
<td>98.90</td>
<td>3,488</td>
</tr>
<tr>
<td>189</td>
<td>564620</td>
<td>Jincheng</td>
<td>29.00</td>
<td>101.50</td>
<td>2,987</td>
</tr>
<tr>
<td>190</td>
<td>564850</td>
<td>Leibo</td>
<td>28.27</td>
<td>103.58</td>
<td>1,474</td>
</tr>
<tr>
<td>191</td>
<td>564920</td>
<td>Yibin</td>
<td>28.80</td>
<td>104.60</td>
<td>340</td>
</tr>
<tr>
<td>192</td>
<td>565710</td>
<td>XiChang</td>
<td>27.90</td>
<td>102.27</td>
<td>1,590</td>
</tr>
<tr>
<td>193</td>
<td>566510</td>
<td>Lijiang</td>
<td>26.87</td>
<td>100.22</td>
<td>2,393</td>
</tr>
<tr>
<td>194</td>
<td>566710</td>
<td>Huili</td>
<td>26.65</td>
<td>102.25</td>
<td>1,787</td>
</tr>
<tr>
<td>195</td>
<td>566910</td>
<td>Weining</td>
<td>26.87</td>
<td>104.28</td>
<td>2,237</td>
</tr>
<tr>
<td>196</td>
<td>567390</td>
<td>TengChong</td>
<td>25.12</td>
<td>98.48</td>
<td>1,647</td>
</tr>
<tr>
<td>197</td>
<td>567510</td>
<td>Dali</td>
<td>25.70</td>
<td>100.18</td>
<td>1,190</td>
</tr>
<tr>
<td>198</td>
<td>567780</td>
<td>KunMing</td>
<td>25.02</td>
<td>102.68</td>
<td>1,891</td>
</tr>
<tr>
<td>199</td>
<td>567930</td>
<td>Panxian</td>
<td>25.78</td>
<td>104.62</td>
<td>1,527</td>
</tr>
<tr>
<td>200</td>
<td>569510</td>
<td>Lincang</td>
<td>23.95</td>
<td>100.22</td>
<td>1,463</td>
</tr>
<tr>
<td>201</td>
<td>569590</td>
<td>Jinghong</td>
<td>22.00</td>
<td>100.80</td>
<td>552</td>
</tr>
<tr>
<td>202</td>
<td>569640</td>
<td>Puer Simao</td>
<td>22.67</td>
<td>101.40</td>
<td>1,302</td>
</tr>
<tr>
<td>203</td>
<td>569850</td>
<td>Mengzi</td>
<td>23.38</td>
<td>103.38</td>
<td>1,300</td>
</tr>
<tr>
<td>204</td>
<td>569890</td>
<td>Hekou</td>
<td>22.60</td>
<td>103.95</td>
<td>1,367</td>
</tr>
<tr>
<td>205</td>
<td>570060</td>
<td>TianShui</td>
<td>34.58</td>
<td>105.75</td>
<td>1,131</td>
</tr>
<tr>
<td>Nr</td>
<td>WMO</td>
<td>Station</td>
<td>Latitude (°N)</td>
<td>Longitude (°E)</td>
<td>Elevation (m)</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>---------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>206</td>
<td>570360</td>
<td>XiAn</td>
<td>34.30</td>
<td>108.93</td>
<td>396</td>
</tr>
<tr>
<td>207</td>
<td>570670</td>
<td>Lushi</td>
<td>34.05</td>
<td>111.03</td>
<td>568</td>
</tr>
<tr>
<td>208</td>
<td>570730</td>
<td>Luoyang</td>
<td>34.67</td>
<td>112.42</td>
<td>154</td>
</tr>
<tr>
<td>209</td>
<td>570830</td>
<td>ZhengZhou</td>
<td>34.72</td>
<td>113.65</td>
<td>110</td>
</tr>
<tr>
<td>210</td>
<td>571270</td>
<td>HanZhong</td>
<td>33.07</td>
<td>107.20</td>
<td>508</td>
</tr>
<tr>
<td>211</td>
<td>571780</td>
<td>Nanyang</td>
<td>33.03</td>
<td>112.58</td>
<td>129</td>
</tr>
<tr>
<td>212</td>
<td>571930</td>
<td>Xihua</td>
<td>33.78</td>
<td>114.52</td>
<td>52</td>
</tr>
<tr>
<td>213</td>
<td>572370</td>
<td>Wanyuan</td>
<td>32.07</td>
<td>108.03</td>
<td>674</td>
</tr>
<tr>
<td>214</td>
<td>572450</td>
<td>Ankang</td>
<td>32.72</td>
<td>109.03</td>
<td>290</td>
</tr>
<tr>
<td>215</td>
<td>572530</td>
<td>Yunxian</td>
<td>32.85</td>
<td>110.82</td>
<td>201</td>
</tr>
<tr>
<td>216</td>
<td>572590</td>
<td>Fangxian</td>
<td>32.03</td>
<td>110.77</td>
<td>435</td>
</tr>
<tr>
<td>217</td>
<td>572650</td>
<td>Laohekou</td>
<td>32.38</td>
<td>111.67</td>
<td>90</td>
</tr>
<tr>
<td>218</td>
<td>572900</td>
<td>Zhumadian</td>
<td>33.00</td>
<td>114.02</td>
<td>82</td>
</tr>
<tr>
<td>219</td>
<td>572970</td>
<td>Xinyang</td>
<td>32.13</td>
<td>114.05</td>
<td>114</td>
</tr>
<tr>
<td>220</td>
<td>573130</td>
<td>Bazhong</td>
<td>31.85</td>
<td>106.77</td>
<td>360</td>
</tr>
<tr>
<td>221</td>
<td>573780</td>
<td>Zhongxiang</td>
<td>31.17</td>
<td>112.57</td>
<td>65</td>
</tr>
<tr>
<td>222</td>
<td>573990</td>
<td>Macheng</td>
<td>31.18</td>
<td>114.97</td>
<td>59</td>
</tr>
<tr>
<td>223</td>
<td>574050</td>
<td>Suining</td>
<td>30.50</td>
<td>105.58</td>
<td>278</td>
</tr>
<tr>
<td>224</td>
<td>574110</td>
<td>Nanchong</td>
<td>30.80</td>
<td>106.08</td>
<td>297</td>
</tr>
<tr>
<td>225</td>
<td>574470</td>
<td>Enshi</td>
<td>30.28</td>
<td>109.47</td>
<td>457</td>
</tr>
<tr>
<td>226</td>
<td>574610</td>
<td>YiChang</td>
<td>30.70</td>
<td>111.30</td>
<td>133</td>
</tr>
<tr>
<td>227</td>
<td>574760</td>
<td>Jingzhou</td>
<td>30.33</td>
<td>112.18</td>
<td>32</td>
</tr>
<tr>
<td>228</td>
<td>574940</td>
<td>WuHan</td>
<td>30.62</td>
<td>114.13</td>
<td>23</td>
</tr>
<tr>
<td>229</td>
<td>575040</td>
<td>Neijiang</td>
<td>29.58</td>
<td>105.05</td>
<td>347</td>
</tr>
<tr>
<td>230</td>
<td>575160</td>
<td>ChongQing</td>
<td>29.58</td>
<td>106.47</td>
<td>259</td>
</tr>
<tr>
<td>231</td>
<td>575370</td>
<td>Pengshui</td>
<td>29.30</td>
<td>108.17</td>
<td>310</td>
</tr>
<tr>
<td>232</td>
<td>575840</td>
<td>Yueyang</td>
<td>29.38</td>
<td>113.08</td>
<td>51</td>
</tr>
<tr>
<td>233</td>
<td>576060</td>
<td>Tongzi</td>
<td>28.13</td>
<td>105.83</td>
<td>972</td>
</tr>
<tr>
<td>234</td>
<td>576330</td>
<td>Youyang</td>
<td>28.83</td>
<td>108.77</td>
<td>663</td>
</tr>
<tr>
<td>235</td>
<td>576550</td>
<td>Yuanling</td>
<td>28.47</td>
<td>110.40</td>
<td>151</td>
</tr>
<tr>
<td>236</td>
<td>576620</td>
<td>Changde</td>
<td>29.05</td>
<td>111.68</td>
<td>35</td>
</tr>
<tr>
<td>237</td>
<td>576790</td>
<td>ChangSha</td>
<td>28.20</td>
<td>113.08</td>
<td>44</td>
</tr>
<tr>
<td>238</td>
<td>577130</td>
<td>Zunyi</td>
<td>27.70</td>
<td>106.88</td>
<td>843</td>
</tr>
<tr>
<td>239</td>
<td>577220</td>
<td>Meitan</td>
<td>27.77</td>
<td>107.47</td>
<td>791</td>
</tr>
<tr>
<td>240</td>
<td>577310</td>
<td>Sinan</td>
<td>27.95</td>
<td>108.25</td>
<td>416</td>
</tr>
<tr>
<td>241</td>
<td>577450</td>
<td>ZhiJiang</td>
<td>27.45</td>
<td>109.68</td>
<td>272</td>
</tr>
<tr>
<td>242</td>
<td>577660</td>
<td>Shaoyang</td>
<td>27.23</td>
<td>111.47</td>
<td>248</td>
</tr>
<tr>
<td>243</td>
<td>577930</td>
<td>Yichun</td>
<td>27.80</td>
<td>114.38</td>
<td>131</td>
</tr>
<tr>
<td>244</td>
<td>577990</td>
<td>Jian</td>
<td>27.12</td>
<td>114.97</td>
<td>76</td>
</tr>
<tr>
<td>245</td>
<td>578160</td>
<td>GuiYang</td>
<td>26.58</td>
<td>106.72</td>
<td>1,071</td>
</tr>
<tr>
<td>246</td>
<td>578530</td>
<td>Wugang</td>
<td>26.73</td>
<td>110.63</td>
<td>341</td>
</tr>
<tr>
<td>247</td>
<td>578660</td>
<td>Lingling</td>
<td>26.23</td>
<td>111.62</td>
<td>172</td>
</tr>
<tr>
<td>248</td>
<td>578720</td>
<td>Hengyang</td>
<td>26.90</td>
<td>112.60</td>
<td>103</td>
</tr>
<tr>
<td>249</td>
<td>578960</td>
<td>Suichang</td>
<td>26.33</td>
<td>114.50</td>
<td>126</td>
</tr>
<tr>
<td>250</td>
<td>579020</td>
<td>Xingren</td>
<td>25.43</td>
<td>105.18</td>
<td>1,378</td>
</tr>
<tr>
<td>Nr</td>
<td>WMO</td>
<td>Station</td>
<td>Latitude (°N)</td>
<td>Longitude (°E)</td>
<td>Elevation (m)</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>251</td>
<td>579160</td>
<td>Luodian</td>
<td>25.43</td>
<td>106.77</td>
<td>440</td>
</tr>
<tr>
<td>252</td>
<td>579570</td>
<td>Guilin</td>
<td>25.33</td>
<td>110.30</td>
<td>161</td>
</tr>
<tr>
<td>253</td>
<td>579720</td>
<td>Chenzhou</td>
<td>25.80</td>
<td>113.03</td>
<td>184</td>
</tr>
<tr>
<td>254</td>
<td>579930</td>
<td>Ganzhou</td>
<td>25.85</td>
<td>114.95</td>
<td>123</td>
</tr>
<tr>
<td>255</td>
<td>580270</td>
<td>XuZhou</td>
<td>34.28</td>
<td>117.15</td>
<td>41</td>
</tr>
<tr>
<td>256</td>
<td>580400</td>
<td>Dayishan</td>
<td>34.83</td>
<td>119.13</td>
<td>10</td>
</tr>
<tr>
<td>257</td>
<td>581020</td>
<td>Haoxian</td>
<td>33.87</td>
<td>115.77</td>
<td>37</td>
</tr>
<tr>
<td>258</td>
<td>581440</td>
<td>QingJiang</td>
<td>33.60</td>
<td>119.03</td>
<td>17</td>
</tr>
<tr>
<td>259</td>
<td>581500</td>
<td>Sheyang</td>
<td>33.77</td>
<td>120.25</td>
<td>2</td>
</tr>
<tr>
<td>260</td>
<td>582030</td>
<td>Fuyang</td>
<td>32.92</td>
<td>115.82</td>
<td>30</td>
</tr>
<tr>
<td>261</td>
<td>582080</td>
<td>Gushi</td>
<td>32.17</td>
<td>115.67</td>
<td>58</td>
</tr>
<tr>
<td>262</td>
<td>582210</td>
<td>Bengfu</td>
<td>32.95</td>
<td>117.38</td>
<td>18</td>
</tr>
<tr>
<td>263</td>
<td>582380</td>
<td>NanJing</td>
<td>32.05</td>
<td>118.78</td>
<td>8</td>
</tr>
<tr>
<td>264</td>
<td>582510</td>
<td>Dongtai</td>
<td>32.85</td>
<td>120.32</td>
<td>4</td>
</tr>
<tr>
<td>265</td>
<td>582590</td>
<td>Nantong</td>
<td>32.02</td>
<td>120.85</td>
<td>5</td>
</tr>
<tr>
<td>266</td>
<td>583140</td>
<td>HuoShan</td>
<td>31.40</td>
<td>116.32</td>
<td>68</td>
</tr>
<tr>
<td>267</td>
<td>583210</td>
<td>Hefei</td>
<td>31.87</td>
<td>117.23</td>
<td>27</td>
</tr>
<tr>
<td>268</td>
<td>583340</td>
<td>Wuhu</td>
<td>31.33</td>
<td>118.38</td>
<td>14</td>
</tr>
<tr>
<td>269</td>
<td>583620</td>
<td>ShangHai</td>
<td>31.40</td>
<td>121.47</td>
<td>4</td>
</tr>
<tr>
<td>270</td>
<td>583670</td>
<td>ShangHai</td>
<td>31.17</td>
<td>121.43</td>
<td>5</td>
</tr>
<tr>
<td>271</td>
<td>584070</td>
<td>Huangshi</td>
<td>30.25</td>
<td>115.05</td>
<td>19</td>
</tr>
<tr>
<td>272</td>
<td>584240</td>
<td>AnQing</td>
<td>30.53</td>
<td>117.05</td>
<td>19</td>
</tr>
<tr>
<td>273</td>
<td>584570</td>
<td>HangZhou</td>
<td>30.23</td>
<td>120.17</td>
<td>41</td>
</tr>
<tr>
<td>274</td>
<td>584770</td>
<td>Dinghai</td>
<td>30.03</td>
<td>122.12</td>
<td>35</td>
</tr>
<tr>
<td>275</td>
<td>585020</td>
<td>Jiujiang</td>
<td>29.73</td>
<td>116.00</td>
<td>33</td>
</tr>
<tr>
<td>276</td>
<td>585310</td>
<td>Tunxi</td>
<td>29.72</td>
<td>118.28</td>
<td>145</td>
</tr>
<tr>
<td>277</td>
<td>585560</td>
<td>Shengxian</td>
<td>29.60</td>
<td>120.82</td>
<td>104</td>
</tr>
<tr>
<td>278</td>
<td>586060</td>
<td>NanChang</td>
<td>28.60</td>
<td>115.92</td>
<td>46</td>
</tr>
<tr>
<td>279</td>
<td>586330</td>
<td>QuZhou</td>
<td>28.97</td>
<td>118.88</td>
<td>66</td>
</tr>
<tr>
<td>280</td>
<td>586460</td>
<td>Li Shui</td>
<td>28.45</td>
<td>119.92</td>
<td>60</td>
</tr>
<tr>
<td>281</td>
<td>586590</td>
<td>WenZhou</td>
<td>28.02</td>
<td>120.67</td>
<td>7</td>
</tr>
<tr>
<td>282</td>
<td>586650</td>
<td>Huangyan</td>
<td>28.63</td>
<td>121.42</td>
<td>1</td>
</tr>
<tr>
<td>283</td>
<td>587310</td>
<td>Pucheng</td>
<td>27.92</td>
<td>118.53</td>
<td>276</td>
</tr>
<tr>
<td>284</td>
<td>587340</td>
<td>Jianyang</td>
<td>27.33</td>
<td>118.12</td>
<td>181</td>
</tr>
<tr>
<td>285</td>
<td>587540</td>
<td>Fuding</td>
<td>27.33</td>
<td>120.20</td>
<td>36</td>
</tr>
<tr>
<td>286</td>
<td>588130</td>
<td>Guangchang</td>
<td>26.85</td>
<td>116.33</td>
<td>143</td>
</tr>
<tr>
<td>287</td>
<td>588340</td>
<td>Nanping</td>
<td>26.65</td>
<td>118.17</td>
<td>125</td>
</tr>
<tr>
<td>288</td>
<td>588470</td>
<td>Fuzhou</td>
<td>26.08</td>
<td>119.28</td>
<td>83</td>
</tr>
<tr>
<td>289</td>
<td>588530</td>
<td>Taishan</td>
<td>27.00</td>
<td>120.70</td>
<td>106</td>
</tr>
<tr>
<td>290</td>
<td>589210</td>
<td>YongAn</td>
<td>25.97</td>
<td>117.35</td>
<td>206</td>
</tr>
<tr>
<td>291</td>
<td>589310</td>
<td>Dehui Jiuxianshan</td>
<td>25.72</td>
<td>118.10</td>
<td>1,653</td>
</tr>
<tr>
<td>292</td>
<td>589650</td>
<td>Taoyuan</td>
<td>25.05</td>
<td>121.22</td>
<td>48</td>
</tr>
<tr>
<td>293</td>
<td>589740</td>
<td>Taipei</td>
<td>25.07</td>
<td>121.53</td>
<td>6</td>
</tr>
<tr>
<td>294</td>
<td>590070</td>
<td>Guangnan</td>
<td>24.70</td>
<td>105.07</td>
<td>1,249</td>
</tr>
<tr>
<td>295</td>
<td>590460</td>
<td>LinZhou</td>
<td>24.47</td>
<td>109.37</td>
<td>97</td>
</tr>
</tbody>
</table>
### Table B.1 (cont’d 7)

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Station</th>
<th>Latitude (°N)</th>
<th>Longitude (°E)</th>
<th>Elevation (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>296</td>
<td>590720</td>
<td>Lianxian</td>
<td>24.78</td>
<td>112.38</td>
<td>97</td>
</tr>
<tr>
<td>297</td>
<td>590820</td>
<td>Shaoguan</td>
<td>24.80</td>
<td>113.58</td>
<td>69</td>
</tr>
<tr>
<td>298</td>
<td>591170</td>
<td>Meixian</td>
<td>24.27</td>
<td>116.10</td>
<td>87</td>
</tr>
<tr>
<td>299</td>
<td>591260</td>
<td>Zhangzhou</td>
<td>24.50</td>
<td>117.65</td>
<td>28</td>
</tr>
<tr>
<td>300</td>
<td>591340</td>
<td>XiaMen</td>
<td>24.48</td>
<td>118.07</td>
<td>139</td>
</tr>
<tr>
<td>301</td>
<td>591580</td>
<td>Taizhong</td>
<td>24.15</td>
<td>120.70</td>
<td>78</td>
</tr>
<tr>
<td>302</td>
<td>592110</td>
<td>Bose</td>
<td>23.90</td>
<td>106.60</td>
<td>173</td>
</tr>
<tr>
<td>303</td>
<td>592540</td>
<td>Guiping</td>
<td>23.40</td>
<td>110.08</td>
<td>42</td>
</tr>
<tr>
<td>304</td>
<td>592650</td>
<td>WuZhou</td>
<td>23.48</td>
<td>111.30</td>
<td>119</td>
</tr>
<tr>
<td>305</td>
<td>592870</td>
<td>GuangZhou</td>
<td>23.13</td>
<td>113.32</td>
<td>6</td>
</tr>
<tr>
<td>306</td>
<td>592930</td>
<td>Heyuan</td>
<td>23.73</td>
<td>114.68</td>
<td>41</td>
</tr>
<tr>
<td>307</td>
<td>593160</td>
<td>ShanTou</td>
<td>23.40</td>
<td>116.68</td>
<td>1</td>
</tr>
<tr>
<td>308</td>
<td>593450</td>
<td>Magong</td>
<td>23.52</td>
<td>119.57</td>
<td>22</td>
</tr>
<tr>
<td>309</td>
<td>593580</td>
<td>Tainan</td>
<td>23.00</td>
<td>120.23</td>
<td>14</td>
</tr>
<tr>
<td>310</td>
<td>593620</td>
<td>Hualian</td>
<td>24.02</td>
<td>121.62</td>
<td>14</td>
</tr>
<tr>
<td>311</td>
<td>594170</td>
<td>Longzhou</td>
<td>22.37</td>
<td>106.75</td>
<td>128</td>
</tr>
<tr>
<td>312</td>
<td>594310</td>
<td>NanNing</td>
<td>22.82</td>
<td>108.35</td>
<td>72</td>
</tr>
<tr>
<td>313</td>
<td>594530</td>
<td>Yulin</td>
<td>22.63</td>
<td>110.17</td>
<td>81</td>
</tr>
<tr>
<td>314</td>
<td>594560</td>
<td>Xinyi</td>
<td>22.35</td>
<td>110.93</td>
<td>84</td>
</tr>
<tr>
<td>315</td>
<td>594930</td>
<td>Shenzhen</td>
<td>22.55</td>
<td>114.10</td>
<td>18</td>
</tr>
<tr>
<td>316</td>
<td>595010</td>
<td>Haifen Shanwei</td>
<td>22.78</td>
<td>115.37</td>
<td>4</td>
</tr>
<tr>
<td>317</td>
<td>595590</td>
<td>Hengchun</td>
<td>22.00</td>
<td>120.77</td>
<td>24</td>
</tr>
<tr>
<td>318</td>
<td>596440</td>
<td>Beihai</td>
<td>21.48</td>
<td>109.10</td>
<td>16</td>
</tr>
<tr>
<td>319</td>
<td>596580</td>
<td>ZhanJiang</td>
<td>21.22</td>
<td>110.35</td>
<td>25</td>
</tr>
<tr>
<td>320</td>
<td>596630</td>
<td>Yangjiang</td>
<td>21.90</td>
<td>111.97</td>
<td>22</td>
</tr>
<tr>
<td>321</td>
<td>597580</td>
<td>HaiKou</td>
<td>20.03</td>
<td>110.37</td>
<td>14</td>
</tr>
<tr>
<td>322</td>
<td>598380</td>
<td>Dongfang</td>
<td>19.10</td>
<td>108.65</td>
<td>8</td>
</tr>
<tr>
<td>323</td>
<td>598450</td>
<td>Danxian</td>
<td>19.52</td>
<td>109.58</td>
<td>168</td>
</tr>
<tr>
<td>324</td>
<td>598550</td>
<td>Qionghai</td>
<td>19.23</td>
<td>110.47</td>
<td>24</td>
</tr>
<tr>
<td>325</td>
<td>599480</td>
<td>Sanya</td>
<td>18.23</td>
<td>109.52</td>
<td>7</td>
</tr>
<tr>
<td>326</td>
<td>599810</td>
<td>Xisha Island</td>
<td>16.83</td>
<td>112.33</td>
<td>5</td>
</tr>
<tr>
<td>327</td>
<td>981350</td>
<td>Basco</td>
<td>20.45</td>
<td>121.97</td>
<td>11</td>
</tr>
<tr>
<td>328</td>
<td>984250</td>
<td>Manila</td>
<td>14.58</td>
<td>120.98</td>
<td>16</td>
</tr>
<tr>
<td>329</td>
<td>998001</td>
<td>Kebuduo</td>
<td>48.00</td>
<td>91.60</td>
<td>1,405</td>
</tr>
<tr>
<td>330</td>
<td>998002</td>
<td>Uliastay</td>
<td>47.80</td>
<td>96.90</td>
<td>1,759</td>
</tr>
<tr>
<td>331</td>
<td>998003</td>
<td>Dalandzadgad</td>
<td>43.60</td>
<td>104.40</td>
<td>1,465</td>
</tr>
<tr>
<td>332</td>
<td>998004</td>
<td>Tsetserleg</td>
<td>47.50</td>
<td>101.50</td>
<td>1,691</td>
</tr>
<tr>
<td>333</td>
<td>998005</td>
<td>Altay</td>
<td>46.40</td>
<td>96.30</td>
<td>2,181</td>
</tr>
<tr>
<td>334</td>
<td>998006</td>
<td>Ulaanbaatar</td>
<td>47.90</td>
<td>106.90</td>
<td>1,306</td>
</tr>
<tr>
<td>335</td>
<td>998007</td>
<td>Choybalsan</td>
<td>48.10</td>
<td>114.60</td>
<td>747</td>
</tr>
<tr>
<td>336</td>
<td>998008</td>
<td>Moron</td>
<td>49.60</td>
<td>100.20</td>
<td>1,283</td>
</tr>
<tr>
<td>337</td>
<td>998009</td>
<td>Hatgal</td>
<td>50.40</td>
<td>100.20</td>
<td>1,668</td>
</tr>
<tr>
<td>338</td>
<td>998010</td>
<td>Olgiy</td>
<td>48.90</td>
<td>89.90</td>
<td>1,715</td>
</tr>
<tr>
<td>339</td>
<td>998011</td>
<td>Ulaangom</td>
<td>49.00</td>
<td>92.10</td>
<td>939</td>
</tr>
<tr>
<td>340</td>
<td>998012</td>
<td>Bayanhongor</td>
<td>46.10</td>
<td>100.70</td>
<td>1,859</td>
</tr>
</tbody>
</table>
## Table B.1 (cont'd 8)

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Station</th>
<th>Latitude (°N)</th>
<th>Longitude (°E)</th>
<th>Elevation (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>341</td>
<td>998013</td>
<td>Bulgan</td>
<td>48.80</td>
<td>103.60</td>
<td>1,208</td>
</tr>
<tr>
<td>342</td>
<td>998014</td>
<td>Mandalgovi</td>
<td>44.80</td>
<td>106.30</td>
<td>1,393</td>
</tr>
<tr>
<td>343</td>
<td>998015</td>
<td>Ondorhaan</td>
<td>47.30</td>
<td>110.60</td>
<td>1,033</td>
</tr>
<tr>
<td>344</td>
<td>998016</td>
<td>Matade</td>
<td>47.20</td>
<td>115.60</td>
<td>907</td>
</tr>
<tr>
<td>345</td>
<td>998017</td>
<td>Bayandeleger</td>
<td>45.70</td>
<td>112.40</td>
<td>1,101</td>
</tr>
<tr>
<td>346</td>
<td>999004</td>
<td>Wugong</td>
<td>34.25</td>
<td>108.22</td>
<td>448</td>
</tr>
<tr>
<td>347</td>
<td>999005</td>
<td>Yuzhong</td>
<td>35.87</td>
<td>104.15</td>
<td>1,871</td>
</tr>
<tr>
<td>348</td>
<td>999009</td>
<td>Changyi</td>
<td>44.02</td>
<td>87.30</td>
<td>577</td>
</tr>
<tr>
<td>349</td>
<td>999012</td>
<td>Pingba</td>
<td>26.42</td>
<td>106.27</td>
<td>1,251</td>
</tr>
<tr>
<td>350</td>
<td>999018</td>
<td>Xinchengzi</td>
<td>42.02</td>
<td>123.52</td>
<td>50</td>
</tr>
<tr>
<td>351</td>
<td>999019</td>
<td>Dongling</td>
<td>41.75</td>
<td>123.67</td>
<td>80</td>
</tr>
<tr>
<td>352</td>
<td>999021</td>
<td>Zhaodong</td>
<td>46.07</td>
<td>125.97</td>
<td>147</td>
</tr>
<tr>
<td>353</td>
<td>999022</td>
<td>Dedu</td>
<td>48.50</td>
<td>126.50</td>
<td>272</td>
</tr>
<tr>
<td>354</td>
<td>999023</td>
<td>Hulan</td>
<td>46.00</td>
<td>126.60</td>
<td>127</td>
</tr>
<tr>
<td>355</td>
<td>999024</td>
<td>Acheng</td>
<td>45.52</td>
<td>126.95</td>
<td>174</td>
</tr>
<tr>
<td>356</td>
<td>999025</td>
<td>Jinxian</td>
<td>28.38</td>
<td>116.28</td>
<td>30</td>
</tr>
<tr>
<td>357</td>
<td>999026</td>
<td>Yanshan</td>
<td>28.32</td>
<td>117.55</td>
<td>100</td>
</tr>
<tr>
<td>358</td>
<td>999027</td>
<td>Anshun</td>
<td>26.25</td>
<td>105.92</td>
<td>1,393</td>
</tr>
<tr>
<td>359</td>
<td>999030</td>
<td>Wengyuan</td>
<td>24.37</td>
<td>114.48</td>
<td>215</td>
</tr>
<tr>
<td>360</td>
<td>999032</td>
<td>Jiekou</td>
<td>23.87</td>
<td>113.53</td>
<td>68</td>
</tr>
<tr>
<td>361</td>
<td>999034</td>
<td>Yuhuang</td>
<td>28.22</td>
<td>116.92</td>
<td>50</td>
</tr>
<tr>
<td>362</td>
<td>999036</td>
<td>Yuanmou</td>
<td>25.73</td>
<td>101.87</td>
<td>1,118</td>
</tr>
<tr>
<td>363</td>
<td>999037</td>
<td>Yuanjiang</td>
<td>23.57</td>
<td>102.15</td>
<td>397</td>
</tr>
<tr>
<td>364</td>
<td>999038</td>
<td>Luliang</td>
<td>25.03</td>
<td>103.67</td>
<td>1,840</td>
</tr>
<tr>
<td>365</td>
<td>999039</td>
<td>Luxi</td>
<td>24.53</td>
<td>103.73</td>
<td>1,703</td>
</tr>
<tr>
<td>366</td>
<td>999040</td>
<td>Ziyang</td>
<td>30.12</td>
<td>104.65</td>
<td>357</td>
</tr>
<tr>
<td>367</td>
<td>999045</td>
<td>Changbai</td>
<td>42.40</td>
<td>128.10</td>
<td>750</td>
</tr>
<tr>
<td>368</td>
<td>999048</td>
<td>Fushongdonggang</td>
<td>42.10</td>
<td>127.57</td>
<td>774</td>
</tr>
<tr>
<td>369</td>
<td>999054</td>
<td>Beijian</td>
<td>48.28</td>
<td>126.52</td>
<td>270</td>
</tr>
<tr>
<td>370</td>
<td>999065</td>
<td>Qiongzhong</td>
<td>19.03</td>
<td>109.83</td>
<td>250</td>
</tr>
<tr>
<td>371</td>
<td>999066</td>
<td>Lingshui</td>
<td>18.50</td>
<td>110.03</td>
<td>10</td>
</tr>
<tr>
<td>372</td>
<td>999068</td>
<td>Nada</td>
<td>19.50</td>
<td>109.50</td>
<td>148</td>
</tr>
<tr>
<td>373</td>
<td>999069</td>
<td>Dinghu</td>
<td>23.10</td>
<td>111.93</td>
<td>18</td>
</tr>
<tr>
<td>374</td>
<td>999070</td>
<td>Menglun</td>
<td>21.87</td>
<td>101.07</td>
<td>553</td>
</tr>
</tbody>
</table>
## B.2 Data source and parameter completeness

Table B.2: *List of climatic stations: data source and parameter completeness*

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source(^{\dagger})</th>
<th>(T_{\text{max}})</th>
<th>(T_{\text{min}})</th>
<th>(P)</th>
<th>(P_{\text{freq}})</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>319600</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>366660</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>368700</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>369740</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>384570</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>409480</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>415710</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>417791</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>419174</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>420270</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>421820</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>444540</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>450050</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>470350</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>470550</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>470580</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>470690</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>471080</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>471530</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>471840</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>478270</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>22</td>
<td>479300</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>480080</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>480420</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>483270</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>26</td>
<td>488190</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>6</td>
</tr>
<tr>
<td>27</td>
<td>501360</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>28</td>
<td>503530</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>29</td>
<td>504340</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>505140</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>505270</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>32</td>
<td>505570</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>33</td>
<td>505640</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>506320</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
</tr>
<tr>
<td>35</td>
<td>506580</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>36</td>
<td>507270</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>507450</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>38</td>
<td>507560</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
</tbody>
</table>

\(^{\dagger}\) a: IAP and CDIAC (1991); b: ISSAS and ISRIC (1994); c: WMO (1996)
### Table B.2 (cont’d 1)

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source</th>
<th>$T_{max}$</th>
<th>$T_{min}$</th>
<th>$P$</th>
<th>$Pfreq$</th>
<th>RH</th>
<th>$n$</th>
<th>$u$</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>507740</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>507880</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>41</td>
<td>508440</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>508540</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>43</td>
<td>509150</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>44</td>
<td>509490</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>45</td>
<td>509530</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>46</td>
<td>509630</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>47</td>
<td>509680</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>509780</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>49</td>
<td>510760</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>511330</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>51</td>
<td>511560</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>513340</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>53</td>
<td>513560</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>54</td>
<td>513790</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>55</td>
<td>514310</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>56</td>
<td>514630</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>57</td>
<td>514950</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>58</td>
<td>515730</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>59</td>
<td>516440</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>60</td>
<td>517090</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>61</td>
<td>517160</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>62</td>
<td>517770</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>6</td>
</tr>
<tr>
<td>63</td>
<td>518110</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>64</td>
<td>518280</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>65</td>
<td>518550</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>66</td>
<td>520330</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>67</td>
<td>522670</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>68</td>
<td>523230</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>69</td>
<td>524180</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
</tr>
<tr>
<td>70</td>
<td>524240</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>71</td>
<td>524360</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>72</td>
<td>524950</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>73</td>
<td>525330</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>74</td>
<td>526020</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>75</td>
<td>526330</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>76</td>
<td>526520</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>77</td>
<td>526810</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>78</td>
<td>527130</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>79</td>
<td>527790</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>527870</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>81</td>
<td>528180</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>82</td>
<td>528360</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>6</td>
</tr>
</tbody>
</table>
Table B.2 (cont’d 2)

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source</th>
<th>$T_{max}$</th>
<th>$T_{min}$</th>
<th>$P$</th>
<th>$Pfreq$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>528420</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>84</td>
<td>528560</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>85</td>
<td>528660</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>86</td>
<td>528890</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>87</td>
<td>529570</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>88</td>
<td>529840</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>89</td>
<td>529960</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>90</td>
<td>530680</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>6</td>
</tr>
<tr>
<td>91</td>
<td>531920</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>92</td>
<td>532760</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>93</td>
<td>533260</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>94</td>
<td>533520</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>533910</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>96</td>
<td>534630</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>97</td>
<td>534870</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>98</td>
<td>535020</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>99</td>
<td>535290</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>535880</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>101</td>
<td>535930</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>536020</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>103</td>
<td>536140</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>104</td>
<td>536460</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>105</td>
<td>536640</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>106</td>
<td>536730</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>107</td>
<td>536980</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>108</td>
<td>537050</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>109</td>
<td>537230</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>110</td>
<td>537720</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>111</td>
<td>537980</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>112</td>
<td>538450</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>113</td>
<td>538630</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>114</td>
<td>538980</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>4</td>
</tr>
<tr>
<td>115</td>
<td>539150</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>116</td>
<td>539230</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>117</td>
<td>539590</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>118</td>
<td>540260</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>119</td>
<td>540270</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>120</td>
<td>540940</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>121</td>
<td>540960</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>122</td>
<td>541020</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>123</td>
<td>541150</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>124</td>
<td>541350</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>125</td>
<td>541570</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>126</td>
<td>541610</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
</tbody>
</table>
## B.2 Data source and parameter completeness

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source†</th>
<th>$T_{max}$</th>
<th>$T_{min}$</th>
<th>$P$</th>
<th>$Pfreq$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>127</td>
<td>541860</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>128</td>
<td>542080</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>129</td>
<td>542180</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>130</td>
<td>542370</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>131</td>
<td>542920</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>6</td>
</tr>
<tr>
<td>132</td>
<td>543110</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>133</td>
<td>543240</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>134</td>
<td>543370</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>135</td>
<td>543420</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>136</td>
<td>543630</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>137</td>
<td>543740</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>138</td>
<td>544010</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>139</td>
<td>544230</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>140</td>
<td>544550</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>141</td>
<td>544710</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>142</td>
<td>544760</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>143</td>
<td>544930</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>144</td>
<td>544970</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>145</td>
<td>545110</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>146</td>
<td>545270</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>147</td>
<td>545340</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>148</td>
<td>546020</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>6</td>
</tr>
<tr>
<td>149</td>
<td>546160</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>150</td>
<td>546620</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>151</td>
<td>547140</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>152</td>
<td>547250</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>153</td>
<td>547510</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>154</td>
<td>547650</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>6</td>
</tr>
<tr>
<td>155</td>
<td>547760</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>156</td>
<td>548230</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>157</td>
<td>548430</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>158</td>
<td>548520</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>159</td>
<td>548570</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>160</td>
<td>549060</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>161</td>
<td>549160</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>162</td>
<td>549360</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>163</td>
<td>549380</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>164</td>
<td>552280</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>165</td>
<td>552990</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>166</td>
<td>555780</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>167</td>
<td>555910</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>7</td>
</tr>
<tr>
<td>168</td>
<td>556960</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>169</td>
<td>557730</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>170</td>
<td>560210</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
</tbody>
</table>
### B.2 Data source and parameter completeness

**Table B.2 (cont’d)**

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source†</th>
<th>$T_{\text{max}}$</th>
<th>$T_{\text{min}}$</th>
<th>$P$</th>
<th>$P_{freq}$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>171</td>
<td>560290</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>172</td>
<td>560330</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>173</td>
<td>560800</td>
<td>c</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>2</td>
</tr>
<tr>
<td>174</td>
<td>560930</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>175</td>
<td>560960</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>176</td>
<td>561370</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>6</td>
</tr>
<tr>
<td>177</td>
<td>561460</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>6</td>
</tr>
<tr>
<td>178</td>
<td>561780</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>179</td>
<td>561820</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>180</td>
<td>561930</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>181</td>
<td>561960</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>182</td>
<td>562570</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>183</td>
<td>562870</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>184</td>
<td>562940</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>185</td>
<td>563120</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>186</td>
<td>563740</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>187</td>
<td>563860</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>188</td>
<td>564440</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>189</td>
<td>564620</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>190</td>
<td>564850</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>191</td>
<td>564920</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>5</td>
</tr>
<tr>
<td>192</td>
<td>565710</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>193</td>
<td>566510</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>5</td>
</tr>
<tr>
<td>194</td>
<td>566710</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>195</td>
<td>566910</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>196</td>
<td>567390</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>197</td>
<td>567510</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>198</td>
<td>567780</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>199</td>
<td>567930</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>569510</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>201</td>
<td>569590</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>202</td>
<td>569640</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>203</td>
<td>569850</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>204</td>
<td>569890</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>205</td>
<td>570060</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>6</td>
</tr>
<tr>
<td>206</td>
<td>570360</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>207</td>
<td>570670</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>208</td>
<td>570730</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>209</td>
<td>570830</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>210</td>
<td>571270</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>7</td>
</tr>
<tr>
<td>211</td>
<td>571780</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
<tr>
<td>212</td>
<td>571930</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>213</td>
<td>572370</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</td>
</tr>
<tr>
<td>214</td>
<td>572450</td>
<td>a</td>
<td>—</td>
<td>—</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>—</td>
<td>3</td>
</tr>
</tbody>
</table>
## B.2 Data source and parameter completeness

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source</th>
<th>$T_{\text{max}}$</th>
<th>$T_{\text{min}}$</th>
<th>$P$</th>
<th>$P_{\text{freq}}$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>215</td>
<td>572530</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>216</td>
<td>572590</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>217</td>
<td>572650</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>218</td>
<td>572900</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>219</td>
<td>572970</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>220</td>
<td>573130</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>221</td>
<td>573780</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>222</td>
<td>573990</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>223</td>
<td>574050</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>224</td>
<td>574110</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>225</td>
<td>574470</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>226</td>
<td>574610</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>227</td>
<td>574760</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>228</td>
<td>574940</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>229</td>
<td>575040</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>230</td>
<td>575160</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>231</td>
<td>575370</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>232</td>
<td>575840</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>233</td>
<td>576060</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>234</td>
<td>576330</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>235</td>
<td>576550</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>236</td>
<td>576620</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>237</td>
<td>576790</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>238</td>
<td>577130</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>239</td>
<td>577220</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>240</td>
<td>577310</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>241</td>
<td>577450</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>242</td>
<td>577660</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>243</td>
<td>577930</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>244</td>
<td>577990</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>245</td>
<td>578160</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>246</td>
<td>578530</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>247</td>
<td>578660</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>248</td>
<td>578720</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>249</td>
<td>578960</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>250</td>
<td>579020</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>251</td>
<td>579160</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>252</td>
<td>579570</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>253</td>
<td>579720</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>254</td>
<td>579930</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>255</td>
<td>580270</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>256</td>
<td>580400</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>257</td>
<td>581020</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>258</td>
<td>581440</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
### Table B.2 (cont'd 6)

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source</th>
<th>$T_{max}$</th>
<th>$T_{min}$</th>
<th>$P$</th>
<th>$Pfreq$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>259</td>
<td>581500</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>260</td>
<td>582030</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>261</td>
<td>582080</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>262</td>
<td>582210</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>263</td>
<td>582380</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>264</td>
<td>582510</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>265</td>
<td>582590</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>266</td>
<td>583140</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>267</td>
<td>583210</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>5</td>
</tr>
<tr>
<td>268</td>
<td>583340</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>269</td>
<td>583620</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>270</td>
<td>583670</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>271</td>
<td>584070</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>272</td>
<td>584240</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>273</td>
<td>584570</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>274</td>
<td>584770</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>275</td>
<td>585020</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>276</td>
<td>585310</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>277</td>
<td>585560</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>278</td>
<td>586060</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>279</td>
<td>586330</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>280</td>
<td>586460</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>281</td>
<td>586590</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
</tr>
<tr>
<td>282</td>
<td>586650</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>283</td>
<td>587310</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>284</td>
<td>587340</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>285</td>
<td>587540</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>286</td>
<td>588130</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>287</td>
<td>588340</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>288</td>
<td>588470</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>289</td>
<td>588530</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>2</td>
</tr>
<tr>
<td>290</td>
<td>589210</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>291</td>
<td>589310</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>292</td>
<td>589650</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>293</td>
<td>589740</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>294</td>
<td>590070</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>295</td>
<td>590460</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
</tr>
<tr>
<td>296</td>
<td>590720</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>297</td>
<td>590820</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>298</td>
<td>591170</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>3</td>
</tr>
<tr>
<td>299</td>
<td>591260</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>300</td>
<td>591340</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
</tr>
<tr>
<td>301</td>
<td>591580</td>
<td>c</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>302</td>
<td>592110</td>
<td>a</td>
<td>−</td>
<td>−</td>
<td>×</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>1</td>
</tr>
</tbody>
</table>
### B.2 Data source and parameter completeness

Table B.2 (cont’d 7)

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source†</th>
<th>$T_{Max}$</th>
<th>$T_{Min}$</th>
<th>$P$</th>
<th>$P_{freq}$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>303</td>
<td>592540</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>304</td>
<td>592650</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>305</td>
<td>592870</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>306</td>
<td>592930</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>307</td>
<td>593160</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>308</td>
<td>593450</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>309</td>
<td>593580</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>593620</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>594170</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>594310</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>313</td>
<td>594530</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>594560</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>594930</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>316</td>
<td>595010</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>317</td>
<td>595590</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>318</td>
<td>596440</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>319</td>
<td>596580</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>596630</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>321</td>
<td>597580</td>
<td>a</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>322</td>
<td>598380</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>323</td>
<td>598450</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>324</td>
<td>598550</td>
<td>a</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>325</td>
<td>599480</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>326</td>
<td>599810</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>327</td>
<td>981350</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>328</td>
<td>984250</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>329</td>
<td>998001</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>330</td>
<td>998002</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>331</td>
<td>998003</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>332</td>
<td>998004</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>333</td>
<td>998005</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>334</td>
<td>998006</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>335</td>
<td>998007</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>336</td>
<td>998008</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>337</td>
<td>998009</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>338</td>
<td>998010</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>339</td>
<td>998011</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>340</td>
<td>998012</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>998013</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>342</td>
<td>998014</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>343</td>
<td>998015</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>344</td>
<td>998016</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>998017</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>346</td>
<td>999004</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
### B.2 Data source and parameter completeness

<table>
<thead>
<tr>
<th>Nr</th>
<th>WMO</th>
<th>Source</th>
<th>$T_{\text{max}}$</th>
<th>$T_{\text{min}}$</th>
<th>$P$</th>
<th>$P_{\text{freq}}$</th>
<th>RH</th>
<th>n</th>
<th>u</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>347</td>
<td>999005</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>348</td>
<td>999009</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>349</td>
<td>999012</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>999018</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>351</td>
<td>999019</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>352</td>
<td>999021</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>999022</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>354</td>
<td>999023</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>999024</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>999025</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>999026</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>358</td>
<td>999027</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>359</td>
<td>999030</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>360</td>
<td>999032</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>999034</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>999036</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>999037</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>999038</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>999039</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>366</td>
<td>999040</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>367</td>
<td>999045</td>
<td>b</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>999048</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>999054</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>370</td>
<td>999065</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>371</td>
<td>999066</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>372</td>
<td>999068</td>
<td>c</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>373</td>
<td>999069</td>
<td>c</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>374</td>
<td>999070</td>
<td>b</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table B.2 (cont’d 8)*
Appendix C

FAO Soil Units and Symbols

Alphabetical order by symbol of soil units (FAO-UNESCO, 1974).

A : ACRISOLS
Ao : Orthic Acrisols
Af : Ferric Acrisols
Ah : Humic Acrisols
Ap : Plinthic Acrisols
Ag : Gleyic Acrisols

B : CAMBISOLS
Be : Eutric Cambisols
Bd : Dystric Cambisols
Bh : Humic Cambisols
Bg : Gleyic Cambisols
Bx : Gelic Cambisols
Bk : Calcic Cambisols
Bc : Chromic Cambisols
Bv : Vertic Cambisols
Bf : Ferralic Cambisols

C : CHERNOZEMS
Ch : Haplic Chernozems
Ck : Calcic Chernozems
Cl : Luvic Chernozems
Cg : Glossic Chernozems

D : PODZOLUVISOLS
De : Eutric Podzoluvisols
Dd : Dystric Podzoluvisols
Dg : Gleyic Podzoluvisols

E : RENDZINAS

F : FERRALSOLS
Fo : Orthic Ferralsols
Fx : Xanthic Ferralsols
Fr : Rhodic Ferralsols
Fh : Humic Ferralsols
Fa : Acric Ferralsols
Fp : Plinthic Ferralsols

G : GLEYSOOLS
Ge : Eutric Gleysols
Gc : Calcaric Gleysols
Gd : Dystric Gleysols
Gm : Mollic Gleysols
Gh : Humic Gleysols
Gp : Plinthic Gleysols
Gx : Gelic Gleysols

H : PHAEOZEMS
Hh : Haplic Phaeozems
Hc : Calcic Phaeozems
Hl : Luvic Phaeozems
Hg : Gleyic Phaeozems

I : LITHOSOLS

J : FLUVISOLS
Je : Eutric Fluvisols
Jc : Calcaric Fluvisols
Jd : Dystric Fluvisols
Jt : Thionic Fluvisols
FAO Soil Units and Symbols

**K**: KASTANOZEMS
- Kh: Haplic Kastanozems
- Kk: Calcic Kastanozems
- Kl: Luvic Kastanozems

**L**: LUVISOLS
- Lo: Orthic Luvisols
- Le: Chromic Luvisols
- Lk: Calcic Luvisols
- Lv: Vertic Luvisols
- Lf: Ferric Luvisols
- La: Albic Luvisols
- Lp: Plinthic Luvisols
- Lg: Gleyic Luvisols

**M**: GREYZEMS
- Mo: Orthic Greyzems
- Mg: Gleyic Greyzems

**N**: NITOSOLS
- Ne: Eutric Nitosols
- Nd: Dystric Nitosols
- Nh: Humic Nitosols

**O**: HISTOSOLS
- Oe: Eutric Histosols
- Od: Dystric Histosols
- Ox: Gelic Histosols

**P**: PODZOLS
- Po: Orthic Podzols
- Pl: Leptic Podzols
- Pf: Ferric Podzols
- Ph: Humic Podzols
- Pp: Placic Podzols
- Pg: Gleyic Podzols

**Q**: ARENOSOLS
- Qc: Cambic Arenosols
- Ql: Luvic Arenosols
- Qf: Ferralic Arenosols
- Qa: Albic Arenosols

**R**: REGOSOLS
- Re: Eutric Gleysols
- Rc: Calcaric Regosols
- Rd: Dystric Regosols

**S**: SOLONETZ
- So: Orthic Solonetz
- Sm: Mollic Solonetz
- Sg: Gleyic Solonetz

**T**: ANDOSOLS
- To: Ochric Andosols
- Tm: Mollic Andosols
- Th: Humic Andosols
- Tv: Vitric Andosols

**U**: RANKERS
- Vp: Pellic Vertisols
- Vc: Chromic Vertisols

**W**: PLANOSOLS
- We: Eutric Planosols
- Wd: Dystric Planosols
- Wh: Humic Planosols
- Ws: Solodic Planosols
- Wx: Gelic Planosols

**X**: XEROSOLS
- Xh: Haplic Xerosols
- Xk: Calcic Xerosols
- Xy: Gypsic Xerosols
- XL: Luvic Xerosols

**Y**: YERMOSOLS
- Yh: Haplic Yermosols
- Yk: Calcic Yermosols
- Yy: Gypsic Yermosols
- Yl: Luvic Yermosols
- Yt: Takyric Yermosols

**Z**: SOLONCHAKS
- Zo: Orthic Solonchaks
- Zm: Mollic Solonchaks
- Zt: Takyric Solonchaks
- Zg: Gleyic Solonchaks
Appendix D

Records of Observed Yields and Factor Inputs

A compilation of observed yields of food crops (rice, wheat, maize, etc.) and crop groups (grain and food) as well as factor inputs during 1978-2005 is given below in Tables D.1 and D.2. Data earlier than 1999 (inclusive) are compiled from NBSC (1999), while for the rest of the years, from NBSC (2006).

Table D.1: Records of observed yields of food crops in kg ha$^{-1}$, 1978-2005

<table>
<thead>
<tr>
<th>Year</th>
<th>Rice</th>
<th>Wheat</th>
<th>Maize</th>
<th>Sorghum</th>
<th>Millet</th>
<th>Grain</th>
<th>Soybean</th>
<th>Potato</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>1978</td>
<td>3,978</td>
<td>1,845</td>
<td>2,803</td>
<td>2,331</td>
<td>1,537</td>
<td>2,863</td>
<td>1,059</td>
<td>2,527</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>4,130</td>
<td>1,889</td>
<td>3,076</td>
<td>2,516</td>
<td>1,406</td>
<td>2,999</td>
<td>1,099</td>
<td>2,830</td>
<td>2,734</td>
</tr>
<tr>
<td>1985</td>
<td>5,256</td>
<td>2,937</td>
<td>3,607</td>
<td>2,896</td>
<td>1,801</td>
<td>3,915</td>
<td>1,362</td>
<td>3,037</td>
<td>3,483</td>
</tr>
<tr>
<td>1986</td>
<td>5,338</td>
<td>3,040</td>
<td>3,705</td>
<td>2,870</td>
<td>1,527</td>
<td>3,995</td>
<td>1,400</td>
<td>2,917</td>
<td>3,529</td>
</tr>
<tr>
<td>1987</td>
<td>5,413</td>
<td>2,983</td>
<td>3,920</td>
<td>2,911</td>
<td>1,621</td>
<td>4,073</td>
<td>1,482</td>
<td>3,181</td>
<td>3,623</td>
</tr>
<tr>
<td>1988</td>
<td>5,287</td>
<td>2,968</td>
<td>3,928</td>
<td>3,136</td>
<td>1,756</td>
<td>4,037</td>
<td>1,434</td>
<td>2,973</td>
<td>3,557</td>
</tr>
<tr>
<td>1989</td>
<td>5,509</td>
<td>3,043</td>
<td>3,878</td>
<td>2,722</td>
<td>1,566</td>
<td>4,119</td>
<td>1,269</td>
<td>3,001</td>
<td>3,632</td>
</tr>
<tr>
<td>1990</td>
<td>4,930</td>
<td>3,194</td>
<td>4,524</td>
<td>3,673</td>
<td>2,008</td>
<td>4,432</td>
<td>1,455</td>
<td>3,008</td>
<td>3,933</td>
</tr>
<tr>
<td>1991</td>
<td>5,340</td>
<td>3,101</td>
<td>4,578</td>
<td>1,911</td>
<td>1,911</td>
<td>4,206</td>
<td>1,361</td>
<td>2,992</td>
<td>3,876</td>
</tr>
<tr>
<td>1992</td>
<td>5,410</td>
<td>3,331</td>
<td>4,533</td>
<td>2,081</td>
<td>2,081</td>
<td>4,342</td>
<td>1,394</td>
<td>3,140</td>
<td>4,004</td>
</tr>
<tr>
<td>1993</td>
<td>5,157</td>
<td>3,519</td>
<td>4,963</td>
<td>2,434</td>
<td>2,434</td>
<td>4,557</td>
<td>1,576</td>
<td>3,450</td>
<td>4,131</td>
</tr>
<tr>
<td>1994</td>
<td>5,509</td>
<td>3,426</td>
<td>4,693</td>
<td>2,680</td>
<td>2,680</td>
<td>4,500</td>
<td>1,645</td>
<td>3,264</td>
<td>4,101</td>
</tr>
<tr>
<td>1995</td>
<td>5,381</td>
<td>3,542</td>
<td>4,917</td>
<td>2,409</td>
<td>2,409</td>
<td>4,659</td>
<td>1,591</td>
<td>3,428</td>
<td>4,240</td>
</tr>
<tr>
<td>1996</td>
<td>5,668</td>
<td>3,734</td>
<td>5,203</td>
<td>2,709</td>
<td>2,709</td>
<td>4,894</td>
<td>1,698</td>
<td>3,609</td>
<td>4,483</td>
</tr>
<tr>
<td>1997</td>
<td>5,832</td>
<td>4,102</td>
<td>4,387</td>
<td>2,381</td>
<td>2,381</td>
<td>4,823</td>
<td>1,680</td>
<td>3,262</td>
<td>4,377</td>
</tr>
<tr>
<td>1998</td>
<td>5,773</td>
<td>3,685</td>
<td>5,268</td>
<td>2,522</td>
<td>2,522</td>
<td>4,953</td>
<td>1,714</td>
<td>3,604</td>
<td>4,502</td>
</tr>
<tr>
<td>1999</td>
<td>5,767</td>
<td>3,947</td>
<td>4,945</td>
<td>2,258</td>
<td>2,258</td>
<td>4,945</td>
<td>1,693</td>
<td>3,516</td>
<td>4,493</td>
</tr>
<tr>
<td>2000</td>
<td>5,857</td>
<td>3,738</td>
<td>4,598</td>
<td>2,088</td>
<td>2,088</td>
<td>4,753</td>
<td>1,588</td>
<td>3,497</td>
<td>4,261</td>
</tr>
<tr>
<td>2001</td>
<td>5,955</td>
<td>3,806</td>
<td>4,698</td>
<td>2,262</td>
<td>2,262</td>
<td>4,800</td>
<td>1,547</td>
<td>3,488</td>
<td>4,267</td>
</tr>
<tr>
<td>2002</td>
<td>6,264</td>
<td>3,777</td>
<td>4,925</td>
<td>2,509</td>
<td>2,509</td>
<td>4,885</td>
<td>1,787</td>
<td>3,710</td>
<td>4,399</td>
</tr>
<tr>
<td>2003</td>
<td>6,259</td>
<td>3,932</td>
<td>4,813</td>
<td>2,670</td>
<td>2,670</td>
<td>4,873</td>
<td>1,649</td>
<td>3,621</td>
<td>4,332</td>
</tr>
<tr>
<td>2004</td>
<td>6,794</td>
<td>4,252</td>
<td>5,120</td>
<td>2,627</td>
<td>2,627</td>
<td>5,187</td>
<td>1,744</td>
<td>3,762</td>
<td>4,620</td>
</tr>
<tr>
<td>2005</td>
<td>7,032</td>
<td>4,541</td>
<td>7,474</td>
<td>3,865</td>
<td>4,150</td>
<td>6,349</td>
<td>3,045</td>
<td>2,940</td>
<td>5,714</td>
</tr>
</tbody>
</table>
For convenience, sown area of agricultural crops and area of farmland are also included in Table D.2.

Table D.2: Agricultural factor inputs, 1978-2005

<table>
<thead>
<tr>
<th>Year</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
<th>(9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1978</td>
<td>150,104</td>
<td>140.8</td>
<td>31.9</td>
<td>8.8</td>
<td>6.5</td>
<td>1.9</td>
<td>0.2</td>
<td>0.2</td>
<td>25.3</td>
</tr>
<tr>
<td>1980</td>
<td>146,380</td>
<td>142.5</td>
<td>31.5</td>
<td>12.7</td>
<td>9.3</td>
<td>2.7</td>
<td>0.3</td>
<td>0.3</td>
<td>32.1</td>
</tr>
<tr>
<td>1985</td>
<td>143,626</td>
<td>141.2</td>
<td>31.2</td>
<td>17.8</td>
<td>12.0</td>
<td>3.1</td>
<td>0.8</td>
<td>1.8</td>
<td>50.9</td>
</tr>
<tr>
<td>1986</td>
<td>144,204</td>
<td>141.0</td>
<td>31.4</td>
<td>19.3</td>
<td>13.1</td>
<td>3.6</td>
<td>0.8</td>
<td>1.8</td>
<td>58.7</td>
</tr>
<tr>
<td>1987</td>
<td>144,957</td>
<td>140.7</td>
<td>31.6</td>
<td>20.0</td>
<td>13.3</td>
<td>3.7</td>
<td>0.9</td>
<td>2.1</td>
<td>65.9</td>
</tr>
<tr>
<td>1988</td>
<td>144,869</td>
<td>140.4</td>
<td>31.6</td>
<td>21.4</td>
<td>14.2</td>
<td>3.8</td>
<td>1.0</td>
<td>2.4</td>
<td>71.2</td>
</tr>
<tr>
<td>1989</td>
<td>146,554</td>
<td>140.2</td>
<td>31.1</td>
<td>23.6</td>
<td>15.4</td>
<td>4.2</td>
<td>1.2</td>
<td>2.8</td>
<td>79.1</td>
</tr>
<tr>
<td>1990</td>
<td>148,363</td>
<td>139.9</td>
<td>33.9</td>
<td>25.9</td>
<td>16.4</td>
<td>4.6</td>
<td>1.5</td>
<td>3.4</td>
<td>84.5</td>
</tr>
<tr>
<td>1991</td>
<td>149,586</td>
<td>139.7</td>
<td>34.2</td>
<td>28.1</td>
<td>17.3</td>
<td>5.0</td>
<td>1.7</td>
<td>4.1</td>
<td>96.3</td>
</tr>
<tr>
<td>1992</td>
<td>149,007</td>
<td>139.4</td>
<td>34.9</td>
<td>29.3</td>
<td>17.6</td>
<td>5.2</td>
<td>2.0</td>
<td>4.6</td>
<td>110.7</td>
</tr>
<tr>
<td>1993</td>
<td>147,741</td>
<td>139.1</td>
<td>35.0</td>
<td>31.5</td>
<td>18.4</td>
<td>5.8</td>
<td>2.1</td>
<td>5.3</td>
<td>124.5</td>
</tr>
<tr>
<td>1994</td>
<td>148,241</td>
<td>138.9</td>
<td>35.1</td>
<td>33.2</td>
<td>18.8</td>
<td>6.0</td>
<td>2.3</td>
<td>6.0</td>
<td>147.4</td>
</tr>
<tr>
<td>1995</td>
<td>149,879</td>
<td>138.6</td>
<td>35.6</td>
<td>35.9</td>
<td>20.2</td>
<td>6.3</td>
<td>2.7</td>
<td>6.7</td>
<td>165.6</td>
</tr>
<tr>
<td>1996</td>
<td>152,381</td>
<td>138.4</td>
<td>36.4</td>
<td>38.3</td>
<td>21.5</td>
<td>6.6</td>
<td>2.9</td>
<td>7.3</td>
<td>181.3</td>
</tr>
<tr>
<td>1997</td>
<td>153,969</td>
<td>138.1</td>
<td>37.1</td>
<td>39.8</td>
<td>21.7</td>
<td>6.9</td>
<td>3.2</td>
<td>8.0</td>
<td>198.0</td>
</tr>
<tr>
<td>1998</td>
<td>155,706</td>
<td>137.8</td>
<td>37.9</td>
<td>40.8</td>
<td>22.3</td>
<td>6.8</td>
<td>3.5</td>
<td>8.2</td>
<td>204.2</td>
</tr>
<tr>
<td>1999</td>
<td>156,373</td>
<td>137.6</td>
<td>38.6</td>
<td>41.2</td>
<td>21.8</td>
<td>7.0</td>
<td>3.7</td>
<td>8.8</td>
<td>217.3</td>
</tr>
<tr>
<td>2000</td>
<td>156,300</td>
<td>136.1</td>
<td>39.5</td>
<td>41.5</td>
<td>21.6</td>
<td>6.9</td>
<td>3.8</td>
<td>9.2</td>
<td>242.1</td>
</tr>
<tr>
<td>2001</td>
<td>155,708</td>
<td>134.7</td>
<td>40.3</td>
<td>42.5</td>
<td>21.6</td>
<td>7.1</td>
<td>4.0</td>
<td>9.8</td>
<td>261.1</td>
</tr>
<tr>
<td>2002</td>
<td>154,636</td>
<td>133.2</td>
<td>40.8</td>
<td>43.4</td>
<td>21.6</td>
<td>7.1</td>
<td>4.2</td>
<td>10.4</td>
<td>299.3</td>
</tr>
<tr>
<td>2003</td>
<td>152,415</td>
<td>131.8</td>
<td>41.0</td>
<td>44.1</td>
<td>21.5</td>
<td>7.1</td>
<td>4.4</td>
<td>11.1</td>
<td>343.3</td>
</tr>
<tr>
<td>2004</td>
<td>153,553</td>
<td>130.3</td>
<td>41.8</td>
<td>46.4</td>
<td>22.2</td>
<td>7.4</td>
<td>4.7</td>
<td>12.0</td>
<td>393.3</td>
</tr>
<tr>
<td>2005</td>
<td>155,488</td>
<td>128.9</td>
<td>42.7</td>
<td>47.7</td>
<td>22.3</td>
<td>7.4</td>
<td>4.9</td>
<td>13.0</td>
<td>437.6</td>
</tr>
</tbody>
</table>

(1) Sown area of agricultural crops, kha  
(2) Area of farmland, kha  
(3) Effective irrigation rate, %  
(4) Fertilizers, million t  
(5) Nitrate fertilizers, million t  
(6) Phosphorus fertilizers, million t  
(7) Potassium fertilizers, million t  
(8) Compound fertilizers, million t  
(9) Electrical consumption, billion kWh
# Appendix E

## Atlas

### E.0 List of maps

Table E.1: *List of maps in Appendix E*

<table>
<thead>
<tr>
<th>No.</th>
<th>Map</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.1</td>
<td>Topographic map of China</td>
<td>174</td>
</tr>
<tr>
<td>E.2</td>
<td>Land cover map of China in the year 2000</td>
<td>175</td>
</tr>
<tr>
<td>E.3</td>
<td>Map of China’s agricultural cropland in the year 2000</td>
<td>176</td>
</tr>
<tr>
<td>E.4</td>
<td>Completeness and integrity of the climatic dataset</td>
<td>177</td>
</tr>
<tr>
<td>E.5</td>
<td>Interpolated mean maximal air temperature in January</td>
<td>178</td>
</tr>
<tr>
<td>E.6</td>
<td>Monthly mean maximal temperature</td>
<td>179</td>
</tr>
<tr>
<td>E.7</td>
<td>Monthly precipitation</td>
<td>180</td>
</tr>
<tr>
<td>E.8</td>
<td>Annual mean minimal air temperature</td>
<td>181</td>
</tr>
<tr>
<td>E.9</td>
<td>Annual mean frequency of precipitation</td>
<td>182</td>
</tr>
<tr>
<td>E.10</td>
<td>Annual mean relative air humidity</td>
<td>183</td>
</tr>
<tr>
<td>E.11</td>
<td>Annual mean daily sunshine duration</td>
<td>184</td>
</tr>
<tr>
<td>E.12</td>
<td>Annual mean wind velocity</td>
<td>185</td>
</tr>
<tr>
<td>E.13</td>
<td>Aridity index-based climate classification</td>
<td>186</td>
</tr>
<tr>
<td>E.14</td>
<td>FAO soil groupings in China</td>
<td>187</td>
</tr>
<tr>
<td>E.15</td>
<td>FAO soil units in zone 5</td>
<td>188</td>
</tr>
<tr>
<td>E.16</td>
<td>Annual reference evapotranspiration</td>
<td>189</td>
</tr>
<tr>
<td>E.17</td>
<td>Average net biomass production, harvest index and the RPP of food crops</td>
<td>190</td>
</tr>
<tr>
<td>E.18</td>
<td>Average available soil water for and rooting depth of food crops</td>
<td>191</td>
</tr>
<tr>
<td>E.19</td>
<td>Combined rainfall effectiveness for food crops</td>
<td>192</td>
</tr>
<tr>
<td>E.20</td>
<td>$K_c$, $ET_m$ and $ET_a$ of food crops</td>
<td>193</td>
</tr>
<tr>
<td>E.21</td>
<td>$ET$ deficit, $k_y$, $f_W$ and the WPP of food crops</td>
<td>194</td>
</tr>
<tr>
<td>E.22</td>
<td>The most limiting soil characteristics for food crops</td>
<td>195</td>
</tr>
<tr>
<td>E.23</td>
<td>Soil index for food production</td>
<td>196</td>
</tr>
<tr>
<td>E.24</td>
<td>Cropland productivity change under BAU scenario</td>
<td>197</td>
</tr>
<tr>
<td>E.25</td>
<td>Cropland productivity change under double degradation scenario</td>
<td>198</td>
</tr>
</tbody>
</table>
E.1 Topographic map of China

Figure E.1: Topographic map of China
E.2 Land cover map of China in the year 2000

Figure E.2: Land cover map of China in the year 2000
E.3 Map of cropland in China in the year 2000

Figure E.3: Map of China’s agricultural cropland in the year 2000
E.4 Completeness and integrity of the climatic dataset

Figure E.4: Completeness and integrity of the climatic dataset
E.5 Interpolated mean maximal air temperature in January

Interpolated Mean Maximum Air Temperatures in January
Before (a) and After (b) Elevation Adjustment

Grid Resolution
5km x 5km

Figure E.5: Interpolated mean maximal air temperatures in January before (a) and after (b) elevation adjustment
E.6 Monthly mean maximal air temperature

Figure E.6: Monthly mean maximal air temperature class, January-December
Figure E.7: Monthly precipitation class, January-December
Figure E.8: Annual mean minimal air temperature class
E.9 Annual mean frequency of precipitation

Figure E.9: Annual mean frequency of precipitation
E.10  Annual mean relative air humidity

Figure E.10: Annual mean relative air humidity
E.11 Annual mean daily sunshine duration

Figure E.11: Annual mean daily sunshine duration
Annual mean wind velocity

Figure E.12: Annual mean wind velocity
E.13 Aridity index-based climate classification

Figure E.13: Aridity index-based climate classification. upper-right: aridity index; lower-left: climatic classes; lower-right: climatic orders
E.14 FAO soil groupings in China

Figure E.14: FAO soil groupings in China
E.15 FAO soil units in Zone 5

Figure E.15: FAO soil units in Zone 5
Figure E.16: Annual reference evapotranspiration, mm
E.17 Net biomass production, harvest index and the RPP of food crops in 2005

Figure E.17: Net biomass production, harvest index and the RPP of food crops in 2005
E.18  Plant available soil water content and capacity, and average rooting depth of food crops in 2005

Figure E.18: Plant available soil water content and capacity for and average rooting depth of food crops in 2005
E.19 Rainfall effectiveness for food crops in 2005

Figure E.19: Combined rainfall effectiveness for food crops in 2005
E.20 Average $K_c$, $ET_m$ and $ET_a$ during crop cycle of food crops in 2005

Figure E.20: Average $K_c$, $ET_m$ and $ET_a$ during crop cycle of food crops in 2005
Figure E.21: ET deficit (a), yield response factor (b), water stress coefficient (c) and the WPP (d) of food crops in 2005
E.22 The most limiting soil characteristics for food crops in 2005

Figure E.22: The most limiting soil characteristics for food crops in 2005
E.23 Soil index for food production in 2005

Figure E.23: Soil index for food production in 2005
E.24 Cropland productivity change under the business-as-usual scenario by 2030 and 2050

Figure E.24: Cropland productivity change under the business-as-usual scenario by 2030 and 2050
E.25 Cropland productivity change under double-degradation scenario by 2030 and 2050

Figure E.25: Cropland productivity change under the double-degradation scenario by 2030 and 2050
Bibliography


BIBLIOGRAPHY


Index

a.s.l., xiii, 12
AEZ, 7, 29, 92
    map of, 9
AI, xiii, 28
ALES, xiii, 77
AML, xiii, 149
ANOVA, xiii, 101, 103, 107
API, xiii
Arc/Info GRID, 12, 142, 145
    ASCII, 147
    binary, 145
ArcGIS, 21, 23
aridity index, xiii, 28, 67
ASCII, xiii, 12, 145–147
ASSOD, xiii, 116, 117
atlas, 173
autocorrelation, 19, 20
autonomous region, 7
autotrophic respiration, 143
backend, 77
Banpo neolithic village, 5
BAU, xiii, xvii, 128, 130, 132, 134, 141
big endian, 149
biomass, 53
BM, 78, 142
BMC, xiii, xvii, 76, 78, 138
Box-and-Whisker plot, 22
BS, xiii, 100
bulk density, xvi, 33
business-as-usual, xiii, 126, 128–130, 132, 134, 136, 141
byte order
    big endian, 149
    little endian, 149
CAAS, xiii
Cambisols, 30–32
carbon sequestration, 36
CAS, xiii, 14
case study, 21
CCIA, xiii, 143
CD, xiii, 57
CDIAC, xiii, 14
CDS, xiii, 40, 43
CEC, xiii, 68, 100
CGP, xiii, 42, 43, 67
Changjiang, 12
climatic
    classification, 28
    grid, 28
    station, 13
climate change, 143
CO₂ fertilization, 143
coarse fragments, xv, 33
crop
    cycle, 39, 42
    development stage, 39, 40, 43
    growth period, 42, 43
    phenology, 39
    rotation, 43
crop coefficient, 57
crop cycle, 39, 42
crop development stage, xiii, 39, 43
    CD, xiii, 57
    IS, xiv, 57
    LS, xiv, 57
    MS, xiv, 57
crop growth period, xiii, 43
cropland, 13
    inventory, 13
cropping system, 38
daily sunshine duration, xvi, 14
daytime temperature, xvi
DBMS, 77
DEM, xiii, 12, 144, 149
    Conversion to GRID, 144
deterministic interpolation, 17
Digital Soil Map of the World, xiii
downscaling, 87, 92
DSMW, xiii, 29–32
DSSAT, xiii, 77
DTM, xiii
EC, xiii, 68, 100
effective irrigation rate, 87, 88
effective rainfall, 61
EOF, xiii, 148
ESP, xiii, 68, 100
ESRI, xiii, 145, 147
Euclidean distance, xv, 19
Eurasia, 5
evapotranspiration
  actual, xv, 63
  maximum, xv
  reference, xv, 46
evapotranspiration:maximum, 56
expansion file, 30, 31
factor input, 88, 89
FAO, xiii, 13
  Digital Soil Map of the World, xiii, 29, 31
  mapping unit, xiv, 29, 33
  Revised Legend, 28, 30
  soil grouping, 30
  soil unit, 30, 33
farmland, 13, 145
FC, xiii, 59
field capacity, xiii, 59
food consumption, 1, 132
  per capita, 1
food demand, 110
  per capita, 114, 132
food insecurity, 2, 137
food production, 1, 114, 136
  capacities, xvii, xviii, 110, 136
  per capita, 1
  scenarios, 116, 121
food productivity, 126, 127
  simulated versus observed, 87
food security, xvii–xix, 1, 110, 114, 116, 124, 126, 131, 134–137, 139, 140, 143
  index, xviii, xx, 110, 134, 137
  threshold, 3
food security index, xiii
food self-sufficiency, 1
food supply, xvii
  per capita, 131, 132
frontend, 77
FSI, xiii, xviii, xx, 110, 134, 135, 139, 140
GDP, xiii, 9
Geostatistical Analyst, 21, 23
geostatistical interpolation, 19
geostatistics, 137
GHG, xiii, 28, 29
GIS, xiii, xvii, 142
GLASOD, xiii, 116, 120
GLC2000, 13
GLF, xiii, 1, 114, 134
global interpolator, 18
GP, xiii
Great Leap Forward, xiii, 1, 114, 134
green house gas, xiii, 28, 29
Green Revolution, 1, 70
grid, 9, 23, 27, 31, 87, 92, 95, 126, 137, 142
  ASCII format, 147
  binary format, 145
  resolution, 12
gross domestic income, xiii
growing period, xiii, 49
GTOPO30, 12, 144
GUI, xiii
harvest index, 1
Hemudu culture, 5
Homo erectus, 5
HTTP, xiii, 78
Huanghe, 12
IAP, xiii, 14
ICAO, xiii
IDW, xiv, 17, 19
IM, 78
input level, 121
Intel, 149
International Standard Atmosphere, xiv, 21
interpolation
  deterministic, 17
  geostatistical, 17
  spatial, 15
interquantile range, xiv, 22
inverse distance weighting, xiv, 17
IQR, xiv, 22
irrigation rate, 88
IS, xiv, 57
ISA, xiv, 21
ISRIC, xiv, 14, 116
ISSAS, xiv, 14
JRC, xiv
kriging, 19, 22
kriging, ordinary, 15, 21
LAI, xiv, 41, 48
Lambert Azimuthal Equal Area, 12, 13, 30, 146, 148, 150
land cover, 13
land degradation, 3
land evaluation, xiv, 3, 29, 46, 138
quantitative methodologies, 46
WLES, 76, 138
LCCS, xiv, 13
LE, xiv, 76–79, 83
leaf area index, xiv, xv, 41
leave-one-out cross-validation, xiv, 25
LGP, xiv, 53
little endian, 149
LLE, xiv, xvii, 76, 78, 138
local interpolator, 18
Loess Plateau, 5, 12
LOOCV, xiv, 25
LPP, xiv, 46, 67
LS, xiv, 57
management index, xv, xviii, xix, 67, 70, 74, 88, 90, 92, 136, 137, 142
map of
agro-ecological zones, 9
annual mean daily sunshine duration, 184
annual mean frequency of precipitation, 182
annual mean minimal temperature, 181
annual mean relative humidity, 183
annual mean wind velocity, 185
annual reference evapotranspiration, 189
aridity index-climate, 186
climatic stations, 177
cropland, 176
Cropland productivity under 2xdegradation scenario, 198
Cropland productivity under BAU scenario, 197
elevation-adjusted temperature, 178
ET deficit, $k_y$, $f_W$ and WPP of food crops, 194
FAO soil groupings in China, 187
FAO soil units in zone 5, 188
$K_c$, $ET_m$ and $ET_a$ of food crops in 2005, 193
land cover, 175
monthly maximum temperature, 179
monthly precipitation, 180
Most limiting soil characteristics for food crops, 195
net biomass production, HI and RPP of food crops in 2005, 190
PAW, rooting depth and $S_a D$ in 2005, 191
provinces, 8
Rainfall effectiveness for food crops in 2005, 192
socio-economic belts, 11
Soil index for food production in 2005, 196
temperature in January, 178
topography, 174
mapping unit, xiv, 33
maximal air temperature, xvi, 13
MCI, xiv, 124, 125
MDG, xiv, xvii, 139
metropolitan, 7
Millennium Development Goal, xiv
minimal air temperature, xvi, 13
MOA, xiv
MOC, xiv
Motorola, 149
Mount Qomolangma, 12
MS, xiv, 57
MU, xiv, 29, 30, 123
multi-cropping index, xiv, 124, 125, 129
NARPC, xiv
NBSC, xiv
nitrogen deposition, 143
NOARP, xiv, 124
North China Plain, 12
Northeast China Plain, 12
NSSO, xiv, 34
ordinary kriging, 19, 21, 22
outlier, 22
INDEX

Pacific, 5
PAW, xiv, 59, 60
permanent wilting point, xiv, 59
photoperiod insensitivity, 1
Plains of the Middle-and-Lower-Reachs of the Yangtze River, 12
plant available water, xiv, 59, 60
population, 110
  composition, 112
  control, 112
  size, 1, 110, 111, 114, 115, 131
prediction location, xvi
production function, 70, 72–74, 142
  Cobb-Douglas, xix, 70, 72, 137
production potential
  land, 67
  LPP, xiv, 67
  radiation-thermal, 53
  RPP, xiv, 53
  water-limited, 56
  WPP, xiv, 56
province, 7
  map of, 8
PWP, xiv, 59
rainfall, xvi, 13
rainfall frequency, xvi, 13
rasterization, 30
regional disparity, 7
relative air humidity, xvi, 14
remote sensing, 145
resolution, 15
respiration, 54
rooting depth, 59
RPP, xiv, 46, 53
SAR, xiv, 7
scenarios, xvii, 4, 90, 116, 121–123, 125, 126, 129, 131, 134, 136
cropland, 123
cropping intensity, 124
soil degradation, 124
urbanization, 121
SEB, xiv, 7, 9, 89
  map of, 11
semivariance, xv, 19, 20, 24
semivariogram, 19–21, 24, 25, 27
cloud, 20, 24
model, 20, 24, 27, 28
  spherical, 24
Sichuan Basin, 12
SNUM, 30, 33
SOC, xiv, 33, 36, 68, 100
  content, 33
  density, 33, 34
  storage, 36
SOCD, xiv, 33, 34, 36
SOCS, xiv, 36
soil degradation, xvii–xix, 4, 110, 115–122, 124–131, 134, 136, 137, 139–141
effect, 126
soil grouping, 30
soil index, xvi, xviii, 67, 68, 70, 136
soil organic carbon, xiv
soil organic carbon density, xiv, 33
soil organic carbon storage, xiv
soil quality, xvii, 3
soil unit, 30, 33
SPAC, xiv
spatial resolution, 144
SPOT-VEGETATION, 13
SRIC, 92
topography, 12
UNDP, xiv
UNEP, xiv
UNESCO, xiv
UNFPA, xiv, 110, 111, 114, 121
upscale, 13, 150
upsampling, 87, 93
urbanization, 2, 113, 121
  pattern, 113, 114
  process, 113
  rate, 121
USDA, xiv
USGS, xiv, 12, 144
V2R, xiv, 30, 33
vector to raster conversion, xiv, 33
vector-to-raster conversion, 30
virtual soil profile, 31
WBS, xiv, xvii, 76, 78, 138
Web-based Land Evaluation System, xiv
INDEX

WGS84, 13, 144
wind velocity, xvi, 14
WISE, xiv, 32, 34, 92
WLES, xiv, xvii, xxi, 3, 76–78, 80, 83, 138, 142, 143
WMO, xiv, 14, 15, 151, 160
WOFOST, xiv, 77
world inventory of soil emission, xiv
WPP, xiv, 46, 56
WRI, xiv

Yangtze River, 12
Yellow River, 12
Yellow Sea, 13
yield
   irrigated, 87
   observed, 87, 96, 97, 99
   rainfed, 87
   simulated, 87, 95–97, 99
   simulated versus observed, 87
   simulation, 92
Simulating long-term food producing capacities in China using a Web-based land evaluation system

Langetermijnsimulaties van voedsel-productiecapaciteit in China door middel van webgebaseerde landevaluatie

Volume II. WLES-Doc

Liming Ye

Thesis submitted in fulfillment of the requirements for the degree of Doctor (Ph.D.) in Geology

Proefschrift voorgedragen tot het bekomen van de graad van Doctor in de Geologie

Op gezag van

Rector: Prof. Dr. P. Van Cauwenberge
Decaan: Prof. Dr. H. Dejonghe
Promotor: Prof. Dr. E. Van Ranst
## Contents

How to Use This Document .......................... 3

Namespace List ........................................ 5

1 Namespace UGent.WLES ............................ 7

1.1 Overview ......................................... 7

1.1.1 Classes ........................................ 7

1.1.2 Structures ..................................... 7

1.1.3 Enumerations ................................... 7

1.2 Global Class ...................................... 8

1.2.1 Summary ........................................ 8

1.2.2 Declaration ..................................... 8

1.2.3 Remarks ........................................ 8

1.2.4 Members ........................................ 8

1.3 Logistics Class .................................... 8

1.3.1 Summary ........................................ 8

1.3.2 Declaration ..................................... 8

1.3.3 Remarks ........................................ 8

1.3.4 Public Static Fields ......................... 9

1.3.5 Public Static Methods ....................... 12

1.4 WlesDatabaseException Class .................... 30

1.4.1 Summary ........................................ 30

1.4.2 Declaration ..................................... 30

1.4.3 Public Instance Constructors ............... 30

1.5 RomanDate Structure ............................. 30

1.5.1 Summary ........................................ 30

1.5.2 Declaration ..................................... 31

1.5.3 Remarks ........................................ 31

1.5.4 Examples ....................................... 31

1.5.5 Public Static Fields ......................... 32

1.5.6 Public Static Methods ....................... 32

1.5.7 Public Static Operators ..................... 33

1.5.8 Public Instance Constructors ............... 34

1.5.9 Public Instance Properties ................. 35

1.5.10 Public Instance Methods .................... 35

1.6 Utilities Structure ............................... 36

1.6.1 Summary ........................................ 36

1.6.2 Remarks ........................................ 36

1.6.3 Public Static Fields ......................... 37

1.6.4 Public Static Methods ....................... 38

1.7 WlesException Class ............................. 48

1.7.1 Summary ........................................ 48

1.7.2 Declaration ..................................... 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.7.3</td>
<td>Public Instance Constructors</td>
<td>48</td>
</tr>
<tr>
<td>1.8</td>
<td>Logistics.InternalLink Enumeration</td>
<td>48</td>
</tr>
<tr>
<td>1.8.1</td>
<td>Summary</td>
<td>48</td>
</tr>
<tr>
<td>1.8.2</td>
<td>Declaration</td>
<td>48</td>
</tr>
<tr>
<td>1.8.3</td>
<td>Members</td>
<td>49</td>
</tr>
<tr>
<td>1.9</td>
<td>RomanDate.CalendarType Enumeration</td>
<td>49</td>
</tr>
<tr>
<td>1.9.1</td>
<td>Summary</td>
<td>49</td>
</tr>
<tr>
<td>1.9.2</td>
<td>Declaration</td>
<td>49</td>
</tr>
<tr>
<td>1.9.3</td>
<td>Remarks</td>
<td>50</td>
</tr>
<tr>
<td>1.9.4</td>
<td>Members</td>
<td>50</td>
</tr>
<tr>
<td>1.10</td>
<td>RomanDate.LengthOfYear Enumeration</td>
<td>50</td>
</tr>
<tr>
<td>1.10.1</td>
<td>Summary</td>
<td>50</td>
</tr>
<tr>
<td>1.10.2</td>
<td>Declaration</td>
<td>50</td>
</tr>
<tr>
<td>1.10.3</td>
<td>Remarks</td>
<td>50</td>
</tr>
<tr>
<td>1.10.4</td>
<td>Members</td>
<td>50</td>
</tr>
<tr>
<td>1.11</td>
<td>RomanDate.MonthOfYear Enumeration</td>
<td>50</td>
</tr>
<tr>
<td>1.11.1</td>
<td>Summary</td>
<td>50</td>
</tr>
<tr>
<td>1.11.2</td>
<td>Declaration</td>
<td>50</td>
</tr>
<tr>
<td>1.11.3</td>
<td>Remarks</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Members</td>
<td>50</td>
</tr>
</tbody>
</table>

2 Namespace UGent.WLES.Data 53

<p>| 2.1    | Overview | 53 |
|        | Classes | 53 |
| 2.1.2  | Structures | 53 |
| 2.1.3  | Enumerations | 53 |
| 2.2    | SoilHorizon Class | 54 |
| 2.2.1  | Summary | 54 |
| 2.2.2  | Declaration | 54 |
| 2.2.3  | Remarks | 54 |
| 2.2.4  | Public Instance Constructors | 55 |
| 2.2.5  | Public Instance Properties | 55 |
| 2.2.6  | Public Instance Methods | 63 |
| 2.3    | SoilProfile Class | 65 |
| 2.3.1  | Summary | 65 |
| 2.3.2  | Declaration | 65 |
| 2.3.3  | Remarks | 65 |
| 2.3.4  | Public Instance Constructors | 67 |
| 2.3.5  | Public Instance Properties | 67 |
| 2.3.6  | Public Instance Methods | 70 |
| 2.4    | SoilTexture Class | 78 |
| 2.4.1  | Summary | 78 |
| 2.4.2  | Declaration | 78 |
| 2.4.3  | Remarks | 79 |
| 2.4.4  | Public Static Methods | 80 |
| 2.4.5  | Public Instance Constructors | 81 |
| 2.4.6  | Public Instance Properties | 81 |
| 2.4.7  | Public Instance Methods | 82 |
| 2.5    | ClimaticData Structure | 82 |
| 2.5.1  | Summary | 82 |
| 2.5.2  | Declaration | 82 |
| 2.5.3  | Public Static Fields | 82 |
| 2.5.4  | Public Static Methods | 83 |
| 2.6    | CropData Structure | 86 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.1 Summary</td>
<td>86</td>
</tr>
<tr>
<td>2.6.2 Declaration</td>
<td>86</td>
</tr>
<tr>
<td>2.6.3 Remarks</td>
<td>86</td>
</tr>
<tr>
<td>2.6.4 Public Static Methods</td>
<td>87</td>
</tr>
<tr>
<td>2.6.5 Internal Static Methods</td>
<td>93</td>
</tr>
<tr>
<td>2.7 Database Structure</td>
<td>93</td>
</tr>
<tr>
<td>2.7.1 Summary</td>
<td>93</td>
</tr>
<tr>
<td>2.7.2 Declaration</td>
<td>93</td>
</tr>
<tr>
<td>2.7.3 Public Static Properties</td>
<td>93</td>
</tr>
<tr>
<td>2.7.4 Public Static Methods</td>
<td>93</td>
</tr>
<tr>
<td>2.7.5 Internal Static Methods</td>
<td>97</td>
</tr>
<tr>
<td>2.8 SoilData Structure</td>
<td>97</td>
</tr>
<tr>
<td>2.8.1 Summary</td>
<td>97</td>
</tr>
<tr>
<td>2.8.2 Declaration</td>
<td>97</td>
</tr>
<tr>
<td>2.8.3 Remarks</td>
<td>97</td>
</tr>
<tr>
<td>2.8.4 Public Static Fields</td>
<td>98</td>
</tr>
<tr>
<td>2.8.5 Public Static Methods</td>
<td>98</td>
</tr>
<tr>
<td>2.9 WlesDataSet Structure</td>
<td>100</td>
</tr>
<tr>
<td>2.9.1 Summary</td>
<td>100</td>
</tr>
<tr>
<td>2.9.2 Declaration</td>
<td>100</td>
</tr>
<tr>
<td>2.9.3 Remarks</td>
<td>100</td>
</tr>
<tr>
<td>2.9.4 Examples</td>
<td>100</td>
</tr>
<tr>
<td>2.9.5 Public Instance Properties</td>
<td>102</td>
</tr>
<tr>
<td>2.9.6 Public Instance Methods</td>
<td>103</td>
</tr>
<tr>
<td>2.10 ClimaticData.ClimaticParamSet Enumeration</td>
<td>113</td>
</tr>
<tr>
<td>2.10.1 Summary</td>
<td>113</td>
</tr>
<tr>
<td>2.10.2 Declaration</td>
<td>113</td>
</tr>
<tr>
<td>2.10.3 Remarks</td>
<td>114</td>
</tr>
<tr>
<td>2.10.4 Members</td>
<td>114</td>
</tr>
<tr>
<td>2.11 ClimaticData.HemisphereOfEarth Enumeration</td>
<td>114</td>
</tr>
<tr>
<td>2.11.1 Summary</td>
<td>114</td>
</tr>
<tr>
<td>2.11.2 Declaration</td>
<td>114</td>
</tr>
<tr>
<td>2.11.3 Remarks</td>
<td>114</td>
</tr>
<tr>
<td>2.11.4 Members</td>
<td>114</td>
</tr>
<tr>
<td>2.12 ClimaticData.TemperatureRegime Enumeration</td>
<td>114</td>
</tr>
<tr>
<td>2.12.1 Summary</td>
<td>114</td>
</tr>
<tr>
<td>2.12.2 Declaration</td>
<td>114</td>
</tr>
<tr>
<td>2.12.3 Members</td>
<td>115</td>
</tr>
<tr>
<td>2.13 CropData.FarmingSystem Enumeration</td>
<td>115</td>
</tr>
<tr>
<td>2.13.1 Summary</td>
<td>115</td>
</tr>
<tr>
<td>2.13.2 Declaration</td>
<td>115</td>
</tr>
<tr>
<td>2.13.3 Members</td>
<td>115</td>
</tr>
<tr>
<td>2.14 SoilData.GravelType Enumeration</td>
<td>115</td>
</tr>
<tr>
<td>2.14.1 Summary</td>
<td>115</td>
</tr>
<tr>
<td>2.14.2 Declaration</td>
<td>115</td>
</tr>
<tr>
<td>2.14.3 Members</td>
<td>115</td>
</tr>
<tr>
<td>2.15 SoilData.LandSuitabilityClass Enumeration</td>
<td>115</td>
</tr>
<tr>
<td>2.15.1 Summary</td>
<td>115</td>
</tr>
<tr>
<td>2.15.2 Declaration</td>
<td>115</td>
</tr>
<tr>
<td>2.15.3 Remarks</td>
<td>115</td>
</tr>
<tr>
<td>2.15.4 Members</td>
<td>116</td>
</tr>
<tr>
<td>2.16 SoilData.SandClass Enumeration</td>
<td>116</td>
</tr>
<tr>
<td>2.16.1 Summary</td>
<td>116</td>
</tr>
<tr>
<td>2.16.2 Declaration</td>
<td>116</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.16.3 Remarks</td>
<td>116</td>
</tr>
<tr>
<td>2.16.4 Members</td>
<td>116</td>
</tr>
<tr>
<td>2.17 SoilData.SoilCharacteristic Enumeration</td>
<td>116</td>
</tr>
<tr>
<td>2.17.1 Summary</td>
<td>116</td>
</tr>
<tr>
<td>2.17.2 Declaration</td>
<td>117</td>
</tr>
<tr>
<td>2.17.3 Remarks</td>
<td>117</td>
</tr>
<tr>
<td>2.17.4 Members</td>
<td>117</td>
</tr>
<tr>
<td>2.18 SoilData.SoilDrainageClass Enumeration</td>
<td>118</td>
</tr>
<tr>
<td>2.18.1 Summary</td>
<td>118</td>
</tr>
<tr>
<td>2.18.2 Declaration</td>
<td>118</td>
</tr>
<tr>
<td>2.18.3 Remarks</td>
<td>118</td>
</tr>
<tr>
<td>2.18.4 Members</td>
<td>118</td>
</tr>
<tr>
<td>2.19 SoilData.SoilFertilityCharacteristic Enumeration</td>
<td>119</td>
</tr>
<tr>
<td>2.19.1 Summary</td>
<td>119</td>
</tr>
<tr>
<td>2.19.2 Declaration</td>
<td>119</td>
</tr>
<tr>
<td>2.19.3 Remarks</td>
<td>119</td>
</tr>
<tr>
<td>2.19.4 Members</td>
<td>119</td>
</tr>
<tr>
<td>2.20 SoilData.SoilStructure Enumeration</td>
<td>119</td>
</tr>
<tr>
<td>2.20.1 Summary</td>
<td>119</td>
</tr>
<tr>
<td>2.20.2 Declaration</td>
<td>119</td>
</tr>
<tr>
<td>2.20.3 Remarks</td>
<td>120</td>
</tr>
<tr>
<td>2.20.4 Members</td>
<td>121</td>
</tr>
<tr>
<td>2.21 SoilData.SoilTextureClass Enumeration</td>
<td>121</td>
</tr>
<tr>
<td>2.21.1 Summary</td>
<td>121</td>
</tr>
<tr>
<td>2.21.2 Declaration</td>
<td>121</td>
</tr>
<tr>
<td>2.21.3 Remarks</td>
<td>121</td>
</tr>
<tr>
<td>2.21.4 Members</td>
<td>122</td>
</tr>
<tr>
<td>2.22 SoilData.WRBSoilGroup Enumeration</td>
<td>122</td>
</tr>
<tr>
<td>2.22.1 Summary</td>
<td>122</td>
</tr>
<tr>
<td>2.22.2 Declaration</td>
<td>122</td>
</tr>
<tr>
<td>2.22.3 Remarks</td>
<td>123</td>
</tr>
<tr>
<td>2.22.4 Members</td>
<td>123</td>
</tr>
<tr>
<td>2.23 WlesDataSet.TemporalResolution Enumeration</td>
<td>128</td>
</tr>
<tr>
<td>2.23.1 Summary</td>
<td>128</td>
</tr>
<tr>
<td>2.23.2 Declaration</td>
<td>128</td>
</tr>
<tr>
<td>2.23.3 Remarks</td>
<td>128</td>
</tr>
<tr>
<td>2.23.4 Members</td>
<td>128</td>
</tr>
<tr>
<td>3 Namespace UGent.WLES.Topo</td>
<td>129</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>129</td>
</tr>
<tr>
<td>3.1.1 Structures</td>
<td>129</td>
</tr>
<tr>
<td>3.2 Line Structure</td>
<td>129</td>
</tr>
<tr>
<td>3.2.1 Summary</td>
<td>129</td>
</tr>
<tr>
<td>3.2.2 Declaration</td>
<td>129</td>
</tr>
<tr>
<td>3.2.3 Remarks</td>
<td>129</td>
</tr>
<tr>
<td>3.2.4 Public Instance Constructors</td>
<td>130</td>
</tr>
<tr>
<td>3.2.5 Public Instance Properties</td>
<td>130</td>
</tr>
<tr>
<td>3.2.6 Public Instance Methods</td>
<td>130</td>
</tr>
<tr>
<td>3.3 Point Structure</td>
<td>130</td>
</tr>
<tr>
<td>3.3.1 Summary</td>
<td>130</td>
</tr>
<tr>
<td>3.3.2 Declaration</td>
<td>130</td>
</tr>
<tr>
<td>3.3.3 Remarks</td>
<td>130</td>
</tr>
<tr>
<td>3.3.4 Public Instance Constructors</td>
<td>131</td>
</tr>
<tr>
<td>3.3.5 Public Instance Properties</td>
<td>131</td>
</tr>
</tbody>
</table>
CONTENTS

3.3.6 Public Instance Methods .................................................. 131
3.4 Polygon Structure ............................................................ 132
  3.4.1 Summary ................................................................. 132
  3.4.2 Declaration .............................................................. 132
  3.4.3 Remarks ................................................................. 132
  3.4.4 Public Instance Constructors ........................................ 132
  3.4.5 Public Instance Properties .......................................... 132
  3.4.6 Public Instance Methods ............................................. 133
3.5 Polyline Structure ........................................................... 133
  3.5.1 Summary ................................................................. 133
  3.5.2 Declaration .............................................................. 133
  3.5.3 Remarks ................................................................. 133
  3.5.4 Public Instance Constructors ....................................... 134
  3.5.5 Public Instance Properties .......................................... 134
  3.5.6 Public Instance Methods ............................................. 134
4 Namespace UGent.WLES.RPP .................................................. 135
  4.1 Overview ........................................................................ 135
    4.1.1 Classes .................................................................... 135
    4.1.2 Enumerations .......................................................... 135
  4.2 Biomass Class ................................................................. 136
    4.2.1 Summary .................................................................. 136
    4.2.2 Declaration .............................................................. 136
    4.2.3 Remarks .................................................................. 136
    4.2.4 Example .................................................................. 136
    4.2.5 Public Static Methods .............................................. 137
    4.2.6 Public Instance Methods ........................................... 143
  4.3 CropCycle Class ................................................................. 145
    4.3.1 Summary .................................................................. 145
    4.3.2 Declaration .............................................................. 145
    4.3.3 Remarks .................................................................. 145
    4.3.4 Example .................................................................. 145
    4.3.5 Public Static Methods .............................................. 146
    4.3.6 Public Instance Constructors ....................................... 149
    4.3.7 Public Instance Properties .......................................... 150
    4.3.8 Public Instance Methods ........................................... 150
  4.4 GrowingPeriod Class .......................................................... 155
    4.4.1 Summary .................................................................. 155
    4.4.2 Declaration .............................................................. 155
    4.4.3 Remarks .................................................................. 156
    4.4.4 Note ...................................................................... 156
    4.4.5 Examples .................................................................. 156
    4.4.6 Public Instance Constructors ....................................... 160
    4.4.7 Public Instance Properties .......................................... 161
    4.4.8 Public Instance Methods ........................................... 166
  4.5 GrowingPeriodUI Class ....................................................... 168
    4.5.1 Summary .................................................................. 168
    4.5.2 Declaration .............................................................. 168
    4.5.3 Remarks .................................................................. 169
    4.5.4 Internal Static Methods ............................................. 169
  4.6 RPPReport Class ................................................................. 169
    4.6.1 Summary .................................................................. 169
    4.6.2 Declaration .............................................................. 169
    4.6.3 Remarks .................................................................. 169
4.6.4 Internal Static Methods .............................................. 169
4.7 Biomass.PhotosynGroup Enumeration ................................. 170
  4.7.1 Summary .................................................. 170
  4.7.2 Declaration ............................................... 170
  4.7.3 Remarks ................................................ 170
  4.7.4 Members ................................................ 170
4.8 Biomass.PhotosynPathway Enumeration .............................. 170
  4.8.1 Summary ................................................ 170
  4.8.2 Declaration ............................................... 170
  4.8.3 Remarks ................................................ 170
  4.8.4 Members ................................................ 170
4.9 CropCycle.CropDevelopmentStage Enumeration ..................... 171
  4.9.1 Summary ................................................ 171
  4.9.2 Declaration ............................................... 171
  4.9.3 Remarks ................................................ 171
  4.9.4 Members ................................................ 172
4.10 CropCycle.CropGrowthPeriod Enumeration .......................... 172
  4.10.1 Summary ................................................ 172
  4.10.2 Declaration ............................................... 172
  4.10.3 Remarks ................................................ 172
  4.10.4 Members ................................................ 173
4.11 CropCycle.SubCGP Enumeration ...................................... 173
  4.11.1 Summary ................................................ 173
  4.11.2 Declaration ............................................... 173
  4.11.3 Remarks ................................................ 173
  4.11.4 Members ................................................ 173

5 Namespace UGent.WLES.WPP ........................................... 175
  5.1 Overview .................................................. 175
  5.1.1 Classes ................................................ 175
  5.1.2 Delegates ............................................... 176
  5.1.3 Enumerations ........................................... 176
5.2 CropCoefficient Class .................................................. 176
  5.2.1 Summary ................................................ 176
  5.2.2 Declaration ............................................... 176
  5.2.3 Remarks ................................................ 176
  5.2.4 Public Static Methods ..................................... 177
  5.2.5 Public Instance Constructors ................................ 179
  5.2.6 Public Instance Properties ................................ 180
  5.2.7 Public Instance Methods .................................. 181
5.3 CropCoefficientUI Class ............................................... 184
  5.3.1 Summary ................................................ 184
  5.3.2 Declaration ............................................... 184
  5.3.3 Remarks ................................................ 184
  5.3.4 Internal Static Methods ................................... 185
5.4 DepletionFractionUI Class ........................................... 185
  5.4.1 Summary ................................................ 185
  5.4.2 Declaration ............................................... 185
  5.4.3 Remarks ................................................ 186
  5.4.4 Internal Static Methods ................................... 186
5.5 EffectiveRainfall Class ............................................... 186
  5.5.1 Summary ................................................ 186
  5.5.2 Declaration ............................................... 186
  5.5.3 Remarks ................................................ 186
CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5.4</td>
<td>Examples</td>
<td>187</td>
</tr>
<tr>
<td>5.5.5</td>
<td>Public Static Methods</td>
<td>190</td>
</tr>
<tr>
<td>5.6</td>
<td>EffectiveRainfallUI Class</td>
<td>194</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Summary</td>
<td>194</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Declaration</td>
<td>194</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Remarks</td>
<td>194</td>
</tr>
<tr>
<td>5.6.4</td>
<td>Internal Static Methods</td>
<td>194</td>
</tr>
<tr>
<td>5.7</td>
<td>PenmanMonteith Class</td>
<td>195</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Summary</td>
<td>195</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Declaration</td>
<td>195</td>
</tr>
<tr>
<td>5.7.3</td>
<td>Remarks</td>
<td>195</td>
</tr>
<tr>
<td>5.7.4</td>
<td>Public Static Fields</td>
<td>195</td>
</tr>
<tr>
<td>5.7.5</td>
<td>Public Static Methods</td>
<td>196</td>
</tr>
<tr>
<td>5.8</td>
<td>RootingDepth Class</td>
<td>200</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Summary</td>
<td>200</td>
</tr>
<tr>
<td>5.8.2</td>
<td>Declaration</td>
<td>200</td>
</tr>
<tr>
<td>5.8.3</td>
<td>Remarks</td>
<td>200</td>
</tr>
<tr>
<td>5.8.4</td>
<td>Public Static Methods</td>
<td>200</td>
</tr>
<tr>
<td>5.8.5</td>
<td>Public Instance Constructors</td>
<td>201</td>
</tr>
<tr>
<td>5.8.6</td>
<td>Public Instance Properties</td>
<td>201</td>
</tr>
<tr>
<td>5.8.7</td>
<td>Public Instance Methods</td>
<td>202</td>
</tr>
<tr>
<td>5.9</td>
<td>RootingDepthUI Class</td>
<td>202</td>
</tr>
<tr>
<td>5.9.1</td>
<td>Summary</td>
<td>202</td>
</tr>
<tr>
<td>5.9.2</td>
<td>Declaration</td>
<td>202</td>
</tr>
<tr>
<td>5.9.3</td>
<td>Remarks</td>
<td>202</td>
</tr>
<tr>
<td>5.9.4</td>
<td>Internal Static Methods</td>
<td>203</td>
</tr>
<tr>
<td>5.10</td>
<td>WaterBalance Class</td>
<td>203</td>
</tr>
<tr>
<td>5.10.1</td>
<td>Summary</td>
<td>203</td>
</tr>
<tr>
<td>5.10.2</td>
<td>Declaration</td>
<td>203</td>
</tr>
<tr>
<td>5.10.3</td>
<td>Remarks</td>
<td>203</td>
</tr>
<tr>
<td>5.10.4</td>
<td>Examples</td>
<td>204</td>
</tr>
<tr>
<td>5.10.5</td>
<td>Public Static Methods</td>
<td>209</td>
</tr>
<tr>
<td>5.10.6</td>
<td>Public Instance Constructors</td>
<td>212</td>
</tr>
<tr>
<td>5.10.7</td>
<td>Public Instance Properties</td>
<td>213</td>
</tr>
<tr>
<td>5.10.8</td>
<td>Public Instance Methods</td>
<td>215</td>
</tr>
<tr>
<td>5.11</td>
<td>WaterBalanceUI Class</td>
<td>218</td>
</tr>
<tr>
<td>5.11.1</td>
<td>Summary</td>
<td>218</td>
</tr>
<tr>
<td>5.11.2</td>
<td>Declaration</td>
<td>218</td>
</tr>
<tr>
<td>5.11.3</td>
<td>Remarks</td>
<td>218</td>
</tr>
<tr>
<td>5.11.4</td>
<td>Internal Static Methods</td>
<td>219</td>
</tr>
<tr>
<td>5.12</td>
<td>WPPReport Class</td>
<td>219</td>
</tr>
<tr>
<td>5.12.1</td>
<td>Summary</td>
<td>219</td>
</tr>
<tr>
<td>5.12.2</td>
<td>Declaration</td>
<td>219</td>
</tr>
<tr>
<td>5.12.3</td>
<td>Remarks</td>
<td>219</td>
</tr>
<tr>
<td>5.12.4</td>
<td>Internal Static Methods</td>
<td>219</td>
</tr>
<tr>
<td>5.13</td>
<td>YieldResponseToWater Class</td>
<td>220</td>
</tr>
<tr>
<td>5.13.1</td>
<td>Summary</td>
<td>220</td>
</tr>
<tr>
<td>5.13.2</td>
<td>Declaration</td>
<td>220</td>
</tr>
<tr>
<td>5.13.3</td>
<td>Remarks</td>
<td>220</td>
</tr>
<tr>
<td>5.13.4</td>
<td>Example</td>
<td>220</td>
</tr>
<tr>
<td>5.13.5</td>
<td>Public Static Methods</td>
<td>222</td>
</tr>
<tr>
<td>5.13.6</td>
<td>Public Instance Constructors</td>
<td>224</td>
</tr>
<tr>
<td>5.13.7</td>
<td>Public Instance Properties</td>
<td>225</td>
</tr>
<tr>
<td>5.13.8</td>
<td>Public Instance Methods</td>
<td>226</td>
</tr>
<tr>
<td>Section</td>
<td>Class/Delegate/Enumeration</td>
<td>Subsections</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>5.14</td>
<td>YieldResponseToWaterUI Class</td>
<td>5.14.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.14.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.14.3 Remarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.14.4 Internal Static Methods</td>
</tr>
<tr>
<td>5.15</td>
<td>EffectiveRainfall.UserDefinedMethod Delegate</td>
<td>5.15.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.15.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.15.3 Example</td>
</tr>
<tr>
<td>5.16</td>
<td>EffectiveRainfall.Approach Enumeration</td>
<td>5.16.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.16.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.16.3 Members</td>
</tr>
<tr>
<td>5.17</td>
<td>WaterBalance.SoilWaterDepletionGroup Enumeration</td>
<td>5.17.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.17.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.17.3 Members</td>
</tr>
<tr>
<td>6</td>
<td>Namespace UGent.WLES.LPP</td>
<td>6.1 Overview</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.1.1 Classes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.1.2 Enumerations</td>
</tr>
<tr>
<td>6.2</td>
<td>CropRequirements Class</td>
<td>6.2.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.3 Remarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.4 Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.5 Public Static Methods</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.6 Public Instance Constructors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.7 Public Instance Properties</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2.8 Public Instance Methods</td>
</tr>
<tr>
<td>6.3</td>
<td>CropRequirementsUI Class</td>
<td>6.3.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.3.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.3.3 Remarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.3.4 Internal Static Methods</td>
</tr>
<tr>
<td>6.4</td>
<td>LPPReport Class</td>
<td>6.4.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.4.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.4.3 Remarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.4.4 Internal Static Methods</td>
</tr>
<tr>
<td>6.5</td>
<td>ManagementIndex Class</td>
<td>6.5.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.5.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.5.3 Remarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.5.4 Example</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.5.5 Public Static Methods</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.5.6 Public Instance Constructors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.5.7 Public Instance Properties</td>
</tr>
<tr>
<td>6.6</td>
<td>ManagementIndexUI Class</td>
<td>6.6.1 Summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.6.2 Declaration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.6.3 Remarks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.6.4 Internal Static Methods</td>
</tr>
</tbody>
</table>
6.7 SoilIndex Class .......................................................... 250
  6.7.1 Summary ........................................................... 250
  6.7.2 Declaration ......................................................... 250
  6.7.3 Remarks ............................................................ 251
  6.7.4 Example ............................................................. 251
  6.7.5 Public Instance Constructors ...................................... 253
  6.7.6 Public Instance Properties ....................................... 254
  6.7.7 Public Instance Methods .......................................... 255
6.8 SoilIndexUI Class ...................................................... 255
  6.8.1 Summary ........................................................... 255
  6.8.2 Declaration ......................................................... 255
  6.8.3 Remarks ............................................................ 255
  6.8.4 Internal Static Methods .......................................... 255
6.9 Yield Class ............................................................. 255
  6.9.1 Summary ........................................................... 255
  6.9.2 Declaration ......................................................... 255
  6.9.3 Remarks ............................................................ 256
  6.9.4 Public Static Methods ............................................. 256
  6.9.5 Public Instance Constructors ..................................... 256
  6.9.6 Public Instance Properties ....................................... 257
  6.9.7 Public Instance Methods .......................................... 257
6.10 ManagementIndex.InputLevel Enumeration .............................. 258
  6.10.1 Summary ........................................................... 258
  6.10.2 Declaration ......................................................... 258
  6.10.3 Remarks ............................................................ 258
  6.10.4 Members ............................................................ 258
6.11 ManagementIndex.ManagementGroup Enumeration ....................... 259
  6.11.1 Summary ........................................................... 259
  6.11.2 Declaration ......................................................... 259
  6.11.3 Remarks ............................................................ 259
  6.11.4 Members ............................................................ 259
6.12 ManagementIndex.Scheme Enumeration .................................. 259
  6.12.1 Summary ........................................................... 259
  6.12.2 Declaration ......................................................... 259
  6.12.3 Members ............................................................ 259
6.13 SoilIndex.Algorithm Enumeration ..................................... 260
  6.13.1 Summary ........................................................... 260
  6.13.2 Declaration ......................................................... 260
  6.13.3 Remarks ............................................................ 260
  6.13.4 Members ............................................................ 260
7 Namespace UGent.WLES.Users .............................................. 261
  7.1 Overview ................................................................ 261
    7.1.1 Classes ............................................................. 261
  7.2 Account Class ........................................................ 262
    7.2.1 Summary ............................................................. 262
    7.2.2 Declaration ........................................................ 262
    7.2.3 Public Static Methods ............................................. 262
    7.2.4 Public Instance Constructors ..................................... 263
    7.2.5 Public Instance Properties ....................................... 263
    7.2.6 Public Instance Methods .......................................... 264
  7.3 AuthCheck Class ....................................................... 266
    7.3.1 Summary ............................................................. 266
    7.3.2 Declaration ........................................................ 266
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Namespace UGent.WLES.Help</td>
<td>275</td>
</tr>
<tr>
<td>8.1</td>
<td>Overview</td>
<td>275</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Classes</td>
<td>275</td>
</tr>
<tr>
<td>8.2</td>
<td>Topics Class</td>
<td>275</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Summary</td>
<td>275</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Declaration</td>
<td>275</td>
</tr>
<tr>
<td>8.2.3</td>
<td>Remarks</td>
<td>275</td>
</tr>
<tr>
<td>9</td>
<td>Namespace UGent.WLES.Interactive</td>
<td>277</td>
</tr>
<tr>
<td>9.1</td>
<td>Overview</td>
<td>277</td>
</tr>
<tr>
<td>9.1.1</td>
<td>Classes</td>
<td>277</td>
</tr>
<tr>
<td>9.2</td>
<td>ClimaticDataUI Class</td>
<td>277</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Summary</td>
<td>277</td>
</tr>
<tr>
<td>9.2.2</td>
<td>Declaration</td>
<td>277</td>
</tr>
<tr>
<td>9.2.3</td>
<td>Remarks</td>
<td>277</td>
</tr>
<tr>
<td>9.2.4</td>
<td>Internal Static Methods</td>
<td>278</td>
</tr>
<tr>
<td>9.3</td>
<td>CropDataUI Class</td>
<td>279</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Summary</td>
<td>279</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Declaration</td>
<td>279</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Remarks</td>
<td>279</td>
</tr>
<tr>
<td>9.3.4</td>
<td>Internal Static Methods</td>
<td>280</td>
</tr>
<tr>
<td>9.4</td>
<td>Evaluation Class</td>
<td>281</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Summary</td>
<td>281</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Declaration</td>
<td>281</td>
</tr>
<tr>
<td>9.5</td>
<td>SoilCharSet Class</td>
<td>281</td>
</tr>
<tr>
<td>9.5.1</td>
<td>Summary</td>
<td>281</td>
</tr>
<tr>
<td>9.5.2</td>
<td>Declaration</td>
<td>281</td>
</tr>
<tr>
<td>9.5.3</td>
<td>Remarks</td>
<td>281</td>
</tr>
<tr>
<td>9.5.4</td>
<td>Internal Static Methods</td>
<td>282</td>
</tr>
<tr>
<td>9.6</td>
<td>SoilDataUI1 Class</td>
<td>282</td>
</tr>
<tr>
<td>9.6.1</td>
<td>Summary</td>
<td>282</td>
</tr>
<tr>
<td>9.6.2</td>
<td>Declaration</td>
<td>282</td>
</tr>
<tr>
<td>9.6.3</td>
<td>Remarks</td>
<td>282</td>
</tr>
<tr>
<td>9.6.4</td>
<td>Internal Static Methods</td>
<td>282</td>
</tr>
<tr>
<td>9.7</td>
<td>SoilDataUI2 Class</td>
<td>284</td>
</tr>
<tr>
<td>9.7.1</td>
<td>Summary</td>
<td>284</td>
</tr>
<tr>
<td>9.7.2</td>
<td>Declaration</td>
<td>284</td>
</tr>
<tr>
<td>9.7.3</td>
<td>Remarks</td>
<td>284</td>
</tr>
<tr>
<td>9.7.4</td>
<td>Internal Static Methods</td>
<td>285</td>
</tr>
<tr>
<td>9.8</td>
<td>StudyArea Class</td>
<td>286</td>
</tr>
<tr>
<td>9.8.1</td>
<td>Summary</td>
<td>286</td>
</tr>
<tr>
<td>9.8.2</td>
<td>Declaration</td>
<td>286</td>
</tr>
<tr>
<td>9.8.3</td>
<td>Remarks</td>
<td>286</td>
</tr>
<tr>
<td>9.8.4</td>
<td>Public Static Methods</td>
<td>287</td>
</tr>
<tr>
<td>9.8.5</td>
<td>Internal Static Methods</td>
<td>287</td>
</tr>
<tr>
<td>10</td>
<td>Namespace UGent.WLES.Batch</td>
<td>289</td>
</tr>
<tr>
<td>10.1</td>
<td>Overview</td>
<td>289</td>
</tr>
<tr>
<td>10.1.1</td>
<td>Classes</td>
<td>289</td>
</tr>
<tr>
<td>10.2</td>
<td>Evaluation Class</td>
<td>289</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Summary</td>
<td>289</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Declaration</td>
<td>289</td>
</tr>
<tr>
<td>10.2.3</td>
<td>Remarks</td>
<td>289</td>
</tr>
</tbody>
</table>
## 11 Namespace UGent.WLES.ErrorHandler

11.1 Overview ........................................ 291
  11.1.1 Classes ..................................... 291

11.2 GeneralError Class .............................. 291
  11.2.1 Summary .................................... 291
  11.2.2 Declaration ................................... 291
  11.2.3 Remarks ..................................... 291

11.3 PageNotFoundError Class ......................... 292
  11.3.1 Summary .................................... 292
  11.3.2 Declaration ................................... 292
  11.3.3 Remarks ..................................... 292

11.4 ServerError Class .............................. 292
  11.4.1 Summary .................................... 292
  11.4.2 Declaration ................................... 292
  11.4.3 Remarks ..................................... 292

## 12 Namespace UGent.IP2Country

12.1 Overview ........................................ 293
  12.1.1 Classes ..................................... 293
  12.1.2 Enumerations ................................ 293

12.2 ClientInfo Class ............................... 293
  12.2.1 Summary .................................... 293
  12.2.2 Declaration ................................... 293
  12.2.3 Example ..................................... 293
  12.2.4 Public Static Methods ....................... 294
  12.2.5 Public Instance Constructors ............... 294
  12.2.6 Public Instance Properties ................. 294
  12.2.7 Public Instance Methods .................... 295
  12.2.8 Protected Instance Properties ............. 295

12.3 ClientInfo.ImageFormat Enumeration .......... 295
  12.3.1 Summary .................................... 295
  12.3.2 Declaration ................................... 295
  12.3.3 Remarks ..................................... 295
  12.3.4 Members .................................... 295

12.4 ClientInfo.ImageSize enumeration .......... 296
  12.4.1 Summary .................................... 296
  12.4.2 Declaration ................................... 296
  12.4.3 Remarks ..................................... 296
  12.4.4 Members .................................... 296

Bibliography ........................................ 297

Index .................................................. 299
WLES Class Library
Documentation
How to Use This Document

The WLES Class Library at its entirety is briefly presented in this document. Great effort has been made to keep this document as up-to-date as possible with the online edition. However, due to the difference of the length of the updating periods, changes made to the WLES Class Library are sooner to be reflected in the online edition (than in this document); Moreover, the online edition is also more preferable when details are concerned.

Readers are recommended to take the hard-copy of this document as a guideline, and the online edition as a reference whenever ambiguity occurs, especially for the definitions of method parameters.

The online edition is found at: http://weble.ugent.be/WLES/doc.
# Namespace List

The following namespaces are included in this document:

<table>
<thead>
<tr>
<th>namespace</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UGent.WLES</td>
<td>7</td>
</tr>
<tr>
<td>UGent.WLES.Data</td>
<td>53</td>
</tr>
<tr>
<td>UGent.WLES.Topo</td>
<td>129</td>
</tr>
<tr>
<td>UGent.WLES.RPP</td>
<td>135</td>
</tr>
<tr>
<td>UGent.WLES.WPP</td>
<td>175</td>
</tr>
<tr>
<td>UGent.WLES.LPP</td>
<td>233</td>
</tr>
<tr>
<td>UGent.WLES.Users</td>
<td>261</td>
</tr>
<tr>
<td>UGent.WLES.Help</td>
<td>275</td>
</tr>
<tr>
<td>UGent.WLES.Interactive</td>
<td>277</td>
</tr>
<tr>
<td>UGent.WLES.Batch</td>
<td>289</td>
</tr>
<tr>
<td>UGent.WLES.ErrorHandler</td>
<td>291</td>
</tr>
<tr>
<td>UGent.IP2Country</td>
<td>293</td>
</tr>
</tbody>
</table>
Chapter 1

Namespace UGent.WLES

The UGent.WLES namespace functions as the Web-GUI of the whole model of the Web-based Land Evaluation System developed by L. Ye at Ghent University, Belgium.

1.1 Overview

1.1.1 Classes

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>Represents the application-wide event handlers for the WLES.</td>
<td>8</td>
</tr>
<tr>
<td>Logistics</td>
<td>Provides common functionalities to other classes and structs and wraps the application settings in the form of consts. It differentiates from the Utilities struct in that Logistics mainly fulfills the needs of the Web UI-related classes and structs, whereas Utilities satisfies more fundamental requirements in building a robustic class library.</td>
<td>8</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Represents database errors which are caused by the WLES application.</td>
<td>30</td>
</tr>
<tr>
<td>WlesException</td>
<td>Represents errors which are caused by the WLES application.</td>
<td>48</td>
</tr>
</tbody>
</table>

1.1.2 Structures

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RomanDate</td>
<td>Represents a date by specifying the month, the day and the calendar.</td>
<td>30</td>
</tr>
<tr>
<td>Utilities</td>
<td>Provides common functionalities to other es and s. It differentiates from the in that mainly fulfills the needs of the Web UI-related es and s, whereas satisfies more fundamental requirements in building a robustic class library.</td>
<td>36</td>
</tr>
</tbody>
</table>

1.1.3 Enumerations

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistics.InternalLink</td>
<td>Represents the most frequently used hyperlinks within WLES. The links can be either relative, to the Webserver root, or absolute.</td>
<td>48</td>
</tr>
<tr>
<td>RomanDate.CalendarType</td>
<td>Specifies the calendar type.</td>
<td>49</td>
</tr>
<tr>
<td>RomanDate.LengthOfYear</td>
<td>Specifies the length of a year in terms of months, decades or days.</td>
<td>50</td>
</tr>
<tr>
<td>RomanDate.MonthOfYear</td>
<td>Specifies the month of the year.</td>
<td>50</td>
</tr>
</tbody>
</table>
1.2 Global Class

1.2.1 Summary
Represents the application-wide event handlers for the WLES.

1.2.2 Declaration
[C#] public class Global : HttpApplication

1.2.3 Remarks
The event handlers defined in this class are triggered the first time any resource or URL within WLES application is activated or requested.

1.2.4 Members
Refer to the HttpApplication class in .NET Framework Documentation.

1.3 Logistics Class

1.3.1 Summary
Provides common functionalities to other classes and structs and wraps the application settings in the form of consts. It differentiates from the Utilities struct in that Logistics mainly fulfills the needs of the Web UI-related classes and structs, whereas Utilities satisfies more fundamental requirements in building a robust class library.

1.3.2 Declaration
[C#] public class Logistics

1.3.3 Remarks
The functionalities implemented by the Logistics class are grouped as:

1. Application configuration settings: The settings are stored as “key=value” pairs in the “appSettings” section of the application-wide configuration file named Web.config which is located in the root directory of WLES at the hosting Web server. Listed below are good examples of such application settings:

- AuthorEmail: The email address of the WLES author;
- CookieName: The name of the cookie used for WLES;
- CookieTimeout: The number of minutes that the CookieName cookie Expires since issued;
- IsCookiePersistent: Determines whether the WLES cookie is persistent;
- MaxLoginRetry: The maximum number of failures a user is allowed to have during authentication;
- MaxLengthUid: The maximum number of characters a username may contain;
- MaxLengthPwd: The maximum number of characters a password phrase may contain;
- MinLengthPwd: The minimum number of characters a password must contain;
- UidRule: Legal Chars that may present in a user ID, together with letters and digits;
- PwdRule: Legal Chars that may present in a password, together with letters and digits.
2. User related operations: The operations are either information retrieving or user account handling.

- CopyrightMessage: Gets the HTML-encoded WLES copyright information that is displayed inside the WLES Web UI;
- GetClientInfo: Gets the HTML-encoded data of the Web client. The data include the username, the country flag image and the IP address;
- IsUidLegal: Determines the legitimacy of a user ID using the the predefined UID rule;
- IsPwdLegal: Determines the legitimacy of a user-provided password according to the predefined password rule;
- IsUserValidated: Validates the user by checking the format and the validity of the credential provided;
- DoLogin: Logs the user in by issuing a cookie with CookieName as its name and an encrypted FormsAuthenticationTicket as its value, and redirects the request to a particular destination;
- DoGuestLogin: Logs the guest user in;
- DoLogout: Logs the user out by signing out of the FormsAuthentication mechanism and by removing the cookie added upon login;
- GetUserNameAndData: Retrieves the username and user data from the user’s HttpRequest to a particular Web Form page;
- Log: Log the user’s visit. The user may be authenticated or otherwise;
- IsAuthenticated: Determines whether a user is authenticated, guest user included;
- IsGuest: Determines whether the guest user is authenticated with a particular Web Form page.

3. Other logistical services: The functionalities of this category ranges from internal URL handling to help content manipulation, and so forth.

- HasKey: Determines whether a particular key-value pair has been defined in the AppSettings file Web.config;
- GetUrlValue: Retrieves the value of a given internal hyperlink in the WLES Web UI. The names of the hyperlinks are maintained by the InternalLink enumeration and the values by the ConfigurationSettings.AppSettings;
- GetUrlValue: Retrieves the URL value of a particular String;
- GetHelpTopics: Retrieves the help topics from the WLES database, which may be used to populate, for example, a DropDownList Web control;
- GetHelpContent: Retrieves the help text from the WLES database for a given help topic;
- ReplaceUrls: Searches a String for all occurrences of Logistics.InternalLinks, and replaces them with values defined in the AppSettings;
- ReplaceKeys: Searches a String for all words starting with “key”, and replaces them with values defined in the AppSettings.

1.3.4 Public Static Fields

- AdminEmail
  The email address of the WLES administrator.
  [C#] public static readonly string AdminEmail;

- AuthorEmail
  The email address of the WLES author.
  [C#] public static readonly string AuthorEmail;
• CookieName
  The name of the cookie used for WLES.
  [C#] public static readonly string CookieName;

• CookieTimeout
  The number of minutes that the WLES cookie expires after being issued.
  [C#] public static readonly int CookieTimeout;

• HorizonIDRule
  A String consisting of legal Char s that may present in a soil horizon ID, together with letters and digits.
  [C#] public static readonly string HorizonIDRule;

• HostRule
  A String consisting of legal Char s that may present in a hostname, together with letters and digits.
  [C#] public static readonly string HostRule;

• IsCookiePersistent
  The persistence of the WLES cookie. true as persistent, whereas false causes the cookie to expire at a certain moment which is defined as a particular number of minutes from now, which is again predefined in the authentication section of the application-wide Web.config file.
  [C#] public static readonly bool IsCookiePersistent;

• LabHomePage
  The homepage address of the laboratory in which WLES is developed.
  [C#] public static readonly string LabHomePage;

• LicenseCookieName
  The name of the cookie used for WLES License.
  [C#] public static readonly string LicenseCookieName;

• LoginCountVar
  The session state variable name to count the times of login failures.
  [C#] public static readonly string LoginCountVar;

• MaxLAI
  An Int32 that represents the upper boundary of the leaf area index (LAI) at maximum growth rate value.
  [C#] public static readonly int MaxLAI;

• MaxLengthName
  The maximum number of characters the first name or the last name of a user may contain.
  [C#] public static readonly int MaxLengthName;

• MaxLengthPwd
  The maximum number of characters a password phrase may contain.
  [C#] public static readonly int MaxLengthPwd;

• MaxLengthUid
  The maximum number of characters a username may contain.
  [C#] public static readonly int MaxLengthUid;

• MaxLoginRetry
  The maximum number of failures a user is allowed to have during authentication.
  [C#] public static readonly int MaxLoginRetry;
1.3 Logistics Class

- MinLengthName
  The minimum number of characters the first name or the last name of a user may contain.
  [C#] public static readonly int MinLengthName;

- MinLengthPwd
  The minimum number of characters a password phrase must contain.
  [C#] public static readonly int MinLengthPwd;

- MinLengthUid
  The minimum number of characters a username must contain.
  [C#] public static readonly int MinLengthUid;

- NameRule
  A String consisting of legal Char's that may present in the first name or last name of a user, together with letters.
  [C#] public static readonly string NameRule;

- ProfileNrRule
  A String consisting of legal Char's that may present in a ProfileNumber, together with letters and digits.
  [C#] public static readonly string ProfileNrRule;

- PwdQuestions
  A String array containing questions which can be asked when a forgotten password is to be retrieved.
  [C#] public static readonly string[] PwdQuestions;

- PwdRule
  A String consisting of legal Char's that may present in a password, together with letters and digits.
  [C#] public static readonly string PwdRule;

- SmtpServer
  The SmtpServer used to Send emails in the WLES application.
  [C#] public static readonly string SmtpServer;

- ThisYear
  The String representation of the 4-digit number of this year.
  [C#] public static readonly string ThisYear;

- UidRule
  A String consisting of legal Char's that may present in a user ID, together with letters and digits.
  [C#] public static readonly string UidRule;

- UseAbsoluteUrl
  A Boolean that determines whether to use absolute addresses in WLES for values of Logistics.InternalLinks. URLs pointing to non-WLES destinations are always absolute.
  [C#] public static readonly bool UseAbsoluteUrl;

- UsrTitles
  A String array containing titles which can be used to address users.
  [C#] public static readonly string[] UsrTitles;

- WebServerURL
  The Uri of the Webserver which hosts the WLES application.
  [C#] public static readonly string WebServerURL;

- WLESRoot
  The WLES application root at the hosting Webserver.
  [C#] public static readonly string WLESRoot;
1.3.5 Public Static Methods

- **AddToEventLog**
  Adds a new entry of error event into the system EventLog of the host machine.
  
  [C#]
  ```
  public static string AddToEventLog(string msg);
  ```
  
  Remarks:
  1. This method writes an EventLogEntry into the system EventLog, under the Source name of “WLES” and LogDisplayName of “Application”;
  2. Special permission is needed to update the log, or in other words, to modify the system registry. The ASP.NET application is running in the ASPNET machine account, and by default the system does not grant “write” privilege to this account;
  3. There are two solutions to overcome this restriction. One is to let the ASP.NET account gain full control over the system event log, whereas the other is to let the ASP.NET application run in another account which has proper privileges. Either solution sabotages the system security especially when the application is exposed to the Internet. However the risks are controllable in an Intranet scenario;
  4. Consult your system administrator for more information about system event log, system registry and permissions for machine accounts.

- **ConcatErrorMessage**
  Concatinates two error messages to form one. The error message `msg` is actually appended to the end of the error message `err`. A new line Char is inserted between these two messages.
  
  [C#]
  ```
  public static string ConcatErrorMessage(string err, string msg);
  ```

- **CopyrightMessage**
  Returns an HTML-encoded String that contains the WLES copyright information.
  
  [C#]
  ```
  public static string CopyrightMessage(Page page);
  ```

- **DisplayMessage**
  Formats and displays a particular message on a particular Label WebControl.
  
  [C#]
  ```
  public static void DisplayMessage(Label label, HtmlTableRow row, string mesg, string helpLink, string topLineText);
  ```

- **DisplayMessageAtBottom**
  Displays a “tip” message at the bottom of the displaying area.
  
  [C#]
  ```
  public static void DisplayMessageAtBottom(Page page, Label bottomLabel);
  ```

- **DoGuestLogin**
  Logs the “guest” user in.
  
  [C#]
  ```
  public static void DoGuestLogin(Page page);
  ```
  Remarks: Refer to the “Remarks” section of the DoLogin method documentation for more information on the WLES security mechanism.

- **DoLogin**
  Logs the user in by issuing a cookie with CookieName as its name and an encrypted FormsAuthenticationTicket as its value.
  
  [C#]
  ```
  public static string DoLogin(Page page, string uid, string usrData);
  ```
  Remarks:
  1. The WLES adopts the FormsAuthentication mechanism to authenticate users;
  2. This method itself does NOT authenticate users. The authentication job is done by the IsUserValidated method, which should be called in prior to this method;
  3. In general, a cookie stores a “key=value” pair, with `key` as the cookie name and `value` as the cookie value. (As a matter of fact, a cookie can hold a collection of “key=value” pairs as its subkeys. See Values and HasKeys for more information.)
4. This method goes a step further – it encrypts the value in the following way: instead of storing the uid as plain text in the cookie named CookieName, it embeds the uid and extra usrData in a FormsAuthenticationTicket and assigns the encrypted ticket as the cookie value;

5. Therefore, at a later time, the uid and the usrData can be read by firstly decrypting the encrypted ticket string to restore the ticket, and secondly accessing them via the Name and the UserData properties of the ticket.

- DoLogout
  Overloaded.

  1. Logs the user out by signing out of the FormsAuthentication mechanism and by removing the cookie added upon login.

```
[C#] public static void DoLogout(Page page);
```

**Remarks:**

(a) Refer to the “Remarks” section of the DoLogin method documentation for more information on the WLES security mechanism.

(b) The difference between this method and its 2-parameter overload: This method logs the user out, whereas the overload not only logs the user out but forwards the visit to another address as well.

  2. Logs the user out by signing out of the FormsAuthentication mechanism and by removing the cookie added upon login, and redirects the page to a particular destination.

```
[C#] public static void DoLogout(Page page, string redirectUrl, bool discardOldQueryString);
```

**Remarks:**

(a) Refer to the “Remarks” section of the DoLogin method documentation for more information on the WLES security mechanism.

(b) The difference between this method and its parameterless overload: This method not only logs the user out but forwards the visit to another address as well, whereas the overload only logs the user out.

(c) Refer to Redirect for the destination the user is to be redirected.

  3. Logs the user out by signing out of the FormsAuthentication mechanism and by removing the cookie added upon login, and redirects the page to a particular destination.

```
[C#] public static void DoLogout(Page page, string redirectUrl, string[][] newPairs, bool discardOldQueryString);
```

**Remarks:**

(a) Refer to the “Remarks” section of the DoLogin method documentation for more information on the WLES security mechanism.

(b) The difference between this method and its parameterless overload: This method not only logs the user out but forwards the visit to another address as well, whereas the overload only logs the user out.

(c) Refer to Redirect for the destination the user is to be redirected.

- GenerateCID
  Generates a 13-char “Comment ID” (CID) for comment bookkeeping.

```
[C#] public static string GenerateCID();
```

- GenerateRID
  Generates a 12-char “Run ID” (RID) for data and results bookkeeping.

```
[C#] public static string GenerateRID();
```
• getClientInfo
  Returns an HTML-encoded String that contains the Web client’s username, if already authenticated, an `<img />` tag pointing to its country flag image, and its IP address.

[C#]
public static string GetClientInfo(Page page);

• GetEvalMode
  Overloaded.

  1. Gets the WLES evaluation mode for a particular evaluation number.

[C#]
public static string GetEvalMode(int number);

  2. Gets the WLES evaluation mode that a particular Web Form page is in.

[C#]
public static string GetEvalMode(Page page);

Exceptions:

  condition
  ArgumentException Thrown when `page` is a null reference.

• GetEvalModeNumber
  Overloaded.

  1. Gets the numerical representation of a particular WLES evaluation mode.

[C#]
public static int GetEvalModeNumber(string name);

Exceptions:

  condition
  ArgumentException Thrown when `name` is empty or a null reference.

  2. Gets the numerical representation of the WLES evaluation mode that a particular Web Form page is in.

[C#]
public static int GetEvalModeNumber(Page page);

• GetEvalModes
  Returns all defined evaluation modes as an `string[]` array.

[C#]
public static string[] GetEvalModes();

• GetEvalPhase
  Overloaded.

  1. Gets the name of the evaluation phase of a particular evaluation mode and phase number, expressed as an `Logistics.InternalLink`.

[C#]
public static InternalLink GetEvalPhase(int mode, int phase);

  2. Gets the name of the evaluation phase that a particular Web Form page is in.

[C#]
public static string GetEvalPhase(Page page);

• GetEvalPhaseNumber
  Overloaded.

  1. Gets the evaluation phase number for a particular phase.

[C#]
public static int GetEvalPhaseNumber(string name);

  2. Gets the number of the evaluation phase that a particular Web Form page is in.

[C#]
public static int GetEvalPhaseNumber(Page page);

• GetEvalPhases
  Returns all defined evaluation phases in a particular mode as a `string[]` array.

[C#]
public static string[] GetEvalPhases(string mode);
1.3 Logistics Class

• GetHelpContent
Retrieves the help text for a particular topic. Logistics.InternalLinks and predefined keys embedded in the help content are to be replaced by their values declared in the Web.config file. Predefined keys appear as a String starting with “key” and followed by the name of the key.

[C#] public static string[] GetHelpContent(Page page, string topic);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when topic is empty;</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when (1) no record or (2) more than one records exist(s) for the particular help topic topic.</td>
</tr>
</tbody>
</table>

• GetHelpTopics
Retrieves the help topics from the WLES database.

[C#] public static OleDbDataReader GetHelpTopics(OleDbConnection activeConn);

Remarks:

1. The name and title of the help topic contained in the returned OleDbDataReader can be iterated and values retrieved may be used to populated a DropDownList server control, for instance.

2. Both the activeConn and the OleDbDataReader returned should be closed by the calling code explicitly:

OleDbConnection activeConn = null;
OleDbDataReader reader = null;
try {
    activeConn = new OleDbConnection(...);
    reader = GetHelpTopics(activeConn);
    DropDownList list;
    list.DataSource = reader;
    list.DataTextField = "topic_title";
    list.DataValueField = "topic";
    list.DataBind();
}
finally {
    if (activeConn != null) activeConn.Dispose();
    if (reader != null) reader.Close();
}

• GetInternalLink
Overloaded.

1. Returns the Logistics.InternalLink equivalence of a particular String.

[C#] public static InternalLink GetInternalLink(string link);

2. Returns the Logistics.InternalLink that a particular Web Form page represents.

[C#] public static InternalLink GetInternalLink(Page page);

• GetLastEvalPhase
Overloaded.

1. Gets the last evaluation phase which has been finished for a particular RID.

[C#] public static InternalLink GetLastEvalPhase(string rid);
2. Gets the last evaluation phase which is associated with a particular Web Form page and has been finished.
[C#] public static InternalLink GetLastEvalPhase(Page page);

- GetNameValueCollection
  Overloaded.

  1. Extracts the key-value pairs from a particular query string and returns them as a NameValueCollection.
  [C#] public static NameValueCollection GetNameValueCollection(string queryString);
  2. Expands a particular query string with a collection of new keys and returns all the key-value pairs as a NameValueCollection.
  [C#] public static NameValueCollection GetNameValueCollection(string queryString, string[][] newPairs);
  3. Extracts the key-value pairs from the query string of a particular Web Form and returns them as a NameValueCollection.
  [C#] public static NameValueCollection GetNameValueCollection(Page page);
  4. Expands the query string of a particular Web Form page with a collection of new keys and returns all the key-value pairs as a NameValueCollection.
  [C#] public static NameValueCollection GetNameValueCollection(Page page, string[][] newPairs);

- GetNextEvalPhase
  Gets the evaluation phase which is immediately next in order to the current phase.
  [C#] public static InternalLink GetNextEvalPhase(Page page);

- GetPreviousEvalPhase
  Gets the evaluation phase which is immediately previous in order to the current phase.
  [C#] public static InternalLink GetPreviousEvalPhase(Page page);

- GetQueryString
  Overloaded.

  1. Returns a query string that represents a particular collection of key-value pairs.
  [C#] public static string GetQueryString(NameValueCollection collection);
  2. Returns a query string that represents a particular key-value pair.
  [C#] public static string GetQueryString(string key, string value);
  3. Returns a query string that represents a particular key-value pair.
  [C#] public static string GetQueryString(string key, InternalLink value);
  4. Returns a query string that represents a particular collection of key-value pairs.
  [C#] public static string GetQueryString(string[][] pairs);
  5. Gets the complete query string of a particular WebForm and returns it as a String.
  [C#] public static string GetQueryString(Page page);

  Example: The following example shows explicitly the format of a query string returned by this method, which contains 3 key-value pairs:
  ```csharp
  // get the query string
  string query = UGent.WLES.Logistics.GetQueryString(Page);
  // writes back through the HttpResponse
  Page.Response.Write(query);
  ```
1.3 Logistics Class

The following String is displayed in the user’s browser:

?uid=jhudson&registered=1&favorite=red

6. Returns a query string that represents the complete collection of key-value pairs contained in the query string of a particular WebForm plus a particular new key.

[C#]
public static string GetQueryString(Page page, string newKey, string newValue);

7. Returns a query string that represents the complete collection of key-value pairs contained in the query string of a particular WebForm plus a particular new key.

[C#]
public static string GetQueryString(Page page, string newKey, InternalLink newValue);

8. Returns a query string that represents the complete collection of key-value pairs contained in the query string of a particular WebForm plus a particular collection of new keys.

[C#]
public static string GetQueryString(Page page, string[][] newPairs);

- GetQueryStringKeyValuePairs
  Overloaded.

1. Returns a particular collection (contained in a query string) as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(NameValueCollection collection);

2. Returns all the key-value pairs of a query string as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(string queryString);

3. Adds a new key to the current key-value pair collection contained in a particular query string and returns the expanded collection as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(string queryString, string newKey, string newValue);

4. Adds a new key to the current key-value pair collection contained in a particular query string and returns the expanded collection as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(string queryString, string newKey, InternalLink newValue);

5. Adds new keys to the current key-value pair collection contained in a particular query string and returns the expanded collection as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(string queryString, string[][] newPairs);

6. Gets the complete query string of a particular WebForm and returns it as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(Page page);

7. Adds a new key to the current key-value pair collection represented by the query string of a particular WebForm and returns the expanded collection as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(Page page, string newKey, string newValue);

8. Adds a new key to the current key-value pair collection represented by the query string of a particular WebForm and returns the expanded collection as a jagged String array.

[C#]
public static string[][] GetQueryStringKeyValuePairs(Page page, string newKey, InternalLink newValue);

9. Adds new keys to the current key-value pair collection represented by the query string of a particular WebForm and returns the expanded collection as a jagged String array.
Namespace UGent.WLES

[C#] public static string[][] GetQueryStringKeyValuePairs(Page page, string[][] newPairs);

- GetRefererQueryString
  Gets the QueryString of a particular Web Form's referrer page and returns its String representation.
  
[C#] public static string GetRefererQueryString(Page page);

**Example:** Suppose page A is the application X's entry page written in HTML, and a hyperlink in page A leads the user to page B which is written in ASP.NET:

Application X:
  Entry point: page_a.html
  Referred page: page_b.aspx

The following code in page B prints page A's query string on the lblDisplay Label in page B:

// clear the display and ensure it is visible
lblDisplay.Text = "";
lblDisplay.Visible = true;

// make sure that only page A's query string is printed
if (Request.UrlReferer.LocalPath.EndsWith("page_a.html")) {
    string query = UGent.WLES.Logistics.GetRefererQueryString(
        Page);
    lblDisplay.Text = query;
}

If page A is accessed by a user with this address:

http://www.appx.com/page_a.html?name=jduke&pwd=db

and when the hyperlink in page A is clicked and the page B will display the following:

?name=jduke&pwd=db

- GetSessionTimeoutString
  Gets an encoded String which can be taken as a value in the query string so that the expiring of current session can be judged.

[C#] public static string GetSessionTimeoutString(Page page);

- GetUrlValue
  Overloaded.

1. Retrieves the URL value of a particular url String. The absoluteness of the returned URL depends on the Boolean value of UseAbsoluteUrl.
   
[C#] public static string GetUrlValue(string url);

2. Retrieves the value of a particular hyperlink pointing to a destination within the WLES Web UI. The absoluteness of the returned URL depends on the Boolean value of UseAbsoluteUrl.
   
[C#] public static string GetUrlValue(InternalLink url);

3. Retrieves the value of a particular hyperlink pointing to a destination within the WLES Web UI.
   
[C#] public static string GetUrlValue(InternalLink url, bool absolute);

Remarks:
1.3 Logistics Class

1. A finite collection of most frequently used hyperlinks within the WLES is explicitly
   specified by the Logistics.InternalLink enumeration;

2. Values of the URLs are given in the appSettings section of the application-wide appli-
   cation configuration file Web.config, located at WLESRoot. Appropriate values have to
   be set beforehand or the application does not function properly;

3. To enlarge the hyperlink collection, new links have to be added both to the
   Logistics.InternalLink enumeration and to the Web.config file. In the later case, new
   links are specified as key-value pairs wrapped by opening and closing tags, strictly
   following the XML well-formedness rule;

4. Multiple parts of the same hyperlink can be merged together following the same rule as
   manipulating filesystem pathes. See: ConcatPath.

• GetUserNameAndData

   Overloaded.

   1. Retrieves the username and user data from the user’s HttpRequest to a particular ap-
      plication.
   [C#]
   public static string[] GetUserNameAndData(HttpApplication app);

   2. Retrieves the username and user data from the user’s HttpRequest to a particular Web
      Form page.
   [C#]
   public static string[] GetUserNameAndData(Page page);

   Remarks:
   The client’s request is supposed to be authenticated using the FormsAuthentication mech-
   anism, plus that the username and additional user data are embedded in an encrypted
   FormsAuthenticationTicket, which is transported as the cookie payload. Refer to DoLogin
   for more information.

• HasKey

   Overloaded.

   1. Determines whether a particular key exists in the specified QueryString.
   [C#]
   public static bool HasKey(string queryString, string key, ref string value);

   2. Determines whether a particular key-value pair is defined in the AppSettings file
      Web.config, and passes the value back by ref.
   [C#]
   public static bool HasKey(string key, ref string value);

   Example: The following examples show content fractions of the Web.config file and
   the testing results of the HasKey method.
   Web.config:
   
   <appSettings>
     <add key = "foo" value = "bar"/>
     <add key = "rab" value = ""/>
   </appSettings>

   Testing with the HasKey method:
   
   bool hasKey;
   string key, value = "";
   key = "foo";
   hasKey = HasKey(key, ref value);
namespace UGent.WLES

/* hasKey: true; value: "bar" */

// defined key
key = "rab";
hasKey = HasKey(key, ref value);

/* hasKey: false; value: "" */

// undefined key
key = "whatever";
hasKey = HasKey(key, ref value);

/* hasKey: false; value: "" */

3. Determines whether a particular key exists in the QueryString of the current Page.
[C#] public static bool HasKey(Page page, string key, ref string value);

• HasLink
Determines whether an equivalent Logistics.InternalLink exists for a particular String.
[C#] public static bool HasLink(string link);

• IsAuthenticated
Determines whether a user is authenticated, guest user included.
[C#] public static bool IsAuthenticated(Page page);

• IsEmailLegal
Determines the legitimacy of an email address.
[C#] public static bool IsEmailLegal(string email, ref string err);

• IsEvalModeDefined
Overloaded.

1. Determines whether a particular evaluation mode has been defined.
[C#] public static bool IsEvalModeDefined(int modeNumber);

2. Determines whether a particular evaluation mode has been defined.
[C#] public static bool IsEvalModeDefined(string modeName);

• IsEvalPhaseDefined
Determines whether a particular evaluation phase has been defined.

1. Determines whether a particular evaluation phase has been defined.
[C#] public static bool IsEvalPhaseDefined(int modeNumber, int phaseNumber);

2. Determines whether a particular evaluation phase has been defined.
[C#] public static bool IsEvalPhaseDefined(string modeName, string phaseName);

• IsGuest
Determines whether the guest user is authenticated with a particular Web Form page.
[C#] public static bool IsGuest(Page page);

• IsHostLegal
Determines the legitimacy of a hostname according to the predefined host rule.
[C#] public static bool IsHostLegal(string host, bool fullyQualified, ref string err);
1.3 Logistics Class

- **IsHostValid**
  Determines whether a machine name or its IP address is a valid IPHostEntry by checking it against a DNS server.
  
  ```csharp
  public static bool IsHostValid(string host);
  ```

- **IsInBatchMode**
  Determines whether a particular Page is in the Batch Evaluation Mode.
  
  ```csharp
  public static bool IsInBatchMode(Page page);
  ```

- **IsInInteractiveMode**
  Determines whether a particular Page is in the Interactive Evaluation Mode.
  
  ```csharp
  public static bool IsInInteractiveMode(Page page);
  ```

- **IsLicensed**
  Overloaded.
  1. Determines whether the license code attached to a particular Web Form page is valid.
     
     ```csharp
     public static bool IsLicensed(Page page);
     ```
  2. Determines whether a particular Web Form page is in accordance with a particular WLES License Code.
     
     ```csharp
     public static bool IsLicensed(Page page, string licenseCode);
     ```

- **IsNameLegal**
  Determines the legitimacy of the first or last name of a user according to the predefined rule.
  
  ```csharp
  public static bool IsNameLegal(string name, ref string err);
  ```

  **Remarks:** A legal name must obey the following rules:

  1. Contains letters; and
  2. Special **Chars** represented by **NameRule** which is defined in the application-wide **Web.config** file;
  3. Its length is between the **MinLengthName** and **MaxLengthName** values which are defined in the application-wide **Web.config** file.

- **IsPwdLegal**
  Determines the legitimacy of a user-provided password according to the predefined password rule.
  
  ```csharp
  public static bool IsPwdLegal(string pwd, ref string err);
  ```

- **IsRefererHome**
  Determines whether a particular Web Form’s referrer page is the application root of the WLES.
  
  ```csharp
  public static bool IsRefererHome(Page page);
  ```

- **IsRidAuthenticated**
  Overloaded.
  1. Determines whether the RID, contained in the query string as a **key-value** pair, belongs to the user who is authenticated and associated with a particular Web Form page.
     
     ```csharp
     public static bool IsRidAuthenticated(Page page);
     ```
  2. Determines whether the RID, contained in the query string as a **key-value** pair, belongs to the user who is authenticated and associated with a particular Web Form page.
     
     ```csharp
     public static bool IsRidAuthenticated(Page page, ref string err);
     ```

- **IsSessionExpired**
  Determines whether the current Session has been expired.
  
  ```csharp
  public static bool IsSessionExpired(Page page);
  ```
• IsThisEvalPhaseFinished
  Determines whether the current evaluation phase, which is associated with a particular Web Form page, has already been conducted.
  [C#] public static bool IsThisEvalPhaseFinished(Page page);

• IsUidLegal
  Determines the legitimacy of a user ID according to the predefined UID rule.
  [C#] public static bool IsUidLegal(string uid, ref string err);

• IsUserValidated
  Validates the user by checking the format and the validity of the particular credential.
  [C#] public static bool IsUserValidated(string uid, string pwd, ref string err);

  Remarks: The method passes back (by ref) a piece of error message and returns false if the validation fails, and true if succeeds. The table below summarizes these two situations:

<table>
<thead>
<tr>
<th>Value returned</th>
<th>Message passed back</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>An empty String</td>
</tr>
<tr>
<td>false</td>
<td>An error message, for instance, &quot;Username contains invalid characters.&quot;</td>
</tr>
</tbody>
</table>

• Log
  Overloaded.

  1. Log the user’s visit, with the possibility of including application error in the log.
  [C#] public static int Log(HttpApplication app);

  2. Log the authenticated or guest user’s visit.
  [C#] public static int Log(Page page);

• MakeHelpUrl
  Overloaded.

  1. Makes a url that points to the WLES Help page, with QueryString retained from the current page.
  [C#] public static string MakeHelpUrl(Page page);

  2. Makes a url that points to the WLES Help page, with QueryString retained from the current page and expanded with 2 new keys: "topic" and "back".
  [C#] public static string MakeHelpUrl(Page page, InternalLink topic);

• MakeUrlWithQueryString
  Overloaded.

  1. Makes a new url from an old one and attaches a newly built single-key query string.
  [C#] public static string MakeUrlWithQueryString(string url, string newKey, string newValue);

  Remarks: This method does the following:
  (a) Discard the QueryString attached to the end of the url parameter, if any.
  (b) The url parameter may be any valid hyperlink, relative or absolute, or a member of the WLES Logistics.InternalLink. See also: GetUrlValue.
  (c) A new QueryString is created from the specific key-value pair and appended to the returned url. See also: GetQueryString.

  2. Makes a new url from an old one and attaches the newly built multi-key query string to it.
  [C#] public static string MakeUrlWithQueryString(string url, string[]]
newPairs);

Remarks: This method does the following:

(a) Make a url from the specific url which may be any valid hyperlink, relative or absolute, or a member of the WLES Logistics/InternalLink. See also: GetUrlValue.
(b) Creates a QueryString from the provided key-value pairs and appends it to the end of the returned url. See also: GetQueryString.

3. Makes a new url from an old one, with query string copied from a particular Web Form page.
[C#] public static string MakeUrlWithQueryString(Page page, string url);

Remarks: This method does the following:

(a) Discard the QueryString attached to the end of the url parameter, if any.
(b) The url parameter may be any valid hyperlink, relative or absolute, or a member of the WLES Logistics/InternalLink. See also: GetUrlValue.
(c) Copy and attach the QueryString of the current Page to the returned url. See also: GetQueryString.

4. Makes a new url from an old one, with query string copied from a particular Web Form page.
[C#] public static string MakeUrlWithQueryString(Page page, string url);

Remarks: This method does the following:

(a) Discard the QueryString attached to the end of the url parameter, if any.
(b) The url parameter may be any valid hyperlink, relative or absolute, or a member of the WLES Logistics/InternalLink. See also: GetUrlValue.
(c) Copy and attach the QueryString of the current Page to the returned url. See also: GetQueryString.

5. Makes a new url from an old one, with query string copied from a particular Web Form page and merged with specific new key-value pairs.
[C#] public static string MakeUrlWithQueryString(Page page, string url, string[] [] newPairs);

Remarks: This method does the following:

(a) Discard the QueryString attached to the end of the url parameter, if any.
(b) The url parameter may be any valid hyperlink, relative or absolute, or a member of the WLES Logistics/InternalLink. See also: GetUrlValue.
(c) A new QueryString is created from the specific key-value pair and appended to the returned url. See also: GetQueryString.

6. Makes a new url from an existing Logistics/InternalLink, with query string copied from the current Web Form page.
[C#] public static string MakeUrlWithQueryString(Page page, InternalLink link);

Remarks: This method does the following:

(a) Make a url from the link parameter. See also: GetUrlValue.
(b) Copies the QueryString of page and appends it to the end of the returned url. See also: GetQueryString.

7. Makes a new url from a particular Logistics/InternalLink, with query string copied from the current Web Form page and merged with a specific new key-value pair.
[C#] public static string MakeUrlWithQueryString(Page page, InternalLink link, string newKey, string newValue);

Remarks: This method does the following:

(a) A url is made from the link parameter. See also: GetUrlValue.
(b) A new key-value pair is added to the existing NameValueCollection and the latter is then appended to the returned url as a QueryString. See also: GetQueryString.
8. Makes a new url from a particular Logistics.InternalLink, with query string copied from the current Web Form page and merged with a specific new key-value pair.

```
[C#] public static string MakeUrlWithQueryString(Page page, InternalLink link, string newKey, InternalLink newValue);
```

Remarks: This method does the following:
(a) A url is made from the link parameter. See also: GetUrlValue.
(b) A new key-value pair is added to the existing NameValueCollection and the latter is then appended to the returned url as a QueryString. See also: GetQueryString.

9. Makes a new url from an existing Logistics.InternalLink, with the query string copied from a particular Web Form page and merged with specific new key-value pairs.

```
[C#] public static string MakeUrlWithQueryString(Page page, InternalLink link, string[] newPairs);
```

Remarks: This method does the following:
(a) Discard the QueryString attached to the end of the url parameter, if any.
(b) Make a url from the provided Logistics.InternalLink. See: GetUrlValue.
(c) Merge the page’s QueryString with the new key-value pairs and append the new QueryString to the returned url. See also: GetQueryString

10. Makes a new url from an existing Logistics.InternalLink and attaches a newly built single-key query string.

```
[C#] public static string MakeUrlWithQueryString(InternalLink link, string newKey, string newValue);
```

Remarks: This method does the following:
(a) Make a url from the link parameter. See also: GetUrlValue.
(b) Creates a QueryString from the provided key-value pair and appends it to the end of the returned url. See also: GetQueryString.

11. Makes a new url from an existing Logistics.InternalLink and attaches a newly built single-key query string.

```
[C#] public static string MakeUrlWithQueryString(InternalLink link, string newKey, InternalLink newValue);
```

Remarks: This method does the following:
(a) Make a url from the link parameter. See also: GetUrlValue.
(b) Creates a QueryString from the provided key-value pair and appends it to the end of the returned url. See also: GetQueryString.

12. Makes a new url from an existing Logistics.InternalLink and attaches a newly built multi-key query string.

```
[C#] public static string MakeUrlWithQueryString(InternalLink link, string[] newPairs);
```

Remarks: This method does the following:
(a) Make a url from the link parameter. See also: GetUrlValue.
(b) Creates a QueryString from the provided key-value pair and appends it to the end of the returned url. See also: GetQueryString.

- PopulateCountries
  Overloaded.

1. Populates a particular DropDownList WebControl which contains names of countries of the world. The SelectedIndex is defaulted to 0, with initial display as text “- Please choose one -”.

```
[C#] public static void PopulateCountries(DropDownList list);
```
2. Populates a particular DropDownList WebControl which contains names of countries of the world, with a particular initial display text. The SelectedIndex is defaulted to 0.

```csharp
public static void PopulateCountries(DropDownList list, string display);
```

3. Populates a particular DropDownList WebControl which contains names of countries of the world, with a particular initial display text and the current selected value.

```csharp
public static void PopulateCountries(DropDownList list, string display, string selectedValue);
```

- ProcessPostalAddress
  Rewrites a postal address from the multi-line format to the single-line format, or vice versa.

```csharp
public static string ProcessPostalAddress(string address, bool multi2one);
```

- Redirect
  Overloaded.

1. Redirects the user to a particular destination with the query string being retained.

```csharp
public static void Redirect(Page page, string url);
```

Remarks:

(a) This method redirects the user to a particular destination and passes the query string to the destination url.

(b) The destination url may take one of the following values: (1) A member of Logistics.InternalLink, casted to a lower-case String; (2) A real URL, either relative, to the Webserver root, or absolute.

2. Redirects the user to a particular destination with the query string being retained and expanded (or updated) with a new key.

```csharp
public static void Redirect(Page page, string url, string newKey, string newValue);
```

Remarks:

(a) This method adds the new key into the current collection represented by the query string, and redirects the user to a specific destination.

(b) If the new key collides with an existing one, the existing value will be replaced by the new value.

(c) The destination url may take one of the following values: (1) A member of Logistics.InternalLink, casted to a lower-case String; A real URL, either relative, to the Webserver root, or absolute.

3. Redirects the user to a particular destination with the query string being retained and expanded (or updated) with a new key.

```csharp
public static void Redirect(Page page, string url, string newKey, InternalLink newValue);
```

Remarks:

(a) This method adds the new key into the current collection represented by the query string, and redirects the user to a specific destination.

(b) If the new key collides with an existing one, the existing value will be replaced by the new value.

(c) The destination url may take one of the following values: (1) A member of Logistics.InternalLink, casted to a lower-case String; A real URL, either relative, to the Webserver root, or absolute.

4. Redirects the user to a particular destination with the query string being retained and expanded (or updated) with new keys.

```csharp
public static void Redirect(Page page, string url, string[] []
```
newPairs);
Remarks:
(a) This method adds new keys into the current collection represented by the query string, and redirects the user to a specific destination.
(b) If one of the new keys collides with an existing one, the existing value will be replaced by the new value.
(c) The destination url may take one of the following values: (1) A member of Logistics.InternalLink, casted to a lower-case String; A real URL, either relative, to the Webserver root, or absolute.

5. Redirects the user to a particular destination with or without the query string being retained and expanded (or updated) with new keys.
[C#] public static void Redirect(Page page, string url, string[] newPairs, bool discardPageQueryString);

6. Redirects the user to a particular destination with the query string being retained.
[C#] public static void Redirect(Page page, InternalLink link);
Remarks:
(a) This method redirects the user to a particular destination and passes the query string to the destination url.
(b) The destination link is represented by a member of the WLES Logistics.InternalLink.

7. Redirects the user to a particular destination with the query string being retained and expanded (or updated) with a new key.
[C#] public static void Redirect(Page page, InternalLink link, string newKey, string newValue);
Remarks:
(a) This method adds the new key into the current collection represented by the query string, and redirects the user to a specific destination.
(b) If the new key collides with an existing one, the existing value will be replaced by the new value.
(c) The destination link is represented by a member of the WLES Logistics.InternalLink.

8. Redirects the user to a particular destination with the query string being retained and expanded (or updated) with a new key.
[C#] public static void Redirect(Page page, InternalLink link, string newKey, InternalLink newValue);
Remarks:
(a) This method adds the new key into the current collection represented by the query string, and redirects the user to a specific destination.
(b) If the new key collides with an existing one, the existing value will be replaced by the new value.
(c) The destination link is represented by a member of the WLES Logistics.InternalLink.

9. Redirects the user to a particular destination with the query string being retained and expanded (or updated) with new keys.
[C#] public static void Redirect(Page page, InternalLink link, string[] newPairs);
Remarks:
(a) This method adds new keys into the current collection represented by the query string, and redirects the user to a specific destination.
(b) If one of the new keys collides with an existing one, the existing value will be replaced by the new value.
1.3 Logistics Class

(c) The destination link is represented by a member of the WLES Logistics.InternalLink.

- RedirectToNextEvalPhase
  Overloaded.

  1. Redirects the user to the next evaluation phase, with the session timeout string being renewed and the query string retained.
  
  ```csharp
  public static void RedirectToNextEvalPhase(Page page);
  Remarks:
  (a) This method redirects the user to the evaluation phase which is immediately next to the current phase in the current evaluation mode;
  (b) The “timeout” key in the current query string is renewed (by extending the Session timeout value) and the resulting query string is then passed to the next evaluation phase.
  
  2. Redirects the user to the next evaluation phase, with the session timeout string being renewed and the query string retained and updated.
  
  ```csharp
  public static void RedirectToNextEvalPhase(Page page, string newKey, string newValue);
  Remarks:
  (a) This method redirects the user to the evaluation phase which is immediately next to the current phase in the current evaluation mode;
  (b) The “timeout” key in the current query string is renewed (by extending the Session timeout value);
  (c) The newKey is added to (if not already existing) or updated in (if existing) the current query string; The resulting query string is then passed to the next evaluation phase.

- RemoveKeys
  Overloaded.

  1. Removes the specified key from a particular QueryString.
  ```csharp
  public static string RemoveKeys(string queryString, string key);
  
  2. Removes specified keys from a particular QueryString.
  ```csharp
  public static string RemoveKeys(string queryString, string[] keys);
  
  3. Removes the specified key from the query string of a particular Web Form page.
  ```csharp
  public static string RemoveKeys(Page page, string key);
  
  4. Removes specified keys from the QueryString of a particular Web Form page.
  ```csharp
  public static string RemoveKeys(Page page, string[] keys);

- RenameKeys
  Renames a key in a particular query string and return.
  ```csharp
  public static string RenameKeys(string queryString, string key1, string key2);
  
- ReplaceKeys
  Searches a particular String for all words starting with substring “key”, and replaces them with corresponding values defined in ConfigurationSettings.AppSettings.
  ```csharp
  public static string ReplaceKeys(string content);
  
  Example: Suppose the following String, stored in a database table, is to be used as part of the help texts of Application X. But before send it as an HTML response, the keys embedded should be replaced with their runtime values.
A password consists of letters, digits, and one or more of the following special characters: `<font face="\"Courier New\">"keyPwdRule"</font> (excluding the quotation mark). Password length is limited to between `keyMinLengthPwd` and `keyMaxLengthPwd` characters inclusive.

All defined keys are prefixed with the lowercase `String` of "key". Solution:

- Step 1 - Edit the Web.config file to ensure the following lines are included in the `appSettings` section:
  ```xml
  <add key = "PwdRule" value = ".^~#$_" />
  <add key = "MinLengthPwd" value = "6" />
  <add key = "MaxLengthPwd" value = "20" />
  </appSettings>
  ```

- Step 2 - The following method call would do the job:
  ```csharp
  string mesg = @"A password consists of letters, digits, " +
  "and one or more of the following special characters: " +
  "<font face="\"Courier New\">"keyPwdRule"</font> " +
  "(excluding the quotation mark). Password length " +
  "is limited to between keyMinLengthPwd and " +
  "keyMaxLengthPwd characters inclusive.";
  string mesg2 = UGent.WLES.Logistics.ReplaceKeys(mesg);
  ```

  ```csharp
  /*
  * Content of mesg2:
  * A password consists of letters, digits, and one or
  * more of the following special characters:
  * <font face="\"Courier New\">".^~#$_"</font> (excluding
  * the quotation mark). Password length is limited to
  * between 6 and 20 characters inclusive.
  */
  ```

- ReplaceUrls
  Searches a particular `String` for all occurrences of any `Logistics.InternalLink`, embedded within HTML hyperlink tags, and replaces them with corresponding values defined in the `ConfigurationSettings.AppSettings`, with query string retained.

  ```csharp
  public static string ReplaceUrls(Page page, string content);
  ```

Remarks:

1. The `Logistics.InternalLink` enumeration defines hyperlink names, which are also called nominal hyperlinks. The predefined values in the application-wide `Web.config` file are real hyperlinks. The nominal and the real hyperlinks form the `key-value` pair;
2. What this method does is to find any nominal hyperlinks from a `String` and replace them with real hyperlinks;
3. The real hyperlink of a nominal hyperlink can be returned by the `GetUrlValue` method.

**Example**: Suppose the following `String`, stored in a database table, is to be used as part of the help texts of Application X. But before send it as an HTML response, the hyperlinks embedded should be replaced with real, accessible addresses.

You may `<a href="LoginURL">login</a>` to `<a href="HomeURL">App-X</a>` by filling in your username and password in corresponding textboxes and click the "Login" button.
1.3 Logistics Class

Solution:
- Step 1 - Edit the Web.config file to ensure the following lines are included in the appSettings section:

```xml
<appSettings>
  <add key = "AppXRoot" value = "/AppX/" />
  <add key = "LoginURL" value = "/login/" />
  <add key = "HomeURL" value = "/" />
</appSettings>

- Step 2 - The following method call would do the job:

```csharp
string mesg = @"You may <a href="/AppX/login/">login</a> " + 
  "to <a href="/App-X/">App-X</a> by filling in your " + 
  "username and password in corresponding textboxes " + 
  "and click the "Login" button.";
string mesg2 = UGent.WLES.Logistics.ReplaceUrls(mesg);

*/
* Content of mesg2:
* You may <a href="/AppX/login/">login</a> to <a href="/App-X/">App-X</a> by filling in your username and 
* password in corresponding textboxes and click the 
* "Login" button.
*/
```

- RetrieveEffectiveCalendar
  Retrieves the user-selected RomanDate.CalendarType associated with the current Web Form page.

[C#]
public static CalendarType RetrieveEffectiveCalendar(Page page);

Remarks:
1. The effective calendar is used to specify the number of data entries in a DAILY data item;
2. It is essential whenever a RomanDate is in concern, for example in determining the GrowingPeriod and the CropCycle;
3. This method is simply a delegate of the RetrieveCalendarType method.

- RetrieveRID
  Retrieves the RID value from the query string of a particular Web Form page.

[C#]
public static string RetrieveRID(Page page);

- SendMail
  Overloaded.

1. Sends an email message via the default SMTP server at the Normal priority.

[C#]
public static string SendMail(string from, string to, string subject, string body, bool withDisclaimer);
2. Sends an email message via the default SMTP server.

[C#]
public static string SendMail(string from, string to, string subject, string body, bool withDisclaimer, MailPriority priority);
3. Sends an email message via a particular SMTP server.

[C#]
public static string SendMail(string smtp, string from, string to, string subject, string body, bool withDisclaimer, MailPriority priority);
4. Sends an email message via a particular SMTP server and encoding the message body in particular colors.

[C#] public static string SendMail(string smtp, string from, string to, string subject, string body, string titleColor, string bodyColor, bool withDisclaimer, MailPriority priority);

- SetKeys
  Overloaded.

  1. Sets the value of a key in a particular query string.

  [C#] public static string SetKeys(string queryString, string key, string value);

  2. Sets the values of keys in a particular query string.

  [C#] public static string SetKeys(string queryString, string[] [] pairs);

Remarks: The action SET has two meanings: replace and add. To set an existing key is to replace its old value with the new one; To set an non-existing key is simply to add a new key-value pair into the query string.

- ShowAuthUid
  Returns an HTML-encoded String that contains the Username of the authenticated Web client; if the authenticated user is not the guest user, an hyperlink is embedded in the Username, pointing to the user’s profile report page.

[C#] public static string ShowAuthUid(Page page);

- UpdateRunningInfo
  Saves the evaluation progress in the WLES database.

[C#] public static void UpdateRunningInfo(Page page);

Remarks: This method updates the evaluation phase that a particular RID has reached. It also updates the timestamp at which the updation happens.

1.4 WlesDatabaseException Class

1.4.1 Summary

Represents database errors which are caused by the WLES application.

1.4.2 Declaration

[C#] public class WlesDatabaseException : Exception

1.4.3 Public Instance Constructors

- public WlesDatabaseException(string message);
  Initializes a new instance of the WlesDatabaseException class.

- public WlesDatabaseException(string message, Exception e);
  Initializes a new instance of the WlesDatabaseException class.

1.5 RomanDate Structure

1.5.1 Summary

Represents a date by specifying the month, the day and the calendar.
1.5.2 Declaration

[C#] public struct RomanDate

1.5.3 Remarks

The RomanDate structure treats every year as the same. In other words, it does not take leap years into account. However, it does distinguish the GregorianCalendar from from WlesCalendar.

In most of its constructors and methods that have a RomanDate as an argument, two versions are implemented: one takes the RomanDate.CalendarType as an argument, the other does not. The general rule is that whenever a RomanDate is involved (for example, in defining a GrowingPeriod or a CropCycle, or in calculating the distance or length from one date to the other), the RomanDate.CalendarType must be known. If it is not explicitly specified, the DefaultCalendar is then applied.

The WlesCalendar is set as the DefaultCalendar in the WLES application.

1.5.4 Examples

The following example shows the way of creating RomanDate instances using one of its constructors, and of measuring the distance, and the length, between two dates. Results are formulated using the ToString instance method.

```csharp
using System;
using UGent.WLES;
using UGent.WLES.Data;

public class RomanDateExample {
    public static void Main(string [] args) {
        RomanDate [] datesFrom = new RomanDate [2],
                      datesTo = new RomanDate[2];
        datesFrom[0] = new RomanDate((RomanDate.MonthOfYear)3, 22);
        datesFrom[1] = new RomanDate(82,
                                    RomanDate.CalendarType.GregorianCalendar);
        datesTo[0] = new RomanDate((RomanDate.MonthOfYear)11, 2,
                                    RomanDate.CalendarType.GregorianCalendar);
        datesTo[1] = new RomanDate(302);

        for (int i=0; i<datesFrom.Length; i++) {
            for (int j=0; j<datesTo.Length; j++) {
                int distGo, distBack, lenGo, lenBack;
                distGo = datesFrom[i].Distance(datesTo[j]);
                distBack = datesTo[j].Distance(datesFrom[i]);
                lenGo = RomanDate.GetLength(datesFrom[i], datesTo[j]);
                lenBack = RomanDate.GetLength(datesTo[j], datesFrom[i]);
                Console.WriteLine("{0} to {1}: \n distance = {2};
                        length = {3}\n", 
                        datesFrom[i], datesTo[j],
                        ((distGo == -1) ? "not computable" : 
                         (distGo.ToString() + " days")),
                        ((lenGo == -1) ? "not computable" : 
                         (lenGo.ToString() + " days"));
                Console.WriteLine("{0} to {1}: \n distance = {2};
                        length = {3}\n", 
                        datesTo[j], datesFrom[i],
                        ((distBack == -1) ? "not computable" : 
                         (distBack.ToString() + " days")),
                        ((lenBack == -1) ? "not computable" : 
                         (lenBack.ToString() + " days");
            }
        }
    }
}
```
((distBack.ToString() + " days")))
((lenBack == -1) ? "not computable" : (lenBack.ToString() + " days"));
}
}
}

The example will generate the following results:

Mar 22 (WLES Calendar) to Nov 02 (Gregorian Calendar):
distance = not computable; length = not computable

Nov 02 (Gregorian Calendar) to Mar 22 (WLES Calendar):
distance = not computable; length = not computable

Mar 22 (WLES Calendar) to Nov 02 (WLES Calendar):
distance = 220 days; length = 221 days

Nov 02 (WLES Calendar) to Mar 22 (WLES Calendar):
distance = 140 days; length = 141 days

Mar 23 (Gregorian Calendar) to Nov 02 (Gregorian Calendar):
distance = 224 days; length = 225 days

Nov 02 (Gregorian Calendar) to Mar 23 (Gregorian Calendar):
distance = 141 days; length = 142 days

Mar 23 (Gregorian Calendar) to Nov 02 (WLES Calendar):
distance = not computable; length = not computable

Nov 02 (WLES Calendar) to Mar 23 (Gregorian Calendar):
distance = not computable; length = not computable

1.5.5 Public Static Fields

- DefaultCalendar
  Specifies the default RomanDate.CalendarType.

[C#] public static readonly CalendarType DefaultCalendar;

1.5.6 Public Static Methods

- CountDays
  Overloaded. Gets the number of days passed since the beginning of the year till the end of a particular decade.

[C#] public static int CountDays(int decade, CalendarType calendar);

- CountDecades
  Overloaded. Counts the number of decades for a particular day since the beginning of the year.

[C#] public static int CountDecades(int days, CalendarType calendar);

- GetDate
  Overloaded.

  1. Gets a date that is the days\textsuperscript{th} day since the year beginning (if days > 0) or the year end (if days < 0) in the calendar RomanDate.CalendarType.

  [C#] public static RomanDate GetDate(int days);

  2. Gets a date that is the days\textsuperscript{th} day since the year beginning (if days > 0) or the year end (if days < 0) in the calendar RomanDate.CalendarType.

  [C#] public static RomanDate GetDate(int days, CalendarType calendar);
1.5 RomanDate Structure

- GetDaysPerDecade
  Gets a 36-element int array; the value of the element represents the number of days per decade.
  
  [C#] public static int[] GetDaysPerDecade(CalendarType calendar);

- GetDaysPerMonth
  Gets a 12-element int array; the value of the element represents the number of days per month, Jan. through Dec.
  
  [C#] public static int[] GetDaysPerMonth(CalendarType calendar);

- GetLength
  Gets the length (i.e., number of days) of a period that is defined by a beginning and an ending.
  
  [C#] public static int GetLength(RomanDate start, RomanDate end);

- GetLengthOfDecade
  Gets the number of days in a particular decade of a particular type of year.
  
  [C#] public static int GetLengthOfDecade(int decade, CalendarType calendar);

- GetLengthOfMonth
  Overloaded.
  1. Gets the number of days in a particular month of a particular type of year.
     
     [C#] public static int GetLengthOfMonth(int month, CalendarType calendar);
  2. Gets the number of days in a particular month of a particular type of year.
     
     [C#] public static int GetLengthOfMonth(MonthOfYear month, CalendarType calendar);

- GetLengthOfYear
  Gets the length (i.e., number of days) of a year in a particular CalendarType.
  
  [C#] public static int GetLengthOfYear(CalendarType calendar);

- GetMonthDay
  Overloaded.
  1. Gets the month and day of a date in the RomanDate.CalendarType calendar.
     
     [C#] public static int[] GetMonthDay(int days);
  2. Gets the month and day of a date in the RomanDate.CalendarType calendar.
     
     [C#] public static int[] GetMonthDay(int days, CalendarType calendar);

- GetMonthDayString
  Overloaded. Gets a string representation of a particular RomanDate instance: ‘Mon day’, where Mon is a 3-char Month string, and day is a 2-digit Day string.
  
  [C#] public static string GetMonthDayString(RomanDate date);

- GetMonthOfYear
  Returns the RomanDate.MonthOfYear equivalence of the string representation of a particular month.
  
  [C#] public static MonthOfYear GetMonthOfYear(string m);

1.5.7 Public Static Operators

- Addition (+)
  Adds a particular number of days to a particular RomanDate.
  
  [C#] public static RomanDate operator +(RomanDate date, int days)
• Decrement (---)
  Decrements a particular RomanDate with one day.
[C#] public static RomanDate operator --(RomanDate date)

• Greater Than (>)
  Determines whether the value of one RomanDate is greater than that of the other.
[C#] public static bool operator >(RomanDate date1, RomanDate date2)

• Greater Than Or Equal (>=)
  Determines whether the value of one RomanDate is greater than or equal to that of the other.
[C#] public static bool operator >=(RomanDate date1, RomanDate date2)

• Increment (++)
  Increments a particular RomanDate with one day.
[C#] public static RomanDate operator ++(RomanDate date)

• Less Than (<)
  Determines whether the value of one RomanDate is smaller than that of the other.
[C#] public static bool operator <(RomanDate date1, RomanDate date2)

• Less Than Or Equal (<=)
  Determines whether the value of one RomanDate is smaller than or equal to that of the other.
[C#] public static bool operator <=(RomanDate date1, RomanDate date2)

• Subtraction (-)
  Subtracts a particular number of days from a particular RomanDate.
[C#] public static RomanDate operator -(RomanDate date, int days)

1.5.8 Public Instance Constructors

• public RomanDate(int);
  Initializes a new instance of the RomanDate structure to a particular number of days since
  the beginning of the year in the DefaultCalendar.

• public RomanDate(int,int);
  Initializes a new instance of the RomanDate structure to a particular month and day in the
  DefaultCalendar.

• public RomanDate(int,int,CalendarType);
  Initializes a new instance of the RomanDate structure to a particular month and day of a
  particular RomanDate.CalendarType.

• public RomanDate(int,CalendarType);
  Initializes a new instance of the RomanDate structure to a particular number of days since
  the beginning of the year in a particular RomanDate.CalendarType.

• public RomanDate(RomanDate);
  Initializes a new instance of RomanDate to a particular instance.

• public RomanDate(RomanDate,int);
  Initializes a new instance of RomanDate that is a particular number of days after a particular
  date in the DefaultCalendar.

• public RomanDate(MonthOfYear,int);
  Initializes a new instance of the RomanDate structure to a particular month and day in the
  DefaultCalendar.

• public RomanDate(MonthOfYear,int,CalendarType);
  Initializes a new instance of the RomanDate structure to a particular month and day in a
  particular RomanDate.CalendarType.
1.5.9 Public Instance Properties

- Calendar
  Gets the type of calendar that is in effect.
  
  ```csharp
  public CalendarType Calendar { get; }
  ```

- Day
  Gets the day of the month.
  
  ```csharp
  public int Day { get; }
  ```

- Month
  Gets the month.
  
  ```csharp
  public MonthOfYear Month { get; }
  ```

1.5.10 Public Instance Methods

- ChangeCalendar
  Overloaded.
  
  1. Changes the calendar type of this `RomanDate` instance.
  
  ```csharp
  public RomanDate ChangeCalendar();
  ```

  2. Changes the calendar type of this `RomanDate` instance.
  
  ```csharp
  public RomanDate ChangeCalendar(CalendarType target);
  ```

- CountDays
  Overloaded. Gets the number of days passed since the beginning of the year.
  
  ```csharp
  public int CountDays();
  ```

- CountDecades
  Gets the number of decades passed since the beginning of the year.
  
  ```csharp
  public int CountDecades();
  ```

- CountMonths
  Gets the number of months since the beginning of the year.
  
  ```csharp
  public int CountMonths();
  ```

- Distance
  Counts the number of days between this instance and the instance passed in as the formal parameter `end`.
  
  ```csharp
  public int Distance(RomanDate end);
  ```

- Equals
  Overloaded.
  
  1. Determines whether a particular `RomanDate` has the same value as that of this instance.
  
  ```csharp
  public bool Equals(RomanDate date);
  ```

  2. Determines whether a particular `RomanDate` has the same value as that of this instance.
  
  ```csharp
  public bool Equals(RomanDate date, bool ignoreCalendar);
  ```

- GetMonthDayString
  Overloaded. Gets a `string` representation of the current `RomanDate` in this format: ‘Mon day’, where Mon is a 3-char Month string, and day is a 2-digit Day string.
  
  ```csharp
  public string GetMonthDayString();
  ```

- IsInPeriod
  Determines whether this `RomanDate` is in a particular period.
  
  ```csharp
  public bool IsInPeriod(RomanDate start, RomanDate end);
  ```
Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the RomanDate.CalendarType of start is different from that of end, or the RomanDate.CalendarType of this is different from that of start or end.</td>
</tr>
</tbody>
</table>

- Length
  Gets the length (i.e., number of days) of the period that starts on this date and ends on another.
  ```csharp
  public int Length(RomanDate end);
  ```

- Next
  Gets a new instance of RomanDate that represents the day which is immediately after this date.
  ```csharp
  public RomanDate Next();
  ```

- Previous
  Gets a new instance of RomanDate that represents the day which is immediately before this date.
  ```csharp
  public RomanDate Previous();
  ```

- ToDateTime
  Returns the DateTime equivalence of the current RomanDate instance.
  ```csharp
  public DateTime ToDateTime();
  ```

- ToString
  Converts the value of this instance to its equivalent string representation.
  ```csharp
  public override string ToString();
  ```

1.6 Utilities Structure

1.6.1 Summary

Provides common functionalities to other classes and structs. It differentiates from the Logistics class in that Logistics mainly fulfills the needs of the Web UI-related classes and structs, whereas Utilities satisfies more fundamental requirements in building a robustic class library.

1.6.2 Remarks

There are three groups of functionalities are implemented in Utilities, namely, judgement, transformation and others.

1. Judgement: The method checks the parameter(s) against a particular condition and returns true or false. Methods in this group:

   - Exists: Overloaded. Determines whether an integer exists in an array;
   - IsDigits: Determines whether a string consists only of decimal digits (0-9);
   - IsDigitsOrLetters: Determines whether a string is a combination of decimal digits (0-9) and letters (a-z and/or A-Z);
   - IsEmptyString: Determines whether a String is empty or a null reference;
   - IsInAscendingOrder: Overloaded. Determines whether the array elements are in ascending order;
   - IsInDatePeriod: Overloaded. Determines whether a particular date is inside a date period;
1.6 Utilities Structure

- IsInDescendingOrder: Overloaded. Determines whether the array elements are in descending order;
- IsInValueRange: Overloaded. Determines whether a particular value is inside a value range (boundary inclusive);
- IsLetters: Determines whether a string consists only of letters (a-z and/or A-Z);
- IsValidIP: Determines whether a string representation of an Internet Protocol (IP) address is well formed.

2. Transformation: The method transforms the Encoding, base, format, etc. of data from one to another and returns the same Type as that of the parameter (or one of parameters) in the target Encoding, base, format, etc.

- ChangeEncoding: Changes the Encoding of a string;
- Dec2Hex: Converts a decimal integer number to its hexadecimal equivalent;
- Degree2Radian: Converts degrees to radians;
- Hex2Dec: Converts a hexadecimal integer number to its decimal equivalent;
- Radian2Degree: Converts radians to degrees;
- ToSentenceForm: Transforms a string into the sentence form, i.e., the leading Char is in upper case and all the rest in lower case;
- ToTitleForm: Transforms a string into the title form, i.e., the leading Char of each word is in uppercase and all the rest in lower case.

3. Helpers: The methods in this group have various functions ranging from finding the position or neighbors within an array, to traverse a directory; from calculating a weighted average to exporting data for web download; and so on.

- Export4Download: Exports text data and make available for download;
- FindNeighbours: Overloaded. Finds elements from a Double array. The found elements should be the closest in magnitude to a particular checker;
- FindPosition: Overloaded. Finds the checker position in the members array;
- GetDateTime: Obtains a String representation of the current date and time on the local machine, in the form of “yyyy-MM-dd HH:mm:ss”;
- RandomList: Overloaded. Generates a list of random values that does or does not allow duplicates;
- Sum: Overloaded. Computes the sum of a particular data array;
- TraverseDirectory: Overloaded. Traverses a (sub-)directory tree to search for a particular group of files;
- WeightedAverage: Overloaded. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.

1.6.3 Public Static Fields

- SmallWords
  Represents an (in)complete list of small words whose initial letter is not changed to uppercase when used in a title.

[C#]
public static readonly string[] SmallWords;
The following small words are represented:

- Conjunction: “and”, “or”, “but”, “nor”, “for”, “so”, “yet”, etc.
- Article: “a”, “an”, “the”, etc.
- Preposition: “about”, “on”, “at”, “in”, “out”, “from”, “to”, “before”, “after”, etc.
1.6.4 Public Static Methods

- **ChangeEncoding**
  Changes the Encoding of a *String*.
  [C#] public static string ChangeEncoding(string source, Encoding from, Encoding to);

- **ConcatPath**
  Concatenates two path *Strings* with a particular delimiting *Char*.
  [C#] public static string ConcatPath(string path1, string path2, char delimiter);
  Exceptions:
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
  | ArgumentException | Thrown when *path1* is empty *String*, or when *delimiter* is not one of these two: the slash ('/') and the back-slash ('\') *Chars*.

- **Dec2Hex**
  Converts a decimal integer number to its hexadecimal equivalent.
  [C#] public static string Dec2Hex(string dec);

- **Degree2DMS**
  Converts the expression of an angle from the decimal degrees to its degree-minute-second components equivalence.
  [C#] public static int[] Degree2DMS(double degrees);
  Exceptions:
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
  | ArgumentException | Thrown when *degree* is negative.

- **Degree2Radian**
  Converts degrees to radians.
  [C#] public static double Degree2Radian(double degrees);

- **Degree2Second**
  Converts the expression of an angle from the decimal degree to the whole number of seconds.
  [C#] public static int Degree2Second(double degrees);
  Exceptions:
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
  | ArgumentException | Thrown when *degree* is negative.

- **DMS2Degree**
  Converts the expression of an angle from the degree-minute-second components to the equivalent decimal degrees.
  [C#] public static double DMS2Degree(int dd, int mm, int ss);
  Exceptions:
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
  | ArgumentException | Thrown when one of *dd*, *mm* and *dd* is negative.

- **DMS2Second**
  Converts the expression of an angle from its degree-minute-second components to the whole number of seconds.
  [C#] public static int DMS2Second(int dd, int mm, int ss);
  Exceptions:
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
  | ArgumentException | Thrown when one of *dd*, *mm* and *dd* is negative.

- **DrawCurves**
  Overloaded.
1. Draws curves in different Colors and widths on a Bitmap image.
[C#] public static Bitmap DrawCurves(PointF[][] points, Color bgColor, Color[] colors, float[] widths, int w, int h, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor);

2. Draws curves (or lines) in different Colors and widths on a Bitmap image.
[C#] public static Bitmap DrawCurves(PointF[][] points, Color bgColor, Color[] colors, float[] widths, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, Color[] fillColors);

3. Draws curves (or lines) in different Colors and widths on a Bitmap image. Such curves (or lines) can be closed areas, each filled with a particular Color.
[C#] public static Bitmap DrawCurves(PointF[][] points, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, bool[] isClosed, Color[] fillColors);

4. Draws curves in different Colors and widths on a 800 x 600 Bitmap image with WhiteSmoke background and DarkGray grid lines.
[C#] public static Bitmap DrawCurves(PointF[][] points, Color[] colors, float[] widths);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when points, colors and widths do not have the same Length.</td>
</tr>
</tbody>
</table>

- Exists
  Overloaded.

1. Determines whether a Double exists in an array.
[C#] public static bool Exists(double d, double[] values);

2. Determines whether an integer exists in an array.
[C#] public static bool Exists(int number, int[] values);

3. Determines whether a String exists in an array.
[C#] public static bool Exists(string s, string[] values);

- Export4Download
  Exports text data and make available for download.
[C#] public static void Export4Download(Page page, string data);

- ExtractData
  Extracts a subset of data from a data source.
[C#] public static double[] ExtractData(double[] source, int[] lenSubSets, int ndx);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when length of source array not equal to the sum of the lengths of all subsets; or when the ndx is out of range.</td>
</tr>
</tbody>
</table>

- ExtractDigits
  Extracts segments of digits in a String.
[C#] public static double[] ExtractDigits(string s);
Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the members array is a null reference or empty, and when the elements of the members array is not in ascending order.</td>
</tr>
</tbody>
</table>

- FindContinuousSubsets

  1. Finds all continuous subsets whose members are greater than a particular critical value in magnitude from a particular data array.

        [C#] public static int[,] FindContinuousSubsets(double[] data, double criticalValue, bool includeEnds);

  2. Finds all continuous subsets whose members are greater than a particular critical value in magnitude from a particular data array.

        [C#] public static int[,] FindContinuousSubsets(int[] data, int criticalValue, bool includeEnds);

- FindNeighbours

  1. Finds elements from a `Double` array. The found elements should be the closest in magnitude to a particular checker.

        [C#] public static double[,] FindNeighbours(double checker, double[] members, double flag);

  2. Finds elements from an `integer` array. The found elements should be the closest in magnitude to a particular checker.

        [C#] public static int[,] FindNeighbours(int checker, int[] members, int flag);

- FindPosition

  1. Finds the checker position in the members array.

        [C#] public static int FindPosition(double checker, double[] members);

  2. Finds the checker position in the members array.

        [C#] public static int FindPosition(int checker, int[] members);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the members array is a null reference or empty, and when the elements of the members array is not in ascending order.</td>
</tr>
</tbody>
</table>

- FirstSubstring

  Reports the index of the first element of a string array that contains a particular substring.

        [C#] public static int FirstSubstring(string[] strs, string s);

- GetDateTime

  Obtains a string representation of the current date and time on the local machine, in the form of “yyyy-MM-dd HH:mm:ss”.

        [C#] public static string GetDateTime();

- GetExceptionalValue

  Overloaded.
1. Gets a non-existing Int32 value which is smaller than any members of the data array.
   \[
   \text{public static int GetExceptionalValue(double[] data);} \\
   \]

2. Gets a non-existing Int32 value which is smaller than any members of the data array.
   \[
   \text{public static int GetExceptionalValue(int[] data);} \\
   \]

- HasNegative
  Overloaded.

1. Determines whether a Double array contains any negative values.
   \[
   \text{public static bool HasNegative(double[] values);} \\
   \]

2. Determines whether a jagged Double array contains any negative values.
   \[
   \text{public static bool HasNegative(double[][] values);} \\
   \]

3. Determines whether an integer array contains any negative values.
   \[
   \text{public static bool HasNegative(int[] values);} \\
   \]

4. Determines whether a jagged integer array contains any negative values.
   \[
   \text{public static bool HasNegative(int[][] values);} \\
   \]

- Hex2Dec
  Converts a hexadecimal integer number to its decimal equivalent.
   \[
   \text{public static string Hex2Dec(string hex);} \\
   \]

- IsDigits
  Determines whether a String consists only of decimal digits (0-9).
   \[
   \text{public static bool IsDigits(string s);} \\
   \]

- IsDigitsOrLetters
  Determines whether a String is a combination of decimal digits (0-9) and letters (a-z and/or A-Z).
   \[
   \text{public static bool IsDigitsOrLetters(string s);} \\
   \]

- IsEmptyString
  Determines whether a String is empty or a null reference.
   \[
   \text{public static bool IsEmptyString(string s);} \\
   \]

- IsInAscendingOrder
  Overloaded.

1. Determines whether the array elements are in ascending order.
   \[
   \text{public static bool IsInAscendingOrder(double[] members);} \\
   \]

2. Determines whether the array elements are in ascending order.
   \[
   \text{public static bool IsInAscendingOrder(int[] members);} \\
   \]

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the members array is a null reference or empty.</td>
</tr>
</tbody>
</table>

- IsInDatePeriod
  Overloaded.

1. Determines whether a particular date is inside a date period.
   \[
   \text{public static bool IsInDatePeriod(int start, int end, int today);} \\
   \]
2. Determines whether a particular date is inside a date period.
[C#] public static bool IsInDatePeriod(RomanDate start, RomanDate end, RomanDate today);

- IsInDescendingOrder
  Overloaded.
  1. Determines whether the array elements are in descending order.
  [C#] public static bool IsInDescendingOrder(double[] members);
  2. Determines whether the array elements are in descending order.
  [C#] public static bool IsInDescendingOrder(int[] members);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the members array is a null reference or empty.</td>
</tr>
</tbody>
</table>

- IsInValueRange
  Overloaded.
  1. Determines whether a particular value is inside a value range (boundary inclusive).
  [C#] public static bool IsInValueRange(double lower, double upper, double checker);
  2. Determines whether a particular value is inside a value range (boundary inclusive).
  [C#] public static bool IsInValueRange(int lower, int upper, int checker);

- IsLetters
  Determines whether a String consists only of letters (a-z and/or A-Z).
  [C#] public static bool IsLetters(string s);

- IsValidIP
  Determines whether a String representation of an Internet Protocol (IP) address is well formed.
  [C#] public static bool IsValidIP(string ip);

- LastSubstring
  Reports the index of the last element of a string array that contains a particular substring.
  [C#] public static int LastSubstring(string[] strs, string s);

- Max
  Overloaded.
  1. Returns the maximum element in a one-dimensional Array.
  [C#] public static double Max(double[] data);
  2. Returns the maximum element in a section of a one-dimensional Array.
  [C#] public static double Max(double[] data, int index, int length);
  3. Returns the maximum element in one-dimensional Array.
  [C#] public static int Max(int[] data);
  4. Returns the maximum element in a section of a one-dimensional Array.
  [C#] public static int Max(int[] data, int index, int length);
1.6 Utilities Structure

- **Min**
  Overloaded.

  1. Returns the minimum element in a one-dimensional `Array`.
  ```csharp
  public static double Min(double[] data);
  ```
  2. Returns the minimum element in a section of a one-dimensional `Array`.
  ```csharp
  public static double Min(double[] data, int index, int length);
  ```
  3. Returns the minimum element in one-dimensional `Array`.
  ```csharp
  public static int Min(int[] data);
  ```
  4. Returns the minimum element in a section of a one-dimensional `Array`.
  ```csharp
  public static int Min(int[] data, int index, int length);
  ```

- **PlotData**
  Overloaded.

  1. Plots data series as polygons or open lines (or closed and open curves) of different colors and widths in an arbitrary size `Gif` and saves it as a `Gif` image in a particular file.
  ```csharp
  public static void PlotData(PointF[][] points, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, bool[] isClosed, Color[] fillColors, string file);
  ```
  2. Plots data series as polygons or open lines (or closed and open curves) of different colors and widths in an arbitrary size `Gif` and returns it through the `HttpResponse` of the invoking Web Form page.
  ```csharp
  public static void PlotData(PointF[][] points, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, bool[] isClosed, Color[] fillColors, Page page);
  ```
  3. Plots data series as lines (or curves) of different colors and widths in an arbitrary size `Gif` and saves it as a `Gif` image in a particular file.
  ```csharp
  public static void PlotData(PointF[][] points, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, string file);
  ```
  4. Plots data series as lines (or curves) of different colors and widths in an arbitrary size `Gif` and returns it through the `HttpResponse` of the invoking Web Form page.
  ```csharp
  public static void PlotData(PointF[][] points, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, Page page);
  ```
  5. Plots data series as lines of different colors and widths in an arbitrary sized `Bitmap` and saves it as a `Gif` image in a particular file.
  ```csharp
  public static void PlotData(float[][] data, Color bgColor, Color[] colors, float[] widths, int w, int h, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, string file);
  ```
6. Plots data series as lines of different colors and widths in an arbitrary size Gif and returns it through the HttpResponse of the invoking Web Form page.

```csharp
public static void PlotData(float[,] data, Color bgColor, Color[] colors, float[] widths, int w, int h, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, Page page);
```

7. Plots data series as polygons or open lines (or closed and open curves) of different colors and widths in an arbitrary size Gif and saves it as a Gif image in a particular file.

```csharp
public static void PlotData(float[,] data, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, bool[] isClosed, Color[] fillColors, string file);
```

8. Plots data series as polygons or open lines (or closed or open curves) of different colors and widths in an arbitrary size Gif and returns it through the HttpResponse of the invoking Web Form page.

```csharp
public static void PlotData(float[,] data, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, bool isClosed, Color[] fillColors, Page page);
```

9. Plots data series as lines (or curves) of different colors and widths in an arbitrary size Gif and saves it as a Gif image in a particular file.

```csharp
public static void PlotData(float[,] data, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, string file);
```

10. Plots data series as lines (or curves) of different colors and widths in an arbitrary size Gif and returns it through the HttpResponse of the invoking Web Form page.

```csharp
public static void PlotData(float[,] data, Color bgColor, Color[] colors, float[] widths, string[] lineTypes, int w, int h, float[] xRange, float[] yRange, string[] xLabels, string[] yLabels, bool drawGridLines, Color gridLineColor, Page page);
```

11. Plots data series as lines of different colors and widths in a 800 x 600 pixels Bitmap with a WhiteSmoke background and saves it as a Gif image in a particular file.

```csharp
public static void PlotData(float[,] data, Color[] colors, float[] widths, string file);
```

12. Plots data series as lines of different colors and widths in an arbitrary size Gif and returns it through the HttpResponse of the invoking Web Form page.

```csharp
public static void PlotData(float[,] data, Color[] colors, float[] widths, Page page);
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when data, colors and widths do not have the same Length.</td>
</tr>
</tbody>
</table>

- Radian2Degree Radian2Degree
1.6 Utilities Structure

Converts radians to degrees.

[C#] public static double Radian2Degree(double radians);

- RandomList
  Overloaded.

  1. Generates a list of random values that allows duplicates.
     [C#] public static int[] RandomList(int length);

  2. Generates a list of random values that does or does not allow duplicates.
     [C#] public static int[] RandomList(int length, bool duplicate);

  3. Generates a list of random values between a particular interval with or without duplicates.
     [C#] public static int[] RandomList(int length, bool allowDuplicate, int min, int max);

- RandomUpperString
  Generates a random String of particular number of uppercase letters.
  [C#] public static string RandomUpperString(int length, bool allowDuplicate);

- RemoveChars
  Overloaded.

  1. Removes particular Chars from a String.
     [C#] public static string RemoveChars(string s, char[] chars);

  2. Removes particular Chars from a String.
     [C#] public static string RemoveChars(string s, string chars);

- Replace
  Replaces all occurings of a collection of Chars in a particular String by a specified Char.
  [C#] public static string Replace(string s, char[] char1, char char2);

- Round
  Rounds a Double to an integer.
  [C#] public static int Round(double d);

- SearchArray
  Overloaded.

  1. Searches for a particular element in a Double Array.
     [C#] public static int SearchArray(double element, double[] array);

  2. Searches for a particular element in an integer Array.
     [C#] public static int SearchArray(int element, int[] array);

  3. Searches for a particular element in an String Array.
     [C#] public static int SearchArray(string element, string[] array);

  4. Searches for a particular element in a SoilData.SoilCharacteristic Array.
     [C#] public static int SearchArray(SoilCharacteristic soilchar, SoilCharacteristic[] characteristics);

  5. Searches for elements in a SoilData.SoilCharacteristic Array.
     [C#] public static int[] SearchArray(SoilCharacteristic[] pair, SoilCharacteristic[] characteristics);
- **Second2Degree**
  Converts the expression of an angle from the whole number of seconds to decimal degrees.

```csharp
public static double Second2Degree(int seconds);
```

**Exceptions**:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>seconds</code> is negative</td>
</tr>
</tbody>
</table>

- **Second2DMS**
  Converts the expression of an angle from the whole number of seconds to its degree-minute-second equivalence.

```csharp
public static int[] Second2DMS(int seconds);
```

**Exceptions**:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>seconds</code> is negative</td>
</tr>
</tbody>
</table>

- **SlopeDegree2Percentage**
  Converts a slope gradient expression in degrees (°) to its percentage (%) equivalent.

```csharp
public static double SlopeDegree2Percentage(double deg);
```

**Exceptions**:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>deg</code> is out of range [0, 90].</td>
</tr>
</tbody>
</table>

- **SlopePercentage2Degree**
  Converts a slope gradient expression in percentage (%) to its degrees(°) equivalent.

```csharp
public static double SlopePercentage2Degree(double pc);
```

**Exceptions**:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>pc</code> is out of range [0, 100].</td>
</tr>
</tbody>
</table>

- **Sum**
  Overloaded.

  1. Computes the sum of a particular data array.

```csharp
public static double Sum(double[] data);
```

  2. Computes the sum of a particular data array.

```csharp
public static double Sum(double[] data, int index, int length);
```

  3. Computes the sum of a particular data array.

```csharp
public static long Sum(int[] data);
```

  4. Computes the sum of a particular data array.

```csharp
public static long Sum(int[] data, int index, int length);
```

- **ToDoubleArray**
  Converts a `String` array to a `Double` array.

```csharp
public static double[] ToDoubleArray(string[] rawdata);
```

**Exceptions**:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown if any <code>rawdata</code> member can not be converted to <code>Double</code>.</td>
</tr>
</tbody>
</table>

- **ToIntegerArray**
  Converts a `String` array to an `Int32` array.

```csharp
public static int[] ToIntegerArray(string[] rawdata);
```

**Exceptions**:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown if any <code>rawdata</code> member can not be converted to <code>Int32</code>.</td>
</tr>
</tbody>
</table>
1.6 Utilities Structure

- ToSentenceForm
  Transforms a String into the sentence form, i.e., the leading character is in upper case and all the rest in lower case.
  
  [C#] public static string ToSentenceForm(string s);

- ToTitleForm
  Overloaded.
  
  1. Transforms a String into the title form, i.e., the leading character of each word is in upper case and the rest in lower case.
     
     [C#] public static string ToTitleForm(string s);
  
  2. Transforms a String into the title form, i.e., the leading character of each non-small word is in upper case and the rest in lower case.
     
     [C#] public static string ToTitleForm(string s, string[] list);

- TraverseDirectory
  Overloaded.
  
  1. Traverses the current directory (and sub-directories) to search for a particular group of files.
     
     [C#] public static void TraverseDirectory(string pattern, ref ArrayList list);
  
  2. Traverses a (sub-)directory tree to search for a particular group of files.
     
     [C#] public static void TraverseDirectory(string path, string pattern, ref ArrayList list);

- Trim
  Trims the specified heading and tailing Chars of a particular String.
  
  [C#] public static string Trim(string s, char[] chars);

- WeightedAverage
  Overloaded.
  
  1. Computes the average of the data array.
     
     [C#] public static double WeightedAverage(double[] data);
  
  2. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.
     
     [C#] public static double WeightedAverage(double[] data, int start, int end);
  
  3. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.
     
     [C#] public static double WeightedAverage(double[] data, int start, int end, CalendarType calendar);

  Exceptions:
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the data array does not contain DAILY, DECADELY or MONTHLY data.</td>
</tr>
</tbody>
</table>

  4. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.
     
     [C#] public static double WeightedAverage(double[] data, int[] period);

  Exceptions:
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the length of period is not 2; and when the data array does not contain DAILY, DECADELY or MONTHLY data.</td>
</tr>
</tbody>
</table>
5. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.

[C#]
public static double WeightedAverage(double[] data, int[] period, CalendarType calendar);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the length of period is not 2; and when the data array does not contain DAILY, DECADELY or MONTHLY data.</td>
</tr>
</tbody>
</table>

6. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.

[C#]
public static double WeightedAverage(double[] data, RomanDate start, RomanDate end);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when start or end is not using the same calendar; and when the data array does not contain DAILY, DECADELY or MONTHLY data.</td>
</tr>
</tbody>
</table>

7. Computes the weighted average of the data parameter in a time interval that is defined by a starting and an ending date.

[C#]
public static double WeightedAverage(double[] data, RomanDate[] period);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the length of period is not 2; or the elements are not using the same calendar; and when the data array does not contain DAILY, DECADELY or MONTHLY data.</td>
</tr>
</tbody>
</table>

### 1.7 WlesException Class

#### 1.7.1 Summary

Represents errors which are caused by the WLES application.

#### 1.7.2 Declaration

[C#] public class WlesException : Exception

#### 1.7.3 Public Instance Constructors

- public WlesException(string message);
  - Initializes a new instance of the WlesException class.

- public WlesException(string message, Exception e);
  - Initializes a new instance of the WlesException class.

### 1.8 Logistics.InternalLink Enumeration

#### 1.8.1 Summary

Represents the most frequently used hyperlinks within WLES. The links can be either relative, to the Webserver root, or absolute.

#### 1.8.2 Declaration

[C#] public enum Logistics.InternalLink
1.8.3 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>The WLES homepage.</td>
</tr>
<tr>
<td>Login</td>
<td>The WLES user login page.</td>
</tr>
<tr>
<td>MyProfile</td>
<td>The WLES user profile report page.</td>
</tr>
<tr>
<td>MyAccount</td>
<td>The WLES user account maintenance page.</td>
</tr>
<tr>
<td>GuestLogin</td>
<td>The WLES guest user login page.</td>
</tr>
<tr>
<td>Register</td>
<td>The WLES user registration page.</td>
</tr>
<tr>
<td>UserFiltering</td>
<td>The WLES user filtering page.</td>
</tr>
<tr>
<td>Interactive</td>
<td>The WLES interactive mode page.</td>
</tr>
<tr>
<td>Batch</td>
<td>The WLES batch mode page.</td>
</tr>
<tr>
<td>Help</td>
<td>The WLES help.</td>
</tr>
<tr>
<td>Doc</td>
<td>The WLES documentation.</td>
</tr>
<tr>
<td>Logout</td>
<td>The WLES user logout page.</td>
</tr>
<tr>
<td>EmailPwd</td>
<td>The WLES password recovery page.</td>
</tr>
<tr>
<td>Comments</td>
<td>The WLES suggestions and comments page.</td>
</tr>
<tr>
<td>ImageBroker</td>
<td>The WLES image broker page.</td>
</tr>
<tr>
<td>CustomImage</td>
<td>The WLES custom image page.</td>
</tr>
<tr>
<td>Err</td>
<td>The WLES general error handling page.</td>
</tr>
<tr>
<td>Err404</td>
<td>The WLES “page-not-found” error handling page.</td>
</tr>
<tr>
<td>Err500</td>
<td>The WLES “server-error” handling page.</td>
</tr>
<tr>
<td>StudyArea</td>
<td>The study area input page in the Interactive mode.</td>
</tr>
<tr>
<td>ClimaticData</td>
<td>The climatic data input page in the Interactive mode.</td>
</tr>
<tr>
<td>GrowingPeriod</td>
<td>The GrowingPeriod determination and report page.</td>
</tr>
<tr>
<td>CropData</td>
<td>The crop definition page in the Interactive mode.</td>
</tr>
<tr>
<td>RPPReport</td>
<td>The UGent.WLES.RPP report page.</td>
</tr>
<tr>
<td>CropCoeff</td>
<td>The CropCoefficient determination page.</td>
</tr>
<tr>
<td>SoilData1</td>
<td>The soil data (part 1) collection page.</td>
</tr>
<tr>
<td>RootingDepth</td>
<td>The rooting depth determination page.</td>
</tr>
<tr>
<td>EffRain</td>
<td>The EffectiveRainfall determination page.</td>
</tr>
<tr>
<td>DepletionFraction</td>
<td>The soil water depletion fraction determination page.</td>
</tr>
<tr>
<td>YieldResponse</td>
<td>The YieldResponseToWater evaluation page.</td>
</tr>
<tr>
<td>WPPReport</td>
<td>The UGent.WLES.WPP report page.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoilCharSet</td>
<td>The SoilData.SoilCharacteristics selection page.</td>
</tr>
<tr>
<td>SoilData2</td>
<td>The soil data (part 2) collection page.</td>
</tr>
<tr>
<td>CropRequirements</td>
<td>The crop requirements table building page.</td>
</tr>
<tr>
<td>SoilIndex</td>
<td>The SoilIndex appraising page.</td>
</tr>
<tr>
<td>ManagementIndex</td>
<td>The ManagementIndex appraising page.</td>
</tr>
<tr>
<td>LPPReport</td>
<td>The UGent.WLES.LPP report page.</td>
</tr>
</tbody>
</table>

1.9 RomanDate.CalendarType Enumeration

1.9.1 Summary

Specifies the calendar type.

1.9.2 Declaration

[C#] public enum RomanDate.CalendarType
1.9.3 Remarks

Number of days in each month can be obtained in a 12-element integer array by calling the GetDaysPerMonth method.

1.9.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GregorianCalendar</td>
<td>Represents the Gregorian Calendar in which a year has 365 days; Leap year is not considered.</td>
</tr>
<tr>
<td>WlesCalendar</td>
<td>Represents the WLES Calendar in which there are 360 days a year, and 30 days a month.</td>
</tr>
</tbody>
</table>

1.10 RomanDate.LengthOfYear Enumeration

1.10.1 Summary

Specifies the length of a year in terms of months, decades or days.

1.10.2 Declaration

[C#] public enum RomanDate.LengthOfYear

1.10.3 Remarks

This enumeration represents the length of a year expressed in the number months, decades, or days in the DefaultCalendar. This enumeration is useful when it is desirable to have a strongly typed specification of the length of a year. For example, this enumeration is essential to the building of the WlesDataSet, or to the defining of the growing period.

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOUTHS</td>
<td>Number of months in a year.</td>
</tr>
<tr>
<td>DECADES</td>
<td>Number of decades in a year.</td>
</tr>
<tr>
<td>DAYS</td>
<td>Number of days in a year.</td>
</tr>
</tbody>
</table>

1.11 RomanDate.MonthOfYear Enumeration

1.11.1 Summary

Specifies the month of the year.

1.11.2 Declaration

[C#] public enum RomanDate.MonthOfYear

1.11.3 Remarks

The MonthOfYear enumeration represents the month of the year. There are 12 members defined in this enumeration, ranging from January to December. The underlying ordinal is 1-based.

This enumeration is useful when it is desirable to have a strongly typed specification of the month of the year. For example, this enumeration is one of the formal parameters of the InterpolateDaylength method.
1.11.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAN</td>
<td>Represents the month of January.</td>
</tr>
<tr>
<td>FEB</td>
<td>Represents the month of February.</td>
</tr>
<tr>
<td>MAR</td>
<td>Represents the month of March.</td>
</tr>
<tr>
<td>APR</td>
<td>Represents the month of April.</td>
</tr>
<tr>
<td>MAY</td>
<td>Represents the month of May.</td>
</tr>
<tr>
<td>JUN</td>
<td>Represents the month of June.</td>
</tr>
<tr>
<td>JUL</td>
<td>Represents the month of July.</td>
</tr>
<tr>
<td>AUG</td>
<td>Represents the month of August.</td>
</tr>
<tr>
<td>SEP</td>
<td>Represents the month of September.</td>
</tr>
<tr>
<td>OCT</td>
<td>Represents the month of October.</td>
</tr>
<tr>
<td>NOV</td>
<td>Represents the month of November.</td>
</tr>
<tr>
<td>DEC</td>
<td>Represents the month of December.</td>
</tr>
</tbody>
</table>
Chapter 2

Namespace UGent.WLES.Data

The UGent.WLES.Data namespace integrates all data manipulation functionalities, including data interpolation, database access and updating, etc.

2.1 Overview

2.1.1 Classes

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoilHorizon</td>
<td>Represents a soil horizon.</td>
<td>54</td>
</tr>
<tr>
<td>SoilProfile</td>
<td>Represents a soil profile.</td>
<td>65</td>
</tr>
<tr>
<td>SoilTexture</td>
<td>Represents the soil texture.</td>
<td>78</td>
</tr>
</tbody>
</table>

2.1.2 Structures

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClimaticData</td>
<td>Manipulates climatic data and constants.</td>
<td>82</td>
</tr>
<tr>
<td>CropData</td>
<td>Manipulates crop data.</td>
<td>86</td>
</tr>
<tr>
<td>Database</td>
<td>Provides database accessing and updating facilities, including fetching data, inserting/updating records, and logging.</td>
<td>93</td>
</tr>
<tr>
<td>SoilData</td>
<td>Manipulates soil data.</td>
<td>97</td>
</tr>
<tr>
<td>WlesDataSet</td>
<td>Maintains a runtime datastore between the UGent.WLES application and the UGent.WLES database. It not only collects data, checks data integrity, flushes data into the database, but also retrieves data from the database upon request.</td>
<td>100</td>
</tr>
</tbody>
</table>

2.1.3 Enumerations

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClimaticData.ClimaticParamSet</td>
<td>Specifies the categories of the climatic parameters.</td>
<td>113</td>
</tr>
<tr>
<td>ClimaticData.HemisphereOfEarth</td>
<td>Specifies the hemispheres of the Earth.</td>
<td>114</td>
</tr>
<tr>
<td>ClimaticData.TemperatureRegime</td>
<td>Specifies the temperature regimes, namely, CELSIUS and FAHRENHEIT.</td>
<td>114</td>
</tr>
<tr>
<td>CropData.FarmingSystem</td>
<td>Represents the rainfed and the irrigated farming systems.</td>
<td>115</td>
</tr>
<tr>
<td>SoilData.GravelType</td>
<td>Represents the types of dominant gravel.</td>
<td>115</td>
</tr>
<tr>
<td>SoilData.LandSuitabilityClass</td>
<td>Represents the land suitability classes.</td>
<td>115</td>
</tr>
<tr>
<td>SoilData.SandClass</td>
<td>Represents the dominant subclass of sands according to the percentage distribution of sand particles.</td>
<td>116</td>
</tr>
<tr>
<td>SoilData.SoilCharacteristic</td>
<td>Specifies the soil characteristics that are included in the crop requirement tables.</td>
<td>116</td>
</tr>
</tbody>
</table>
2.2 SoilHorizon Class

2.2.1 Summary

Represents a soil horizon.

2.2.2 Declaration

[C#] public class SoilHorizon

2.2.3 Remarks

A soil horizon is a layer of soil or soil material approximately parallel to the land surface; it differs from adjacent pedogenetically related layers in properties such as color, structure, texture, consistence, and chemical, biological, and mineralogical compositions.

The SoilHorizon class represents a soil horizon and encapsulates the properties and characteristics of it. The class should be used in the following way:

1. Declare a new instance using one of the two constructors with the horizon ID and description. Statement

   SoilHorizon Bt = new SoilHorizon("Bt");

   initializes a new instance to represent a Bt horizon; and statements

   string id, description;
   id = "Ah";
   description = "10YR 3/3 dry, 10YR 3/2 moist; sandy loam; " +
   moderate medium subangular blocky; ";
   description += "soft, friable; no effervescence; abundant " +
   "fine roots.";

   SoilHorizon Ah = new SoilHorizon(id, description);

   initialize a new instance of the Ah horizon with some descriptive texts.

2. Set a property before use. A corresponding exception will be thrown if otherwise. The following code segment first sets the horizon’s particle size distribution, and then gets its texture classes.

   [C#]
   double clay, silt, sand;
   clay = 20;
SoilHorizon Class

2.2

silt = 40;
sand = 100 - (clay + silt);

Bt.ParticleDistribution = new double[] {clay, silt, sand};
SoilData.SoilTextureClass[] texture;
// 1-element array: loam
texture = Bt.GetTextureClasses();

3. It is good practice to set all properties immediately after a new horizon is declared, although
the class is designed in such a way that data can be fed by category so that lacking data of
one category will not block the functionality of the class as a whole.

A list of typical horizons in a profile can be found in the documentation of the SoilProfile class.
A soil profile, or in general any soil, may be properly represented by an array of SoilHorizons.

2.2.4 Public Instance Constructors

• public SoilHorizon(string id);
  Initializes a new instance of SoilHorizon.

• public SoilHorizon(string id, string description);
  Initializes a new instance of SoilHorizon.

2.2.5 Public Instance Properties

• ACEC
  Gets or sets the apparent cation exchange capacity (ACEC) of the horizon, expressed in
  centimoles of charge per kilogram of clay [cmol(+) kg\(^{-1}\) clay].

  [C#]
  public double ACEC { get; set; }

  Remarks: Apparent CEC (ACEC or CEC\(_{\text{clay}}\)) is the CEC expressed as a proportion of the
  clay, usually for soils low in organic matter:

  \[
  ACEC = \frac{CEC \cdot 100}{%\text{clay}}
  \]

  where: CEC = the cation exchange capacity of the mineral soil [cmol(+) kg\(^{-1}\) soil]; %\text{clay} =
  clay content (%);

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the ACEC was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the ACEC is not positive.</td>
</tr>
</tbody>
</table>

• BaseSaturation
  Gets or sets the BaseSaturation of the horizon.

  [C#]
  public double BaseSaturation { get; set; }

  Remarks:

  1. This property - A SoilProfile instance keeps a BaseSaturation property whose pur-
     pose is to receive external values of base saturation in case it cannot be calculated by
     the GetBaseSaturation method due to data incompleteness. The BaseSaturation
     property has to be set with a meaningful value before it can be accessed. Otherwise a
     WlesException is thrown;

  2. The GetBaseSaturation method - The GetBaseSaturation method is used to calcu-
     late the base saturation from the concentrations of individual bases and the CEC. The
     WlesException is thrown if any of these values are not available. A successful call to
     this method also resets the BaseSaturation property value;

  3. About base saturation -
(a) The proportion of CEC satisfied by the basic cations (Ca, Mg, K, and Na) is termed base saturation (BS%).

(b) This property is inversely related to soil acidity. The soil pH increases as the BS% increases. The availability of nutrient cations such as Ca, Mg, and K to plants increases with increasing BS%.

(c) Base saturation is usually close to 100% in arid region soils. Base saturation below 100% indicates that part of the CEC is occupied by hydrogen and/or aluminium ions. Base saturation above 100% indicates that free soluble salts or lime may be present, or that there is a procedural problem with the analysis.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when base saturation was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the base saturation is not positive.</td>
</tr>
</tbody>
</table>

• BulkDensity

Gets or sets the bulk density (kg m$^{-3}$).

[C#]
```csharp
public double BulkDensity { get; set; }
```

Remarks:

1. Soil bulk density is a measure of how dense and tightly packed a sample of soil is. It is determined by measuring the mass of dry soil per unit of volume (kg m$^{-3}$).

2. The bulk density of soil depends on the structure (shape) of the soil peds, how tightly they are packed, the number of spaces (pores), and the composition of the soil particles. Soils made of minerals will have a different bulk density than soils made of organic material.

3. In general, soil bulk density can range from 500 kg m$^{-3}$ or less in organic soils with many pore spaces, to as high as 2,000 kg m$^{-3}$ or greater in very compact mineral horizons.

4. Bulk density is used to convert between mass and volume of a soil sample. The volume of a soil sample can be calculated by dividing the sample mass by the bulk density of the soil. Conversely, the mass of a soil sample can be calculated by multiplying the sample volume by the bulk density of the soil. The fraction of pore space in a soil, its porosity, is calculated as one minus the ratio of bulk density to particle density:

\[
\text{Porosity} = 1 - \frac{\text{BulkDensity}}{\text{ParticleDensity}}
\]

5. The bulk density of a soil sample should be adjusted for any rocks or coarse fragments it contains. For instance: the GetRevisedPAW method.

6. Unit conversion: 1 kg m$^{-3} = 0.001$ g cm$^{-3}$

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the bulk density was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the bulk density is not positive.</td>
</tr>
</tbody>
</table>

• Ca

Gets or sets the calcium cation (Ca$^{2+}$) concentration of the horizon, expressed in centimoles of charge per kilogram of soil [cmol(+)] kg$^{-1}$ soil].

[C#]
```csharp
public double Ca { get; set; }
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when Ca was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set Ca is not positive.</td>
</tr>
</tbody>
</table>
2.2 SoilHorizon Class

- CaCO3
  Gets or sets the calcium carbonate content (CaCO₃, %) of the horizon.

[C#] public double CaCO3 { get; set; }

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the CaCO₃ content was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the CaCO₃ content is negative.</td>
</tr>
</tbody>
</table>

- CEC
  Gets or sets the cation exchange capacity (CEC) of the horizon, expressed in cmol(+) per kilogram of soil [cmol(+)/kg⁻¹ soil].

[C#] public double CEC { get; set; }

**Remarks:**

1. The cation exchange capacity (CEC) is the maximum amount of exchangeable cations that a soil can adsorb. CEC is viewed as an indicator of the soil’s capacity to hold cation nutrients. Primarily there are two groups of cations, i.e., the basic cations and the acidic cations. The amount of these positively charged cations a soil can hold is described as the CEC and is usually expressed in centimoles per kilograms of soil [cmol(+)/kg⁻¹ soil].
   - Basic cations: calcium (Ca²⁺), magnesium (Mg²⁺), potassium (K⁺) and sodium (Na⁺);
   - Acidic cations: hydrogen (H⁺) and aluminium (Al³⁺).

2. The CEC of the soil is determined by the amount of clay and/or humus present. These two colloidal substances are essentially the cation warehouse or reservoir of the soil and are very important because they improve the nutrient and water holding capacity of the soil. Sandy soils with very little organic matter (OM) have a low CEC, but heavy clay soils with high levels of OM would have a much greater capacity to hold cations.

3. **Note** The proportion of CEC satisfied by basic cations (Ca, Mg, K, and Na) is termed base saturation (BS%). This property is inversely related to exchangeable acidity when the pH falls between 6 and 7. (In other words, the BS% increases as the pH increases from 6 to 7, and the two are uncorrelated outside this pH interval). The availability of nutrient cations such as Ca, Mg, and K to plants increases with increasing BS%. Base saturation is usually close to 100% in arid region soils. Base saturation below 100% indicates that part of the CEC is occupied by hydrogen and/or aluminium ions. Base saturation above 100% indicates that soluble salts or lime may be present, or that there is a procedural problem with the analysis.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the CEC was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the CEC is not positive.</td>
</tr>
</tbody>
</table>

- Depth
  Gets or sets the depth of this horizon, m.

[C#] public double[] Depth { get; set; }

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when horizon depth was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the horizon depth is not of a 2-element array or any of the members is negative.</td>
</tr>
</tbody>
</table>

- Description
  Gets or sets the textual description of the horizon.

[C#] public string Description { get; set; }
• **EC**

  Gets or sets the electrical conductivity of the horizon, expressed in deciSiemens per meter (dS m$^{-1}$).

  ```csharp
  public double EC { get; set; }
  ```

  **Remarks:**

  1. Soil electrical conductivity (EC) is a measure of the conduction of electricity through a standard soil extract or paste. The value can reflect the amount of soluble salts in an extract and therefore provides an indication of soil salinity.

  2. The predominant mechanism causing the salt accumulation in irrigated agricultural soils is evapotranspiration. The salt contained in the irrigation water is left behind in the soil as the pure water passes back to the atmosphere through the processes of evaporation and plant transpiration. The effects of salinity are manifested in loss of stand, reduced rates of plant growth, reduced yields, and in severe cases, total crop failure. Salinity limits water uptake by plants by reducing the osmotic potential and thus the total soil water potential. Salinity may also cause specific ion toxicity or upset the nutritional balance. In addition, the salt composition of the soil water influences the composition of cations on the exchange complex of soil particles, which influences the thickness of the diffuse double layer of the clay, causing swelling clays and thus reducing soil permeability and tilth, depending on salinity level and exchangeable cation composition.

  3. From a global perspective, irrigated agriculture makes an essential contribution to the food needs of the world. While only 15% of the world’s farmland is irrigated, roughly 35 to 40% of the total supply of food and fiber comes from irrigated agriculture. However, vast areas of irrigated land are threatened by salinization. Although accurate worldwide data are not available, it is estimated that roughly half of all existing irrigation systems (totaling about 250 million ha) are affected by salinity and waterlogging.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the EC was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the EC is not positive.</td>
</tr>
</tbody>
</table>

• **ESP**

  Gets or sets the exchangeable sodium percentage (ESP, %) of the horizon.

  ```csharp
  public double ESP { get; set; }
  ```

  **Remarks:** Exchangeable sodium percentage (ESP) is the extent to which the adsorption complex of a soil is occupied by sodium. It is expressed as the proportion of the exchangeable sodium to the cation exchange capacity (CEC), usually as a percentage:

  \[ ESP = \left(\frac{Na}{CEC}\right) \cdot 100 \]  

  where: \(Na\) = the sodium concentration of the horizon; \(CEC\) = the cation exchange capacity. Both expressed in centimoles of charge per kilogram of soil [cmol$(+)$ kg$^{-1}$ soil].

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the ESP was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the ESP is negative.</td>
</tr>
</tbody>
</table>

• **FC, WP**

  Gets or sets the soil moisture contents at the field capacity and the wilting point (% w/w).

  ```csharp
  public double[] FC, WP { get; set; }
  ```

  **Remarks:**

  1. Field capacity - The percentage of water remaining in the soil 2 or 3 days after the soil has been saturated and free drainage has practically ceased. The percentage is usually expressed in terms of weight in WLES.
2. Wilting point - The moisture content of a soil at which plants (specifically sunflower plants) wilt and fail to recover their turgidity when placed in a dark, humid atmosphere. The wilting point is commonly estimated by measuring the water content at the 15-bar matric suction of a soil. Also called permanent wilting point.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the field capacity and the wilting point were not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the field capacity and the wilting point is not of a 2-element array, or any of the members is negative.</td>
</tr>
</tbody>
</table>

- Gravel
  Gets or sets the gravel content (% w/w).

  [C#] public double Gravel { get; set; }

  Remarks: Clay, silt and sand are mineral soil particles that are smaller than 2 mm in diameter. Particles greater than 2 mm are called gravels, stones, or generally, coarse fragments and are not considered to be soil material (hence to correct such parameters as soil water storage with volume % of coarse fragments).

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the gravel content was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the gravel content is not in range [0, 100].</td>
</tr>
</tbody>
</table>

- GravelType
  Gets or sets the gravel type.

  [C#] public GravelType GravelType { get; set; }

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the gravel type was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the gravel type is not a member of SoilData.GravelType.</td>
</tr>
</tbody>
</table>

- Gypsum
  Gets or sets the gypsum content (CaSO₄·2H₂O, %) of the horizon.

  [C#] public double Gypsum { get; set; }

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when gypsum content was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the gypsum content is negative.</td>
</tr>
</tbody>
</table>

- ID
  Gets the horizon ID.

  [C#] public string ID { get; }

- K
  Gets or sets the potassium cation (K⁺) concentration of the horizon, expressed in centimoles of charge per kilogram of soil [cmol(+) kg⁻¹ soil].

  [C#] public double K { get; set; }

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when K was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set K is not positive.</td>
</tr>
</tbody>
</table>
• LimitingLayerDepth
  Gets or sets the depth of the root limiting layer, m.
  
  [C#] public double LimitingLayerDepth { get; set; }

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the value used to set this property is not in range of the horizon boundaries.</td>
</tr>
</tbody>
</table>

• Mg
  Gets or sets the magnesium cation ($\text{Mg}^{2+}$) concentration of the horizon, expressed in centimoles of charge per kilogram of soil \([\text{cmol}(+) \text{ kg}^{-1} \text{ soil}]\).
  
  [C#] public double Mg { get; set; }

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when Mg was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set Mg is not positive.</td>
</tr>
</tbody>
</table>

• Na
  Gets or sets the sodium cation ($\text{Na}^{+}$) concentration of the horizon, expressed in centimoles of charge per kilogram of soil \([\text{cmol}(+) \text{ kg}^{-1} \text{ soil}]\).
  
  [C#] public double Na { get; set; }

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when Na was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set Na is not positive.</td>
</tr>
</tbody>
</table>

• OC
  Gets or sets the organic carbon content (O.C, %) of the horizon.
  
  [C#] public double OC { get; set; }

  Remarks:

  1. Measurement of soil organic carbon gives an estimate of the amount of organic matter in a soil as a percentage by weight. The level of organic matter in the soil is a broad indicator of soil condition and its concentration in the soil is largely determined by the addition of surface litter (fallen leaves, manure and dead organisms) and root material and the rate at which microbes break down organic compounds. Much carbon is added to the soil from root material as well.

  2. Carbon is essential for plant growth, due to its effects on other soil properties. Organic matter is important since it binds soil particles together into stable aggregates which are necessary for soil structural stability. It is also involved in adsorption of cations, such as calcium, magnesium, potassium and sodium, which are important in plant nutrition, and can significantly influence soil water holding capacity, especially in more sandy soils.

  3. Soil organic carbon varies with depth. Levels are usually the highest in the topsoil and generally decrease rapidly with depth. In many situations the level of organic carbon does not appear to be strongly related to soil types although some soil landscapes are defined by having high organic carbon. Typically it varies as a function of climate and land use. It is usually higher in forest and pasture areas than in cultivated areas. It generally follows continental rainfall and temperature patterns (carbon accumulation increases with increasing rainfall; carbon decomposition increases with increasing temperature - these factors are then influenced by the annual input of carbon to the soil system). The soil carbon levels are the result of the balance between inputs of carbon into the soil, and decomposition and removal of carbon from the soil.

  Exceptions:
2.2 SoilHorizon Class

- ParticleDistribution
  Gets or sets the particle size distribution (i.e., clay, silt, sand contents in % w/w) of the horizon.

[C#] public double[] ParticleDistribution { get; set; }

Remarks:
1. The amount of each particle size group (sand, silt, or clay) in a soil is known as the soil particle size distribution. The sum of the weight percentage of sand, silt and clay equals to 100:

\[ \text{clay} + \text{silt} + \text{sand} = 100 \]  \hspace{2cm} (2.4)

2. Sand is the largest soil particle size (2.0-0.063 mm), silt is intermediate in size (0.063-0.002 mm), and clay is the smallest (less than 0.002 mm). Particles greater than 2 mm are called gravels, stones, or generally, coarse fragments and are not considered to be soil material.

3. For more information, refer to SoilTexture.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the particle distribution was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the particle distribution is not of a 3-element array, or any of the members is negative, or the sum of all 3 members does not equal to 100.</td>
</tr>
</tbody>
</table>

- pH\_H2O
  Gets or sets the pH (in water, \(H_2O\)) value of the horizon.

[C#] public double pH\_H2O { get; set; }

Remarks:
1. Soil pH is the pH of a solution in equilibrium with soil. A pH of 7 is neutral. pH gives an indication of the availability of plant nutrients and relates to the growth requirements of particular crops. In most agricultural soils, pH is usually between 4 and 8.5. Excessive acidity or alkalinity makes the soils inhospitable for plant and microbial growth. Microorganisms don’t flourish in low pH and enzyme activity also changes with the change in pH.

2. The effect of soil pH is great on the solubility of minerals or nutrients. Before a nutrient can be used by plants it must be dissolved in the soil solution. Most minerals and nutrients are more soluble or available in acid soils than in neutral or slightly alkaline soils. However strongly acid soils (pH 4.0-5.0) may be toxic to the growth of some plants. A pH range of approximately 6 to 7 promotes the most ready availability of plant nutrients.

3. The soil pH can also influence plant growth by its effect on activity of beneficial microorganisms. Bacteria that decompose soil organic matter are hindered in strong acid soils. This prevents organic matter from breaking down, resulting in an accumulation of organic matter and the tie up of nutrients, particularly nitrogen, that are held in the organic matter.

4. Soils tend to become acidic as a result of:
   (a) rainwater leaching away basic ions (calcium, magnesium, potassium and sodium);
   (b) carbon dioxide from decomposing organic matter and root respiration dissolving in soil water to form a weak organic acid;
(c) formation of strong organic and inorganic acids, such as nitric and sulfuric acid, from decaying organic matter and oxidation of ammonium and sulfur fertilizers. Strongly acid soils are usually the result of the action of these strong organic and inorganic acids.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the pH was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the pH is negative.</td>
</tr>
</tbody>
</table>

- **pH**

  Gets or sets the pH (in potassium chloride, KCl) value of the horizon.

  ```csharp
  public double pHKCl { get; set; }
  ```

  **Remarks:**

  1. pH-H$_2$O is usually viewed as actual acidity; and pH-KCl as potential acidity.
  2. The relative magnitude of pH-H$_2$O and pH-KCl reflects the sign of variable net charge of the soil colloids:
     - Negative charges: pH-H$_2$O > pH-KCl;
     - Positive charges: pH-H$_2$O < pH-KCl.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the pH was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the pH is negative.</td>
</tr>
</tbody>
</table>

- **SandClass**

  Gets or sets the sub-categories of sands (i.e., fine sand, coarse sand, etc.) of the horizon.

  ```csharp
  public SandClass SandClass { get; set; }
  ```

  **Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the sand class was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the sand class is not a legal member of the SoilData.SandClass enumeration.</td>
</tr>
</tbody>
</table>

- **Structure**

  Gets or sets the soil structure of the horizon.

  ```csharp
  public SoilStructure Structure { get; set; }
  ```

  **Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the soil structure was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the soil structure is not a legal member of the SoilData.SoilStructure enumeration.</td>
</tr>
</tbody>
</table>

- **SumOfBasicCations**

  Gets or sets the sum of the basic cations (Ca, Mg and K; Na excluded since being used by ESP) of the horizon, expressed in centimoles of charge per kilogram of soil [cmol(+) kg$^{-1}$ soil].

  ```csharp
  public double SumOfBasicCations { get; set; }
  ```

  **Remarks:**

  A SoilHorizon instance holds a separate value of the sum of basic cations while keeps the concentrations of individual bases (Ca, Mg, K and Na): The former is not necessarily the arithmetic sum of the latter although it should be. The reason for this arrangement is that a soil dataset may provide the sum while the individual concentrations are omitted; Difference between this property, the GetBasesSum and the GetBases methods:
– SumOfBasicCations property: (1) Sum of Ca, Mg and K concentrations; used in evaluation of the SoilIndex. Na is omitted out since it is included in the ESP property; (2) A WlesException is thrown when accessed if its value was not explicitly set. If this is the case, the sum should be manually calculated using the individual property values;

– GetBases method: Returns a 4-element Double array which takes values of the individual concentrations of Ca, Mg, K and Na.

– GetBasesSum method: Calculates the sum of Ca, Mg, K and Na using the individual property values.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when sum of basic cations was not properly set upon call.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument passed in to set the sum of basic cations is not positive.</td>
</tr>
</tbody>
</table>

### 2.2.6 Public Instance Methods

- **CalculateESP**
  Calculates the exchangeable sodium percentage (ESP, %) of the horizon.

  
  ```csharp
  public double CalculateESP();
  ```

  **Remarks:**
  1. For the definition of ESP and the relationship between sodium content and soil alkalinity, refer to the ESP property;
  2. Difference between this method and the ESP property: the ESP value was initially set via the property which does not necessarily have direct link to the sodium concentration (Na) and the CEC. Therefore a manual re-evaluation of the ESP value can be conducted by calling this method; the call updates the ESP property with the return value as well.

- **GetBases**
  Gets the concentrations of the exchangeable bases of the horizon, expressed in centimoles of charge per kilogram of soil [cmol(+) kg\(^{-1}\) soil].

  
  ```csharp
  public double[] GetBases();
  ```

  **Remarks:** Difference between this method, the SumOfBasicCations property and the GetBasesSum method:

<table>
<thead>
<tr>
<th>Property/Method</th>
<th>Details of implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SumOfBasicCations</td>
<td>(1) Sum of Ca, Mg and K concentrations; used in evaluation of the SoilIndex. Na is omitted out since it is included in the ESP property; (2) A WlesException is thrown when accessed if its value was not explicitly set. If this is the case, the sum should be manually calculated using the individual property values;</td>
</tr>
<tr>
<td>GetBases</td>
<td>Returns a 4-element Double array which takes values of the individual concentrations of Ca, Mg, K and Na.</td>
</tr>
<tr>
<td>GetBasesSum</td>
<td>Calculates the sum of Ca, Mg, K and Na using the individual property values.</td>
</tr>
</tbody>
</table>

- **GetBaseSaturation**
  Gets the base saturation of the horizon, expressed as a percentage (%).

  
  ```csharp
  public double GetBaseSaturation();
  ```

  **Remarks:**
  1. The BaseSaturation property - A SoilProfile instance keeps a BaseSaturation property whose purpose is to receive external values of base saturation in case it cannot be calculated by the GetBaseSaturation method due to data incompleteness. The
**BaseSaturation** property has to be set with a meaningful value before it can be accessed. Otherwise a **WlesException** is thrown;

2. This method - The **GetBaseSaturation** method is used to *calculate* the base saturation from the concentrations of individual bases and the **CEC**. The **WlesException** is thrown if any of these values are not available. A successful call to this method also resets the **BaseSaturation** property value;

3. About base saturation -
   (a) The proportion of **CEC** satisfied by the basic cations (Ca, Mg, K, and Na) is termed base saturation (BS%).
   (b) This property is inversely related to soil acidity. The soil pH increases as the BS% increases. The availability of nutrient cations such as Ca, Mg, and K to plants increases with increasing BS%.
   (c) Base saturation is usually close to 100% in arid region soils. Base saturation below 100% indicates that part of the **CEC** is occupied by hydrogen and/or aluminium ions. Base saturation above 100% indicates that free soluble salts or lime may be present, or that there is a procedural problem with the analysis.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WlesException</strong></td>
<td>Thrown when the concentration of one or more of the exchangeable bases is negative; or when that of the <strong>CEC</strong> is not positive.</td>
</tr>
</tbody>
</table>

- **GetBasesSum**
  Gets the sum of the exchangeable bases of the horizon, expressed in centimoles of charge per kilogram of soil [cmol(+) kg⁻¹ soil].

[C#] public double GetBasesSum();

**Remarks:**

1. The **BaseSaturation** property - A **SoilProfile** instance keeps a **BaseSaturation** property whose purpose is to receive external values of base saturation in case it cannot be calculated by the **GetBaseSaturation** method due to data incompleteness. The **BaseSaturation** property has to be set with a meaningful value before it can be accessed. Otherwise a **WlesException** is thrown;

2. This method - The **GetBaseSaturation** method is used to *calculate* the base saturation from the concentrations of individual bases and the **CEC**. The **WlesException** is thrown if any of these values are not available. A successful call to this method also resets the **BaseSaturation** property value;

3. About base saturation -
   (a) The proportion of **CEC** satisfied by the basic cations (Ca, Mg, K, and Na) is termed base saturation (BS%).
   (b) This property is inversely related to soil acidity. The soil pH increases as the BS% increases. The availability of nutrient cations such as Ca, Mg, and K to plants increases with increasing BS%.
   (c) Base saturation is usually close to 100% in arid region soils. Base saturation below 100% indicates that part of the **CEC** is occupied by hydrogen and/or aluminium ions. Base saturation above 100% indicates that free soluble salts or lime may be present, or that there is a procedural problem with the analysis.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WlesException</strong></td>
<td>Thrown when the concentration of one or more of the exchangeable bases is negative.</td>
</tr>
</tbody>
</table>
2.3 SoilProfile Class

- **GetPAW**
  Gets the plant available soil water (% v/v) of this horizon.
  
  \[
  \text{public double GetPAW();}
  \]

  **Remarks**: This method calculates the plant available soil water in volumetric percentage (% v/v) of this horizon, in the following order:
  
  1. **Primary**: It attempts to evaluate the PAW using a pF curve, if available; otherwise
  2. **Alternative**: It evaluates the PAW using texture classes;
  3. **Correction**: The PAW value is further corrected using the coarse fragments data, if available.

- **GetSoilWaterStorage**
  Gets the plant available soil water storage (S\textsubscript{a}D) in this horizon, mm.
  
  \[
  \text{public double GetSoilWaterStorage();}
  \]

- **GetTextureClasses**
  Gets all the qualified \texttt{SoilData.SoilTextureClasses} of this horizon.
  
  \[
  \text{public SoilTextureClass[]} GetTextureClasses();
  \]

- **GetThickness**
  Gets the thickness of this horizon, m.
  
  \[
  \text{public double GetThickness();}
  \]

- **HasLimitingLayer**
  Determines whether a root limiting layer is presented in this horizon.
  
  \[
  \text{public bool HasLimitingLayer();}
  \]

### 2.3 SoilProfile Class

#### 2.3.1 Summary

Represents a soil profile.

#### 2.3.2 Declaration

\[
\text{[C#] public class SoilProfile}
\]

#### 2.3.3 Remarks

A soil profile is the vertical display of soil horizons.

The \texttt{SoilProfile} class represents a soil profile and encapsulates the common properties and the composing horizons. Figure 2.1 depicts a typical profile with O, A, B and C horizons. \texttt{SoilHorizons} within a \texttt{SoilProfile} may be identified by using capital letters O, A, B, C, and E, together with lowercase letters for distinctions of sub-horizons. Most soils have three major horizons – the surface horizon (A), the subsoil (B), and the substratum (C). Some soils have an organic horizon (O) on the surface, but this horizon can also be buried. The master horizon, E, is used for subsurface horizons that have a significant loss of minerals (eluviation). Hard bedrock, which is not soil, uses the letter R.

- **O horizon**: At the top of the profile is the O horizon. The O horizon is primarily composed of organic matter. Fresh litter is found at the surface, while at depth all signs of vegetation structure have been destroyed by decomposition. The decomposed organic matter, or humus, enriches the soil with nutrients (nitrogen, potassium, \textit{etc.}), aids soil structure (acts to bind particles), and enhances soil moisture retention.
• A horizon - Beneath the O horizon is the A horizon. The A horizon marks the beginning of the true mineral soil. In this horizon organic material mixes with inorganic products of weathering. The A horizon typically is dark colored due to the presence of organic matter. Eluviation, the removal of inorganic and organic substances from a horizon by leaching occurs in the A horizon. Eluviation is driven by the downward movement of soil water.

• E horizon - The E horizon generally is a light-colored horizon with eluviation being the dominant process. Leaching, or the removal of clay particles, organic matter, and/or oxides of iron and aluminium is active in this horizon. Under coniferous forests, the E horizon often has a high concentration of quartz giving the horizon an ashy-gray appearance.

• B horizon - Beneath the E horizon lies the B horizon. The B horizon is a zone of illuviation where downward moving, especially fine material, is accumulated. The accumulation of fine material leads to the creation of a dense layer in the soil. In some soils the B horizon is enriched with calcium carbonate in the form of nodules or as a layer. This occurs when the carbonate precipitates out of downward moving soil water or from capillary action.

• C horizon - The C horizon represents the soil parent material, either created in situ or transported into its present location. Beneath the C horizon lies bedrock (R).

The SoilProfile class is supposed to contain all necessary data items found in a soil profile description, which is part of a soil survey report. Such data include both descriptive texts and analytical results. For instance, a profile may be identified by its ProfileNumber and its Coordinates. Surface characteristics (LandUse, Vegetation, Topography, Microrelief, Slope and so forth) and profile properties (Structure, Sand-Class, InternalDrainage, Depth, etc.) are all represented as instance properties, whose values must be assigned explicitly before being called.

The most important component of a SoilProfile is its composing horizons. Among others, a horizon encapsulates more detailed analytical data, for instance the particle size distribution, the organic carbon content, the cation exchange capacity and so on.

Wrapping a collection of SoilHorizons, a SoilProfile provides a generalized interface (RecalSoilCharacteristic) to recalculate the profile’s weighted mean(s) of one or more SoilData.SoilCharacteristic(s), either for the effective rooting zone or to a particular depth. The mean value of a SoilData.SoilCharacteristic can also be calculated by using one of the two overloading methods [static (Shared in Visual Basic) and instance] for each SoilData.SoilCharacteristic. For example, the RecalParticleDistribution method is used to recalculate the average particle size distribution in the whole profile or up to a particular depth. As a result, overall SoilData.SoilTextureClasses can be determined.
However, operations such as the recalculation of the profile mean and the extraction of a section value of a `SoilData.SoilCharacteristic` are all feasible (and preferable!) through the properties and methods exposed by the composing `SoilHorizons`.

### 2.3.4 Public Instance Constructors

- ```
  public SoilProfile(string number, SoilHorizon[] horizons);
  ```
  Initializes a new instance of `SoilProfile`.

### 2.3.5 Public Instance Properties

- **Classification**
  Gets or sets the soil classification.
  ```
  public string Classification { get; set; }
  ```
  **Remarks:**
  1. There is not a universal soil classification system and each specific system is designed to cope with its own requirements. Hence this property takes a `String` value although the WRB Reference Soil Groups (FAO-ISRIC-ISSS, 1998) are enumerated in WLES.
  2. The following classification names are preferred in defining this property:
     - A WRB Reference Soil Group name;
     - A FAO Revised Legend of Soil Map of the World (?) name;
     - A FAO Legend of Soil Map of the World (FAO-UNESCO, 1974) name;
     - A USDA Soil Taxonomy (USDA Soil Survey Staff, 1999) name; or
     - A national classification name.

- **Coordinates**
  Gets or sets the geo-coordinates of the profile.
  ```
  public double[] Coordinates { get; set; }
  ```
  **Remarks:** Longitudes in the Western Hemisphere and latitudes in the Southern Hemisphere are all expressed in negative values.
  **Exceptions:**
  ```
  exception condition
  ArgumentException Thrown when the `Double` array does not contain 2 elements; or the values of the coordinates are out of range: longitudes \([-180, +180]\] and latitudes \([-90, +90]\].
  WlesException Thrown when the coordinates value has not been properly set upon call.
  ```

- **Depth**
  Gets the depth (m) of the profile, i.e., the distance from soil surface to the bottom of the profile.
  ```
  public double Depth { get; }
  ```
  **Exceptions:**
  ```
  exception condition
  WlesException Thrown when the depth of the profile’s bottom horizon was not properly set upon call.
  ```

- **Elevation**
  Gets or sets the elevation of the profile, m.
  ```
  public double Elevation { get; set; }
  ```
  **Exceptions:**
  ```
  exception condition
  WlesException Thrown when elevation has not been properly set upon call.
  ```
• Horizons
  Gets all the soil horizons of the profile.

[C#] public SoilHorizon[] Horizons { get; }

• InternalDrainage
  Gets or sets the drainage class within the profile.

[C#] public string InternalDrainage { get; set; }

  Example: The following statement sets the internal drainage class of the profile with a
  SoilData.SoilDrainageClass enumeration value:

  SoilProfile profile = new SoilProfile(...);
  profile.InternalDrainage = SoilData.SoilDrainageClass.Well.ToString();

• IsCoordinatesOk
  Determines whether the coordinates of the profile has been properly set.

[C#] public bool IsCoordinatesOk { get; }

• IsElevationOk
  Determines whether the elevation datum has been properly set upon call.

[C#] public bool IsElevationOk { get; }

• IsParentMaterialCalcareous
  Gets or sets the flag that determines whether the parent material is calcareous.

[C#] public bool IsParentMaterialCalcareous { get; set; }

• IsParentMaterialKaolinitic
  Gets or sets the flag that determines whether the parent material is kaolinitic.

[C#] public bool IsParentMaterialKaolinitic { get; set; }

• IsParentMaterialKnown
  Determines whether the type of the parent material is known. In other words, this flag deter-
  mines whether the values of IsParentMaterialKaolinitic and/or IsParentMaterialCalcareous
  properties have been explicitly assigned.

[C#] public bool IsParentMaterialKnown { get; }

• IsSandClassOk
  Determines whether the sand class datum is properly set.

[C#] public bool IsSandClassOk { get; }

• IsSlopeOk
  Determines whether the surface slope gradient datum has been properly set upon calling.

[C#] public bool IsSlopeOk { get; }

• IsStructureOk
  Determines whether the soil structure datum is properly set.

[C#] public bool IsStructureOk { get; }

• LandUse
  Gets or sets the land use types.

[C#] public string LandUse { get; set; }

• LimitingLayerDepth
  Gets or sets the depth of the root limiting layer in the profile, m.

[C#] public double LimitingLayerDepth { get; set; }

Remarks:
  1. The depth of a root limiting layer is usually attributed to a particular SoilHorizon.
     Therefore, the Horizons are searched everytime this property value is accessed or re-
     assigned. Under such circumstances, this property is simply a delegate of the SoilHor-
     zon.LimitingLayerDepth property;
2. This property may also keep a value that is deeper than the bottom of all horizons, which may happen in theory under special circumstances.

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>Thrown when the value used to set the depth is not positive.</td>
</tr>
</tbody>
</table>

- **Location**
  Gets or sets the location of the profile.

  [C#] public string Location { get; set; }

- **Microrelief**
  Gets or sets the surface microrelief.

  [C#] public string Microrelief { get; set; }

- **ParentMaterial**
  Gets or sets the parent material descriptions.

  [C#] public string ParentMaterial { get; set; }

- **ProfileDescription**
  Gets or sets the profile description.

  [C#] public string ProfileDescription { get; set; }

  **Remarks**: This property keeps a copy of the profile description, which may be redundant to other properties such as Vegetation, LandUse, ParentMaterial, etc.

- **ProfileNumber**
  Gets or sets the profile number or profile ID.

  [C#] public string ProfileNumber { get; set; }

- **SandClass**
  Gets or sets the sand class of the profile.

  [C#] public SandClass SandClass { get; set; }

  **Exceptions**: Thrown when the value used to set the property is not a member of SoilData.SandClass.

  Thrown when sand class property has not been properly set upon call.

- **Slope**
  Gets or sets the slope gradient, expressed in percentage (%).

  [C#] public double Slope { get; set; }

  **Remarks**: The slope gradient is expressed in percentage in this method. Slope gradient can also be expressed as an angle. Converting methods can be found in the Utilities structure:

  - Convert degree to percentage - SlopeDegree2Percentage;
  - Convert percentage to degree - SlopePercentage2Degree.

  **Exceptions**: Thrown when the slope gradient value is negative or greater than 100.

  Thrown when slope gradient datum has not been properly set upon call.

- **Structure**
  Gets or sets the soil structure of the profile.

  [C#] public SoilStructure Structure { get; set; }

  **Exceptions**: Thrown when the value used to set the soil structure property is not a valid member of SoilData.SoilStructure.

  Thrown when the property has not been properly set upon call.
• Topography
  Gets or sets the surface topography.
  
  [C#] public string Topography { get; set; }

• Vegetation
  Gets or sets the surface vegetation.
  
  [C#] public string Vegetation { get; set; }

2.3.6 Public Instance Methods

• Average
  Overloaded.

  1. Recalculates a SoilData.SoilCharacteristic's average value of the profile, up to a
     particular soil depth, with or without using weighting factors (of equal sections).
     
     [C#] public double Average(double[] data, double depth, bool weighting);

  2. Recalculates the average value(s) of one or more SoilCharacteristic(s) in the profile,
     up to a particular soil depth, with or without using weighting factors (of equal sections).
     
     [C#] public double[] Average(double[][] data, double depth, bool weighting);

  Remarks: This method recalculates the average(s) of the considered soil characteristic(s)
  of the profile as a whole, i.e., from the soil surface to a particular depth, using one of the
  following approaches:

  – Weighted mean of horizons - Mean values of clay, silt and sand contents are calculated
    on the basis of horizons, using the thickness of each horizon as a weighting factor;

  – Weighted mean of equal sections - The profile is firstly cut into equal sections and
    weighting factors are attributed to them in a descending order. The magnitude of the
    weighting factor is inversely proportional to the distance of a section to the soil surface.
    Hence the closer the section is to the surface, the more it contributes to the mean value
    of the profile.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive; and when data does not contain entries for all Horizons.</td>
</tr>
</tbody>
</table>

• GetHorizon
  Overloaded.

  1. Returns the horizon at a particular soil depth.
     
     [C#] public SoilHorizon GetHorizon(double depth);

  2. Returns all horizons whose ID starts with a particular string, case insensitive.
     
     [C#] public SoilHorizon[] GetHorizon(string id);

     Example: The following method call returns all B-horizons, including “Bt”, “Bw”, etc.

     UGent.WLES.Data.SoilProfile profile =
       new UGent.WLES.Data.SoilProfile(...);
     profile.GetHorizon("B");

     The following method call will return all horizons encapsulated with profile:

     UGent.WLES.Data.SoilHorizon [] set1, set2;
     set1 = profile.GetHorizon("");
     set2 = profile.Horizons;
// true
    bool equal = set1.Equals(set2);

And the following will return a null reference (Nothing in Visual Basic):

    profile.GetHorizon("foo");

- GetSoilWaterStorage
  Gets the available soil water storage ($S_a D$, mm) at a particular rooting depth.

  [C#] public double GetSoilWaterStorage(double depth);

  Remarks:
  1. The purpose of this method is to calculate the available soil water storage ($S_a D$, mm) at any rooting depth to obtain the $S_a D$ dynamics during the whole CropCycle.
  2. The rooting depth is in most circumstances within the range of the soil horizons. However, if the availability of soil data is limited, depth of all data-available horizons may be shallower than the rooting depth. In this case, the method will extend the depth of the last known horizon till the maximum rooting depth.

- GetSoilWaterStorageCropCycle
  Gets the daily available soil water storage ($S_a D$, mm) during a CropCycle.

  [C#] public double[] GetSoilWaterStorageCropCycle(double[] rootCropCycle);

- GetTextureClasses
  Overloaded.

  1. Gets all the qualified SoilData.SoilTextureClasses on the basis of the recalculated average of particle size distributions in the profile.

  [C#] public SoilTextureClass[] GetTextureClasses(bool weighting);

  2. Gets all the qualified SoilData.SoilTextureClasses on the basis of the average particle size distribution that is recalculated up to a particular depth in the profile.

  [C#] public SoilTextureClass[] GetTextureClasses(double depth, bool weighting);

  Exceptions:

  - ArgumentException
    Thrown when depth is not positive.

- HasLimitingLayer
  Determines whether a root limiting layer is presented in the profile.

  [C#] public bool HasLimitingLayer();

- HasNegativeCharges
  Determines whether soil colloids have negative net charge.

  [C#] public bool HasNegativeCharges();

  Remarks: The sign of variable net charge of the soil colloids can be classified by comparing the relative magnitudes of the pH-H$_2$O and pH-KCl values of the B-horizon or at 50 cm of an A-C profile:

  - Negative charge - if ($pH_{KCl} - pH_{H_2O}$) < 0
  - Positive charge - if ($pH_{KCl} - pH_{H_2O}$) > 0

- RecalACEC
  Recalculates the average ACEC [cmol(+) kg$^{-1}$ clay] in the B-horizon, or at 50 cm in an A-C profile, without correction for organic matter.

  [C#] public double RecalACEC();

  Remarks: The apparent CEC, or CEC of the clay fraction, is calculated in the B-horizon, or at 50 cm in an A-C profile, without correction for organic matter.
1. The ACEC value used in this method, is not taken from the ACEC property of the SoilHorizon class (unless the CEC value is not available). To the contrary, it is recalculated from the CEC value in the following way:

\[
CEC_{\text{clay}} = CEC_{\text{soil}} \cdot \frac{100}{\% \text{clay}}
\]

where: \( CEC_{\text{soil}} \) = the cation exchange capacity of the mineral soil [cmol(+) kg\(^{-1}\) soil]; \( \% \text{clay} \) = clay content (%);

2. For relationship between ACEC and CEC, and ways to estimate ACEC with or without corrections for organic matter, refer to the ACEC property of the SoilHorizon class.

- **RecalBS**
  Overloaded.

  1. Recalculates the average base saturation (%) of the profile, with or without using weighting factors (of equal sections).

    ```csharp
    [C#] public double RecalBS(bool weighting);
    ```

  2. Recalculates the average base saturation (%) of the profile, up to a particular depth, with or without using weighting factors (of equal sections).

    ```csharp
    [C#] public double RecalBS(double depth, bool weighting);
    ```

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive.</td>
</tr>
</tbody>
</table>

- **RecalCaCO3**
  Overloaded.

  1. Recalculates the average CaCO\(_3\) content (%, w/w) of the profile, with or without using weighting factors (of equal sections).

    ```csharp
    [C#] public double RecalCaCO3(bool weighting);
    ```

  2. Recalculates the average CaCO\(_3\) content (%, w/w) of the profile, to a particular soil depth, with or without using weighting factors (of equal sections).

    ```csharp
    [C#] public double RecalCaCO3(double depth, bool weighting);
    ```

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive.</td>
</tr>
</tbody>
</table>

- **RecalEC**
  Overloaded.

  1. Recalculates the average electrical conductivity (dS m\(^{-1}\)) of the profile, with or without using weighting factors (of equal sections).

    ```csharp
    [C#] public double RecalEC(bool weighting);
    ```

  2. Recalculates the average electrical conductivity (dS m\(^{-1}\)) of the profile, up to a particular depth, with or without using weighting factors (of equal sections).

    ```csharp
    [C#] public double RecalEC(double depth, bool weighting);
    ```

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive.</td>
</tr>
</tbody>
</table>

- **RecalESP**
  Overloaded.
2.3 SoilProfile Class

SoilProfile Class 73

1. Returns the highest ESP value of horizons within 1 meter depth or within the root limiting layer, whichever shallower.

[C#] public double RecalESP();

2. Recalculates the average exchangeable sodium percentage (%) of the profile, with or without using weighting factors (of equal sections).

[C#] public double RecalESP(bool weighting);

3. Recalculates the average exchangeable sodium percentage (%) of the profile, up to a particular depth, with or without using weighting factors (of equal sections).

[C#] public double RecalESP(double depth, bool weighting);

Exceptions:

ArgumentException Thrown when depth value is not positive.

- RecalGravel
  Overloaded.

1. Recalculates the average gravel content (%, w/w) of the profile, with or without using weighting factors (of equal sections).

[C#] public double RecalGravel(bool weighting);

2. Recalculates the average gravel content (%, w/w) of the profile, up to a particular soil depth, with or without using weighting factors (of equal sections).

[C#] public double RecalGravel(double depth, bool weighting);

Exceptions:

ArgumentException Thrown when depth value is not positive.

- RecalGypsum
  Overloaded.

1. Recalculates the average gypsum content (%, w/w) of the profile, with or without using weighting factors (of equal sections).

[C#] public double RecalGypsum(bool weighting);

2. Recalculates the average gypsum content (%, w/w) of the profile, up to a particular soil depth, with or without using weighting factors (of equal sections).

[C#] public double RecalGypsum(double depth, bool weighting);

Exceptions:

ArgumentException Thrown when depth value is not positive.

- RecalOC
  Overloaded.

1. Recalculates the average organic carbon content (%, w/w) of the profile, with or without using weighting factors (of equal sections).

[C#] public double RecalOC(bool weighting);

2. Recalculates the average organic carbon content (%, w/w) of the profile, up to a particular depth, with or without using weighting factors (of equal sections).

[C#] public double RecalOC(double depth, bool weighting);

Exceptions:

ArgumentException Thrown when depth value is not positive.
• RecalParticleDistribution
  Overloaded.

1. Recalculates the average particle size distribution of the profile.
  \[\text{C#}] \quad \text{public double[]} \text{RecalParticleDistribution(bool \textit{weighting});}\]

2. Recalculates the average particle size distribution in the profile, up to a particular soil depth.
  \[\text{C#}] \quad \text{public double[]} \text{RecalParticleDistribution(double \textit{depth}, bool \textit{weighting});}\]

Remarks:

1. This method recalculates the average particle size distribution of the profile, either as a whole (i.e., from the soil surface to the bottom of the profile) or till a certain depth, using one of the following approaches:
   - Weighted mean of horizons - Mean values of clay, silt and sand contents are calculated on the basis of horizons, using the thickness of each horizon as a weighting factor:
     \[
     PSD \left\{ \begin{array}{l}
     \text{clay} = \frac{1}{D} \cdot \sum_{i=1}^{n} (\text{clay}_i \cdot d_i) \\
     \text{silt} = \frac{1}{D} \cdot \sum_{i=1}^{n} (\text{silt}_i \cdot d_i) \\
     \text{sand} = \frac{1}{D} \cdot \sum_{i=1}^{n} (\text{sand}_i \cdot d_i)
     \end{array} \right.
     \] (2.6)
   
   where: \(PSD\) = the average particle content of the profile; \(i\) = serial number of a horizon; \(n\) = total number of horizons; \(d_i\) = thickness of horizon \(i\); \(D\) = depth of the profile.
   - Weighted mean of equal sections - The profile is firstly cut into equal sections and weighting factors are attributed to them in a descending order. The magnitude of the weighting factor is inversely proportional to the distance of a section to the soil surface. Hence the closer the section is to the surface, the more it contributes to the mean value of the profile:
     \[
     PSD \left\{ \begin{array}{l}
     \text{clay} = \frac{1}{N} \cdot \sum_{i=1}^{n} (\text{clay}_i \cdot w_i) \\
     \text{silt} = \frac{1}{N} \cdot \sum_{i=1}^{n} (\text{silt}_i \cdot w_i) \\
     \text{sand} = \frac{1}{N} \cdot \sum_{i=1}^{n} (\text{sand}_i \cdot w_i)
     \end{array} \right.
     \] (2.7)
   
   where: \(PSD\) = the average particle content of the profile; \(i\) = serial number of a section; \(n\) = total number of sections; \(w_i\) = weighting factor of section \(i\); \(N\) = total number of sections.

2. The purposes and the way of calling the overloaded RecalParticleDistribution methods are summarized below:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Method call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile mean, using horizon values</td>
<td>RecalParticleDistribution(false)</td>
</tr>
<tr>
<td>Profile mean, using weighting factors of equal sections</td>
<td>RecalParticleDistribution(true)</td>
</tr>
<tr>
<td>Profile mean to a particular depth, using horizon values</td>
<td>RecalParticleDistribution(d, false)</td>
</tr>
<tr>
<td>Profile mean to a particular depth, using weighting factors of equal sections</td>
<td>RecalParticleDistribution(d, true)</td>
</tr>
</tbody>
</table>

3. Soil particle size distribution of a horizon is usually laboratory measured.
2.3 SoilProfile Class

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive; and when data does not contain entries for all Horizons.</td>
</tr>
</tbody>
</table>

Example:
The following example recalculates the clay, silt and sand contents over the rooting zone of soybean in an hypothetical site. Part of the profile description: physical and chemical data

<table>
<thead>
<tr>
<th>Horizon</th>
<th>depth(cm)</th>
<th>clay(%)</th>
<th>silt(%)</th>
<th>sand(%)</th>
<th>gravel(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ah</td>
<td>0-20</td>
<td>16</td>
<td>4</td>
<td>80</td>
<td>8</td>
</tr>
<tr>
<td>Bw1</td>
<td>20-55</td>
<td>26</td>
<td>3</td>
<td>71</td>
<td>0</td>
</tr>
<tr>
<td>Bw2</td>
<td>55-90</td>
<td>43</td>
<td>13</td>
<td>44</td>
<td>13</td>
</tr>
<tr>
<td>C</td>
<td>90-150</td>
<td>33</td>
<td>7</td>
<td>60</td>
<td>2</td>
</tr>
</tbody>
</table>

A continuous stoneline (5 cm thick) of abundant calcareous concretions presents at the base of the Bw horizon.

using System;
using UGent.WLES;
using UGent.WLES.Data;

public class RecalTextureExample {
    public static void Main(string[] args) {
        SoilHorizon[] horizons = new SoilHorizon[4];
        horizons[0] = new SoilHorizon("Ah");
        horizons[1] = new SoilHorizon("Bw1");
        horizons[2] = new SoilHorizon("Bw2");
        horizons[3] = new SoilHorizon("C");

        double[] clay = new double[] {16, 26, 43, 33,};
        double[] silt = new double[] {4, 3, 13, 7,};
        double[][] boundaries = new double[4][] {
            new double[] {0,.2},
            new double[] {.2,.55},
            new double[] {.55,.9},
            new double[] {.9,1.5},
        };

        for (int i=0; i<horizons.Length; i++) {
            horizons[i].ParticleDistribution = new double[3] {clay[i], silt[i], 100-clay[i]-silt[i]};
            horizons[i].Depth = boundaries[i];
        }

        SoilProfile profile = new SoilProfile("Test-1", horizons);
        double depth = double.Parse(args[0]);
        double[] pd = profile.RecalParticleDistribution(depth, true);
        double avgClay = pd[0];
        double avgSilt = pd[1];
        double avgSand = pd[2];

        Console.WriteLine("Particle size distribution, weighting factors used:");
    }
}
namespace UGent.WLES.Data

double avgClay = pd[0];
double avgSilt = pd[1];
double avgSand = pd[2];

Console.WriteLine("Particle size distribution, weighting factors not used:");
Console.WriteLine(" Clay = {0}%
Silt = {1}%
Sand = {2}%

A root restriction layer presents in the profile at 90 cm depth. Hence run the example with an argument of 0.9:

C:\Examples>RecalTextureExample 0.9

will generate the following results:

Particle size distribution, weighting factors used:
Clay = 24.94%
Silt = 5.06%
Sand = 70%

Particle size distribution, weighting factors not used:
Clay = 30.39%
Silt = 7.11%
Sand = 62.5%

• RecalPH_H2O
  Overloaded.

  1. Recalculates the average pH_H2O value (dimensionless, 7 neutral) of the profile, with or without using weighting factors (of equal sections).
  [C#] public double RecalPH_H2O(bool weighting);

  2. Recalculates the average pH_H2O value (dimensionless, 7 neutral) of the profile, up to a particular depth, with or without using weighting factors (of equal sections).
  [C#] public double RecalPH_H2O(double depth, bool weighting);

  Exceptions:
  ArgumentException Thrown when depth value is not positive.

• RecalPH_KCl
  Overloaded.

  1. Recalculates the average pH_KCl value (dimensionless) of the profile, with or without using weighting factors (of equal sections).
  [C#] public double RecalPH_KCl(bool weighting);
2. Recalculates the average pH\textsubscript{KCl} value (dimensionless) of the profile, up to a particular depth, with or without using weighting factors (of equal sections).

[C#] public double RecalPH\textsubscript{KCl}(double depth, bool weighting);

Exceptions:

- ArgumentException
  Thrown when depth value is not positive.

- RecalSoilCharacteristic
  Overloaded.

1. Provides a generalized interface to recalculate a particular SoilData.Soil-Characteristic’s average value in the profile, with or without using weighting factors (of equal sections).

[C#] public double[] RecalSoilCharacteristic(SoilCharacteristic characteristic, bool weighting);

Return Value: A Double array that represents the average value; the length of the array is 1 or 3, depending on the characteristic value:

- Length = 3 - The characteristic has value Texture. The average clay, silt and sand contents are the 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} elements.
- Length = 1 - The characteristic has a value other than Texture.
- Exceptional - If argument characteristic does not take a valid value from the SoilData.SoilCharacteristic enumeration, an 1-element Double array is returned, and the element takes value -1.

2. Provides a generalized interface to recalculate a particular SoilData.Soil-Characteristic’s average value in the profile, up to a particular soil depth, with or without using weighting factors (of equal sections).

[C#] public double[] RecalSoilCharacteristic(SoilCharacteristic characteristic, double depth, bool weighting);

Return Value: A Double array that represents the average value; the length of the array is 1 or 3, depending on the characteristic value:

- Length = 3 - The characteristic has value Texture. The average clay, silt and sand contents are the 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} elements.
- Length = 1 - The characteristic has a value other than Texture.
- Exceptional - If argument characteristic does not take a valid value from the SoilData.SoilCharacteristic enumeration, an 1-element Double array is returned, and the element takes value -1.

3. Provides a generalized interface to recalculate a group of SoilData.Soil-Characteristics’ average values in the profile, with or without using weighting factors (of equal sections).

[C#] public double[][] RecalSoilCharacteristic(SoilCharacteristic[] characteristics, bool weighting);

Return Value: A jagged Double array that represents the average values; the length of the outer array equals the length of the characteristics array; the inner array has a length of 1 or 3, depending on the value of the corresponding member of characteristics:

- Inner array length = 3 - The corresponding member of characteristic has value Texture. The average clay, silt and sand contents are the 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} elements.
- Inner array length = 1 - The corresponding member of characteristic has a value other than Texture.
- Exceptional - If argument characteristic does not take a valid value from the SoilData.SoilCharacteristic enumeration, the inner array will be an 1-element Double array, and the element has value -1.

4. Provides a generalized interface to recalculate a group of SoilData.Soil-Characteristics’ average values in the profile, up to a particular soil depth, with or without using weighting factors (of equal sections).
public double[][] RecalSoilCharacteristic(SoilCharacteristic[] characteristics, double depth, bool weighting);

Return Value: A jagged Double array that represents the average values; the length of the outer array equals the length of the characteristics array; the inner array has a length of 1 or 3, depending on the value of the corresponding member of characteristics:

- Inner array length = 3 - The corresponding member of characteristic has value Texture. The average clay, silt and sand contents are the 1st, 2nd and 3rd elements.
- Inner array length = 1 - The corresponding member of characteristic has a value other than Texture.
- Exceptional - If argument characteristic does not take a valid value from the SoilData.SoilCharacteristic enumeration, the inner array will be an 1-element Double array, and the element has value -1.

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive.</td>
</tr>
</tbody>
</table>

- RecalSumBC

Overloaded.

1. Recalculates the average sum of basic cations of Ca\(^{2+}\), Mg\(^{2+}\) and K\(^{+}\) [cmol(+) kg\(^{-1}\) soil] of the profile, with or without using weighting factors (of equal sections).

[C#] public double RecalSumBC(bool weighting);

2. Recalculates the average sum of basic cations of Ca\(^{2+}\), Mg\(^{2+}\) and K\(^{+}\) [cmol(+) kg\(^{-1}\) soil] of the profile, up to a particular depth, with or without using weighting factors (of equal sections).

[C#] public double RecalSumBC(double depth, bool weighting);

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive.</td>
</tr>
</tbody>
</table>

Remarks: The term “sum of basic cations” used here is slightly different from what has been defined by the SoilHorizon.GetBases and the GetBasesSum methods. The sum of Ca\(^{2+}\), Mg\(^{2+}\) and K\(^{+}\) concentrations is calculated in this method, while in the latter case all exchangeable bases (Ca\(^{2+}\), Mg\(^{2+}\), K\(^{+}\) and Na\(^{+}\)) are considered.

- SimpleAverage

Calculates the average value of a SoilData.SoilCharacteristic of the profile, up to a particular depth, without using weighting factors.

[C#] public double SimpleAverage(double[] data, double depth);

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when depth value is not positive.</td>
</tr>
</tbody>
</table>

2.4 SoilTexture Class

2.4.1 Summary

Represents the soil texture.

2.4.2 Declaration

[C#] public class SoilTexture
2.4.3 Remarks

1. Soil texture is a term commonly used to designate the proportionate distribution of the different sizes of mineral particles in a soil. It does not include any organic matter. These mineral particles are commonly grouped into 3 main classes: clay, silt and sand. The ranges of the particle sizes are specified in the table below:

<table>
<thead>
<tr>
<th>Particle</th>
<th>Size (diameter in mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clay</td>
<td>&lt; 0.002</td>
</tr>
<tr>
<td>Silt</td>
<td>0.002-0.063</td>
</tr>
<tr>
<td>Sand</td>
<td>0.063-2.000</td>
</tr>
</tbody>
</table>

2. A soil texture triangle (see Figure 2.2a) is used to classify the texture class of a soil. The sides of the soil texture triangle are scaled for the percentages of clay, silt and sand. Clay percentages are read from left to right across the triangle. Silt is read from the upper right to lower left. Sand from lower right towards the upper left portion of the triangle. The intersection of the three sizes on the triangle give the texture class.

A virtual soil texture triangle (see Figure 2.2b) in the Cartesian coordinate system is supposed in WLES when treating the textural triangle and the textural classes within it as topological entities, with the lower-left corner of the triangle (clay=0, silt=0, sand=100) as the origin of the Cartesian plane.

Therefore, a point within the textural triangle can be solely determined either by the clay-silt-sand particle distributions (Figure 2.2a), or by an x-y coordinate pair (Figure 2.2b); and these two representations can be easily mapped to each other.

3. Conversion from textural representation (clay, silt) to Cartesian representation (x, y):

\[
\text{Cartesian} \leftarrow \begin{cases} 
  x = \text{silt} + \text{clay} \cdot [1 - \cos(60^\circ)] \\
  y = \frac{\text{clay} \cdot \sin(60^\circ)}{\text{silt}} 
\end{cases}
\]
Conversion from Cartesian representation \((x, y)\) to textural representation (clay, silt):

\[
\text{Texture} \rightarrow \begin{cases} 
  \text{clay} & = \frac{y}{\sin(60^\circ)} \\
  \text{silt} & = x - \text{clay} \cdot [1 - \cos(60^\circ)] \\
  \text{sand} & = 100 - (\text{clay} + \text{silt})
\end{cases}
\]  

(2.9)

### 2.4.4 Public Static Methods

- **GetPolygon**
  Defines a *Polygon* using clay and silt content data.

  ```csharp
  public static Polygon GetPolygon(double[][] textures);
  ```

  **Parameters:**
  - `textures` A jagged `Double` array. The inner array is a 2-element `Double` array with clay and silt content as its 1st and 2nd elements.

  **Return Value:** A *Polygon* that represents a closed area within the virtual soil texture triangle (Figure 2.2b). The closed area is usually overlapping with that of a `SoilData.SoilTextureClass` or group of `SoilData.SoilTextureClasses`.

- **GetTextureClasses**
  Overloaded.

  1. Gets all the qualified `SoilData.SoilTextureClasses` for the given conditions.
   ```csharp
   public static SoilTextureClass[] GetTextureClasses(double clay, double silt);
   ```

  2. Gets all the qualified `SoilData.SoilTextureClasses` for the given conditions.
   ```csharp
   public static SoilTextureClass[] GetTextureClasses(double clay, double silt, SandClass sc);
   ```

  3. Gets all the qualified `SoilData.SoilTextureClasses` for the given conditions.
   ```csharp
   public static SoilTextureClass[] GetTextureClasses(double clay, double silt, SoilStructure structure);
   ```

  4. Gets all the qualified `SoilData.SoilTextureClasses` for the given conditions.
   ```csharp
   public static SoilTextureClass[] GetTextureClasses(double clay, double silt, SoilStructure structure, SandClass sc);
   ```

  **Remarks:** Conditions to determining the `SoilData.SoilTextureClass` include:

  1. The proportions of clay, silt and sand as weight percentages. Given any two of them, the third is known.
  2. Optional: the structure of the soil.

  A texture (i.e., the clay, silt, sand composites) point (see: `Texture2Point`), that falls on a common border of 2 neighboring classes, or overlaps a sharing node of 3 or more classes, has all the involving classes included in the qualified set of classes.

- **Point2Texture**
  Gets the Clay and Silt contents (%, w/w) of the *SoilTexture* represented by a particular *Point* within the virtual soil texture triangle (Figure 2.2b).

  ```csharp
  public static double[] Point2Texture(Point p);
  ```

  **Return Value:** A 2-element `Double` array that contains the texture composites of the *SoilTexture* represented by \(p\): clay content (%, w/w) as its 1st element, and silt content (%, w/w) as the 2nd.
2.4 SoilTexture Class

- **Texture2Point**
  Gets a Point (in the virtual soil texture triangle, see: Figure 2.2b) that represents the SoilTexture with particular clay and silt contents.
  
  ```csharp
  [C#] public static Point Texture2Point(double clay, double silt);
  ```

- **TextureClass2Polygon**
  Transforms a SoilData.SoilTextureClass into a Polygon that represents a closed area of this class within the virtual soil texture triangle (Figure 2.2b).
  
  ```csharp
  [C#] public static Polygon TextureClass2Polygon(SoilTextureClass cls);
  ```

2.4.5 Public Instance Constructors

- **public SoilTexture(double clay, double silt);**
  Initializes a new instance of SoilTexture.
  **Exceptions:**

  ```
  exception condition
  ArgumentException Thrown when any or sum of clay and silt contents not in range [0, 100].
  ```

- **public SoilTexture(double clay, double silt, SandClass sc);**
  Initializes a new instance of SoilTexture.
  **Exceptions:**

  ```
  exception condition
  ArgumentException Thrown when any or sum of clay and silt contents not in range [0, 100].
  ```

- **public SoilTexture(double clay, double silt, SoilStructure structure);**
  Initializes a new instance of SoilTexture.
  **Exceptions:**

  ```
  exception condition
  ArgumentException Thrown when any or sum of clay and silt contents not in range [0, 100].
  ```

- **public SoilTexture(double clay, double silt, SoilStructure structure, SandClass sc);**
  Initializes a new instance of SoilTexture.
  **Exceptions:**

  ```
  exception condition
  ArgumentException Thrown when any or sum of clay and silt contents not in range [0, 100].
  ```

- **public SoilTexture(SoilHorizon horizon);**
  Initializes a new instance of SoilTexture.

- **public SoilTexture(Point p);**
  Initializes a new instance of SoilTexture.
  **Exceptions:**

  ```
  exception condition
  ArgumentException Thrown when any or sum of the clay and silt contents, which are transformed from p, not in range [0, 100].
  ```

2.4.6 Public Instance Properties

- **Clay**
  Gets the clay content (% w/w).
  
  ```csharp
  [C#] public double Clay { get; }
  ```
• IsSandClassOk
  Determines whether the SoilData.SandClass data is properly set.
  [C#] public bool IsSandClassOk { get; }

• IsSoilStructureOk
  Determines whether the SoilData.SoilStructure data is properly set.
  [C#] public bool IsSoilStructureOk { get; }

• Sand
  Gets the sand content (% w/w).
  [C#] public double Sand { get; }

• SandClass
  Gets the SoilData.SandClass of this soil.
  [C#] public SandClass SandClass { get; }

• Silt
  Gets the silt content (% w/w).
  [C#] public double Silt { get; }

• SoilStructure
  Gets the structure of this soil.
  [C#] public SoilStructure SoilStructure { get; }

2.4.7 Public Instance Methods
• GetPoint
  Gets a Point that represents this SoilTexture.
  [C#] public Point GetPoint();

• GetTextureClasses
  Gets all the qualified SoilData.SoilTextureClasses for this SoilTexture.
  [C#] public SoilTextureClass[] GetTextureClasses();

• SetSandClass
  Sets the SoilData.SandClass data.
  [C#] public void SetSandClass(SandClass sc);

• SetSoilStructure
  Sets the SoilData.SoilStructure data.
  [C#] public void SetSoilStructure(SoilStructure structure);

2.5 ClimaticData Structure

2.5.1 Summary
Manipulates climatic data and constants.

2.5.2 Declaration
[C#] public struct ClimaticData

2.5.3 Public Static Fields
• ClimaticData.ABSOLUTE_ZERO
  Zero of the absolute temperature at the Kelvin scale (K).
  [C#] public const double ABSOLUTE_ZERO = 273.15;
2.5 ClimaticData Structure

- ClimaticData.LAMBDA
  Latent heat of evaporation in MJ kg\(^{-1}\).
  \[\text{public const double LAMBDA = 2.45;}\]

- ClimaticData.SIGMA
  Stefan Boltzmann constant in MJ m\(^{-2}\) K\(^{-4}\) day\(^{-1}\).
  \[\text{public const double SIGMA = 4.903E-09;}\]

2.5.4 Public Static Methods

- CalculateClearSkyRatio
  Calculates the fraction of daytime in which the sky is clear (1 – \(f\)).
  \[\text{public static double CalculateClearSkyRatio(double \(n\), double \(latitude\), HemisphereOfEarth \(hemisphere\), MonthOfYear \(month\));}\]

- CalculateDayNightTemperatures
  Overloaded.
  1. Calculates the average daytime and nighttime temperatures.
     \[\text{public static double[]} \text{CalculateDayNightTemperatures(double \(tmax\), double \(tmin\), double \(daylength\));}\]
  2. Calculates the average daytime and nighttime temperatures.
     \[\text{public static double[]} \text{CalculateDayNightTemperatures(double \(tmax\), double \(tmin\), double \(latitude\), HemisphereOfEarth \(hemisphere\), MonthOfYear \(month\));}\]

Remarks: The algorithm, originally presented by Petricevic (Gommes, 1983), takes not only the daily maximum and minimum temperatures but also the \(daylength\) as input parameters. If not readily available, \(daylength\) can be determined as a function of the latitude of the location and the time of the year, using InterpolateDaylength method.

- CalculateDecadelyFromDaily
  Calculates the year-round decadely values from daily values.
  \[\text{public static double[]} \text{CalculateDecadelyFromDaily(double[]} \text{daily, ClimaticParamSet \(param\));}\]
  \text{Return Value: A 36-element Double array that represents decadely values.}\n
- CalculateDecadelyFromMonthly
  Calculates the year-round decadely values from monthly values.
  \[\text{public static double[]} \text{CalculateDecadelyFromMonthly(double[]} \text{monthly, ClimaticParamSet \(param\));}\]
  \text{Return Value: A 36-element Double array that represents decadely values.}\n
- CalculateMonthlyFromDaily
  Calculates the year-round monthly values from daily values.
  \[\text{public static double[]} \text{CalculateMonthlyFromDaily(double[]} \text{daily, ClimaticParamSet \(param\));}\]
  \text{Return Value: A 12-element Double array; a null reference if \(daily\) is not a 360- or 365-element Double array.}\n
- CalculateMonthlyFromDecadely
  Calculates the year-round monthly values from decadely values.
  \[\text{public static double[]} \text{CalculateMonthlyFromDecadely(double[]} \text{decadely, ClimaticParamSet \(param\));}\]
  \text{Return Value: A 12-element Double array; a null reference if \(daily\) is not a 360- or 365-element Double array.}\n
• **CalculateOvercastRatio**
  Calculates the fraction of daytime in which the sky is overcast ($f$).

[C#]  

```csharp
public static double CalculateOvercastRatio(double n, double latitude, HemisphereOfEarth hemisphere, MonthOfYear month);
```

• **ChangeCalendarForDailyArray**
  Changes the `RomanDate.CalendarType`(127,382),(203,396) of a particular daily data array.

[C#]  

```csharp
public static double[] ChangeCalendarForDailyArray(double[] daily, ClimaticParamSet param, CalendarType calendar);
```

**Return Value**: Daily data array of the calendar `RomanDate.CalendarType`. Length is 360 if target calendar is `WlesCalendar` or 365 if `GregorianCalendar`.

• **GetDailyArray**
  Overloaded.

  1. Gets year-round daily data from monthly or decadely data.

   ```csharp
   public static double[] GetDailyArray(double[] monthlyOrDecadely, ClimaticParamSet param);
   ```

  2. Gets year-round daily data from monthly or decadely data.

   ```csharp
   public static double[] GetDailyArray(double[] monthlyOrDecadely, ClimaticParamSet param, CalendarType calendar);
   ```

**Return Value**: A 360-element `Double` array, provided that the `DefaultCalendar` is set to `WlesCalendar`.

**Remarks**: This method simply rewrites the decadely data in the daily format. No interpolation algorithms are used in the rewriting process, although monthly data are properly transformed into decadely data using the interpolation algorithm implemented in the `GetDecadeData` method.

The table below summarizes the method behaviors in relation to the `monthlyOrDecadely` parameter.

<table>
<thead>
<tr>
<th>Parameter <code>monthlyOrDecadely</code></th>
<th>Method behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monthly data, array length = 12</td>
<td>Interpolates monthly data into decadely data; Rewrites decadely data in daily format.</td>
</tr>
<tr>
<td>Decadely data, array length = 36</td>
<td>Rewrites decadely data in daily format.</td>
</tr>
<tr>
<td>Daily data, array length = 360 (for <code>WlesCalendar</code>) or 365 (for <code>GregorianCalendar</code>)</td>
<td>Rewrites the daily data so that the returned array length agrees with the <code>DefaultCalendar</code>.</td>
</tr>
</tbody>
</table>

**Exceptions**:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentException</code></td>
<td>Thrown when <code>monthlyOrDecadely</code> is not of daily, decadely or monthly data.</td>
</tr>
</tbody>
</table>

• **GetDecadeData**
  Creates decadely data by interpolating readily available monthly data.

[C#]  

```csharp
public static double[] GetDecadeData(double[] monthly, ClimaticParamSet param);
```

**Return Value**: A 3-element array of 64-bit floating point numbers containing the decade values, namely the 1st element as decade 1, the 2nd element as decade 2 and the 3rd element as decade 3, respectively.

**Remarks**: The algorithm makes a difference between climatic parameters that constitute a sum (rainfall, evapotranspiration, etc.) and parameters that are arithmetic mean values (temperature, windspeed, etc.) (Gommes, 1983).
The input monthly values must be in a time series of 3 consecutive months, so that the decade values of the middle month can be interpolated. For example, to estimate the decade values of rainfall in May, the monthly values of rainfall in April, May and June must be available.

**Example:** The following example shows how to get decade rainfall data in June by interpolating monthly rainfall data in May, June and July.

```csharp
using System;
using UGent.WLES.Data;

public class Rainfall {
    public static void Main() {
        double may = 25; // Rainfall in May: 25 mm.
        double june = 30; // Rainfall in June: 30 mm.
        double july = 50; // Rainfall in July: 50 mm.
        double[] months = new double[] {may, june, july};

        double[] decades;
        decades = ClimaticData.GetDecadeData(months,
                                              ClimaticData.ClimaticParamSet.RAINFALL);

        Console.WriteLine("Monthly rainfall data:");
        Console.WriteLine("May: {0} mm", may);
        Console.WriteLine("June: {0} mm", june);
        Console.WriteLine("July: {0} mm", july);

        Console.WriteLine("Decade rainfall data in June:");
        for ( int i=0; i<decades.Length; i++ )
            Console.WriteLine("Decade {0}: {1} mm", i, decades[i]);
    }
}
```

This code produces the following output.

**Monthly rainfall data:**
- May: 25 mm
- June: 30 mm
- July: 50 mm

**Decade rainfall data in June:**
- Decade 1: 8.7 mm
- Decade 2: 9.8 mm
- Decade 3: 11.5 mm

- **HasNegativeValues**
  Determines whether a climatic parameter can have negative values.

  ```csharp
  [C#] public static bool HasNegativeValues(ClimaticParamSet param);
  ```

- **InterpolateDaylength**
  Interpolates/extrapolates the daylength \(N\) datum using known values stored in the database.

  ```csharp
  [C#] public static double InterpolateDaylength(double latitude,
                                                   HemisphereOfEarth hemisphere, MonthOfYear month);
  ```

  **Remarks:** The concept of daylength is defined here as the astronomically possible sunshine duration which depends on the geographical location and the month of the year. The daylength can be divided into two parts: the duration in which the sky is clear, and, the duration in which the sky is overcast (Sys et al., 1991).

  **Example:** The following examples demonstrate the way to obtain daylength data for different geographical locations.
using UGent.WLES.Data;

// Daylength of a location on 3840'N in July
double lat = 38 + 40/60;
ClimaticData.HemisphereOfEarth hemisphere =
    ClimaticData.HemisphereOfEarth.NorthernHemisphere;
RomanDate.MonthOfYear mon = RomanDate.MonthOfYear.JULY;
double daylength;
daylength = InterpolateDaylength(lat, hemisphere, mon);

// Daylength of location on 20S in February
lat = 20;
hemisphere = ClimaticData.HemisphereOfEarth.SOUTHERN;
mon = RomanDate.MonthOfYear.FEBRUARY;
daylength = InterpolateDaylength(lat, hemisphere, mon);

// Daylength of location on 5530'S in December
lat = 55 + 30/60;
hemisphere = ClimaticData.HemisphereOfEarth.SOUTHERN;
mon = RomanDate.MonthOfYear.DECEMBER;
daylength = InterpolateDaylength(lat, hemisphere, mon);

• IsAccumulative
  Determines whether a climatic parameter has an accumulative nature, which means the measured value constitutes a sum in a certain period of time, usually an hour, a day, or a month. Rainfall, evapotranspiration and sunshine hours are in this category.
[C#] public static bool IsAccumulative(ClimaticParamSet param);

• IsAverage
  Determines whether a climatic parameter has an average nature, which means the values are measured as arithmetic means in a certain time interval, usually an hour, a day, or a month.
[C#] public static bool IsAverage(ClimaticParamSet param);

• TransformTemperature
  Inter-transforms temperature values between Celsius and Fahrenheit regimes.
[C#] public static double TransformTemperature(double t, TemperatureRegime target);

• TransformWindVelocity
  Inter-transforms wind velocity at different heights above ground.
[C#] public static double TransformWindVelocity(double v1, double h1, double h2);

### 2.6 CropData Structure

#### 2.6.1 Summary
Manipulates crop data.

#### 2.6.2 Declaration
[C#] public struct CropData

#### 2.6.3 Remarks
1. Definition - A crop is defined by its name and cultivar, and sometimes its alias as well. A unique ID is associated with a crop to facilitate the data manipulation.
2. Reference crops - predefined important annual crops are called reference crops. Crop IDs are already assigned to the reference crops in the WLES Database.

3. Data manipulation - Publicly accessible [public static] methods are used to provide common functionalities, which range from defining a new crop and retrieving existing crop data, to determining the crop-specific parameters such as harvest index and leaf area index.

2.6.4 Public Static Methods

- AddNewCrop
  Overloaded.

  | 1. Adds a new reference crop into the WLES database. |
  | [C#] public static int AddNewCrop(string crop); |
  | 2. Adds a new reference crop into the WLES database. |
  | [C#] public static int AddNewCrop(string crop, string cultivar); |
  | 3. Adds a new reference crop into the WLES database. |
  | [C#] public static int AddNewCrop(string crop, string cultivar, string alias); |

  Remarks: A crop is defined by its name and cultivar which is optional. A unique ID is assigned to each reference crop in the WLES database.

  Exceptions:
  | exception | condition |
  | ArgumentException | Thrown when crop is a null reference or empty. |
  | WlesDatabaseException | Thrown when the underlying database error occurs. |

- GetCropID
  Gets the crop ID of a particular crop and cultivar.
  [C#] public static int GetCropID(string crop, string cultivar);

  Exceptions:
  | exception | condition |
  | ArgumentException | Thrown when (1) the crop string is a null reference or empty; (2) the crop/cultivar pair is not predefined as a reference crop; and (3) the cultivar is ambiguous and thus the ID cannot be uniquely determined. |
  | WlesDatabaseException | Thrown when the underlying database error occurs. |

- GetCropNameAndCultivar
  Gets the crop name and cultivar of a particular crop ID.
  [C#] public static string[] GetCropNameAndCultivar(int cropid);

  Exceptions:
  | exception | condition |
  | ArgumentException | Thrown when the cropid value is zero or negative and when the cropid does not exist in the database. |
  | WlesDatabaseException | Thrown when the underlying database error occurs. |

- GetCropYields
  Gets reference crop yields (ton ha$^{-1}$) under either Rainfed or Irrigated farming conditions.
  [C#] public static string[][] GetCropYields(string crop, string cultivar, FarmingSystem farming);

  Return Value: A jagged String array. The length of the outer array indicates the number of yield records existing under the particular farming conditions; the inner array has 3 string elements: the minimum yield, the maximum yield, and the textual remarks.

  Exceptions:
### Example

The following example extracts reference yield data for sorghum and maize, under both rainfed and irrigation farming conditions.

```csharp
using System;
using UGent.WLES;
using UGent.WLES.Data;

class YieldExample {
    public static void Main(string[] args) {
        CropData.Farming[] farmings =
            new CropData.Farming[] {CropData.Farming.Rainfed, 
                                    CropData.Farming.Irrigated};
        string crop, cultivar;
        crop = args[0];
        cultivar = args[1];

        foreach (CropData.Farming farming in farmings) {
            yields = CropData.GetCropYields(crop, cultivar, farming);
            if (yields != null) {
                Console.WriteLine("Crop yield of {0} ({1}) under {2} farming: ",
                    crop, cultivar, farming.ToString().ToLower());
                foreach (string[] record in yields) {
                    Console.WriteLine(" {0}: {1} - {2} ton/ha.",
                        Utilities.ToSentenceForm(record[2]),
                        record[0], record[1]);
                }
            } else Console.WriteLine("Data not available!");
        }
    }
}
```

Call the program with sorghum (cultivar is left blank since we don’t care about it in this moment):

`YieldExample sorghum ""

will generate the following results:

- **Crop yield of sorghum () under rainfed farming:**
  - Good commercial yield: 2.5 - 3.5 ton/ha.
  - Average farmer yield: 1.3 - 2 ton/ha.

- **Crop yield of sorghum under irrigated farming:**
  - Good management: 3.5 - 5 ton/ha.

Call the program with maize (tropical):

`YieldExample maize tropical`

will generate the following results:
Crop yield of maize (tropical) under rainfed farming:
  Good commercial yield of grain: 6 - 9 ton/ha.
  Good commercial yield of fodder: 33 - 33 ton/ha.
Crop yield of sorghum under irrigated farming:
  Good commercial yield of grain: 6 - 9 ton/ha.
  Good commercial yield of fodder: 80 - 80 ton/ha.

- GetHarvestIndex
  Gets the harvest index value of a particular crop and cultivar.

[C#]
public static double GetHarvestIndex(string crop, string cultivar);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when argument <code>crop</code> or <code>cultivar</code> is empty string; or when the database query returns nothing.</td>
</tr>
<tr>
<td>WlseDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

- GetHarvestIndexRange
  Gets the range (i.e., the minimum and maximum values) of the harvest index values of a particular crop and cultivar.

[C#]
public static double[] GetHarvestIndexRange(string crop, string cultivar);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when argument <code>crop</code> or <code>cultivar</code> is empty string; or when the database query returns nothing.</td>
</tr>
<tr>
<td>WlseDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

- GetLeafAreaIndex
  Gets the leaf area index value at a particular crop’s maximum growth rate.

[C#]
public static double GetLeafAreaIndex(string crop);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when argument <code>crop</code> is empty string; or when the database query returns nothing.</td>
</tr>
<tr>
<td>WlseDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

- GetLeafAreaIndexRange
  Gets the range (i.e., the minimum and maximum values) of the leaf area index values of a particular crop at its maximum growth rate.

[C#]
public static double[] GetLeafAreaIndexRange(string crop);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when argument <code>crop</code> is empty string; or when the database query returns nothing.</td>
</tr>
<tr>
<td>WlseDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

- GetManagementIndex
  Overloaded.

  1. Computes the management index (dimensionless, \([0, 1]\)) of a particular crop and cultivar under a given input level.

[C#]
public static double GetManagementIndex(string crop, string cultivar, Scheme scheme, InputLevel input);

Exceptions:
GetOptimalRootingDepth

1. Gets the length (days) of CropCycle.CropGrowthPeriods of a reference crop (and cultivar). The length is expressed as a value range.

public static double GetOptimalRootingDepth(string crop);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop string is empty or the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

GetOptimalRootingDepthRange

1. Computes the management index (dimensionless, \([0, 1]\)) of a particular crop under a given input level.

public static double GetManagementIndex(string crop, Scheme scheme, InputLevel input);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is empty or a null reference or unknown; (2) scheme is not a valid member of ManagementIndex.Scheme; (3) group is not a valid member of ManagementIndex.ManagementGroup when scheme takes value CropSpecific.</td>
</tr>
</tbody>
</table>

3. Computes the management index of a particular ManagementGroup under a given input level, using a specific appraising ManagementIndex.Scheme.

public static double GetManagementIndex(Scheme scheme, ManagementGroup group, InputLevel input, bool leguminous);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) input is not a valid member of Input; (2) scheme is not a valid member of ManagementIndex.Scheme; (3) group is not a valid member of ManagementIndex.ManagementGroup when scheme takes value CropSpecific.</td>
</tr>
</tbody>
</table>

GetOptimalRootingDepth

1. Gets the optimal rooting depth (m) of a particular crop at the end of the CD stage.

public static double GetOptimalRootingDepth(string crop);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop string is empty or the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

GetOptimalRootingDepthRange

1. Gets the range (i.e., the minimum and maximum values) of the optimal rooting depth of a particular crop at the end of the CD stage.

public static double[] GetOptimalRootingDepthRange(string crop);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop string is empty or the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

GetRefCGP

Overloaded.

GetOptimalRootingDepth

1. Computes the management index (dimensionless, \([0, 1]\)) of a particular crop under a given input level.

public static double GetManagementIndex(string crop, Scheme scheme, InputLevel input);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is empty or a null reference or unknown; (2) crop and/or cultivar are/is unknown; in most cases, it is because that cultivar is not properly specified; (3) input is not a valid member of Input; (4) scheme is not a valid member of ManagementIndex.Scheme; (5) the ManagementIndex.ManagementGroup cannot be determined for the current crop and cultivar.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is empty or a null reference; or (2) crop and/or cultivar are/is unknown; in most cases, it is because that cultivar is not properly specified; (3) input is not a valid member of Input; (4) scheme is not a valid member of ManagementIndex.Scheme; (5) the ManagementIndex.ManagementGroup cannot be determined for the current crop and cultivar.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) input is not a valid member of Input; (2) scheme is not a valid member of ManagementIndex.Scheme; (3) group is not a valid member of ManagementIndex.ManagementGroup when scheme takes value CropSpecific.</td>
</tr>
</tbody>
</table>
2.6 CropData Structure

[C#] public static int[,] GetRefCGP(string crop, string cultivar, bool cropCycle);

Return Value: A jagged integer array that represents the value ranges of lengths (days) of CropCycle.CropGrowthPeriods of the crop and cultivar.

Remarks:
(a) The method gets the lengths of the following period: Establishment; Vegetative; Flowering; YieldFormation; Ripening.
(b) The total length of the CropCycle is included if cropCycle is true; not included if false.
(c) The length of a CropCycle.CropGrowthPeriod is expressed as a 2-element integer array, which is again an element of the jagged array returned. Length of the returned array is 5 or 6 depending on the cropCycle value.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is empty or a null reference; or (2) crop and/or cultivar are/is ambiguous; in most cases, it is because that cultivar is not properly specified.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database query returns nothing. In most cases, it is because that the crop (and cultivar) is not a reference or a user-defined crop.</td>
</tr>
</tbody>
</table>

2. Gets the length (days) of a particular CropCycle.CropGrowthPeriod of a reference crop (and cultivar). The length is expressed as a value range.

[C#] public static int[,] GetRefCGP(string crop, string cultivar, CropGrowthPeriod period);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is empty or a null reference; or (2) crop and/or cultivar are/is ambiguous; in most cases, it is because that cultivar is not properly specified; or (3) period is not one of the following: Establishment, Vegetative, Flowering, YieldFormation, Ripening.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database query returns nothing. In most cases, it is because that the crop (and cultivar) is not a reference or a user-defined crop.</td>
</tr>
</tbody>
</table>

- GetSoilWaterDepletionFraction
  Overloaded.

1. Gets the soil water depletion fraction ($p$), to which the crop can evapotranspirate at the maximum rate.

[C#] public static double GetSoilWaterDepletionFraction(double ETm);

Remarks:
(a) This method estimates the soil water depletion fraction ($p$) using the maximum crop evapotranspiration ($ET_m$, mm day$^{-1}$) data.
(b) The overloading GetSoilWaterDepletionFraction method is more crop-specific. Therefore $p$ should be estimated using the overload as long as data is available.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when $ET_m$ is negative.</td>
</tr>
</tbody>
</table>

2. Gets the soil water depletion fraction ($p$), to which the crop can evapotranspirate at the maximum rate.

[C#] public static double GetSoilWaterDepletionFraction(string crop, double ETm);
3. Gets the soil water depletion fraction ($\rho$), to which the crop can evapotranspire at the maximum rate.

```csharp
public static double GetSoilWaterDepletionFraction(string crop, string cultivar, double $ET_m$);
```

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>crop</code> is a null reference or empty; when $ET_m$ is negative; and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

- **GetYieldResponseFactor**
  - Overloaded.

  1. Gets the yield response factor ($k_y$) for the whole crop cycle of a particular `crop` and `cultivar`.

```csharp
public static double GetYieldResponseFactor(string crop, string cultivar);
```

  2. Gets the yield response factor ($k_y$) of a particular `crop` and `cultivar` during the given `CropCycle.CropGrowthPeriod`.

```csharp
public static double GetYieldResponseFactor(string crop, string cultivar, CropGrowthPeriod cgp);
```

  3. Gets the yield response factor ($k_y$) of a particular crop and cultivar during the given `CropCycle.CropGrowthPeriod` and sub period.

```csharp
public static double GetYieldResponseFactor(string crop, string cultivar, CropGrowthPeriod cgp, SubCGP earlyOrLate);
```

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>crop</code> is a null reference or empty.</td>
</tr>
</tbody>
</table>

- **GetYieldResponseFactorRange**
  - Overloaded.

  1. Gets the yield response factor ($k_y$) for the whole crop cycle of a particular `crop` and `cultivar`.

```csharp
public static double[] GetYieldResponseFactor(string crop, string cultivar);
```

  2. Gets the yield response factor ($k_y$) of a particular `crop` and `cultivar` during the given `CropCycle.CropGrowthPeriod`.

```csharp
public static double[] GetYieldResponseFactor(string crop, string cultivar, CropGrowthPeriod cgp);
```

  3. Gets the yield response factor ($k_y$) of a particular crop and cultivar during the given `CropCycle.CropGrowthPeriod` and sub period.

```csharp
public static double[] GetYieldResponseFactor(string crop, string cultivar, CropGrowthPeriod cgp, SubCGP earlyOrLate);
```

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>crop</code> is a null reference or empty.</td>
</tr>
</tbody>
</table>
2.7 Database Structure

2.7.1 Summary
Provides database accessing and updating facilities, including fetching data, inserting/updating records, and logging.

2.7.2 Declaration
[C#] public struct Database

2.7.3 Public Static Properties

- ConnectionStringIP2Country
  The readonly connection string for the IP2Country database.
  [C#] public static string ConnectionStringIP2Country { get; }

- ConnectionStringWLES
  The readonly connection string for the WLES database.
  [C#] public static string ConnectionStringWLES { get; }

2.7.4 Public Static Methods

- AddNewRefCrop
  Adds a new reference crop into the WLES database.
  [C#] public static int AddNewRefCrop(string crop, string cultivar, string alias);
  **Remarks:** A crop is defined by its name and cultivar which is optional. A crop may also have an alias. A unique ID is assigned to each reference crop in the WLES database.
  **Exceptions:**
<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
</table>
  | ArgumentException | Thrown when `crop` is a null reference or empty.
  | WlesDatabaseException | Thrown when the underlying database error occurs.

- FetchDataColumn
  Fetches a column of data from a database and returns them in an Object array.
  [C#] public static object[] FetchDataColumn(string sql, string connString);
  **Remarks:**
1. **FetchDataColumn** is different from other database-related methods. **ReadCell** reads one datum from the database and returns it as an **Object**; **ReadData** reads any data from the database (and returns the OleDbDataReader), but it depends on the caller to interpret the result and to close the connection to the database. **FetchDataColumn** not only reads data from the database, but also returns the data items in an **Object** array. Furthermore, it takes responsibility to properly close the database connection.

2. However, **FetchDataColumn** fetches only one column of data from the database. To be more accurate, it processes the 1st column of data and discards other columns if any.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database query returns nothing or when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

**Example:** The following example fetches the latitudes values which were stored in the database table named **Photosynthesis** and converts the resulting **Object** array to a **Double** array.

```csharp
using System;
using UGent.WLES.Data;

public class Photosynthesis {
    public static void Main(string[] args) {
        string sql, connString;
        sql = "SELECT DISTINCT latitude FROM Photosynthesis";
        connString = args[0];

        object[] temp;
        temp = Database.FetchDataColumn(sql, connString);

        double[] latitudes;
        latitudes = new double[temp.Length];
        for (int i=0; i<temp.Length; i++) {
            latitudes[i] = (double) temp[i];
            Console.Write("{0} ", latitudes[i]);
        }
        Console.WriteLine();
    }
}
```

- **Log**
  Overloaded.

1. Logs the unauthenticated Web client’s visit.
   ```csharp
   [C#] public static int Log(string usrIP, string page);
   ```

2. Logs the authenticated Web client’s visit.
   ```csharp
   [C#] public static int Log(string usrIP, string usrID, string page);
   ```

3. Logs the authenticated Web client’s visit, together with the error message in case an application error occurred.
   ```csharp
   [C#] public static int Log(string usrIP, string usrID, string page,
                            string errTitle, string errTrace);
   ```

**Return Value:** An integer exit code: ‘1’ for normal exit (1 record added to the database); ‘0’ if IP address passed in is not valid (database not updated).

**Remarks:**
2.7 Database Structure

1. The Web client is identified as an authenticated user. The userid together with the client’s remote IP address is logged to the database. If the client is not authenticated, only the client’s remote IP address is logged using the overloading method;

2. The database the method connects to is identified by `ConnectionStringWLES`.

**Exceptions:**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

- **ReadCell**

  Reads a single value from a database.

```csharp
public static object ReadCell(string sql, string connString);
```

**Return Value:** The database cell value which is boxed as an `Object`.

**Remarks:** This method reads a single value from a database which is specified by the connection string (`connString` parameter). The fetched value is wrapped in an `Object` instance and returned. The `Object` needs to be unboxed before the value can be used.

**Exceptions:**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database query returns nothing or when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

**Example:** The following example shows how to read a single value from a database. It also demonstrates the way to unbox the `Object` returned by this method.

```csharp
using System;
using System.Data;
using System.Data.OleDb;
using UGent.WLES.Data;

public class ReadDatabaseCell {
    public static void Main() {
        string connStr, sql;
        int lat = 35;
        int mon = Convert.ToByte(RomanDate.MonthOfYear.JANUARY);

        connStr = ConnectionStringWLES;
        sql = string.Format("SELECT daylength FROM DaylengthByLatitude WHERE latitude = {0} AND monthid = {1}", lat, mon);

        object result;
        result = ReadCell(sql, connStr);

        // unboxing
        float daylength = Convert.ToDouble(result);

        Console.WriteLine("The daylength of {0} degree N in {1} = {2}", lat, (RomanDate.MonthOfYear) mon, daylength);
    }
}
```

This code produces the following result.

```
The daylength of 35 degree N in JANUARY = 10.1
```
• ReadData
Reads data from the WLES database.

[C#]
public static OleDbDataReader ReadData(string sql, OleDbConnection activeConnection);

Return Value: An OleDbDataReader instance, or a null reference if the formal parameter sql does not contain valid SELECT statements.

Remarks: Only the SELECT commands can be used in the formal parameter sql.
OleDbDataReader.Close() and OleDbConnection.Close() methods need to be called in order, upon data reading task is finished.

Example: The following example shows how to read the Daylength data of a whole year, for locations on the 35 degree N latitude, from the WLES.DaylengthByLatitude table. It also demonstrates the proper way to close the database connection.

```csharp
using System;
using System.Data;
using System.Data.OleDb;
using UGent.WLES.Data;

public class ReadDaylength {
    public static void Main() {
        OleDbDataReader reader;
        string sqlCommand;
        OleDbConnection conn;
        try {
            conn = new OleDbConnection(ConnectionStringWLES);
            sqlCommand = "SELECT daylength FROM DaylengthByLatitude
            WHERE latitude = '35';";
            reader = ReadData(sqlCommand, conn);

            // Iterating to fetch data.
            Console.WriteLine("{0}	{1}", "Month", "Daylength");
            int monthid = 1;
            while (reader.Read()) {
                int intDaylengthOrdinal = reader.GetOrdinal("daylength");
                if (intDaylengthOrdinal != -1)
                    Console.WriteLine("{0}	{1}", monthid++, reader.GetValue(intDaylengthOrdinal));
            }
        } finally {
            // Close up!
            reader.Close();
            conn.Close();
        }
    }
}
```

This code produces the following result.

<table>
<thead>
<tr>
<th>Month</th>
<th>Daylength</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.1</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>
2.8 SoilData Structure

2.8.1 Summary
Manipulates soil data.

2.8.2 Declaration

\[\text{C\#]\]
\[
\text{public struct SoilData}
\]

2.8.3 Remarks
The SoilData structure provides common functionalities for soil data manipulation using public static properties and methods.

- It represents the soil ParticleDensity as a field and WRBSSoilGroup, SoilTextureClass, SoilStructure, SandClass, SoilFertilityCharacteristic and SoilDrainageClass as enumerations;
- It estimates the plant available soil water (PAW) using a pF curve or on the basis of the SoilTextureClass;
- It further revises the PAW estimation by taking the coarse fragments into account;
- As a result, the evaluation of the available soil water storage in the rooting zone is made possible;
- On the other hand, it rearranges the SoilProfile into equal sections. The value of a particular soil property for a given section may be extracted and a weighting factor may be attributed to each section;
- Upon request, the weighted average of the considered property can be recalculated up to a particular depth using the already-mentioned weighting factors of equal sections.

To represent the soil texture as topological entity, refer to the SoilTexture class and the UGent.WLES.Topo namespace: To represent a soil profile and its composing soil horizons, refer to the SoilProfile and the SoilHorizon classes; To manipulate crop data, refer to the CropData structure; To manipulate climatic data, refer to the ClimaticData structure.
2.8.4 Public Static Fields

- ParticleDensity
  Particle density in kg m\(^{-3}\).
  
  [C#] public const double ParticleDensity = 2650;

2.8.5 Public Static Methods

- ExtractSectionValue
  Recalculates the data value for a particular section in a SoilProfile.
  
  [C#] public static double ExtractSectionValue(double[][] horizons, double[] data, double[] section);

Remarks: A SoilProfile is consisted of SoilHorizons, and a soil property is defined with values in different horizons across the profile. Hence the value of the property is not readily available for a soil section which differs from a horizon.

This method extracts, or in other words, recalculates the property value for a particular section of the profile. The section may overlap multiple parts of adjacent horizons, or any sub or superset of horizons as long as they are continuous.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when one of the following occurs: (1) Horizons not continuous - the upper boundary of horizon ( i ) does not overlap the lower boundary of horizon ( i-1 ); (2) Horizon or section not properly defined - any horizon or the section is not defined by a 2-element Double array; or thickness is not positive; (3) Data missing or mis-matching - data and horizons arrays do not have the same length.</td>
</tr>
</tbody>
</table>

Example: The following example calculates the clay contents at depth ranges of 2-24, 24-48 and 48-72 cm in a profile with 7 horizons. The clay content for each horizon is given below:

<table>
<thead>
<tr>
<th>horizons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>depth (cm)</th>
<th>0-5</th>
<th>5-15</th>
<th>15-25</th>
<th>25-40</th>
<th>40-55</th>
<th>55-75</th>
<th>75-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>clay (w%)</td>
<td>16</td>
<td>26</td>
<td>43</td>
<td>33</td>
<td>25</td>
<td>45</td>
<td>12</td>
</tr>
</tbody>
</table>

using System;
using UGent.WLES;
using UGent.WLES.Data;

public class RecalSectionExample {
  public static void Main() {
    double [] clay = new double [] { 16, 26, 43, 33, 25, 45, 12, };
    double [][] horizons = new double [clay.Length][];
    horizons[0] = new double [] {0, 5};
    horizons[1] = new double [] {5, 15};
    horizons[3] = new double [] {25, 40};
    horizons[4] = new double [] {40, 55};
SoilData Structure

```csharp

double[][] sections = new double[3][];
sections[0] = new double[2] {2,24};

foreach (double[] section in sections) {
    double c;
    c = SoilData.ExtractSectionValue(horizons, clay, section);
    Console.WriteLine("Clay content {0,2}-{1,2} cm = {2} ",
                      section[0], section[1], Math.Round(c,2));
}
}
```

The example generates the following results:

Clay content: 2-24 cm = 31.59%
Clay content: 24-48 cm = 30.75%
Clay content: 48-72 cm = 39.17%

• GetPAW

Overloaded.

1. Calculates the plant available soil water (PAW) using a pF curve, % v/v or cm m⁻¹.
```csharp
public static double GetPAW(double fc, double wp);
```

2. Calculates the plant available soil water (PAW) using a pF curve, % v/v or cm m⁻¹.
```csharp
public static double GetPAW(double fc, double wp, double bd);
```

3. Estimates the plant available soil water (PAW) for a particular SoilTextureClass, % v/v or cm m⁻¹.
```csharp
public static double GetPAW(SoilTextureClass texture);
```

Remarks:

(a) Different soils have different water holding capacities. The maximum volume of water a saturated soil can hold after free drainage is termed as “field capacity”, while the minimum content of soil water on which a crop still survives is called “wilting point”. The volume of water between these two is therefore “plant available water”.

(b) A soil’s water holding capacity is closely related to its texture. Thus the PAW can be estimated using the texture class.

4. Estimates the plant available soil water (PAW) using neighboring SoilData.SoilTextureClasses, % v/v or cm m⁻¹.
```csharp
public static double GetPAW(SoilTextureClass[] textures);
```

Remarks:

- Soil texture classes are defined as polygons in the soil texture triangle (Figure 2.2). A point within a polygon is viewed as a member of the texture class that the polygon stands for. However, a point on the common border of two neighboring polygons is viewed as a member of both classes. Hence a point that overlaps a common node of three or more neighboring polygons is member of all the involving classes.
- Different soils have different water holding capacities. The maximum volume of water a saturated soil can hold after free drainage is termed as “field capacity”,
while the minimum content of soil water on which a crop still survives is called “wilting point”. The volume of water between these two is therefore “plant available water”.

- A soil’s water holding capacity is closely related to its texture. Thus the PAW can be estimated using the texture class.
- Refer to SoilData.SoilTextureClass enumeration for a list of predefined texture classes.

**Exceptions:**

<table>
<thead>
<tr>
<th>condition</th>
<th>exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <em>textures</em> array empty or a null reference.</td>
</tr>
</tbody>
</table>

2.9 **WlesDataSet Structure**

2.9.1 **Summary**

Maintains a runtime datastore between the UGent.WLES application and the UGent.WLES database. It not only collects data, checks data integrity, flushes data into the database, but also retrieves data from the database upon request.

2.9.2 **Declaration**

[C#] public struct WlesDataSet

2.9.3 **Remarks**

1. This structure acts as a data provider for the WLES application. It wraps the data with the setter and getter methods. Data are organized in categories such as locational data, climatic data, crop data, etc.

2. This structure should be initialized through its parameterless default constructor. Data must be set through its setters (such as SetLocationalData, SetClimaticData, etc.) before it can be retrieved.

3. A Boolean value property has been designed for each data category. Such a property, as IsClimaticDataOk or IsCropCycleOk, etc., may be used as a testing flag to determine whether a particular category/item of data had already been initialized properly (before it is retrieved).

2.9.4 **Examples**

**Example 1:** This hypothetical example show the standard ways to contract a WlesDataSet instance, and to set data to and to retrieve data from it.

For a more realistic example, see example 2.

    using System;
    using UGent.WLES;
    using UGent.WLES.RPP;

    ...
    // initialize
    WlesDataSet data = new WlesDataSet();
    // set data
    // Station Ukkel (50 N, 4 E) 105 m a.s.l.
    data.SetLocationalData("Brussels (Belgium)", 50, 4, 105);
2.9 WlesDataSet Structure

// double arrays of climatic parameters ...
data.SetClimaticData(...);

// test the flag before calling
if (IsClimaticDataOk) {
  // retrieve the decadely rainfall data in a year
  double[] rainfall;
  rainfall = data.GetRainfall();
  ...
}
...

**Example 2**: The following example constructs a `WlesDataSet` instance, and sets necessary data to it. The instance is then used to initialize a `Biomass` instance to evaluate the RPP.

```csharp
using System;
using UGent.Data;
using UGent.WLES;
using UGent.WLES.RPP;

public class DataSetExample {
    public static void Main() {
        // prepare the data
        // double latitude, elevation;
        string place;
        double[] tmin, tmax, RH, n, rain, wind;

        place = "Ngaoundere (Cameroon)";
        tmax = new double[] {30.1, 31.5, 31.2, 30.2, 28.5, 26.6, 25.8,
                               26, 27, 28, 3, 29.3, 30.5};
        tmin = new double[] {12.7, 13.5, 16.6, 17.3, 17.1, 17, 16.8, 16.7,
                               16, 15.7, 14, 2, 12};
        RH = new double[] {34.5, 35.7, 49.4, 65.1, 75.3, 78.1, 80.1, 81.3,
                               78.7, 74.1, 59.6, 43.1};
        wind = new double[] {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
        n = new double[] {8.81, 8.43, 7.01, 5.9, 5.86, 5.02, 3.75, 3.83,
                               4.38, 5.63, 8.49, 9.24};
        rain = new double[] {2, 2, 49, 134, 206, 237, 280, 300, 235, 146, 12,
                               2};
        latitude = 7 + 21.0/60.0;
        elevation = 1014;

        // the data set
        WlesDataSet data = new WlesDataSet();
        // set the locational data before climatic!
        data.SetLocationalData(place, latitude, elevation);
        // then the climatic
        data.SetClimaticData(tmax, tmin, RH, wind, n, rain);
        // reference evapotranspiration is derived from
        // climatic data automatically and not necessary to
        // set it separately, unless you have measured data

        // the crop and crop cycle data
        RomanDate cc1, cc2;
        cc1 = new RomanDate(4, 1);
    }
}
```
Namespace UGent.WLES.Data

```csharp
c = new RomanDate(8, 10);
string crop = "Soybean";
string cultivar = "";
double LAImax, hi;
LAImax = CropData.GetLeafAreaIndex(crop);
hi = CropData.GetHarvestIndex(crop, cultivar);
data.SetCropData(crop, cultivar, LAImax, hi);

// crop development stages: (days)
// IS = 20, CD = 30, MS = 40, LS = 40
data.SetCropCycle(cc1, 20, 30, 40, 40);
```

```csharp
// initialize Biomass class with the dataset
Biomass biomass = new Biomass(data);
biomass.Evaluate();

// the report
Console.WriteLine(biomass);
```

This example generates the following result:

Soybean in Ngaoundere (Cameroon)
Crop cycle: Apr 01 (WLES Calendar) - Aug 10 (WLES Calendar), 130 days
RPP: 3558.99 kg/ha

2.9.5 Public Instance Properties

- IsClimaticDataOk
  A Boolean that indicates the status of the climatic data: `true` if the data has been set; `false` otherwise.
  
```csharp
  public bool IsClimaticDataOk { get; }
  ```

- IsCropCycleOk
  A Boolean that indicates the status of the crop cycle data: `true` if the data has been set; `false` otherwise.
  
```csharp
  public bool IsCropCycleOk { get; }
  ```

- IsCropDataOk
  A Boolean that indicates the status of the crop data: `true` if the data has been set; `false` otherwise.
  
```csharp
  public bool IsCropDataOk { get; }
  ```

- IsCropGrowthPeriodDataOk
  A Boolean that indicates the status of the CropCycle.CropGrowthPeriod data: `true` if the data has been set; `false` otherwise.
  
```csharp
  public bool IsCropGrowthPeriodDataOk { get; }
  ```

- IsEToDataOk
  A Boolean that indicates the status of the ET₀ data: `true` if the data has been set; `false` otherwise.
  
```csharp
  public bool IsEToDataOk { get; }
  ```

- IsLocationalDataOk
  A Boolean that indicates the status of the locational data: `true` if the data has been set; `false` otherwise.
  
```csharp
  public bool IsLocationalDataOk { get; }
  ```
2.9.6 Public Instance Methods

- Flush
  Flushes data into the UGent.WLES database.
  ```csharp
  public bool Flush();
  ```

- GetAlgorithm
  Gets the algorithm used to compute the soil index from the ratings of the selected
  SoilData.SoilCharacteristics.
  ```csharp
  public Algorithm GetAlgorithm();
  ```
  **Exceptions:**
  ```text
  exception   condition
  WlesException Thrown when the soil data were not set upon call.
  ```

- GetAppraisingScheme
  Gets the appraising scheme to apply in management index determination.
  ```csharp
  public Scheme GetAppraisingScheme();
  ```
  **Exceptions:**
  ```text
  exception   condition
  WlesException Thrown when the management data were not set upon call.
  ```

- GetCrop
  Gets a String that represents the crop name.
  ```csharp
  public string GetCrop();
  ```
  **Exceptions:**
  ```text
  exception   condition
  WlesException Thrown when the crop data were not set upon call.
  ```

- GetCropCycleBeginningRomanDate
  Gets a RomanDate that represents the start of the crop cycle.
  ```csharp
  public RomanDate GetCropCycleBeginningRomanDate();
  ```
  **Exceptions:**
  ```text
  exception   condition
  WlesException Thrown when the crop data were not set upon call.
  ```

- GetCropCycleEndingRomanDate
  Gets a RomanDate that represents the end of the crop cycle.
  ```csharp
  public RomanDate GetCropCycleEndingRomanDate();
  ```
  **Exceptions:**
  ```text
  exception   condition
  WlesException Thrown when the crop data were not set upon call.
  ```
• GetCropCycleRomanDates
  Gets a RomanDate array that represents the start and the end of the crop cycle.

  [C#] public RomanDate[] GetCropCycleRomanDates();

  Exceptions:
  exception condition
  WlesException Thrown when the crop data were not set upon call.

• GetCropDevelopmentStageRomanDates
  Gets a RomanDate array that represents the beginning and the ending dates of a particular CropCycle.CropDevelopmentStage.

  [C#] public RomanDate[] GetCropDevelopmentStageRomanDates(CropDevelopmentStage stage);

  Exceptions:
  exception condition
  WlesException Thrown when the crop data were not set upon call.

• GetCropDevelopmentStages
  Gets a 2-dimensional RomanDate array that represents the beginning and the ending dates of IS, CD, MS and LS stages.

  [C#] public RomanDate[,] GetCropDevelopmentStages();

  Return Value: A 2-dimensional RomanDate array that contains the beginning and the ending dates of IS, CD, MS and LS stages. The array is 4 by 2 in dimension.

  Exceptions:
  exception condition
  WlesException Thrown when the crop data were not set upon call.

• GetCropGrowthPeriodLength
  Overloaded.

  1. Gets the lengths (days) of the following periods: Establishment, Vegetative, Flowering, YieldFormation, and Ripening.

     [C#] public int[] GetCropGrowthPeriodLength();

     Return Value: A 5-element integer array that represents the lengths (days) of CropCycle.CropGrowthPeriods.

     Exceptions:
     exception condition
     WlesException Thrown when the crop growth data were not set upon call.

  2. Gets the length (days) of a particular CropCycle.CropGrowthPeriod.

     [C#] public int GetCropGrowthPeriodLength(CropGrowthPeriod period);

     Return Value: An integer that represents the length (days) of CropCycle.CropGrowthPeriod period. -1 if period value out of range.

     Exceptions:
     exception condition
     WlesException Thrown when the crop growth data were not set upon call.

• GetCultivar
  Gets a String that represents the cultivar name.

  [C#] public string GetCultivar();

  Exceptions:
  exception condition
  WlesException Thrown when the crop data were not set upon call.

• GetDataResolution
  Gets the WlesDataSet.TemporalResolution of the climatic data.

  [C#] public TemporalResolution GetDataResolution();
Exceptions:

- WlesException Thrown when the climatic data were not set upon call.

GetElevation

- Gets a Double value that represents the elevation (m, above sea level).

[CS#]

```csharp
public double GetElevation();
```

Exceptions:

- WlesException Thrown when the locational data were not set upon call.

GetHarvestIndex

- Gets a Double that represents the harvest index (dimensionless, 0-1).

[CS#]

```csharp
public double GetHarvestIndex();
```

Exceptions:

- WlesException Thrown when the crop data were not set upon call.

GetInputLevel

- Gets the input level that represents the efficiency of field management practices.

[CS#]

```csharp
public InputLevel GetInputLevel();
```

Exceptions:

- WlesException Thrown when the management data were not set upon call.

GetLAI

- Gets a Double that represents the leaf area index at the maximum crop growth rate (m² m⁻²).

[CS#]

```csharp
public double GetLAI();
```

Exceptions:

- WlesException Thrown when the management data were not set upon call.

GetLatitude

- Gets a Double that represents the latitude (°).

[CS#]

```csharp
public double GetLatitude();
```

Remarks: Northern Hemishphere locations have positive latitudes, and Southern Hemisphere negative.

Exceptions:

- WlesException Thrown when the locational data were not set upon call.

GetLimitingLayerDepth

- Gets a Double that represents the depth (m) of the root limiting layer in the soil.

[CS#]

```csharp
public double GetLimitingLayerDepth();
```

Return Value: A Double that represents the depth (m) of the limiting layer in the soil; -1 if no limiting layer is presented.

Remarks:

1. This method returns the depth of the root limiting layer in the soil. As a matter of fact, the dataset does not store the limiting layer depth value by itself. Instead the value is attributed to the soil profile that is associated with the dataset. Therefore make sure that the LimitingLayerDepth property of the SoilProfile class is properly initialized before calling this method, or value -1 will be returned.

2. Refer to GetMaximumRootingDepth for the difference of the terms of the optimal depth and the limiting layer.
3. This method is implemented with the `SoilProfile.LimitingLayerDepth` property.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the soil data were not set upon call.</td>
</tr>
</tbody>
</table>

- **GetLongitude**
  Gets a Double that represents the longitude (°).

  ```csharp
  [C#]
  public double GetLongitude();
  ```

  **Remarks:** Eastern Hemisphere locations have positive longitudes, and Western Hemisphere negative.

  **Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the locational data were not set upon call.</td>
</tr>
</tbody>
</table>

- **GetMaximumAirTemperature**
  Gets a Double array that represents the year-round maximum air temperature (°C).

  ```csharp
  [C#]
  public double[] GetMaximumAirTemperature();
  ```

  **Remarks:**

  1. The TemporalResolution of the returned array is DAILY or DECADELY, depending the data array fed to the `SetClimaticData` method.
  2. MONTHLY data were automatically transformed into DECADELY data, using the interpolation algorithm implemented in `GetDecadeData`.
  3. The difference between this method and the `GetMaximumAirTemperatureDailyArray` method is that the latter simply rewrites the decadely data in the daily format.

  **Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the climatic data were not set upon call.</td>
</tr>
</tbody>
</table>

- **GetMaximumAirTemperatureDailyArray**
  Gets a Double array that represents the year-round daily maximum air temperature (°C).

  ```csharp
  [C#]
  public double[] GetMaximumAirTemperatureDailyArray();
  ```

  **Return Value:** A Double array that represents the daily maximum air temperature (°C). Array length depends on the working calendar associated with current dataset: 360 for `WlesCalendar` and 365 for `GregorianCalendar`.

- **GetMaximumRootingDepth**
  Gets a Double that represents the maximum rooting depth (m), taking both optimal rooting depth of the crop and the depth of the soil limiting layer into account.

  ```csharp
  [C#]
  public double GetMaximumRootingDepth();
  ```

  **Remarks:** The maximum rooting depth depends on two parameters:

  1. Optimal rooting depth - For reference crops, the optimal rooting depth has already been defined in the WLES database. Measured rooting depth data can be used instead of the predefined database data when setting the crop data (`SetCropData`). For user defined crops, the rooting depth data has to be provided by the user.
  2. Soil restriction layer - A soil restriction layer is also called soil limiting layer. Such a layer prevents the rooting from penetrating into deeper horizons. The depth of the restriction layer is included in the soil dataset (`SetSoilData`), if it is presented.

  **Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the crop data were not set upon call.</td>
</tr>
</tbody>
</table>
2.9 WlesDataSet Structure

- **GetMeanAirTemperature**
  Gets a `Double` array that represents the year-round mean air temperature (°C).

```csharp
public double[] GetMeanAirTemperature();
```

**Remarks:**

1. The `TemporalResolution` of the returned array is DAILY or DECADELY, depending the data array fed to the `SetClimaticData` method.
2. MONTHLY data were automatically transformed into DECADELY data, using the interpolation algorithm implemented in `GetDecadeData`.
3. The difference between this method and the `GetMeanAirTemperatureDailyArray` method is that the latter simply rewrites the decadely data in the daily format.

**Exceptions:**

- WlesException  Thrown when the climatic data were not set upon call.

- **GetMeanAirTemperatureDailyArray**
  Gets a `Double` array that represents the year-round daily mean air temperature (°C).

```csharp
public double[] GetMeanAirTemperatureDailyArray();
```

**Return Value:** A `Double` array that represents the daily mean air temperature (°C).

Array length depends on the working calendar associated with current dataset: 360 for `WlesCalendar` and 365 for `GregorianCalendar`.

- **GetOptimalDepth**
  Gets a `Double` that represents the optimal rooting depth (m) of a crop.

```csharp
public double GetOptimalDepth();
```

**Remarks:** Refer to `GetMaximumRootingDepth` for the difference of the terms of the optimal depth and the limiting layer.

- **GetPlace**
  Gets a `String` that represents the name of the study area.

```csharp
public string GetPlace();
```

- **GetRainfall**
  Gets a `Double` array that represents the year-round rainfall (mm day\(^{-1}\) or mm decade\(^{-1}\)).

```csharp
public double[] GetRainfall();
```

**Remarks:**

1. The `TemporalResolution` of the returned array is DAILY or DECADELY, depending the data array fed to the `SetClimaticData` method.
2. MONTHLY data were automatically transformed into DECADELY data, using the interpolation algorithm implemented in `GetDecadeData`.
3. The difference between this method and the `GetRainfallDailyArray` method is that the latter simply rewrites the decadely data in the daily format.

**Exceptions:**

- WlesException  Thrown when the climatic data were not set upon call.
• GetRainfallDailyArray
  Gets a Double array that represents the year-round daily rainfall (mm day\(^{-1}\)).
  
  \texttt{\texttt{public double\[\] GetRainfallDailyArray();}}
  \textit{Return Value:} A Double array that represents the daily rainfall (mm day\(^{-1}\)). Array length depends on the working calendar associated with current dataset: 360 for \texttt{WlesCalendar} and 365 for \texttt{GregorianCalendar}.

• GetReferenceEvapotranspiration
  Gets a Double array that represents the year-round reference evapotranspiration (mm day\(^{-1}\) or mm decade\(^{-1}\)).
  
  \texttt{\texttt{public double\[\] GetReferenceEvapotranspiration();}}
  \textit{Remarks:}
  
  1. The \texttt{TemporalResolution} of the returned array is \texttt{DAILY} or \texttt{DECADELY}, depending the data array fed to the \texttt{SetClimaticData} method.
  2. \texttt{MONTHLY} data were automatically transformed into \texttt{DECADELY} data, using the interpolation algorithm implemented in \texttt{GetDecadeData}.
  3. The difference between this method and the \texttt{GetRainfallDailyArray} method is that the latter simply \texttt{rewrites} the decadely data in the daily format.

\textit{Exceptions:}

\begin{tabular}{ll}
\texttt{exception} & \texttt{condition}\\
\texttt{WlesException} & Thrown when the reference evapotranspiration data were not set upon call.
\end{tabular}

• GetReferenceEvapotranspirationDailyArray
  Gets a Double array that represents the year-round daily evapotranspiration (mm day\(^{-1}\)).
  
  \texttt{\texttt{public double\[\] GetReferenceEvapotranspirationDailyArray();}}
  \textit{Return Value:} A Double array that represents the daily reference evapotranspiration (mm day\(^{-1}\)). Array length depends on the working calendar associated with current dataset: 360 for \texttt{WlesCalendar} and 365 for \texttt{GregorianCalendar}.

• GetRelativeAirHumidity
  Gets a Double array that represents the year-round relative air humidity (%).
  
  \texttt{\texttt{public double\[\] GetRelativeAirHumidity();}}
  \textit{Remarks:}
  
  1. The \texttt{TemporalResolution} of the returned array is \texttt{DAILY} or \texttt{DECADELY}, depending the data array fed to the \texttt{SetClimaticData} method.
  2. \texttt{MONTHLY} data were automatically transformed into \texttt{DECADELY} data, using the interpolation algorithm implemented in \texttt{GetDecadeData}.
  3. The difference between this method and the \texttt{GetRainfallDailyArray} method is that the latter simply \texttt{rewrites} the decadely data in the daily format.

\textit{Exceptions:}

\begin{tabular}{ll}
\texttt{exception} & \texttt{condition}\\
\texttt{WlesException} & Thrown when the climatic data were not set upon call.
\end{tabular}

• GetRelativeAirHumidityDailyArray
  Gets a Double array that represents the year-round daily relative humidity (%).
  
  \texttt{\texttt{public double\[\] GetRelativeAirHumidityDailyArray();}}
  \textit{Return Value:} A Double array that represents the daily air humidy (%) . Array length depends on the working calendar associated with current dataset: 360 for \texttt{WlesCalendar} and 365 for \texttt{GregorianCalendar}.

• GetSoilHorizons
  Gets an array of \texttt{SoilHorizon} that represents the horizons of the soil.
2.9 WlesDataSet Structure

[C#] public SoilHorizon[] GetSoilHorizons();

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the soil data were not set upon call.</td>
</tr>
</tbody>
</table>

- GetSoilProfile
  Gets a SoilProfile that represents the soil data.

[C#] public SoilProfile GetSoilProfile();

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the soil data were not set upon call.</td>
</tr>
</tbody>
</table>

- GetSunshineHours
  Gets a Double array that represents the year-round sunshine hours (hr day\(^{-1}\) or hr decade\(^{-1}\)).

[C#] public double[] GetSunshineHours();

Remarks:

1. The TemporalResolution of the returned array is DAILY or DECADELY, depending the data array fed to the SetClimaticData method.
2. MONTHLY data were automatically transformed into DECADELY data, using the interpolation algorithm implemented in GetDecadeData.
3. The difference between this method and the GetRainfallDailyArray method is that the latter simply rewrites the decadely data in the daily format.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the climatic data were not set upon call.</td>
</tr>
</tbody>
</table>

- GetSunshineHoursDailyArray
  Gets a Double array that represents the year-round daily sunshine hours (hr day\(^{-1}\)).

[C#] public double[] GetSunshineHoursDailyArray();

Return Value: A Double array that represents the daily sunshine hours (hr day\(^{-1}\)). Array length depends on the working calendar associated with current dataset: 360 for WlesCalendar and 365 for GregorianCalendar.

- GetWettingIntervalIS
  Gets the wetting interval value of the IS stage.

[C#] public double GetWettingIntervalIS();

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the wetting interval was not set upon call.</td>
</tr>
</tbody>
</table>

- GetWindSpeedAt2M
  Gets a Double array that represents the year-round wind velocity (m s\(^{-1}\)).

[C#] public double[] GetWindSpeedAt2M();

Remarks:

1. The TemporalResolution of the returned array is DAILY or DECADELY, depending the data array fed to the SetClimaticData method.
2. MONTHLY data were automatically transformed into DECADELY data, using the interpolation algorithm implemented in GetDecadeData.
3. The difference between this method and the GetRainfallDailyArray method is that the latter simply rewrites the decadely data in the daily format.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the climatic data were not set upon call.</td>
</tr>
</tbody>
</table>
• GetWindSpeedAt2MDailyArray
  Gets a Double array that represents the year-round daily wind velocity (m s\(^{-1}\)).
  
  ```csharp
  public double[] GetWindSpeedAt2MDailyArray();
  ```
  
  **Return Value:** A Double array that represents the daily wind velocity (m s\(^{-1}\)). Array length depends on the working calendar associated with current dataset: 360 for WlesCalendar and 365 for GregorianCalendar.

• GetWorkingCalendar
  Gets the working calendar of current WlesDataSet.
  
  ```csharp
  public CalendarType GetWorkingCalendar();
  ```

• HasLimitingLayer
  Determines whether a root restriction layer is present in the soil.
  
  ```csharp
  public bool HasLimitingLayer();
  ```

• IsHumidClimate
  Determines whether the climate is humid.
  
  ```csharp
  public bool IsHumidClimate();
  ```
  
  **Return Value:** true if the annual total of the rainfall exceeds half of the annual total of the reference evapotranspiration; false otherwise.
  
  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the climatic data (and thus the reference evapotranspiration data) are not set.</td>
</tr>
</tbody>
</table>

• SetClimaticData
  Overloaded.

  1. Sets the climatic data.
  
  ```csharp
  public void SetClimaticData(double[] tmean, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine, double[] rainfall, CalendarType calendar);
  ```

  2. Sets the climatic data.
  
  ```csharp
  public void SetClimaticData(double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine, double[] rainfall, CalendarType calendar);
  ```

  **Remarks:**

  1. The method sets the climatic data and automatically calculates \(ET_o\) using procedures defined in CalculateETo method.

  2. The SetLocationalData method should be called prior to this method.

  **Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when this method is called prior to SetLocationalData method.</td>
</tr>
</tbody>
</table>

• SetCropCycle
  Overloaded.

  1. Sets the beginning and ending dates of the CropCycle and of its CropCycle.CropDevelopmentStages.
  
  ```csharp
  public void SetCropCycle(RomanDate start, int lenIS, int lenCD, int lenMS, int lenLS);
  ```
2.9 WlesDataSet Structure

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the length of one or more of the crop development stages is not positive.</td>
</tr>
<tr>
<td>WlesException</td>
<td>Thrown when the length of the crop cycle not confirmative to the sum of the lengths of crop development stages.</td>
</tr>
</tbody>
</table>

2. Sets the beginning and ending dates of the CropCycle and of its CropCycle.CropDevelopmentStages.

[C#]
```csharp
public void SetCropCycle(RomanDate start, int[] lenStages);
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when lenStages array does not contain 4 elements; and when any element value is not positive.</td>
</tr>
<tr>
<td>WlesException</td>
<td>Thrown when the length of the crop cycle not confirmative to the sum of the lengths of crop development stages.</td>
</tr>
</tbody>
</table>


[C#]
```csharp
public void SetCropCycle(RomanDate start, RomanDate cd1, RomanDate ms1, RomanDate ls1, RomanDate end);
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the length of the crop cycle not confirmative to the sum of the lengths of crop development stages.</td>
</tr>
</tbody>
</table>

- SetCropData
  Overloaded.

  1. Sets the reference crop data.

  [C#]
  ```csharp
  public void SetCropData(string crop, string cultivar);
  ```

  Remarks: The difference between this method and its overloads:

  (a) This method sets a reference crop, or in other words, a known crop. Hence the crop name and cultivar are enough and all the rest data will be taken from database automatically.

  (b) The overloading SetCropData methods is generally preferred over this one, since it takes all necessary data entry as a parameter. Hence it works for a user-defined crop and a reference crop.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is a null reference or empty.</td>
</tr>
</tbody>
</table>

  2. Sets the crop data.

  [C#]
  ```csharp
  public void SetCropData(string crop, string cultivar, double laimax, double hi, double depth);
  ```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is a null reference or empty; (2) laimax is negative or greater than an arbitrarily chosen value of 10 m² m⁻²; (3) hi is out of range [0, 1]; or (4) depth is negative or greater than an arbitrarily chosen value of 5 m.</td>
</tr>
</tbody>
</table>

- SetCropGrowthPeriods
  Sets the CropCycle.CropGrowthPeriods data.

  [C#]
  ```csharp
  public void SetCropGrowthPeriods(int[] cgp, bool autoAdjust);
  ```

Parameters:
cgp A 5-element integer array that contains lengths (days) of the following periods: Establishment, Vegetative, Flowering, YieldFormation, and Ripening.

autoAdjust true to automatically adjust the lengths of the CropGrowthPeriods contained in array cgp so that the sum of them equals the length of the CropCycle.

Remarks:

1. This method should be called after the SetCropData and the SetCropCycle methods have been called.

2. The sum of the lengths of the CropCycle.CropGrowthPeriods should equal to the length of the CropCycle. This is guaranteed by this method when autoAdjust is true; it is the caller's responsibility to guarantee that when autoAdjust is false.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when length of array cgp does not equal to 5; or sum of lengths of CropCycle.CropGrowthPeriods not same as length of crop cycle.</td>
</tr>
<tr>
<td>WlesException</td>
<td>Thrown when autoAdjust is true and crop cycle data has not been set.</td>
</tr>
</tbody>
</table>

- SetEToData
  Sets the reference evapotranspiration (mm day$^{-1}$ or mm decade$^{-1}$).

[C#]
```csharp
public void SetEToData(double[] eto);
```

Remarks: Although reference evapotranspiration is automatically set with the climatic data (SetClimaticData), this method provides a possibility to replace the derived $ET_o$ data with, for instance, measured data.

- SetLocationalData
  Overloaded.

  1. Sets the locational data.

  [C#]
  ```csharp
  public void SetLocationalData(string place, double latitude, double elevation);
  ```

  2. Sets the locational data.

  [C#]
  ```csharp
  public void SetLocationalData(string place, double latitude, double longitude, double elevation);
  ```

- SetManagementData
  Sets the management data.

  [C#]
  ```csharp
  public void SetManagementData(Scheme scheme, InputLevel input);
  ```

Parameters:

scheme A ManagementIndex.Scheme member that represents the scheme used to appraise the index.

input A ManagementIndex.InputLevel member that represents the input level under which the field management is carried out.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when input is not a valid member of ManagementIndex.InputLevel; and when scheme is not a valid member of ManagementIndex.Scheme.</td>
</tr>
</tbody>
</table>

- SetSoilData
  Overloaded.
2.10 ClimaticData.ClimaticParamSet Enumeration

2.10.1 Summary
Specifies the categories of the climatic parameters.

2.10.2 Declaration

[C#] public enum ClimaticData.ClimaticParamSet
2.10.3 Remarks

The `ClimaticParamSet` enumeration represents the categories of the climatic parameters used in the UGent.WLES application.

This enumeration is useful when it is desirable to have a strongly typed specification of the categories of the climatic parameters. For example, this enumeration is among the formal parameters of the `IsAccumulative`, `IsAverage` and `GetDecadeData` methods.

2.10.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAYLENGTH</td>
<td>Represents daylength, which is function of location and month of year.</td>
</tr>
<tr>
<td>EVAPOTRANSPIRATION</td>
<td>Represents evapotranspiration.</td>
</tr>
<tr>
<td>HUMIDITY</td>
<td>Represents humidity.</td>
</tr>
<tr>
<td>RADIATION</td>
<td>Represents radiation.</td>
</tr>
<tr>
<td>RAINFALL</td>
<td>Represents rainfall.</td>
</tr>
<tr>
<td>SUNSHINEHOURS</td>
<td>Represents sunshine hours, or duration of clear sky.</td>
</tr>
<tr>
<td>TEMPERATURE</td>
<td>Represents air temperature.</td>
</tr>
<tr>
<td>WINDSPEED</td>
<td>Represents wind speed.</td>
</tr>
</tbody>
</table>

2.11 `ClimaticData.HemisphereOfEarth` Enumeration

2.11.1 Summary

Specifies the hemispheres of the Earth.

2.11.2 Declaration

[C#] public enum ClimaticData.HemisphereOfEarth

2.11.3 Remarks

The `HemisphereOfEarth` enumeration represents the geographical locations in the Northern and Southern Hemispheres of the Earth.

This enumeration is useful when it is desirable to have a strongly typed specification of the Hemispheres of the Earth. For example, this enumeration is one of the formal parameters of the `InterpolateDaylength` method.

Note The Eastern and Western Hemispheres are not represented in this enumeration.

2.11.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORTHERN</td>
<td>The Northern Hemisphere.</td>
</tr>
<tr>
<td>SOUTHERN</td>
<td>The Southern Hemisphere.</td>
</tr>
</tbody>
</table>

2.12 `ClimaticData.TemperatureRegime` Enumeration

2.12.1 Summary

Specifies the temperature regimes, namely, CELSIUS and FAHRENHEIT.

2.12.2 Declaration

[C#] public enum ClimaticData.TemperatureRegime
2.13 CropData.FarmingSystem Enumeration

2.13.1 Summary
Represents the rainfed and the irrigated farming systems.

2.13.2 Declaration
[C#] public enum CropData.FarmingSystem.

2.13.3 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainfed</td>
<td>Represents rainfed farming.</td>
</tr>
<tr>
<td>Irrigated</td>
<td>Represents irrigated farming.</td>
</tr>
</tbody>
</table>

2.14 SoilData.GravelType Enumeration

2.14.1 Summary
Represents the types of dominant gravel.

2.14.2 Declaration
[C#] public enum SoilData.GravelType.

2.14.3 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quartz</td>
<td>Quartz (SiO₂) gravel.</td>
</tr>
<tr>
<td>IronOxide</td>
<td>Laterite gravel.</td>
</tr>
<tr>
<td>RockFragments</td>
<td>Rock fragments.</td>
</tr>
</tbody>
</table>

2.15 SoilData.LandSuitabilityClass Enumeration

2.15.1 Summary
Represents the land suitability classes.

2.15.2 Declaration
[C#] public enum SoilData.LandSuitabilityClass.

2.15.3 Remarks
There are 5 classes defined under 2 orders in the system of land capability (suitability) classification. The 2 orders are:

- Order ‘S’ - Suitable land: land on which sustained use for defined purpose in the defined manner is expected to yield benefits that will justify required recurrent inputs without unacceptable risks to land resources on the site or in adjacent areas;
- Order ‘N’ - Unsuitable land: land having characteristics which appear to preclude its sustained use for the defined purpose in the defined manner or which would create production, unkeep and/or conservation problems requiring a level of recurrent inputs unacceptable at the time of the interpretation.

### 2.15.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>The suitable class.</td>
</tr>
<tr>
<td>S2</td>
<td>The moderately suitable class.</td>
</tr>
<tr>
<td>S3</td>
<td>The marginally suitable class.</td>
</tr>
<tr>
<td>N1</td>
<td>The actually unsuitable, but potentially suitable class.</td>
</tr>
<tr>
<td>N2</td>
<td>The actually and potentially unsuitable class.</td>
</tr>
</tbody>
</table>

### 2.16 SoilData.SandClass Enumeration

#### 2.16.1 Summary

Represents the dominant subclass of sands according to the percentage distribution of sand particles.

#### 2.16.2 Declaration

[C#] public enum SoilData.SandClass

#### 2.16.3 Remarks

Sands are usually grouped into five categories, namely very fine sand, fine sand, medium sand, coarse sand and very coarse sand, on the basis of the average diameter of the sand particles (FAO, 1990). However, WLES adopts three classes for its own use.

<table>
<thead>
<tr>
<th>Sand</th>
<th>Diameter (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very fine sand</td>
<td>0.063-0.125</td>
</tr>
<tr>
<td>Fine sand</td>
<td>0.125-0.2000</td>
</tr>
<tr>
<td>Medium sand</td>
<td>0.200-0.630</td>
</tr>
<tr>
<td>Coarse sand</td>
<td>0.630-1.250</td>
</tr>
<tr>
<td>Very coarse sand</td>
<td>1.250-2.000</td>
</tr>
</tbody>
</table>

#### 2.16.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FineSand</td>
<td>(1) Fine sand: &gt; 50%; (therefore clay, silt and other sands: &lt; 50%); or (2) Coarse, very coarse and medium sand: &lt; 25%; very fine sand: &lt; 50%; (therefore clay, silt and fine sand: &gt; 25%).</td>
</tr>
<tr>
<td>Sand</td>
<td>Very coarse, coarse and medium sand: &gt; 25%; and fine and very fine sand: &lt; 50% (clay and silt take the rest percentage).</td>
</tr>
<tr>
<td>CoarseSand</td>
<td>Very coarse and coarse sand: &gt; 25%; and any other one grade of sand: &lt; 50% (clay and silt take the rest percentage).</td>
</tr>
</tbody>
</table>

### 2.17 SoilData.SoilCharacteristic Enumeration

#### 2.17.1 Summary

Specifies the soil characteristics that are included in the crop requirement tables.
2.17 SoilData.SoilCharacteristic Enumeration

2.17.2 Declaration

[C#] public enum SoilData.SoilCharacteristic

2.17.3 Remarks

The soil characteristics that are represented in this enumeration can be divided into 3 subcategories, namely the physical soil characteristics, the fertility characteristics, and salinity and alkalinity, as summarized below.

1. Physical soil characteristics:
   - Texture/structure
   - Coarse fragments
   - Soil depth
   - Calcium carbonate (CaCO₃)
   - Gypsum (CaSO₄·2H₂O)

2. Soil fertility characteristics:
   - Apparent CEC
   - Base saturation
   - Sum of basic cations
   - pH H₂O
   - Organic carbon

3. Salinity and alkalinity:
   - Electrical conductivity
   - Exchangeable sodium percentage

2.17.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texture</td>
<td>Represents the soil texture/structure characteristics, expressed in</td>
</tr>
<tr>
<td></td>
<td>SoilData.SoilTextureClasses.</td>
</tr>
<tr>
<td>CF</td>
<td>Represents the soil coarse fragments content, expressed in volumetric</td>
</tr>
<tr>
<td></td>
<td>percentages (% v/v).</td>
</tr>
<tr>
<td>SoilDepth</td>
<td>Represents the effective soil depth. The effective depth of a soil for plant</td>
</tr>
<tr>
<td></td>
<td>growth is the vertical distance into the soil from the surface to a layer</td>
</tr>
<tr>
<td></td>
<td>that essentially stops the downward growth of plant roots, expressed in</td>
</tr>
<tr>
<td></td>
<td>meters (m).</td>
</tr>
<tr>
<td>CaCO3</td>
<td>Represents the soil calcium carbonate content, expressed in weight</td>
</tr>
<tr>
<td></td>
<td>percentages (% w/w).</td>
</tr>
<tr>
<td>Gypsum</td>
<td>Represents the soil gypsum (CaSO₄·2H₂O) content, expressed in weight</td>
</tr>
<tr>
<td></td>
<td>percentages (% w/w).</td>
</tr>
<tr>
<td>ACEC</td>
<td>Represents the apparent CEC, expressed in centimoles of charge per kilogram</td>
</tr>
<tr>
<td></td>
<td>of clay [cmol(+) kg⁻¹ clay].</td>
</tr>
<tr>
<td>BS</td>
<td>Represents the base saturation, expressed in percentages (%).</td>
</tr>
<tr>
<td>SumBC</td>
<td>Represents the sum of basic cations, expressed in centimoles of charge per</td>
</tr>
<tr>
<td></td>
<td>kilogram of soil [cmol(+) kg⁻¹ soil].</td>
</tr>
<tr>
<td>pHH₂O</td>
<td>Represents the soil pH in water, dimensionless.</td>
</tr>
<tr>
<td>OC</td>
<td>Represents the organic carbon content, expressed in weight percentages (%).</td>
</tr>
<tr>
<td>EC</td>
<td>Represents the electrical conductivity, expressed in deciSiemens per meter</td>
</tr>
<tr>
<td></td>
<td>(dS m⁻¹).</td>
</tr>
<tr>
<td>ESP</td>
<td>Represents the exchangeable sodium percentage (%).</td>
</tr>
</tbody>
</table>
2.18 SoilData.SoilDrainageClass Enumeration

2.18.1 Summary
Represents the soil drainage class.

2.18.2 Declaration

```csharp
public enum SoilData.SoilDrainageClass
```

2.18.3 Remarks
The following drainage classes are defined (Schoeneberger et al., 2002):

<table>
<thead>
<tr>
<th>class</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Poorly Drained</td>
<td>VP</td>
</tr>
<tr>
<td>Poorly Drained</td>
<td>PD</td>
</tr>
<tr>
<td>Somewhat Poorly Drained</td>
<td>SP</td>
</tr>
<tr>
<td>Moderately Well Drained</td>
<td>MW</td>
</tr>
<tr>
<td>Well Drained</td>
<td>WD</td>
</tr>
<tr>
<td>Somewhat Excessively Drained</td>
<td>SE</td>
</tr>
<tr>
<td>Excessively Drained</td>
<td>ED</td>
</tr>
</tbody>
</table>

2.18.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VeryPoorlyDrained</td>
<td>Water is at or near the soil surface during much of the growing season. Internal free-water is very shallow and persistent or permanent. Unless the soil is artificially drained, most mesophytic crops cannot be grown. Commonly, the soil occupies a depression or is level. If rainfall is persistent or high, the soil can be sloping.</td>
</tr>
<tr>
<td>PoorlyDrained</td>
<td>The soil is wet at shallow depths periodically during the growing season or remains wet for long periods. Internal free-water is shallow or very shallow and common or persistent. Unless the soil is artificially drained, most mesophytic crops cannot be grown. The soil, however, is not continuously wet directly below plow depth. The water table is commonly the result of low or very low saturated hydraulic conductivity class or persistent rainfall, or a combination of both factors.</td>
</tr>
<tr>
<td>SomewhatPoorlyDrained</td>
<td>The soil is wet at a shallow depth for significant periods during the growing season. Internal free-water is commonly shallow to moderately deep and transitory to permanent. Unless the soil is artificially drained, the growth of most mesophytic plants is markedly restricted. The soil commonly has a low or very low saturated hydraulic conductivity class, or a high water table, or receives water from lateral flow, or persistent rainfall, or some combination of these factors.</td>
</tr>
<tr>
<td>ModeratelyWellDrained</td>
<td>Water is removed from the soil somewhat slowly during some periods of the year. Internal free water commonly is moderately deep and may be transitory or permanent. The soil is wet for only a short time within the rooting depth during the growing season, but long enough that most mesophytic crops are affected. The soil commonly has a moderately low, or lower, saturated hydraulic conductivity class within 1 meter of the surface, or periodically receives high rainfall, or both.</td>
</tr>
<tr>
<td>WellDrained</td>
<td>Water is removed from the soil readily, but not rapidly. Internal free-water commonly is deep or very deep; annual duration is not specified. Water is available to plants in humid regions during much of the growing season. Wetness does not inhibit growth of roots for significant periods during most growing seasons.</td>
</tr>
</tbody>
</table>
2.19 SoilData.SoilFertilityCharacteristic Enumeration

2.19.1 Summary

Specifies the soil fertility characteristics.

2.19.2 Declaration

[C#] public enum SoilData.SoilFertilityCharacteristic

2.19.3 Remarks

The following fertility characteristics are defined:

<table>
<thead>
<tr>
<th>characteristic</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEC</td>
<td>Apparent CEC</td>
</tr>
<tr>
<td>BS</td>
<td>Base saturation</td>
</tr>
<tr>
<td>SumBC</td>
<td>Sum of basic cations</td>
</tr>
<tr>
<td>pH</td>
<td>pH-H$_2$O</td>
</tr>
<tr>
<td>OC</td>
<td>Organic carbon content</td>
</tr>
</tbody>
</table>

2.19.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEC</td>
<td>Represents the apparent CEC, expressed in centimoles of charge per kilogram of clay [cmol(+) kg$^{-1}$ clay].</td>
</tr>
<tr>
<td>BS</td>
<td>Represents the base saturation, expressed as a percentage (%).</td>
</tr>
<tr>
<td>SumBC</td>
<td>Represents the sum of the basic cations, expressed in centimoles of charge per kilogram of clay [cmol(+) kg$^{-1}$ clay].</td>
</tr>
<tr>
<td>pH</td>
<td>Represents the soil pH (dimensionless).</td>
</tr>
<tr>
<td>OC</td>
<td>Represents the soil organic carbon content, expressed as a weight percentage (%, w/w).</td>
</tr>
</tbody>
</table>

2.20 SoilData.SoilStructure Enumeration

2.20.1 Summary

Represents soil structures that are associated with determining the SoilTextureClass.

2.20.2 Declaration

[C#] public enum SoilData.SoilStructure
2.20.3 Remarks

Soil structure refers to the natural organization of soil particles into discrete soil units (peds) which are separated from each other by persistent surfaces of weakness (FAO, 1990).

Members of this enumeration can be classified into 4 subgroups:

1. Basic types - Granular, Blocky, Prismatic and Platy;
2. Subtype - Columnar;
3. Structureless types - Massive and SingleGrained;

A sketch of selected soil structure types is shown in Figure 2.3.

![Figure 2.3: Examples of soil structure types](image1)

![Figure 2.4: Cross-section of a Vertisol pedon. Source: Dudal and Eswaran (1988)](image2)
2.21 SoilData.SoilTextureClass Enumeration

2.21.1 Summary

Represents the soil texture classes used in WLES, which is based on the Soil Survey Manual (USDA Soil Survey Division Staff, 1993).

2.21.2 Declaration

[C#] public enum SoilData.SoilTextureClass

2.21.3 Remarks

Refer to the “Remarks” section of the SoilTexture documentation for the Soil Texture Triangle (see Figure 2.2) and more.
### 2.21.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C60v</td>
<td>Very fine clay with vertic structure; Clay content greater than 60% (w/w). <strong>Note:</strong> C60v is also noted as C+60v.</td>
</tr>
<tr>
<td>C60s</td>
<td>Very fine clay with blocky structure; Clay content greater than 60% (w/w). <strong>Note:</strong> C60s is also noted as C+60s.</td>
</tr>
<tr>
<td>C.60v</td>
<td>Very fine clay with vertic structure; Clay content less than 60% (w/w). <strong>Note:</strong> C.60v is also noted as C-60v.</td>
</tr>
<tr>
<td>C.60s</td>
<td>Very fine clay with blocky structure; Clay content less than 60% (w/w). <strong>Note:</strong> C.60s is also noted as C-60s.</td>
</tr>
<tr>
<td>Co</td>
<td>Clay with oxic structure.</td>
</tr>
<tr>
<td>Cm</td>
<td>Clay with massive structure.</td>
</tr>
<tr>
<td>C</td>
<td>Clay</td>
</tr>
<tr>
<td>SiCm</td>
<td>Silty clay with massive structure.</td>
</tr>
<tr>
<td>SiCs</td>
<td>Silt clay with blocky structure.</td>
</tr>
<tr>
<td>SiC</td>
<td>Silt clay</td>
</tr>
<tr>
<td>SiCL</td>
<td>Silty clay loam</td>
</tr>
<tr>
<td>CL</td>
<td>Clay loam</td>
</tr>
<tr>
<td>L</td>
<td>Loam</td>
</tr>
<tr>
<td>SiL</td>
<td>Silt loam</td>
</tr>
<tr>
<td>Si</td>
<td>Silt</td>
</tr>
<tr>
<td>SC</td>
<td>Sandy clay</td>
</tr>
<tr>
<td>SCL</td>
<td>Sandy clay loam</td>
</tr>
<tr>
<td>fSL</td>
<td>Fine sand loam. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>SL</td>
<td>Sand loam. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>cSL</td>
<td>Coarse sand loam. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>LfS</td>
<td>Loamy fine sand. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>LS</td>
<td>Loam sand. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>LeS</td>
<td>Loamy coarse sand. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>fS</td>
<td>Fine sand. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>S</td>
<td>Sand. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
<tr>
<td>cS</td>
<td>Coarse sand. Refer to SoilData.SandClass for all subclasses of sand based on the diameter in millimeters of sand particles.</td>
</tr>
</tbody>
</table>

---

### 2.22 SoilData.WRBSoilGroup Enumeration

#### 2.22.1 Summary

Represents the WRB Reference Soil Groups.

#### 2.22.2 Declaration

```csharp
[C#] public enum SoilData.WRBSoilGroup
```
2.22.3 Remarks

1. The World Reference Base for Soil Resources (WRB) is the international standard soil classification system endorsed by the International Union of Soil Sciences (IUSS). It replaces the FAO Soil Classification System.

2. WRB defines 30 Reference Soil Groups (1st level) and more than 200 Soil Units (2nd level).

2.22.4 Members

1. Acrisols: Soils with subsurface accumulation of low activity clays and low base saturation.
   - Abbreviation - AC;
   - Distribution - Most extensive on acid rocks in Southeast Asia, Southeast USA, the southern fringes of the Amazon basin and in both east and west Africa;
   - Coverage - 1,000 million hectares;
   - Association - Associated with Nitisols, Ferralsols, Plinthosols, Lixisols, Arenosols, Regosols and Cambisols.

2. Albeluvisols: Acid soils with a bleached horizon penetrating into a clay-rich subsurface horizon.
   - Abbreviation - AB;
   - Distribution - An extensive belt of Albeluvisols stretches eastward from the Baltic Sea across Russia into central Siberia. Scattered smaller areas occur in western Europe and the United States;
   - Coverage - 320 million hectares;
   - Association - Associated with Luvisols, Gleysols and Podzols.

3. Alisols: Soils with subsurface accumulation of high activity clays, rich in exchangeable aluminium.
   - Abbreviation - AL;
   - Distribution - The total area of Alisols has not been accurately ascertained as these soils have only been mapped in association with the Acrisols. However, it is thought to be in the region of 100 million hectares. Alisols are typically found in the southeastern United States, Latin America, Indonesia and China;
   - Coverage - (Thought to be) 100 million hectares;
   - Association - Associated with Acrisols, Lixisols and Ferralsols.

   - Abbreviation - AN;
   - Distribution - Andosols are associated with pyroclastic parent materials and therefore volcanic regions of the world. They occur around the Pacific rim, including North and Central America, the Andean region, Indonesia, Philippines, Japan and New Zealand as well as along the African Rift Valley with scattered occurrence elsewhere;
   - Coverage - 110 million hectares;
   - Association - Associated with Cambisols, Luvisols and Vertisols.

5. Anthrosols: Soils in which human activities have resulted in profound modification of their properties.
   - Abbreviation - AT;
• Distribution - Anthrosols are found wherever people have lived for a long time. Plaggic and Terric Anthrosols extend over more than 500,000 hectares in north-western Europe. Irragric Anthrosols are found in irrigation areas in dry regions, e.g. in Mesopotamia and in parts of India. Hydragric Anthrosols (‘paddy soils’) occupy vast areas in China and in parts of South and Southeast Asia (e.g. Sri Lanka, Vietnam, Thailand, and Indonesia). Hortic Anthrosols are found all over the world where Man has fertilized the soil with household wastes and manure. The ‘Terra Preta do Indio’ in the Amazon Region of Brazil belong to this group;
  • Coverage - N.A.;
  • Association - Associated with virtually any soil, but most likely with Acrisols, Alisols, Arenosols, Calcisols, Cambisols, Fluvisols, Gleysols, Gypsisols, Histosols, Lixisols, Luvisols, Podzols, Regosols, Solonchaks and Solonetz.

6. Arenosols : Sandy soils featuring very weak or no soil development.
  • Abbreviation - AR;
  • Distribution - These soils occur on deep aeolian, marine, lacustrine and alluvial sands, mainly in the Kalahari and Sahel regions of Africa, and in Western Australia and South America;
  • Coverage - 900 million hectares;
  • Association - Associated with Leptosols, Regosols, Calcisols, Solonchaks and Podzols.

7. Calcisols : Soils with accumulation of secondary calcium carbonates.
  • Abbreviation - CL;
  • Distribution - The Calcisols occur in western USA, Saharan Africa, Southwest Africa, the Near East and Central Asia;
  • Coverage - 800 million hectares;
  • Association - Associated with Regosols, Cambisols, Gypsisols and Solonchaks.

8. Cambisols : Weakly to moderately developed soils.
  • Abbreviation - CM;
  • Distribution - Cambisols occur worldwide, with a dominance in the temperate regions. They are common on Pleistocene and other parent materials, and are one of the most widespread soils;
  • Coverage - 1,500 million hectares;
  • Association - Associated with Gleysols, Leptosols, Fluvisols, Acrisols and Ferralsols.

  • Abbreviation - CH;
  • Distribution - Chernozems occur on the cooler mid-latitude steppes and prairies of Eurasia and North America. They occur mainly on lossial parent materials;
  • Coverage - 230 million hectares;
  • Association - Associated with Luvisols, Phaeozems and Kastanozems.

10. Cryosols : Soils with permafrost within 1 m depth.
  • Abbreviation - CR;
  • Distribution - Occurring in the Arctic and subarctic regions of Canada, Alaska, Russia, China and in Antarctica. They also occur at high elevation in mountainous areas;
2.22 SoilData.WRBSoilGroup Enumeration

- Coverage - 1,770 million hectares;
- Association - Associated with Podzols, Histosols and Gleysols.

- Abbreviation - DU;
- Distribution - These soils are extensive in Australia, South Africa and America. As they have not been mapped separately, an estimate of their extent is not available;
- Coverage - N.A.;
- Association - Associated with Gypsisols, Calcisols, Arenosols, Cambisols and Vertisols.

- Abbreviation - FR;
- Distribution - Ferralsols are restricted to tropical regions, mainly South and Central America and Central Africa, with scattered areas elsewhere;
- Coverage - 750 million hectares;
- Association - Associated with Acrisols, Nitisols, Plinthosols and Cambisols.

13. Fluvisols: Young soils in alluvial deposits.
- Abbreviation - FL;
- Distribution - Worldwide occurrence on river flood plains, deltaic areas, and coastal marine lowlands;
- Coverage - 350 million hectares;
- Association - Associated with Histosols and Gleysols.

14. Gleysols: Soils with permanent or temporary wetness near the surface.
- Abbreviation - GL;
- Distribution - About half of the Gleysols of the world occur in sub-arctic areas of northern Russia, Siberia, Canada and Alaska. The remaining areas occur mainly in humid temperate and low-land inter-tropical regions;
- Coverage - 720 million hectares;
- Association - Associated with Cryosols, Podzols, Histosols, Fluvisols, Calcisols, Gypsisols, Acrisols, Lixisols, Alisols and Nitisols.

   (a) Abbreviation - GY;
   (b) Distribution - Distribution similar to that of Calcisols. Excellent examples are to be found in Bahrain, Oman and Tunisia.
   (c) Coverage - 90 million hectares;
   (d) Association - N.A.

16. Histosols: Soils which are composed of organic materials.
- Abbreviation - HS;
- Distribution - Histosols occur in the northern parts of America, Europe and Asia, and on coastal low-lands of the subtropics and tropics;
- Coverage - 315 million hectares;
- Association - Associated with Podzols, Gleysols and Fluvisols.
17. Kastanozems: Soils with a thick, dark brown topsoil, rich in organic matter and a calcareous or gypsum-rich subsoil.
   - Abbreviation - KS;
   - Distribution - Kastanozems occur on the southern steppes of Ukraine, southern Russia, Mongolia and the Great Plains of the USA;
   - Coverage - 315 million hectares;
   - Association - Associated with Chernozems, Calcisols, Gypsisols, Solonetz and Solonchaks.

18. Leptosols: Very shallow soils over hard rock or in unconsolidated very gravelly material.
   - Abbreviation - LP;
   - Distribution - Leptosols are most common in mountainous areas (545 million hectares) and deserts (420 million hectares);
   - Coverage - 1,655 million hectares;
   - Association - Associated with Regosols and Cambisols in mountainous areas, and with Arenosols, Calcisols and Gypsisols in the deserts.

19. Lixisols: Soils with subsurface accumulation of low activity clays and high base saturation.
   - Abbreviation - LX;
   - Distribution - Lixisols occur mainly in Brazil, West Africa, East Africa and India;
   - Coverage - 435 million hectares;
   - Association - N.A.

20. Luvisols: Soils with subsurface accumulation of high activity clays.
   - Abbreviation - LV;
   - Distribution - Luvisols concentrates in central and western Europe, the lands around the Mediterranean Sea and North America. Smaller areas occur in Australia and south-eastern parts of the Republic of South Africa;
   - Coverage - 650 million hectares;
   - Association - Associated with Cambisols and Gleysols.

   - Abbreviation - NT;
   - Distribution - Nitisols occur on high plateaux in eastern Africa, particularly in Kenya, Ethiopia and Tanzania. Smaller areas occur at lower altitudes in India, the Philippines, Java, Central America and Brazil;
   - Coverage - 200 million hectares;
   - Association - N.A.

   - Abbreviation - PH;
   - Distribution - Phaeozems distribute mainly in the more humid steppes of Russia, the prairies of USA and Canada, the pampas of Argentina and Uruguay, China and south-eastern parts of Europe;
   - Coverage - 190 million hectares;
   - Association - N.A.
   - Abbreviation - PL;
   - Distribution - Planosols are extensive in Brazil, northern Argentina, South Africa and eastern Australia. Smaller areas occur in southeast Asia from Bangladesh to Vietnam and in the eastern United States. Most of Planosols occur in South America and Australia;
   - Coverage - 130 million hectares;
   - Association - Associated with Acrisols, Luvisols and Vertisols.

24. Plinthosols: Wet soils with an irreversibly hardening mixture of iron, clay and quartz in the subsoil.
   - Abbreviation - PT;
   - Distribution - Active formation of plinthite is taking place in West Africa, parts of South America, India and Western Australia. Ironstone, or hardened plinthite is more widely spread and often associated with old, high level pene-plains of the southern continents;
   - Coverage - 60 million hectares;
   - Association - Associated with Ferralsols, Acrisols and Alisols.

25. Podzols: Acid soils with a blackish/brownish/reddish subsoil with alluvial iron-aluminium-organic compounds.
   - Abbreviation - PZ;
   - Distribution - Podzols occur mainly in northern Russia, Siberia and northern Canada. Scattered, smaller areas occur on coarse parent materials associated with heathland;
   - Coverage - 485 million hectares;
   - Association - Associated with Anthrosols, Cryosols, Cambisols, Gleysols and Histosols; and in the humid tropics, mainly Acrisols and Arenosols.

26. Regosols: Soils with very limited soil development.
   - Abbreviation - RG;
   - Distribution - Regosols occur mainly in arid areas, the dry tropics and in mountainous regions;
   - Coverage - 260 million hectares;
   - Association - Associated with Leptosols and Arenosols.

27. Solonchaks: Strongly saline soils.
   - Abbreviation - SC;
   - Distribution - Solonchaks occur where evaporation exceeds rainfall and there is a seasonal or permanent water table close to the soil surface, or in coastal areas influenced by saline intrusions. Widespread but scattered distribution in Saharan Africa, East Africa, Namibia, Central Asia, Australia and South America;
   - Coverage - 260-340 million hectares;
   - Association - Associated with Gleysols and Solonetzes.

28. Solonetzes: Soils with subsurface clay accumulation, rich in sodium.
   - Abbreviation - SN;
   - Distribution - Scattered areas throughout the world where there is predominance of sodium over calcium salts in soils;
- Coverage - 135 million hectares;
- Association - Associated with Solonchaks and Gleysols.

29. Umbrisols: Acid soils with a thick, dark topsoil rich in organic matter.
- Abbreviation - UM;
- Distribution - Umbrisols occur mainly in Western Europe, the northwest seaboard of USA and Canada, the mountain ranges of the Himalayas and the mountain ranges of South America.
- Coverage - 100 million hectares;
- Association - N.A.

30. Vertisols: Dark-coloured cracking and swelling clays.
- Abbreviation - VR;
- Distribution - Vertisols occur in central Sudan, East Africa, the Deccan Plateau of India, Texas, South America and Australia;
- Coverage - 335 million hectares;
- Association - Associated with Luvisols, Cambisols, Gypsisols and Solonchaks.

2.23 WlesDataSet.TemporalResolution Enumeration

2.23.1 Summary
Specifies the temporal resolutions that a containing WlesDataSet instance may have.

2.23.2 Declaration

[C#]
public enum WlesDataSet.TemporalResolution

2.23.3 Remarks
Attention has to be drawn on DAILY data! The length of a DAILY data array varies with the value of the working calendar: 360 for WlesCalendar and 365 for GregorianCalendar.

2.23.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAILY</td>
<td>Data acquired on the daily basis.</td>
</tr>
<tr>
<td>DECADELY</td>
<td>Data acquired on the decadely basis.</td>
</tr>
<tr>
<td>MONTHLY</td>
<td>Data acquired on the monthly basis.</td>
</tr>
</tbody>
</table>
Chapter 3

Namespace UGent.WLES.Topo

The UGent.WLES.Topo namespace packages all necessary classes and structures for conducting topological analysis, such as point-on-line judgement, point-in-polygon judgement, line inclination calculation, and so on. The topological entities of point, line, polyline and polygon are represented in this namespace.

3.1 Overview

3.1.1 Structures

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>Represents a line in the Cartesian coordinate system.</td>
<td>129</td>
</tr>
<tr>
<td>Point</td>
<td>Represents a point in the Cartesian coordinate system.</td>
<td>130</td>
</tr>
<tr>
<td>Polygon</td>
<td>Represents a polygon in the Cartesian coordinate system.</td>
<td>132</td>
</tr>
<tr>
<td>Polyline</td>
<td>Represents a polyline in the Cartesian coordinate system.</td>
<td>133</td>
</tr>
</tbody>
</table>

3.2 Line Structure

3.2.1 Summary

Represents a line in the Cartesian coordinate system.

3.2.2 Declaration

[C#] public struct Line

3.2.3 Remarks

Definitions and differences of topological entities:

1. Point - A Point has two coordinates, namely x and y, whose values are offsets in the horizontal and vertical axises compared to the origin. The origin is the Point with value (0, 0).

2. Line - A Line has two nodes, each defined by a Point. A line is a simplified version of a Polyline with no vertex.

3. Polyline - A Polyline has two nodes and at least one vertex in between. In other words, a polyline consists of Line segments.

4. Polygon - A Polygon is a closed Polyline when its ending node overlaps the beginning node. Nevertheless, a Polygon is significantly different from a Polyline on that a Polygon covers a closed area while a Polyline usually does not.
3.2.4 Public Instance Constructors

- public Line(Point head, Point tail);
  Initializes a new instance of Line.

3.2.5 Public Instance Properties

- Head
  Gets a Point as the head node.
  public Point Head { get; }

- Tail
  Gets a Point as the tail node.
  public Point Tail { get; }

3.2.6 Public Instance Methods

- GetTiltAngle
  Calculates the angle between the x-axis and this Line.
  public double GetTiltAngle();
  Return Value: A Double that represents the tilt angle in degrees: counterclockwise angle has positive value; clockwise has negative.

- HasPoint
  Determines whether a particular Point is on this Line (node exclusive).
  public bool HasPoint(Point p);
  Return Value: true if p is online and different from the Head or the Tail node; false otherwise.
  Remarks: A Point is not viewed online if it equals to the Head or the Tail node of the Line.

- IsHeadOrTail
  Determines whether a particular Point is the same as the Head or the Tail node.
  public bool IsHeadOrTail(Point p);
  Return Value: true if p equals to the Head or the Tail node; false otherwise.

- ToString
  Gets a String representation of this Line.
  public override string ToString();
  Return Value: A String that represents this Line, in the following pattern: “(head.x, head.y),(tail.x, tail.y)”.

3.3 Point Structure

3.3.1 Summary

Represents a point in the Cartesian coordinate system.

3.3.2 Declaration

[C#] public struct Point

3.3.3 Remarks

Definitions and differences of topological entities:

1. Point - A Point has two coordinates, namely x and y, whose values are offsets in the horizontal and vertical axises compared to the origin. The origin is the Point with value (0, 0).
2. Line - A Line has two nodes, each defined by a Point. A line is a simplified version of a Polyline with no vertex.

3. Polyline - A Polyline has two nodes and at least one vertex in between. In other words, a polyline consists of Line segments.

4. Polygon - A Polygon is a closed Polyline when its ending node overlaps the beginning node. Nevertheless, a Polygon is significantly different from a Polyline on that a Polygon covers a closed area while a Polyline usually does not.

3.3.4 Public Instance Constructors
- public Point(double x, double y);
  Initializes a new instance of Point.

3.3.5 Public Instance Properties
- X
  Gets the X coordinate.
  [C#] public double X { get; }
- Y
  Gets the Y coordinate.
  [C#] public double Y { get; }

3.3.6 Public Instance Methods
- Equals
  Determines whether a particular Point has the same value as this.
  [C#] public bool Equals(Point p);
  Return Value: true if the X and Y coordinates of p equal to those of this Point; false otherwise.
- IsInside
  Determines whether this Point is inside a particular Polygon.
  [C#] public bool IsInside(Polygon polygon);
  Return Value: true if this Point is inside polygon; false otherwise.
  Remarks:
  1. A Point is not viewed as inside a Polygon if it is on the outline, or if it is overlapping with one of the Vertices or nodes.
  2. Use the IsOnline method instead to determine whether a Point is on the outline (including the vertices or nodes).

- IsOnline
  Overloaded.
  1. Determines whether this Point is on a particular Line.
     [C#] public bool IsOnline(Line line);
     Remarks: This method is implemented using the HasPoint method.
  2. Determines whether this Point is on the outline of a particular Polygon, or the same as one of the Vertices or nodes of the Polygon.
     [C#] public bool IsOnline(Polygon polygon);
     Return Value: true if this Point is on the outline of polygon, or the same as one of its Vertices or nodes; false otherwise.
     Remarks: Please note that a Point is viewed online if it has the same value as any of
the Vertices of the Polygon. This behaviour is different from those of the overloading methods.

3. Determines whether this Point is on a particular Polyline.
[C#] public bool IsOnline(Polyline polyline);
Remarks: This method is implemented using the HasPoint method.

- ToString
  Gets a String representation of this Point.
  [C#] public override string ToString();
  Return Value: A String that represents this Point, in the following pattern: “(x, y)”.

3.4 Polygon Structure

3.4.1 Summary
Represents a polygon in the Cartesian coordinate system.

3.4.2 Declaration
[C#] public struct Polygon

3.4.3 Remarks
Definitions and differences of topological entities:

1. Point - A Point has two coordinates, namely x and y, whose values are offsets in the horizontal and vertical axises compared to the origin. The origin is the Point with value (0, 0).

2. Line - A Line has two nodes, each defined by a Point. A line is a simplified version of a Polyline with no vertex.

3. Polyline - A Polyline has two nodes and at least one vertex in between. In other words, a polyline consists of Line segments.

4. Polygon - A Polygon is a closed Polyline when its ending node overlaps the beginning node. Nevertheless, a Polygon is significantly different from a Polyline on that a Polygon covers a closed area while a Polyline usually does not.

3.4.4 Public Instance Constructors
  • public Polygon(Point[] vertices);
    Initializes a new instance of Polygon.
  • public Polygon(Polyline polyline);
    Initializes a new instance of Polygon with a Polyline.
    Parameters:
    polyline A Polyline that is used to initialize this Polygon. The Head of the Polyline is virtually viewed as both the starting and the ending Point of the Polygon.

3.4.5 Public Instance Properties
  • Vertices
    Gets a Point array of all the vertices.
    [C#] public Point[] Vertices { get; }
### 3.4.6 Public Instance Methods

- **IsInside**
  Determines whether a particular Point is inside this Polygon.

[C#] public bool IsInside(Point p);

Remarks:
1. A Point is not viewed as inside a Polygon if it is on the outline, or if it is overlapping with one of the Vertices or nodes.
2. Use the IsOnline method instead to determine whether a Point is on the outline (including the vertices or nodes).

- **IsOnline**
  Determines whether a particular Point is on the outline of this Polygon, or as one of the Vertices or nodes.

[C#] public bool IsOnline(Point p);

Return Value: true if p is on the outline, or the same as one of the Vertices or nodes; false otherwise.

Remarks: Please note that a Point is viewed online if it has the same value as any of the Vertices of the Polygon. This behaviour is different from that of the HasPoint method of the Line structure. In the latter case, a Point is not viewed online if it is the same as the Head or the Tail.

- **ToString**
  Gets a String representation of this Polygon.

[C#] public override string ToString();

Return Value: A String that represents this Polygon, in the following pattern:

```
(vertex1.x, vertex1.y), ..., (vertexi.x, vertexi.y), ..., 
(vertexn-1.x, vertexn-1.y), (vertex1.x, vertex1.y)
```

Note: The last vertex is the same as the first one, since a Polygon is always closed.

### 3.5 Polyline Structure

#### 3.5.1 Summary
Represents a polyline in the Cartesian coordinate system.

#### 3.5.2 Declaration

[C#] public struct Polyline

#### 3.5.3 Remarks
Definitions and differences of topological entities:

1. Point - A Point has two coordinates, namely x and y, whose values are offsets in the horizontal and vertical axes compared to the origin. The origin is the Point with value (0, 0).

2. Line - A Line has two nodes, each defined by a Point. A line is a simplified version of a Polyline with no vertex.

3. Polyline - A Polyline has two nodes and at least one vertex in between. In other words, a polyline consists of Line segments.

4. Polygon - A Polygon is a closed Polyline when its ending node overlaps the beginning node. Nevertheless, a Polygon is significantly different from a Polyline on that a Polygon covers a closed area while a Polyline usually does not.
3.5.4 Public Instance Constructors

- public Polyline(Point head, Point tail, Point[] vertices);
  Initializes a new instance of Polyline.

- public Polyline(Point[] points);
  Initializes a new instance of Polyline.

  Parameters:

  points A Point array that contains the nodes and the vertices. The 1st and the last Points are viewed as nodes and the rest, vertices.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
  | ArgumentException   | Thrown when the length of the points array is 2 or less.

3.5.5 Public Instance Properties

- Head
  Gets a Point as the head node.

  ```csharp
  public Point Head { get; }
  ```

- Tail
  Gets a Point as the tail node.

  ```csharp
  public Point Tail { get; }
  ```

- Vertices
  Gets a Point array as the vertices.

  ```csharp
  public Point[] Vertices { get; }
  ```

3.5.6 Public Instance Methods

- GetSegments
  Gets all the segments that consists of this Polyline.

  ```csharp
  public Line[] GetSegments();
  ```

- HasPoint
  Determines whether a particular Point is on this Polyline (node exclusive).

  ```csharp
  public bool HasPoint(Point p);
  ```

  Return Value: true if p is online and different from the Head or the Tail node; false otherwise.

  Remarks: A Point is not viewed online if it equals to the Head or the Tail node of the Polyline.

- IsHeadOrTail
  Determines whether a particular Point is the same as the Head or the Tail node.

  ```csharp
  public bool IsHeadOrTail(Point p);
  ```

- ToString
  Gets a String representation of this Polyline.

  ```csharp
  public override string ToString();
  ```

  Return Value: A String that represents this Polyline, in the following pattern:

  “(head.x, head.y), ..., (vertex_i.x, vertex_i.y), ..., (tail.x, tail.y)”.
Chapter 4

Namespace UGent.WLES.RPP

The UGent.WLES.RPP namespace contains all necessary classes and structures for estimating the radiation-thermal crop production potential (RPP). RPP reflects the maximum production that a high-yielding variety at a specific location can achieve under optimal conditions, i.e., being free of stress of water, nutrients, pests and diseases, and on a constraint-free soil.

4.1 Overview

4.1.1 Classes

<table>
<thead>
<tr>
<th>type summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biomass</td>
<td>Represents the total net biomass production ($B_n$) and its economic yield, namely the radiation-thermal production potential (RPP) of a specific crop in a particular geographic location.</td>
</tr>
<tr>
<td>CropCycle</td>
<td>Represents the crop cycle of an annual crop within the time frame defined by the growing period (see: GrowingPeriod class).</td>
</tr>
<tr>
<td>GrowingPeriod</td>
<td>Represents the growing period of annual crops within a yearly time frame when temperature, soil water content and other climatic conditions permit crop growth and development.</td>
</tr>
<tr>
<td>GrowingPeriodUI</td>
<td>Represents the GrowingPeriod determination and reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>RPPReport</td>
<td>Represents the UGent.WLES.RPP reporting interface of the WLES Web UI.</td>
</tr>
</tbody>
</table>

4.1.2 Enumerations

<table>
<thead>
<tr>
<th>type summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biomass.PhotosynGroup</td>
<td>Specifies the photosynthetical group ID of crops, i.e., I through V.</td>
</tr>
<tr>
<td>Biomass.PhotosynPathway</td>
<td>Specifies the photosynthetical pathways of crops.</td>
</tr>
<tr>
<td>CropCycle.CropDevelopmentStage</td>
<td>Specifies the crop development stages (CDS) of a CropCycle.</td>
</tr>
<tr>
<td>CropCycle.CropGrowthPeriod</td>
<td>Specifies the crop growth periods (CGP) within a CropCycle.</td>
</tr>
<tr>
<td>CropCycle.SubCGP</td>
<td>Specifies the sub division within a crop growth period (CGP).</td>
</tr>
</tbody>
</table>
4.2 Biomass Class

4.2.1 Summary
Represents the total net biomass production ($B_n$) and its economic yield, namely the radiation-thermal production potential (RPP) of a specific crop in a particular geographic location.

4.2.2 Declaration

[C#] public class Biomass

4.2.3 Remarks
1. The net biomass ($B_n$) is obtained from the difference between the gross biomass production and the respiration losses.

2. The estimation of $B_n$ is based on the solar radiation, the air temperature and the intrinsic characteristics of the crop.

3. The economic yield of the biomass production, or the radiation-thermal production potential (RPP), is obtained on the basis of the net biomass production, using the harvest index as a reducing factor:

$$ RPP = B_n \cdot f_C $$  \hspace{1cm} (4.1)

where: $f_C$ = yield reducing coefficient due to harvest loss, i.e., the harvest index ($H_i$).

4.2.4 Example
The following example shows how to determine the radiation-thermal production potential of groundnut under optimal irrigated agricultural conditions in Kigali, Rwanda. Groundnut is known as a PhotosynGroup II leguminous crop, with a LAI at maximum crop growth of $3.5 \text{ m}^2 \text{ m}^{-2}$ and a harvest index of 0.30. The crop cycle of groundnut starts on Feb. 21 and ends on Jun. 20.

using System;
using UGent.WLES;
using UGent.RPP;

class RppEvaluation {
    public static void Main(string [] args) {
        // collect data first
        string place = "Kigali (Rwanda)";
        double elevation = 1500;
        double latitude = (1 + 58.0/60.0) * -1;
        // monthly climatic data
        WlesDataSet.TemporalResolution resolution =
            WlesDataSet.TemporalResolution.MONTHLY;
        double [] tmax = new double []
        double [] tmin = new double []
            {14.9,14.9,15.2,15.5,15.5,14.6,14.5,15.3,15.1,15.2,14.9,15.1};
        double [] RH = new double []
            {76.5,74.77,81.5,77.5,71,61.5,62.5,67.5,73.5,76.5,76};
        double [] wind = new double []
            {1.8,1.8,1.8,1.8,1.8,1.8,1.8,1.8,1.8,1.8,1.8,1.8};
        double [] n = new double []
            {5.4,5.5,5.2,4.9,5.5,7.0,7.0,6.7,5.6,5.6,5.0,5.3};
        double [] rain = new double []
    }
}
4.2 Biomass Class

\{68.2,110.6,113.9,178.5,91.8,22.5,9.7,31.1,74.1,94.7,132,88.6\};

GrowingPeriod gp;
gp = new GrowingPeriod(resolution,latitude,elevation,tmax,tmin,RH,wind,n,rain);

// crop cycle
RomanDate start, end;
start = new RomanDate(2, 21);
end = new RomanDate(6, 20);
string crop, cultivar;
crop = "Groundnut";
cultivar = "";

CropCycle cc;
cc = new CropCycle(crop, cultivar, start, end, place);

// biomass production
bool isLeguminous = true;
double LAI, HI;
LAI = 3.5;
HI = 0.3;
PhotosynGroup group = Biomass.PhotosynGroup.II;

Biomass biomass;
biomass = new Biomass(gp, cc, group, isLeguminous, LAI, HI);
// tro to get the RPP result
// returns -1 that means evaluation has not been done.
Console.WriteLine("RPP = {0} kg ha\(^{-1}\)", biomass.RPP);

// do the evaluation
biomass.Evaluate();

// the RPP result is available now
Console.WriteLine("RPP = {0} kg/ha", biomass.RPP);
Console.WriteLine();
// the report
Console.WriteLine(biomass);
}

The example generates the following results:

RPP = -1 kg/ha
RPP = 2886.48 kg/ha

Groundnut in Kigali (Rwanda)
Crop cycle: February 21 (WLES Calendar) - June 20 (WLES Calendar)
RPP: 2886.48 kg/ha

4.2.5 Public Static Methods

- CalculatePm
  Calculates the maximum leaf photosynthesis rate \(P_m\) (in kg CH\(_2\)O ha\(^{-1}\) hr\(^{-1}\)) at light saturation for a particular photosynthetic group of crops.

[C#] public static double CalculatePm(PhotosynGroup group, double tday);

Parameters:
Namespace UGent.WLES.RPP

**group** The photosynthesital group ID of crops; a Biomass.PhotosynGroup member.

**tday** The day-time temperature in Celsius degrees, usually in the range of [5, 45]. If not readily available, daytime and nighttime temperatures can be obtained from the daily maximum and minimum temperatures (together with the day length data). See: CalculateDayNightTemperatures.

**Remarks**: The algorithm applied here is derived from Sys et al. (1991, p.130).

**Exceptions**:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when arguments <strong>tday</strong> has invalid values.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

**Example**: The following example helps to export the $P_m$ data of crops in groups I through V to a comma-delimited data file which can be used to plot the $P_m$ curves.

```csharp
using System;
using System.IO;
using UGent.WLES.Data;
using UGent.WLES.RPP;

public class PmData {
    public static void Main(string[] args) {
        if (args.Length != 1)
            Console.WriteLine(
                "Please specify valid path and file name!");
        else {
            string file;
            file = args[0];
            ExportData(file);
            Console.WriteLine(string.Format(
                "Data have been exported to {0}", file));
        }
    }

    public static void ExportData(string file) {
        double Pm;
        double t = 0;
        StreamWriter sw = new StreamWriter(file);
        sw.WriteLine("Temp,I,II,III,IV,V");
        while (t <= 50) {
            sw.Write(t);
            for (int i=1; i<=5; i++) {
                Pm = CalculatePm((PhotosynGroup)i,t);
                sw.Write(string.Format(",{0}",Pm));
            }
            sw.WriteLine();
            t += 0.1;
            t = Math.Round(t,2);
        }
        sw.Close();
    }
}
```

Refer to GetPhotosynGroupID method for an example of calculating the $P_m$ value of a particular crop at a particular temperature.
4.2 Biomass Class

- CalculateRPP
  Calculates the radiation-thermal production potential (kg ha\(^{-1}\)).

  ```csharp
  public static double CalculateRPP(PhotosynGroup group, bool isLeguminous, double tmean, double tday, double f, double bc, double bo, int L, double LAImax, double Hi);
  ```

  **Parameters:**

  - `group` Crop’s photosynthetic group ID; a `Biomass.PhotosynGroup` member.
  - `isLeguminous` A `Boolean` value that judges if the considered crop is leguminous.
  - `tmean` Mean daily air temperature in Celsius degrees (°C).
  - `tday` Average daytime air temperature in Celsius degrees (°C).
  - `f` Overcast ratio. See: `CalculateOvercastRatio`.
  - `bc` Daily gross photosynthesis rate of crop canopies on very clear days (in kg ha\(^{-1}\) day\(^{-1}\)).
  - `bo` Daily gross photosynthesis rate of crop canopies on overcast days (in kg ha\(^{-1}\) day\(^{-1}\)).
  - `L` Length of the crop cycle, i.e., the number of days counted from the start (inclusive) to the end (inclusive) of the crop cycle.
  - `LAImax` Leaf area index at crop’s maximum growth rate (m\(^2\) m\(^{-2}\)).
  - `Hi` Harvest index.

  **Exceptions:**

  - `ArgumentException` Thrown when arguments `f`, `bc`, `bo`, `LAImax` or `Hi` invalid.
  - `WiesDatabaseException` Thrown when the underlying database error occurs.

- ExportPmData
  Overloaded.

  1. Exports the maximum leaf photosynthesis rate data, with default settings.

  ```csharp
  public static string ExportPmData();
  ```

  **Return Value:** A `String` that contains the comma-separated \(P_m\) data (kg CH\(_2\)O ha\(^{-1}\) h\(^{-1}\)).

  **Example:** The following example shows explicitly the format of an exported dataset.

<table>
<thead>
<tr>
<th>Title: Relationship between (P_m) (kg CH(_2)O ha(^{-1}) h(^{-1})) and daytime temperature (C)</th>
<th>X-axis: Daytime temperature (C)</th>
<th>Y-axis: Maximum leaf photosynthesis rate (kg CH(_2)O ha(^{-1}) h(^{-1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daytime Temperature, Group I, Group II, Group III, Group IV</td>
<td>Daytime Temperature, Group I, Group II, Group III, Group IV</td>
<td></td>
</tr>
<tr>
<td>15.6, 19.66097968, 20.627922499999999999999999</td>
<td>15.7, 19.70210414, 20.907722399999999999999999</td>
<td>7.549757999999999999999999</td>
</tr>
<tr>
<td>15.8, 19.74144576, 20.185077999999999999999999</td>
<td>15.9, 19.77900442, 20.460051999999999999999999</td>
<td>8.543926999999999999999999</td>
</tr>
<tr>
<td>15.5, 19.42861898, 19.192327999999999999999999</td>
<td>15.6, 19.47865584, 19.484414399999999999999999</td>
<td>11.508096799999999999999999</td>
</tr>
</tbody>
</table>
2. Exports the maximum leaf photosynthesis rate data.

```csharp
public static string ExportPmData(double[] tdayRange, double tdayStep, PhotosynGroup[] groups);
```

**Remarks:** The photosynthesis process provides plants with assimilates that can be used for growth. The rate of photosynthesis depends on the photosynthesis pathways and its response to temperature and radiation.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the <code>tdayRange</code> array does not contain two elements, or the 2nd element is smaller than the 1st in magnitude; and when the <code>tdayStep</code> is smaller than or equal to zero.</td>
</tr>
</tbody>
</table>

- **GetCropCycleLength**
  
  Gets the length (i.e., number of days) of the crop cycle.

```csharp
public int GetCropCycleLength();
```

- **GetCropCycleRomanDates**
  
  Gets the crop cycle starting and ending dates, expressed in a `RomanDate` array.

```csharp
public RomanDate[] GetCropCycleRomanDates();
```

- **GetCropsByPhotosynGroup**
  
  Gets all crops (including cultivar and alias, if applicable) in a particular photosynthetical group.

```csharp
public static string[][] GetCropsByPhotosynGroup(PhotosynGroup group);
```

**Return Value:** A jagged `String` array. The outer array contains the records of crops which in turn are 3-element arrays. The length of the outer array equals the number of crops in that particular group. Each record contains three element strings, in the order of crop name, cultivar name and alias name. As a result, the whole returned array is in the following format: `crop1,cultivar1,alias1`, ... `cropi,cultivar1,alias1`, ... `cropn,cultivar1,alias1`

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

**Example:** The following example shows the way of extracting all the crop, cultivar and alias names in crop group II. It also demonstrates how to iterate the resulting array and prints the list of crops on a console screen.

```csharp
using System;
using UGent.WLES.Data;
using UGent.WLES.RPP;

public class Test {
    public static void Main() {
        // extract the records
        string[][] records = Biomass.GetCropsByPhotosynGroup(
```
Biomass.PhotosynGroup.II);
string list = "";
// iterate the array
for (int i=0; i<records.Length; i++) {
    string[] record = records[i];
    list += string.Format("{0:D2}. ", i+1);
    for (int j=0; j<record.Length; j++) {
        list += string.Format("{0}{1}",
            Utilities.IsEmptyString(record[j])
            ? "-" : record[j].Trim(),
            (j==record.Length-1) ? "" : "/");
    }
    list += 
    } }
Console.WriteLine(list);

The following results will be generated by the example.

01. alfalfa/tropical/-
02. beet/-/-
03. cabbage/-/-
04. cassava/-/-
05. cotton/-/-
06. dry bean/tropical/pulse
07. grape/-/-
08. green bean/tropical/pulse
09. rice/-/-
10. soybean/-/-
11. spring wheat/tropical/-
12. sugarbeet/beet/-
13. sugarbeet/sugar/-
14. sunflower/-/-
15. sweet potato/-/-
16. tobacco/-/-
17. winter wheat/tropical/-

• GetPhotosynGroupID
  Overloaded.

1. Gets the photosynthetical group ID of a particular crop.

[C#]
public static PhotosynGroup GetPhotosynGroupID(string crop,
string cultivar);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when argument crop or cultivar is empty string; and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlseDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

Example: The following example calculates the $P_m$ value of a particular crop cultivar at a particular temperature, and prints the result on the console screen. The example first gets the photosynthetical group ID of the cultivar and then calls the CalculatePm method with both the photosynthetical group ID and the temperature as inputs.
using System;
using UGent.WLES.Data;
using UGent.WLES.RPP;

public class PmValue {
    public static void Main(string[] args) {
        if (args.Length != 3)
            Console.WriteLine("Please specify the crop, the
cultivar and the temperature!"四周);
        else {
            string crop = args[0];
            string cultivar = args[1];
            double t = Convert.ToDouble(args[2]);
            // step 1, get the photosynthetical group ID.
            Biomass.PhotosynGroup id;
            id = Biomass.GetPhotosynGroupID(crop, cultivar);
            // step 2, calculate the Pm value.
            double pm;
            pm = Biomass.CalculatePm(id, t);
            // print the result.
            Console.WriteLine(string.Format("Crop: {0}; Cultivar:
                {1}; Temperature: {2}", crop, cultivar, t);
            Console.WriteLine(string.Format(
                "Maximum leaf photosynthesis rate: {0}"", pm));
        }
    }
}

Refer to CalculatePm method for an example of calculating a continuous series of \( P_m \)
values and exporting them to a comma-delimited data file.

2. Gets the photosynthetical group ID for a particular Biomass.PhotosynPathway and a
particular optimal photosynthetical temperature.

[C#]
public static PhotosynGroup GetPhotosynGroupID(PhotosynPathway
    pathway, double optiTemp);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

Example: The following example determines the photosynthetical group ID of a hypo-
thetical crop. The crop follows the C3 pathway and its optimal photosynthetical
temperature ranges between 16.5 and 18.9 °C.

using System;
using UGent.WLES;
using UGent.WLES.RPP;
double temp = (16.5 + 18.9) / 2;
Biomass.PhotosynGroup group;
    group = Biomass.GetPhotosynGroupID(pathway, temp);
    Console.WriteLine(group);  // I */

3. Gets the photosynthetical group ID for a particular Biomass.PhotosynPathway and a
particular range of the optimal photosynthetical temperatures.

[C#]
public static PhotosynGroup GetPhotosynGroupID(PhotosynPathway
    pathway, double optiTempMin, double optiTempMax);
4.2 Biomass Class

4.2.6 Public Instance Methods

• Evaluate
  Evaluates the radiation-thermal production potential, or the RPP (kg ha⁻¹).

[C#] public void Evaluate();

Remarks: It is this parameterless Evaluate method that does the evaluation. The constructor initializes a Biomass instance with data but does not call the Evaluate method automatically. This calculation intensive method must be called separately to assign value to the RPP property.

Example: The following hypothetical code fragment shows the way to evaluate the radiation-thermal production potential, or RPP.

```csharp
using System;
using UGent.WLES;
using UGent.WLES.RPP;
GrowingPeriod period = new GrowingPeriod(...);
CropCycle cycle = new CropCycle(...);
Biomass biomass = new Biomass(period, cycle);
// DO the evaluation!
biomass.Evaluate();
```
// the RPP result alone
Console.WriteLine(biomass.RPP);
// or the RPP report
Console.WriteLine(biomass.ToString());

Refer to the example section of the Biomass class for a complete example.

- GetCropCycleLength
  Gets the length (i.e., number of days) of the crop cycle.
  [C#] public int GetCropCycleLength();

- GetCropCycleRomanDates
  Gets the crop cycle starting and ending dates, expressed in a RomanDate array.
  [C#] public RomanDate[] GetCropCycleRomanDates();

- ToString
  Overloaded.

  1. Delivers a radiation-thermal production potential (RPP) report that consists of the crop cycle and the RPP value, using default settings.
  [C#] public override string ToString();
  
  Example: The following example shows the format of a RPP report.

  ```csharp
  using System;
  using UGent.WLES;
  using UGent.RPP;
  // declare and initialize a GrowingPeriod and a CropCycle class
  GrowingPeriod gp = new GrowingPeriod(...);
  CropCycle cc = new CropCycle(...);
  // initialize a Biomass instance
  Biomass bio = new Biomass(gp, cc);
  // do the evaluation!!
  bio.Evaluate();
  // the RPP value is ready
  Console.WriteLine(bio.RPP);
  Console.WriteLine();
  // print the report
  Console.WriteLine(bio);
  
  The code generates the following result:

  2886.48

  Groundnut in Kigali (Rwanda)
  Crop cycle: February 21 (WLES Calendar) - June 20 (WLES Calendar)
  RPP: 2886.48 kg/ha

  2. Delivers a radiation-thermal production potential (RPP) report that consists of the crop cycle and the RPP value.
  [C#] public string ToString(string newLine);
  Parameters:

  newLine A String that represents the new-line (or carriage return) command in string formatting. The new-line command in GUI or console applications is different from that in Web applications or html pages.
4.3 CropCycle Class

4.3.1 Summary
Represents the crop cycle of an annual crop within the time frame defined by the growing period (see: GrowingPeriod class).

4.3.2 Declaration
[C#] public class CropCycle

4.3.3 Remarks
The CropCycle concept is crop centric. The class wraps the crop name, the cultivar name (if applicable), the crop ID which is the unique identification number assigned by the WLES application, the name of the location and the beginning and ending dates of the crop cycle.

In the simplest case, a crop cycle can be merely defined by its beginning and the ending dates (and therefore its length is known either). A crop cycle is completely defined only after the beginning and the ending dates of the following crop development stages have been specified: Initial Stage, Crop Development Stage, Mid-Season Stage, Late-Season Stage.

Refer to the remarks section in the GrowingPeriod class documentation for the semantic differences between the concepts of crop cycle and growing period.

4.3.4 Example
The following example shows the way to construct a CropCycle instance and the way to set the CropCycle.CropDevelopmentStage properties of the instance. A crop cycle report is generated in the end by calling the ToString instance method.

```csharp
using System;
using UGent.WLES;
using UGent.WLES.RPP;

class CropCycleDemo {
    public static void Main() {
        CropCycle cycle1, cycle2;
        RomanDate date1, date2;
        string crop, cultivar, place;
        date1 = new RomanDate(RomanDate.MonthOfYear.JULY, 10);
        date2 = new RomanDate(300);
        cycle1 = new CropCycle(date1, date2);
        crop = "Maize";
        cultivar = "Tropical";
        place = "Cairo";
        cycle2 = new CropCycle(crop, cultivar, date1, date2, place);
        // report
        Console.WriteLine(cycle1);
        // Crop development stages:
        // IS = 20 days, CD = 30 days, MS = 40 days and LS = 21 days
        int stageIS, stageCD, stageMS, stageLS;
        stageIS = 20;
        stageCD = 30;
        stageMS = 40;
        stageLS = 21;
        cycle2.SetCDSRomanDates(stageIS, stageCD, stageMS, stageLS);
        // report
```
The example generates the following results:

Crop cycle: July 10 (WLES Calendar) - October 30 (WLES Calendar)
Length of crop cycle: 111 days

Crop cycle of Maize (Tropical) in Cairo: July 10 (WLES Calendar) - October 30 (WLES Calendar)
Length of crop cycle: 111 days
Crop development stages:
  IS: July 10 (WLES Calendar) - July 29 (WLES Calendar), 20 days
  CD: July 30 (WLES Calendar) - August 29 (WLES Calendar), 30 days
  MS: August 30 (WLES Calendar) - October 9 (WLES Calendar), 40 days
  LS: October 10 (WLES Calendar) - October 30 (WLES Calendar), 21 days

In an heavier application scenario, all data including the climatic data, crop data, soil data, etc. are wrapped in a WlesDataSet instance. Therefore the CropCycle instance can be initialized in a much simpler way:

```csharp
// data
WlesDataSet data = new WlesDataSet();
data.SetLocationalData(...);
data.SetClimaticData(...);
data.SetCropData(...);
data.SetCropCycle(...);
GrowingPeriod gp = new GrowingPeriod(data);
CropCycle cc = new CropCycle(data);
...

// report
Console.WriteLine(cc);
```

### 4.3.5 Public Static Methods

- ExportRefCDSData
  Overloaded.

1. Exports the length of the reference crop development stages data, with default settings.

```csharp
[C#]
public static string ExportRefCDSData();
```

**Remarks**: The following values are set as the default:

(a) Crop development stages - All 4 crop development stages are included in the exported data, *i.e.*, IS, CD, MS and LS.
(b) Crop cycle - Included.

**Exceptions**:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

**Example**: The following example shows explicitly the format of an exported dataset.

```csharp
using System;
using System.IO;
using UGent.WLES;
using UGent.WLES.WPP;
```
class CropDevStagesData {
    public static void Main() {
        string data;
        data = CropCycle.ExportRefCDSData();
        StringReader sr = new StringReader(data);
        string line;
        while ( (line = sr.ReadLine()) != null )
            Console.WriteLine(line);
    }
}

The example generates the following results:

Title: Crop development stages (number of days)
X-axis: Crop (name)
Y-axis: CDS (days)
IS = Initial Stage
CD = Crop Development Stage
MS = Mid-Season Stage
LS = Late Season Stage
CC = Crop Cycle
Crop,IS1,IS2,CD1,CD2,MS1,MS2,LS1,LS2, CC1, CC2
Artichokes,20,40,40,40,220,250,30,30,310,360
Banana,60,90,90,120,120,150,90,90,360,450
Barley,15,20,25,30,50,65,30,40,120,155
Beet,25,30,35,60,50,70,30,50,140,210
Cabbage,20,30,30,50,20,30,10,20,80,130
Carrot,20,25,30,35,30,80,20,20,100,160
Castorbean,25,30,40,55,95,105,25,20,180,210
Celery,25,30,40,55,45,105,15,20,125,210
Cotton,20,30,40,50,50,60,40,55,150,195
Cucumber,20,25,30,35,40,50,15,20,105,130
Dry bean,15,20,25,30,40,50,15,15,95,115
...

2. Exports the length data of particular crop development stages of all reference crops.
[C#]
public static string ExportRefCDSData(CropDevelopmentStage[] stages, bool cropCycle);
Remarks:
(a) The length of a crop development stage is expressed as a range of number of days;
(b) Reference crops are those that are predefined in the model, besides the user defined crops.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when stages is a null reference or an empty array;</td>
</tr>
<tr>
<td></td>
<td>and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlcsDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

• GetRefCDS
Overloaded.

1. Gets the length (i.e., number of days) of a given CropDevelopmentStage of a reference crop. The length is expressed as a range.
[C#]
public static int[] GetRefCDS(int cropID, CropDevelopmentStage stage);
Return Value: A 2-element integer array that represents the range of the crop development stage.
Remarks: The crop ID is defined in the database by crop name, cultivar name and crop alias. The crop ID is strictly unique. Different cultivars of the same crop will have different IDs.
Although the list of crop IDs is extensible, the IDs of well known crops and cultivars are predefined.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when cropID is not positive, and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

2. Gets the length (i.e., number of days) of a particular CropDevelopmentStage of a reference crop. The length is expressed as a range.

[C#] public static int[] GetRefCDS(string crop, CropDevelopmentStage stage);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is a null reference or an empty string, and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

• GetRefCropCycle Overloaded.

1. Gets the length (i.e., number of days) of the crop cycle of a reference crop. The length is expressed as a range.

[C#] public static int[] GetRefCropCycle(int cropID);

Return Value: A 2-element integer array that represents the range of the crop cycle.
Remarks: The crop ID is defined in the database by crop name, cultivar name and crop alias. The crop ID is strictly unique. Different cultivars of the same crop will have different IDs.
Although the list of crop IDs is extensible, the IDs of well known crops and cultivars are predefined.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when cropID is not positive, and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

2. Gets the length (i.e., number of days) of the crop cycle of a reference crop. The length is expressed as a range.

[C#] public static int[] GetRefCropCycle(string crop);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is a null reference or an empty string, and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

• TransformData Overloaded.

1. Transforms a daily data array covering the CropCycle to a daily data array covering a calendar year, or vice versa.

[C#] public static double[] TransformData(double[] source, RomanDate
4.3 CropCycle Class

```csharp
cc1, RomanDate cc2);

Parameters:

source A Double array that represents the source data.
cc1 A RomanDate that represents the start of the CropCycle.
cc2 A RomanDate that represents the end of the CropCycle.

transforms a daily data array covering the CropCycle to a daily data array covering a calendar year, or vice versa.
```

```csharp
public static double[] TransformData(double[] source, CropCycle cc);
```

Parameters:

source A Double array that represents the source data.
cc The CropCycle.

3. Transforms a daily data array covering the CropCycle to a daily data array covering a calendar year, or vice versa.

```csharp
public static double[] TransformData(double[] source, CropCycle cc, CalendarType calendar);
```

Parameters:

source A Double array that represents the source data.
cc The CropCycle.
calendar The RomanDate.CalendarType of the returned data array.

Return Value: A Double array. If the source array covers the time span of the CropCycle, then the returned array covers the time span of a calendar year; and vice versa.

Remarks: This method behaves differently on the length of the source data array.

1. If the source data array covers a calendar year - In other words, the length of the source data array is 360 (if current CropCycle instance uses WlesCalendar) or 365 (if GregorianCalendar). This method copies the part of the data which cover the CropCycle into the returned array. As a result, the returned array is a subset of the source data array.

2. If the source data array covers the crop cycle - In other words, the length of the source data array equals the Length of the current CropCycle. This method copies the source data array into the corresponding slots of the returned array, padding the rest slots with an exceptional value which is guaranteed non-existing in source so that returned array has a length of 360 (if current CropCycle instance uses WlesCalendar) or 365 (if GregorianCalendar). As a result, the returned array is a superset of the source data array.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the source data array is a null reference.</td>
</tr>
</tbody>
</table>

4.3.6 Public Instance Constructors

- public CropCycle(string crop, string cultivar, RomanDate start, RomanDate end, string place)
  Initializes a new instance of the CropCycle class.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when start and end use different calendars.</td>
</tr>
</tbody>
</table>

- public CropCycle(WlesDataSet data);
  Initializes a new instance of the CropCycle class.

Parameters:
data  A WlesDataSet instance that contains data to initialize a new instance of the CropCycle class.

- public CropCycle(RomanDate start, RomanDate end);
  Initializes a new instance of the CropCycle class.
  Remarks: With this constructor, the Place, Crop and Cultivar properties are initialized to empty strings. Their values may be set separately.
  Exceptions:
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when start and end use different calendars.</td>
</tr>
</tbody>
</table>

4.3.7 Public Instance Properties

- Calendar
  Gets the RomanDate.CalendarType of the current CropCycle.
  [C#] public CalendarType Calendar { get; }

- Crop
  Gets or sets the name of the crop.
  [C#] public string Crop { get; set; }

- CropCycleBeginningRomanDate
  Gets the beginning date of the crop cycle.
  [C#] public RomanDate CropCycleBeginningRomanDate { get; }

- CropCycleEndingRomanDate
  Gets the ending date of the crop cycle.
  [C#] public RomanDate CropCycleEndingRomanDate { get; }

- Cultivar
  Gets or sets the name of the cultivar.
  [C#] public string Cultivar { get; set; }

- IsCDSOk
  Determines whether the CropCycle.CropDevelopmentStages values have been assigned.
  [C#] public bool IsCDSOk { get; }

- Length
  Gets the length of the crop cycle.
  [C#] public int Length { get; }

- Place
  Gets or sets the name of the place.
  [C#] public string Place { get; set; }

4.3.8 Public Instance Methods

- DetermineStage
  Overloaded.

  1. Determines the CropCycle.CropDevelopmentStage that a particular date belongs to.
  [C#] public CropDevelopmentStage DetermineStage(int days);
  Parameters:
  days  Number of days (1-based) since the beginning of the CropCycle, suppose the CropCycle begins at day 1.
  Return Value: A CropCycle.CropDevelopmentStage member; -1 if days is zero or negative or greater than the Length of the CropCycle.
2. Determines the CropCycle.CropDevelopmentStage that a particular date belongs to.

[C#] public CropDevelopmentStage DetermineStage(RomanDate date);

Parameters:

date A RomanDate.

Return Value: A CropCycle.CropDevelopmentStage member; -1 if the date is out of the CropCycle.

- ExtractData
Extracts a subset of data for a particular CropCycle.CropDevelopmentStage from a data source that covers the entire CropCycle.

[C#] public double[] ExtractData(double[] source, CropDevelopmentStage cds);

Return Value: A Double array that contains daily data for cds, which is a continuous subset of source.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when length of source array not equal to that of the CropCycle; or cds value not defined.</td>
</tr>
</tbody>
</table>

- GetCDSBeginningRomanDate
Gets the beginning RomanDate of a particular CropCycle.CropDevelopmentStage.

[C#] public RomanDate GetCDSBeginningRomanDate(CropDevelopmentStage stage);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when stage does not equal to one of the following: IS, CD, MS, LS.</td>
</tr>
<tr>
<td>WlesException</td>
<td>Thrown when the values of the crop development stage dates have not been set.</td>
</tr>
</tbody>
</table>

- GetCDSDailyArray
Gets a daily integer array that represents the CropCycle.CropDevelopmentStages within an one-year timeframe.

[C#] public int[] GetCDSDailyArray();

Return Value: An integer array that represents the crop development stages. The length of the array is Calendar dependent: 360 for WlesCalendar and 365 for GregorianCalendar.

Remarks:

1. Length of array - The array has the same length as that of a WlesCalendar or a GregorianCalendar year. The array elements are organized in the same order as the days in a year. Please note that the array is 0 based, while the days of a year in reality are 1 based.

2. Value of element - There are four values: -1 for days outside the CropCycle; 1 for IS stage; 2 for CD stage; 3 for MS stage; and 4 for LS stage.

3. Information conveyed - The following results can be interpreted from the array: (1) the beginning and ending dates of the CropCycle and its length; (2) the beginning and ending dates of the IS stage and its length; (3) the beginning and ending dates of the CD stage and its length; (4) the beginning and ending dates of the MS stage and its length; and (5) the beginning and ending dates of the LS stage and its length.

- GetCDSEndingRomanDate
Gets the ending RomanDate of a particular CropCycle.CropDevelopmentStage.

[C#] public RomanDate GetCDSEndingRomanDate(CropDevelopmentStage stage);

Exceptions:
### Namespace UGent.WLES.RPP

#### exception description
- **ArgumentNullException** Thrown when `stage` does not equal to one of the following: IS, CD, MS, LS.
- **WlesException** Thrown when the values of the crop development stage dates have not been set.

#### GetCDSRomanDates
Overloaded.

1. Gets the beginning and the ending `RomanDates` of the IS, CD, MS and LS stages.
   
   **[C#]**
   ```csharp
   public RomanDate[,] GetCDSRomanDates();
   ```
   
   **Return Value:** A 2-dimensional `RomanDate` array that contains the beginning and the ending dates of the `CropCycle.CropDevelopmentStage` of the `CropCycle`. The array is 4 by 2 in dimension.

   **Exceptions:**
   - **ArgumentNullException** Thrown when `stage` does not equal to one of the following: IS, CD, MS, LS.
   - **WlesException** Thrown when the values of the crop development stage dates have not been set.

2. Gets the beginning and the ending `RomanDates` of a particular `CropCycle.CropDevelopmentStage`.
   
   **[C#]**
   ```csharp
   public RomanDate[] GetCDSRomanDates(CropDevelopmentStage stage);
   ```
   
   **Return Value:** A `RomanDate` array that contains the beginning and the ending `RomanDates` of `stage` as its 1\textsuperscript{st} and 2\textsuperscript{nd} elements.

   **Exceptions:**
   - **ArgumentNullException** Thrown when `stage` does not equal to one of the following: IS, CD, MS, LS.
   - **WlesException** Thrown when the values of the crop development stage dates have not been set.

#### GetCropCycleRomanDates

Gets the beginning and ending dates of the crop cycle expressed in a `RomanDate` array.

**[C#]**
```csharp
public RomanDate[] GetCropCycleRomanDates();
```

**Return Value:** A `RomanDate` array that contains the beginning and the ending `RomanDates` of `stage` as its 1\textsuperscript{st} and 2\textsuperscript{nd} elements.

#### GetLength
Overloaded.

1. Gets the lengths (number of days) of the following crop development stages: IS, CD, MS and LS.
   
   **[C#]**
   ```csharp
   public int[] GetLength();
   ```
   
   **Return Value:** A 4-element `integer` array that contains the length (number of days) of the IS, CD, MS and LS stage as its 1\textsuperscript{st}, 2\textsuperscript{nd}, 3\textsuperscript{rd} and 4\textsuperscript{th} element, respectively.

   **Exceptions:**
   - **WlesException** Thrown when the values of the crop development stage dates have not been set.

2. Gets the length (number of days) of a particular crop development stage.
   
   **[C#]**
   ```csharp
   public int GetLength(CropDevelopmentStage stage);
   ```
   
   **Exceptions:**
   - **WlesException** Thrown when the values of the crop development stage dates have not been set.
4.3 CropCycle Class

- **SetCDSRomanDates**
  Overloaded.

  1. Sets the beginning and ending dates of the `CropCycle.CropDevelopmentStages` of this `CropCycle`.

     **C#**
     ```
     public void SetCDSRomanDates(int lenIS, int lenCD, int lenMS, int lenLS);
     ```

     **Exceptions**:
     
     | exception | description |
     |-----------|-------------|
     | ArgumentException | Thrown when the length of one or more of the crop development stages is not positive. |
     | WlesException | Thrown when the length of the crop cycle not confirmative to the sum of the lengths of crop development stages. |

  2. Sets the beginning and ending dates of the `CropCycle.CropDevelopmentStages` of the current `CropCycle`.

     **C#**
     ```
     public void SetCDSRomanDates(int[] lenCDS);
     ```

     **Parameters**:
     
     `lenCDS` A 4-element integer array that contains the length (number of days) of IS, CD, MS and LS stage as its 1st, 2nd, 3rd and 4th element, respectively.

     **Exceptions**:
     
     | exception | description |
     |-----------|-------------|
     | ArgumentException | Thrown when the length of `lenCDS` array not equal to 4; and when one of the array element not greater than 1, or the sum of all array elements not equal to the length of the crop cycle. |
     | WlesException | Thrown when the length of the crop cycle not confirmative to the sum of the lengths of crop development stages. |

  3. Sets the beginning and ending dates of the `CropCycle.CropDevelopmentStages` of this `CropCycle`.

     **C#**
     ```
     public void SetCDSRomanDates(RomanDate[,] stages);
     ```

     **Parameters**:
     
     `stages` A 2-dimensional `RomanDate` array that contains the beginning and ending dates of the `CropCycle.CropDevelopmentStages` of the `CropCycle`. The array must be 4 by 2 in dimension.

     **Exceptions**:
     
     | exception | description |
     |-----------|-------------|
     | ArgumentException | Thrown when `stages` array is not 4 by 2 in dimension. |
     | WlesException | Thrown when (1) any of the `stages` member not using the same calendar as that of the crop cycle; or (2) the length of the crop cycle not confirmative to the sum of the lengths of crop development stages. |

- **ToString**
  Overloaded.

  1. Delivers a crop cycle report that includes the crop, the cultivar, the place and the beginning and ending dates of the crop cycle and of its crop development stages using default settings.

     **C#**
     ```
     public override string ToString();
     ```

     **Example**: The following example shows the String representation of arbitrary CropCycle instances.

     ```
     using System;
     using UGent.WLES;
     ```
using UGent.WLES.Data;

CropCycle cycle1, cycle2;
RomanDate date1, date2;
string crop, cultivar, place;
// date in GregorianCalendar
date1 = new RomanDate(RomanDate.MonthOfYear.JULY, 10);
// date in WlesCalendar
date2 = new RomanDate(300);
cycle1 = new CropCycle(date1, date2);
crop = "Maize";
cultivar = "Tropical";
place = "Cairo";
cycle2 = new CropCycle(crop, cultivar, date1, date2, place);
Console.WriteLine(cycle1);
/*
 * "Crop cycle: July 10 (WLES Calendar) - October 30 (WLES Calendar)"
 * "Length of crop cycle: 111 days"
 */

// Crop development stages:
// IS = 20 days, CD = 30 days, MS = 40 days and LS = 21 days
cycle2.SetCDSRomanDates(20,30,40,21);
Console.WriteLine(cycle2);
/*
 * "Crop cycle of Maize (Tropical) in Cairo:
 * July 10 (WLES Calendar) - October 30 (WLES Calendar)"
 * "Length of crop cycle: 111 days"
 * "Crop development stages:
 * " IS: July 10 (WLES Calendar) -
 * July 29 (WLES Calendar), 20 days [Initial]"
 * " CD: July 30 (WLES Calendar) -
 * August 9 (WLES Calendar), 30 days [Crop Development]"
 * " MS: August 30 (WLES Calendar) -
 * October 9 (WLES Calendar), 40 days [Mid-Season]"
 * " LS: October 10 (WLES Calendar) -
 * October 30 (WLES Calendar), 21 days [Late Season]"
 */

2. Delivers a crop cycle report that includes the crop, the cultivar, the place and the beginning and ending dates of the crop cycle and of its crop development stages.

[C#]
public string ToString(string newline, string space);

Parameters:

newline A String that represents the new-line (or carriage return) command in string formatting. The new-line command in GUI or console applications is different from that in Web applications or html pages.

space A String that represents a space. The string representation of space in GUI or console applications is different from that in Web applications or html pages.

3. Delivers a crop cycle report that includes the crop, the cultivar, the place and the beginning and ending dates of the crop cycle and of its crop development stages using default settings.

[C#]
public string ToString(CalendarType calendar);

4. Delivers a crop cycle report that includes the crop, the cultivar, the place and the beginning and ending dates of the crop cycle and of its crop development stages.
4.4 GrowingPeriod Class

4.4.1 Summary

Represents the growing period of annual crops within a yearly time frame when temperature, soil water content and other climatic conditions permit crop growth and development.

4.4.2 Declaration

[C#] public class GrowingPeriod

[C#] public string ToString(CalendarType calendar, string newLine, string space);

- TransformData
  Overloaded.
  1. Transforms a daily data array covering the CropCycle to a daily data array covering a calendar year, or vice versa.
  [C#] public double[] TransformData(double[] source);
  2. Transforms a daily data array covering the CropCycle to a daily data array covering a calendar year, or vice versa.
  [C#] public double[] TransformData(double[] source, CalendarType calendar);

Return Value: A Double array. If the source array covers the CropCycle, then the returned array covers a calendar year; and vice versa.
Remarks: This method behaves differently on the length of the source data array.

  1. If the source data array covers a calendar year - In other words, the length of the source data array is 360 (if current CropCycle instance uses WiesCalendar) or 365 (if GregorianCalendar). This method copies the part of the data which cover the CropCycle into the returned array. As a result, the returned array is a subset of the source data array.

  2. If the source data array covers the crop cycle - In other words, the length of the source data array equals the Length of the current CropCycle. This method copies the source data array into the corresponding slots of the returned array, padding the rest slots with an exceptional value which is guaranteed non-existing in source so that returned array has a length of 360 (if current CropCycle instance uses WiesCalendar) or 365 (if GregorianCalendar). As a result, the returned array is a superset of the source data array.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the source data array is a null reference.</td>
</tr>
</tbody>
</table>

- WeightedAverage
  Computes the parameter’s weighted average during the time span of the crop cycle.
  [C#] public double WeightedAverage(double[] parameter);

Remarks: This method is implemented using the Utilities.WeightedAverage method.
4.4.3 Remarks

Growing periods can have wet or humid conditions that limit the possibilities for the ripening and drying of the crops, or which lead to problems of quality of the harvest. The growing period can be defined as the periods of the year in which agricultural production is possible as a result of adequate moisture availability and absence of temperature limitations (Sys et al., 1991).

A growing period may also be called a growing season. But there are subtle differences between these two terms. If there is no ambiguity in the context, a growing season can be interpreted as the synonym of the growing period (see table below). However, a growing season may be the superset of growing periods if discontinuity(-ies) exist(s).

WLES defines growing period as the combination of the rainy period and the subhumid period, from which the cold period has to be excluded. The definition of the growing period concept is summarized in the following table.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humid Period</td>
<td>Rainfall &gt; $ET_o$</td>
<td>[SUBSET of] Rainy Period</td>
</tr>
<tr>
<td>Rainy Period</td>
<td>Rainfall &gt; $ET_o/2$</td>
<td>[SUBSET of] Growing Period</td>
</tr>
<tr>
<td>Subhumid Period</td>
<td>Immediately after Rainy Period, before 100 mm soil $H_2O$ consumed</td>
<td>[SUBSET of] Growing Period</td>
</tr>
<tr>
<td>Dry Period</td>
<td>Rainfall &lt; $ET_o/2$ but after 100 mm soil $H_2O$ depleted</td>
<td>1 - [Rainy Period] + [Subhumid Period]</td>
</tr>
<tr>
<td>Cold Period</td>
<td>$T_{mean}$ &lt; threshold (6.5 °Cdefault)</td>
<td>Has no intersection with Growing Period</td>
</tr>
<tr>
<td>Growing Period</td>
<td>Temperate, moist period that allows crops to grow</td>
<td>[Rainy Period] + [Subhumid Period] - [Cold Period]</td>
</tr>
</tbody>
</table>

4.4.4 Note

The concept of growing period is different from that of the crop (growing) cycle. The crop cycle is the period required for an annual crop to complete its annual cycle of establishment, growth and production of the harvested part. Perennial crops have crop cycles of more than one year (FAO, 1985). In a word, the crop cycle is a property of the crop (namely crop requirements), while the growing period is a property of the land (namely land quality or land characteristic).

4.4.5 Examples

Example 1: The following example shows how to use theGrowingPeriod class to determine the growing period for Kigali in Rwanda.

```csharp
using System;
using UGent.WLES;
using UGent.WLES.Data;
using UGent.WLES.RPP;

public class GrowingPeriodExample1 {
    public static void Main (string [] args) {
        // collect data first
        string place = "Kigali (Rwanda)";
        double elevation = 1500;
        double latitude = (1 + 58.0/60.0) * -1;
        // monthly climatic data
        WlesDataSet.TemporalResolution resolution = WlesDataSet.TemporalResolution.MONTHLY;
        double [] tmax = new double [] {
        };
        double [] tmin = new double [] {
            14.9, 14.9, 15.2, 15.5, 15.5, 14.6, 14.5, 15.3, 15.1, 15.2, 14.9, 15.1
        };
```
GrowingPeriod gp;
gp = new GrowingPeriod(
    resolution, latitude, elevation, tmax, tmin, RH, wind, n, rain);
// name of the place can only be set through the property
gp.Place = place;
Console.WriteLine("{0} (Latitude: {1}degree{2}, Elevation: {3} m",
    gp.Place, gp.Latitude, (gp.Latitude > 0) ? "N" : "S", gp.Elevation);
Console.WriteLine("Growing period inventory: ");
Console.WriteLine();
Console.WriteLine("Growing periods: {0}",
    formDates(gp.GetGrowingSeasonDates()));
Console.WriteLine("{0}",
    countRomanDates(gp.GetGrowingSeasonRomanDates()));
Console.WriteLine("Rainy periods: {0}",
    formDates(gp.GetRainySeasonDates()));
Console.WriteLine("{0}",
    countRomanDates(gp.GetRainySeasonRomanDates()));
Console.WriteLine("Subhumid periods: {0}",
    formDates(gp.GetSubhumidSeasonDates()));
Console.WriteLine("{0}",
    countRomanDates(gp.GetSubhumidSeasonRomanDates()));
Console.WriteLine("Humid periods: {0}",
    formDates(gp.GetHumidSeasonDates()));
Console.WriteLine("{0}",
    countRomanDates(gp.GetHumidSeasonRomanDates()));
Console.WriteLine("Dry periods: {0}",
    formDates(gp.GetDrySeasonDates()));
Console.WriteLine("{0}",
    countRomanDates(gp.GetDrySeasonRomanDates()));
Console.WriteLine("Cold periods: {0}",
    formDates(gp.GetColdSeasonDates()));
Console.WriteLine("{0}",
    countRomanDates(gp.GetColdSeasonRomanDates()));
)

/* print out each period and calculate the length of the period. */
private static string countRomanDates(RomanDate [] [] dates) {
    string result = "";
    foreach (RomanDate [] pair in dates) {
        RomanDate d1, d2;
        d1 = pair[0];
        d2 = pair[1];
        result += string.Format("\t{0} to {1}: {2} days\n", d1, d2, RomanDate.GetLength(d1, d2));
    }
return result;
}

private static string formDates(int[][] dates) {
    string result = "";
    foreach (int[] pair in dates)
    {
        result += string.Format("{0}-{1} ", pair[0], pair[1]);
        result += "\n";
    }
    return result;
}

The example will generate the follow results:

Kigali (Rwanda) (Latitude: 1.97degreeS; Elevation: 1500 m)
Growing period inventory:

Growing periods: 242-178
   September 2 (WLES Calendar) to June 28 (WLES Calendar): 297 days
Rainy periods: 242-150
   September 2 (WLES Calendar) to May 30 (WLES Calendar): 269 days
Subhumid periods: 151-178
   June 1 (WLES Calendar) to June 28 (WLES Calendar): 28 days
Humid periods: 41-62 67-134 295-334
   February 11 (WLES Calendar) to March 2 (WLES Calendar): 22 days
   March 7 (WLES Calendar) to May 14 (WLES Calendar): 68 days
   October 25 (WLES Calendar) to December 4 (WLES Calendar): 40 days
Dry periods: 179-241
   June 29 (WLES Calendar) to September 1 (WLES Calendar): 63 days

Cold periods:

Example 2: The following example shows how to generate a text report of the growing period in Kigali (Rwanda). A sample report is given as the result of this example.

using System;
using System.IO;
using UGent.WLES;
using UGent.WLES.Data;
using UGent.WLES.RPP;

// using same GrowingPeriod instance as in example 1.
GrowingPeriod gp = ...
string report;
// the following method calls are all equivalent!
report = gp.ExportData();
report = gp.ToString();
report = Convert.ToString(gp);
// write the report to file
string filename = "kigali.csv";
using (StreamWriter sw = new StreamWriter(filename)) {
    sw.Write(report);
    /*
     * another equivalent call:
     * sw.Write(gp);
     */
}
The file “kigali.csv” is located in the current directory, and has the following content:

Station: Kigali (Rwanda)
Latitude: 158°12’’ S
Elevation: 1500 M
T-threshold: 6.5C

Growing periods: 242-178
Rainy periods: 242-150
Humid periods: 41-62 67-134 295-334
Subhumid periods: 151-178
Cold periods:

<table>
<thead>
<tr>
<th>Date</th>
<th>Tmean</th>
<th>Rainfall</th>
<th>ETo</th>
<th>ETohalf</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20.54</td>
<td>21.9</td>
<td>33.97</td>
<td>16.985</td>
</tr>
<tr>
<td>2</td>
<td>20.59</td>
<td>21.96</td>
<td>34.36</td>
<td>17.18</td>
</tr>
<tr>
<td>3</td>
<td>20.66</td>
<td>24.34</td>
<td>34.87</td>
<td>17.435</td>
</tr>
<tr>
<td>4</td>
<td>20.83</td>
<td>34.09</td>
<td>36.19</td>
<td>18.095</td>
</tr>
<tr>
<td>5</td>
<td>20.87</td>
<td>37.35</td>
<td>36.43</td>
<td>18.215</td>
</tr>
<tr>
<td>6</td>
<td>20.85</td>
<td>39.16</td>
<td>36.29</td>
<td>18.145</td>
</tr>
<tr>
<td>7</td>
<td>20.7</td>
<td>34.57</td>
<td>35.49</td>
<td>17.745</td>
</tr>
<tr>
<td>8</td>
<td>20.65</td>
<td>37.21</td>
<td>34.76</td>
<td>17.38</td>
</tr>
<tr>
<td>9</td>
<td>20.6</td>
<td>42.12</td>
<td>33.85</td>
<td>16.925</td>
</tr>
<tr>
<td>10</td>
<td>20.58</td>
<td>59.79</td>
<td>32.01</td>
<td>16.005</td>
</tr>
<tr>
<td>11</td>
<td>20.55</td>
<td>61.37</td>
<td>31.28</td>
<td>15.64</td>
</tr>
<tr>
<td>12</td>
<td>20.52</td>
<td>57.34</td>
<td>30.91</td>
<td>15.455</td>
</tr>
<tr>
<td>13</td>
<td>20.51</td>
<td>39.37</td>
<td>30.99</td>
<td>15.495</td>
</tr>
<tr>
<td>14</td>
<td>20.46</td>
<td>30.39</td>
<td>31.3</td>
<td>15.65</td>
</tr>
<tr>
<td>15</td>
<td>20.39</td>
<td>22.04</td>
<td>31.92</td>
<td>15.96</td>
</tr>
<tr>
<td>16</td>
<td>20.2</td>
<td>12.41</td>
<td>33.12</td>
<td>16.56</td>
</tr>
<tr>
<td>17</td>
<td>20.18</td>
<td>6.8</td>
<td>34.16</td>
<td>17.08</td>
</tr>
<tr>
<td>18</td>
<td>20.22</td>
<td>3.29</td>
<td>35.32</td>
<td>17.66</td>
</tr>
<tr>
<td>19</td>
<td>20.33</td>
<td>2.97</td>
<td>37.18</td>
<td>18.5</td>
</tr>
<tr>
<td>20</td>
<td>20.48</td>
<td>2.81</td>
<td>38.07</td>
<td>19.035</td>
</tr>
<tr>
<td>21</td>
<td>21.69</td>
<td>3.92</td>
<td>38.93</td>
<td>19.465</td>
</tr>
<tr>
<td>22</td>
<td>21.16</td>
<td>6.92</td>
<td>39.91</td>
<td>19.955</td>
</tr>
<tr>
<td>23</td>
<td>21.33</td>
<td>10.14</td>
<td>40.14</td>
<td>20.07</td>
</tr>
<tr>
<td>24</td>
<td>21.41</td>
<td>14.08</td>
<td>39.94</td>
<td>19.97</td>
</tr>
<tr>
<td>25</td>
<td>21.3</td>
<td>21.03</td>
<td>38.72</td>
<td>19.36</td>
</tr>
<tr>
<td>26</td>
<td>21.26</td>
<td>24.98</td>
<td>38.11</td>
<td>19.05</td>
</tr>
<tr>
<td>27</td>
<td>21.19</td>
<td>28.15</td>
<td>37.49</td>
<td>18.745</td>
</tr>
<tr>
<td>28</td>
<td>21.1</td>
<td>28.45</td>
<td>37.09</td>
<td>18.545</td>
</tr>
<tr>
<td>29</td>
<td>20.96</td>
<td>31.36</td>
<td>36.35</td>
<td>18.175</td>
</tr>
<tr>
<td>30</td>
<td>20.78</td>
<td>34.89</td>
<td>35.46</td>
<td>17.73</td>
</tr>
<tr>
<td>31</td>
<td>20.34</td>
<td>43.84</td>
<td>33.71</td>
<td>16.855</td>
</tr>
<tr>
<td>32</td>
<td>20.21</td>
<td>45.33</td>
<td>33.07</td>
<td>16.535</td>
</tr>
<tr>
<td>33</td>
<td>20.19</td>
<td>43.16</td>
<td>32.81</td>
<td>16.405</td>
</tr>
<tr>
<td>34</td>
<td>20.44</td>
<td>33.22</td>
<td>33.41</td>
<td>16.705</td>
</tr>
<tr>
<td>35</td>
<td>20.52</td>
<td>29.25</td>
<td>33.59</td>
<td>16.795</td>
</tr>
<tr>
<td>36</td>
<td>20.56</td>
<td>26.13</td>
<td>33.81</td>
<td>16.905</td>
</tr>
</tbody>
</table>

The data can be imported into a third party program, such as Microsoft Excel, to draw curves and to visually interpret the relationships.
4.4.6 Public Instance Constructors

- public GrowingPeriod(WlesDataSet data);
  Initializes a new instance of the GrowingPeriod class.
  Parameters:

  data A WlesDataSet that contains data to initialize a new instance of the GrowingPeriod class.

- public GrowingPeriod(TemporalResolution resolution, double latitude, double elevation, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine, double[] rainfall);
  Initializes a new instance of the GrowingPeriod class.
  Parameters:

  resolution The WlesDataSet.TemporalResolution of the data it passes.
  latitude Latitude of the considered area. Northern Hemisphere has positive values, while Southern Hemisphere has negative. Values are expressed in decimal degrees.
  elevation Elevation of the considered area. Values are expressed in meters above sea level.
  tmax Maximum daily air temperature (°C).
  tmin Minimum daily air temperature (°C).
  RH Relative air humidity (%).
  U2 Wind velocity at 2 m above ground (m s⁻¹).
  sunshine Sunshine hours (hr).
  rainfall Rainfall (mm).

  Exceptions:

  WlesException Thrown when (1) latitude not in range [-90, 90]; or (2) Parameters tmax, tmin, RH, U2, sunshine and rainfall do not have the same length or do not confirm with the specified resolution.

- public GrowingPeriod(TemporalResolution resolution, CalendarType calendar, double latitude, double elevation, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine, double[] rainfall);
  Initializes a new instance of the GrowingPeriod class.
  Parameters:

  resolution The WlesDataSet.TemporalResolution of the data it passes.
  calendar The RomanDate.CalendarType associated with the current instance.
  latitude Latitude of the considered area. Northern Hemisphere has positive values, while Southern Hemisphere has negative. Values are expressed in decimal degrees.
  elevation Elevation of the considered area. Values are expressed in meters above sea level.
  tmax Maximum daily air temperature (°C).
  tmin Minimum daily air temperature (°C).
  RH Relative air humidity (%).
  U2 Wind velocity at 2 m above ground (m s⁻¹).
  sunshine Sunshine hours (hr).
  rainfall Rainfall (mm).
4.4 GrowingPeriod Class

Exceptions:

```
exception   description
WlesException Thrown when (1) latitude not in range [-90, 90]; or (2) Parameters tmax, tmin, RH, U2, sunshine and rainfall do not have the same length or do not confirm with the specified resolution.
```

Remarks: The constructor takes the dataset which is needed for the determination of the growing period. Depending on the WlesDataSet.TemporalResolution, the dataset contains either monthly, decadal, or daily data.

4.4.7 Public Instance Properties

- **ColdPeriodBeginningDates**
  Gets an integer array that represents the start(s) of the cold period(s).

  ```c#```
  public int[] ColdPeriodBeginningDates { get; }
  ```

  **Property Value:** An integer array. The length of the array is zero if the cold season does not exist.

  **Remarks:** Depending on the climatic conditions, the cold season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

- **ColdPeriodEndingDates**
  Gets an integer array that represents the end(s) of the cold period(s).

  ```c#```
  public int[] ColdPeriodEndingDates { get; }
  ```

  **Property Value:** An integer array. The length of the array is zero if the cold season does not exist.

  **Remarks:** Depending on the climatic conditions, the cold season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

- **ColdSeason**
  Gets a Boolean array representation of the cold season.

  ```c#```
  public bool[] ColdSeason { get; }
  ```

  **Property Value:** A Boolean array. Length of the array is WorkingCalendar-dependent: 360 if WlesCalendar and 360 if GregorianCalendar.

  **Remarks:**
  1. Returned array is 0-based.
  2. The \(i\)th element is true means the \((i + 1)\)th day of the year is in the growing season.
  3. Please refer to the remarks section of the GrowingPeriod class for the semantical differences between terms period and season.

- **DataResolution**
  Returns the WlesDataSet.TemporalResolution of input data.

  ```c#```
  public TemporalResolution DataResolution { get; }
  ```

- **DryPeriodBeginningDates**
  Gets an integer array that represents the start(s) of the dry period(s).

  ```c#```
  public int[] DryPeriodBeginningDates { get; }
  ```

  **Property Value:** An integer array. The length of the array is zero if the dry season does not exist.

  **Remarks:** Depending on the climatic conditions, the dry season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.
- **DryPeriodEndingDates**
  Gets an integer array that represents the end(s) of the dry period(s).

```csharp
public int[] DryPeriodEndingDates { get; }
```

**Property Value:** An integer array. The length of the array is zero if the dry season does not exist.

**Remarks:** Depending on the climatic conditions, the dry season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

- **DrySeason**
  Gets a Boolean array representation of the dry season.

```csharp
public bool[] DrySeason { get; }
```

**Property Value:** A Boolean array. Length of the array is WorkingCalendar-dependent: 360 if WlesCalendar and 360 if GregorianCalendar.

**Remarks:**

1. Returned array is 0-based.
2. The $i^{th}$ element is true means the $(i + 1)^{th}$ day of the year is in the growing season.
3. Please refer to the remarks section of the GrowingPeriod class for the semantical differences between terms period and season.

- **Elevation**
  Gets the Double representation of the elevation (m, above sea level).

```csharp
public double Elevation { get; }
```

- **GrowingPeriodBeginningDates**
  Gets an integer array that represents the start(s) of the growing period(s).

```csharp
public int[] GrowingPeriodBeginningDates { get; }
```

**Property Value:** An integer array. The length of the array is zero if the dry season does not exist.

**Remarks:** Depending on the climatic conditions, the dry season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

- **GrowingPeriodEndingDates**
  Gets an integer array that represents the end(s) of the growing period(s).

```csharp
public int[] GrowingPeriodEndingDates { get; }
```

**Property Value:** An integer array. The length of the array is zero if the dry season does not exist.

**Remarks:** Depending on the climatic conditions, the dry season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

- **GrowingSeason**
  Gets a Boolean array representation of the growing season.

```csharp
public bool[] GrowingSeason { get; }
```

**Property Value:** A Boolean array. Length of the array is WorkingCalendar-dependent: 360 if WlesCalendar and 360 if GregorianCalendar.

**Remarks:**

1. Returned array is 0-based.
2. The $i^{th}$ element is true means the $(i + 1)^{th}$ day of the year is in the growing season.
3. Please refer to the remarks section of the GrowingPeriod class for the semantical differences between terms period and season.
• HumidPeriodBeginningDates
  Gets an integer array that represents the start(s) of the humid period(s).

  [C#] public int[] HumidPeriodBeginningDates { get; }

  Property Value: An integer array. The length of the array is zero if the dry season does not exist.

  Remarks: Depending on the climatic conditions, the dry season may consists of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

• HumidPeriodEndingDates
  Gets an integer array that represents the end(s) of the humid period(s).

  [C#] public int[] HumidPeriodEndingDates { get; }

  Property Value: An integer array. The length of the array is zero if the dry season does not exist.

  Remarks: Depending on the climatic conditions, the dry season may consists of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

• HumidSeason
  Gets a Boolean array representation of the humid season.

  [C#] public bool[] HumidSeason { get; }

  Property Value: A Boolean array. Length of the array is WorkingCalendar-dependent: 360 if WlesCalendar and 360 if GregorianCalendar.

  Remarks:
  1. Returned array is 0-based.
  2. The \(i^{th}\) element is true means the \((i + 1)^{th}\) day of the year is in the growing season.
  3. Please refer to the remarks section of the GrowingPeriod class for the semantical differences between terms period and season.

• Latitude
  Gets the Double representation of the latitude (°).

  [C#] public double Latitude { get; }

  Remarks: Northern Hemisphere locations have positive latitudes, and Southern Hemisphere have negative.

• Longitude
  Gets the Double representation of the longitude (°).

  [C#] public double Longitude { get; set; }

  Remarks:
  1. Eastern Hemisphere locations have positive latitudes, and Western Hemisphere negative.
  2. The longitude value should be set through this property rather than the constructors.
  3. Its state can be checked by the IsLongitudeOk method.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown if the Longitude value is unknown when accessed.</td>
</tr>
</tbody>
</table>

• MaximumAirTemperature
  Gets the Double array representation of the year-round maximum air temperature (°C).

  [C#] public double[] MaximumAirTemperature { get; }

  Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.
• MeanAirTemperature
  Gets the Double array representation of the year-round mean air temperature (°C).
  
  [C#] public double[] MeanAirTemperature { get; }
  Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.

• MinimumAirTemperature
  Gets the Double array representation of the year-round minimum air temperature (°C).
  
  [C#] public double[] MinimumAirTemperature { get; }
  Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.

• Place
  Gets or sets the String representation of the name of the study area.
  
  [C#] public string Place { get; set; }
  Remarks: Before a String is assigned to this property, the leading and tailing white spaces, if any, are trimmed. If the string is a null reference, then an empty string is passed in instead.

• Rainfall
  Gets the Double array representation of the year-round rainfall (mm).
  
  [C#] public double[] Rainfall { get; }
  Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.

• RainyPeriodBeginningDates
  Gets an integer array that represents the start(s) of the rainy period(s).
  
  [C#] public int[] RainyPeriodBeginningDates { get; }
  Property Value: An integer array. The length of the array is zero if the dry season does not exist.
  Remarks: Depending on the climatic conditions, the dry season may consists of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

• RainyPeriodEndingDates
  Gets an integer array that represents the end(s) of the rainy period(s).
  
  [C#] public int[] RainyPeriodEndingDates { get; }
  Property Value: An integer array. The length of the array is zero if the dry season does not exist.
  Remarks: Depending on the climatic conditions, the dry season may consists of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

• RainySeason
  Gets a Boolean array representation of the rainy season.
  
  [C#] public bool[] RainySeason { get; }
  Property Value: A Boolean array. Length of the array is WorkingCalendar-dependent: 360 if WlesCalendar and 360 if GregorianCalendar.
  Remarks:
  1. Returned array is 0-based.
  2. The \(i\)th element is true means the \((i + 1)\)th day of the year is in the growing season.
  3. Please refer to the remarks section of the GrowingPeriod class for the semantical differences between terms period and season.

• ReferenceEvapotranspiration
  Gets the Double array representation of the year-round reference evapotranspiration (mm decade\(^{-1}\), or mm day\(^{-1}\)).
4.4 GrowingPeriod Class

[C#] public double[] ReferenceEvapotranspiration { get; }
Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.

• RelativeAirHumidity
  Gets the Double array representation of the year-round RelativeAirHumidity (%).
[C#] public double[] RelativeAirHumidity { get; }
Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.

• SubhumidPeriodBeginningDates
  Gets an integer array that represents the start(s) of the subhumid period(s).
[C#] public int[] SubhumidPeriodBeginningDates { get; }
Property Value: An integer array. The length of the array is zero if the dry season does not exist.
Remarks: Depending on the climatic conditions, the dry season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

• SubhumidPeriodEndingDates
  Gets an integer array that represents the end(s) of the subhumid period(s).
[C#] public int[] SubhumidPeriodEndingDates { get; }
Property Value: An integer array. The length of the array is zero if the dry season does not exist.
Remarks: Depending on the climatic conditions, the dry season may consist of more than one periods. The length of the array represents the number of periods in the season; The element of the array represents the number of days (1-based) since the beginning of the year.

• SubhumidSeason
  Gets a Boolean array representation of the subhumid season.
[C#] public bool[] SubhumidSeason { get; }
Property Value: A Boolean array. Length of the array is WorkingCalendar-dependent: 360 if WiesCalendar and 360 if GregorianCalendar.
Remarks:
1. Returned array is 0-based.
2. The i<sup>th</sup> element is true means the (i + 1)<sup>th</sup> day of the year is in the growing season.
3. Please refer to the remarks section of the GrowingPeriod class for the semantical differences between terms period and season.

• SunshineHours
  Gets the Double array representation of the year-round sunshine hours (hr).
[C#] public double[] SunshineHours { get; }
Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.

• TemperatureThreshold
  Gets or sets the value of the temperature threshold (°C).
[C#] public double TemperatureThreshold { get; set; }
Property Value: A Double that represents the threshold, initially set as 6.5 °C.
Remarks: Crops stop growing when temperature decreases below the threshold. The default value is 6.5 °C. The threshold may be set through the TemperatureThreshold property.

• WindSpeedAt2M
  Gets the Double array representation of the year-round wind velocity (m s<sup>-1</sup>).
[C#] public double[] WindSpeedAt2M { get; }
Remarks: The length of the returned array is dependent on the data passed to the GrowingPeriod class constructor: 36 if decadely or monthly data are passed; 360 if daily data.
• WorkingCalendar
  Returns the `RomanDate.CalendarType` of the current instance.

  ```c#
  public CalendarType WorkingCalendar { get; }
  ```

4.4.8 Public Instance Methods

• ExportData
  Overloaded.
  
  1. Exports a list of comma delimited data, which may be imported to a third party program to plot curves, together with a summary report of the starting and the ending dates of the growing periods, using default settings.

     ```c#
     public string ExportData();
     ```
  
  2. Exports a list of comma delimited data, which may be imported to a third party program to plot curves, together with a summary report of the starting and the ending dates of the growing periods, using default settings.

     ```c#
     public string ExportData(string newLine);
     ```
     
     **Example:** Please refer to example 2 in the `GrowingPeriod` class overview.

• GetColdSeasonDates
  Gets a jagged `integer` array that represents the date pairs of the cold periods within the whole cold season.

  ```c#
  public int[][] GetColdSeasonDates();
  ```

• GetColdSeasonRomanDates
  Gets a jagged `RomanDate` array that represents the date pairs of the cold periods within the whole cold season.

  ```c#
  public RomanDate[][] GetColdSeasonRomanDates();
  ```

• GetDrySeasonDates
  Gets a jagged `integer` array that represents the date pairs of the dry periods within the whole dry season.

  ```c#
  public int[][] GetDrySeasonDates();
  ```

• GetDrySeasonRomanDates
  Gets a jagged `RomanDate` array that represents the date pairs of the dry periods within the whole dry season.

  ```c#
  public RomanDate[][] GetDrySeasonRomanDates();
  ```

• GetGrowingSeasonDates
  Gets a jagged `integer` array that represents the date pairs of the growing periods within the whole growing season.

  ```c#
  public int[][] GetGrowingSeasonDates();
  ```

**Example:** The following example shows the way of calling `GetGrowingSeasonDates()` method and interprets the result.

```c#
using System;
using UGent.WLES.RPP;

GrowingPeriod gp;
// instantiate with data
gp = new GrowingPeriod(...);
int numberOfGrowingPeriods, startDate, endDate;
int [][] datePairs;
```
datePairs = gp.GetGrowingSeasonDates();
numberOfGrowingPeriods = datePairs.Length;
Console.WriteLine("There are {0} periods in the growing season: ",
    numberOfGrowingPeriods);
int [] len = new int [numberOfGrowingPeriods];
int total_len = 0;
for ( int i = 0; i < numberOfGrowingPeriods; i++ ) {
    len[i] = datePairs[i][1] - datePairs[i][0] + 1;
    Console.WriteLine("Period {0}: starts on date {1} and ends on
        date {2}, lasts for {3} days.", i+1, datePairs[i][0],
        datePairs[i][1], len[i]);
    total_len += len[i];
}
Console.WriteLine("The total length of the growing season is {0}
    days.", total_len);

The example generates the following hypothetical results:

There are 3 periods in the growing season:
Period 1: starts on date 10 and ends on date 156, lasts for 147 days.
Period 2: starts on date 200 and ends on date 280, lasts for 81 days.
Period 3: starts on date 305 and ends on date 325, lasts for 21 days.
The total length of the growing season is 249 days.

- GetGrowingSeasonRomanDates
  Gets a jagged RomanDate array that represents the date pairs of the growing periods within
  the whole growing season.
[C#] public RomanDate[][] GetGrowingSeasonRomanDates();

- GetHumidSeasonDates
  Gets a jagged integer array that represents the date pairs of the humid periods within
  the whole humid season.
[C#] public int[][] GetHumidSeasonDates();

- GetHumidSeasonRomanDates
  Gets a jagged RomanDate array that represents the date pairs of the humid periods within
  the whole humid season.
[C#] public RomanDate[][] GetHumidSeasonRomanDates();

- GetLength
  Gets the length of the growing season, expressed in number of days.
[C#] public int GetLength();
  Remarks: This method is equivalent to the GetLengthOfGrowingSeason method.

- GetLengthOfColdSeason
  Gets the length of the cold season, expressed in number of days.
[C#] public int GetLengthOfColdSeason();

- GetLengthOfDrySeason
  Gets the length of the dry season, expressed in number of days.
[C#] public int GetLengthOfDrySeason();

- GetLengthOfGrowingSeason
  Gets the length of the growing season, expressed in number of days.
[C#] public int GetLengthOfGrowingSeason();
- GetLengthOfHumidSeason
  Gets the length of the humid season, expressed in number of days.
  [C#] public int GetLengthOfHumidSeason();

- GetLengthOfRainySeason
  Gets the length of the rainy season, expressed in number of days.
  [C#] public int GetLengthOfRainySeason();

- GetLengthOfSubhumidSeason
  Gets the length of the subhumid season, expressed in number of days.
  [C#] public int GetLengthOfSubhumidSeason();

- GetRainySeasonDates
  Gets a jagged integer array that represents the date pairs of the rainy periods within the whole rainy season.
  [C#] public int[][] GetRainySeasonDates();

- GetRainySeasonRomanDates
  Gets a jagged RomanDate array that represents the date pairs of the rainy periods within the whole rainy season.
  [C#] public RomanDate[][] GetRainySeasonRomanDates();

- GetSubhumidSeasonDates
  Gets a jagged integer array that represents the date pairs of the subhumid periods within the whole subhumid season.
  [C#] public int[][] GetSubhumidSeasonDates();

- GetSubhumidSeasonRomanDates
  Gets a jagged RomanDate array that represents the date pairs of the subhumid periods within the whole subhumid season.
  [C#] public RomanDate[][] GetSubhumidSeasonRomanDates();

- IsLongitudeOk
  Determines whether the Longitude value is initialized.
  [C#] public bool IsLongitudeOk();

- ToString
  Delivers a growing period report that consists of (1) a summary of the starting and ending dates of the growing periods and (2) a list of comma delimited data that may be imported to a third party program to plot curves.
  [C#] public override string ToString();
  Remarks: The ToString method is a pointer that points to the ExportData method.

### 4.5 GrowingPeriodUI Class

#### 4.5.1 Summary

Represents the GrowingPeriod determination and reporting interface of the WLES Web UI.

#### 4.5.2 Declaration

[C#] public class GrowingPeriodUI : Page
4.5.3 Remarks
This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

```
gp.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

4.5.4 Internal Static Methods

- SetupGrowingPeriod
  Sets up a GrowingPeriod using data associated with a particular Web Form page.
  
  [C#] internal static GrowingPeriod SetupGrowingPeriod(Page page);

  Parameters:

  `page` The Web Form page being accessed.

  Return Value: A GrowingPeriod instance.

4.6 RPPReport Class

4.6.1 Summary
Represents the UGent.WLES.RPP reporting interface of the WLES Web UI.

4.6.2 Declaration

[C#] public class RPPReport : Page

4.6.3 Remarks
This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

```
rpp.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

4.6.4 Internal Static Methods

- SetupBiomass
  Sets up a Biomass instance using data associated with a particular Web Form page.
  
  [C#] internal static Biomass SetupBiomass(Page page);

  Parameters:

  `page` The Web Form page being accessed.

  Return Value: A Biomass instance, whose Evaluate method must be called to make the RPP value meaningful.
4.7 Biomass.PhotosynGroup Enumeration

4.7.1 Summary

Specifies the photosynthetical group ID of crops, i.e., I through V.

4.7.2 Declaration

[C#] public enum Biomass.PhotosynGroup

4.7.3 Remarks

The PhotosynGroup enumeration represents the group IDs to which crops are classified on the basis of the photosynthesis pathways and other physiological and geophysical criteria (FAO-UNESCO, 1981).

This enumeration is useful when it is desirable to have a strongly typed specification of the crop groupings. For example, this enumeration is among the formal parameters of the CalculatePm method.

4.7.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Group I</td>
</tr>
<tr>
<td>II</td>
<td>Group II</td>
</tr>
<tr>
<td>III</td>
<td>Group III</td>
</tr>
<tr>
<td>IV</td>
<td>Group IV</td>
</tr>
<tr>
<td>V</td>
<td>Group V</td>
</tr>
</tbody>
</table>

4.8 Biomass.PhotosynPathway Enumeration

4.8.1 Summary

Specifies the photosynthetical pathways of crops.

4.8.2 Declaration

[C#] public enum Biomass.PhotosynPathway

4.8.3 Remarks

Photosynthetical pathways reflect crop’s photosynthetical adaptability. The two major pathways are the C3 pathway and the C4 pathway. The third, and the minor, pathway is the CAM pathway.

4.8.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3</td>
<td>C3 pathway. The 1st product of photosynthesis in this pathway is a 3-carbon organic acid (3-phosphoglyceric acid).</td>
</tr>
<tr>
<td>C4</td>
<td>C4 pathway. The 1st product of photosynthesis in this pathway is a 4-carbon organic acid (malate and asparate).</td>
</tr>
<tr>
<td>CAM</td>
<td>CAM pathway. CAM stands for crassulacean acid metabolism.</td>
</tr>
</tbody>
</table>
4.9 CropCycle.CropDevelopmentStage Enumeration

4.9.1 Summary
Specifies the crop development stages (CDS) of a CropCycle.

4.9.2 Declaration
[C#] public enum CropCycle.CropDevelopmentStage

4.9.3 Remarks
As the crop develops, the ground cover, crop height and the leaf area change. Due to differences in evapotranspiration during the various development stages, the CropCoefficient \( k_c \) for a given crop varies over the CropCycle.

The whole CropCycle can be divided into four distinct development stages:

1. Initial
2. Crop development
3. Mid-season
4. Late season

An illustration is given in Figure 4.1 for general sequence and proportion of above mentioned stages for different types of crops.

The differences and similarities between the crop development stages and the crop growth periods are discussed in the documentation of the CropCycle.CropGrowthPeriod enumeration.

Figure 4.1: Sequence and proportion of crop development stages for different types of crops. Source: Allen et al. (1998)
4.9.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>Represents the initial stage. <strong>Note:</strong> The initial stage runs from planting date to approximately 10% ground cover. The length of the initial period is highly dependent on the crop, the crop variety, the planting date and the climate. The end of the initial period is determined as the time when approximately 10% of the ground surface is covered by green vegetation. For perennial crops, the planting date is replaced by the “green-up” date, <em>i.e.</em>, the time when the initiation of new leaves occurs.</td>
</tr>
<tr>
<td>CD</td>
<td>Represents the crop development stage. <strong>Note:</strong> (1) The crop development stage runs from 10% ground cover to effective full cover. Effective full cover for many crops occurs at the initiation of flowering. For row crops where rows commonly interlock leaves such as beans, sugar beets, potatoes and corn, effective cover can be defined as the time when some leaves of plants in adjacent rows begin to intermingle so that soil shading becomes nearly complete, or when plants reach nearly full size if no intermingling occurs. Because it is difficult to visually determine when densely sown vegetation such as winter and spring cereals and some grasses reach effective full cover, the more easily detectable stage of heading (flowering) is generally used for these types of crops. (2) Another way to estimate the occurrence of effective full cover is when the leaf area index (LAI) reaches 3 m$^2$ m$^{-2}$. LAI is defined as the average total area of leaves (one side) per unit area of ground surface.</td>
</tr>
<tr>
<td>MS</td>
<td>Represents the mid-season stage. <strong>Note:</strong> (1) The mid-season stage runs from effective full cover to the start of maturity. The start of maturity is often indicated by the beginning of the ageing, yellowing or senescence of leaves, leaf drop, or the browning of fruit to the degree that the crop evapotranspiration ($ET_o$) is reduced relative to the reference $ET_o$. The mid-season stage is the longest stage for perennials and for many annuals, but it may be relatively short for vegetable crops that are harvested fresh for their green vegetation.</td>
</tr>
<tr>
<td>LS</td>
<td>Represents the late-season stage. <strong>Note:</strong> The late season stage runs from the start of maturity to harvest or full senescence. For some perennial vegetation in frost free climates, crops may grow year round so that the date of termination may be taken as the same as the date of “planting”.</td>
</tr>
<tr>
<td>AH</td>
<td>Represents the at-harvest stage. <strong>Note:</strong> At-harvest is a virtual stage. It is defined in WLES as the same as the end of the late-season stage. Therefore the length of AH stage is one day.</td>
</tr>
</tbody>
</table>

4.10 CropCycle.CropGrowthPeriod Enumeration

4.10.1 Summary

Specifies the crop growth periods (CGP) within a CropCycle.

4.10.2 Declaration

[C#] public enum CropCycle.CropGrowthPeriod

4.10.3 Remarks

During the growth span, the crop passes through various phases and the periods of growth. The growth rhythm of crops is slow during some periods and fast during some others. Hence the demand of water supply and, more importantly, response to water deficit varies during the whole CropCycle.

The CropCycle is divided into five distinct growth periods:

1. Establishment(0)
2. Vegetative(1)
3. Flowering(2)
4. Yield formation(3)
5. Ripening(4)

The *CropCycle.CropGrowthPeriods* and the *CropCycle.CropDevelopmentStages* are defined on the basis of different criteria and served for different purposes. Among others, the *CropCycle.CropDevelopmentStages* are used for determination of the *CropCoefficient* in the process of estimating the maximum crop evapotranspiration ($ET_c$ or $ET_m$), whereas the *CropCycle.CropGrowthPeriods* are used to differentiate the crop sensitivity to water deficit in the process of evaluating the *UGent.WLES.WPP*.

The *CropCycle.CropDevelopmentStages* and the *CropCycle.CropGrowthPeriods* of the same *CropCycle* can be roughly *inter-mapped* to each other (see table below) if independent measurements are not available for one of the two.

<table>
<thead>
<tr>
<th>Crop Growth Periods</th>
<th>Crop Development Stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP 0: Establishment</td>
<td>CDS 1: Initial</td>
</tr>
<tr>
<td>CGP 1: Vegetative</td>
<td>CDS 2: Crop development</td>
</tr>
<tr>
<td>CGP 2: Flowering &amp; CGP 3: Yield formation</td>
<td>CDS 3: Mid-season</td>
</tr>
<tr>
<td>CGP 4: Ripening</td>
<td>CDS 4: Late season</td>
</tr>
<tr>
<td>Total = Crop cycle</td>
<td>Total = Crop cycle</td>
</tr>
</tbody>
</table>

### 4.11.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establishment</td>
<td>Represents the establishment period.</td>
</tr>
<tr>
<td>Vegetative</td>
<td>Represents the vegetative period.</td>
</tr>
<tr>
<td>Flowering</td>
<td>Represents the flowering period.</td>
</tr>
<tr>
<td>YieldFormation</td>
<td>Represents the yield formation period.</td>
</tr>
<tr>
<td>Ripening</td>
<td>Represents the ripening period.</td>
</tr>
</tbody>
</table>

### 4.11 CropCycle.SubCGP Enumeration

#### 4.11.1 Summary

Specifies the sub division within a crop growth period (CGP).

#### 4.11.2 Declaration

[C#] public enum CropCycle.SubCGP

#### 4.11.3 Remarks

A *CropCycle.CropGrowthPeriod* may be divided into two sub-periods: the *early-half* and the *late-half*, to reflect the intra-period fluctuations of the yield response factor values.

#### 4.11.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early</td>
<td>The first half of a <em>CropCycle.CropGrowthPeriod</em>.</td>
</tr>
<tr>
<td>Late</td>
<td>The second half of a <em>CropCycle.CropGrowthPeriod</em>.</td>
</tr>
</tbody>
</table>
Chapter 5

Namespace UGent.WLES.WPP

The UGent.WLES.WPP namespace contains all necessary classes and structures for estimating the water limited crop production potential (WPP). WPP is based on the rainfed crop production scenario in which water stress occurs in the crop cycle. The anticipated rainfed yield is estimated from the potential irrigated yield (RPP) and a reduction factor which expresses the relative yield loss as the result of the water deficit.

5.1 Overview

5.1.1 Classes

<table>
<thead>
<tr>
<th>type summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CropCoefficient</td>
<td>Represents the crop coefficient and encapsulates the daily crop coefficient data during the CropCycle of a particular crop.</td>
</tr>
<tr>
<td>CropCoefficientUI</td>
<td>Represents the CropCoefficient determination and reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>DepletionFractionUI</td>
<td>Represents the soil water depletion fraction (p) determination and reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>EffectiveRainfall</td>
<td>Represents the effective rainfall.</td>
</tr>
<tr>
<td>EffectiveRainfallUI</td>
<td>Represents the EffectiveRainfall determination and reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>PenmanMonteith</td>
<td>Represents the estimation of the reference evapotranspiration (ET₀) using the Penman-Monteith approach.</td>
</tr>
<tr>
<td>RootingDepth</td>
<td>Represents the rooting depth dynamics during the CropCycle.</td>
</tr>
<tr>
<td>RootingDepthUI</td>
<td>Represents the RootingDepth determination and reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>WaterBalance</td>
<td>Represents the simulation of the plant available soil water dynamics during the CropCycle, and the evaluation of the actual crop evapotranspiration (ETₐ), the yield reducing coefficient due to water stress (fₖ) and the water-limited production potential (WPP).</td>
</tr>
<tr>
<td>WaterBalanceUI</td>
<td>Represents the WaterBalance simulation and reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>WPPReport</td>
<td>Represents the UGent.WLES.WPP reporting interface of the WLES Web UI.</td>
</tr>
<tr>
<td>YieldResponseToWater</td>
<td>Represents the crop’s yield response to water stress.</td>
</tr>
<tr>
<td>YieldResponseToWaterUI</td>
<td>Represents the yield response factor determination and reporting interface of the WLES Web UI.</td>
</tr>
</tbody>
</table>
5.1.2 Delegates

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectiveRainfall.UserDefinedMethod</td>
<td>Provides an interface for the client to plug in his/her algorithm to estimate the effective rainfall.</td>
<td>231</td>
</tr>
</tbody>
</table>

5.1.3 Enumerations

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectiveRainfall.Approach</td>
<td>Specifies available methods to estimate the effective rainfall. The WLES implements 4 methods, namely the fixed percentage method, the FAO/AGLW method, the USBR method and the USDA/SCS method. The class also allows a user defined method to be plugged in to estimate the effective rainfall using an ad hoc approach that is well adapted to the local conditions.</td>
<td>231</td>
</tr>
<tr>
<td>WaterBalance.SoilWaterDepletionGroup</td>
<td>Specifies the crop group ID according to the fraction to which the available soil water can be depleted while maintaining $ET_c$ equal to $ET_m$.</td>
<td>232</td>
</tr>
</tbody>
</table>

5.2 CropCoefficient Class

5.2.1 Summary

Represents the crop coefficient and encapsulates the daily crop coefficient data during the CropCycle of a particular crop.

5.2.2 Declaration

[C#] public class CropCoefficient

5.2.3 Remarks

1. Crop coefficient ($k_c$) is defined as the ratio of the crop evapotranspiration ($ET_c$ or $ET_m$) over the reference evapotranspiration ($ET_o$). It accounts for the effect of crop characteristics on crop water requirements:

$$ET_c = ET_m = k_c \cdot ET_o$$

2. Crop coefficient is a dimensionless coefficient and originally its value is empirically determined. It is affected by crop characteristics, time of planting or sowing, stages of crop development and general climatic conditions which include in the 1st place the wind velocity and the minimum relative air humidity.

3. A CropCoefficient instance is instanciated with climatic parameters during particular CropCycle.CropDevelopmentStage enumerations. This process can be simplified by using a WlesDataSet. The $k_c$ values are automatically evaluated for the entire CropCycle.

4. Nevertheless, the $k_c$ values for IS, MS and AH can be explicitly set through the KcIS, KcMS and KcAH properties respectively.

The following stages are defined in a crop cycle for the determination of the crop coefficient (see: CropCycle.CropDevelopmentStage enumeration):
5.2 CropCoefficient Class

1. IS - Initial stage
2. CD - Crop development stage
3. MS - Mid-season stage
4. LS - Late-season stage
5. AH - At-harvest stage

5.2.4 Public Static Methods

- **DetermineKcAH**
  Determines the crop coefficient \(k_c\) value for the AH stage.

  ```csharp
  public static double DetermineKcAH(string crop, string cultivar, double RH, double U2);
  ``

  **Parameters:**

  - `crop` A `String` that represents the name of the crop.
  - `cultivar` A `String` that represents the name of the cultivar.
  - `RH` A `Double` that represents the average relative humidity of the AH stage, expressed in percentage (%).
  - `U2` A `Double` that represents the average wind velocity (at 2 m above ground) of the AH stage, expressed in m s\(^{-1}\).

  **Exceptions:**

  - `ArgumentException` Thrown when one or more of the following situations encountered: (1) `crop` string is empty; (2) `cultivar` string is empty if `crop` is cucumber or sugarbeet; (3) RH or U2 is negative; (4) database query returns nothing.
  - `WlesDatabaseException` Thrown when humidity or wind velocity values can not be determined.

- **DetermineKcIS**
  Determines the crop coefficient \(k_c\) value for the IS stage.

  ```csharp
  public static double DetermineKcIS(double interval, double ETo);
  ``

  **Parameters:**

  - `interval` Average number of days between adjacent significant rains or irrigations during the IS stage.
  - `ETo` Average daily reference evapotranspiration during the IS stage, mm day\(^{-1}\).

- **DetermineKcMS**
  Determines the crop coefficient \(k_c\) value for the MS stage.

  ```csharp
  public static double DetermineKcMS(string crop, string cultivar, double RH, double U2);
  ``

  **Parameters:**

  - `crop` A `String` that represents the name of the crop.
  - `cultivar` A `String` that represents the name of the cultivar.
  - `RH` A `Double` that represents the average relative humidity of the AH stage, expressed in percentage (%).
  - `U2` A `Double` that represents the average wind velocity (at 2 m above ground) of the AH stage, expressed in m s\(^{-1}\).
Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when one or more of the following situations encountered: (1) <em>crop</em> string is empty; (2) <em>cultivar</em> string is empty if <em>crop</em> is cucumber or sugarbeet; (3) <em>RH</em> or <em>U2</em> is negative; (4) database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when humidity or wind velocity values can not be determined.</td>
</tr>
</tbody>
</table>

- **ExportKcDataIS**
  Overloaded.

1. Exports the initial stage crop coefficient data, with default settings.

**[C#]**

```csharp
public static string ExportKcDataIS();
```

Remarks: The following values are set as the default:

(a) $ET_0$ range - 0 to 10 mm day$^{-1}$.
(b) $ET_0$ increment - 0.1 mm day$^{-1}$.
(c) Wetting interval - 2, 4, 7, 10, 20 days.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

Example: The following example shows explicitly the format of an exported dataset.

```csharp
using System;
using System.IO;
using UGent.WLES;
using UGent.WLES.WPP;

class InitialStageKcData {
    public static void Main() {
        string data;
        data = CropCoefficient.ExportKcDataIS();
        StringReader sr = new StringReader(data);
        string line;
        while ( (line = sr.ReadLine()) != null )
            Console.WriteLine(line);
    }
}
```

The example generates the following results:

**Title:** Kc values for initial growth stage (IS) as related to $ET_0$ level and wetting frequency  
**X-axis:** $ET_0$ (mm day$^{-1}$)  
**Y-axis:** Kc (dimensionless)

<table>
<thead>
<tr>
<th>ETo, 2 days, 4 days, 7 days, 10 days, 20 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1, 1.1105, 1.0512, 0.8764, 0.7148, 0.5425</td>
</tr>
<tr>
<td>0.2, 1.093226108, 1.070867372, 0.97294912, 0.8793368, 0.630048, 0.4654824</td>
</tr>
</tbody>
</table>
2. Exports the initial stage crop coefficient data.

```csharp
[CS]
public static string ExportKcDataIS(double[] ETo_Range, double ETo_Step, double[] intervals);
```

**Parameters:**

- `ETo_Range` A 2-element `Double` array that contains the minimum (as 1\textsuperscript{st} element) and the maximum (as 2\textsuperscript{nd} element) `ET_0` values.
- `ETo_Step` A `Double` that represents the `ET_0` increment.
- `intervals` An `Double` array that represents the frequencies of irrigation or significant rain. The array element is expressed as the number of days between two adjacent irrigations or significant rains.

**Remarks:** The exported data is in comma-separated plain text format. The number of columns equals the length of the `intervals` array plus one. The 1\textsuperscript{st} data column is `ET_0`, followed by `k_c` values of each wetting interval. The data rows start at the minimum `ET_0` value and end at the maximum, with a particular incremental value given by the `ETo_Step` argument. The exported data can be imported into a third party application like Microsoft Excel to plot curves or to facilitate further analysis.

**Exceptions:**

- `ArgumentException` Thrown when the `ETo_Range` array does not contain two elements, or the 2\textsuperscript{nd} element is smaller than the 1\textsuperscript{st} in magnitude; and when the `ETo_Step` is smaller than or equal to zero.

5.2.5 Public Instance Constructors

- `• public CropCoefficient(WlesDataSet data);`

  Initializes a new `CropCoefficient` instance for a reference crop.

  **Parameters:**

  - `data` A `WlesDataSet` that represents the runtime dataset of the WLES application.

  **Exceptions:**

  - `WlesException` Thrown when the wetting interval during the IS stage is unknown.

- `• public CropCoefficient(WlesDataSet data, double kcIS, double kcMS, double kcAH);`

  Initializes a new `CropCoefficient` instance for a user-defined crop.

  **Parameters:**

  - `data` A `WlesDataSet` that represents the runtime dataset of the WLES application.
  - `kcIS` A `Double` that represents the crop coefficient value during the IS stage.
  - `kcMS` A `Double` that represents the crop coefficient value during the MS stage.
  - `kcAH` A `Double` that represents the crop coefficient value at the AH stage.

  **Exceptions:**

  - `WlesException` Thrown when the wetting interval during the IS stage is unknown.
• public CropCoefficient(CropCycle cycle, double interval, double EToIS, double RHMS, double RHAH, double U2MS, double U2AH);
  Initializes a new CropCoefficient instance for a reference crop.
  
  Parameters:
  
cycle The CropCycle.
interval A Double that represents the wetting interval (days) in the IS stage. Refer to the SetWettingInterval method for a more detailed definition of this term.
EToIS A Double that represents the average daily reference evapotranspiration (ET\(_o\)) value (mm day\(^{-1}\)) during the IS stage.
RHMS A Double that represents the average relative air humidity (%) during the MS stage.
RHAH A Double that represents the average relative air humidity (%) during the AH stage.
U2MS A Double that represents the average wind velocity at 2 m above ground (m s\(^{-1}\)) during the MS stage.
U2AH A Double that represents the average wind velocity at 2 m above ground (m s\(^{-1}\)) during the AH stage.

• public CropCoefficient(CropCycle cycle, double interval, double EToIS, double RHMS, double RHAH, double U2MS, double U2AH, double kcIS, double kcMS, double kcAH);
  Initializes a new CropCoefficient instance for a user-defined crop.
  
  Parameters:
  
cycle The CropCycle.
interval A Double that represents the wetting interval (days) in the IS stage. Refer to the SetWettingInterval method for a more detailed definition of this term.
EToIS A Double that represents the average daily reference evapotranspiration (ET\(_o\)) value (mm day\(^{-1}\)) during the IS stage.
RHMS A Double that represents the average relative air humidity (%) during the MS stage.
RHAH A Double that represents the average relative air humidity (%) during the AH stage.
U2MS A Double that represents the average wind velocity at 2 m above ground (m s\(^{-1}\)) during the MS stage.
U2AH A Double that represents the average wind velocity at 2 m above ground (m s\(^{-1}\)) during the AH stage.
kcIS A Double that represents the crop coefficient value during the IS stage.
kcMS A Double that represents the crop coefficient value during the MS stage.
kcAH A Double that represents the crop coefficient value at the AH stage.

5.2.6 Public Instance Properties

• Crop
  Gets the crop name String.
  [C#] public string Crop { get; }

• CropCycleInstance
  Gets the CropCycle instance on which this CropCoefficient instance is based.
  [C#] public CropCycle CropCycleInstance { get; }

• Cultivar
  Gets the cultivar name String.
  [C#] public string Cultivar { get; }
5.2 CropCoefficient Class

- **KcAH**
  Gets or sets the crop coefficient of the AH stage (dimensionless).
  ```csharp
  [C#] public double KcAH { get; set; }
  ```

- **Exceptions**:
  ```
  exception description
  ```
  ```
  WlesException Thrown when trying to set KcAH with a negative value.
  ```

- **KcCD**
  Gets a Double array that represents the daily crop coefficient values during the CD stage (dimensionless).
  ```csharp
  [C#] public double[] KcCD { get; }
  ```

- **KcCropCycle**
  Gets a Double array that represents the daily crop coefficient values during the whole CropCycle (dimensionless).
  ```csharp
  [C#] public double[] KcCropCycle { get; }
  ```

- **KcIS**
  Gets or sets the crop coefficient of the IS stage (dimensionless).
  ```csharp
  [C#] public double KcIS { get; set; }
  ```

- **KcLS**
  Gets the crop coefficient of the LS stage (dimensionless).
  ```csharp
  [C#] public double KcLS { get; }
  ```

- **KcMS**
  Gets or sets the crop coefficient of the MS stage (dimensionless).
  ```csharp
  [C#] public double KcMS { get; set; }
  ```

- **ReferenceEvapotranspirationIS**
  Gets the average reference evapotranspiration ($ET_o$) value of the IS stage (mm day$^{-1}$).
  ```csharp
  [C#] public double ReferenceEvapotranspirationIS { get; }
  ```

- **RelativeHumidityAH**
  Gets the average air humidity value of the AH stage (%).
  ```csharp
  [C#] public double RelativeHumidityAH { get; }
  ```

- **RelativeHumidityMS**
  Gets the average air humidity value of the MS stage (%).
  ```csharp
  [C#] public double RelativeHumidityMS { get; }
  ```

- **WettingIntervalIS**
  Gets the wetting interval value of the IS stage.
  ```csharp
  [C#] public double WettingIntervalIS { get; }
  ```

- **WindSpeedAH**
  Gets the average wind velocity (at 2 m above ground) value of the AH stage (m s$^{-1}$).
  ```csharp
  [C#] public double WindSpeedAH { get; }
  ```

- **WindSpeedMS**
  Gets the average wind velocity (at 2 m above ground) value of the MS stage (m s$^{-1}$).
  ```csharp
  [C#] public double WindSpeedMS { get; }
  ```

### 5.2.7 Public Instance Methods

- **ExportKcData**
  Exports the crop coefficient data of the whole CropCycle.
  ```csharp
  [C#] public string ExportData(bool yearRound);
  ```
  Parameters:
A Boolean that indicates whether the data is exported in the timeframe of a year (true) or of the crop cycle (false).

Examples: The following examples explicitly show the format of the exported data.

Example 1: This example shows the format of exported $k_c$ data of the whole crop cycle (130 days).

```csharp
using System;
using UGent.WLES;
using UGent.Wles.Data;
using UGent.WLES.WPP;

WlesDataSet data = new WlesDataSet();
data.SetLocationalData(...);
data.SetClimaticData(...);
data.SetCropData(...);
data.SetCropCycle(...);
data.SetWettingInterval(...);
CropCoefficient kc = new CropCoefficient(data);
string kcData;
kcData = kc.ExportData(false);
Console.WriteLine(kcData);
```

The example generates the following results:

Title: Crop Coefficient
X-axis: Crop cycle (day)
Y-axis: $k_c$ (dimensionless)
Crop: soybean
Cultivar:
day,$k_c$
1,0.63885 /* start of crop cycle; start of IS stage */
2,0.63885
...
20,0.63885 /* end of IS stage */
21,0.650888333333333 /* start of CD stage */
22,0.662926666666667 /* start of CD stage */
23,0.674965
24,0.687003333333333
25,0.699041666666667
...
45,0.92777
46,0.939808333333333
47,0.951846666666667
48,0.963885
49,0.975923333333333 /* end of CD stage */
50,1 /* start of MS stage */
51,1
...
90,1
91,1 /* end of MS stage */
92,0.98625 /* start of LS stage */
93,0.9725
94,0.95875
Example 2: This example shows the format of exported $k_c$ data in the timeframe of one year (360 days).

```csharp
/* same as example 1 */
kpData = kc.ExportData(true);
Console.WriteLine(kpData);
```

The example generates the following results:

Title: Crop Coefficient
X-axis: Crop cycle (day)
Y-axis: kc (dimensionless)
Crop: soybean
Cultivar:

```
day,kc
1,-1 /* start of the year */
2,-1
... 
90,-1 /* -1: out of crop cycle */
91,0.63885 /* start of crop cycle */
92,0.63885
... 
110,0.63885
111,0.63885
112,0.650883333333333
113,0.662926666666667
... 
138,0.963885
139,0.975923333333333
140,1
141,1
... 
180,1
181,1
182,0.98625
183,0.9725
... 
218,0.49125
219,0.4775
220,0.45 /* end of crop cycle */
221,-1 /* -1: out of crop cycle */
222,-1
... 
359,-1
360,-1 /* end of the year */
```
• GetKcDailyArray
  Gets a Double array that represents the daily CropCoefficient values of the whole CropCycle within the timeframe of one year.
  
  [C#] public double[] GetKcDailyArray();

  Return Value: A Double array that represents the crop coefficient values for the whole CropCycle; Array length varies, depending on the calendar associated with the CropCycleInstance: 360 for WlesCalendar and 365 for GregorianCalendar.

  Remarks:
  1. Length of array - The array has the same length as that of a WlesCalendar year. The array elements are organized in the same order as the days in a year. Please note that the array is 0 based, although the year is 1 based.
  2. Exceptional value - -1 is used as an exceptional value for those days that are outside the CropCycle. Normal k_c values are positive floating point numbers.

• GetMaximumEvapotranspirationCropCycle
  Gets a Double array that represents the daily maximum (crop) evapotranspiration (ET_c or ET_m) during this crop cycle.
  
  [C#] public double[] GetMaximumEvapotranspirationCropCycle(double[] EToCropCycle);

  Exceptions:
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when length of EToCropCycle not equal to that of this crop cycle, or the array is a null reference.</td>
</tr>
</tbody>
</table>

• GetMaximumEvapotranspirationDailyArray
  Gets a Double array that represents the daily maximum (crop) evapotranspiration (ET_c or ET_m) during this crop cycle within the timeframe of one year.
  
  [C#] public double[] GetMaximumEvapotranspirationDailyArray(double[] EToCropCycle);

  Remarks:
  1. Length of array - The array has the same length as that of a WlesCalendar year. The array elements are organized in the same order as the days in a year. Please note that the array is 0 based, although the year is 1 based.
  2. Exceptional value - -1 is used as an exceptional value for those days that are outside the CropCycle. Normal k_c values are positive floating point numbers.

  Exceptions:
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when length of EToCropCycle not equal to that of this crop cycle, or the array is a null reference.</td>
</tr>
</tbody>
</table>

5.3 CropCoefficientUI Class

5.3.1 Summary

Represents the CropCoefficient determination and reporting interface of the WLES Web UI.

5.3.2 Declaration

  [C#] public class CropCoefficientUI : Page

5.3.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:
A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

### 5.3.4 Internal Static Methods

- **CalculateETmCropCycle**
  Calculates the daily maximum crop evapotranspiration ($ET_m$ or $ET_c$, mm day$^{-1}$) during the CropCycle.
  ```
  [C#] internal static double[] CalculateETmCropCycle(Page page);
  ```

- **CalculateETmDailyArray**
  Calculates the year-round daily maximum crop evapotranspiration ($ET_m$ or $ET_c$, mm day$^{-1}$).
  ```
  [C#] internal static double[] CalculateETmDailyArray(Page page);
  ```

- **FormKcValuesString**
  Formulates a String that represents the critical $k_c$ values within the CropCycle.
  ```
  [C#] internal static string FormKcValuesString(Page page);
  ```

  **Return Value:** A String of the following format:

  IS: x.xx; MS: y.yy; AH: z.zz

  where $x.xx$, $y.yy$ and $z.zz$ are $k_c$ values of the initial, mid-season and at-harvest stages, respectively.

- **RetrieveKcValues**
  Retrieves the critical CropCoefficient ($k_c$) values for the IS, MS and AH stages that are associated with a particular Web Form page.
  ```
  [C#] internal static double[] RetrieveKcValues(Page page);
  ```

- **SetupCropCoefficient**
  Sets up a CropCoefficient instance using crop and climatic data that are associated with the current Web Form page.
  ```
  [C#] internal static CropCoefficient SetupCropCoefficient(Page page);
  ```

  **Remarks:** The CropCoefficient instance returned was initialized in one of the two ways:

  1. Re-evaluation - The $k_c$ values for the IS, MS and AH stages are retrieved from the previous evaluation session and used explicitly to initialize a new CropCoefficient instance;

  2. New evaluation - The $k_c$ values for the IS, MS and AH stages are determined with the crop being attempted as a reference crop. If failed, the $k_c$ values for the whole CropCycle will have the default value of 0.

### 5.4 DepletionFractionUI Class

#### 5.4.1 Summary

Represents the soil water depletion fraction ($p$) determination and reporting interface of the WLES Web UI.

#### 5.4.2 Declaration

```
[C#] public class DepletionFractionUI : Page
```
5.4.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

depletion.aspx?rid=U0X8N5F6J0M6&timeout=332325051506&foo=bar

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

5.4.4 Internal Static Methods

- **CalculateSoilWaterDepletionFractionCropCycle**
  Calculates the daily soil water depletion fraction values during the CropCycle using either the crop-specific or the general approach based on user preference associated with a particular Web Form page.

[C#] internal static double[] CalculateSoilWaterDepletionFractionCropCycle(Page page);

**Return Value:** A Double array that represents the daily soil water depletion fraction values during the CropCycle. Array length equals the length of the crop cycle.

- **FormDepletionString**
  Formulates a String representation of the approach used and the WaterBalance, SoilWaterDepletionGroup identified in evaluation of the soil water depletion fraction (p) which is associated with a particular Web Form page.

[C#] internal static string FormDepletionString(Page page);

**Return Value:** A String of this format: “Group-specific approach: Group IV” or “General approach: no groups distinguished”.

- **RetrieveDepletionGroup**
  Retrieves the soil water depletion group ID of the crop associated with a particular Web Form.

[C#] internal static SoilWaterDepletionGroup RetrieveDepletionGroup(Page page);

**Return Value:** A WaterBalance.SoilWaterDepletionGroup member: 1-4 as group I-IV; 0 if groups are not distinguished (general approach); -1 if the group has not been defined yet.

5.5 EffectiveRainfall Class

5.5.1 Summary

Represents the effective rainfall.

5.5.2 Declaration

[C#] public class EffectiveRainfall

5.5.3 Remarks

Effective rainfall is defined as the portion of total annual or seasonal rainfall which is useful directly and/or indirectly for crop production at the site where it falls but without pumping. It therefore includes (Allen et al., 1998; Dastane, 1978; Doorenbos and Pruitt, 1977):

1. water intercepted by living or dry vegetation;
2. water lost by evaporation from the soil surface;
3. the precipitation lost by evapotranspiration during growth, and
4. that fraction which contributes to leaching and percolation.

5.5.4 Examples

Example 1: This example calculates the decadely effective rainfall during the crop cycle of tropical maize in Kigali (Rwanda) using the USBR approach.

```java
// year-round monthly rainfall in mm
double[] rainfall = {68.2, 110.6, 113.9, 178.5, 91.8, 22.5, 9.7, 31.1, 74.1, 94.7, 132, 88.6};
// convert it into decadely data: array of 36 doubles
rainfall = ClimaticData.CalculateDecadelyFromMonthly(rainfall, ClimaticData.ClimaticParamSet.RAINFALL);

// the crop cycle: starts on Jul. 10, ends on the 300th day of the year
RomanDate date1, date2;
date1 = new RomanDate(RomanDate.MonthOfYear.JUL, 10);
date2 = new RomanDate(300);
// the number of decades since year beginning
int start, end;
start = date1.CountDecades();
end = date2.CountDecades();
// rainfall during the crop cycle, decadely data, mm
double[] rainCC = new double[end - start + 1];
for (int i=start; i<end; i++)
    rainCC[i] = rainfall[i-1];
// calculate using USBR approach
double[] peff = EffectiveRainfall.UseUSBR(rainCC, WlesDataSet.TemporalResolution.DECADELY);
```

Listed below are the total rainfall and the effective rainfall during the crop cycle in mm.

<table>
<thead>
<tr>
<th>Decade no.</th>
<th>Total rainfall</th>
<th>Effective rainfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.97</td>
<td>2.92</td>
</tr>
<tr>
<td>2</td>
<td>2.81</td>
<td>2.77</td>
</tr>
<tr>
<td>3</td>
<td>3.92</td>
<td>3.85</td>
</tr>
<tr>
<td>4</td>
<td>6.92</td>
<td>6.69</td>
</tr>
<tr>
<td>5</td>
<td>10.10</td>
<td>9.61</td>
</tr>
<tr>
<td>6</td>
<td>14.08</td>
<td>13.13</td>
</tr>
<tr>
<td>7</td>
<td>21.03</td>
<td>18.91</td>
</tr>
<tr>
<td>8</td>
<td>24.98</td>
<td>21.98</td>
</tr>
<tr>
<td>9</td>
<td>28.10</td>
<td>24.31</td>
</tr>
<tr>
<td>10</td>
<td>28.45</td>
<td>24.57</td>
</tr>
<tr>
<td>11</td>
<td>31.36</td>
<td>26.64</td>
</tr>
<tr>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Example 2: This example calculates the daily effective rainfall during the crop cycle of tropical winter wheat in Namaacha (Mozambique) using the USDA_SCS approach.

```java
// prepring data
double latitude, longitude, elevation;
string place;
double[] tmin, tmax, RH, n, rain, wind;
SoilHorizon[] horizons;
int rainfall_interval;
```
double limiting_layer = -1;
place = "Namaacha (Mozambique)";
tmax = new double[] {29.1, 28.9, 28.2, 26.8, 25.3, 23.6, 23.4, 24.6, 26.1, 27.2, 27.6, 29.0};
tmin = new double[] {18.4, 18.3, 17.9, 16.4, 14.4, 12.4, 12.4, 13.0, 14.0, 15.4, 16.4, 17.4};
RH = new double[] {80, 81, 82, 79, 74, 70, 69, 71, 71, 74, 77, 79};
wind = new double[] {2.2, 1.8, 1.9, 1.9, 2.2, 2.2, 2.2, 2.3, 2.2, 2.0};
n = new double[] {5.1, 5.7, 5.7, 5.9, 7.3, 7.4, 8.0, 7.4, 6.4, 4.5, 4.5, 5.0};
rain = new double[] {151, 117, 117, 71, 29, 24, 22, 22, 43, 80, 115, 105};
rainfall_interval = 7;
latitude = 25 + 29.0 / 60.0;
longitude = 32 + 1.0 / 60.0;
elevation = 523;
horizons = new SoilHorizon[4];
string[] id = new string[] {"Ah", "Bw1", "Bw2", "C",};
for (int i = 0; i < id.Length; i++)
    horizons[i] = new SoilHorizon(id[i]);
horizons[0].Depth = new double[] {0, 20};
horizons[1].Depth = new double[] {20, 55};
horizons[2].Depth = new double[] {55, 90};
horizons[3].Depth = new double[] {90, 160};
horizons[0].ParticleDistribution = new double[] {16, 4, 80};
horizons[1].ParticleDistribution = new double[] {26, 3, 71};
horizons[2].ParticleDistribution = new double[] {43, 13, 44};
horizons[3].ParticleDistribution = new double[] {33, 7, 60};
horizons[0].Gravel = 8;
horizons[1].Gravel = 0;
horizons[2].Gravel = 13;
horizons[3].Gravel = 2;
horizons[0].BulkDensity = 1.21;
horizons[1].BulkDensity = 1.21;
horizons[2].BulkDensity = 1.18;
horizons[3].BulkDensity = 1.65;
horizons[0].Structure = SoilData.SoilStructure.Blocky;
limiting_layer = 0.90;
// declare a WlesDataSet instance
WlesDataSet data = new WlesDataSet();
data.SetLocationalData(place, latitude, longitude, elevation);
data.SetClimaticData(tmax, tmin, RH, wind, n, rain);
// the crop cycle
RomanDate cc1, cc2;
cc1 = new RomanDate(11, 1);
cc2 = new RomanDate(6, 30);
string crop = "winter wheat";
cultivar = "tropical";
data.SetCropData(crop, cultivar, CropData.GetLeafAreaIndex(crop),
    CropData.GetHarvestIndex(crop, cultivar),
    CropData.GetOptimalRootingDepth(crop));
data.SetCropCycle(cc1, 30, 70, 100, 40);
data.SetWettingInterval(rainfall_interval);
data.SetSoilData(horizons, limiting_layer);
s = "";
5.5 EffectiveRainfall Class

CropCycle cc = new CropCycle(data);
s += string.Format("Crop cycle: {0}<br>", cc);
double [] prec = CropCycle.TransformData(data.GetRainfallDailyArray(), cc);
double [] peff = EffectiveRainfall.UseUSDA_SCS(data);

// print rainfall crop cycle, mm
s += "Rainfall crop cycle, mm"
for (int i=0; i<prec.Length; i++) {
    s += string.Format("{0} ",Math.Round(prec[i],2));
    if ((i+1)%10==0)
        s += "<br>";
}s += "<br>";

// effective rainfall crop cycle, mm
s += "Effective rainfall crop cycle, mm"
for (int i=0; i<peff.Length; i++) {
    s += string.Format("{0} ",Math.Round(peff[i],2));
    if ((i+1)%10 == 0)
        s += "<br>";
}s += "<br>";

The result of the example is stored in String s, which has the following content:

Crop cycle:
Crop cycle of winter wheat (tropical) in Namaacha (Mozambique):
    Nov 01 (WLES Calendar) - Jun 30 (WLES Calendar)
Length of crop cycle: 240 days
Crop development stages:
    IS: Nov 01 (WLES Calendar) - Nov 30 (WLES Calendar), 30 days
    CD: Dec 01 (WLES Calendar) - Feb 10 (WLES Calendar), 70 days
    MS: Feb 11 (WLES Calendar) - May 20 (WLES Calendar), 100 days
    LS: May 21 (WLES Calendar) - Jun 30 (WLES Calendar), 40 days
Rainfall year, mm
    3.89 3.89 3.89 3.89 3.89 3.89 3.89 3.89 3.89 3.89
    3.94 3.94 3.94 3.94 3.94 3.94 3.94 3.94 3.94 3.94
    3.33 3.33 3.33 3.33 3.33 3.33 3.33 3.33 3.33 3.33
    3.43 3.43 3.43 3.43 3.43 3.43 3.43 3.43 3.43 3.43
    3.73 3.73 3.73 3.73 3.73 3.73 3.73 3.73 3.73 3.73
    5.13 5.13 5.13 5.13 5.13 5.13 5.13 5.13 5.13 5.13
    5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05 5.05
    4.11 4.11 4.11 4.11 4.11 4.11 4.11 4.11 4.11 4.11
    3.86 3.86 3.86 3.86 3.86 3.86 3.86 3.86 3.86 3.86
    3.73 3.73 3.73 3.73 3.73 3.73 3.73 3.73 3.73 3.73
    3.96 3.96 3.96 3.96 3.96 3.96 3.96 3.96 3.96 3.96
    2.86 2.86 2.86 2.86 2.86 2.86 2.86 2.86 2.86 2.86
    2.36 2.36 2.36 2.36 2.36 2.36 2.36 2.36 2.36 2.36
    1.88 1.88 1.88 1.88 1.88 1.88 1.88 1.88 1.88 1.88
    1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25
    0.92 0.92 0.92 0.92 0.92 0.92 0.92 0.92 0.92 0.92
Effective rainfall year, mm
1.47 1.95 2.31 2.41 2.47 2.52 2.53 2.53 2.53 2.53
2.69 2.69 2.69 2.69 2.69 2.69 2.69 2.69 2.69 2.69
2.74 2.74 2.74 2.74 2.74 2.74 2.74 2.74 2.74 2.74
2.33 2.32 2.31 2.3 2.29 2.28 2.27 2.27 2.27 2.27
2.33 2.32 2.31 2.31 2.31 2.31 2.31 2.31 2.31 2.31
2.49 2.49 2.49 2.49 2.49 2.49 2.49 2.49 2.49 2.49
3.31 3.31 3.31 3.31 3.31 3.31 3.31 3.31 3.31 3.31
3.28 3.28 3.29 3.29 3.29 3.29 3.29 3.29 3.29 3.29
2.82 2.83 2.83 2.84 2.84 2.85 2.85 2.85 2.85 2.85
2.71 2.71 2.71 2.71 2.71 2.71 2.71 2.71 2.71 2.71
2.63 2.63 2.63 2.63 2.63 2.63 2.63 2.63 2.63 2.63
2.94 2.94 2.94 2.94 2.94 2.94 2.94 2.94 2.94 2.94
2.83 2.83 2.83 2.83 2.83 2.83 2.83 2.83 2.83 2.83
2.61 2.61 2.61 2.61 2.61 2.61 2.61 2.61 2.61 2.61
2.16 2.16 2.16 2.16 2.16 2.16 2.16 2.16 2.16 2.16
1.81 1.81 1.81 1.81 1.81 1.81 1.81 1.81 1.81 1.81
1.47 1.47 1.47 1.47 1.47 1.47 1.47 1.47 1.47 1.47
1.02 1.02 1.02 1.02 1.02 1.02 1.02 1.02 1.02 1.02
0.76 0.76 0.76 0.76 0.76 0.76 0.76 0.76 0.76 0.76
0.61 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6
0.66 0.66 0.66 0.65 0.65 0.65 0.64 0.64 0.64 0.63
0.6 0.6 0.59 0.59 0.59 0.59 0.59 0.59 0.59 0.59
0.55 0.54 0.54 0.54 0.54 0.53 0.53 0.53 0.53 0.53

5.5.5 Public Static Methods

- **FAO_AGLW**
  Calculates the monthly effective rainfall using the FAO/AGLW method.

  ```csharp
  public static double FAO_AGLW(double rain);
  ```

  **Remarks:** FAO/AGLW developed an empirical formula based on analysis carried out for
different climatic data to determine the dependable effective rainfall, i.e., the dependable
rainfall at 80% probability corrected for assumed losses due to runoff and percolation (Smith,
1988).

- **FixedPercentage**
  Calculates the effective rainfall using the fixed percentage method.

  ```csharp
  public static double FixedPercentage(double rain, double pc);
  ```

  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the value of argument pc is not in range [0, 100].</td>
</tr>
</tbody>
</table>

- **InterpolateUSDAFactor**
  Interpolates the correction factor used in the USDA Soil Conservation Service method.

  ```csharp
  public static double InterpolateUSDAFactor(double pSaD);
  ```

  **Parameters:**

  | pSaD | Easily available soil water (see: Remarks section of the WaterBalance class) at rooting
depth, mm. |
5.5 EffectiveRainfall Class

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the database query returns nothing.</td>
</tr>
</tbody>
</table>

- **USBR**
  Calculates the monthly effective rainfall using the U.S. Bureau of Reclamation method.
  ```csharp
  public static double USBR(double rain);  
  ```

- **USDA_SCS**
  Calculates the monthly effective rainfall using the USDA Soil Conservation Service method.
  ```csharp
  public static double USDA_SCS(double rain, double ETc, double pSaD);  
  ```

**Parameters:**

- **rain** Monthly total rainfall, expressed in mm.
- **ETc** Monthly maximum (crop) evapotranspiration (see: Remarks section of the WaterBalance class and calculation), expressed in mm: \( ET_c = ET_m = kc \cdot ET_o \).
- **pSaD** Easily available soil water (see: Remarks section of the WaterBalance class) at rooting depth, mm.

**Example:** The following example calculates the monthly effective rainfall using the USDA SCS method.

**Known:**

- A crop with a rooting depth of 0.80 m;
- Soil moisture content at field capacity: 180 mm m\(^{-1}\);
- Soil moisture content at wilting point: 90 mm m\(^{-1}\);
- Fraction of easily available soil water: 2/3;
- Rainfall: 150 mm;
- Crop evapotranspiration: 107.5 mm.

```csharp
using System;
using UGent.WLES;
using UGent.WLES.WPP;

public class ERExample {
    public static void Main() {
        // field capacity and wilting point, mm/m
        double fc, wp;
        fc = 180;
        wp = 90;
        // soil water storage, mm/m
        double storage;
        storage = fc - wp;
        // root depth, m
        double depth;
        depth = 0.80;
        // fraction p, dimensionless
        double p;
        p = 2.0 / 3.0;
        // easily available soil water at root zone, mm
        double pSaD;
        pSaD = p * storage * depth;
        // total rainfall and ETa, mm
        double rain, ETa;
```
// effective rainfall
double er;
er = EffectiveRainfall.USDA_SCS(rain, ETc, pSaD);
er = Math.Round(er, 0);

Console.WriteLine("Effective rainfall: {0} mm.", er);

The example generates the following results:

Effective rainfall: 91 mm.

• UseFAO_AGLW
  Computes the effective rainfall (mm) for the same time interval as that of rainfall, using the
  FAO_AGLW method.
[C#] public static double[] UseFAO_AGLW(double[] rainfall, TemporalResolution dataRes);
Parameters:
  rainfall A Double array that represents the rainfall (mm) data.
  dataRes A WlesDataSet.TemporalResolution member that represents the data resolution of
  rainfall.

Return Value: A Double array that represents the effective rainfall (mm) for the same
  time interval as that of rainfall.

• UseFixedPercentage
  Computes the effective rainfall (mm) for the same time interval as that of rainfall, using the
  FixedPercentage method.
[C#] public static double[] UseFixedPercentage(double[] rainfall, double pc);

• UserMethodPlugin
  Calculates the effective rainfall using a client-defined method.
[C#] public static double UserMethodPlugin(UserDefinedMethod callback, double rain);
Parameters:
  callback A EffectiveRainfall.UserDefinedMethod instance that points to the client-defined
  method (in the client’s code).
  rain A Double that represents the rainfall.

Return Value: A Double that represents the effective rainfall that was estimated using the
  callback.

Example: The following example shows how to define a “third-party” method to estimate
  the effective rainfall from rainfall data, and plug this method into the EffectiveRainfall
  class so that the results of different methods can be compared.

using System;
using UGent.WLES;
using UGent.WLES.WPP;

public class UserEstimateER {
  public static void Main() {
    double rain = 150; /* mm in July */
/* first calculate the ER using my method */
EffectiveRainfall.UserDefinedMethod mine =
    new EffectiveRainfall.UserDefinedMethod(MyMethod);
double myResult;
myResult = EffectiveRainfall.UserMethodPlugin(mine, rain);
/* calculate using already defined methods */
double fao, usbr;
fao = EffectiveRainfall.FAO_AGLW(rain);
usbr = EffectiveRainfall.USBR(rainfall);
/* print the results to compare */
Console.WriteLine("Effective rainfall estimations:
  150 mm total rain in July");
Console.WriteLine(" My estimation = {0,3}"; myResult);
Console.WriteLine(" FAO estimation = {0,3}"; fao);
Console.WriteLine(" USBR estimation = {0,3}"; usbr);

// my method to estimate ER, which is
// adapted to local conditions
public static double MyMethod(double rain) {
    // in my study area, the historical data
    // reveal that 1/3 rainfall in July was
    // lost. in other words, effective rainfall
    // is 2/3 of the total rainfall
    return rain * 2 / 3;
}

The following results will be generated:

Effective rainfall estimations: 150 mm total rain in July
  My estimation = 100
  FAO estimation = 95
  USBR estimation = 114

- UseUSBR
  Computes the effective rainfall (mm) for the same time interval as that of rainfall, using the
  USBR method.

[C#]
public static double[] UseUSBR(double[] rainfall, TemporalResolution dataRes);

- UseUSDA_SCS
  Overloaded.

1. Computes the effective rainfall (mm) for the same time interval as that of rainfall, using
the USDA_SCS method.

[C#]
public static double[] UseUSDA_SCS(double[] rainfall, double[] ETc, double[]
  pSaD, TemporalResolution dataRes);

Parameters:

- rainfall A Double array that represents the rainfall (mm) data.
- ETc Maximum (crop) evapotranspiration (see: Remarks section of the WaterBalance
  class and calculation), expressed in mm: \( ET_c = ET_m = k_c \cdot ET_o \).
- pSaD Easily available soil water (see: Remarks section of the WaterBalance class) at
  rooting depth, mm.
a WlesDataSet.TemporalResolution member that represents the data resolution of rainfall, ETc and pSaD.

**Note:** The value of this temporal resolution is dependent from the length of any data array. For instance, a 36-element rainfall array does not necessarily mean the rainfall data of 36 decades of an entire year, unless dataRes equals DECADELY; On the other hand, if dataRes equals DAILY, then the same data array contains rainfalls of 36 days!

**Return Value:** A Double array that represents the effective rainfall (mm) for the same time interval as that of rainfall.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when data arrays rainfall, ETc and pSaD not having the same length.</td>
</tr>
</tbody>
</table>

2. Computes the daily effective rainfall (mm) for the crop cycle that data represents, using the USDA_SCS method.

```csharp
[C#] public static double[] UseUSDA_SCS(WlesDataSet data);
```

**Parameters:**

data A WlesDataSet that was properly initialized.

### 5.6 EffectiveRainfallUI Class

#### 5.6.1 Summary

Represents the EffectiveRainfall determination and reporting interface of the WLES Web UI.

#### 5.6.2 Declaration

```csharp
[C#] public class EffectiveRainfallUI : Page
```

#### 5.6.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

```
er.aspx?rid=U0X8NOF5MOJ6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

#### 5.6.4 Internal Static Methods

- GetEffectiveRainfallCropCycle
  
  Gets the DAILY EffectiveRainfall values during the CropCycle, mm day\(^{-1}\).

  ```csharp
  [C#] internal static double[] GetEffectiveRainfallCropCycle(Page page);
  ```

- GetEffectiveRainfallDailyArray
  
  Gets the DAILY EffectiveRainfall values during a calendar year, mm day\(^{-1}\).

  ```csharp
  [C#] internal static double[] GetEffectiveRainfallDailyArray(Page page);
  ```

  **Return Value:** A 360- or 365-element Double array that represents the daily EffectiveRainfall values during a RomanDate.CalendarType year, mm day\(^{-1}\). The length of the array depends on the effective calendar.

  **Remarks:**

  1. The length of the returned array depends on the type of the effective calendar: 360 if WlesCalendar, 365 if GregorianCalendar.
2. The content the returned array depends on the method used to evaluate the EffectiveRainfall values. If USDA_SCS is used, the daily effective rainfall values only during the CropCycle are obtained. The rest slots of the array are padded with value -1. For all other methods, the daily effective rainfall values for the whole year are obtained.

- RetrieveEffectiveRainfall
  Retrieves the year- or crop-cycle-round EffectiveRainfall values (mm) associated with a particular Web Form page, depending on the method used.

  ```csharp
  internal static double[] RetrieveEffectiveRainfall(Page page);
  ```

  Return Value: A multi-element Double array; a null reference if EffectiveRainfall has not yet been evaluated. See Remarks for the length of the array.

  Remarks: The length of the returned Double array is evaluation method dependent:

<table>
<thead>
<tr>
<th>method</th>
<th>length of returned array</th>
</tr>
</thead>
<tbody>
<tr>
<td>FixedPercentage</td>
<td>Same as the rainfall data, specified by the RetrieveDataResolution method.</td>
</tr>
<tr>
<td>FAO_AGLW</td>
<td>Same as the rainfall data, specified by the RetrieveDataResolution method.</td>
</tr>
<tr>
<td>USBR</td>
<td>Same as the rainfall data, specified by the RetrieveDataResolution method.</td>
</tr>
<tr>
<td>USDA_SCS</td>
<td>Same as the length of the crop cycle (a daily array covering the entire crop cycle).</td>
</tr>
<tr>
<td>User</td>
<td>User defined: either 12 (MONTHLY), 36 (DECADELY), 360 (DAILY with WlesCalendar) or 365 (DAILY with GregorianCalendar).</td>
</tr>
</tbody>
</table>

- RetrieveEffectiveRainfallMethod
  Retrieves the method used to estimate the EffectiveRainfall in the evaluation associated with a particular Web Form page.

  ```csharp
  internal static Approach RetrieveEffectiveRainfallMethod(Page page);
  ```

  Return Value: An EffectiveRainfall.Approach member; -1 if unknown.

### 5.7 PenmanMonteith Class

#### 5.7.1 Summary

Represents the estimation of the reference evapotranspiration using the Penman-Monteith approach.

#### 5.7.2 Declaration

```csharp
public class PenmanMonteith
```

#### 5.7.3 Remarks

1. The reference evapotranspiration ($ET_o$) is the potential evapotranspiration from a grass reference surface, i.e., an extensive green grass cover of uniform height (12 cm), completely shading the ground, actively growing, and experiencing no shortage of water.

2. There are many procedures have been invented to estimate the $ET_o$ since Penman (1948) published his formula. Recent comparative studies (Smith, 1991) recommend the Penman-Monteith (Monteith, 1965) approach as the best-performing combination equation.

#### 5.7.4 Public Static Fields

- ALPHA
  The canopy reflection coefficient, or albedo, of the hypothetical reference crop.

  ```csharp
  public const double ALPHA = 0.23;
  ```
5.7.5 Public Static Methods

- CalculateETo
  Overloaded.

1. Calculates the reference evapotranspiration using the Penman-Monteith approach (mm day$^{-1}$).

[C#] public static double CalculateETo(double $t_{\text{max}}$, double $t_{\text{min}}$, double $RH$, double $U_2$, double $z$, double $f$, double $Ra$);

Parameters:

$t_{\text{max}}$ Maximum daily temperature($^\circ$C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature:

$$T_{\text{max}} = 2 \cdot T_{\text{mean}} - T_{\text{min}}$$  \hspace{1cm} (5.2)

$t_{\text{min}}$ Minimum daily temperature($^\circ$C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature:

$$T_{\text{min}} = 2 \cdot T_{\text{mean}} - T_{\text{max}}$$  \hspace{1cm} (5.3)

$RH$ Relative air humidity (%). Note: Value 70% is written as 70.

$U_2$ Wind velocity at 2 meters above ground (m s$^{-1}$).

$z$ Altitude of the considered area (m).

$f$ Ratio of sunshine duration per day (hr) over daylength (hr), or clear-sky ratio. Same as the $n/N$ notation.

$Ra$ The extra terrestrial radiation (mm day$^{-1}$), which is a function of the latitude and the period in the year. Values can be found in literatures (Sys et al., 1991, Table 43, p195). Values in MJ m$^{-2}$ day$^{-1}$ must be converted to mm day$^{-1}$ by multiplying with 1/2.45.

Return Value: A Double as the reference evapotranspiration in mm day$^{-1}$.

2. Calculates the reference evapotranspiration using the Penman-Monteith approach (mm day$^{-1}$).

[C#] public static double CalculateETo(double $t_{\text{mean}}$, double $t_{\text{max}}$, double $t_{\text{min}}$, double $RH$, double $U_2$, double $z$, double $f$, double $Ra$);

Parameters:

$t_{\text{mean}}$ Mean air temperature ($^\circ$C). If not readily available, can be obtained from the maximum and the minimum daily temperature:

$$T_{\text{mean}} = (T_{\text{max}} + T_{\text{min}})/2$$  \hspace{1cm} (5.4)

$t_{\text{max}}$ Maximum daily temperature($^\circ$C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).

$t_{\text{min}}$ Minimum daily temperature($^\circ$C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).

$RH$ Relative air humidity (%). Note: Value 70% is written as 70.

$U_2$ Wind velocity at 2 meters above ground (m s$^{-1}$).

$z$ Altitude of the considered area (m).

$f$ Ratio of sunshine duration per day (hr) over daylength (hr), or clear-sky ratio. Same as the $n/N$ notation.

$Ra$ The extra terrestrial radiation (mm day$^{-1}$), which is a function of the latitude and the period in the year. Values can be found in literatures (Sys et al., 1991, Table 43, p195). Values in MJ m$^{-2}$ day$^{-1}$ must be converted to mm day$^{-1}$ by multiplying with 1/2.45.
5.7 PenmanMonteith Class

Return Value: A Double as the reference evapotranspiration in mm day\(^{-1}\).

3. Calculates the year-round daily, decadally or monthly reference evapotranspiration \((ET_o, \text{ mm})\) for a particular location using the Penman-Monteith approach (Monteith, 1965).

```csharp
public static double[] CalculateETo(double latitude, double elevation, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine);
```

Parameters:

latitude Latitude of the considered area. Northern Hemisphere has positive values, while Southern Hemisphere has negative. Values are expressed in decimal degrees.

elevation Elevation of the considered area. Values are expressed in meters above sea level.

tmax Maximum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).

tmin Minimum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).

RH Relative air humidity (%). Note: Value 70% is written as 70.

U2 Wind velocity at 2 meters above ground (m s\(^{-1}\)).

sunshine A Double array that represents the sunshine hours per day (hr).

Return Value: A Double as the reference evapotranspiration in mm day\(^{-1}\), mm decade\(^{-1}\) or mm month\(^{-1}\).

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument arrays ((tmax, tmin, RH, U2, \text{ and sunshine})) do not have same length or the length does not equal to 12, 36, 360.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

4. Calculates the year-round daily, decadely or monthly reference evapotranspiration \((ET_o, \text{ mm})\) for a particular location using the Penman-Monteith approach (Monteith, 1965).

```csharp
public static double[] CalculateETo(double latitude, double elevation, double[] tmean, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine);
```

Parameters:

latitude Latitude of the considered area. Northern Hemisphere has positive values, while Southern Hemisphere has negative. Values are expressed in decimal degrees.

elevation Elevation of the considered area. Values are expressed in meters above sea level.

tmean Mean air temperature (°C). If not readily available, can be obtained from the maximum and the minimum daily temperature (see Equation 5.4).

tmax Maximum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).

tmin Minimum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).

RH Relative air humidity (%). Note: Value 70% is written as 70.

U2 Wind velocity at 2 meters above ground (m s\(^{-1}\)).

sunshine A Double array that represents the sunshine hours per day (hr).

Return Value: A Double as the reference evapotranspiration in mm day\(^{-1}\), mm decade\(^{-1}\) or mm month\(^{-1}\).

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument arrays ((tmean, tmax, tmin, RH, U2, \text{ and sunshine})) do not have same length or the length does not equal to 12, 36, 360.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>
5. Calculates the year-round daily, decadelly or monthly reference evapotranspiration ($ET_0$, mm) for a particular location using the Penman-Monteith approach (Monteith, 1965).

[C#] public static double[] CalculateETo(double latitude, double elevation, double[] tmean, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine, CalendarType calendar);

Parameters:

latitude Latitude of the considered area. Northern Hemisphere has positive values, while Southern Hemisphere has negative. Values are expressed in decimal degrees.
elevation Elevation of the considered area. Values are expressed in meters above sea level.
tmean Mean air temperature (°C). If not readily available, can be obtained from the maximum and the minimum daily temperature (see Equation 5.4).
tmax Maximum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).
tmin Minimum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).
RH Relative air humidity (%). Note: Value 70% is written as 70.
U2 Wind velocity at 2 meters above ground (m s$^{-1}$).
sunshine A Double array that represents the sunshine hours per day (hr).
calendar The RomanDate.CalendarType of the input and output data.

Return Value: A Double as the reference evapotranspiration in mm day$^{-1}$, mm decade$^{-1}$ or mm month$^{-1}$.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the argument arrays (tmean, tmax, tmin, RH, U2, and sunshine) do not have same length or the length does not equal to 12, 36, 360 (or 365, depending on calendar).</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

6. Calculates the year-round daily, decadelly or monthly reference evapotranspiration ($ET_0$, mm) for a particular location using the Penman-Monteith approach (Monteith, 1965).

[C#] public static double[] CalculateETo(double latitude, double elevation, double[] tmax, double[] tmin, double[] RH, double[] U2, double[] sunshine, CalendarType calendar);

Parameters:

latitude Latitude of the considered area. Northern Hemisphere has positive values, while Southern Hemisphere has negative. Values are expressed in decimal degrees.
elevation Elevation of the considered area. Values are expressed in meters above sea level.
tmax Maximum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).
tmin Minimum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).
RH Relative air humidity (%). Note: Value 70% is written as 70.
U2 Wind velocity at 2 meters above ground (m s$^{-1}$).
sunshine A Double array that represents the sunshine hours per day (hr).
calendar The RomanDate.CalendarType of the input and output data.

Return Value: A Double as the reference evapotranspiration in mm day$^{-1}$, mm decade$^{-1}$ or mm month$^{-1}$.

Exceptions:
5.7 PenmanMonteith Class

### Exception Description

- **ArgumentException**: Thrown when the argument arrays (`tmax`, `tmin`, `RH`, `U2`, and `sunshine`) do not have the same length or the length does not equal to 12, 36, 360 (or 365, depending on calendar).
- **WlesDatabaseException**: Thrown when the underlying database error occurs.

7. Calculates the reference evapotranspiration using the Penman-Monteith approach (mm day\(^{-1}\)).

```csharp
public static double CalculateETo(double latitude, HemisphereOfEarth hemisphere, MonthOfYear month, double tmax, double tmin, double RH, double U2, double z, double n);
```

**Parameters**:

- `latitude`: Latitude of the study area. Values are expressed in decimal degrees. Signs (+/-) preceding the numeric value will be ignored. Northern or Southern Hemispheres are specified by the `hemisphere` parameter.
- `hemisphere`: A `ClimaticData.HemisphereOfEarth` member representing the hemisphere of the Earth in which the study area is located.
- `month`: A `RomanDate.MonthOfYear` member representing the month in a year.
- `tmax`: Maximum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).
- `tmin`: Minimum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).
- `RH`: Relative air humidity (%). **Note**: Value 70% is written as 70.
- `U2`: Wind velocity at 2 meters above ground (m s\(^{-1}\)).
- `z`: Altitude of the considered area (m).
- `n`: Actual sunshine duration in hours.

**Return Value**: A `Double` as the reference evapotranspiration in mm day\(^{-1}\).

8. Calculates the reference evapotranspiration using the Penman-Monteith approach (mm day\(^{-1}\)).

```csharp
public static double CalculateETo(double latitude, HemisphereOfEarth hemisphere, MonthOfYear month, double tmean, double tmax, double tmin, double RH, double U2, double z, double n);
```

**Parameters**:

- `latitude`: Latitude of the study area. Values are expressed in decimal degrees. Signs (+/-) preceding the numeric value will be ignored. Northern or Southern Hemispheres are specified by the `hemisphere` parameter.
- `hemisphere`: A `ClimaticData.HemisphereOfEarth` member representing the hemisphere of the Earth in which the study area is located.
- `month`: A `RomanDate.MonthOfYear` member representing the month in a year.
- `tmean`: Mean air temperature (°C). If not readily available, can be obtained from the maximum and the minimum daily temperature (see Equation 5.4).
- `tmax`: Maximum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the minimum air temperature (see Equation 5.2).
- `tmin`: Minimum daily temperature(°C). If not readily available, can be obtained from the mean air temperature and the maximum air temperature (see Equation 5.3).
- `RH`: Relative air humidity (%). **Note**: Value 70% is written as 70.
- `U2`: Wind velocity at 2 meters above ground (m s\(^{-1}\)).
- `z`: Altitude of the considered area (m).
- `n`: Actual sunshine duration in hours.

**Return Value**: A `Double` as the reference evapotranspiration in mm day\(^{-1}\).
9. Calculates the year-round daily, decadal or monthly reference evapotranspiration \( \text{ET}_0 \), mm) for a particular location using the Penman-Monteith approach (Monteith, 1965).

\[\text{public static double[]} \text{CalculateETo(WlesDataSet data);}\]

**Parameters:**

- `data` A `WlesDataSet` instance that contains data.

**Return Value:** A `Double` as the reference evapotranspiration in mm day\(^{-1}\) or mm decade\(^{-1}\).

## 5.8 RootingDepth Class

### 5.8.1 Summary

Represents the rooting depth dynamics during the `CropCycle`.

### 5.8.2 Declaration

\[\text{public class RootingDepth}\]

### 5.8.3 Remarks

1. The rooting depth increases linearly from the start of the `CropCycle` till the end of the CD stage, when the optimal depth is reached.

2. The *optimal* rooting depth of a reference crop is defined as a range and stored in the `WLES` database.

3. A soil *limiting* layer prevents roots from penetrating deeper and therefore replaces the optimal rooting depth figure as the maximum depth in the resulting rooting depth dynamics.

The following criteria are commonly used to determine the presence of a soil limiting layer:

1. An unconsolidated gravelly or stony horizon with at least 75% (w/w) coarse fragments; or

2. A continuous CaCO\(_3\)-layer of at least 30 cm thickness and containing at least 60% CaCO\(_3\); or

3. A continuous gypsum layer of at least 30 cm thickness and containing at least 25% gypsum; or

4. A continuous layer of hard rock or a hardpan of at least 10 cm thickness.

### 5.8.4 Public Static Methods

- **GetOptimalDepth**

  Gets the optimal rooting depth (m) of a particular crop at the end of the CD stage.

\[\text{public static double[]} \text{GetOptimalDepth(string crop);}\]

**Exceptions:**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>crop</code> string is empty or the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

- **GetOptimalDepthRange**

  Gets the range (i.e., the minimum and maximum values) of the optimal rooting depth of a particular `crop` at the end of the CD stage.

\[\text{public static double[]} \text{GetOptimalDepthRange(string crop);}\]
5.8 RootingDepth Class

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop string is empty or the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

5.8.5 Public Instance Constructors

- public RootingDepth(WlesDataSet data);
  Initializes a new instance of RootingDepth with a particular WlesDataSet instance.

- public RootingDepth(CropCycle cycle);
  Initializes a new instance of RootingDepth with the CropCycle and the optimal rooting depth that is predefined in the WLES database.

- public RootingDepth(CropCycle cycle, double optimal, double limiting);
  Initializes a new instance of RootingDepth with the CropCycle, the optimal rooting depth and the depth of the soil limiting layer.

5.8.6 Public Instance Properties

- Crop
  Gets the crop name.
  [C#] public string Crop { get; }

- CropCycleBeginningRomanDate
  Gets the beginning RomanDate of the CropCycle associated with this instance of RootingDepth.
  [C#] public RomanDate CropCycleBeginningRomanDate { get; }

- CropCycleEndingRomanDate
  Gets the ending RomanDate of the CropCycle associated with this instance of RootingDepth.
  [C#] public RomanDate CropCycleEndingRomanDate { get; }

- Cultivar
  Gets the cultivar name.
  [C#] public string Cultivar { get; }

- LimitingLayerDepth
  Gets the depth of the soil limiting layer (m); -1.0 if it does not exist.
  [C#] public double LimitingLayerDepth { get; }

- MaximumDepth
  Gets the maximum rooting depth (m), which is the shallower between the optimal rooting depth and the soil limiting layer depth if the latter presents, or simply the optimal rooting depth if there is no limiting layer in the soil.
  [C#] public double MaximumDepth { get; }

- OptimalDepth
  Gets the optimal rooting depth of the Crop (m).
  [C#] public double OptimalDepth { get; }

- RootingDepthCropCycle
  Gets a Double array that represents the daily rooting depth (m) dynamics during the whole CropCycle.
  [C#] public double[] RootingDepthCropCycle { get; }

Remarks:
1. The length of this array equals the length of the crop cycle.
2. This array is 0 based.

5.8.7 Public Instance Methods

- GetRootingDepthDailyArray
  Overloaded.

1. Gets a `Double` array that represents the daily rooting depth (m) dynamics in a calendar year.
   ```csharp
   [C#] public double[] GetRootingDepthDailyArray();
   Return Value: A Double array of length 360 if CropCycleBeginningRomanDate and CropCycleEndingRomanDate use WlesCalendar or 365 if GregorianCalendar.
   ```

2. Gets a `Double` array that represents the daily rooting depth (m) dynamics in a calendar year.
   ```csharp
   [C#] public double[] GetRootingDepthDailyArray(CalendarType calendar);
   Return Value: A Double array of length 360 if calendar equals WlesCalendar or 365 if GregorianCalendar.
   ```

Remarks:

1. The length of the returned array is `RomanDate.CalendarType` dependent: 360 if both CropCycleBeginningRomanDate and CropCycleEndingRomanDate use WlesCalendar; 365 if GregorianCalendar.

2. Elements that fall outside the CropCycle will all have an exceptional value.

- GetSoilWaterStorageCropCycle
  Gets the daily available soil water storage \( S_a D \) (mm) during the CropCycle.
  ```csharp
  [C#] public double[] GetSoilWaterStorageCropCycle(SoilProfile profile);
  Return Value: A Double array that represents the available soil water storage (mm) in the root zone during the CropCycle; a null reference if profile is null.
  ```

5.9 RootingDepthUI Class

5.9.1 summary

Represents the RootingDepth determination and reporting interface of the WLES Web UI.

5.9.2 Declaration

```csharp
[C#] public class RootingDepthUI : Page
```

5.9.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

```
rooting.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.
5.10 WaterBalance Class

5.10.1 Summary

Represents the simulation of the plant available soil water dynamics during the CropCycle, and the evaluation of the actual crop evapotranspiration ($ET_a$), the yield reducing coefficient due to water stress ($f_W$) and the water-limited production potential ($WPP$).

5.10.2 Declaration

[C#] public class WaterBalance

5.10.3 Remarks

1. The plant available soil water dynamics are simulated on a daily basis;

2. The maximum crop evapotranspiration ($ET_m$, or simply crop evapotranspiration $ET_c$) is evaluated using a two-step approach: the $ET_m$ is obtained by transforming the reference evapotranspiration ($ET_o$) using the crop coefficient ($k_c$):

$$ET_m = k_c \cdot ET_o$$  (5.5)

3. The actual crop evapotranspiration ($ET_a$) is the quantity of water that is actually removed from the soil due to the processes of evaporation and transpiration. It consists of two parts: the part when the crop evapotranspirates at its maximum rate ($ET_m$) and the part when the crop evapotranspirates at a reduced rate. To distinguish these two, the soil water depletion fraction, $p$, is used:

$$ET_a = ET_m, \quad \{if:\; S_i D > (1-p) \cdot S_a D\}$$

$$ET_a < ET_m, \quad \{if:\; S_i D \leq (1-p) \cdot S_a D\}$$  (5.6)
where: $S_t D = \text{total available soil water storage over the rooting depth at time } t$; $S_a D = \text{plant available water over the rooting depth}$. The part $(p \cdot S_a D)$ is called \textit{easily available soil water}, and $(1 - p) \cdot S_a D$ \textit{scarcely available soil water}.

4. The accumulative effect of water stress on crop yield is measured by the “yield reducing coefficient due to water stress” ($f_W$):

$$f_W = \sum_{i=1}^{n} \left[ 1 - k_{y,i} \cdot \left( 1 - \frac{ET_{a,i}}{ET_{m,i}} \right) \right]$$  \hspace{1cm} (5.7)

where: $i = \text{number of days since the beginning of the CropCycle}$; $n = \text{length of the CropCycle}$; $k_{y,i} = \text{yield response factor of the } i^{th} \text{ day}$; $ET_{a,i} = \text{actual crop evapotranspiration of the } i^{th} \text{ day}$; $ET_{m,i} = \text{maximum crop evapotranspiration of the } i^{th} \text{ day}$.

5. The water-limited production potential ($WPP$) is achieved as a product of the $f_W$ and the RPP:

$$WPP = f_W \cdot RPP$$  \hspace{1cm} (5.8)

5.10.4 Examples

The following two examples demonstrate how to evaluate the yield reducing coefficient due to water stress ($f_W$) and the water-limited production potential ($WPP$) in Kigali (Rwanda) and in Ngaoundere (Cameroon) for soybean plantation during April and August. The rainfall difference between these two places is clearly reflected in the resulting $f_W$’s. \textbf{Example 1}

```csharp
using System;
using UGent.WLES;
using UGent.WLES.RPP;
using UGent.WLES.WPP;

class WPPKigali {
    public static void Main() {
        double latitude, longitude, elevation;
        string place;
        double[] tmin, tmax, RH, n, rain, wind;
        SoilHorizon[] horizons;
        int rainfall_interval;
        double limiting_layer;
        // the dataset
        place = "Kigali (Rwanda)";
        tmax = new double[] {
        tmin = new double[] {
            14.9, 14.9, 15.2, 15.5, 15.5, 14.6, 14.5, 15.3, 15.1, 15.2, 14.9, 15.1};
        RH = new double[] {
            76.5, 74.77, 81.5, 77.5, 71, 61.5, 62.5, 67.5, 73.5, 76.5, 76.5, 76.5};
        wind = new double[] {
            1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8};
        n = new double[] {
            5.4, 5.5, 5.2, 4.9, 5.5, 7.0, 7.0, 6.7, 5.6, 5.6, 5.0, 5.3};
        rain = new double[] {
            68.2, 110.6, 113.9, 178.5, 91.8, 22.5, 9.7, 31.1, 74.1, 94.7, 132, 88.6};
        rainfall_interval = 3;
        latitude = (1 + 58.0 / 60.0) * (-1);
        longitude = 30 + 7.0 / 60.0;
        elevation = 1500;
    }
}
```
horizons = new SoilHorizon [6];
string [] id = new string []
   {"Ap", "AB", "Bt", "Btu1", "Btu2", "2Cr", };
for (int i=0; i<id.Length; i++) {
    horizons[i] = new SoilHorizon(id[i]);
    horizons[i].Gravel = 0;
}

horizons[0].Depth = new double [] {0, 30};
horizons[1].Depth = new double [] {30, 50};
horizons[2].Depth = new double [] {50, 101};
horizons[3].Depth = new double [] {101, 140};
horizons[4].Depth = new double [] {140, 160};
horizons[5].Depth = new double [] {160, 185};

horizons[0].ParticleDistribution = new double [] {38.6,25.3,36.1};
horizons[1].ParticleDistribution = new double [] {39.4,24.2,36.4};
horizons[2].ParticleDistribution = new double [] {44.5,22.6,32.9};
horizons[3].ParticleDistribution = new double [] {38.5,22.8,38.7};
horizons[4].ParticleDistribution = new double [] {42.1,21.1,36.8};
horizons[5].ParticleDistribution = new double [] {60.4,12.5,27.1};

string result="";
try {
    WlesDataSet data = new WlesDataSet();
data.SetLocationalData(place, latitude, longitude, elevation);
data.SetClimaticData(tmax, tmin, RH, wind, n, rain);
    RomanDate cc1, cc2;
    string crop, cultivar;
    int lenIS, lenCD, lenMS, lenLS;
    // crop: soybean
cc1 = new RomanDate(4,1); cc2 = new RomanDate(8,10);
crop = "soybean"; cultivar = "";
lenIS = 20; lenCD = 40; lenMS = 50; lenLS = 20;
data.SetCropData(crop, cultivar, CropData.GetLeafAreaIndex(crop),
    CropData.GetHarvestIndex(crop,cultivar),
    CropData.GetOptimalRootingDepth(crop));
data.SetCropCycle(cc1, lenIS, lenCD, lenMS, lenLS);
CropCycle cc = new CropCycle(data);
result += string.Format("Crop cycle: \n{0}\n", cc);
Biomass biomass = new Biomass(data);
biomass.Evaluate();
data.SetWettingInterval(rainfall_interval);
data.SetSoilData(horizons, limiting_layer);
WaterBalance wb = new WaterBalance(data);
// soil water storage
result += "Easily available soil water storage: mm\n";
double[] psaw = wb.EasilyAvailableSoilWaterCropCycle;
for (int i=0; i<psaw.Length; i++) {
    result += string.Format("{0} ", Math.Round(psaw[i],2));
    if ((i+1)%10==0) result += "\n";
}
result += "\n";
double[] prec = CropCycle.TransformData(data.GetRainfallDailyArray(), cc);
double[] peff = EffectiveRainfall.UseUSDA_SCS(data);
// effective rainfall crop cycle, mm
result += "\nEffective rainfall crop cycle, mm";
result += "\n";
for (int i=0; i<peff.Length; i++) {
    result += string.Format("{0} ", Math.Round(peff[i],2));
    if ((i+1)%10==0) result += "\n";
}
result += "\n";
wb.Simulate(peff);
result += string.Format("\nETm = {0} mm, ETa = {1} mm\n", 
    Math.Round(wb.GetMaximumEvapotranspirationSum(), 2), 
    Math.Round(wb.GetActualEvapotranspirationSum(), 2));
result += "\n";
data.SetCropGrowthPeriods(new int[] {11,37,32,37,13}, true);
double[][] ky = new double[5][];
ky[0] = new double[11];
for (int i=0; i<ky[0].Length; i++) ky[0][i] = 0.086;
for (int i=0; i<ky[4].Length; i++) ky[4][i] = 0.23;
ky[1] = new double[] {0.15, 0.25};
ky[2] = new double[] {0.75};
ky[3] = new double[] {1};
YieldResponseToWater response = new YieldResponseToWater(data, ky);
wba.Evaluate(biomass.RPP, response);
// print CGP
result += response.CGPToString();
result += "\n";
result += string.Format("RPP = {0} kg/ha; fW = {1}; WPP = {2} kg/ha", 
    Math.Round(biomass.RPP, 2), 
    Math.Round(wb.WPP, 2));
}
catch (Exception e) {
    result = e.Message;
}
Console.WriteLine(result);
}

This example generates the following results:

Crop cycle:
Crop cycle of soybean in Kigali (Rwanda):
5.10 WaterBalance Class

Apr 01 (WLES Calendar) - Aug 10 (WLES Calendar)
Length of crop cycle: 130 days
Crop development stages:
IS: Apr 01 (WLES Calendar) - Apr 20 (WLES Calendar),
20 days [Initial]
CD: Apr 21 (WLES Calendar) - May 30 (WLES Calendar),
40 days [Crop Development]
MS: Jun 01 (WLES Calendar) - Jul 20 (WLES Calendar),
50 days [Mid-Season]
LS: Jul 21 (WLES Calendar) - Aug 10 (WLES Calendar),
20 days [Late Season]

Easily available soil water storage: mm
0 0.73 1.47 2.2 2.93 3.67 4.4 5.13 5.87 6.6
14.66 15.35 16.03 16.72 17.39 18.07 18.74 19.4 20.06 20.72
21.53 22.19 22.84 23.49 24.14 24.78 25.42 26.05 26.68 27.31
27.92 28.53 28.94 29.34 30.14 30.73 31.32 31.91 32.49 33.07
33.2 33.72 34.24 34.75 35.15 35.25 35.75 36.25 36.73 37.21 37.53
37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32
37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32
37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32 37.32
36.66 36.66 36.66 36.66 36.66 36.66 36.66 36.66 36.66 36.66
36.06 36.06 36.06 36.06 36.06 36.06 36.06 36.06 36.06 36.06
35.04 35.04 35.04 35.04 35.04 35.04 35.04 35.04 35.04 35.04
34.66 34.66 34.66 34.66 34.66 34.66 34.66 34.66 34.66 34.66
34.46 35.03 35.6 36.17 36.74 37.32 37.89 38.46 39.03 39.59
39.39 39.89 40.34 40.79 41.24 41.69 42.14 42.59 43.05 44.59

Effective rainfall crop cycle, mm
2.19 2.63 3.18 3.46 3.57 3.64 3.7 3.75 3.78 3.78
3.86 3.86 3.86 3.86 3.86 3.86 3.86 3.86 3.86 3.86
2.67 2.67 2.68 2.68 2.69 2.69 2.7 2.7 2.71
2.16 2.16 2.17 2.17 2.18 2.18 2.18 2.19 2.19 2.2
1.65 1.65 1.66 1.66 1.66 1.67 1.67 1.68 1.68 1.68
0.98 0.98 0.98 0.98 0.98 0.98 0.98 0.98 0.98 0.98
0.55 0.55 0.55 0.55 0.55 0.55 0.55 0.55 0.55 0.55
0.27 0.27 0.27 0.27 0.27 0.27 0.27 0.27 0.27 0.27
0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25
0.24 0.24 0.24 0.24 0.24 0.24 0.24 0.24 0.24 0.24
0.33 0.33 0.32 0.32 0.32 0.32 0.32 0.31 0.31 0.31
0.54 0.54 0.53 0.53 0.52 0.52 0.52 0.51 0.51 0.5
ETm = 392.52 mm, ETA = 182.11 mm

Crop growth periods:
CGP(0): Apr 01 (WLES Calendar) - Apr 11 (WLES Calendar), 11 days [Establishment]
CGP(1): Apr 12 (WLES Calendar) - May 18 (WLES Calendar), 37 days [Vegetative]
CGP(2): May 19 (WLES Calendar) - Jun 20 (WLES Calendar), 32 days [Flowering]
CGP(3): Jun 21 (WLES Calendar) - Jul 27 (WLES Calendar), 37 days [Yield Formation]
CGP(4): Jul 28 (WLES Calendar) - Aug 10 (WLES Calendar), 13 days [Ripening]

RPP = 3969.31 kg/ha; fW = 0.63; WPP = 2507.64 kg/ha

Example 2
Evaluation of fW and WPP for soybean in Ngaoundere (Cameroon). Same code but with the following data:
place = "Ngaoundere (Cameroon)";
Namespace UGent.WLES.WPP

tmin = new double [] {12.7, 13.5, 16.6, 17.3, 17.1, 17.6, 16.7, 16.5, 17.2, 14.2, 12};
RH = new double [] {34.5, 35.7, 49.4, 65.1, 75.3, 78.1, 80.1, 81.3, 78.7, 74.1, 59.6, 43.1};
wind = new double [] {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
n = new double [] {8.81, 8.43, 7.01, 5.9, 5.86, 5.02, 3.75, 3.83, 4.38, 5.63, 8.49, 9.24};
rain = new double [] {2, 2, 49, 134, 206, 237, 280, 300, 235, 146, 12, 2};
rainfall_interval = 3;
latitude = 7 + 21.0 / 60.0;
longitude = 13 + 34.0 / 60.0;
elevation = 1014;

horizons = new SoilHorizon [4];
string [] id = new string [] {"Ap", "Ah", "ABh", "Box", }; for (int i=0; i<id.Length; i++) {
    horizons[i] = new SoilHorizon(id[i]);
    horizons[i].Structure = SoilData.SoilStructure.Blocky;
    horizons[i].BulkDensity = 1.45;
}

horizons[0].Depth = new double [] {0, 24};
horizons[1].Depth = new double [] {24, 45};
horizons[2].Depth = new double [] {45, 72};
horizons[3].Depth = new double [] {72, 120};

horizons[0].ParticleDistribution = new double [] {37, 44, 19};
horizons[1].ParticleDistribution = new double [] {65, 17, 18};
horizons[2].ParticleDistribution = new double [] {69, 16, 15};
horizons[3].ParticleDistribution = new double [] {46, 27, 27};

horizons[0].Gravel = 15;
horizons[1].Gravel = 25;
horizons[2].Gravel = 0;
horizons[3].Gravel = 18;

This example generates the following results (compare to example 1):

Crop cycle:
Crop cycle of soybean in Ngaoundere (Cameroon):
   Apr 01 (WLES Calendar) – Aug 10 (WLES Calendar)
Length of crop cycle: 130 days
Crop development stages:
   IS: Apr 01 (WLES Calendar) – Apr 20 (WLES Calendar), 20 days [Initial]
   CD: Apr 21 (WLES Calendar) – May 30 (WLES Calendar), 40 days [Crop Development]
   MS: Jun 01 (WLES Calendar) – Jul 20 (WLES Calendar), 50 days [Mid-Season]
   LS: Jul 21 (WLES Calendar) – Aug 10 (WLES Calendar), 20 days [Late Season]

Easily available soil water storage: mm
0 2.45 4.91 7.36 9.81 12.27 14.72 17.17 19.63 22.08
24.76 27.24 29.72 32.19 34.67 37.15 39.62 42.1 44.58 47.05
49.9 52.24 54.56 56.87 59.16 61.43 63.69 65.94 68.17 70.39
73.33 75.56 77.75 79.95 82.12 84.28 86.43 88.56 90.68 92.78
95.6 97.71 99.79 101.87 103.88 105.82 107.74 109.64 111.52 113.38
116.59 118.45 120.3 122.13 123.94 125.74 127.51 129.27 131.01 132.19
135.46 135.46 135.46 135.46 135.46 135.46 135.46 135.46 135.46
137.21 137.21 137.21 137.21 137.21 137.21 137.21 137.21 137.21 137.21
140.69 140.69 140.69 140.69 140.69 140.69 140.69 140.69 140.69 140.69
141.61 141.61 141.61 141.61 141.61 141.61 141.61 141.61 141.61 141.61
142.08 143.09 144.1 145.11 146.12 147.13 148.13 149.14 150.15 151.16
151.36 152.4 155.13 156.58 158.03 159.5 160.97 162.45 163.95 166.96
Effective rainfall crop cycle, mm
1.45 2.33 2.47 2.5 2.5 2.5 2.5 2.5 2.5
3.02 3.02 3.02 3.02 3.02 3.02 3.02 3.02 3.02
3.49 3.49 3.5 3.51 3.51 3.52 3.53 3.53
4.01 4.02 4.02 4.03 4.04 4.04 4.05 4.05 4.06 4.07
4.37 4.38 4.39 4.4 4.41 4.41 4.42 4.43 4.44
4.65 4.66 4.67 4.68 4.69 4.7 4.71 4.73
4.72 4.72 4.72 4.72 4.72 4.72 4.72 4.72
4.87 4.87 4.87 4.87 4.87 4.87 4.87 4.87
5.02 5.02 5.02 5.02 5.02 5.02 5.02 5.02 5.02
5.26 5.26 5.26 5.26 5.26 5.26 5.26 5.26
5.39 5.39 5.39 5.39 5.39 5.39 5.39 5.39
5.49 5.48 5.47 5.46 5.45 5.44 5.44 5.44
5.63 5.64 5.66 5.68 5.7 5.74 5.78 5.82 5.88
5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9 5.9
6.3 6.4 6.5 6.6 6.6 6.7 6.8 6.8
ETm = 360.70 mm, ETA = 355.03 mm

Crop growth periods:
CGP(0): Apr 01 (WLES Calendar) - Apr 11 (WLES Calendar), 11 days [Establishment]
CGP(1): Apr 12 (WLES Calendar) - May 18 (WLES Calendar), 37 days [Vegetative]
CGP(2): May 19 (WLES Calendar) - Jun 20 (WLES Calendar), 32 days [Flowering]
CGP(3): Jun 21 (WLES Calendar) - Jul 27 (WLES Calendar), 37 days [Yield Formation]
CGP(4): Jul 28 (WLES Calendar) - Aug 10 (WLES Calendar), 13 days [Ripening]

RPP = 3737.77 kg/ha; fW = 0.99; WPP = 3732.04 kg/ha

5.10.5 Public Static Methods

- CalculateYieldReducingCoefWater
  Overloaded.

  1. Calculates the aggregate yield reducing coefficient due to water stress ($f_W$, dimensionless, [0, 1]) during a single time interval.
  ```csharp
  [C#] public static double CalculateYieldReducingCoefWater(double ky, double ETA, double ETm);
  ```

  2. Calculates the aggregate yield reducing coefficient due to water stress ($f_W$, dimensionless, [0, 1]) during a compound time interval.
  ```csharp
  [C#] public static double CalculateYieldReducingCoefWater(double[] ky, double[] ETA, double[] ETm, int[] length);
  ```

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the parameters of the method do not have the same length.</td>
</tr>
</tbody>
</table>

- GetCropsBySoilWaterDepletionGroup
  Gets all crops (including cultivar and alias, if applicable) in a particular soil water depletion group.
  ```csharp
  [C#] public static string[][] GetCropsBySoilWaterDepletionGroup(SoilWaterDepletionGroup group);
  ```

  Return Value: A jagged `String` array. The outer array contains the records of crops which in turn are 3-element arrays. The length of the outer array equals the number of crops in that particular group. Each record contains three element strings, in the order of crop name, cultivar name and alias name. As a result, the whole returned array is in the following format: `[crop1,cultivar1,alias1],...[crop1,cultivar1,alias1],...[cropn,cultivarn,aliasn],...[cogn,cultivarn,aliasn]`.  

- CalculateYieldReducingCoefWater
  Overloaded.

  1. Calculates the aggregate yield reducing coefficient due to water stress ($f_W$, dimensionless, [0, 1]) during a single time interval.
  ```csharp
  [C#] public static double CalculateYieldReducingCoefWater(double ky, double ETA, double ETm);
  ```

  2. Calculates the aggregate yield reducing coefficient due to water stress ($f_W$, dimensionless, [0, 1]) during a compound time interval.
  ```csharp
  [C#] public static double CalculateYieldReducingCoefWater(double[] ky, double[] ETA, double[] ETm, int[] length);
  ```

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the parameters of the method do not have the same length.</td>
</tr>
</tbody>
</table>
Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when underlying database errors occur.</td>
</tr>
</tbody>
</table>

- GetPAW
  Overloaded.

  1. Calculates the plant available soil water (PAW) using a $pF$ curve, % v/v or cm m$^{-1}$.

  ```csharp
  public static double GetPAW(double fc, double wp);
  ```

  2. Calculates the plant available soil water (PAW) using a $pF$ curve, % v/v or cm m$^{-1}$.

  ```csharp
  public static double GetPAW(double fc, double wp, double bd);
  ```

  Parameters:
  - `fc` A `Double` that represents the field capacity, % w/w.
  - `wp` A `Double` that represents the permanent wilting point, % w/w.
  - `bd` A `Double` that represents the bulk density, kg m$^{-3}$.

  3. Estimates the plant available soil water (PAW) for a particular `SoilData.SoilTextureClass`, % v/v or cm m$^{-1}$.

  ```csharp
  public static double GetPAW(SoilTextureClass texture);
  ```

  Remarks:
  - (a) Different soils have different water holding capacities. The maximum volume of water a saturated soil can hold after free drainage is termed as “field capacity”, while the minimum content of soil water on which a crop still survives is called “wilting point”. The volume of water between these two is therefore “plant available water”
  - (b) A soil’s water holding capacity is closely related to its texture. Thus the PAW can be estimated using the texture class.
  - (c) This method is implemented using the equivalent `SoilData.GetPAW` method.

  4. Estimates the plant available soil water (PAW) for a particular `SoilData.SoilTextureClass`, % v/v or cm m$^{-1}$.

  ```csharp
  public static double GetPAW(SoilTextureClass[] textures);
  ```

  Remarks: Soil texture classes are defined as polygons in the soil texture triangle. A point within a polygon is viewed as a member of the texture class that the polygon stands for. However, a point on the common border of two neighbouring polygons is viewed as a member of both classes. Hence a point that overlaps a common node of three or more neighbouring polygons is member of all the involving classes.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>textures</code> array empty or a null reference.</td>
</tr>
</tbody>
</table>

- GetRevisedPAW
  Overloaded.

  1. Revises the plant available soil water (PAW, % v/v) with coarse fragments (% v/v).

  ```csharp
  public static double GetRevisedPAW(double paw, double cf);
  ```

  2. Revises the plant available soil water (PAW, % v/v) with coarse fragments (% w/w).

  ```csharp
  public static double GetRevisedPAW(double paw, double cf, double bd);
  ```
### 5.10 WaterBalance Class

- **GetSoilWaterDepletionFraction**
  
  Overloaded.

  1. Gets the soil water depletion fraction \((p)\), to which the crop can evapotranspirate at the maximum rate.

  [C#]
  ```
  public static double GetSoilWaterDepletionFraction(double ETm);
  ```

  **Remarks:**
  
  (a) This method estimates the soil water depletion fraction \((p)\) using the maximum crop evapotranspiration \((ET_m, \text{ mm day}^{-1})\) data.

  (b) The overloading `GetSoilWaterDepletionFraction` method is more crop-specific. Therefore \(p\) should be estimated using the overloading method as long as data is available.

  (c) This method is implemented using the `CropData.GetSoilWaterDepletionFraction` method.

  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (ETm) is negative.</td>
</tr>
</tbody>
</table>

  2. Gets the soil water depletion fraction \((p)\), to which the crop can evapotranspirate at the maximum rate.

  [C#]
  ```
  public static double GetSoilWaterDepletionFraction(string crop, double ETm);
  ```

  **Remarks:** This method is implemented using the `CropData.GetSoilWaterDepletionFraction` method.

  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (crop) is a null reference or empty; and when (ETm) is negative; and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

  3. Gets the soil water depletion fraction \((p)\), to which the crop can evapotranspirate at the maximum rate.

  [C#]
  ```
  public static double GetSoilWaterDepletionFraction(string crop, string cultivar, double ETm);
  ```

  **Remarks:** This method is implemented using the `CropData.GetSoilWaterDepletionFraction` method.

  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (crop) is a null reference or empty; and when (ETm) is negative; and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>

- **GetSoilWaterDepletionGroupID**

  Gets the soil water depletion group ID of a particular crop.

  [C#]
  ```
  public static SoilWaterDepletionGroup GetSoilWaterDepletionGroupID(string crop, string cultivar);
  ```

  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when argument (crop) or (cultivar) is empty string; and when the database query returns nothing.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when the underlying database error occurs.</td>
</tr>
</tbody>
</table>
- GetSoilWaterStorage
  Overloaded.

1. Gets the available soil water storage \( S_aD \) mm at a particular rooting depth.
   
   `[C#]` public static double GetSoilWaterStorage(SoilHorizon[] horizons, double root);

2. Gets the available soil water storage \( S_aD \) mm at a particular rooting depth.
   
   `[C#]` public static double GetSoilWaterStorage(SoilProfile profile, double root);

Remarks:

1. The purpose of this method is to calculate the available soil water storage \( S_aD \) mm at any rooting depth to obtain the \( S_aD \) dynamics during the whole `CropCycle`.

2. The rooting depth is in most circumstances within the range of the soil horizons. However, if the availability of soil data is limited, depth of all data-available horizons may be shallower than the rooting depth. In this case, the method will extend the depth of the last known horizon till the maximum rooting depth.

3. This method is implemented using the `GetSoilWaterStorage` method.

5.10.6 Public Instance Constructors

- `public WaterBalance(WlesDataSet data);`
  Initializes a new instance of the `WaterBalance` class.

- `public WaterBalance(CropCoefficient kc, double[] ETo, SoilHorizon[] horizons, double optiRoot, double limitingLayer);`
  Initializes a new instance of the `WaterBalance` class.

  Parameters:

  `kc` The `CropCoefficient`.

  `ETo` A `Double` array of the year-round daily, decadely or monthly reference evapotranspiration data (mm).

  `horizons` An array of `SoilHorizon` that represents the soil.

  `optiRoot` A `Double` that represents the optimal rooting depth.

  `limitingLayer` A `Double` that represents the depth of the root limiting layer in soil. Use a negative value (e.g. -1) if it is not presented.

- `public WaterBalance(CropCoefficient kc, double[] ETo, SoilHorizon[] horizons, double optiRoot, double limitingLayer, double[] p);`
  Initializes a new instance of the `WaterBalance` class.

  Parameters:

  `kc` The `CropCoefficient`.

  `ETo` A `Double` array of the year-round daily, decadely or monthly reference evapotranspiration data (mm).

  `horizons` An array of `SoilHorizon` that represents the soil.

  `optiRoot` A `Double` that represents the optimal rooting depth.

  `limitingLayer` A `Double` that represents the depth of the root limiting layer in soil. Use a negative value (e.g. -1) if it is not presented.

  `p` A `Double` array of daily soil water depletion fraction values (dimensionless). The length of the array can be either that of a calendar year or of the `CropCycle`. 
5.10.7 Public Instance Properties

- **ActualEvapotranspirationCropCycle**
  Gets a `Double` array that represents the daily actual crop evapotranspiration \((ET_a, \text{ mm day}^{-1})\) during the CropCycle.

  ```csharp
  [C#] public double[] ActualEvapotranspirationCropCycle { get; }
  ```
  
  **Remarks:** The daily actual crop evapotranspiration \((ET_a)\) during the CropCycle is known only after the water balance is *simulated* and the yield reducing effects are *evaluated*. Prior to these steps, a null reference is returned upon access.

- **Crop**
  Gets the crop name.

  ```csharp
  [C#] public string Crop { get; }
  ```

- **CropCoefficientCropCycle**
  Gets a `Double` array that represents the crop coefficient (dimensionless) during the CropCycle.

  ```csharp
  [C#] public double[] CropCoefficientCropCycle { get; }
  ```

- **CropCycleBeginningRomanDate**
  Gets a `RomanDate` instance that represents the start of the CropCycle.

  ```csharp
  [C#] public RomanDate CropCycleBeginningRomanDate { get; }
  ```

- **CropCycleEndingRomanDate**
  Gets a `RomanDate` instance that represents the end of the CropCycle.

  ```csharp
  [C#] public RomanDate CropCycleEndingRomanDate { get; }
  ```

- **Cultivar**
  Gets the cultivar name.

  ```csharp
  [C#] public string Cultivar { get; }
  ```

- **EasilyAvailableSoilWaterCropCycle**
  Gets a `Double` array that represents the daily *easily* available soil water storage \((pS_a D, \text{ mm day}^{-1})\) in the root zone during the CropCycle.

  ```csharp
  [C#] public double[] EasilyAvailableSoilWaterCropCycle { get; }
  ```

- **EffectiveRainfallCropCycle**
  Gets a `Double` array that represents the daily EffectiveRainfall \((\text{mm day}^{-1})\) during the CropCycle.

  ```csharp
  [C#] public double[] EffectiveRainfallCropCycle { get; }
  ```
  
  **Remarks:** The EffectiveRainfall during the CropCycle is known only after the water balance is *simulated* and the yield reducing effects are *evaluated*. Prior to these steps, a null reference is returned upon access.

- **EvapotranspirationAtMaximumRateCropCycle**
  Gets a `Double` array that represents the daily crop evapotranspiration at the maximum rate \((ET_{am}, \text{ mm day}^{-1})\) during the CropCycle.

  ```csharp
  [C#] public double[] EvapotranspirationAtMaximumRateCropCycle { get; }
  ```
  
  **Remarks:**
  
  1. Crops evapotranspirate at different rates in response to soil water conditions. As long as the available soil water storage \((S_t D, \text{ where } t \text{ stands for time})\) exceeds the *scarce*ly available soil water storage \([(1 - p) S_a D, \text{ where } p \text{ is the soil water depletion fraction}]\), crop evapotranspirates at the maximum rate \((ET_{am})\), until:

  \[
  S_t D = (1 - p) \cdot S_a D 
  \]  
  (5.9)

  2. Water stress occurs as soon as:

  \[
  S_t D \leq (1 - p) \cdot S_a D 
  \]  
  (5.10)
Crops evapotranspirate now at a reduced rate ($ET_{ar}$), until the permanent wilting point ($PWT$) is reached.

3. For relationships among $ET_o$, $ET_m$ and $ET_a$, refer to the remarks section of the WaterBalance class documentation.

4. The daily crop evapotranspiration at maximum rate ($ET_{am}$) during the CropCycle is known only after the water balance is simulated and the yield reducing effects are evaluated. Prior to these steps, a null reference is returned upon access.

- EvapotranspirationAtReducedRateCropCycle
  Gets a Double array that represents the daily crop evapotranspiration at reduced rate ($ET_{ar}$, mm day$^{-1}$) during the CropCycle.

  ```csharp
  public double[] EvapotranspirationAtReducedRateCropCycle { get; }
  ```

  Remarks: See remarks section of EvapotranspirationAtMaximumRateCropCycle.

- MaximumEvapotranspirationCropCycle
  Gets a Double array that represents the maximum (crop) evapotranspiration ($ET_m$ or $ET_c$, mm day$^{-1}$) during the CropCycle.

  ```csharp
  public double[] MaximumEvapotranspirationCropCycle { get; }
  ```

- ReferenceEvapotranspirationCropCycle
  Gets a Double array that represents the reference evapotranspiration ($ET_o$, mm day$^{-1}$) during the CropCycle.

  ```csharp
  public double[] ReferenceEvapotranspirationCropCycle { get; }
  ```

- RootingDepthCropCycle
  Gets a Double array that represents the daily rooting depth (m) of the crop during the CropCycle.

  ```csharp
  public double[] RootingDepthCropCycle { get; }
  ```

- RPP
  Gets the radiation-thermal production potential (RPP) in kg ha$^{-1}$.

  ```csharp
  public double RPP { get; }
  ```

  Remarks:
  1. RPP is represented and manipulated by the RPP namespace, especially its Biomass class. The value provided here is merely a symbolic link pointing to the same name argument of the Evaluate method.
  2. Hence a meaningful RPP value is available only after the Evaluate method is called. Prior to this, -1 is returned upon access.

- ScarcelyAvailableSoilWaterCropCycle
  Gets a Double array that represents the daily scarcely available soil water storage $[(1-p)S_a D$, mm day$^{-1}$] in the root zone during the CropCycle.

  ```csharp
  public double[] ScarcelyAvailableSoilWaterCropCycle { get; }
  ```

- SoilHorizons
  Gets an array of SoilHorizon that represents the horizons of the soil.

  ```csharp
  public SoilHorizon[] SoilHorizons { get; }
  ```

- SoilWaterDepletionFractionCropCycle
  Gets a Double array that represents the soil water depletion fraction ($p$, dimensionless) during the CropCycle.

  ```csharp
  public double[] SoilWaterDepletionFractionCropCycle { get; }
  ```

- SoilWaterStorageCropCycle
  Gets a Double array that represents the available soil water storage at the root zone ($S_a D$, mm day$^{-1}$) during the CropCycle.
5.10 WaterBalance Class

- **WPP**
  Gets the water-limited crop productional potential (WPP) in kg ha\(^{-1}\)  
  \[
  \text{public double WPP} \{ \text{get;} \}
  \]
  **Remarks:**
  
  1. WPP is based on the rainfed crop production scenario in which water stress occurs in the crop cycle. The anticipated rainfed yield is estimated from the potential irrigated yield (RPP) and a reduction factor which expresses the relative yield loss as the result of the water deficit.
  
  2. A meaningful WPP value is available only after the water balance is simulated and the yield reducing effects are evaluated. Prior to these steps, -1 is returned upon access.

- **YieldReducingCoefWater**
  Gets the aggregate yield reducing coefficient due to water stress \((f_W, \text{dimensionless})\) during the whole CropCycle.  
  \[
  \text{public double YieldReducingCoefWater} \{ \text{get;} \}
  \]
  **Remarks:**
  
  1. \(f_W\) reflects the aggregate reducing effects of water stress on crop yield during the CropCycle. It should be evaluated on a per CropCycle.CropGrowthPeriod basis and the aggregation of the period-specific effects makes the value of the coefficient for the whole CropCycle. Depending on data availability, some critical CropCycle.CropGrowthPeriods may be divided into sub periods to reflect the intra-period discontinuity.
  
  2. A meaningful \(f_W\) value is available only after the water balance is simulated and the yield reducing effects are evaluated. Prior to these steps, -1 is returned upon access.

### 5.10.8 Public Instance Methods

- **Evaluate**
  Evaluates the yield reducing coefficient due to water stress \((f_W, \text{dimensionless})\) and estimates the water-limited production potential \((WPP, \text{kg ha}^{-1})\).  
  \[
  \text{public void Evaluate(double RPP, YieldResponseToWater response);} 
  \]
  **Parameters:**
  
  - **RPP** A Double that represents the radiation-thermal production potential (RPP) which should be evaluated separately and before-hand; expressed in kg ha\(^{-1}\).
  
  - **response** A YieldResponseToWater instance that brings the crop growth periods and \(k_y\) data into the evaluation process.

- **ExportResults**
  Exports the water balance simulation results in the “comma-separated values” (csv) format.  
  \[
  \text{public string ExportResults();} 
  \]
  **Remarks:**
  
  1. The following data columns are exported: days since beginning of crop cycle, \(ET_m\), rooting depth, \(S_a D\), \(p\), effective rainfall, \((1-p)S_a D\), \(ET\) at the maximum rate, \(ET\) at reduced rate, \(ET_a\).
  
  2. The collection of daily values of all data columns form one record. All records of the CropCycle form the export dataset.
  
  3. The sum of the following data columns are appended to the end of the export dataset: \(ET_m\), effective rainfall, \(ET\) at the maximum rate, \(ET\) at reduced rate, \(ET_a\).

**Example:** The following example shows explicitly the format of the export dataset for a hypothetical location.
Namespace UGent.WLES.WPP

Day, ETm, Root, SaD, p, Peff, (1-p)SaD, ETam, ETar, ETa
1, 2.6, 0, 0, 0.83, 1.45, 0, 1.45
2, 2.6, 0.02, 2.96, 0.83, 2.33, 0.5, 1.83, 0.39, 2.22
3, 2.6, 0.03, 5.91, 0.83, 2.47, 1.01, 1.57, 0.64, 2.22
4, 2.6, 0.05, 8.87, 0.83, 2.51, 1.36, 1.85, 0.85, 2.2
5, 2.6, 0.06, 11.82, 0.83, 2.5, 2.01, 1.15, 1.03, 2.18
6, 2.6, 0.08, 14.78, 0.83, 2.5, 2.51, 0.96, 1.2, 2.17
7, 2.6, 0.1, 17.74, 0.83, 2.5, 3.02, 0.78, 1.36, 2.15
8, 2.6, 0.11, 20.69, 0.83, 2.5, 3.52, 0.64, 1.5, 2.14

...  
125, 1.78, 0.95, 174.41, 0.91, 5.7, 16.37, 1.78, 0, 1.78
126, 1.7, 0.95, 174.41, 0.91, 5.74, 14.91, 1.7, 0, 1.7
127, 1.62, 0.95, 174.41, 0.92, 5.7, 13.43, 1.62, 0, 1.62
128, 1.54, 0.95, 174.41, 0.93, 5.82, 11.95, 1.54, 0, 1.54
129, 1.46, 0.95, 174.41, 0.94, 5.88, 10.46, 1.46, 0, 1.46
130, 1.3, 0.95, 174.41, 0.96, 6.03, 7.44, 1.3, 0, 1.3
Sum, 360.7, 355, 336.87, 18.16, 355.03

• GetActualEvapotranspirationDailyArray
  Gets a Double array that represents the daily actual evapotranspiration ($ET_a$, mm day$^{-1}$) during the CropCycle in an one-year timeframe.

[C#] public double[] GetActualEvapotranspirationDailyArray();
  **Return Value:** A 360- or 365-element Double array that contains the daily actual evapotranspiration (mm day$^{-1}$) during the CropCycle in an one-year timeframe.
  **Remarks:** The actual crop evapotranspiration ($ET_a$) during the CropCycle is known only after the water balance is simulated and the yield reducing effects are evaluated. Prior to these steps, a null reference is returned upon access.
  1. The returned array has 360 or 365 elements, depending on the CalendarType that the associated crop cycle dates use;
  2. Elements that fall outside the CropCycle have value -1.

• GetActualEvapotranspirationSum
  Gets the sum of the daily actual crop evapotranspiration ($ET_a$, mm) during the CropCycle.

[C#] public double GetActualEvapotranspirationSum();
  **Remarks:** The actual crop evapotranspiration ($ET_a$) during the CropCycle is known only after the water balance is simulated and the yield reducing effects are evaluated. Prior to these steps, 0 is returned upon access.

• GetCropCoefficientDailyArray
  Gets a Double array that represents the daily CropCoefficient values (dimensionless) during the CropCycle.

[C#] public double[] GetCropCoefficientDailyArray();

• GetEasilyAvailableSoilWaterDailyArray
  Gets a Double array that represents the daily easily available soil water storage ($pS_aD$, mm day$^{-1}$) during the CropCycle in an one-year timeframe.

[C#] public double[] GetEasilyAvailableSoilWaterDailyArray();

• GetEffectiveRainfallDailyArray
  Gets a Double array that represents the daily EffectiveRainfall (mm day$^{-1}$) during the CropCycle in an one-year timeframe.

[C#] public double[] GetEffectiveRainfallDailyArray();

• GetEvapotranspirationAtMaximumRateDailyArray
  Gets a Double array that represents the daily crop evapotranspiration at the maximum rate
5.10 WaterBalance Class

\((ET_{am}, \text{mm day}^{-1})\) during the CropCycle in an one-year timeframe.

[C#] public double[] GetEvapotranspirationAtMaximumRateDailyArray();

- GetEvapotranspirationAtMaximumRateSum
  Gets the sum of the daily crop evapotranspiration at the maximum rate \((ET_{am}, \text{mm day}^{-1})\) during the CropCycle.

[C#] public double GetEvapotranspirationAtMaximumRateSum();

- GetEvapotranspirationAtReducedRateDailyArray
  Gets a Double array that represents the daily crop evapotranspiration at reduced rate \((ET_{ar}, \text{mm day}^{-1})\) during the CropCycle.

[C#] public double[] GetEvapotranspirationAtReducedRateDailyArray();

- GetEvapotranspirationAtReducedRateSum
  Gets the sum of the daily crop evapotranspiration at reduced rate \((ET_{ar}, \text{mm day}^{-1})\) during the CropCycle.

[C#] public double GetEvapotranspirationAtReducedRateSum();

- GetMaximumEvapotranspirationDailyArray
  Gets a Double array that represents the daily maximum evapotranspiration \((ET_m \text{ or } ET_c, \text{mm day}^{-1})\) during the CropCycle.

[C#] public double[] GetMaximumEvapotranspirationDailyArray();

- GetMaximumEvapotranspirationSum
  Gets the sum of the maximum evapotranspiration \((ET_m \text{ or } ET_c, \text{mm})\) during the CropCycle.

[C#] public double GetMaximumEvapotranspirationSum();

- GetReferenceEvapotranspirationDailyArray
  Gets a Double array that represents the daily reference evapotranspiration \((ET_o, \text{mm day}^{-1})\) during the CropCycle in an one-year timeframe.

[C#] public double[] GetReferenceEvapotranspirationDailyArray();

- GetReferenceEvapotranspirationSum
  Gets the sum of the reference evapotranspiration \((ET_o, \text{mm})\) during the CropCycle.

[C#] public double GetReferenceEvapotranspirationSum();

- GetRootingDepthDailyArray
  Gets a Double array that represents the daily rooting depth (m) during the CropCycle in an one-year timeframe.

[C#] public double[] GetRootingDepthDailyArray();

- GetScarcelyAvailableSoilWaterDailyArray
  Gets a Double array that represents the daily scarcely available soil water storage \([(1-p)S_aD, \text{mm day}^{-1}]\) during the CropCycle in an one-year timeframe.

[C#] public double[] GetScarcelyAvailableSoilWaterDailyArray();

- GetSoilWaterDepletionFractionDailyArray
  Gets a Double array that represents the soil water depletion fraction \((p, \text{dimensionless})\) during the CropCycle in an one-year timeframe.

[C#] public double[] GetSoilWaterDepletionFractionDailyArray();

- GetSoilWaterStorageDailyArray
  Gets a Double array that represents the daily soil water storage at the root zone \((S_aD, \text{mm day}^{-1})\) during the CropCycle in an one-year timeframe.

[C#] public double[] GetSoilWaterStorageDailyArray();

- Simulate
  Simulates the soil water balance during the CropCycle in order to evaluate the actual crop evapotranspiration \((ET_a)\).
[C#] public void Simulate(double[] peffCropCycle);

Parameters:

peffCropCycle  A Double array that represents the daily EffectiveRainfall (mm) during the CropCycle.

Remarks:

1. This method only accepts daily EffectiveRainfall (mm) during the CropCycle as its input; in other words, the length of peffCropCycle array equals the length (days) of the CropCycle.

2. The daily EffectiveRainfall (mm) data of the CropCycle may be extracted from the DAILY, DECADELY or MONTHLY EffectiveRainfall (mm) data of a year:
   (a) Monthly or decedly data (year) to daily data (year):
       ClimaticData.GetDailyArray(monthly,
       ClimaticData.ClimaticParamSet.RAINFALL);
       or
       ClimaticData.GetDailyArray(decadely,
       ClimaticData.ClimaticParamSet.RAINFALL);
   (b) Daily data (year) to daily data (crop cycle):
       CropCycle.TransformData(daily, startingDate, endingDate);

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when peffCropCycle not same length as crop cycle.</td>
</tr>
</tbody>
</table>

• ToString

Delivers a water balance evaluation results report that contains $ET_a$, $ET_m$, yield reducing coefficient due to water stress ($f_W$) and the WPP.

[C#] public override string ToString();

Example: The following example shows explicitly the format of the report of water balance evaluation for soybean plantation during Apr. 1 to Aug. 10 in Kigali (Rwanda).

$ET_m = 392.52$ mm, $ET_a = 182.11$ mm, $f_W = 0.63$, WPP = 2507.64 kg/ha

5.11 WaterBalanceUI Class

5.11.1 Summary

Represents the WaterBalance simulation and reporting interface of the WLES Web UI.

5.11.2 Declaration

[C#] public class WaterBalanceUI : Page

5.11.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

`wb.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar`

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.
5.12 WPPReport Class

5.11.4 Internal Static Methods

- **SetupWaterBalance**
  Sets up a WaterBalance instance using the data associated with a particular Web Form page.

  ```c#
  [C#] internal static WaterBalance SetupWaterBalance(Page page);
  ```

  **Return Value:** A WaterBalance instance with the water balance simulation conducted on a daily basis during the crop cycle.

  **Remarks:**
  
  1. Simulation - The water balance has been *simulated* on a daily basis during the crop cycle. As a result, the actual evapotranspiration \( ET_a \) has been *evaluated* and thus available;
  2. Evaluation - However, the yield response factor \( k_y \) has not been evaluated (via the Evaluate method) and thus the yield reducing coefficient due to water stress \( f_W \) and the water-limited production potential are both not available before the YieldResponse EvalPhase is conducted;
  3. Data access - The WaterBalance instance encapsulates all the data arrays used during the simulation and the evaluation processes, which are exposed through its properties and methods.

5.12 WPPReport Class

5.12.1 Summary

Represents the UGent.WLES.WPP reporting interface of the WLES Web UI.

5.12.2 Declaration

  ```c#
  [C#] public class WPPReport : Page
  ```

5.12.3 Remarks

This page can be accessed with a query string which contains, among others, two *key-value* pairs entitled “rid” and “timeout”, as in the following example:

  ```
  wpp.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar
  ```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

5.12.4 Internal Static Methods

- **SetupWaterBalance**
  Sets up a WaterBalance instance using the data associated with a particular Web Form page.

  ```c#
  [C#] internal static WaterBalance SetupWaterBalance(Page page);
  ```

  **Return Value:** A WaterBalance instance that (1) the water balance has been *simulated* on a daily basis during the crop cycle; and (2) the WPP has been *evaluated*.

  **Remarks:**
  
  1. The main difference between this method and the same name method provided by the WaterBalanceUI is that the water balance simulation is conducted by the latter, while the WPP is evaluated by the former;
  2. For more information about the simulation, evaluation and data access please refer to the SetupWaterBalance method provided by the WaterBalanceUI.
5.13 YieldResponseToWater Class

5.13.1 Summary
Represents the crop’s yield response to water stress.

5.13.2 Declaration

[C#] public class YieldResponseToWater

5.13.3 Remarks

1. The relationship between crop yield and water supply can be determined when crop water requirements and crop water deficits, on the one hand, and maximum and actual crop yield on the other can be quantified. Water deficits in crops, and the resulting water stress on the plant, have an effect on crop evapotranspiration and crop yield. Water stress in the plant can be quantified by the rate of actual evapotranspiration ($ET_a$) in relation to the rate of maximum evapotranspiration ($ET_m$).

2. When the full crop water requirements are not met, water deficit can develop to a point where crop growth and yield are affected. The manner in which water deficit affects crop growth and yield varies with crop species and crop growth period. The effect of water stress on yield decrease can be evaluated through the quantification of relative evapotranspiration deficit ($1 - \frac{ET_a}{ET_m}$) and the crop’s yield response factor ($k_y$):

$$
(1 - \frac{Y_a}{Y_m}) = k_y \cdot (1 - \frac{ET_a}{ET_m})
$$

(5.11)

where: $Y_a = $ actual harvested yield; $Y_m = $ maximum harvested yield; $k_y = $ yield response factor; $ET_a = $ actual evapotranspiration; and $ET_m = $ maximum evapotranspiration.

3. The yield response factor ($k_y$) is empirically determined through the relative yield decrease in response to the evapotranspiration deficit. The magnitude of $k_y$ represents the crop’s sensitivity to water stress and varies from one CropCycle.CropGrowthPeriod to another during the whole CropCycle.

4. The accumulative effect of water stress on crop yield is measured by the “yield reducing coefficient due to water stress” ($f_W$), which is presented by Equation 5.7 (p. 204).

5. Therefore the water-limited production potential ($WPP$) is achieved as a product of the $f_W$ and the $RPP$, as seen in Equation 5.8 (p. 204).

5.13.4 Example

The following examples demonstrate the flexibilities of constructing a YieldResponseToWater instances in reference to the availability of the yield response factor ($k_y$) data.

1. **Single** $k_y$ value for crop cycle (Apr. 1 to Aug. 10) of soybean:

   ```csharp
   string crop = "soybean", cultivar="";
   RomanDate start = new RomanDate(4,1), end = new RomanDate(8,10);
   double ky = 0.85;
   YieldResponseToWater response =
      new YieldResponseToWater(crop,cultivar,start,end,ky);
   ```

   The division of CropCycle.CropGrowthPeriods is not important in this situation since the yield reducing coefficient due to water stress ($f_W$) is going to be evaluated against the whole CropCycle, instead of against CropCycle.CropGrowthPeriods and sub periods.
2. **Daily** \( k_y \) values for crop cycle (Apr. 1 to Aug. 10) of soybean:

```java
string crop = "soybean", cultivar="";
RomanDate start = new RomanDate(4,1), end = new RomanDate(8,10);
int len = start.Length(end);
double [] ky = new double[len];
ky[0] = 0.09; ...; ky[129] = 0.56;
YieldResponseToWater response =
    new YieldResponseToWater(crop,cultivar,start,end,ky);
```

The division of `CropCycle.CropGrowthPeriods` is not important in this situation since the yield reducing coefficient due to water stress \( f_W \) is going to be evaluated on a daily basis, instead of on `CropCycle.CropGrowthPeriods` and sub periods.

3. Heterogeneous \( k_y \) values for crop cycle (Apr. 1 to Aug. 10) of soybean as:

- **Establishment** period: single \( k_y = 0.08 \);
- **Vegetative** period: early (0.15) and late (0.25) sub-periods;
- **Flowering** period: daily \( k_y \) (0.65, 0.68, 0.70, 0.82, ..., 0.85, 0.86, 0.77, 0.83);
- **YieldFormation** period: single \( k_y = 1.00 \);
- **Ripening** period: single \( k_y = 0.35 \).

Lengths of `CropCycle.CropGrowthPeriods` are 11, 37, 32, 37, and 13 days.

```java
string crop = "soybean", cultivar="";
RomanDate start = new RomanDate(4,1), end = new RomanDate(8,10);
int lenCGPs = new int [] {11,37,32,37,13};
double [][] ky = new double[5][];
    ky[0] = new double [] {0.08};
    ky[1] = new double [] {0.15, 0.25};
        ky[2][0] = 0.65; ky[2][1] = 0.68; ...;
            ky[2][30] = 0.77; ky[2][31] = 0.83;
        ky[3] = new double [] {1.0};
        ky[4] = new double [] {0.35};
YieldResponseToWater response =
    new YieldResponseToWater(crop,cultivar,start,end,lenCGPs,ky);
```

The division of `CropCycle.CropGrowthPeriods` is important in this situation since the yield reducing coefficient due to water stress \( f_W \) is going to be evaluated on the daily/period/sub-period basis (**mixed-mode**).

4. Same heterogeneous \( k_y \) values as in previous example, and the data-set including the locational, crop, climatic and soil data had been well prepared elsewhere:

```java
WlesDataSet data = ...;
YieldResponseToWater response = new YieldResponseToWater(data,ky);
```

The division of `CropCycle.CropGrowthPeriods` is important in this situation since the yield reducing coefficient due to water stress \( f_W \) is going to be evaluated on the daily/period/sub-period basis (**mixed-mode**).

Refer to the `WaterBalance` class for a more realistic example of including the yield response factor \( k_y \) in evaluating the water-limited crop production potential \( WPP \).
5.13.5 Public Static Methods

- ExportRefKyData
  Exports yield response factor ($k_y$) values of the reference and user-defined crops in HTML format.
  
  `[C#] public static string ExportRefKyData();`

- GetRefCGP
  Overloaded.

  1. Gets the length (days) of `CropCycle.CropGrowthPeriods` of a reference crop (and cultivar). The length is expressed as a value range.
  
  `[C#] public static int[,] GetRefCGP(string crop, string cultivar, bool cropCycle);`

  Parameters:
  
  - `crop` A String that represents the crop.
  - `cultivar` A String that represents the cultivar.
  - `cropCycle` A Boolean to indicate whether to include the `CropCycle` length (in reference to the `CropCycle.CropGrowthPeriods`) in the returned array: `true` to include; `false` not.

  Return Value: A jagged integer array that represents the value ranges of lengths (days) of `CropCycle.CropGrowthPeriods` of the `crop` and `cultivar`.

  Remarks:

  (a) The method gets the lengths of the following `CropCycle.CropGrowthPeriods`: Establishment; Vegetative; Flowering; YieldFormation; and Ripening.
  (b) The total length of the `CropCycle` is included if `cropCycle` is `true`; not if `false`.
  (c) The length of a `CropCycle.CropGrowthPeriod` is expressed as a 2-element integer array, which is again an element of the jagged array returned. Length of the returned array is 5 or 6 depending on the `cropCycle` value.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) <code>crop</code> is empty or a null reference; or (2) <code>crop</code> and/or <code>cultivar</code> are/is ambiguous; in most cases, it is because that <code>cultivar</code> is not properly specified.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database query returns nothing. In most cases, it is because that the <code>crop</code> (and <code>cultivar</code>) is not a reference or a user-defined crop.</td>
</tr>
</tbody>
</table>

  2. Gets the length (days) of a particular `CropCycle.CropGrowthPeriod` of a reference crop (and cultivar). The length is expressed as a value range.

  `[C#] public static int[] GetRefCGP(string crop, string cultivar, CropGrowthPeriod period);`

  Parameters:

  - `crop` A String that represents the crop.
  - `cultivar` A String that represents the cultivar.
  - `period` The `CropCycle.CropGrowthPeriod` whose length (days) is to be returned.

  Return Value: A 2-element integer array that represents the value range of the period length (days) of the `crop` and `cultivar`.

  Exceptions:
5.13 YieldResponseToWater Class

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) crop is empty or a null reference; or (2) period is not one of the following: Establishment, Vegetative, Flowering, YieldFormation, and Ripening; or (3) crop and/or cultivar are/is ambiguous; in most cases, it is because that cultivar is not properly specified.</td>
</tr>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database query returns nothing. In most cases, it is because that the crop (and cultivar) is not a reference or a user-defined crop.</td>
</tr>
</tbody>
</table>

- **GetYieldResponseFactor**
  Overloaded.

  1. Gets the yield response factor \( (k_y) \) for the whole crop cycle of a particular crop and cultivar.

```csharp
[C#] public static double GetYieldResponseFactor(string crop, string cultivar);
```

**Return Value:** A Double that represents the yield response factor of the particular crop and cultivar during the entire CropCycle. -1 if the crop is not a reference or a user-defined one.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string or a null reference.</td>
</tr>
</tbody>
</table>

  2. Gets the yield response factor \( (k_y) \) of a particular crop and cultivar during the given CropCycle.CropGrowthPeriod.

```csharp
[C#] public static double GetYieldResponseFactor(string crop, string cultivar, CropGrowthPeriod cgp);
```

**Return Value:** A Double that represents the yield response factor of the particular crop and cultivar during the given CropCycle.CropGrowthPeriod. -1 if the crop is not a reference or a user-defined one.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string or a null reference.</td>
</tr>
</tbody>
</table>

  3. Gets the yield response factor \( (k_y) \) of a particular crop and cultivar during the given CropCycle.CropGrowthPeriod and sub period.

```csharp
[C#] public static double GetYieldResponseFactor(string crop, string cultivar, CropGrowthPeriod cgp, SubCGP earlyOrLate);
```

**Return Value:** A Double that represents the yield response factor of the particular crop and cultivar during the given CropCycle.CropGrowthPeriod and sub period. -1 if the crop is not a reference or a user-defined one.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string or a null reference.</td>
</tr>
</tbody>
</table>

- **GetYieldResponseFactorRange**
  Overloaded.

  1. Gets the range of the yield response factor \( (k_y) \) for the whole crop cycle of a particular crop and cultivar.

```csharp
[C#] public static double[] GetYieldResponseFactorRange(string crop, string cultivar);
```

**Return Value:** A 2-element Double array that represents the range of the yield response factor value of a particular crop and cultivar during the entire CropCycle. The
1st and 2nd elements of the array are the minimum and the maximum values of the range. Both elements have value -1 if the crop is not a reference or a user-defined one.

2. Gets the range of the yield response factor ($k_y$) of a particular crop and cultivar during the given crop growth period.

   \[
   \text{public static double[]} \text{GetYieldResponseFactorRange(string crop, string cultivar, CropGrowthPeriod cgp);} \\
   \text{Return Value: A 2-element Double array that represents the range of the yield response factor value of a particular crop and cultivar during the given CropCycle.CropGrowthPeriod. The 1st and 2nd elements of the array are the minimum and the maximum values of the range. Both elements have value -1 if the crop is not a reference or a user-defined one.}
   \]

3. Gets the range of the yield response factor ($k_y$) of a particular crop and cultivar during the given crop growth period and sub-period.

   \[
   \text{public static double[]} \text{GetYieldResponseFactorRange(string crop, string cultivar, CropGrowthPeriod cgp, SubCGP earlyOrLate);} \\
   \text{Return Value: A 2-element Double array that represents the range of the yield response factor value of a particular crop and cultivar during the given CropCycle.CropGrowthPeriod and sub-period. The 1st and 2nd elements of the array are the minimum and the maximum values of the range. Both elements have value -1 if the crop is not a reference or a user-defined one.}
   \]

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string or a null reference.</td>
</tr>
</tbody>
</table>

5.13.6 Public Instance Constructors

1. public YieldResponseToWater(string crop, string cultivar, RomanDate start, RomanDate end);
   Initializes a new instance of YieldResponseToWater.

2. public YieldResponseToWater(string crop, string cultivar, RomanDate start, RomanDate end, double ky);
   Initializes a new instance of YieldResponseToWater.

3. public YieldResponseToWater(string crop, string cultivar, RomanDate start, RomanDate end, double[] ky);
   Initializes a new instance of YieldResponseToWater.

4. public YieldResponseToWater(string crop, string cultivar, RomanDate start, RomanDate end, int[] lenCGPs, double[][] ky);
   Initializes a new instance of YieldResponseToWater.

5. public YieldResponseToWater(WlesDataSet data, double[][] ky);
   Initializes a new instance of YieldResponseToWater.

Exceptions:

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string or a null reference.</td>
</tr>
</tbody>
</table>

Remarks: Ways and situations in which the YieldResponseToWater instances are initialized
YieldResponseToWater Class

are summarized in the table below:

<table>
<thead>
<tr>
<th>constructor mode</th>
<th>situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 auto</td>
<td>Initialize a new instance to a reference or a user-defined crop. The crop and cultivar are provided by caller; the constructor determines the lengths of the CropCycle.CropGrowthPeriods and automatically adjust them so as to fit into the crop cycle. The constructor fetches $k_y$ for each of the CropCycle.CropGrowthPeriods and CropCycle.SubCGPs, if applicable.</td>
</tr>
<tr>
<td>2 semi-auto</td>
<td>Initialize a new instance to a reference or a user-defined crop. The crop, cultivar and the $k_y$ value for the whole CropCycle are provided by caller; the constructor determines the lengths of the CropCycle.CropGrowthPeriods and automatically adjust them so as to fit into the crop cycle.</td>
</tr>
<tr>
<td>3 semi-auto</td>
<td>Initialize a new instance to a reference or a user-defined crop. The crop, cultivar and the daily $k_y$ values for the whole CropCycle are provided by caller; the constructor determines the lengths of the CropCycle.CropGrowthPeriods and automatically adjust them so as to fit into the crop cycle.</td>
</tr>
<tr>
<td>4 manual</td>
<td>Initialize a new instance to a reference or a user-defined crop. The caller provides all data needed and the constructor accepts the data without doing any adjustment on the CropCycle.CropGrowthPeriods and the corresponding $k_y$ values.</td>
</tr>
<tr>
<td>5 easiest</td>
<td>Initialize a new instance using a WlesDataSet instance. The crop, CropCycle, and CropCycle.CropGrowthPeriods data are encapsulated in the WlesDataSet. The WlesDataSet has its own mechanisms to check the consistency of and make adjustments to the lengths of the CropCycle.CropGrowthPeriods. The call still needs to provide the $k_y$ values.</td>
</tr>
</tbody>
</table>

### 5.13.7 Public Instance Properties

- CGP0BeginningRomanDate
  Gets the beginning date of the establishment period.
  
  ```csharp
  public RomanDate CGP0BeginningRomanDate { get; }
  ```

- CGP0EndingRomanDate
  Gets the ending date of the establishment period.
  
  ```csharp
  public RomanDate CGP0EndingRomanDate { get; }
  ```

- CGP1BeginningRomanDate
  Gets the beginning date of the vegetative period.
  
  ```csharp
  public RomanDate CGP1BeginningRomanDate { get; }
  ```

- CGP1EndingRomanDate
  Gets the ending date of the vegetative period.
  
  ```csharp
  public RomanDate CGP1EndingRomanDate { get; }
  ```

- CGP2BeginningRomanDate
  Gets the beginning date of the flowering period.
  
  ```csharp
  public RomanDate CGP2BeginningRomanDate { get; }
  ```

- CGP2EndingRomanDate
  Gets the ending date of the flowering period.
  
  ```csharp
  public RomanDate CGP2EndingRomanDate { get; }
  ```

- CGP3BeginningRomanDate
  Gets the beginning date of the yield formation period.
  
  ```csharp
  public RomanDate CGP3BeginningRomanDate { get; }
  ```
• CGP3EndingRomanDate
  Gets the ending date of the yield formation period.
  [C#] public RomanDate CGP3EndingRomanDate { get; }

• CGP4BeginningRomanDate
  Gets the beginning date of the ripening period.
  [C#] public RomanDate CGP4BeginningRomanDate { get; }

• CGP4EndingRomanDate
  Gets the ending date of the ripening period.
  [C#] public RomanDate CGP4EndingRomanDate { get; }

• Crop
  Gets the crop name.
  [C#] public string Crop { get; }

• CropCycleBeginningDate
  Gets the beginning date of the CropCycle.
  [C#] public RomanDate CropCycleBeginningDate { get; }

• CropCycleEndingDate
  Gets the ending date of the CropCycle.
  [C#] public RomanDate CropCycleEndingDate { get; }

• Cultivar
  Gets the cultivar name.
  [C#] public string Cultivar { get; }

### 5.13.8 Public Instance Methods

- CGPToString
  Overloaded.

  1. Gets a String representation of all CropCycle.CropGrowthPeriods, formulated in HTML format.
     [C#] public string CGPToString();
     **Example:** The following example shows explicitly the CropCycle.CropGrowthPeriods of soybean in Ngaoundere (Cameroon). Crop cycle starts on Apr. 1 and ends on Aug. 10, which lasts for 130 days.
     
     Crop growth periods:
     - CGP(0): Apr 01 (WLES Calendar) - Apr 11 (WLES Calendar), 11 days [Establishment]
     - CGP(1): Apr 12 (WLES Calendar) - May 18 (WLES Calendar), 37 days [Vegetative]
     - CGP(2): May 19 (WLES Calendar) - Jun 20 (WLES Calendar), 32 days [Flowering]
     - CGP(3): Jun 21 (WLES Calendar) - Jul 27 (WLES Calendar), 37 days [Yield Formation]
     - CGP(4): Jul 28 (WLES Calendar) - Aug 10 (WLES Calendar), 13 days [Ripening]

  2. Gets a String representation of all CropCycle.CropGrowthPeriods, formulated in HTML format.
     [C#] public string CGPToString(string newline, string space);
     **Parameters:**
newLine A `String` that represents the new-line (or carriage return) command in string formatting. The new-line command in GUI or console applications is different from that in Web applications or HTML pages.

`space` A `String` that represents a space. The string representation of space in GUI or console applications is different from that in Web applications or HTML pages.

**Remarks:** The “new line” and “space” commands used in a Web application are different from those in a console application. The `String` representation of those commands are listed below:

- Console application - New line: `\n`; Space: ` `.
- Web application - New line: `<br>`; Space: `&nbsp;`.

- **ExtractData**
  Extracts a subset of data for a particular `CropCycle.CropGrowthPeriod` from a data source that covers the entire `CropCycle`.

[C#] public double[] ExtractData(double[] source, CropGrowthPeriod cgp);

**Parameters:**

*source* A `Double` array that contains daily data during the `CropCycle`.


**Return Value:** A `Double` array that contains daily data for `cgp`, which is a continuous subset of `source`.

**Exceptions:**

- `ArgumentException` Thrown when length of `source` array not equal to that of the `CropCycle`; or `cgp` value not defined.

- **GetCGPDailyArray**
  Gets a daily `integer` array that represents the `CropCycle.CropGrowthPeriods` within an one-year timeframe.

[C#] public int[] GetCGPDailyArray();

**Return Value:** A 360- or 365-element `integer` array that represents the `CropCycle.CropGrowthPeriods`.

**Remarks:**

1. Length of array - The array has the length of 360 or 365 depending on the intrinsic `RomanDate.CalendarType` of the crop cycle. The array elements are organized in the same order as the days in a year. Please note that the array is 0 based, although the year is 1 based.
2. Value of element - There are six values: -1 for days outside the `CropCycle`; 0 for `Establishment` period; 1 for `Vegetative` period; 2 for `Flowering` period; 3 for `YieldFormation` period, and 4 for `Ripening` period.
3. Information conveyed - The following results can be interpreted from the array:
   (a) The beginning and ending dates of the `CropCycle` and its length;
   (b) The beginning and ending dates of the `Establishment` period and its length;
   (c) The beginning and ending dates of the `Vegetative` period and its length;
   (d) The beginning and ending dates of the `Flowering` period and its length;
   (e) The beginning and ending dates of the `YieldFormation` period and its length; and
   (f) The beginning and ending dates of the `Ripening` period and its length.

- **GetCGPDates**
  Overloaded.
1. Gets the beginning and the ending dates of all CropCycle.CropGrowthPeriods in the CropCycle.

[C#]
public RomanDate[][] GetCGPDates();

Return Value: A 5-by-2-element jagged RomanDate array that contains the beginning and the ending dates of all CropCycle.CropGrowthPeriods in the CropCycle.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
</table>
| ArgumentException | Thrown when cgp not equal to one of the following: Establishment, Vegetative, Flowering, YieldFormation and Ripening.

2. Gets the beginning and the ending dates of a particular CropGrowthPeriod.

[C#]
public RomanDate[] GetCGPDates(CropGrowthPeriod cgp);

Return Value: A 2-element RomanDate array that contains the beginning and the ending dates of cgp as its 1st and 2nd elements, respectively.

1. Gets the lengths (days) of all CropCycle.CropGrowthPeriods within the CropCycle.

[C#]
public int[] GetCGPLength();

Return Value: A 5-element integer array that contains the lengths (days) of periods Establishment, Vegetative, Flowering, YieldFormation and Ripening, respectively.

2. Gets the length (days) of a particular CropCycle.CropGrowthPeriod.

[C#]
public int GetCGPLength(CropGrowthPeriod cgp);

Return Value: An integer that represents the length (days) of the CropCycle.CropGrowthPeriod cgp. -1 if cgp not one of the following: Establishment, Vegetative, Flowering, YieldFormation and Ripening.

1. Gets the yield response factor \( (k_y) \) of all CropCycle.CropGrowthPeriods and CropCycle.SubCGPs, if applicable, within the CropCycle.

[C#]
public double[][] GetYieldResponseFactor();

Return Value: A jagged Double array that represents the values of the yield response factor \( (k_y) \) for the CropCycle. The array is arranged in one of the following ways:

(a) One \( k_y \) value for the whole CropCycle - Hence the jagged array contains 1 element which itself is an 1-element Double array:

\[
[[k_y]]
\]

(b) Daily \( k_y \) values for the CropCycle - Two different ways to arrange the array. Suppose \( N \) is the length of the CropCycle and \( i \) the \( i \)th day:

- A 1-element jagged Double array containing a \( N \)-element Double array:

\[
[[k_{y,1}, ..., k_{y,i}, ..., k_{y,N}]]
\]

- A \( N \)-element jagged Double array with each of its element containing an 1-element Double array:

\[
[[k_{y,1}], ..., [k_{y,i}], ..., [k_{y,N}]]
\]

• GetCGPLength
Overloaded.

• GetCropCycleLength
Gets the length (days) of the CropCycle.

[C#]
public int GetCropCycleLength();

• GetYieldResponseFactor
Overloaded.
(c) $k_y$ values for CropCycle.CropGrowthPeriods - A 5-element jagged Double array contains $k_y$ values for the following CropCycle.CropGrowthPeriods: establishment(0), vegetative(1), flowering(2), yield formation(3), and ripening(4). Read on to see the three possible ways to organize $k_y$ values for a CropCycle.CropGrowthPeriod.

There are three ways to organize the $k_y$ values for a CropCycle.CropGrowthPeriod:

(a) One value for the whole period - [...] $[k_y]_{CGP_i}$ ...
(b) Two values for two equal-length sub-periods - [...] $[k_y,a, k_y,b]_{CGP_i}$ ...
(c) Daily values for the period - [...] $[k_y,1, ..., k_y,n]_{CGP_i}$ ...

2. Gets the yield response factor ($k_y$) of a particular CropCycle.CropGrowthPeriod.

[C#] public double[] GetYieldResponseFactor(CropGrowthPeriod cgp);

Return Value: A Double array that represents the yield response factor ($k_y$) of the CropCycle.CropGrowthPeriod cgp:

(a) One value for the whole period - Length of the array equals 1 and the sole element contains the $k_y$ value.
(b) Two values for two equal-length sub-periods - Length of the array equals 2; and the 1st and 2nd elements contain the $k_y$ values for the 1st and 2nd half of the period, respectively.
(c) Daily $k_y$ values are available - Length of the array equals the length (days) of the period.
(d) $k_y$ value unknown - Length of the array equals 1 and the element value is -1.

• ToString
Overloaded.

1. Delivers a CropCycle.CropGrowthPeriod and yield response factor ($k_y$) report in HTML format.

[C#] public override string ToString();

Example: The following example shows explicitly the format of the text report returned by this method.

Crop growth periods:
  CGP(0): Apr 01 (WLES Calendar) - Apr 11 (WLES Calendar),
          11 days [Establishment]
  CGP(1): Apr 12 (WLES Calendar) - May 18 (WLES Calendar),
          37 days [Vegetative]
  CGP(2): May 19 (WLES Calendar) - Jun 20 (WLES Calendar),
          32 days [Flowering]
  CGP(3): Jun 21 (WLES Calendar) - Jul 27 (WLES Calendar),
          37 days [Yield Formation]
  CGP(4): Jul 28 (WLES Calendar) - Aug 10 (WLES Calendar),
          13 days [Ripening]

Yield response factor:
  CGP0: 0.086 0.08 0.075 0.096 0.094 0.089 0.086 0.089 0.098 0.105
       0.13
  CGP1: 0.15(early), 0.25(late)
  CGP2: 0.75
  CGP3: 1
  CGP4: 0.95 0.94 0.93 0.85 0.74 0.60 0.44 0.32 0.30 0.20
       0.15 0.10 0.05

2. Delivers a CropCycle.CropGrowthPeriod and yield response factor ($k_y$) report.

[C#] public string ToString(string newLine, string space);

Parameters:
newLine A String that represents the new-line (or carriage return) command in string formatting. The new-line command in GUI or console applications is different from that in Web applications or HTML pages.

space A String that represents a space. The string representation of space in GUI or console applications is different from that in Web applications or HTML pages.

5.14 YieldResponseToWaterUI Class

5.14.1 Summary
Represents the yield response factor determination and reporting interface of the WLES Web UI.

5.14.2 Declaration
[C#] public class YieldResponseToWaterUI : Page

5.14.3 Remarks
This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

ky.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

5.14.4 Internal Static Methods

- FormYieldResponseString
  Formulates a String representation of the yield response factor evaluation, including the scheme, the CGP, and the resulting $k_y$ values, which are associated with a particular Web Form page.
  
  [C#] internal static string FormYieldResponseString(Page page);
  
  Remarks: The returned String is $k_y$-scheme dependent:
  
  - Scheme “Mono” - Scheme Mono; 0.75 [suppose 0.75 is the $k_y$ value for the crop cycle as a whole]
  - Scheme “Daily” - Scheme Daily; average 0.63 [suppose 0.63 is the average of the daily $k_y$ values of the crop cycle]
  - Scheme “CGP-specific” - Scheme CGP-specific; CGP: 15,25,25,35,20 [suppose “15,25,25,35,20” is the length of the CGPs]

- RetrieveCGP
  Retrieves the length (days) of the CropCycle.CropGrowthPeriods for the crop associated with the current Web Form page.
  
  [C#] internal static int[] RetrieveCGP(Page page);
  
  Return Value: A 5-member integer array containing lengths of the Establishment, Vegetative, Flowering, YieldFormation and Ripening periods; a null reference if data unavailable.

- RetrieveKyScheme
  Retrieves the yield response factor evaluation scheme associated with a particular Web Form page.
  
  [C#] internal static string RetrieveKyScheme(Page page);
  
  Return Value: A String of “mono”, “daily” or “CGP-specific”; an empty String if value not available.
5.15 EffectiveRainfall.UserDefinedMethod Delegate

5.15.1 Summary

Provides an interface for the client to plug in his/her algorithm to estimate the effective rainfall.

5.15.2 Declaration

[C#] public delegate double EffectiveRainfall.UserDefinedMethod(double rain)

5.15.3 Example

Refer to EffectiveRainfall.UserMethodPlugin for an example of how to define and plug in a user-defined method in the client code.

5.16 EffectiveRainfall.Approach Enumeration

5.16.1 Summary

Specifies available methods to estimate the effective rainfall. The WLES implements 4 methods, namely the fixed percentage method, the FAO/AGLW method, the USBR method and the USDA/SCS method. The class also allows a user defined method to be plugged in to estimate the effective rainfall using an ad hoc approach that is well adapted to the local conditions.

5.16.2 Declaration

[C#] public enum EffectiveRainfall.Approach

5.16.3 Members

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FixedPercen</td>
<td>Fixed percentage approach.</td>
</tr>
<tr>
<td>FAO_AGLW</td>
<td>The FAO/AGLW approach.</td>
</tr>
<tr>
<td>USBR</td>
<td>The U.S. Bureau of Reclamation approach.</td>
</tr>
<tr>
<td>USDA_SCS</td>
<td>The USDA Soil Conservation Service approach.</td>
</tr>
<tr>
<td>User</td>
<td>User defined method.</td>
</tr>
</tbody>
</table>

• RetrieveKyValues
  Retrieves the yield response factor \( k_y \) values of the crop associated with a particular Web Form page.

[C#] internal static double[,] RetrieveKyValues(Page page);

Return Value: A jagged Double array (see format and meaning); a null reference if values not available.

• SetupYieldResponseToWater
  Sets up a YieldResponseToWater instance using the CGP and \( k_y \) data associated with a particular Web Form page.

[C#] internal static YieldResponseToWater SetupYieldResponseToWater(Page page);
5.17 WaterBalance.SoilWaterDepletionGroup Enumeration

5.17.1 Summary

Specifies the crop group ID according to the fraction to which the available soil water can be depleted while maintaining $ET_c = ET_m$.

5.17.2 Declaration

[C#] public enum WaterBalance.SoilWaterDepletionGroup

5.17.3 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Group I</td>
</tr>
<tr>
<td>II</td>
<td>Group II</td>
</tr>
<tr>
<td>III</td>
<td>Group III</td>
</tr>
<tr>
<td>IV</td>
<td>Group IV</td>
</tr>
</tbody>
</table>
Chapter 6

Namespace UGent.WLES.LPP

The UGent.WLES.LPP namespace contains all necessary classes and structures for estimating the land production potential (LPP). LPP is the closest approximation of the realistic crop yield. Following the same philosophy that was used to evaluate the WPP from the RPP, the LPP is achieved by computing the soil index, which accounts for the reducing effects of limited soil fertility, and the management index, for the management level in which the production is carried out.

6.1 Overview

6.1.1 Classes

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CropRequirements</td>
<td>Represents the crop requirements with regard to one or more SoilData.SoilCharacteristic(s).</td>
<td>234</td>
</tr>
<tr>
<td>CropRequirementsUI</td>
<td>Represents the crop requirements table building interface of the WLES Web UI.</td>
<td>244</td>
</tr>
<tr>
<td>LPPReport</td>
<td>Represents the UGent.WLES.LPP reporting interface of the WLES Web UI.</td>
<td>245</td>
</tr>
<tr>
<td>ManagementIndex</td>
<td>Represents the management index which is used in conjunction with the soil index to transform the WPP into the LPP.</td>
<td>246</td>
</tr>
<tr>
<td>ManagementIndexUI</td>
<td>Represents the ManagementIndex determination and reporting interface of the WLES Web UI.</td>
<td>249</td>
</tr>
<tr>
<td>SoilIndex</td>
<td>Represents the soil index which is used in conjunction with the management index to transform the WPP into the LPP.</td>
<td>250</td>
</tr>
<tr>
<td>SoilIndexUI</td>
<td>Represents the SoilIndex reporting interface of the WLES Web UI.</td>
<td>255</td>
</tr>
<tr>
<td>Yield</td>
<td>Represents the computation of the LPP, or the final yield of the crop being evaluated in term of kg ha^{-1}.</td>
<td>255</td>
</tr>
</tbody>
</table>

6.1.2 Enumerations

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManagementIndex.InputLevel</td>
<td>Specifies the input levels of field management practices, i.e., low through high.</td>
<td>258</td>
</tr>
<tr>
<td>ManagementIndex.ManagementGroup</td>
<td>Specifies the management group ID of crops, i.e., I through VII.</td>
<td>259</td>
</tr>
<tr>
<td>ManagementIndex.Scheme</td>
<td>Specifies the schemes that are applied to appraise management index.</td>
<td>259</td>
</tr>
<tr>
<td>SoilIndex.Algorithm</td>
<td>Represents the algorithms used to calculate the soil index.</td>
<td>260</td>
</tr>
</tbody>
</table>
6.2 CropRequirements Class

6.2.1 Summary

Represents the crop requirements with regard to one or more SoilData.SoilCharacteristic(s).

6.2.2 Declaration

[C#] public class CropRequirements

6.2.3 Remarks

1. A CropRequirements instance encapsulates both crop (Crop and Cultivar) and soil (Profile) data, and manipulates the crop requirement table and its composing crop requirement strings.

2. The CropRequirements class provides two sets of interfaces to compute a SoilData. SoilCharacteristic's rating value: one is static methods and the other, instance methods. In the latter case, a CropRequirements instance also exposes an additional SoilCharacteristicRating method such that ratings of different SoilData. SoilCharacteristics can be obtained in a uniform way.

3. The CropRequirements class not only rates the SoilData.SoilCharacteristics; it evaluates them as well by assigning SoilData.LandSuitabilityClasses to them. This is done either by the static ConvertRating2Suitability method, or by the instance method of SuitabilityClass.

6.2.4 Example

The following example computes the rating values and evaluates the suitability classes for soil texture, apparent CEC and ESP in an hypothetical site.

Part of the profile description: physical and chemical data

<table>
<thead>
<tr>
<th>Horizon depth(cm)</th>
<th>clay(%)</th>
<th>silt(%)</th>
<th>sand(%)</th>
<th>gravel(%)</th>
<th>CEC</th>
<th>ESP</th>
<th>pH2O</th>
<th>pHKCl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ah</td>
<td>0-20</td>
<td>16</td>
<td>4</td>
<td>80</td>
<td>8</td>
<td>9.7</td>
<td>2.4</td>
<td>6.13</td>
</tr>
<tr>
<td>Bw1</td>
<td>20-55</td>
<td>26</td>
<td>3</td>
<td>71</td>
<td>0</td>
<td>25.3</td>
<td>8.7</td>
<td>5.64</td>
</tr>
<tr>
<td>Bw2</td>
<td>55-90</td>
<td>43</td>
<td>13</td>
<td>44</td>
<td>13</td>
<td>27.2</td>
<td>3.0</td>
<td>5.35</td>
</tr>
<tr>
<td>C</td>
<td>90-150</td>
<td>33</td>
<td>7</td>
<td>60</td>
<td>2</td>
<td>36.5</td>
<td>13.5</td>
<td>5.55</td>
</tr>
</tbody>
</table>

using System;
using UGent.WLES;
using UGent.WLES.Data;
using UGent.WLES.LPP;

public class CropRequirementsExample {
    public static void Main(string[] args) {
        SoilHorizon[] horizons = new SoilHorizon[4];
        horizons[0] = new SoilHorizon("Ah");
        horizons[1] = new SoilHorizon("Bw1");
        horizons[2] = new SoilHorizon("Bw2");
        horizons[3] = new SoilHorizon("C");
        double[] clay = new double[] {16, 26, 43, 33};
        double[] silt = new double[] {4, 3, 13, 7};
        double[] cf = new double[] {8, 0, 0, 0};
double [] cec = new double [] {9.7, 25.3, 27.2, 36.5,};
double [] esp = new double [] {2.4, 8.7, 3.0, 13.5,};
double [] phH2O = new double [] {6.13, 5.64, 5.35, 5.55,};
double [] phKCl = new double [] {4.74, 4.82, 5.47, 5.65,};
double [][] boundaries = new double [][] {
    new double [] {0,.2},
    new double [] {.2,.55},
    new double [] {.55,.9},
    new double [] {.9,1.5},
};
SoilData.GravelType [] types = new SoilData.GravelType [] {
    SoilData.GravelType.Quartz,
    SoilData.GravelType.IronOxide,
    SoilData.GravelType.IronOxide,
    SoilData.GravelType.RockFragments,
};
for (int i=0; i<horizons.Length; i++) {
    horizons[i].ParticleDistribution =
        new double [3] {clay[i], silt[i], 100-clay[i]-silt[i]};
    horizons[i].Depth = boundaries[i];
    horizons[i].Gravel = cf[i];
    horizons[i].GravelType = types[i];
    horizons[i].CEC = cec[i];
    horizons[i].ESP = esp[i];
    horizons[i].pH_H2O = phH2O[i];
    horizons[i].pH_KCl = phKCl[i];
}
SoilProfile profile = new SoilProfile("hypo1", horizons);
string crop = args[0];
CropRequirements requirements = new CropRequirements(crop, profile);
SoilData.SoilCharacteristic [] characteristics =
    new SoilData.SoilCharacteristic [] {
        SoilData.SoilCharacteristic.Texture,
        SoilData.SoilCharacteristic.ACEC,
        SoilData.SoilCharacteristic.ESP,
    };
foreach (SoilData.SoilCharacteristic characteristic
    in characteristics) {
    double rating =
        requirements.SoilCharacteristicRating(characteristic);
    Console.WriteLine("{0}:", characteristic.ToString());
    Console.WriteLine(" rating = {0}, suitability = {1}",
        Math.Round(rating, 2),
        LPP.CropRequirements.ConvertRating2Suitability(rating));
}

Run the example with a command-line argument which specifies the crop name

    CropRequirementsExample alfalfa

will generate the following example:

    Texture:
        rating = 100, suitability = S1
    ACEC:
        rating = 100, suitability = S1
    ESP:
        rating = 90.42, suitability = S1
6.2.5 Public Static Methods

- ACECRating
  Retrieves or interpolates the soil characteristic rating for a particular ACEC value \([\text{cmol}(+) \text{ kg}^{-1} \text{ clay}]\).
  
  \[\text{[C#]}\]
  ```
  public static double ACECRating(string requirement, double acec, bool negativeCharges);
  ```
  
  **Parameters:**
  
  - **requirement** The encoded `String` that represents the ACEC requirement.
  
  See `BuildRequirementTable` for more about the string encoding.
  
  - **acec** A `Double` that represents the ACEC value.
  
  - **negativeCharges** A `Boolean` to indicate whether the soil colloids have negative variable net charge.
    
    This can be determined by the difference between \(\text{pH-H}_2\text{O}\) and \(\text{pH-KCl}\): negative net charge if \((\text{pH-H}_2\text{O} > \text{pH-KCl})\), and positive net charge if \((\text{pH-H}_2\text{O} < \text{pH-KCl})\). (See: `HasNegativeCharges`.)
  
  **Return Value:** A `Double` that represents the soil characteristic rating (dimensionless, \([0, 100]\)) for the given `acec` value.

- BSRating
  Retrieves or interpolates the soil characteristic rating for a particular BS value (%)..
  
  \[\text{[C#]}\]
  ```
  public static double BSRating(string requirement, double bs);
  ```
  
  **Parameters:**
  
  - **requirement** The encoded `String` that represents the BS requirement.
  
  See `BuildRequirementTable` for more about the string encoding.
  
  - **bs** A `Double` that represents the base saturation value \([0, 100]\).
  
  **Return Value:** A `Double` that represents the soil characteristic rating (dimensionless, \([0, 100]\)) for the given `bs` value.

- BuildRequirementTable
  Builds the requirement table for a particular reference `crop` and `cultivar`.
  
  \[\text{[C#]}\]
  ```
  public static string[] BuildRequirementTable(string crop, string cultivar);
  ```
  
  **Return Value:** A 12-element `String` array that contains encoded strings of crop requirements, one string per requirement. Within each string, char ‘|’ is used as the rating delimiter which separates conditions (of ratings); Char ‘&’ is used to delimit multiple sets of rating conditions, and char ‘+’ to combine characteristic values for a single rating; a null reference if not reference `crop`.
  
  **Example:** The following statement returns a string array that represents the encoded requirements table for crop alfalfa:
  
  ```
  UGent.WLES.LPP.CropRequirements.BuildRequirementTable("alfalfa", "");
  ```
  
  The returned 12-element array contains the following strings (serial number added):

  1. SL+SL+SC+L+CL+SiCL|C_60s+Co+SiCs|C_60v+LS+C60s+Lfs|C60v+fS+S+LcS||Cm+SiCm
  2. 0|3|15|35|55||>55
  3. >1|10.7|10.5|10.2|<0.2
  4. 0|6|15|25|35||>35
  5. 0|2|4|10|20||>20
  6. >24|24|16|<16(-)|<16(+)||
  7. >50|50|35|20|<20||
6.2 CropRequirements Class  237

8. >8|8|5|3.5|2|<2|
9. 7.4|7.0|6|5.5|5.2|<5.2|&7.4|7.8|8|8.2|8.5|>8.5
10. >2|2|1.2|0.8|<0.8|
11. 0|3|5|9|12|>12
12. 0|8|20|35|50|>50

• CaCO3Rating
  Retrieves or interpolates the soil characteristic rating for a particular CaCO3 content (%).
  
  ```csharp
  public static double CaCO3Rating(string requirement, double caco3);
  ```

  **Parameters:**
  
  - `requirement` The encoded String that represents the CaCO3 requirement.
    See BuildRequirementTable for more about the string encoding.
  - `caco3` A Double that represents the CaCO3 value ([0, 100]).

  **Return Value:** A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given CaCO3 value.

• CFCorrectionCoef
  Computes a correction coefficient of coarse fragments that can be used to downgrade the texture rating.
  
  ```csharp
  public static double CFCorrectionCoef(GravelType type, double cf);
  ```

  **Parameters:**
  
  - `type` A SoilData.GravelType member.
  - `cf` A Double that represents the coarse fragment content, expressed in percentages, [0, 100].

  **Return Value:** A Double that represents the correction coefficient, expressed in percentages, [0, 100].

  **Exceptions:**
  
  - `ArgumentException` Thrown when `cf` is not between 0 and 100.

• CFRating
  Retrieves or interpolates the soil characteristic rating for a particular coarse fragments content.
  
  ```csharp
  public static double CFRating(string requirement, double cf);
  ```

  **Parameters:**
  
  - `requirement` The encoded String that represents the coarse fragments requirement.
    See BuildRequirementTable for more about the string encoding.
  - `cf` A Double that represents the coarse fragment content, expressed in percentages, [0, 100].

  **Return Value:** A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given `cf` value.

• ConvertRating2Suitability
  Converts a SoilData.SoilCharacteristic’s rating to its suitability class equivalence.
  
  ```csharp
  public static LandSuitabilityClass ConvertRating2Suitability(double rating);
  ```

• ECRating
  Retrieves or interpolates the soil characteristic rating for a particular EC value (dS m⁻¹).
  
  ```csharp
  public static double ECRating(string requirement, double ec);
  ```

  **Parameters:**
  
  - `requirement` The encoded String that represents the EC requirements.
  - `ec` A Double that represents the EC value (dS m⁻¹).
As the encoded String that represents the EC requirement.
See BuildRequirementTable for more about the string encoding.

- **ec**: A value that represents the electrical conductivity value.

**Return Value**: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given ec value.

- **ESPRating**: Retrieves or interpolates the soil characteristic rating for a particular ESP value (%).

```csharp
public static double ESPRating(string requirement, double esp);
```

**Parameters**:

- **requirement**: The encoded String that represents the ESP requirement.
  See BuildRequirementTable for more about the string encoding.
- **esp**: A Double that represents the exchangeable sodium percentage value.

**Return Value**: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given esp value.

- **GypsumRating**: Retrieves or interpolates the soil characteristic rating for a particular Gypsum content (%).

```csharp
public static double GypsumRating(string requirement, double gypsum);
```

**Parameters**:

- **requirement**: The encoded String that represents the gypsum requirement.
  See BuildRequirementTable for more about the string encoding.
- **gypsum**: A Double that represents the Gypsum content.

**Return Value**: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given gypsum value.

- **OCRating**: Interpolates the soil characteristic rating for a particular OC value (%, w/w) with a known/unknown parent material type.

```csharp
public static double OCRating(string requirement, double oc, bool parentMaterialKnown, bool kaolinitic, bool calcareous);
```

**Parameters**:

- **requirement**: The encoded String that represents the O.C requirement.
  See BuildRequirementTable for more about the string encoding.
- **oc**: A Double that represents the OC content.
- **parentMaterialKnown**: true if parent material types are known (i.e., IsParentMaterialKnown == true); false otherwise.
- **kaolinitic**: true for kaolinitic materials; false otherwise.
- **calcareous**: true for calcareous materials; false otherwise.

**Return Value**: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given oc value. **Exceptions**:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when kaolinitic and calcareous are both true.</td>
</tr>
</tbody>
</table>

- **pH_H2ORating**: Retrieves or interpolates the soil characteristic rating for a particular pH_H2O value (dimensionless, neutral at 7).

```csharp
public static double pH_H2ORating(string requirement, double ph);
```

**Parameters**:
6.2 CropRequirements Class

requirement  The encoded String that represents the pH-H$_2$O requirement.
    See BuildRequirementTable for more about the string encoding.

    ph  A Double that represents the pH in water value (dimensionless, neutral at 7).

    Return Value: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given ph value.

• SoilDepthRating
    Retrieves or interpolates the soil characteristic rating for a particular soil depth (m).
    [C#]  public static double SoilDepthRating(string requirement, double depth);
    Parameters:

requirement  The encoded String that represents the soil depth requirement.
    See BuildRequirementTable for more about the string encoding.

    depth  A Double that represents the soil depth (m).

    Return Value: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given depth value.

• SumBCRating
    Retrieves or interpolates the soil characteristic rating for a particular SumBC value [cmol(+) kg$^{-1}$ soil].
    [C#]  public static double SumBCRating(string requirement, double bc);
    Parameters:

requirement  The encoded String that represents the SumBC requirement.
    See BuildRequirementTable for more about the string encoding.

    bc  A Double that represents the sum of basic cations value [cmol(+) kg$^{-1}$ soil].

    Return Value: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given bc value.

• TextureRating
    Retrieves or interpolates the soil characteristic rating for a particular texture class.
    [C#]  public static double TextureRating(string requirement, SoilTextureClass texture);
    Parameters:

requirement  The encoded String that represents the soil texture requirement.
    See BuildRequirementTable for more about the string encoding.

    texture  A SoilData.SoilTextureClass member that represents the considered texture class.

    Return Value: A Double that represents the soil characteristic rating (dimensionless, [0, 100]) for the given texture value.

6.2.6 Public Instance Constructors

• public CropRequirements(string crop, string cultivar, string[] table, SoilProfile profile);
    Initializes a new instance of CropRequirements with the (reference) crop data, the requirements table, and the soil data.
    Parameters:

    crop  A String as the crop name.

    cultivar  A String as the cultivar name.

    table  A String array that represents the crop requirement table, which consists of RequirementStrings of SoilData.SoilCharacteristics in the following order: Texture, CF, SoilDepth, CaC03, Gypsum, ACEC, BS, SumBC, pHH2O, OC, EC, ESP.
profile. A SoilProfile that represents the soil profile associated with this.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string; when profile is a null reference; or when table is a null reference.</td>
</tr>
</tbody>
</table>

- public CropRequirements(string crop, string cultivar, string[] table, SoilProfile profile, double optimalRootingDepth);

  Initializes a new instance of CropRequirements with the crop data, the requirements table, and the soil data.

  **Parameters:**

  - **crop** A String as the crop name.
  - **cultivar** A String as the cultivar name.
  - **table** A String array that represents the crop requirement table, which consists of RequirementStrings of SoilData.SoilCharacteristics in the following order: Texture, CF, SoilDepth, CaCO3, Gypsum, ACEC, BS, SumBC, pH_H2O, OC, EC, ESP.
  - **profile** A SoilProfile that represents the soil profile associated with this.
  - **optimalRootingDepth** A Double that represents the optimal rooting depth (m) of the crop.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string; when profile is a null reference; or when table is a null reference.</td>
</tr>
</tbody>
</table>

- public CropRequirements(string crop, string cultivar, SoilProfile profile);

  Initializes a new instance of CropRequirements with (reference) crop and soil data.

  **Parameters:**

  - **crop** A String as the crop name.
  - **cultivar** A String as the cultivar name.
  - **profile** A SoilProfile that represents the soil profile associated with this.

**Exceptions:**

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when crop is an empty string; when profile is a null reference.</td>
</tr>
</tbody>
</table>

- public CropRequirements(WlesDataSet data);

  Initializes a new instance of CropRequirements with a WlesDataSet.

  **Parameters:**

  - **data** A WlesDataSet that is used to instantiate this.
6.2 CropRequirements Class

6.2.7 Public Instance Properties

- **Crop**
  Returns the crop name.

  ```csharp
  public string Crop { get; }
  ```

- **Cultivar**
  Returns the cultivar name.

  ```csharp
  public string Cultivar { get; }
  ```

- **Profile**
  Returns the `SoilProfile` that associated with *this*.

  ```csharp
  public SoilProfile Profile { get; }
  ```

- **RequirementTable**
  Returns a `String` array that represents the requirement table of the *Crop*.

  ```csharp
  public string[] RequirementTable { get; }
  ```

6.2.8 Public Instance Methods

- **ACECRating**
  Computes the apparent CEC rating for the soil associated with *this*.

  ```csharp
  public double ACECRating()
  ```

  **Return Value**: A `Double` that represents the apparent CEC rating (dimensionless, [0, 100]) of *this* instance.

  **Remarks**: The apparent CEC (or CEC of the clay fraction) value of the B-horizon, or at 50 cm in an A-C profile, is recalculated first; and the sign of variable net charge of the soil colloids are then determined. The rating is lastly obtained by interpolating the ACEC value against the requirement string by calling the static ACECRating method.

- **BSRating**
  Computes the base saturation rating for the soil associated with *this*.

  ```csharp
  public double BSRating();
  ```

  **Return Value**: A `Double` that represents the base saturation rating (dimensionless, [0, 100]) of *this* instance.

  **Remarks**: The weighted mean of the base saturation of the upper 25 cm of the mineral soil is first recalculated without using weighting factors; and then a rating is obtained by interpolating the mean value against the requirement string by calling the static BSRating method.

- **CaCO3Rating**
  Computes the CaCO3 rating for the soil associated with *this*.

  ```csharp
  public double CaCO3Rating();
  ```

  **Return Value**: A `Double` that represents the calcium carbonate rating (dimensionless, [0, 100]) of *this* instance.

  **Remarks**: The weighted mean of calcium carbonate (CaCO3) content of the rooting zone is first recalculated using weighting factors of equal sections; and then a rating is obtained by interpolating the mean value against the requirement string by calling the static CaCO3Rating method.

- **CFRating**
  Computes the coarse fragments rating for the soil associated with *this*.

  ```csharp
  public double CFRating();
  ```

  **Return Value**: A `Double` that represents the gravel content rating (dimensionless, [0, 100]) of *this* instance.

  **Remarks**: The weighted mean of the gravel content of the rooting zone is first recalculated using weighting factors of equal sections; and then a rating is obtained by interpolating the mean value against the requirement string by calling the static CFRating method.
• CombinedFineEarthRating
  Computes the combined fine earth rating for the following three
  SoilData.SoilCharacteristics: texture, coarse fragments and soil depth.

  [C#] public double CombinedFineEarthRating();

  **Return Value:** A Double as the combined rating (dimensionless, [0, 100]).
  **Remarks:** In parametric approach of the crop-specific land evaluation methods, the follow-
  ing three soil characteristics (texture, coarse fragments and soil depth) are evaluated together
  and presented in one combined rating, if:

  1. Soil depth is not optimal, meaning (1) the soil is shallower than the crop roots can go,
     or (2) a root limiting layer presents;

  2. Coarse fragments present.

• ECRating
  Computes the electrical conductivity rating for the soil associated with this.

  [C#] public double ECRating();

  **Return Value:** A Double that represents the electrical conductivity rating (dimensionless,
  [0, 100]) of this instance.
  **Remarks:** The weighted mean of the electrical conductivity of the rooting zone is first
  recalculated using weighting factors of equal sections; and then a rating is obtained by in-
  terpolating the mean value against the requirement string by calling the static ECRating
  method.

• ESPRating
  Computes the exchangeable sodium percentage rating for the soil associated with this.

  [C#] public double ESPRating();

  **Return Value:** A Double that represents the exchangeable sodium percentage rating (dimen-
  sionless, [0, 100]) of this instance.
  **Remarks:** The weighted mean of the exchangeable sodium percentage within 1 meter soil
  depth or within the root limiting layer is first recalculated using weighting factors of equal
  sections; and then a rating is obtained by interpolating the mean value against the require-
  ment string by calling the static ESPRating method.

• GypsumRating
  Computes the Gypsum rating for the soil associated with this.

  [C#] public double GypsumRating();

  **Return Value:** A Double that represents the gypsum rating (dimensionless, [0, 100]) of
  this instance.
  **Remarks:** The weighted mean of gypsum (CaSO$_4$.2H$_2$O) content of the rooting zone is first
  recalculated using weighting factors of equal sections; and then a rating is obtained by inter-
  polating the mean value against the requirement string by calling the static GypsumRating
  method.

• OCRating
  Computes the organic carbon rating for the soil associated with this.

  [C#] public double OCRating();

  **Return Value:** A Double that represents the organic carbon rating (dimensionless, [0, 100])
  of this instance.
  **Remarks:** The weighted mean of the organic carbon content of the upper 25 cm of the
  mineral soil is first recalculated without using weighting factors; and then a rating is obtained
  by interpolating the mean value against the requirement string by calling the static OCRating
  method.

• pH_H2ORating
  Computes the pH-H$_2$O rating for the soil associated with this.

  [C#] public double pH_H2ORating();
**6.2 CropRequirements Class**

**Return Value:** A Double that represents the soil acidity (pH-H$_2$O) rating (dimensionless, [0, 100]) of *this* instance.

**Remarks:**

1. The weighted mean of the pH-H$_2$O value of the upper 25 cm of the mineral soil is first recalculated without using weighting factors; and then a rating is obtained by interpolating the mean value against the requirement string by calling the *static* pH$_2$ORating method.

2. This characteristic is not considered for Low Activity Clay (LAC) soils. LACs are those with ACEC < 24 cmol(+)/kg clay.

- **RequirementString**
  Retrieves the requirement string of a particular SoilData.SoilCharacteristic from the Crop's requirements table.

```csharp
[C#] public string RequirementString(SoilCharacteristic characteristic);
```

- **RootingDepth**
  Gets the Crop's maximum rooting depth (m), taking both optimal rooting depth of the crop and the depth of the soil limiting layer, if present, into account.

```csharp
[C#] public double RootingDepth();
```

**Remarks:** Refer to the GetMaximumRootingDepth method for the relationship between the optimal depth and the depth of the limiting layer.

- **SoilCharacteristicRating**
  Overloaded.

  1. Provides a generalized interface to calculate the rating of a particular SoilData.SoilCharacteristic.

```csharp
[C#] public double SoilCharacteristicRating(SoilCharacteristic characteristic);
```

  2. Provides a generalized interface to calculate the ratings of a group of SoilData.SoilCharacteristics.

```csharp
[C#] public double[] SoilCharacteristicRating(SoilCharacteristic[] characteristics);
```

- **SoilDepthRating**
  Computes the SoilDepth rating for the soil associated with *this*.

```csharp
[C#] public double SoilDepthRating();
```

**Return Value:** A Double that represents the soil depth rating (dimensionless, [0, 100]) of *this* instance.

**Remarks:** The soil depth is first determined by choosing the shallower between the root limiting layer and the depth of the profile; and then a rating is obtained by interpolating this depth against the requirement string by calling the *static* SoilDepthRating method.

- **SuitabilityClass**
  Overloaded.

  1. Provides a generalized interface to evaluate the SoilData.LandSuitabilityClass of a particular SoilData.SoilCharacteristic.

```csharp
[C#] public LandSuitabilityClass SuitabilityClass(SoilCharacteristic characteristic);
```

  2. Provides a generalized interface to evaluate the SoilData.LandSuitabilityClass of a group of SoilData.SoilCharacteristics.
|C#| public LandSuitabilityClass[] SuitabilityClass(SoilCharacteristic[] characteristics);

- SumBCRating
  Computes the sum of basic cations rating for the soil associated with this.
  
  ```c#
  public double SumBCRating();
  ```

  **Return Value**: A `Double` that represents the sum of basic cations rating (dimensionless, [0, 100]) of this instance.

  **Remarks**: The weighted mean of the sum of basic cations of the upper 25 cm of the mineral soil is first recalculated without using weighting factors; and then a rating is obtained by interpolating the mean value against the requirement string by calling the `static SumBCRating` method.

- TextureRating
  Computes the Texture rating for the soil associated with this.

  ```c#
  public double TextureRating();
  ```

  **Return Value**: A `Double` that represents the soil texture rating (dimensionless, [0, 100]) of this instance.

  **Remarks**: The average soil texture of the rooting zone is first recalculated using weighting factors of equal sections; and then a rating is obtained by interpolating this texture against the requirement string by calling the `static TextureRating` method.

### 6.3 CropRequirementsUI Class

#### 6.3.1 Summary

Represents the crop requirements table building interface of the WLES Web UI.

#### 6.3.2 Declaration

```c#
public class CropRequirementsUI : Page
```

#### 6.3.3 Remarks

This page can be accessed with a query string which contains, among others, two *key-value* pairs entitled “rid” and “timeout”, as in the following example:

```text
reqtab.aspx?rid=U0X80F5M0J6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

#### 6.3.4 Internal Static Methods

- RetrieveRequirementTable
  Retrieves the `CropRequirements` table associated with a particular Web Form page.

  ```c#
  internal static string[] RetrieveRequirementTable(Page page);
  ```

  **Return Value**: A 12-element `String` array that represents the `RequirementTable` of the current crop.

  **Remarks**: The `RequirementTable` returned by this method contains 12 requirement strings, among which the first 3 are empty ones:

  1. Texture - empty
  2. CF - empty
  3. SoilDepth - empty
6.4 LPPReport Class

6.4.1 Summary

Represents the UGent.WLES.LPP reporting interface of the WLES Web UI.

6.4.2 Declaration

[C#] public class LPPReport : Page

6.4.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

```
the.aspx?rid=U0X8N0F5MOJ6&timeout=332325051056&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

6.4.4 Internal Static Methods

- SetupLPPYield
  
  Sets up a Yield instance using the data associated with a particular Web Form page.

  [C#] internal static Yield SetupLPPYield(Page page);

  Return Value: A Yield instance that represents the estimated land production potential (kg ha$^{-1}$) value.

  Remarks:

  1. The Yield instance returned by this method represents the determination of the land production potential, i.e., the final yield the WLES projects.
2. The projected UGent.WLES.LPP result can be accessed via the Yield.LPP property. The Yield instance also encapsulates the WPP, the SoilIndex \((S_y)\), the ManagementIndex \((M_y)\) and the yield reducing coefficient due to land limitations \((f_L)\). The following relationships apply:

\[
\begin{align*}
  f_L &= S_y \cdot M_y \\
  LPP &= f_L \cdot WPP
\end{align*}
\]

where: \(f_L\) = yield reducing coefficient due to land limitations; \(S_y\) = soil index; \(M_y\) = management index; \(LPP\) = land production potential; \(WPP\) = water-limited production potential.

### 6.5 ManagementIndex Class

#### 6.5.1 Summary

Represents the management index which is used in conjunction with the soil index to transform the WPP into the LPP.

#### 6.5.2 Declaration

[C#] public class ManagementIndex

#### 6.5.3 Remarks

The ManagementIndex class is used to appraise the field management practices of a crop so that an index can be attributed to them as a whole in terms of impacts of such practices on the final yields. In other words, the management index, or \(M_y\), brings the management factors into the evaluation of the UGent.WLES.LPP.

There are two schemes implemented with the ManagementIndex class: the baseline scheme and the crop-specific scheme. In the baseline scheme, a distinction is made between the leguminous crops and non-leguminous ones; while in the crop-specific scheme, the crops are grouped into 7 management groupings on the similarity of their response to management activities. A management index is then appraised on the basis of input levels which are correlated to the level of management practices.

The management index can be appraised by calling one of the overloads of the GetIndex method. The crop, the input level and the appraising scheme may be passed directly to the static overload. As an alternative, a ManagementIndex instance may be initialized with these data or with an existing WlesDataSet instance. The management index value is immediately available through the Index property, once the ManagementIndex instance is successfully instantiated.

#### 6.5.4 Example

The following examples demonstrate the ways to appraise the management index for the tropical maize under intermediate management level, following the crop-specific appraising scheme.

```csharp
using UGent.WLES;
using UGent.WLES.Data;
using UGent.WLES.LPP;

string crop, cultivar;
crop = "maize";
cultivar = "tropical";

ManagementIndex.InputLevel input = ManagementIndex.InputLevel.Intermediate;
```
ManagementIndex.Scheme scheme = ManagementIndex.Scheme.CropSpecific;

double My;

1. Using the static method call - handy and concise if appraising the $M_y$ alone:

   $M_y = \text{ManagementIndex.GetIndex(crop, cultivar, scheme, input)}$;

2. Using a ManagementIndex instance - fulfilled with an instance. Equivalent to the static method call.

   ManagementIndex Index = new ManagementIndex(crop, cultivar, scheme, input);
   $M_y = \text{Index.Index}$;

3. Using a ManagementIndex instance - preferred way of using the WLES API since the WlesDataSet is the general purpose data interface used to satisfy the data need in the production potentials evaluation process. Seems more lines of coding, but the data are more gracefully taken care by the WlesDataSet whose instance can be used in not only the UGent.WLES.LPP namespace but other ones like UGent.WLES.RPP, UGent.WLES.WPP as well.

   WlesDataSet data = new WlesDataSet();
   data.SetCropData(crop, cultivar);
   data.SetManagementData(scheme, input);
   ManagementIndex Index = new ManagementIndex(data);
   $M_y = \text{Index.Index}$;

### 6.5.5 Public Static Methods

- **GetIndex**
  Overloaded.

  1. Computes the management index (dimensionless, $[0, 1]$) of a particular *crop* and *cultivar* under a given *input* level.

     ```csharp
     public static double GetIndex(string crop, string cultivar, Scheme scheme, InputLevel input);
     ```

     **Parameters:**
     - *crop* A *String* that represents the crop name.
     - *cultivar* A *String* that represents the cultivar name.
     - *scheme* A ManagementIndex.Scheme member that represents the appraising scheme to use.
     - *input* A ManagementIndex.InputLevel member that represents the input level.

     **Exceptions:**
     - *ArgumentOutOfRangeException* Thrown when (1) *crop* is empty or a null reference; or (2) *crop* and/or *cultivar* are/is unknown; in most cases, it is because that *cultivar* is not properly specified; (3) *input* is not a valid member of Input; (4) *scheme* is not a valid member of ManagementIndex.Scheme.

  2. Computes the management index (dimensionless, $[0, 1]$) of a particular *crop* under a given *input* level.

     ```csharp
     public static double GetIndex(string crop, Scheme scheme, InputLevel input);
     ```

     **Parameters:**
crop A `String` that represents the crop name.
scheme A `ManagementIndex.Scheme` member that represents the appraising scheme to use.
input A `ManagementIndex.InputLevel` member that represents the input level.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) <code>crop</code> is empty or a null reference; (2) <code>input</code> is not a valid member of <code>Input</code>; (3) <code>scheme</code> is not a valid member of <code>ManagementIndex.Scheme</code>.</td>
</tr>
</tbody>
</table>

3. Computes the management index (dimensionless, [0, 1]) for a particular `ManagementIndex.ManagementGroup`, under a given `input` level, using a specific appraising `scheme`.

```csharp
public static double GetIndex(Scheme scheme, ManagementGroup group, InputLevel input, bool leguminous);
```

Parameters:

- `scheme` A `ManagementIndex.Scheme` member that represents the appraising scheme to use.
- `group` A `ManagementIndex.ManagementGroup` member that represents the management group ID.
- `input` A `ManagementIndex.InputLevel` member that represents the input level.
- `leguminous` true for a leguminous crop, false otherwise.

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) <code>input</code> is not a valid member of <code>Input</code>; (2) <code>scheme</code> is not a valid member of <code>ManagementIndex.Scheme</code>; (3) <code>group</code> is not a valid member of <code>ManagementIndex.ManagementGroup</code> when <code>scheme</code> takes value <code>CropSpecific</code>.</td>
</tr>
</tbody>
</table>

- GetManagementGroupID
  Overloaded.

1. Gets the management group ID of a particular `crop` and `cultivar`.

```csharp
public static ManagementGroup GetManagementGroupID(string crop);
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when <code>crop</code> is empty or a null reference or unknown.</td>
</tr>
</tbody>
</table>

2. Gets the management group ID of a particular `crop` and `cultivar`.

```csharp
public static ManagementGroup GetManagementGroupID(string crop, string cultivar);
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) <code>crop</code> is empty or a null reference; or (2) <code>crop</code> and/or <code>cultivar</code> are/is unknown; in most cases, it is because that <code>cultivar</code> is not properly specified.</td>
</tr>
</tbody>
</table>

6.5.6 Public Instance Constructors

- `public ManagementIndex(string crop, string cultivar, Scheme scheme, InputLevel input);`
  Initializes a new instance of `ManagementIndex` with the `crop`, the `input` level and the appraising `scheme` to apply.
### Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) <code>crop</code> is empty or a null reference; or (2) <code>input</code> is not a valid member of <code>ManagementIndex.Scheme</code>.</td>
</tr>
</tbody>
</table>

- **public ManagementIndex(string crop, Scheme scheme, InputLevel input);**
  Initializes a new instance of `ManagementIndex` with the `crop`, the `input` level and the appraising `scheme` to apply.
  **Exceptions:**
  
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when (1) <code>crop</code> is empty or a null reference; or (2) <code>input</code> is not a valid member of <code>ManagementIndex.Scheme</code>.</td>
</tr>
</tbody>
</table>

- **public ManagementIndex(WlesDataSet data);**
  Initializes a new instance of `ManagementIndex` with a `WlesDataSet`.

- **public ManagementIndex(Scheme scheme, ManagementGroup group, InputLevel input, bool leguminous);**
  Initializes a new instance of `ManagementIndex` with a particular appraising `scheme`, a management `group`, an `input` level and an indicator determining whether a leguminous crop is being evaluated.

### 6.5.7 Public Instance Properties

- **AppraisingScheme**
  Gets the appraising scheme to apply.
  ```csharp
  public Scheme AppraisingScheme { get; }
  ```

- **Crop**
  Gets the crop name.
  ```csharp
  public string Crop { get; }
  ```

- **Cultivar**
  Gets the cultivar.
  ```csharp
  public string Cultivar { get; }
  ```

- **Index**
  Gets or sets the management index value (dimensionless, [0, 1]).
  ```csharp
  public double Index { get; set; }
  ```

- **Input**
  Gets the input level.
  ```csharp
  public InputLevel Input { get; }
  ```

- **IsLeguminous**
  Determines if the current crop is leguminous.
  ```csharp
  public bool IsLeguminous { get; }
  ```

- **TheGroup**
  Gets the `ManagementIndex.ManagementGroup`.
  ```csharp
  public ManagementGroup TheGroup { get; }
  ```

### 6.6 ManagementIndexUI Class

#### 6.6.1 Summary

Represents the `ManagementIndex` determination and reporting interface of the WLES Web UI.
6.6.2 Declaration

[C#] public class ManagementIndexUI : Page

6.6.3 Remarks

This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

my.aspx?rid=U0X8N0F5M0J6&timeout=332325051506&foo=bar

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

6.6.4 Internal Static Methods

- IsLeguminous
  Determines whether the crop associated with the current Web Form page is leguminous.
  [C#] internal static bool IsLeguminous(Page page);

- RetrieveInputLevel
  Retrieves the ManagementIndex.InputLevel associated with a particular Web Form page.
  [C#] internal static InputLevel RetrieveInputLevel(Page page);

- RetrieveManagementGroup
  Retrieves the ManagementIndex.ManagementGroup that a particular crop associated with a particular Web Form page is in.
  [C#] internal static ManagementGroup RetrieveManagementGroup(Page page);

- RetrieveManagementIndex
  Retrieves the management index associated with a particular Web Form page.
  [C#] internal static double RetrieveManagementIndex(Page page);

- RetrieveManagementIndexScheme
  Retrieve the ManagementIndex appraising ManagementIndex.Scheme associated with a particular Web Form page.
  [C#] internal static Scheme RetrieveManagementIndexScheme(Page page);

- SetupManagementIndex
  Sets up a new ManagementIndex instance and returns it, using data associated with a particular Web Form page.
  [C#] internal static ManagementIndex SetupManagementIndex(Page page);

6.7 SoilIndex Class

6.7.1 Summary

Represents the soil index which is used in conjunction with the management index to transform the WPP into the LPP.

6.7.2 Declaration

[C#] public class SoilIndex
6.7 SoilIndex Class

6.7.3 Remarks

1. The SoilIndex class is used to evaluate the soil index from the soil data and a given set of SoilData.SoilCharacteristics.

2. The soil data are usually represented by a SoilProfile, which in turn contains an array of SoilHorizons. A SoilData.SoilCharacteristic’s value may be explicitly accessed through a SoilProfile; it may also be recalculated by using such methods as RecalParticleDistribution of the SoilProfile class.

3. Ratings of SoilData.SoilCharacteristics are obtained with assistance of the CropRequirements class, which provides two sets of application interfaces (static and instance) to calculate the rating values.

4. The soil index is returned by calling the parameterless GetIndex method.

6.7.4 Example

The following example calculates the soil index for an unknown site with known climatic and soil data.

The climatic data:

Station: unknown (721’N, 1324’E) at 1,414 m a.s.l.

<table>
<thead>
<tr>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sept</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tmax</td>
<td>30.1</td>
<td>31.5</td>
<td>31.2</td>
<td>30.2</td>
<td>28.5</td>
<td>26.6</td>
<td>25.8</td>
<td>26</td>
<td>27</td>
<td>28.3</td>
<td>29.3</td>
</tr>
<tr>
<td>Tmin</td>
<td>12.7</td>
<td>13.5</td>
<td>16.6</td>
<td>17.3</td>
<td>17.1</td>
<td>17</td>
<td>16.8</td>
<td>16.7</td>
<td>15.7</td>
<td>14.2</td>
<td>12</td>
</tr>
<tr>
<td>RH</td>
<td>34.5</td>
<td>35.7</td>
<td>49.4</td>
<td>65.1</td>
<td>75.3</td>
<td>78.1</td>
<td>80.1</td>
<td>81.3</td>
<td>78.7</td>
<td>74.1</td>
<td>59.6</td>
</tr>
<tr>
<td>Windspeed</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sunshine</td>
<td>8.81</td>
<td>8.43</td>
<td>7.01</td>
<td>5.9</td>
<td>5.86</td>
<td>3.75</td>
<td>3.83</td>
<td>4.38</td>
<td>5.63</td>
<td>8.49</td>
<td>9.24</td>
</tr>
<tr>
<td>Rainfall</td>
<td>2</td>
<td>2</td>
<td>49</td>
<td>134</td>
<td>206</td>
<td>237</td>
<td>280</td>
<td>300</td>
<td>235</td>
<td>146</td>
<td>12</td>
</tr>
</tbody>
</table>

Rainfall interval: 3 days

The soil data:

Soil profile: SB31

<table>
<thead>
<tr>
<th>Depth</th>
<th>clay</th>
<th>silt</th>
<th>sand</th>
<th>gravel</th>
<th>pHH2O</th>
<th>pHKCl</th>
<th>O.C</th>
<th>Ca</th>
<th>Mg</th>
<th>K</th>
<th>Na</th>
<th>CEC</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ap</td>
<td>0-24</td>
<td>37</td>
<td>44</td>
<td>19</td>
<td>15</td>
<td>6.13</td>
<td>4.74</td>
<td>1.87</td>
<td>1.75</td>
<td>0.76</td>
<td>0.28</td>
<td>0.06</td>
<td>22.04</td>
</tr>
<tr>
<td>Ah</td>
<td>24-45</td>
<td>65</td>
<td>17</td>
<td>18</td>
<td>25</td>
<td>5.64</td>
<td>4.82</td>
<td>1.23</td>
<td>1.21</td>
<td>1.08</td>
<td>0.09</td>
<td>0.05</td>
<td>12.54</td>
</tr>
<tr>
<td>ABh</td>
<td>45-72</td>
<td>69</td>
<td>16</td>
<td>15</td>
<td>0</td>
<td>5.35</td>
<td>5.47</td>
<td>0.46</td>
<td>1.83</td>
<td>0.55</td>
<td>0.04</td>
<td>0.06</td>
<td>9.62</td>
</tr>
<tr>
<td>Box</td>
<td>72-120</td>
<td>64</td>
<td>27</td>
<td>27</td>
<td>18</td>
<td>5.55</td>
<td>5.65</td>
<td>0.35</td>
<td>1.17</td>
<td>0.12</td>
<td>0.02</td>
<td>0.05</td>
<td>4.38</td>
</tr>
</tbody>
</table>

Parent material: basic rocks

The source code:

```csharp
using System;
using UGent.WLES;
using UGent.WLES.Data;
using UGent.WLES.Topo;
using UGent.WLES.RPP;
using UGent.WLES.WPP;
using UGent.WLES.LPP;
```
public class SoilIndexDemo {
    public static void Main() {
        // climatic data
        string place = "unknown";
        double[] tmax, tmin, RH, wind, n, rain;
        tmax = new double[] {30.1, 31.5, 31.2, 30.2, 28.5, 26.6, 25.8, 26.2, 27.8, 29.3, 29.5, 30.5};
        tmin = new double[] {12.7, 13.5, 16.6, 17.3, 17.1, 17.6, 16.7, 16.5, 17.2, 12.1, 12.1, 12.1};
        RH = new double[] {34.5, 35.7, 49.4, 65.1, 75.3, 78.1, 80.1, 81.3, 78.7, 74.1, 59.6, 43.1};
        wind = new double[] {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
        n = new double[] {8.81, 8.43, 7.01, 5.9, 5.86, 5.02, 3.75, 3.83, 4.38, 5.63, 8.49, 9.24};
        rain = new double[] {2, 2, 49, 134, 206, 237, 280, 300, 235, 146, 12, 2};
        int rainfall_interval = 3;
        double latitude = 7 + 21.0 / 60.0;
        double longitude = 13 + 34.0 / 60.0;
        double elevation = 1014;
        // soil data
        string soilname = "SB31";
        double[][] depth, psd, ph, cations;
        double[] gravel, oc, cec, bs;
        depth = new double[][] {
            new double[] {0, .24}, new double[] {.24, .45},
            new double[] {.45, .72}, new double[] {.72, 1.2},
        };
        psd = new double[][] {
            new double[] {37, 44, 19}, new double[] {65, 17, 18},
            new double[] {69, 16, 15}, new double[] {64, 27, 27},
        };
        ph = new double[][] {
            new double[] {6.13, 4.74}, new double[] {5.64, 4.82},
            new double[] {5.35, 5.47}, new double[] {5.55, 5.65},
        };
        cations = new double[][] {
            new double[] {1.75, 0.76, 0.28, 0.06},
            new double[] {1.21, 1.08, 0.09, 0.05},
            new double[] {1.83, 0.55, 0.04, 0.06},
            new double[] {1.17, 0.12, 0.02, 0.05},
        };
        gravel = new double[] {15, 25, 0, 18};
        oc = new double[] {1.87, 1.23, 0.46, 0.35};
        cec = new double[] {22.04, 12.54, 9.62, 4.38};
        bs = new double[] {13, 19, 26, 31};
        string[] ids = new string[] {"Ap", "Ah", "ABh", "Box"};
        SoilHorizon[] horizons = new SoilHorizon[ids.Length];
        for (int i = 0; i < horizons.Length; i++) {
            horizons[i] = new SoilHorizon(ids[i]);
            horizons[i].Depth = depth[i];
            horizons[i].ParticleDistribution = psd[i];
            horizons[i].pH_H2O = ph[i][0];
            horizons[i].pH_KCl = ph[i][1];
            horizons[i].Ca = cations[i][0];
6.7 SoilIndex Class

```csharp
horizons[i].Mg = cations[i][1];
horizons[i].K = cations[i][2];
horizons[i].Na = cations[i][3];
horizons[i].Gravel = gravel[i];
horizons[i].OC = oc[i];
horizons[i].CEC = cec[i];
horizons[i].BS = bs[i];
}
WlesDataSet data = new WlesDataSet();
data.SetLocationalData(place, latitude, longitude, elevation);
data.SetClimaticData(tmax, tmin, RH, wind, n, rain);
string crop = "soybean", cultivar = ""
;data.SetCropData(crop, cultivar, CropData.GetLeafAreaIndex(crop),
    CropData.GetHarvestIndex(crop, cultivar),
    CropData.GetOptimalRootingDepth(crop));
SoilIndex.Algorithm algorithm = SoilIndex.Algorithm.Storie;
SoilProfile profile = new SoilProfile(soilname, horizons);
data.SetSoilData(profile, algorithm);
SoilIndex Sy = new SoilIndex(data);
Console.WriteLine("Soil index = {0}", Math.Round(Sy.GetIndex(), 2));
```

The example generates the following result:

Soil index = 0.27

### 6.7.5 Public Instance Constructors

1. `public SoilIndex(string crop, SoilProfile profile, bool humid);`
   Initializes a new instance of SoilIndex with the `crop` name, the `SoilProfile`, the climate classifier, the default set of `SoilData.SoilCharacteristics` and the default `Storie` algorithm.

2. `public SoilIndex(string crop, SoilProfile profile, bool humid, Algorithm algorithm);`
   Initializes a new instance of SoilIndex with the `crop` name, the `SoilProfile`, the climate classifier, the default set of `SoilData.SoilCharacteristics` and a chosen `algorithm`.

3. `public SoilIndex(string crop, SoilProfile profile, SoilCharacteristic[] characteristics);`
   Initializes a new instance of SoilIndex with the `crop` name, the `SoilProfile`, a selected set of `SoilData.SoilCharacteristics` and the default `Storie` algorithm.

4. `public SoilIndex(string crop, SoilProfile profile, SoilCharacteristic[] characteristics, Algorithm algorithm);`
   Initializes a new instance of SoilIndex with the `crop` name, the `SoilProfile`, a selected set of `SoilData.SoilCharacteristics` and a chosen `algorithm`.

5. `public SoilIndex(WlesDataSet data);`
   Initializes a new instance of SoilIndex with a `WlesDataSet` and the default set of `SoilData.SoilCharacteristics`.

6. `public SoilIndex(WlesDataSet data, SoilCharacteristic[] characteristics);`
   Initializes a new instance of SoilIndex with a `WlesDataSet` and a selected set of `SoilData.SoilCharacteristics`. 
7. public SoilIndex(CropRequirements requirements, bool humid);
   Initializes a new instance of SoilIndex with the CropRequirements, the humid-climate indicator, the default set of SoilData.SoilCharacteristics and the default Storie algorithm.

8. public SoilIndex(CropRequirements requirements, bool humid, Algorithm algorithm);
   Initializes a new instance of SoilIndex with the CropRequirements, the humid-climate indicator, the default set of SoilData.SoilCharacteristics and a chosen algorithm.

9. public SoilIndex(CropRequirements requirements, SoilCharacteristic[] characteristics);
   Initializes a new instance of SoilIndex with the CropRequirements, a selected set of SoilData.SoilCharacteristics and the default Storie algorithm.

10. public SoilIndex(CropRequirements requirements, SoilCharacteristic[] characteristics, Algorithm algorithm);
    Initializes a new instance of SoilIndex with the CropRequirements, a selected set of SoilData.SoilCharacteristics and a chosen algorithm.

Remarks:

1. To calculate the soil index, all SoilData.SoilCharacteristics that have not been taken into account for the UGent.WLES.WPP and that have a direct impact on yield need to be evaluated. The selection has to be based on the local production conditions. Nevertheless, there is no much difference in the final set of SoilData.SoilCharacteristics.

2. A default set of SoilData.SoilCharacteristics are selected in WLES either for a humid climate or for an arid one.

3. For humid climates or soils with fertility problems:
   - ACEC
   - OC
   - The most limiting of pH_H2O and SumBC

4. For arid climates or soils with salinity problems:
   - CaCO3
   - Gypsum
   - The most limiting of EC and ESP

6.7.6 Public Instance Properties

- EffectiveAlgorithm
  The SoilIndex.Algorithm that is used in soil index determination.
[C#] public Algorithm EffectiveAlgorithm { get; }

- IsHumidClimate
  Determines whether the climate is humid.
[C#] public bool IsHumidClimate { get; }

Remarks:

1. The climatic data are used to determine whether a climate is humid by comparing the magnitudes of the rainfall and half of the reference evapotranspiration.

2. The WlesDataSet.IsHumidClimate method does exactly this job.
6.8 SoilIndexUI Class

6.8.1 Summary
Represents the SoilIndex reporting interface of the WLES Web UI.

6.8.2 Declaration
[C#] public class SoilIndexUI : Page

6.8.3 Remarks
This page can be accessed with a query string which contains, among others, two key-value pairs entitled “rid” and “timeout”, as in the following example:

```
sy.aspx?rid=U0XN50M0J6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

6.8.4 Internal Static Methods
- RetrieveSoilIndexMethod
  Retrieves the SoilIndex.Algorithm used to calculate the soil index ($S_y$).
  [C#] internal static Algorithm RetrieveSoilIndexMethod(Page page);

- SetupSoilIndex
  Sets up a SoilIndex instance using data associated with a particular Web Form page.
  [C#] internal static SoilIndex SetupSoilIndex(Page page);

6.9 Yield Class

6.9.1 Summary
Represents the computation of the LPP, or the final yield of the crop being evaluated in term of kg ha$^{-1}$.

6.9.2 Declaration
[C#] public class Yield
6.9.3 Remarks

1. It is the last step in LPP evaluation to compute the yield. Although the computation itself is a trivial task, the advantage is obvious to represent it by a class, i.e., the exceptional values can be properly handled such that the application is either more robust or easier to use.

2. Given the \( WPP \) (kg ha\(^{-1}\), obtained via the \texttt{WaterBalance.WPP} property after the \texttt{WaterBalance.Simulate} and the \texttt{WaterBalance.Evaluate} methods called), the soil index \( S_Y \) (obtained via the \texttt{SoilIndex.GetIndex} method) and the management index \( M_Y \) (obtained via the \texttt{ManagementIndex.GetIndex} method), there are two ways to evaluate the LPP yield:

   (a) By accessing the \texttt{Yield.LPP} property - the yield is immediately available once a \texttt{Yield} instance is successfully initialized;

   (b) By calling the static \texttt{GetLPP} method - The method returns the yield with exceptions properly handled as well.

3. To evaluate the aggregate impacts of the soil and management limitations, the soil index \( S_Y \) and the management index \( M_Y \) are combined to form the yield reducing coefficient due to land limitations \( f_L \). See equations 6.1 and 6.2.

4. A report of LPP evaluation is delivered by calling the \texttt{ToString} method. Example to how the format of the report:

\[
S_Y = 0.2, \ M_Y = 0.65, \ f_L = 0.13, \ LPP = 863.63 \ \text{kg/ha}
\]

6.9.4 Public Static Methods

- **GetLPP**
  Computes the land production potential (LPP, kg ha\(^{-1}\)).
  
  \texttt{[C#] public static double GetLPP(double wpp, double sy, double my);} \\
  \texttt{Parameters:}
  
  \texttt{ wpp}  The water-limited production potential (WPP) in kg ha\(^{-1}\).
  
  \texttt{ sy}  The soil index \( S_Y \), dimensionless, \([0, 1]\).
  
  \texttt{ my}  The management index \( M_Y \), dimensionless, \([0, 1]\).
  
  \texttt{Return Value:} A \texttt{Double} that represents the LPP (kg ha\(^{-1}\)). \texttt{Exceptions:}
  
  \texttt{ArgumentException}  Thrown when \texttt{wpp} has a negative value; or either \texttt{sy} or \texttt{my}'s value is out of range \([0, 1]\).

6.9.5 Public Instance Constructors

- **Initializes a new instance of Yield.**
  
  \texttt{[C#] public Yield(double wpp, double sy, double my);} \\
  \texttt{Parameters:}
  
  \texttt{ wpp}  The water-limited production potential (WPP) in kg ha\(^{-1}\).
  
  \texttt{ sy}  The soil index \( S_Y \), dimensionless, \([0, 1]\).
  
  \texttt{ my}  The management index \( M_Y \), dimensionless, \([0, 1]\).
  
  \texttt{Exceptions:}
  
  \texttt{ArgumentException}  Thrown when \texttt{wpp} has a negative value; or either \texttt{sy} or \texttt{my}'s value is out of range \([0, 1]\).
6.9 Yield Class

- Initializes a new instance of Yield with a WaterBalance (provide that WPP has been evaluated) instance, a SoilIndex instance and a ManagementIndex instance.

```csharp
public Yield(WaterBalance wb, SoilIndex si, ManagementIndex mi);
```

**Parameters:**
- `wb` The WaterBalance instance.
- `si` The SoilIndex instance.
- `mi` The ManagementIndex instance.

### 6.9.6 Public Instance Properties

- **LPP**
  The land production potential (LPP) in kg ha\(^{-1}\).

```csharp
public double LPP { get; }
```

- **ManagementIndex**
  The management index \((M_y, \text{dimensionless}, [0, 1])\).

```csharp
public double ManagementIndex { get; }
```

- **ManagementIndexInstance**
  The ManagementIndex instance associated with the current instance.

```csharp
public ManagementIndex ManagementIndexInstance { get; }
```

- **SoilIndex**
  The soil index \((S_y, \text{dimensionless}, [0, 1])\).

```csharp
public double SoilIndex { get; }
```

- **SoilIndexInstance**
  The SoilIndex instance associated with the current instance.

```csharp
public SoilIndex SoilIndexInstance { get; }
```

- **WaterBalanceInstance**
  The WaterBalance instance associated with the current instance.

```csharp
public WaterBalance WaterBalanceInstance { get; }
```

- **WPP**
  The water-limited production potential (WPP) in kg ha\(^{-1}\).

```csharp
public double WPP { get; }
```

- **YieldReducingCoefLand**
  The yield reducing coefficient due to land limitations \((f_L, \text{dimensionless}, [0, 1])\).

```csharp
public double YieldReducingCoefLand { get; }
```

**Remarks:** To evaluate the aggregated impacts of the soil and management limitations, the soil index \((S_y)\) and the management index \((M_y)\) are combined to form this new coefficient \(f_L\) that reveals the magnitude by which the LPP is reduced from WPP. See equations 6.1 and 6.2 (p.246).

### 6.9.7 Public Instance Methods

- **ToString**
  Delivers a land production potential evaluation results report that contains the soil index \((S_y)\), the management index \((M_y)\), yield reducing coefficient due to land limitations \((f_L)\) and the LPP.

**Example:** The following example shows explicitly the format of the report of LPP evaluation for maize plantation in an hypothetical tropical location:

\[ S_y = 0.2, M_y = 0.65, f_L = 0.13, \text{LPP} = 863.63 \text{ kg/ha} \]
6.10 ManagementIndex.InputLevel Enumeration

6.10.1 Summary
Specifies the input levels of field management practices, i.e., low through high.

6.10.2 Declaration
[C#] public enum ManagementIndex.InputLevel

6.10.3 Remarks
Definition of levels of inputs (after FAO, 1982):

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Input Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>low</td>
</tr>
<tr>
<td>Production system</td>
<td>Rainfed cultivation of presently grown mixture of crops</td>
</tr>
<tr>
<td>Technology employed</td>
<td>Local cultivars. No fertilizer or chemical pest, disease and weed control. Fallow periods. No soil conservation practices</td>
</tr>
<tr>
<td>Labor intensity</td>
<td>High, essentially family labor</td>
</tr>
<tr>
<td>Ecosystem management</td>
<td>Nil</td>
</tr>
<tr>
<td>Capital intensity</td>
<td>Very low to low</td>
</tr>
<tr>
<td>Market orientation</td>
<td>Subsistence production</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Family based</td>
</tr>
<tr>
<td>Technology availability</td>
<td>Usually none</td>
</tr>
<tr>
<td>Land holdings</td>
<td>Fragmented</td>
</tr>
<tr>
<td>Land titles</td>
<td>Societal (usually no legal)</td>
</tr>
</tbody>
</table>

6.10.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>The low input level, which assumes no fertilizer and pesticide applications, no soil conservation measures but land labour, with productivity losses arising from land degradation.</td>
</tr>
<tr>
<td>Intermediate</td>
<td>The intermediate input level, which assumes use of improves hand tools and/or draught implements, some fertilizer and pesticide application, some simple soil conservation measures lessening productivity losses from land degradation.</td>
</tr>
</tbody>
</table>
6.11 ManagementIndex.ManagementGroup Enumeration

6.11.1 Summary
Specifies the management group ID of crops, i.e., I through VII.

6.11.2 Declaration

[C#] public enum ManagementIndex.ManagementGroup

6.11.3 Remarks
The ManagementIndex.ManagementGroup enumeration represents the group IDs to which crops are classified on the basis of yield responses to the field management practices. Such practices are further categorized into three input levels, namely, high, intermediate and low.

This enumeration is useful when it is desirable to have a strongly typed specification of the crop groupings on the management basis. For example, this enumeration is among the formal parameters of the GetManagementGroupID method.

Call the GetManagementGroupID method to determine the group ID to which a crop belongs.

6.11.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Group I</td>
</tr>
<tr>
<td>II</td>
<td>Group II</td>
</tr>
<tr>
<td>III</td>
<td>Group III</td>
</tr>
<tr>
<td>IV</td>
<td>Group IV</td>
</tr>
<tr>
<td>V</td>
<td>Group V</td>
</tr>
<tr>
<td>VI</td>
<td>Group VI</td>
</tr>
<tr>
<td>VII</td>
<td>Group VII</td>
</tr>
</tbody>
</table>

6.12 ManagementIndex.Scheme Enumeration

6.12.1 Summary
Specifies the schemes that are applied to appraise management index.

6.12.2 Declaration

[C#] public enum ManagementIndex.Scheme

6.12.3 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>The baseline scheme. A distinction is made between the leguminous crops and non-leguminous ones. The management index is then appraised on the basis of input levels.</td>
</tr>
<tr>
<td>CropSpecific</td>
<td>The crop-specific scheme. The crops are classified into management groups. The management index is then appraised on the basis of input levels.</td>
</tr>
</tbody>
</table>
6.13 SoilIndex.Algorithm Enumeration

6.13.1 Summary

Represents the algorithms used to calculate the soil index.

6.13.2 Declaration

[C#] public enum SoilIndex.Algorithm

6.13.3 Remarks

The Storie (1976) and the Square Root (Khiddir, 1986) Methods are modified and represented here. The original methods were designed to compute the land index as a value in range [0, 100]; however, the WLES requires the land index to be a fractional value in range [0, 1] which can be directly applied to reduce the WPP.

Suppose \( n \) SoilData.SoilCharacteristics are chosen for the calculation of the soil index \( S_y \):

1. Storie Method - The product of all the selected ratings gives the soil index.

\[
S_y = \prod_{i=1}^{n} \left( \frac{R_i}{100} \right)
\]

where: \( S_y \) = soil index, \([0, 1]\); \( R_i \) = rating of the \( i^{th} \) SoilData.SoilCharacteristic, expressed as a value in range \([0, 100]\).

2. Square Root Method - The product of the minimum rating and the square-root of the product of all the rest ratings gives the soil index.

\[
S_y = \frac{R_{\text{min}}}{100} \cdot \sqrt[100]{\prod_{i=1}^{n-1} \left( \frac{R_i}{100} \right)}
\]

where: \( S_y \) = soil index, \([0, 1]\); \( R_{\text{min}} \) = the minimum rating of all the SoilData.SoilCharacteristics, expressed as a value in range \([0, 100]\); \( R_i \) = rating of the \( i^{th} \) (out of \( n - 1 \)) SoilData.SoilCharacteristic, expressed as a value in range \([0, 100]\).

6.13.4 Members

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storie</td>
<td>The Storie Method.</td>
</tr>
<tr>
<td>SquareRoot</td>
<td>The Square Root Method.</td>
</tr>
</tbody>
</table>
Chapter 7

Namespace UGent.WLES.Users

The UGent.WLES.Users namespace represents the users of the WLES. Users are grouped in the WLES as subscribed users, guest users and anonymous users. The effective security policy requires that users have to be authenticated before services can be accessed. A hierarchical content control is applied in WLES: whereas subscribed users are eligible to access any content, guest users are turned away from the Batch Evaluation Service; Although anonymous users may have access to the Help and Documentation Subsystems, they are not eligible to any evaluation services. Registered users and the guest user are all viewed as authenticated users. There are two ways for an anonymous user to become authenticated: one is to register though the user registration interface and the other is to login as a guest. The UGent.WLES.Users namespace wraps such classes as Register, Login, Account and Logout, and all relevant WebControls and HtmlControls which form the Web UI of the WLES user subsystem.

7.1 Overview

7.1.1 Classes

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>Represents a user account. The class wraps all properties a user account has and provides methods to determine whether the account represented by the current instance exists in the WLES database; to read an existing account from database, and to save the current account to the database.</td>
<td>262</td>
</tr>
<tr>
<td>AuthCheck</td>
<td>Determination of user authentication status. It distinguishes three categories of users (i.e., the anonymous users, the guest users and registered users) and redirects the user to the corresponding image file.</td>
<td>266</td>
</tr>
<tr>
<td>Comments</td>
<td>Represents the suggestions and comments interface of the WLES Web UI.</td>
<td>266</td>
</tr>
<tr>
<td>CountryFlag</td>
<td>Represents the Web client’s country flag, returned as an image file.</td>
<td>266</td>
</tr>
<tr>
<td>CustomImage</td>
<td>Represents a customized image generator which receives data via the query string and sends the generated Gif image back through its response.</td>
<td>267</td>
</tr>
<tr>
<td>Filter</td>
<td>Represents the user filtering interface of the WLES Web UI.</td>
<td>269</td>
</tr>
<tr>
<td>GuestLogin</td>
<td>Represents the guest user login interface of the WLES Web UI.</td>
<td>270</td>
</tr>
<tr>
<td>ImageBroker</td>
<td>Represents an image broker which receives data via the query string and sends the generated Gif image back through its response.</td>
<td>270</td>
</tr>
<tr>
<td>Login</td>
<td>Represents the user login interface of the WLES Web UI.</td>
<td>272</td>
</tr>
<tr>
<td>type</td>
<td>summary</td>
<td>page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Logout</td>
<td>Represents the user logout page. It signs the logged user out and removes the WLES cookie that the DoLogin method has added, and redirects the request to a destination which is specified in the query string under the name “redirect” or “back”.</td>
<td>272</td>
</tr>
<tr>
<td>MyAccount</td>
<td>Represents the user account maintenance interface of the WLES Web UI.</td>
<td>273</td>
</tr>
<tr>
<td>MyProfile</td>
<td>Represents the user profile report interface of the WLES Web UI.</td>
<td>273</td>
</tr>
<tr>
<td>Password</td>
<td>Represents the user password retrieval interface of the WLES Web UI.</td>
<td>273</td>
</tr>
<tr>
<td>Register</td>
<td>Represents the user registration interface of the WLES Web UI.</td>
<td>274</td>
</tr>
</tbody>
</table>

### 7.2 Account Class

#### 7.2.1 Summary

Represents a user account. The class wraps all properties a user account has and provides methods to determine whether the account represented by the current instance exists in the WLES database; to read an existing account from database, and to save the current account to the database.

#### 7.2.2 Declaration

```csharp
public class Account
```

#### 7.2.3 Public Static Methods

- **Exists**
  - Overloaded.
    - Determines whether an account with a particular `Username` already exists in the WLES database.
      ```csharp
      public static bool Exists(string uid);
      ```

- **GenerateUserName**
  - Generates the username from the user’s first and last names.
    ```csharp
    public static string GenerateUserName(string fname, string lname, ref string err);
    ```
    **Return Value:** The username generated from `fname` and `lname`; empty string ("") if error.
    **Remarks:**
    - The rules used to generate the username:
      1. `fname = fname.Trim().ToLower(); lname = lname.Trim().ToLower();`
      2. Remove all special `Chars` defined in `NameRule` from `fname` and `lname;`
      3. `username = fname[0] + lname;`
    - The error message generated is passed back by reference. The relationship between the error message and the return username is summarized below.
    | returned username | passed back error message |
    |-------------------|---------------------------|
    | non-empty string, username ok | empty string (""), no error occurred |
    | empty string (""), no username returned | non-empty string, error occurred |

- **UpdateDB**
  - Updates a particular `field` in the database record of the account with a particular `uid` to a given new value.
    ```csharp
    public static int UpdateDB(string field, string uid, string newval);
    ```
    **Return Value:** An `integer` as the number of database records affected.
7.2 Account Class

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WesDatabaseException</td>
<td>Thrown when database exception occurs.</td>
</tr>
</tbody>
</table>

7.2.4 Public Instance Constructors

- **public Account();**
  Initializes a new instance of Account. Instance properties must be set manually elsewhere, either by setting the **Username** and calling the ReadDB method, or by setting all properties explicitly.

- **public Account(string **username**);**
  Initializes a new instance of Account with the **username**. This is usually used to read an existing account.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the user account does not exist.</td>
</tr>
</tbody>
</table>

- **public Account(string **fname**, string **lname**);**
  Initializes a new instance of Account with the first and last names of the user. This is usually used to read an existing account.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when the user account does not exist, and when the user account cannot be solely determined using the first and last names. Use the username (i.e., UID) instead.</td>
</tr>
</tbody>
</table>

- **public Account(string **fname**, string **lname**, string **question**, string **answer**);**
  Initializes a new instance of Account with the first and last names of the user. This is usually used to read an existing account.

  Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesException</td>
<td>Thrown when the user account does not exist, or the user account cannot be solely determined using the first and last names; and when the question is not predefined in PwdQuestions. Use overloading constructors instead.</td>
</tr>
</tbody>
</table>

  Initializes a new instance of Account by providing complete account data. This is usually used to create a new account.

7.2.5 Public Instance Properties

- **Answer**
  Gets or sets the password answer.

  [C#] public string Answer { get; set; }

- **Country**
  Gets or sets the user’s country of origin.

  [C#] public string Country { get; set; }

- **Email**
  Gets or sets the email address of the user.

  [C#] public string Email { get; set; }
• Fax
  Gets or sets the fax number of the user.
  
  ```csharp
  [C#] public string Fax { get; set; }
  ```

• Firm
  Gets or sets the affiliation of the user.
  
  ```csharp
  [C#] public string Firm { get; set; }
  ```

• FirstName
  Gets or sets the first name of the user.
  
  ```csharp
  [C#] public string FirstName { get; set; }
  ```

• FullName
  Gets the full name of the user.
  
  ```csharp
  [C#] public string FullName { get; }
  ```

• Homepage
  Gets or sets the homepage of the user.
  
  ```csharp
  [C#] public string Homepage { get; set; }
  ```

• LastName
  Gets or sets the first name of the user.
  
  ```csharp
  [C#] public string LastName { get; set; }
  ```

• Password
  Gets or sets the password phrase.
  
  ```csharp
  [C#] public string Password { get; set; }
  ```

• Phone
  Gets or sets the telephone number of the user.
  
  ```csharp
  [C#] public string Phone { get; set; }
  ```

• PostalAddress
  Gets or sets the postal address of the user.
  
  ```csharp
  [C#] public string PostalAddress { get; set; }
  ```

• Question
  Gets or sets the secret question.
  
  ```csharp
  [C#] public string Question { get; set; }
  ```

• Title
  Gets or sets the title of the user.
  
  ```csharp
  [C#] public string Title { get; set; }
  ```

• Username
  Gets or sets the username (or in other words, user ID).
  
  ```csharp
  [C#] public string Username { get; set; }
  ```

### 7.2.6 Public Instance Methods

• Exists
  Overloaded.
  
  - Determines whether the account represented by the current instance already exists in the WLES database.
    
    ```csharp
    [C#] public bool Exists();
    ```
7.2 Account Class

- **FlushToDB**
  Saves the account data into the WLES database. Database operation may be update or insert (See: Remarks).
  ```csharp
  [C#] public int FlushToDB();
  ```
  **Return Value:** An integer as the number of database records affected.
  **Remarks:** This method calls Exists to determine if a record with the same UID already exists in the database. The existing record will be updated if the testing returns true; a new record will be inserted if false.
  **Exceptions:**
<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WlesDatabaseException</td>
<td>Thrown when database exception occurs.</td>
</tr>
</tbody>
</table>

- **GetProfileReport**
  Overloaded.
  1. Generates an HTML-encoded profile report for the current user, which contains the complete account data, with title and password included by default.
     ```csharp
     [C#] public string GetProfileReport();
     ```
     **Example:** The following example shows explicitly the format of an hypothetical user profile.
     ```
     Profile Report
     Username : foo.bar
     Password : fake_pwd
     Name in full : Dr. Foo Bar
     Organization : Pioneer Soft
     Country : Germany
     Email : fbar@pioneersoft.com
     Secret question : Name of my secondary school?
     Secret answer : Wisba 2nd High
     Phone : +49 87 985 6325
     Fax : +49 87 985 6326
     Homepage : http://www.pioneersoft.com/fbar
     Postal address : Pioneer Park 1, 60254 Aachen, Germany
     ```
  2. Generates an HTML-encoded profile report for the current user, which contains the complete account data.
     ```csharp
     [C#] public string GetProfileReport(bool header, bool pwd);
     ```
     **Parameters:**
     - header true to include the single line report title, namely, “Profile Report”; false not to include.
     - pwd true to show the password as plain text; false to hide it (as a security-enhancing measure). Instead, the user may have his password sent to him (or her) by email.

- **ReadDB**
  Overloaded.
  1. Reads account data from the WLES database and resets all instance properties.
     ```csharp
     [C#] public void ReadDB(string username);
     ```
     **Exceptions:**
     | exception         | description                      |
     |-------------------|----------------------------------|
     | ArgumentException | Thrown when this record does not exist. |
  2. Reads account data from the WLES database and resets all instance properties.
     ```csharp
     [C#] public void ReadDB(string fname, string lname);
     ```
Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when this record does not exist and when more than one records exist.</td>
</tr>
</tbody>
</table>

3. Reads account data from the WLES database and resets all instance properties.

```csharp
public void ReadDB(string fname, string lname, string question, string answer);
```

Exceptions:

<table>
<thead>
<tr>
<th>exception</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when this record does not exist and when more than one records exist.</td>
</tr>
<tr>
<td>WlesException</td>
<td>Thrown when question is not one of the predefined by PwdQuestions.</td>
</tr>
</tbody>
</table>

- SendProfileByEmail
  Sends an HTML-encoded mail message, that contains the user profile report of the current Account instance, to the user’s email address.

```csharp
public string SendProfileByEmail(bool isNewUser);
```

7.3 AuthCheck Class

7.3.1 Summary

Determination of user authentication status. It distinguishes three categories of users (i.e., the anonymous users, the guest users and registered users) and redirects the user to the corresponding image file.

7.3.2 Declaration

```csharp
public class AuthCheck : Page
```

7.4 Comments Class

7.4.1 Summary

Represents the suggestions and comments interface of the WLES Web UI.

7.4.2 Declaration

```csharp
public class Comments : Page
```

7.4.3 Remarks

1. This class wraps the WebControls and HtmlControls that constitutes the MyAccount Web UI, and their event handlers;

2. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.

7.5 CountryFlag Class

7.5.1 Summary

Represents the Web client’s country flag, returned as an image file.
**7.6 CustomImage Class**

**7.6.1 Summary**
Represents a customized image generator which receives data via the query string and sends the generated Gif image back through its response.

**7.6.2 Declaration**
[C#] public class CustomImage : Page

**7.6.3 Remarks**
1. This page should be accessed with a query string which contains, among others, the following keys:

   - **image** carries the data series to plot. One data series is separated from the other by the “|” sign, and data items in a series are separated by the commas (“,”):
     
     \[
     \text{image} = \{\text{series}_1\}|...|\text{series}_i|...|\text{series}_n
     \]
     
     where each series can be defined as:
     
     \[
     \text{series} = \text{type}, \text{color}, \text{width}, \text{isclosed}, \text{fillcolor}, \text{point}_1, ..., \text{point}_i, ..., \text{point}_n
     \]

   - **type** = line or curve
   - **color** = blue, red, black, etc. ¹/²
   - **width** = a Single value, for instance: 1, 1.5, 2, etc.
   - **point** = \(x, y\)
   - **isclosed** = 0 or 1, where 0 means false and 1 means true ²/³
   - **fillcolor** = the Color to fill the closed area; ignored if isclosed key has value 0. Use named colors such as blue, red, black, etc. ¹/²

¹/²/³notation is as per the context provided in the text.
Note

1/ If the color is given by a named Color instance, a String representation of it is accessible by the Name property. The same Color instance can NOT be reconstructed by using the String returned by the Color.ToString method.

2/ If value is 0, the fillcolor key will be ignored; If value is 1 and type is “line”, a polygon is to be drawn; if type is “curve”, a closed curve is to be drawn.

The following example specifies 2 data series to draw, one drawn as a blue curve of 5 points; the other as a red line of 7 points:

image = curve,blue,2,0,,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8|line,red,2,0,,7,7,6,6,5,5,4,4,3,3,2,2,1,1

bg assigns the background Color (default value of WhiteSmoke is taken if this key is missing);

Note If the color is given by a named Color instance, a String representation of it is accessible by the Name property. The same Color instance can NOT be reconstructed by using the String returned by the Color.ToString method.

size specifies the width and height of the resulting image in pixels (default value of 800x600 is taken if this key is missing);

xrange specifies the value range (Single) of the X-axis; consisting of a minimum and a maximum value separated by a comma ("."). The minimum and the maximum X-values of the data series are used if this key is missing;

yrange specifies the value range (Single) of the Y-axis; consisting of a minimum and a maximum value separated by a comma ("."). The minimum and the maximum Y-values of the data series are used if this key is missing.

xlabels provides labelling texts for the tic marks of the X-axis, separated by the comma (",") sign. The side effect is that the number of tic marks of the X-axis will be set to the number of labels contained in this key. If missing, the following labels will be used instead: J,F,M,A,M,J,J,Jun,Jul,Aug,Sep,Oct,Nov,Dec (when image width < 500 pixels) or Jan,Feb,Mar,Apr,May, Jun,Jul,Aug,Sept,Oct,Nov,Dec (when image width ≥ 500 pixels);

ylabels provides labelling texts for the tic marks of the Y-axis, separated by the comma (",") sign;

gridlines determines whether to draw grid-lines: value 1 to draw; value 0 not;

gridcolor specifies the Color of the grid-lines.

Note If the color is given by a named Color instance, a String representation of it is accessible by the Name property. The same Color instance can NOT be reconstructed by using the String returned by the Color.ToString method.

2. The image key is mandatory and the rest are optional.

3. A query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign.

4. A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

7.6.4 Example

The following example plots 2 crossing lines on a 400x300 canvas, each consisting of 5 points:

customimage.aspx ? image = line,red,1,0,pad,1.1,1.1,2.2,2.2,3.3,3.3,4.4,4.4,5.5,5.5,6.6,6.6,7.7,7.7,8.8,8.8&
yrange = 0,10 & bg = white & size = 400x300 & xlabels = J,F,M,A,M & gridcolor = darkgray
7.7 Filter Class

Detailed specification of the image generated:

- **Image size**: 400 x 300
- **Background**: Color.White
- **X-range**: auto setting
- **Y-range**: [0, 10]
- **X-axis**: 5 tic marks, labelled with "J", "F", "M", "A" and "M", respectively
- **Y-axis**: auto labelling (2 tic marks, labelled with "0" and "10")
- **Grid-lines**: draw in DarkGray
- **Line 1**: Red, 1 pixel width
  - (1.1, 1.1), (2.2, 2.2), (3.3, 3.3), (4.4, 4.4), (5.5, 5.5)
- **Line 2**: Blue, 2 pixels width
  - (5.1, 5.1), (4.2, 4.2), (3.3, 3.3), (2.4, 2.4), (1.5, 1.5)

The following example draws a closed curved area of red inside a blue rectangular frame on a default 800x600 canvas:

```
customimage.aspx ? image = line,blue,5,1,2.3,3.6,2.3,100.25,58.6,100.25,58.6,3.6|curve,navy,1,1,red,10,20,20,80,50,60,40,30,20,10 & yrange = 0,120 & xrange = 0,90 & gridlines = 1
```

Detailed specification of the image generated:

- **Image size**: 800 x 600
- **Background**: Color.WhiteSmoke
- **X-range**: [0, 90]
- **Y-range**: [0, 120]
- **Y-axis**: auto labelling (6 tic marks, labelled with equal-distance figures "0", "24", "48", ..., "120")
- **Grid-lines**: draw in DarkGray
- **Area 1**: Blue, 5 pixel width rectangle, transparent (i.e., not filled with a color) (2.3,3.6), (2.3,100.25), (58.6,100.25), (58.6,3.6)
- **Area 2**: Navy-outlined Red closed curve
  - (10,20), (20,80), (50,60), (40,30), (20,10)

7.7 Filter Class

7.7.1 Summary

Represents the user filtering interface of the WLES Web UI.

7.7.2 Declaration

[C#]
```
public class Filter : Page
```

7.7.3 Remarks

1. This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:
the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. The logic behind this example is that the user is about to authenticate himself or herself before to write some commenting words via the “Suggestions and Comments” page;

2. The value of the “redirect” query string can either be one of the InternalLink members of the Logistics class or any legal url String;

3. A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method;

4. This class wraps the WebControls and HtmlControls that constitutes the Login Web UI, and their event handlers;

5. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.

7.8 GuestLogin Class

7.8.1 Summary

Represents the guest user login interface of the WLES Web UI.

7.8.2 Declaration

[C#] public class GuestLogin : Page

7.8.3 Remarks

This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:

guestlogin.aspx?redirect=interactive&foo=bar

the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. The logic behind this query string is that the user will be automatically Redirected to the “interactive” page after the user is logged-in as “guest”. A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

7.9 ImageBroker Class

7.9.1 Summary

Represents an image broker which receives data via the query string and sends the generated Gif image back through its response.

7.9.2 Declaration

[C#] public class ImageBroker : Page
7.9.3 Remarks

1. This page should be accessed with a query string which contains, among others, the following keys:

   - **data** carries the data series to plot. One data series is separated from the other by the “|” sign, and data items in a series are separated by the commas (",”);
   - **bg** assigns the background Color (default value of WhiteSmoke is taken if this key is missing);
   - **colors** stands for the Colors in which the data series are plotted (as curves), separated by the comma (",”) sign;
   - **widths** represents the widths (pixels) of the curves, also separated by the comma (",”) sign;
   - **size** specifies the width and height of the resulting image in pixels (default value of 800x600 is taken if this key is missing);
   - **xlabels** provides labelling texts for the tic marks of the X-axis, separated by the comma (",”) sign. The side effect is that the number of tic marks of the X-axis will be set to the number of labels contained in this key. If missing, the following labels will be used instead: J,F,M,A,M,J,J,A,SA,M,Jun,Jul,Aug,Sept,Oct,Nov,Dec (when image width < 500 pixels) or Jan,Feb,Mar,Apr,May, Jun,Jul,Aug,Sept,Oct,Nov,Dec (when image width ⩾ 500 pixels);
   - **ylabels** provides labelling texts for the tic marks of the Y-axis, separated by the comma (",”) sign;
   - **gridlines** determines whether to draw grid-lines: value 1 to draw; value 0 not;
   - **gridcolor** specifies the Color of the grid-lines.

   **Note** If the color is given by a named Color instance, a String representation of it is accessible by the Name property. The same Color instance can NOT be reconstructed by using the String returned by the Color.ToString method.

2. The keys of **data**, **colors** and **widths** are mandatory and the rest are optional.

3. Curves do not necessarily consist of the same number of points, but the number of curves must agree with the number of colors and widths. Otherwise no image will be passed back.

4. A query string begins after the “,” sign, and the key-value pairs are separated by the “&” sign. The logic behind this example is that the user passes 2 series of points data to the imagebroker.aspx page and expects a 800 x 600 pixel Gif image to be passed back as the HttpResponse.

5. A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

7.9.4 Example

The following example plots 2 curves on a 400x300 canvas, each consisting of 5 points:

```
imagebroker.aspx?data=1.1,2.2,3.3,4.4,5.5|5.1,4.2,3.3,2.4,1.5&
bg=white&colors=red,blue&widths=1,2&size=400x300&xlabels=
J,F,M,A,M&gridcolor=darkgray
```

Detailed specification of the image generated:
Image size: 400 x 300
Background: Color.White
X-axis: 5 tic marks, labelled with "J", "F", "M" "A" and "M", respectively
Y-axis: auto labelling (2 tic marks, labelled with "1.1" and "5.5")
Grid-lines: draw in DarkGray
Curve 1:
Red, 1 pixel width
(1, 1.1), (2, 2.2), (3, 3.3), (4, 4.4), (5, 5.5)
Curve 2:
Blue, 2 pixels width
(1, 5.1), (2, 4.2), (3, 3.3), (4, 2.4), (5, 1.5)

7.10 Login Class

7.10.1 Summary
Represents the user login interface of the WLES Web UI.

7.10.2 Declaration
[C#] public class Login : Page

7.10.3 Remarks
1. This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:

   login.aspx?redirect=comments&foo=bar

   the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. The logic behind this example is that the user is about to authenticate himself or herself before to write some commenting words via the “Suggestions and Comments” page;

2. The value of the “redirect” query string can either be one of the InternalLink members of the Logistics class or any legal url String;

3. A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method;

4. This class wraps the WebControls and HtmlControls that constitutes the Login Web UI, and their event handlers;

5. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.

7.11 Logout Class

7.11.1 Summary
Represents the user logout page. It signs the logged user out and removes the WLES cookie that the DoLogin method has added, and redirects the request to a destination which is specified in the query string under the name “redirect” or “back”.

7.11.2 Declaration

[C#] public class Logout : Page

7.11.3 Remarks
The name of WLES cookie is defined in the authentication section of the application-wide Web.config file. The name is accessible via the FormsCookieName property.

7.12 MyAccount Class

7.12.1 Summary
Represents the user account maintenance interface of the WLES Web UI.

7.12.2 Declaration
[C#] public class MyAccount : Page

7.12.3 Remarks
1. This class wraps the WebControls and HtmlControls that constitutes the MyAccount Web UI, and their event handlers;

2. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.

7.13 MyProfile Class

7.13.1 Summary
Represents the user profile report interface of the WLES Web UI.

7.13.2 Declaration
[C#] public class MyProfile : Page

7.13.3 Remarks
1. This class wraps the WebControls and HtmlControls that constitutes the MyProfile Web UI, and their event handlers;

2. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.

7.14 Password Class

7.14.1 Summary
Represents the user password retrieval interface of the WLES Web UI.

7.14.2 Declaration
[C#] public class Password : Page
7.14.3 Remarks
1. This class wraps the WebControls and HtmlControls that constitutes the Password Web UI, and their event handlers;
2. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.

7.15 Register Class

7.15.1 Summary
Represents the user registration interface of the WLES Web UI.

7.15.2 Declaration
[C#] public class Register : Page

7.15.3 Remarks
1. This class wraps the WebControls and HtmlControls that constitutes the Register Web UI, and their event handlers;
2. Fundamental functionalities behind the Web UI are mostly provided by the Logistics class.
Chapter 8

Namespace UGent.WLES.Help

The UGent.WLES.Help namespace represents the Web UI of the WLES help subsystem. Topics is the only class implemented in the namespace. The class wraps relevant WebControls and HtmlControls which are building blocks of the help subsystem Web UI.

8.1 Overview

8.1.1 Classes

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topics</td>
<td>Represents the topic-specific help contents management in either context-sensitive or context-free styles.</td>
<td>275</td>
</tr>
</tbody>
</table>

8.2 Topics Class

8.2.1 Summary

Represents the topic-specific help contents management in either context-sensitive or context-free styles.

8.2.2 Declaration

[C#] public class Topics : Page

8.2.3 Remarks

This this the help contents handling infrastructure embedded inside the WLES user interface. The help contents are stored separately in the WLES database. Newly added topics in the database table are shown automatically on the user interface.

This page can be accessed with a query string which contains any combination of the following two keys: “topic” and “back”. In the following example:

help.aspx?topic=register&back=register&foo=bar

the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. The logic behind this example is that the WLES Help Subsystem is referred to from within the user registration page and thus the registration-related help topic is displayed in a context sensitive manner. A “Go Back” hyperlink is provided within the help page so that the user can be easily brought back to its previous page, which is in this case the user registration page.

A help topic is chosen in one of the following two ways:
1. **Context-sensitive** - The help subsystem is invoked from embedded hyperlinks within the WLES application. The “topic” is submitted to the help subsystem as a “key=value” pair added to the HTTP query string;

2. **Context-free** - It does not matter whether the help subsystem is invoked with or without a query string. The initial topic (or subsequent topics) must be chosen by the user manually.
Chapter 9

Namespace
UGent.WLES.Interactive

The UGent.WLES.Interactive namespace contains all necessary classes and structures for guiding the users to fulfil their evaluation tasks in the interactive mode. An interactive evaluation is made of sequential evaluation phases which cover such tasks as data collection, parameter ratification, report generation, result interpretation and exportation, and so on.

9.1 Overview

9.1.1 Classes

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClimaticDataUI</td>
<td>Represents the climatic data input interface of the WLES Web UI.</td>
<td>277</td>
</tr>
<tr>
<td>CropDataUI</td>
<td>Represents the crop data input interface of the WLES Web UI.</td>
<td>279</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Represents the starting point of the interactive evaluation of the WLES Web UI.</td>
<td>281</td>
</tr>
<tr>
<td>SoilCharSet</td>
<td>Represents the SoilData.SoilCharacteristics selecting interface of the WLES Web UI.</td>
<td>281</td>
</tr>
<tr>
<td>SoilDataUI1</td>
<td>Represents the soil data (part 1) input interface of the WLES Web UI.</td>
<td>282</td>
</tr>
<tr>
<td>SoilDataUI2</td>
<td>Represents the soil data (part 2) input interface of the WLES Web UI.</td>
<td>284</td>
</tr>
<tr>
<td>StudyArea</td>
<td>Represents the study area input interface of the WLES Web UI.</td>
<td>286</td>
</tr>
</tbody>
</table>

9.2 ClimaticDataUI Class

9.2.1 Summary

Represents the climatic data input interface of the WLES Web UI.

9.2.2 Declaration

[C#] public class ClimaticDataUI : Page

9.2.3 Remarks

This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:

climaticdata.aspx?rid=U0X8N0F5MOJ6&timeout=332325051506&foo=bar
A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

9.2.4 Internal Static Methods

- **CalculateETo**
  Calculates the year-round reference evapotranspiration using climatic parameters associated with a particular Web Form `page`.

  ```csharp
  internal static double[] CalculateETo(Page page);
  ```

- **IsHumidClimate**
  Determines whether the climate associated with a particular Web Form `page` is humid.

  ```csharp
  internal static bool IsHumidClimate(Page page);
  ```

- **RetrieveDataResolution**
  Retrieves the `WlesDataSet.TemporalResolution` of the climatic data which are associated with the current `page` from the WLES database.

  ```csharp
  internal static TemporalResolution RetrieveDataResolution(Page page);
  ```

  **Return Value:** The `WlesDataSet.TemporalResolution` of the climatic data; -1 if the current `page` is not RID-authenticated.

- **RetrieveDelimiters**
  Retrieves the delimiting chars used to separate the climatic data which are associated with the current `page` from the WLES database.

  ```csharp
  internal static char[] RetrieveDelimiters(Page page);
  ```

  **Return Value:** A `Char` array containing delimiters; a null reference if the current `page` is not RID-authenticated.

- **RetrieveRainfall**
  Retrieves the time series of the rainfall (mm) data which are associated with the current `page` from the WLES database.

  ```csharp
  internal static double[] RetrieveRainfall(Page page);
  ```

  **Return Value:** A `Double` array. Array length equals the ordinal value represented by the temporal resolution; a null reference if the current `page` is not RID-authenticated.

- **RetrieveRH**
  Retrieves the time series of the relative air humidity (%) data which are associated with the current `page` from the WLES database.

  ```csharp
  internal static double[] RetrieveRH(Page page);
  ```

  **Return Value:** A `Double` array. Array length equals the ordinal value represented by the temporal resolution; a null reference if the current `page` is not RID-authenticated.

- **RetrieveSunshineDuration**
  Retrieves the time series of the sunshine duration (hours) data which are associated with the current `page` from the WLES database.

  ```csharp
  internal static double[] RetrieveSunshineDuration(Page page);
  ```

  **Return Value:** A `Double` array. Array length equals the ordinal value represented by the temporal resolution; a null reference if the current `page` is not RID-authenticated.

- **RetrieveTemperatureThreshold**
  Retrieves the temperature threshold (°C) which is associated with the current `page` from the WLES database.

  ```csharp
  internal static double RetrieveTemperatureThreshold(Page page);
  ```

  **Return Value:** An `Int32` as number of days; -1 if the current `page` is not RID-authenticated.

  **Remarks:** Crops stop growing when temperature decreases below the threshold. The default value is 6.5 °C. The threshold may be set to another value.
9.3 CropDataUI Class

9.3.1 Summary
Represents the crop data input interface of the WLES Web UI.

9.3.2 Declaration

[C#] public class CropDataUI : Page

9.3.3 Remarks
This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:

cropdata.aspx?rid=UOX8N0F5M0J6&timeout=332325051506&foo=bar

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.
9.3.4 Internal Static Methods

- CalculatePercentageCCinGP
  Calculates the percentage of the number of days of the CropCycle that falls inside the GrowingPeriod over the length of the CropCycle.

  ```csharp
  internal static double CalculatePercentageCCinGP(Page page);
  ```

- FormCropCycleString
  Forms a String that represents the CropCycle associated with a particular Web Form page.

  ```csharp
  internal static string FormCropCycleString(Page page);
  ```

  **Return Value:** A String with this format:

  "Mon day - Mon day, length days [CDS: cds1,cds2,cds3,cds4]"

  **Example:** The following example shows explicitly the format the returned String.

  Sep 20 - Jan 14, 117 days [CDS: 22,37,37,21]

- FormCropString
  Forms a String that represents the crop associated with a particular Web Form page.

  ```csharp
  internal static string FormCropString(Page page);
  ```

  **Return Value:** A String with this format: "crop (alias), cultivar cultivar".

  **Example:** The following example shows explicitly the format the returned String.

  maize (corn), tropical cultivar

- IsLeguminous
  Determines whether the crop associated with the current Web Form page is leguminous.

  ```csharp
  internal static bool IsLeguminous(Page page);
  ```

- RetrieveCDS
  Retrieves the length (days) of the CropCycle.CropDevelopmentStages for the crop associated with the current Web Form page.

  ```csharp
  internal static int[] RetrieveCDS(Page page);
  ```

  **Return Value:** A 4-member Int32 array containing lengths of the IS, CD, MS and LS stages.

- RetrieveCropCultivarAlias
  Retrieves the crop name, cultivar and alias data which are associated with a particular Web Form page.

  ```csharp
  internal static string[] RetrieveCropCultivarAlias(Page page);
  ```

  **Return Value:** A 3-member String array containing crop, cultivar and alias as its 1st, 2nd and 3rd member; a null reference if data not available.

- RetrieveCropCycleRomanDates
  Retrieves the beginning and ending RomanDates of the crop associated with the current Web Form page.

  ```csharp
  internal static RomanDate[] RetrieveCropCycleRomanDates(Page page);
  ```

- RetrieveHarvestIndex
  Retrieves the harvest index of the crop associated with the current Web Form page.

  ```csharp
  internal static double RetrieveHarvestIndex(Page page);
  ```

- RetrieveLAIMax
  Retrieves the leaf area index at maximum growth rate of the crop associated with the current Web Form page.

  ```csharp
  internal static double RetrieveLAIMax(Page page);
  ```
9.4 Evaluation Class

9.4.1 Summary

Represents the starting point of the interactive evaluation of the WLES Web UI.

9.4.2 Declaration

[C#] public class Evaluation : Page

9.5 SoilCharSet Class

9.5.1 Summary

Represents the SoilData.SoilCharacteristics selecting interface of the WLES Web UI.

9.5.2 Declaration

[C#] public class SoilCharSet : Page

9.5.3 Remarks

This page can be accessed with a query string which contains at least one key-value pair entitled "redirect". In the following example:

```
soilchar.aspx?rid=U0X8N0F5MOJ6&timeout=332325051506&foo=bar
```

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.
9.5.4 Internal Static Methods

- **FormSelectedSoilCharacteristicsString**
  Forms a **String** that represents the selected soil characteristics associated with a particular Web Form **page**.

  ```csharp
  [C#] internal static string FormSelectedSoilCharacteristicsString(Page page);
  Return Value: A **String** of comma-separated **SoilData.SoilCharacteristic** names; empty **String** if the selection is empty.
  ```

- **IsHumidClimate**
  Determines whether the climate associated with a particular Web Form **page** is humid.

  ```csharp
  [C#] internal static bool IsHumidClimate(Page page);
  Return Value: **true** if rainfall exceeds half of the reference evapotranspiration; **false** otherwise.
  ```

- **RetrieveSelectedSoilCharacteristics**
  Retrieves the selected **SoilData.SoilCharacteristic**s on which the **SoilIndex** is to be based.

  ```csharp
  [C#] internal static SoilCharacteristic[] RetrieveSelectedSoilCharacteristics(Page page);
  ```

9.6 SoilDataUI1 Class

9.6.1 Summary

Represents the soil data (part 1) input interface of the WLES Web UI.

9.6.2 Declaration

```csharp
[C#] public class SoilDataUI1 : Page
``` 

9.6.3 Remarks

This page can be accessed with a query string which contains at least one key-value pair entitled “**redirect**”. In the following example:

```text
soildata1.aspx?rid=20050101ABCDEF&foo=bar
```

the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. A new interactive evaluation is initialized if the “rid” query string does not present or is empty; if it presents with a non-empty value, an existing evaluation is to be revisited.

A query string may be generated automatically when an HTML FORM is submitted using the HTTP **GET** method.

9.6.4 Internal Static Methods

- **FormParentMaterialString**
  Forms a **String** that represents the **ParentMaterial** of the soil associated with a particular Web Form **page**.

  ```csharp
  [C#] internal static string FormParentMaterialString(Page page);
  Return Value: A **String** as in this example: Calcareous material “SE-65”.
  ```

- **FormSoilString**
  Forms a **String** that represents the soil associated with a particular Web Form **page**.

  ```csharp
  [C#] internal static string FormSoilString(Page page);
  Return Value: A **String** with this format:
  ```
9.6 SoilDataUI1 Class

profile "ProfileNumber", horizons NumberOfHorizons, depth SoilDepth m.

**Example**: The following example shows explicitly the format the returned String.

profile "spf15", horizons 4, depth 1.25 m

- **IsHorizonIDValid**
  Determines whether a particular soil horizon ID is valid.

  ```csharp
  internal static bool IsHorizonIDValid(string id, ref string msg);
  ```

  **Remarks**: A valid ID should obey the following rules:
  
  1. It is a non-empty *String*;
  2. It consists of letters and/or digits; It may contain one or more special chars defined by
     the same name entry as HorizonIDRule in the Web.config file;
  3. It starts with a letter or digit;
  4. It can be as short as 1 and as long as 10 chars.

- **IsProfileNumberValid**
  Determines whether a particular soil profile number is valid.

  ```csharp
  internal static bool IsProfileNumberValid(ref string nr, ref string msg);
  ```

  **Parameters**:

  - *nr* The soil profile number to be checked. This parameter will be passed back by *ref* with
    leading and tailing special chars trimmed.
  - *msg* A multi-line *String* passed back by *ref* as the error message generated during the
    evaluation, should the method returns *false*.

  **Remarks**: A valid ProfileNumber should obey the following rules:
  
  1. It is a non-empty *String*;
  2. It consists of letters and/or digits; It may contain one or more special chars defined by
     the same name entry as ProfileNrRule in the Web.config file;
  3. It starts and ends with a letter or digit;
  4. It can be as short as 2 and as long as 20 chars.

- **RetrieveHorizonDepths**
  Retrieves the (upper and lower) boundary depths of all horizons in the soil profile associated
  with a particular Web Form *page*.

  ```csharp
  internal static double[,] RetrieveHorizonDepths(Page page);
  ```

  **Return Value**: A 2-dimensional *Double* array that contains the horizon-specific boundary
  depths of a *SoilProfile*. Length of the 1<sup>st</sup> dimension equals the number of horizons
  and length of the 2<sup>nd</sup> dimension equals 2. A null reference if the horizons data not
  available.

- **RetrieveNumberOfHorizons**
  Retrieves the number of *SoilHorizons* which consists of the *SoilProfile* associated with a
  particular Web Form *page*.

  ```csharp
  internal static int RetrieveNumberOfHorizons(Page page);
  ```

  **Return Value**: The number of soil horizons included with the soil data; -1 if soil data is
  not available yet.

- **RetrieveParticleSizeDistribution**
  Retrieves the particle size distribution of all horizons in the *SoilProfile* associated with a
  particular Web Form *page*.

  ```csharp
  internal static double[,] RetrieveParticleSizeDistribution(Page page);
  ```

  **Return Value**: A 2-dimensional *Double* array that contains the horizon-specific clay, silt
  and sand contents of a *SoilProfile*. Length of the 1<sup>st</sup> dimension equals the number of
  horizons and length of the 2<sup>nd</sup> dimension equals 3. A null reference if the horizons data not
  available.
Namespace UGent.WLES.Interactive

- **RetrieveProfileDepth**
  Retrieves the depth (m) of the SoilProfile associated with a particular Web Form page.
  
  ```c#```
  internal static double RetrieveProfileDepth(Page page);
  ```
  Return Value: The depth of the soil profile; -1 if soil data is not available yet.

- **RetrieveProfileNumber**
  Retrieves the ProfileNumber of the SoilProfile associated with a particular Web Form page.
  
  ```c#```
  internal static string RetrieveProfileNumber(Page page);
  ```
  Return Value: The soil profile number included with the soil data; empty string if soil data is not available yet.

- **RetrieveRootLimitingLayerDepth**
  Retrieves the depth (m) of the root limiting layer associated with a particular Web Form page.
  
  ```c#```
  internal static double RetrieveRootLimitingLayerDepth(Page page);
  ```
  Return Value: The depth of the root limiting layer; -1 if not present.

- **RetrieveSandClass**
  Retrieves the SoilData.SandClass associated with a particular Web Form page.
  
  ```c#```
  internal static SandClass RetrieveSandClass(Page page);
  ```
  Return Value: A SoilData.SandClass member; -1 if sand class unknown.

- **RetrieveSoilStructure**
  Retrieves the SoilData.SoilStructure associated with a particular Web Form page.
  
  ```c#```
  internal static SoilStructure RetrieveSoilStructure(Page page);
  ```
  Return Value: A SoilData.SoilStructure member; -1 if structure unknown.

- **SetupSoilProfile**
  Sets up a SoilProfile instance using the soil data associated with a particular Web Form page.
  
  ```c#```
  internal static SoilProfile SetupSoilProfile(Page page);
  ```
  Return Value: A SoilProfile; a null reference if data not available or incomplete.

Remarks:
Difference between this method and the same name method defined in SoilDataUI2:
1. The SoilDataUI2.SetupSoilProfile method is implemented on top of this method;
2. This method assigns values of the physical properties and general descriptions to the profile, while the SoilDataUI2.SetupSoilProfile method further adds chemical properties values.

### 9.7 SoilDataUI2 Class

#### 9.7.1 Summary

Represents the soil data (part 2) input interface of the WLES Web UI.

#### 9.7.2 Declaration

```c#```
public class SoilDataUI2 : Page
```

#### 9.7.3 Remarks

This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:

soildata2.aspx?rid=20050101ABCDEF&foo=bar
the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. A new interactive evaluation is initialized if the “rid” query string does not present or is empty; if it presents with a non-empty value, an existing evaluation is to be revisited.

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.

### 9.7.4 Internal Static Methods

- **FormParentMaterialString**
  Forms a `String` that represents the `ParentMaterial` of the soil associated with a particular Web Form page.
  ```csharp
  internal static string FormParentMaterialString(Page page);
  ```
  **Return Value:** A `String` as in this example: Calcareous material "SE-65".

- **FormSoilString**
  Forms a `String` that represents the soil associated with a particular Web Form page.
  ```csharp
  internal static string FormSoilString(Page page);
  ```
  **Return Value:** A `String` with this format:
  ```
  profile "ProfileNumber", horizons NumberOfHorizons, depth SoilDepth m.
  ```
  **Example:** The following example shows explicitly the format the returned `String`.
  ```
  profile "spf15", horizons 4, depth 1.25 m
  ```

- **RetrieveHorizonDepths**
  Retrieves the (upper and lower) boundary depths of all horizons in the soil profile associated with a particular Web Form page.
  ```csharp
  internal static double[,] RetrieveHorizonDepths(Page page);
  ```
  **Return Value:** A 2-dimensional `Double` array that contains the horizon-specific boundary depths of a `SoilProfile`. Length of the 1\textsuperscript{st} dimension equals the number of horizons and length of the 2\textsuperscript{nd} dimension equals 2. A null reference if the horizons data not available.

- **RetrieveNumberOfHorizons**
  Retrieves the number of `SoilHorizon`\s which consists of the `SoilProfile` associated with a particular Web Form page.
  ```csharp
  internal static int RetrieveNumberOfHorizons(Page page);
  ```
  **Return Value:** The number of soil horizons included with the soil data; -1 if soil data is not available yet.

- **RetrieveParticleSizeDistribution**
  Retrieves the particle size distribution of all horizons in the `SoilProfile` associated with a particular Web Form page.
  ```csharp
  internal static double[,] RetrieveParticleSizeDistribution(Page page);
  ```
  **Return Value:** A 2-dimensional `Double` array that contains the horizon-specific clay, silt and sand contents of a `SoilProfile`. Length of the 1\textsuperscript{st} dimension equals the number of horizons and length of the 2\textsuperscript{nd} dimension equals 3. A null reference if the horizons data not available.

- **RetrieveProfileDepth**
  Retrieves the depth (m) of the `SoilProfile` associated with a particular Web Form page.
  ```csharp
  internal static double RetrieveProfileDepth(Page page);
  ```
  **Return Value:** The depth of the soil profile; -1 if soil data is not available yet.

- **RetrieveProfileNumber**
  Retrieves the `ProfileNumber` of the `SoilProfile` associated with a particular Web Form page.
Namespace UGent.WLES.Interactive

```csharp
internal static string RetrieveProfileNumber(Page page);
Return Value: The soil profile number included with the soil data; empty string if soil data is not available yet.
```

- **RetrieveRootLimitingLayerDepth**
  Retrives the depth (m) of the root limiting layer associated with a particular Web Form page.
  ```csharp
  internal static double RetrieveRootLimitingLayerDepth(Page page);
  Return Value: The depth of the root limiting layer; -1 if not present.
  ```

- **RetrieveSandClass**
  Retrives the SoilData.SandClass associated with a particular Web Form page.
  ```csharp
  internal static SandClass RetrieveSandClass(Page page);
  Return Value: A SoilData.SandClass member; -1 if sand class unknown.
  ```

- **RetrieveSoilStructure**
  Retrives the SoilData.SoilStructure associated with a particular Web Form page.
  ```csharp
  internal static SoilStructure RetrieveSoilStructure(Page page);
  Return Value: A SoilData.SoilStructure member; -1 if structure unknown.
  ```

- **SetupSoilProfile**
  Sets up a SoilProfile instance using the soil data associated with a particular Web Form page.
  ```csharp
  internal static SoilProfile SetupSoilProfile(Page page);
  Return Value: A SoilProfile; a null reference if data not available or incomplete.
  Remarks: Difference between this method and the same name method defined in SoilDataUI1:
  1. This method is implemented on top of the SoilDataUI1.SetupSoilProfile method;
  2. This method assigns chemical properties values to the profile, while the physical properties and general descriptions were added by the SetupSoilProfile method of the SoilDataUI1 class.

### 9.8 StudyArea Class

#### 9.8.1 Summary
Represents the study area input interface of the WLES Web UI.

#### 9.8.2 Declaration
```csharp
public class StudyArea : Page
```

#### 9.8.3 Remarks
This page can be accessed with a query string which contains at least one key-value pair entitled “redirect”. In the following example:

```
studyarea.aspx?rid=20050101ABCDEF&foo=bar
```

the query string begins after the “?” sign, and the key-value pairs are separated by the “&” sign. A new interactive evaluation is initialized if the “rid” query string does not present or is empty; if it presents with a non-empty value, an existing evaluation is to be revisited.

A query string may be generated automatically when an HTML FORM is submitted using the HTTP GET method.
9.8 StudyArea Class

9.8.4 Public Static Methods

- **FormCoordinatesString**
  Returns an HTML-encoded String that represents the geo-coordinates of the climatic station with the following format:

  $$(DDD^{\circ}MM^{'}SS^{''}\ N|S,\ DDD^{\circ}MM^{'}SS^{''}\ E|W)$$

  where: $DDD$ is the 3-digit degree component, $MM$ the 2-digit minute component, and $SS$ the 2-digit second component of the latitude or the longitude; $N|S$ specifies the **Northern** or the **Southern** Hemisphere, and $E|W$ the **Eastern** or the **Western** Hemisphere.

  
  `[C#] public static string FormCoordinatesString(Page page);`

  Examples:

  - **Example 1**: The following example shows explicitly the format of a returned String:
    $$(032^{\circ}25^{'}03^{''}\ N,\ 118^{\circ}35^{'}45^{''}\ E)$$
    It displays in a Web-page as:
    $$(032^{\circ}\ 25^{'}\ 03^{''}\ N,\ 118^{\circ}\ 35^{'}\ 45^{''}\ E)$$
  - **Example 2**: The HTML entities in the returned String may be replaced for display as plain text:
    ```
    string coordinates = StudyArea.FormCoordinatesString(Page);
    coordinates = coordinates.Replace("\&deg;","d")
    .Replace("\''","s") .Replace("\"","m")
    ```

- **FormLocationString**
  Forms a String that contains the station name, the country code and the geo-coordinates of the study area which is associated with a particular Web Form.

  ```
  [C#] public static string FormLocationString(Page page);
  ```

  **Return Value**: A String; a null reference if the current page is not RID-authenticated.

  **Example**: The following example shows explicitly the format the returned String:

  **Lakeside/XX (032°25′03″ N, 118°35′45″ E)**

  where **XX** stands for the **Unknown** country.

9.8.5 Internal Static Methods

- **DetermineHemisphereEW**
  Determines the hemisphere of a particular longitude value.

  ```
  [C#] internal static char DetermineHemisphereEW(double longitude);
  ```

  **Return Value**: ‘E’ for the **Eastern** Hemisphere; ‘W’ for the **Western** Hemisphere.

- **DetermineHemisphereNS**
  Determines the hemisphere of a particular latitude value.

  ```
  [C#] internal static char DetermineHemisphereNS(double latitude);
  ```

  **Return Value**: ‘S’ for the **Southern** Hemisphere; ‘N’ for the **Northern** Hemisphere.

- **RetrieveCalendarType**
  Retrieves the **RomanDate.CalendarType** of the dataset associated with the current page from the WLES database.

  ```
  [C#] internal static CalendarType RetrieveCalendarType(Page page);
  ```

  **Return Value**: A **RomanDate.CalendarType** member; -1 if the current page is not RID-authenticated.
• RetrieveCountryCode
Retrieves the 2-char ISO 3166 country code that is associated with the current page from the WLES database.

[C#] internal static string RetrieveCountryCode(Page page);
Return Value: A String as the country code; a null reference if the current page is not RID-authenticated.

• RetrieveDataResolution
Retrieves the WlesDataSet.TemporalResolution of the dataset associated with the current page from the WLES database.

[C#] internal static TemporalResolution RetrieveDataResolution(Page page);
Return Value: A WlesDataSet.TemporalResolution member; -1 if the current page is not RID-authenticated.

• RetrieveElevation
Retrieves the elevation of the climatic station that is associated with the current page from the WLES database.

[C#] internal static double RetrieveElevation(Page page);
Return Value: A Double as the elevation; -9999.0 if the current page is not RID-authenticated.
Remarks:
1. Valid elevation values are in the range [-100, +100];
2. The hemisphere can be subsequently determined;
3. Elevation values in meters can be converted to the degree-minute-second expression by using the Degree2DMS method.

• RetrieveLongitude
Retrieves the longitude of the climatic station that is associated with the current page from the WLES database.

[C#] internal static double RetrieveLongitude(Page page);
Return Value: A Double as the longitude; -250.0 if the current page is not RID-authenticated.
Remarks:
1. Valid longitude values are in the range [-180, +180];
2. The hemisphere can be subsequently determined;
3. Latitude values in decimal degrees can be converted to the degree-minute-second expression by using the Degree2DMS method.

• RetrieveStationName
Retrieves the climatic station name associated with the current page from WLES database.

[C#] internal static string RetrieveStationName(Page page);
Return Value: A String as station name; a null reference if the current page is not RID-authenticated.
Chapter 10

Namespace UGent.WLES.Batch

The UGent.WLES.Batch namespace contains all necessary classes and structures for guiding the users to fulfil their evaluation tasks in the batch mode. In the batch mode, users feed a complete dataset to the system and receive the evaluation results once, provided that no logical and format errors are found in the dataset.

10.1 Overview

10.1.1 Classes

<table>
<thead>
<tr>
<th>type</th>
<th>summary</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>Represents the batch evaluation interface of the WLES Web UI.</td>
<td>289</td>
</tr>
</tbody>
</table>

10.2 Evaluation Class

10.2.1 Summary

Represents the batch evaluation interface of the WLES Web UI.

10.2.2 Declaration

[C#] public class Evaluation : Page

10.2.3 Remarks

1. This class wraps the WebControls and HtmlControls that constitutes the Web UI of the UGent.WLES.Batch.Evaluation class, and their event handlers;

2. Fundamental functionalities behind the Web UI are mostly provided by the UGent.WLES Class Library.
Chapter 11

Namespace
UGent.WLES.ErrorHandler

The WLES possesses a centralized error handling mechanism with a 3-level hierarchical structure, which consists of the Page_Error event handler at the page level, the Application_Error event handler at the application level, and the application error configuration using declarative error handling directives defined in the Web.config file. The UGent.WLES.ErrorHandler namespace wraps Web Forms and classes that represent the customized redirecting target pages declared in the “customErrors” section of the Web.config file.

11.1 Overview

11.1.1 Classes

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneralError</td>
<td>Represents the WLES general application error notification interface</td>
<td>291</td>
</tr>
<tr>
<td>PageNotFoundError</td>
<td>Represents the WLES page-not-found error notification interface.</td>
<td>292</td>
</tr>
<tr>
<td>ServerError</td>
<td>Represents the WLES server error notification interface.</td>
<td>292</td>
</tr>
</tbody>
</table>

11.2 GeneralError Class

11.2.1 Summary

Represents the WLES general application error notification interface.

11.2.2 Declaration

[C#]
public class GeneralError : Page

11.2.3 Remarks

The user is redirected to here when an unhandled general application error occurs. The redirection is declared in the customErrors section of the application configuration file Web.config, as part of the centralized error handling mechanism of WLES.

Refer to the Application_ErrorEventHandler for other measures taken to handle this application error.
11.3 **PageNotFoundError Class**

11.3.1 **Summary**

Represents the WLES *page-not-found* error notification interface.

11.3.2 **Declaration**

[C#] public class PageNotFoundError : Page

11.3.3 **Remarks**

The user is redirected to here when an unhandled page-not-found error occurs. The redirection is declared in the `customErrors` section of the application configuration file `Web.config`, as part of the centralized error handling mechanism of WLES.

Refer to the `Application_Error EventHandler` for other measures taken to handle this application error.

11.4 **ServerError Class**

11.4.1 **Summary**

Represents the WLES server error notification interface.

11.4.2 **Declaration**

[C#] public class ServerError : Page

11.4.3 **Remarks**

The user is redirected to here when an unhandled server error occurs. The redirection is declared in the `customErrors` section of the application configuration file `Web.config`, as part of the centralized error handling mechanism of WLES.

Refer to the `Application_Error EventHandler` for other measures taken to handle this application error.
Chapter 12

Namespace UGent.IP2Country

The UGent.IP2Country namespace provides IP to country resolution. As a result, the country-specific fact sheets can be fetched as long as long the client IP is known.

12.1 Overview

12.1.1 Classes

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientInfo</td>
<td>Determines the identity of a Web client. The identity data include hostname, IP address and country of origin.</td>
<td>293</td>
</tr>
</tbody>
</table>

12.1.2 Enumerations

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientInfo.ImageFormat</td>
<td>Specifies the supported image formats.</td>
<td>295</td>
</tr>
<tr>
<td>ClientInfo.ImageSize</td>
<td>Specifies the image size the country flag.</td>
<td>296</td>
</tr>
</tbody>
</table>

12.2 ClientInfo Class

12.2.1 Summary

Determines the identity of a Web client. The identity data include hostname, IP address and country of origin.

12.2.2 Declaration

[C#] public class ClientInfo

12.2.3 Example

The following example demonstrates two ways of instantiating the ClientInfo class. The example also shows how to obtain meta data about the Web client. The meta data include the IP address, hostname, country of origin and other country-specific resources.

Further actions can be taken on the basis of the client’s IP address or of the country of origin. Users of the ClientInfo class can fulfill functionalities like load balancing, address forwarding, and etc. with minor efforts.

```csharp
using UGent.IP2Country;

// Approach 1: The ClientInfo class can be
```
// initialized directly with an IP address
string client = "207.46.250.252";
ClientInfo ci = new ClientInfo(client);

string hostname = ci.ClientName;
string country = ci.GetCountryName();
string flag = ci.GetFlag(ImageSize.BIG);

// Approach 2: The ClientInfo class is initialized
// with its default constructor, and the IP address
// is then set through its ClientIP property.
ClientInfo info = new ClientInfo();

string[] clients = new string[4];
clients[0] = "157.193.40.42";
clients[1] = "202.69.73.254";
clients[2] = "66.94.203.45";
clients[3] = "66.102.11.99";

foreach (string c in clients) {
    info.ClientIP = c;
    hostname = info.ClientName;
    country = info.GetCountryName();
    flag = info.GetFlag(ImageSize.SMALL);
    System.Console.WriteLine("{0}, {1}, {2}
{3}
", c, hostname, country, flag);
}

12.2.4 Public Static Methods

- GetCountryCode
  Returns the 2-char ISO 3166 country code of a particular country.
[C#] public static string GetCountryCode(string countryName);

- GetCountryName
  Returns the full name of a particular country.
[C#] public static string GetCountryName(string countryCode);

12.2.5 Public Instance Constructors

- public ClientInfo();
  Default constructor which initializes a new instance of the ClientInfo class with a null reference as the IP address.

- public ClientInfo(string clientIP);
  Initializes a new instance of the ClientInfo class with a valid IP address.
  **Remarks:** If clientIP is not valid, then a null reference is associated with the new instance of ClientInfo as the IP address.

12.2.6 Public Instance Properties

- ClientHostName
  Gets the client hostname which is obtained from the DNS server by using its IP address.
[C#] public string ClientHostName { get; }

12.3 ClientInfo.ImageFormat Enumeration

- ClientIP
  Gets or sets the client IP address. The client IP address is set to a null reference if the value passed in is invalid.
  
  [C#] public string ClientIP { get; set; }

12.2.7 Public Instance Methods

- GetCountryCode
  Returns the 2-char ISO 3166 country code.
  
  [C#] public string GetCountryCode();

- GetCountryName
  Returns the full name of the country.
  
  [C#] public string GetCountryName();

- GetFlag
  Returns the URL pointing to the flag (i.e., an image file) of the Web client’s originating country.
  
  [C#] public string GetFlag(ImageSize size);
  
  **Return Value:** A String representing the URL of the flag image file. An empty String (i.e., "") is returned if the image file does not exist, or if the FlagFilePath is not accessible from within the Web Server.
  
  **Remarks:** Only the image formats specified in ImageFormat Enumeration are supported.

12.2.8 Protected Instance Properties

- FlagFilePath
  Gets the absolute path pointing to the directory of country flag image files.
  
  [C#] protected string FlagFilePath { get; }

- WebServerURL
  Gets the URL of the containing Web Server.
  
  [C#] protected string WebServerURL { get; }

12.3 ClientInfo.ImageFormat Enumeration

12.3.1 Summary

Specifies the supported image formats.

12.3.2 Declaration

  [C#] public enum ClientInfo.ImageFormat

12.3.3 Remarks

The ImageFormat enumeration represents the image format that the ClientInfo class supported.

This enumeration is useful when it is desirable to have a strongly typed specification of the image format.

12.3.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF</td>
<td>The GIF format.</td>
</tr>
<tr>
<td>JPG</td>
<td>The JPG format.</td>
</tr>
</tbody>
</table>
12.4 ClientInfo.ImageSize enumeration

12.4.1 Summary
 Specifies the image size the country flag.

12.4.2 Declaration

[C#] public enum ClientInfo.ImageSize

12.4.3 Remarks
 The ImageSize enumeration represents the size of the image returned. Two sizes are defined, namely the BIG size and the SMALL size. The BIG size is equivalent to roughly 320 by 200 pixels, while the SMALL size is 32 by 20.

This enumeration is useful when it is desirable to have a strongly typed specification of the image size. For example, this enumeration is among the formal parameters of the GetFlag method.

12.4.4 Members

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALL</td>
<td>Width by height: 32 by 20 pixels, roughly.</td>
</tr>
<tr>
<td>BIG</td>
<td>Width by height: 320 by 200 pixels, roughly.</td>
</tr>
</tbody>
</table>
Bibliography


# Index

<p>| ABSOLUTE_ZERO, 82 | BSRating, 236, 241 |
| Account class, 262 | BuildRequirementTable, 236 |
| Constructors, 263 | BulkDensity, 56 |
| Properties, 263 | C, 122 |
| ACEC, 55, 117, 119 | C3, 170 |
| ACECRating, 236, 241 | C4, 170 |
| Acrisols, 123 | C60s, 122 |
| ActualEvapotranspirationCropCycle, 213 | C60v, 122 |
| AddNewCrop, 87 | C60s, 122 |
| AddNewRefCrop, 93 | C60v, 122 |
| AddToEventLog, 12 | Ca, 56 |
| AdminEmail, 9 | CaCO3, 57, 117 |
| AH, 172 | CaCO3Rating, 237, 241 |
| Albeluvisols, 123 | Calcisols, 124 |
| Algorithm enumeration, 260 | CalculateClearSkyRatio, 83 |
| Members, 260 | CalculateDayNightTemperatures, 83 |
| Remarks, 260 | CalculateDecadelyFromDaily, 83 |
| Alisols, 123 | CalculateDecadelyFromMonthly, 83 |
| ALPHA, 195 | CalculateESP, 63 |
| Andosols, 123 | CalculateETmCropCycle, 185 |
| Answer, 263 | CalculateETmDailyArray, 185 |
| Anthrosols, 123 | CalculateETo, 196, 278 |
| AppraisingScheme, 249 | CalculateMonthlyFromDaily, 83 |
| Approach enumeration, 231 | CalculateMonthlyFromDecadely, 83 |
| Members, 231 | CalculateOvercastRatio, 84 |
| APR, 51 | CalculatePercentageCCinGP, 280 |
| Arenosols, 124 | CalculatePm, 137 |
| AUG, 51 | CalculateRPP, 139 |
| AuthCheck class, 266 | CalculateSoilWaterDepletionFractionCropCycle, 186 |
| authentication, 8, 10, 12, 261, 266, 273 | CalculateYieldReducingCoefWater, 209 |
| forms authentication, 9, 12, 13, 19 | Calendar, 35, 150 |
| ticket, 9, 12, 13, 19 | CalendarType enumeration, 49 |
| AuthorEmail, 9 | Members, 50 |
| Average, 70 | CAM, 170 |
| BaseLine, 259 | Cambisols, 124 |
| BaseSaturation, 55 | CD, 172 |
| Batch, 49 | CEC, 57 |
| Batch namespace, 289 | CELSIUS, 115 |
| Evaluation class, 289 | CF, 117 |
| Overview, 289 | CFCorrectionCoef, 237 |
| Biomass class, 136 | CFRating, 237, 241 |
| Example, 136 | CGP0BeginningRomanDate, 225 |
| Remarks, 136 | CGP0EndingRomanDate, 225 |
| Blocky, 121 | CGP1BeginningRomanDate, 225 |
| BS, 117, 119 | 299 |</p>
<table>
<thead>
<tr>
<th>CGP1EndingRomanDate</th>
<th>225</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGP2BeginningRomanDate</td>
<td>225</td>
</tr>
<tr>
<td>CGP2EndingRomanDate</td>
<td>225</td>
</tr>
<tr>
<td>CGP3BeginningRomanDate</td>
<td>225</td>
</tr>
<tr>
<td>CGP3EndingRomanDate</td>
<td>226</td>
</tr>
<tr>
<td>CGP4BeginningRomanDate</td>
<td>226</td>
</tr>
<tr>
<td>CGP4EndingRomanDate</td>
<td>226</td>
</tr>
<tr>
<td>CGPToString</td>
<td>226</td>
</tr>
<tr>
<td>ChangeCalendar</td>
<td>35</td>
</tr>
<tr>
<td>ChangeCalendarForDailyArray</td>
<td>84</td>
</tr>
<tr>
<td>ChangeEncoding</td>
<td>38</td>
</tr>
<tr>
<td>Chernozems</td>
<td>124</td>
</tr>
<tr>
<td>CL</td>
<td>122</td>
</tr>
<tr>
<td>classification</td>
<td>67</td>
</tr>
<tr>
<td>Clay</td>
<td>81</td>
</tr>
<tr>
<td>ClientHostName</td>
<td>294</td>
</tr>
<tr>
<td>ClientInfo class</td>
<td>293</td>
</tr>
<tr>
<td>constructors</td>
<td>294</td>
</tr>
<tr>
<td>example</td>
<td>293</td>
</tr>
<tr>
<td>properties</td>
<td>294, 295</td>
</tr>
<tr>
<td>ClientInfo.ImageFormat enumeration</td>
<td>295</td>
</tr>
<tr>
<td>members</td>
<td>295</td>
</tr>
<tr>
<td>Remarks</td>
<td>295</td>
</tr>
<tr>
<td>ClientInfo.ImageSize enumeration</td>
<td>296</td>
</tr>
<tr>
<td>members</td>
<td>296</td>
</tr>
<tr>
<td>Remarks</td>
<td>296</td>
</tr>
<tr>
<td>ClientIP</td>
<td>295</td>
</tr>
<tr>
<td>ClimaticData</td>
<td>49</td>
</tr>
<tr>
<td>ClimaticData structure</td>
<td>82</td>
</tr>
<tr>
<td>ClimaticDataUI class</td>
<td>277</td>
</tr>
<tr>
<td>Remarks</td>
<td>277</td>
</tr>
<tr>
<td>ClimaticParamSet enumeration</td>
<td>113</td>
</tr>
<tr>
<td>members</td>
<td>114</td>
</tr>
<tr>
<td>Remarks</td>
<td>114</td>
</tr>
<tr>
<td>Cm</td>
<td>122</td>
</tr>
<tr>
<td>Co</td>
<td>122</td>
</tr>
<tr>
<td>CoarseSand</td>
<td>116</td>
</tr>
<tr>
<td>ColdPeriodBeginningDates</td>
<td>161</td>
</tr>
<tr>
<td>ColdPeriodEndingDates</td>
<td>161</td>
</tr>
<tr>
<td>ColdSeason</td>
<td>161</td>
</tr>
<tr>
<td>Columnar</td>
<td>121</td>
</tr>
<tr>
<td>CombinedFineEarthRating</td>
<td>242</td>
</tr>
<tr>
<td>Comments</td>
<td>49</td>
</tr>
<tr>
<td>Comments class</td>
<td>266</td>
</tr>
<tr>
<td>Remarks</td>
<td>266</td>
</tr>
<tr>
<td>ConcatErrorMessage</td>
<td>12</td>
</tr>
<tr>
<td>ConcatPath</td>
<td>38</td>
</tr>
<tr>
<td>ConnectionStringIP2Country</td>
<td>93</td>
</tr>
<tr>
<td>ConnectionStringWLES</td>
<td>93</td>
</tr>
<tr>
<td>ConvertRating2Suitability</td>
<td>237</td>
</tr>
<tr>
<td>cookie</td>
<td>8–10, 12</td>
</tr>
<tr>
<td>Expires</td>
<td>8, 10</td>
</tr>
<tr>
<td>LicenseCookie</td>
<td>10</td>
</tr>
<tr>
<td>name</td>
<td>8, 10, 12</td>
</tr>
<tr>
<td>payload</td>
<td>19</td>
</tr>
<tr>
<td>persistent</td>
<td>8, 10</td>
</tr>
<tr>
<td>timeout</td>
<td>8, 10</td>
</tr>
<tr>
<td>value</td>
<td>12</td>
</tr>
<tr>
<td>WLESCookie</td>
<td>273</td>
</tr>
<tr>
<td>CookieName</td>
<td>10</td>
</tr>
<tr>
<td>CookieTimeout</td>
<td>10</td>
</tr>
<tr>
<td>Coordinates</td>
<td>67</td>
</tr>
<tr>
<td>CopyrightMessage</td>
<td>12</td>
</tr>
<tr>
<td>CountDays</td>
<td>32, 35</td>
</tr>
<tr>
<td>CountDecades</td>
<td>32, 35</td>
</tr>
<tr>
<td>CountMonths</td>
<td>35</td>
</tr>
<tr>
<td>Country</td>
<td>263</td>
</tr>
<tr>
<td>CountryFlag class</td>
<td>266</td>
</tr>
<tr>
<td>Remarks</td>
<td>267</td>
</tr>
<tr>
<td>Crop</td>
<td>150, 180, 201, 213, 226, 241, 249</td>
</tr>
<tr>
<td>CropCoeff</td>
<td>49</td>
</tr>
<tr>
<td>CropCoefficient class</td>
<td>176</td>
</tr>
<tr>
<td>constructors</td>
<td>179</td>
</tr>
<tr>
<td>properties</td>
<td>180</td>
</tr>
<tr>
<td>Remarks</td>
<td>176</td>
</tr>
<tr>
<td>CropCoefficientCropCycle</td>
<td>213</td>
</tr>
<tr>
<td>CropCoefficientUI class</td>
<td>184</td>
</tr>
<tr>
<td>Remarks</td>
<td>184</td>
</tr>
<tr>
<td>CropCycle class</td>
<td>145</td>
</tr>
<tr>
<td>constructors</td>
<td>149</td>
</tr>
<tr>
<td>example</td>
<td>145</td>
</tr>
<tr>
<td>properties</td>
<td>150</td>
</tr>
<tr>
<td>Remarks</td>
<td>145</td>
</tr>
<tr>
<td>CropCycleBeginningDate</td>
<td>226</td>
</tr>
<tr>
<td>CropCycleBeginningRomanDate</td>
<td>150, 201, 213</td>
</tr>
<tr>
<td>CropCycleEndingDate</td>
<td>226</td>
</tr>
<tr>
<td>CropCycleEndingRomanDate</td>
<td>150, 201, 213</td>
</tr>
<tr>
<td>CropCycleInstance</td>
<td>180</td>
</tr>
<tr>
<td>CropData</td>
<td>49</td>
</tr>
<tr>
<td>CropData structure</td>
<td>86</td>
</tr>
<tr>
<td>Remarks</td>
<td>86</td>
</tr>
<tr>
<td>CropDataUI class</td>
<td>279</td>
</tr>
<tr>
<td>Remarks</td>
<td>279</td>
</tr>
<tr>
<td>CropDevelopmentStage enumeration</td>
<td>171</td>
</tr>
<tr>
<td>members</td>
<td>172</td>
</tr>
<tr>
<td>Remarks</td>
<td>171</td>
</tr>
<tr>
<td>CropGrowthPeriod enumeration</td>
<td>172</td>
</tr>
<tr>
<td>members</td>
<td>173</td>
</tr>
<tr>
<td>Remarks</td>
<td>172</td>
</tr>
<tr>
<td>CropRequirements</td>
<td>49</td>
</tr>
<tr>
<td>CropRequirements class</td>
<td>234</td>
</tr>
<tr>
<td>constructors</td>
<td>239</td>
</tr>
<tr>
<td>examples</td>
<td>234</td>
</tr>
<tr>
<td>properties</td>
<td>241</td>
</tr>
<tr>
<td>Remarks</td>
<td>234</td>
</tr>
<tr>
<td>CropRequirementsUI class</td>
<td>244</td>
</tr>
<tr>
<td>Remarks</td>
<td>244</td>
</tr>
<tr>
<td>CropSpecific</td>
<td>259</td>
</tr>
</tbody>
</table>
Cryosols, 124
Cultivar, 150, 180, 201, 213, 226, 241, 249

INDEX

Cryosols, 124

cS, 122
cSL, 122

CustomImage class, 267
Example, 268
Remarks, 267
CustomImage, 49

DAILY, 128

Data namespace, 53

ClimaticData structure, 82
CropData structure, 86

Database structure, 93

Overview, 53

SoilData structure, 97
SoilData.GravelType enum, 115
SoilData.SoilTextureClass enum, 115
SoilData.SoilClass enum, 116
SoilData.SoilCharacteristic enum, 116
SoilData.SoilDrainageClass enum, 118
SoilData.SoilFertilityCharacteristic enum, 119
SoilData.SoilStructure enum, 119
SoilData.SoilTextureClass enum, 121
SoilData.WRBSoilGroup enum, 122
SoilProfile class, 54
SoilTexture class, 78
WlesDataSet structure, 100
WlesDataSet.TemporalResolution enum, 128

Database structure, 93

Properties, 93

DataResolution, 161
Day, 35
DAYLENGTH, 114
DAYS, 50

DEC, 51
Dec2Hex, 38
DECADELY, 128
DECADES, 50
DefaultCalendar, 32
Degree2DMS, 38
Degree2Radian, 38
Degree2Second, 38
DepletionFraction, 49

DepletionFractionUI class, 185
Remarks, 186

Depth, 57, 67
Description, 57

DetermineHemisphereEW, 287
DetermineKcAH, 177
DetermineKcIS, 177
DetermineKcMS, 177

DetermineStage, 150

DisplayMessage, 12
DisplayMessageAtBottom, 12
Distance, 35

DMS2Degree, 38
DMS2Second, 38

Doc, 49
DoGuestLogin, 12
DoLogin, 12
DoLogout, 13

DrawCurves, 38
DryPeriodBeginningDates, 161
DryPeriodEndingDates, 162
DrySeason, 162
Durisols, 125

Early, 173

ESPRating, 238, 242

Establishment, 173

Evaluate, 143, 215
Evaluation class, 281, 289
Remarks, 289
EVAPOTRANSPIRATION, 114
EvapotranspirationAtMaximumRate-CropCycle, 213
EvapotranspirationAtReducedRate-CropCycle, 214
ExcessivelyDrained, 119
Exists, 39, 262, 264
Export4Download, 39
ExportData, 166
ExportKeData, 181
ExportKeDataDS, 178
ExportPmData, 139
ExportRefCDSData, 146
ExportRefKyData, 222
ExportResults, 215
ExtractData, 39, 151, 227
ExtractDigits, 39
ExtractSectionValue, 98
FAHRENHEIT, 115
FAO_AGLW, 190, 231
FarmingSystem enumeration, 115
   Members, 115
Fax, 264
FC_WP, 58
FEB, 51
Ferralsols, 125
FetchDataColumn, 93
Filter class, 269
   Remarks, 269
FindContinuousSubsets, 40
FindNeighbours, 40
FindPosition, 40
FineSands, 116
Firm, 264
FirstName, 264
FirstSubstring, 40
FixedPercentage, 190, 231
FlagFilePath, 295
Flowering, 173
Flush, 103
FlushToDB, 265
Fluvisols, 125
FormCoordinatesString, 287
FormCropCycleString, 280
FormCropString, 280
FormDepletionString, 186
FormKcValuesString, 185
FormLocationString, 287
FormParentMaterialString, 282, 285
FormRootingDepthString, 203
FormSelectedSoilCharacteristicsString, 282
FormSoilString, 282, 285
FormYieldResponseString, 230
FS, 122
FSL, 122
FullName, 264
GeneralError class, 291
   Remarks, 291
GenerateCID, 13
GenerateRID, 13
GenerateUserName, 262
GetActualEvapotranspiration-DailyArray, 216
GetActualEvapotranspirationSum, 216
GetAlgorithm, 103
GetAppraisingScheme, 103
GetBases, 63
GetBaseSaturation, 63
GetBasesSum, 64
GetCDSBeginningRomanDate, 151
GetCDSDailyArray, 151
GetCDSEndingRomanDate, 151
GetCDSRomanDates, 152
GetCGPDailyArray, 227
GetCGPDates, 227
GetCGPLength, 228
GetClientInfo, 14
GetColdSeasonDates, 166
GetColdSeasonRomanDates, 166
GetCountryCode, 294, 295
GetCountryName, 294, 295
GetCrop, 103
GetCropCoefficientDailyArray, 216
GetCropCycleBeginningRomanDate, 103
GetCropCycleEndingRomanDate, 103
GetCropCycleLength, 140, 144, 228
GetCropCycleRomanDates, 104, 140, 144, 152
GetCropDevelopmentStageRomanDates, 104
GetCropDevelopmentStages, 104
GetCropGrowthPeriodLength, 104
GetCropID, 87
GetCropNameAndCultivar, 87
GetCropsByPhotosynGroup, 140
GetCropsBySoilWaterDepletionGroup, 209
GetCropYields, 87
GetCultivar, 104
GetDailyArray, 84
GetDataResolution, 104
GetDate, 32
GetDateTime, 40
GetDaysPerDecade, 33
GetDaysPerMonth, 33
GetDecadeData, 84
GetDrySeasonDates, 166
GetDrySeasonRomanDates, 166
INDEX

GetEasilyAvailableSoilWaterDailyArray, 216
GetEffectiveRainfallCropCycle, 194
GetEffectiveRainfallDailyArray, 194, 216
GetElevation, 105
GetEvalMode, 14
GetEvalModeNumber, 14
GetEvalModes, 14
GetEvalPhase, 14
GetEvalPhaseNumber, 14
GetEvalPhases, 14
GetEvapotranspirationAtMaximumRateDailyArray, 216
GetEvapotranspirationAtMaximumRateSum, 217
GetEvapotranspirationAtReducedRateDailyArray, 217
GetEvapotranspirationAtReducedRateSum, 217
GetExceptionalValue, 40
GetFlag, 295
GetGrowingSeasonDates, 166
GetGrowingSeasonRomanDates, 167
GetHarvestIndex, 89, 105
GetHarvestIndexRange, 89
GetHelpContent, 15
GetHelpTopics, 15
GetHorizon, 70
GetHumidSeasonDates, 167
GetHumidSeasonRomanDates, 167
GetIndex, 247, 255
GetInputLevel, 105
GetInternalLink, 15
GetKcDailyArray, 184
GetLAImax, 105
GetLastEvalPhase, 16
GetLatitude, 105
GetLeafAreaIndex, 89
GetLeafAreaIndexRange, 89
GetLength, 33, 152, 167
GetLengthOfColdSeason, 167
GetLengthOfDecade, 33
GetLengthOfDrySeason, 167
GetLengthOfGrowingSeason, 167
GetLengthOfHumidSeason, 168
GetLengthOfMonth, 33
GetLengthOfRainySeason, 168
GetLengthOfSubhumidSeason, 168
GetLengthOfYear, 33
GetLimitingLayerDepth, 105
GetLongitude, 106
GetLPP, 256
GetManagementGroupID, 248
GetManagementIndex, 89
GetMaximumAirTemperature, 106
GetMaximumAirTemperatureDailyArray, 106
GetMaximumEvapotranspirationCropCycle, 184
GetMaximumEvapotranspirationDailyArray, 184, 217
GetMaximumEvapotranspirationSum, 217
GetMaximumRootingDepth, 106
GetMeanAirTemperature, 107
GetMeanAirTemperatureDailyArray, 107
GetMonthDay, 33
GetMonthDayString, 33, 35
GetMonthOfYear, 33
GetNameValueCollection, 16
GetNextEvalPhase, 16
GetOptimalDepth, 107, 200
GetOptimalDepthRange, 200
GetOptimalRootingDepth, 90
GetOptimalRootingDepthRange, 90
GetPAW, 65, 99, 210
GetPhotosynGroupID, 141
GetPlace, 107
GetPoint, 82
GetPolygon, 80
GetPreviousEvalPhase, 16
GetProfileReport, 265
GetQueryString, 16
GetQueryStringKeyValuePairs, 17
GetRainfall, 107
GetRainfallDailyArray, 108
GetRainySeasonDates, 168
GetRainySeasonRomanDates, 168
GetRefCDS, 147
GetRefCGP, 90, 222
GetRefCGPTotal, 93
GetRefCropCycle, 148
GetReferenceEvapotranspiration, 108
GetReferenceEvapotranspirationDailyArray, 108, 217
GetReferenceEvapotranspirationSum, 217
GetRefererQueryString, 18
GetRelativeAirHumidity, 108
GetRelativeAirHumidityDailyArray, 108
GetRevisedPAW, 210
GetRootingDepthDailyArray, 202, 217
GetScarcelyAvailableSoilWaterDailyArray, 217
GetSegments, 134
GetSessionTimeoutString, 18
GetSoilHorizons, 108
GetSoilProfile, 109
GetSoilWaterDepletionFraction, 91, 211
GetSoilWaterDepletionFractionDailyArray, 217
INDEX 305

IronOxide, 115
Irrigated, 115
IS, 172
IsAccumulative, 86
IsAuthenticated, 20
IsAverage, 86
IsCDSOk, 150
IsClimaticDataOk, 102
IsCookiePersistent, 10
IsCoordinatesOk, 68
IsCropCycleOk, 102
IsCropDataOk, 102
IsCropGrowthPeriodDataOk, 102
IsDigits, 41
IsDigitsOrLetters, 41
IsElevationOk, 68
IsEmailLegal, 20
IsEmptyString, 41
IsEToDataOk, 102
IsEvalModeDefined, 20
IsEvalPhaseDefined, 20
IsGuest, 20
IsHeadOrTail, 130, 134
IsHorizonIDValid, 283
IsHostLegal, 20
IsHostValid, 21
IsHumidClimate, 110, 254, 278, 282
IsInAscendingOrder, 41
IsInBatchMode, 21
IsInDatePeriod, 41
IsInDescendingOrder, 42
IsInInteractiveMode, 21
IsInPeriod, 35
IsInside, 131, 133
IsInRange, 42
IsLeguminous, 93, 249, 250, 280
IsLetters, 42
IsLicensed, 21
IsLocationalDataOk, 102
IsLongitudeOk, 168
IsManagementDataOk, 103
IsNameLegal, 21
IsOnline, 131, 133
IsParentMaterialCalcic, 68
IsParentMaterialKalifloric, 68
IsParentMaterialKnown, 68
IsProfileNumberValid, 283
IsPwdLegal, 21
IsReferrerHome, 21
IsRidAuthenticated, 21
IsSandClassOk, 68, 82
IsSessionExpired, 21
IsSlopeOk, 68
IsSoilDataOk, 103
IsSoilStructureOk, 82
IsStructureOk, 68
IsThisEvalPhaseFinished, 22
IsUidLegal, 22
IsUserValidated, 22
IsValidIP, 42
IsWettingIntervalOk, 103
IV, 170, 232, 259
JAN, 51
JUL, 51
JUN, 51
K, 59
Kastanozems, 126
KcAH, 181
KcCD, 181
KcCropCycle, 181
KcIS, 181
KcLS, 181
KcMS, 181
L, 122
LabHomePage, 10
LAMBDA, 83
LandSuitabilityClass enumeration, 115
Members, 116
Remarks, 115
LandUse, 68
LastName, 264
LastSubstring, 42
Late, 173
Latitude, 163
LeS, 122
Length, 36, 150
LengthOfYear enumeration, 50
Members, 50
Leptosols, 126
LiS, 122
LicenseCookieName, 10
LimitingLayerDepth, 60, 68, 201
Line structure, 129
Constructors, 130
Properties, 130
Remarks, 129
Lixisols, 126
Location, 69
Log, 22, 94
Login, 49
Login class, 272
Remarks, 272
LoginCounterVar, 10
Logistics class, 8
Fields, 9
Logout, 49
Logout class, 272
   Remarks, 273
Longitude, 163
Low, 258
LPP, 257
LPP namespace, 233
   CropRequirements class, 234
   CropRequirementsUI class, 244
   LPPReport class, 245
   ManagementIndex class, 246
   ManagementIndex.InputLevel enum, 258
   ManagementIndex.ManagementGroup enum, 259
   ManagementIndex.Scheme enum, 259
   ManagementIndexUI class, 249
   Overview, 233
   SoilIndex class, 250
   SoilIndex.Algorithm enum, 260
   SoilIndexUI class, 255
   Yield class, 255
LPPReport, 49
LPPReport class, 245
   Remarks, 245
LS, 122, 172
Luvisols, 126
MakeHelpUrl, 22
MakeUrlWithQueryString, 22
ManagementGroup enumeration, 259
   Members, 259
   Remarks, 259
ManagementIndex, 49, 257
ManagementIndex class, 246
   Constructors, 248
   Example, 246
   Properties, 249
   Remarks, 246
ManagementIndexInstance, 257
ManagementIndexUI class, 249
   Remarks, 250
MAR, 51
Massive, 121
Max, 42
MaximumAirTemperature, 163
MaximumDepth, 201
MaximumEvapotranspirationCropCycle, 214
MaxLAI, 10
MaxLengthName, 10
MaxLengthPwd, 10
MaxLengthUid, 10
MaxLoginRetry, 10
MAY, 51
MeanAirTemperature, 164
Mg, 60
Micromineral, 69
Min, 43
MinimumAirTemperature, 164
MinLengthName, 11
MinLengthPwd, 11
MinLengthUid, 11
ModeratelyWellDrained, 118
Month, 35
MONTHLY, 128
MonthOfYear enumeration, 50
   Members, 51
MONTHS, 50
MS, 172
MyAccount, 49
MyAccount class, 273
   Remarks, 273
MyProfile, 49
MyProfile class, 273
   Remarks, 273
N1, 116
N2, 116
Na, 60
NameRule, 11
Next, 36
Nitisols, 126
NORTHERN, 114
NOV, 51
OC, 60, 117, 119
OCRating, 238, 242
OCT, 51
OptimalDepth, 201
Oxic, 121
PageNotFoundError class, 292
   Remarks, 292
ParentMaterial, 69
ParticleDensity, 98
ParticleDistribution, 61
Password, 264
Password class, 273
   Remarks, 274
PenmanMonteith class, 195
   Fields, 195
   Remarks, 195
pH, 119
pH_H2O, 61, 117
pH_H2ORating, 238, 242
pH_KCl, 62
Phaeozems, 126
Phone, 264
PhotosynGroup enumeration, 170
   Members, 170
INDEX

Remarks, 170
PhotosynPathway enumeration, 170
   Members, 170
   Remarks, 170
Place, 150, 164
Planosols, 127
Platy, 121
Plinthosols, 127
PlotData, 43
Podzols, 127
Point structure, 130
   Constructors, 131
   Properties, 131
   Remarks, 130
Point2Texture, 80
Polygon structure, 132
   Constructors, 132
   Properties, 132
   Remarks, 132
Polyline structure, 133
   Constructors, 134
   Properties, 134
   Remarks, 133
PoorlyDrained, 118
PopulateCountries, 24
PostalAddress, 264
Previous, 36
Prismatic, 121
ProcessPostalAddress, 25
Profile, 241
ProfileDescription, 69
ProfileNrRule, 11
ProfileNumber, 69
PwdQuestions, 11
PwdRule, 11
Quartz, 115
Questions, 264
Radian2Degree, 44
RADIATION, 114
RAINFALL, 114
Rainfall, 164
Rainfed, 115
RainyPeriodBeginningDates, 164
RainyPeriodEndingDates, 164
RainySeason, 164
RandomList, 45
RandomUpperString, 45
ReadCell, 95
ReadData, 96
ReadDB, 265
RecalACEC, 71
RecalBS, 72
RecalCaCO3, 72
RecalEC, 72
RecalESP, 72
RecalGravel, 73
RecalGypsum, 73
RecalOC, 73
RecalParticleDistribution, 74
RecalPHH2O, 76
RecalPHKCl, 76
RecalSoilCharacteristic, 77
RecalSumBC, 78
Redirect, 25
RedirectToNextEvalPhase, 27
ReferenceEvapotranspiration, 164
ReferenceEvapotranspirationCropCycle, 214
ReferenceEvapotranspirationIS, 181
Register, 49
Register class, 274
   Remarks, 274
Regosols, 127
RelativeAirHumidity, 165
RelativeHumidityAH, 181
RelativeHumidityMS, 181
RemoveChars, 45
RemoveKeys, 27
RenameKeys, 27
Replace, 45
ReplaceKeys, 27
ReplaceUrls, 28
Requirements, 255
RequirementString, 243
RequirementTable, 241
RetrieveCalendarType, 287
RetrieveCDS, 280
RetrieveCGP, 230
RetrieveCountryCode, 288
RetrieveCropCultivarAlias, 280
RetrieveCropCycleRomanDates, 280
RetrieveDataResolution, 278, 288
RetrieveDelimiters, 278
RetrieveDepletionGroup, 186
RetrieveEffectiveCalendar, 29
RetrieveEffectiveRainfall, 195
RetrieveEffectiveRainfallMethod, 195
RetrieveElevation, 288
RetrieveHarvestIndex, 280
RetrieveHorizonDepths, 283, 285
RetrieveInputLevel, 250
RetrieveKcValues, 185
RetrieveKyScheme, 230
RetrieveKyValues, 231
RetrieveLAI, 280
RetrieveLatitude, 288
RetrieveLongitude, 288
RetrieveManagementGroup, 250
RetrieveManagementIndex, 250
RetrieveManagementIndexScheme, 250
RetrieveNumberOfHorizons, 283, 285
RetrieveOptimalRootingDepth, 203, 281
RetrieveParticleSizeDistribution, 283, 285
RetrievePhotosynGroup, 281
RetrievePhotosynPathway, 281
RetrieveProfileDepth, 284, 285
RetrieveProfileNumber, 284, 285
RetrieveRainfall, 278
RetrieveRequirementTable, 244
RetrieveRH, 278
RetrieveRID, 29
RetrieveRootLimitingLayerDepth, 203, 284, 286
RetrieveSandClass, 284, 286
RetrieveSelectedSoilCharacteristics, 282
RetrieveSoilIndexMethod, 255
RetrieveSoilStructure, 284, 286
RetrieveStationName, 288
RetrieveSunshineDuration, 278
RetrieveTemperatureThreshold, 278
RetrieveTmax, 279
RetrieveU2, 279
RetrieveWettingIntervalIS, 279
Ripening, 173
RockFragments, 115
RomanDate structure, 30
    Constructors, 34
    Examples, 31
    Operators, 33
    Remarks, 31
RootingDepth, 49, 243
RootingDepth class, 200
    Constructors, 201
    Properties, 201
    Remarks, 200
RootingDepthCropCycle, 201, 214
RootingDepthUI class, 202
    Remarks, 202
Round, 45
RPP namespace, 135
    Biomass class, 136
        Biomass.PhotosynGroup enum, 170
        Biomass.PhotosynPathway enum, 170
    CropCycle class, 145
        CropCycle.CropDevelopmentStage enum, 171
    CropCycle.CropGrowthPeriod enum, 172
    CropCycle.SubCGP enum, 173
    GrowingPeriod class, 155
    GrowingPeriodUI class, 168
    Overview, 135
RPPReport class, 169
RPPReport, 49
RPPReport class, 169
Remarks, 169
S, 122
S1, 116
S2, 116
S3, 116
Sand, 82, 116
SandClass, 62, 69, 82
SandClass enumeration, 116
Members, 116
Remarks, 116
SC, 122
ScarcelyAvailableSoilWaterCropCycle, 214
Scheme enumeration, 259
Members, 259
SCL, 122
SearchArray, 45
Second2Degree, 46
Second2DMS, 46
SendMail, 29
SendProfileByEmail, 266
SEP, 51
ServerError class, 292
Remarks, 292
SetCDSRomanDates, 153
SetClimaticData, 110
SetCropCycle, 110
SetCropData, 111
SetCropGrowthPeriods, 111
SetETaData, 112
SetKeys, 30
SetLocationalData, 112
SetManagementData, 112
SetSandClass, 82
SetSoilData, 112
SetSoilStructure, 82
SetTemperatureThreshold, 279
SetupBiomass, 169
SetupCropCoefficient, 185
SetupCropCycle, 281
SetupCropRequirements, 245
SetupGrowingPeriod, 169
SetupLPPYield, 245
SetupManagementIndex, 250
SetupRootingDepth, 203
SetupSoilIndex, 255
SetupSoilProfile, 284, 286
SetupWaterBalance, 219
SetupYieldResponseToWater, 231
SetWettingInterval, 113
SetWorkingCalendar, 113
INDEX

ShowAuthUid, 30
Si, 122
SiC, 122
SiCL, 122
SiCm, 122
SiCs, 122
SIGMA, 83
SiL, 122
Silt, 82
SimpleAverage, 78
Simulate, 217
SingleGrained, 121
SL, 122
Slope, 69
SlopeDegree2Percentage, 46
SlopePercentage2Degree, 46
SmallWords, 37
SmtpServer, 11
SoilCharacteristic enumeration, 116
  Members, 117
  Remarks, 117
SoilCharacteristicRating, 243
SoilCharacteristics, 255
SoilCharSet, 49
SoilCharSet class, 281
  Remarks, 281
SoilData structure, 97
  Remarks, 97
SoilData1, 49
SoilData2, 49
SoilDataUI1 class, 282
  Remarks, 282
SoilDataUI2 class, 284
  Remarks, 284
SoilDepth, 117
SoilDepthRating, 239, 243
SoilDrainageClass enumeration, 118
  Members, 118
  Remarks, 118
SoilFertilityCharacteristic enumeration, 119
  Members, 119
  Remarks, 119
SoilHorizon class, 54
  Constructors, 55
  Remarks, 54
SoilHorizons, 214
SoilIndex, 49, 257
SoilIndex class, 250
  Constructors, 253
  Example, 251
  Properties, 254
  Remarks, 251
SoilIndexInstance, 257
SoilIndexUI class, 255
  Remarks, 255
SoilProfile class, 65
  Constructors, 67
  Remarks, 65
SoilStructure, 82
SoilStructure enumeration, 119
  Members, 121
  Remarks, 120
SoilTexture class, 78
  Constructors, 81
  Remarks, 79
SoilTextureClass enumeration, 121
  Members, 122
  Remarks, 121
SoilWaterDepletionFractionCropCycle, 214
SoilWaterDepletionGroup enumeration, 232
  Members, 232
SoilWaterStorageCropCycle, 214
Solonchaks, 127
Solonetz, 127
SomewhatExcessivelyDrained, 119
SomewhatPoorlyDrained, 118
SOUTHERN, 114
SquareRoot, 260
Storie, 260
Structure, 62, 69
StudyArea, 49
  StudyArea class, 286
    Remarks, 286
SubCGP enumeration, 173
  Members, 173
  Remarks, 173
SubhumidPeriodBeginningDates, 165
SubhumidPeriodEndingDates, 165
SubhumidSeason, 165
SuitabilityClass, 243
Sum, 46
SumBC, 117, 119
  SumBCRating, 239, 244
SumOfBasicCations, 62
SUNSHINEHOURS, 114
SunshineHours, 165
Tail, 130, 134
TEMPERATURE, 114
TemperatureRegime enumeration, 114
  Members, 115
TemperatureThreshold, 165
TemporalResolution enumeration, 128
  Members, 128
  Remarks, 128
Texture, 117
Texture2Point, 81
  TextureClass2Polygon, 81
INDEX

WlesException class, 48
  Constructors, 48
WLESRoot, 11
WorkingCalendar, 166
WPP, 215, 257
WPP namespace, 175
  CropCoefficient class, 176
  CropCoefficientUI class, 184
  DepletionFractionUI class, 185
  EffectiveRainfall class, 186
  EffectiveRainfall.Approach enum, 231
  EffectiveRainfall.UserDefinedMethod delegate, 231
  EffectiveRainfallUI class, 194
  Overview, 175
  PenmanMonteith class, 195
  RootingDepth class, 200
  RootingDepthUI class, 202
  WaterBalance class, 203
  WaterBalance.SoilWaterDepletionGroup enum, 232
  WaterBalanceUI class, 218
  WPPReport class, 219
  YieldResponseToWater class, 220
  YieldResponseToWaterUI class, 230
WPPReport, 49
WPPReport class, 219
  Remarks, 219
  WRBSoilGroup enumeration, 122
    Members, 123
    Remarks, 123

X, 131

Y, 131
Yield class, 255
  Constructors, 256
  Properties, 257
  Remarks, 256
YieldFormation, 173
YieldReducingCoefLand, 257
YieldReducingCoefWater, 215
YieldResponse, 49
YieldResponseToWater class, 220
  Constructors, 224
  Example, 220
  Properties, 225
  Remarks, 220
YieldResponseToWaterUI class, 230
  Remarks, 230