WetSpa PCRaster-Python
Step-by-step user manual

Elga Salvadore
Department of Hydrology and Hydraulic Engineering
Vrije Universiteit Brussel
VITO: Flemish Institute for Technological Research
July 2013

Extra references:
The WetSpa PCRaster-Python model has been developed by Elga Salvadore for her PhD work. The model is still in a developing phase, no graphical user interface is available and new versions will be released in the coming months. For any technical question and for bugs report, please contact: esalvado@vub.ac.be

- Liu, Y.B., and De Smedt, F., "WetSpa Extension, A GIS-based Hydrological Model for Flood Prediction and Watershed Management. Documentation and User Manual" for an introduction to the model and as a reference for the physically based equations (http://www.vub.ac.be/WetSpa/downloads.html),
- PCRaster Team, "PCRaster Documentation, Release 3.0.1, May 12, 2011" for GIS functions (http://pcraster.geo.uu.nl/support/documentation/), in particular PCRasterPython (used by the model) and Aguila (data visualization).
- basic concepts Python programming can be found: http://www.python.org/
- The numerical library of Python used for WetSpa PCRaster-Python is NumPy: http://www.numpy.org/
- All the spatially-distributed variables of the models are of the type masked arrays (matrices): http://docs.scipy.org/doc/numpy/reference/maskedarray.html

In this manual you can find: (1) an introduction to the WetSpa PCRaster-Python model and to the original WetSpa model, (2) information on installation requirements, folders structure and model components, (3) input requirement and preparation (3.1), description of the GIS preprocess (3.2), description of the Main.py script (3.3), and outputs (3.4), and finally (4) a description of the test cases provided. A more detailed description of the model components will be provided in the next version of the manual.

1. Introduction to the WetSpa PCRaster-Python Model

WetSpa PCRaster-Python Model is a reimplementation of the WetSpa model (Water and energy transfer between Soil plant and atmosphere). The structure and the programming language of the original WetSpa model was modified, the main physical equations solved by both models (original WetSpa model and the PCRaster-Python versions) are essentially the same.
The new structure of the model is modular and written in Python language. Every physically-based process (such as interception, runoff, infiltration, etc.) is coded in a separate module and the modules exchange data during running time through a modeling framework prototype. The user can select which processes will be simulated and the spatial and temporal resolution of each process separately, while the framework takes care of the order of calculation and of data exchange (i.e. no need for manual pre- and post-processing). The new structure allows the user to modify the model complexity according to data availability and case specific requirements in a straightforward way. Moreover different formulations of the same hydrological process can be easily tested as the modules have a standard and interchangeable format.

Original WetSpa model

The WetSpa model is a GIS-based, spatially distributed hydrological model for rainfall-runoff simulations at the catchment scale. It has been developed by the Department of Hydrology and Hydraulic Engineering of the Vrije Universiteit Brussel [Batelaan et al. 1996, De Smedt et al. 2000, Liu et al. 2003, Safari et al. 2012]. The model can simulate several physical processes at the raster cell level, such as: interception, snow melt, depression storage, evapotranspiration, runoff, interflow, groundwater recharge and groundwater flow (at sub-catchment level), see Figure 1 for an overview of the model water balance at cell level. The inputs of the model are: (a) precipitation, potential evapotranspiration and temperature time series, which are distributed over the catchment with the Thiessen polygons method if not available in a distributed way such as radar data; and (b) spatially distributed parameters are derived with help of the GIS software ArcView 3.2 (or ArcGIS 9.3) on basis of three maps: Topography, Soil texture and Land use. Typical outputs are: flow hydrographs at catchment and sub-catchment outlets, maps of evapotranspiration, soil moisture, interception, surface runoff, groundwater recharge and interflow for each time step required by the user.

Surface runoff is produced using a modified coefficient method based on physical characteristics of each raster cell, the runoff is then routed as overland and channel flow with the method of linear diffusive wave approximation. Interflow is computed based on the Darcy's law and the kinematic approximation as a function of the effective hydraulic conductivity and the hydraulic

gradient, while groundwater flow is estimated with the linear reservoir method on small sub-catchment scale as a function of groundwater storage and a recession coefficient. Time step resolution of inputs and outputs ranges from one hour to days, months and years. The model uses eleven physically based global parameters which can be tuned in the calibration phase. Calibration can be manually or automatically performed, the automatic calibration is performed with the model independent Parameter ESTimator (PEST) [Liu et al. 2005], recently other techniques were also experimented, i.e. multi-objective calibration with genetic algorithm [Shafii and de Smedt, 2009].

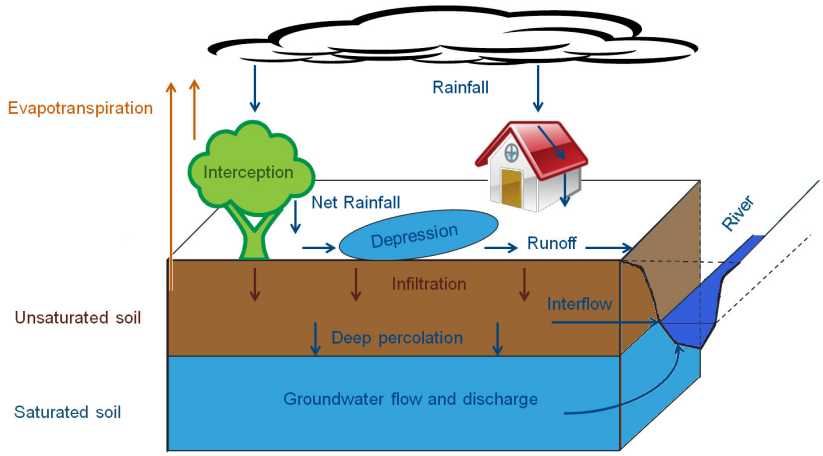

Figure 1: WetSpa model performs at every time step a water balance at cell level, the main considered processes are here illustrated.

Main limitations of the models are:

(a) Values assigned to any raster cell represent an average value over the area of each cell. The greater the variability over the cell, the greater will be the error induced through the use of an average value.
(b) The snow accumulation and snowmelt are modeled in a simple way by the degree-day coefficient method, where the redistribution of snow pack, the influence of aspect, local slope, land use, etc., to the snowmelt are not into account.
(c) The WetSpa model generates runoff by an empirical-based coefficient method rather than from equations more closely representing the physical processes, but it takes into account morphological characteristics of the river basin.
(d) The WetSpa model employ many default parameters, which are interpolated from literature and used over the entire catchment or for single cells. Due to the vast variation

range, parameters such are hydraulic conductivity, roughness coefficient, etc. may change greatly when apply the model to different geographical locations. Therefore model calibration is preferable although this action might brings difficulties in model parameterization in un-gauged river basins.

(e) The WetSpa model simulates groundwater flow on a small sub-catchment scale. It estimates the groundwater flow at each time step, but cannot predict the spatial distribution of groundwater table, as well as its variation during the simulation period.

(f) The WetSpa model assumes that the groundwater table is below the root zone. This constrains the use of the model in wetlands areas.

2. Installation requirements, folder structure and model components

Installation of the WetSpa PCRaster-Python model requires several preinstalled software programs:

- Python version 2.5.4;
- Numpy version 1.3.0;
- PCRaster version 3.0.0 beta.

Instructions on how to install and run Python and/or PCRaster are out of the scope of this manual; nevertheless some specific tips are provided; for more information refer to Python and PCRaster user guides.

To be able to run your Python-PCRaster scripts you need to modify/edit environmental user and system variables.

Control panel → System → Advanced → Environmental Variables

User variables:

- PATH: C:\Program Files\PCRaster\apps
- PYTHONPAH: C:\Program Files\PCRaster\Python

System variables:

- PATH: C:\Python25

If you have more version of Python installed on your computer make sure that in the paths Python25 is listed first. If you do not wish to modify the order of paths in your environmental variables you can explicitly call Python 2.5 when running the model from command prompt or from the Python 2.5 Shall.

The WetSpa PCRaster-Python model is a Python script and it can just be copied and pasted in your project folder. The directory where you wish to copy and run the PCRaster-Python WetSpa

model can be any existing or new directory with *no spaces* in its path. Figure 2 gives a schematic overview of the structure that your project folder should have.

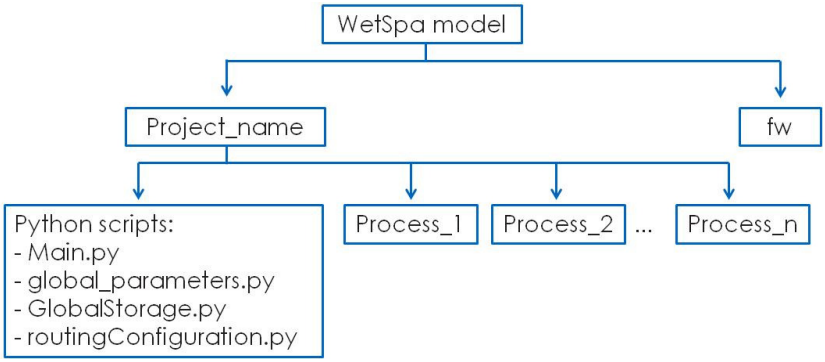

Figure 2: the WetSpaOO model folders structure

The folder ***fw*** contains the modeling framework prototype codes, these codes are responsible for the model component interactions.

Project_name is a generic folder of the modeling project where the Python scripts are stored and where all the model components are saved in a separate folder (Process_1, Process_2, etc.).

Python scripts provided in the Project_name folder are:
- *Main.py*: the one you need to call if you want to run the model. In this script the modeling framework prototype modules are imported together with the simulation modules, the WetSpa PCRaster-Python model is created and components are added from the simulation modules, links are established between components and finally the model can run.
- *global_paprameters.py*: location where all the global model parameters are stored. Global model parameters are used to calibrate the model and are selected by the user either manually or automatically.
- *GlobalStorage.py*: is the soil moisture component;
- *routingConfiguration.py*: location where routing configuration parameters are stored, in particular: maximum length of the iuh (maxt), total number of simulation time steps (lenp), and number of sub-catchments (nsub). The values of these parameters are calculated by the model as a preprocess and are accessed by several model components.

Available model components in the Project_name folder are:
- ***Preprocess***: module to derive all the necessary input parameters from three basic maps (elevation, soil and land use). This component is separated into two script: *static_preproc.py* and *dynamic_preproc.py*;

- ***IUH***: module to calculate the instantaneous unit hydrograph of each cell to the outlet of the catchment (uh_cell), the instantaneous unit hydrograph of each cell to the sub-catchment outlet (iuh_sub) and the river iuh (iuh_link);
- ***rain*****:** module to generate rainfall maps (v_rain) based on time series and Thiessen polygons;
- ***pet***: module generate actual evapotranspiration maps (v_pet) based on time series of potential evapotranspiration and Thiessen polygons;
- ***interception***: module to calculate interception (v_interc), interception storage (s_int), evaporation from interception storage (v_ei), net precipitation (v_netp) maps;
- ***RainExc_Infil***: module to calculate surface runoff (v_rs), and infiltration (v_infil) maps;
- ***depression***: module to calculate depression (v_depre), depression storage (s_dep), extra infiltration from depression storage, and reduced surface runoff after filling the depressions;
- ***Interflow***: module to calculate interflow maps (v_ri);
- ***evap_soil***: module to calculate maps of evapo(traspi)ration from the unsaturated zone (v_es);
- ***gw_recharge***: module to calculate deep infiltration maps (v_perco);
- ***Routing***: module to calculate the contribution of surface runoff (qs) and interflow (qi) to the hydrograph at the catchment outlet based on the iuh routing.
- ***balance***: module to calculate the catchment and sub-catchment averages of many hydrological variables;
- ***gw_routing***: module to calculate the base flow (qg) at the catchment outlet, absed on the linear reservoirs method;
- ***total_discharge***: module to sum up the different components of the hydrograph (qs, qi, qg);
- ***Output***: module to write on file global outputs (balance.txt and discharge.txt).

The GIS preprocess is subdivided into two scripts the *static_preproc.py* and the *dynamic_preproc.py*, both included in the folder Preprocess (Fig.3). The purpose of these two script is to allow coupling of the WetSpa PCRaster-Python mode with a land use change model. The static_precroc.py is executed only once at the start of the simulation and provides parameter maps in function of elevation and soil types, while the dynamic_preproc.py runs with the same time step resolution as the dynamic land use change model. Other Python scripts included in the Preprocess folder are:
- *CloneInfo.py*: used to store information on the Clone map. Information includes coordinates x and y of the lower left corner, cell size in meters, and number of rows and columns. These information are used by both static and dynamic preprocess. The values are automatically provided by the model (no user inputs required).
- *UserInputs.py*: all the necessary inputs thresholds and option selection needed to automatically perform the GIS preprocess.

Inside the Preprocess folder, four sub-folders are provided:
- **ascii**: contains all the output maps converted into ASCII format;
- **basic_maps**: contains the five necessary input maps: two clone maps, elevation, soil type and land use;
- **PCRmaps**: contains all the output maps in PCRaster format;
- **tables**: contains model lookup tables, where parameter values are stored (PCRaster tables .tbl)

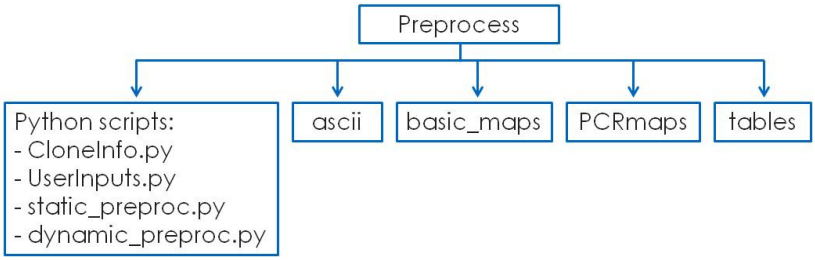

Figure 3: sub-folder structure of the Preprocess module

3. Prepare you own WetSpa PCRaster-Python model step by step

This chapter helps you in creating your WetSpa PCRaster-Python model. The first part discusses the data requirements and their preparation, the second describes the GIS preprocess, the third part describes Main.py script and how to run the model, and the last part discusses the outputs.

3.1 Inputs requirements and preparation

Input1: meteorological time series.
The WetSpaOO model requires meteorological data for dynamic simulations. These data can be in the form of time series or as a stack of maps (radar data).
Time series of precipitation (in mm) and potential evapotranspiration need to be prepared according to Fig.4 and they have to be saved as .txt files.
"p.txt" (precipitation time series) is saved in the folder *rain*, while "pet.txt" is saved in the folder *pet*.
The coordinates of the measurement stations are used to compute Thiessen polygons in case of multiple stations. These coordinates need to be saved in the folder *tables* as precipitation_stations.tbl and evapotranspiration_stations.tbl (simple text files). The two tables can be found in Preprocess/tables/ and they both have a fix structure with no header:
X coordinate Y coordinate IDn

Where IDn is a unique value for each station, this number should correspond to the column in the input files that contains data for that station (p.txt or pet.txt).
You have several options to create the tables, the easiest is to modify the sample tables which are provided in the sub-folder tables, but you can also create them with excel, any text editor or PCRaster.

year	month	day	hour	ID1	ID2
1960	1	1	0	0	0.1
1960	1	2	0	0.8	0.6
1960	1	3	0	1.2	0.9
1960	1	4	0	0.3	1.2
1960	1	5	0	0.1	0.8

Figure 4: Example of p.txt or pet.txt

Input2: Global parameters.
An initial set of global parameters needs to be selected according to your study area. To do so, modify global_parameters.py. For more information on the physical meaning of the global parameters refer to the WetSpa manual.

Input3: parameter maps.
Spatially distributed inputs need to be created for the GIS preprocess components.
In the sub-folder Preprocess/basic_maps you need to input three maps:

- DEM: digital elevation model;
- Land use map;
- Soil type map;

A Scalar Clone map is also necessary to provide information regarding geographical and cartographical location attributes of all the other maps (for more information the reader is referred to the PCRaster manual).

The DEM, land use and soil type maps are the base maps for deriving all model parameters. These maps are in raster format with the same cell size and the same spatial extent. They can be prepared with any type of GIS software but they need to be converted to PCRaster format before being stored in the sub-folder basic_maps. The names of these maps must be: elevation_start, landuse_start, soil_start.[1]

[1] Make sure the names are correct otherwise you will not be able to run WetSpa pre-process. Remember: Python is case sensitive!

More info on Clone scalar can be found in the next section (2.1.1 convert ArcView ASCII maps to PCRaster maps).
Soil and Land Use classification should be made according to the tables 3.1 and 3.2 in the WetSpa manual. If your catchment is not correctly described by these classes, you can always define your own classes as long as all the necessary parameters are known (see tables 3.1 and 3.2 in the WetSpa manual). If you decide to do so, the parameter tables in .../Project_name/Preprocess/tables need to be adapted.

Input4: thresholds and option selection.
The WetSpaOO GIS preprocess involves the use of several GIS functions (PCRaster functions), many of them require the use of case-specific parameters and thresholds. To automatically perform the calculation all these thresholds and options have to be selected before the start of the simulations. The used can do so by modifying the UserInput.py file.

Convert ArcView ASCII maps to PCRaster maps
When you want to convert and ArcView map into a PCRaster map you need to remember that these 2 programs use different conventional location attributes, i.e. the coordinates of the map corner (for ArcView is the lower left -ll- and for PCRaster is the upper left -UL-)[2]:

yUL = yllcorner + (nrows * cellsize) and xUL = xll

To convert your map to PCRaster format you need to create a clone map which implies the knowledge of the following data:

- Data type: scalar
- Projections: x coordinates always increase from left to right, y coordinates increase from top to bottom or from bottom to top (to chose);
- yUL and xUL;
- cell length;
- number of rows and columns;
- angle in between the horizontal direction on the PCRaster map and the x axis of the real world coordinate system (normally 0).

The clone map can be created using the PCRaster operator *mapattr* in the DOS window (command prompt).

[2] For any other GIS software please check carefully the coordinate system which might be different and so the way of converting from and to.

Once you have reached your project folder type: `mapattr CloneScalar`

a small program will start and it will ask you all the necessary parameters to create your clone map.
To adjust later some of the map attributes type: `mapattr -e CloneScalar`

Save the file CloneScalar in the sub-folder basic_maps.

Now you are finally ready to convert your ASCII map. Here is an example to convert your elevation map:

```
asc2map -a "C:\Oslovak\WetSpa\ArcView\Data\elevation.asc" elevation --clone CloneScalar
```

Converting back is possible:

```
map2asc -m -9999 elevation elevationCr.asc
```

However, the ASCII file is not completely the same as the original one, due to rounding issues. By adding `-f 10.2f` you can select your rounding options (10 numbers before the comma, 2 behind in this example).

A CloneScalar is needed for the elevation map (scalar values), for land use and soil maps you need to create a different clone of the type Nominal as values in these maps are integers (classes).

3.2. WetSpa GIS preprocess (PCRaster)

For users which are familiar with the WetSpa preprocess made via ArcView 3.2:
the main difference of this version is that all the necessary steps for the preparation of the parameter maps are done automatically in the correct order and all you need to do is to prepare the UserInput.py script (Fig.3). The calculations are divided into land use and non-land use dependent. Few calculations have been modified, e.g. slope calculation, stream order, for more details check the rest of the manual.

General info:

- at the end of every calculation, the parameter map will be saved in the PCRmaps folder;
- the necessary maps for WetSpa simulations will be automatically converted to ASCII format and saved in the sub-folder ascii;

- all the thresholds/parameters/choices of calculation procedure you will select to run WetSpa pre-process are stored in UserInput.py, also the one that are changed during running time by the user (for example, parameters for flow direction).

Preliminary task (in the Python shell)

1) Set your working directory: a question will be asked at the start of the simulation. The expected directory is the path till Project_name (Fig.2)
2) In case some parameters (in the UserInput.py) are not properly chosen, you will be asked to change them;
3) In the shell or in the command prompt window, you will receive information related with your clone map, check them carefully before you start!

Topography based parameters (in static_preproc.py)

1) *Mask:* the script will run a function to generate a mask of your catchment which will be used for several other calculation;
2) *Flow direction:* 4 parameters are necessary for the calculation of flow direction (core depth, core volume, core area, catchment precipitation). These parameters are used to fill possible sinks/pits (for more information: PCRaster manual function lddcreate()).
   Ones you have chosen the set of parameters, the model will calculate a flow direction map and it will count the number of pits that are present. Depending on the type of application, it can be satisfactory to have more than one pit but for most of the hydrological calculations you need to reduce the number of pits/outlets to 1. To do so you can adjust the value of your parameters till you reach the number of pits/outlet you want. If your first choice of parameters will produce a flow direction map with only 1 outlet no information about the number of pits will be given.
3) *Flow accumulation:* no user input necessary.
4) *Stream network:* set a cell threshold value when extracting streams from the flow accumulation map upon the basin characteristics.
5) *Stream order:* this function creates a stream order map following the scheme of Strahaler[3], which will be used for interpolating the channel roughness coefficient. No user input necessary.
6) *Slope:* this function will create a slope grid for both overland flow areas and river channels using the third-order finite difference method, yielding a value between (0, 1)[4].

[3] The previous version of WetSpa pre-process (ArcView 3.2) uses the method of Shreve for calculating stream order map.

[4] In the previous version of WetSpa pre-process (ArcView 3.2) the slope map has percentage values.

If necessary, the user can set a minimum slope threshold in the UserInput.py. The area with a slope less than the threshold can be saved in a separate map for inspection.

7) *Hydraulic radius:* this function will create a hydraulic radius grid based on the flow accumulation map. Select a return period for the flood under consideration: 2 years, 10 years, and 100 years typing T2 or T10 or T100. Generally, a flood frequency of 2-year return period is chosen for normal floods. The 2 parameters a and b for calculating hydraulic radius can be edited in the lookup tables: radius_T2.tbl, radius_T10.tbl and radius_T100.tbl in the sub-folder tables.
8) *Stream links:* this function will create a stream links grid. The user needs to input a cell threshold value to extract stream from flow accumulation. This value may be much higher than the threshold value for delineating the stream network grid. The stream links map will be used to calculate the sub-watersheds grid, so by adjusting the threshold value, different numbers of sub-catchments can be derived.
9) *Subwatersheds:* this function will create a sub-catchments grid used for semi-distributed modeling and for the simulation of groundwater balance using the stream links map. No user input necessary.

Soil based parameters in static_preproc.py

1) *Conductivity:* this function will create a conductivity grid based on the lookup table conductivity.tbl in the sub-folder tables. No user input necessary.
2) *Porosity:* this function will create a porosity map based on the lookup table posority.tbl in the sub-folder tables. No user input necessary.
3) *Field capacity:* this function will create a filed capacity map based on the lookup table fieldcap.tbl in the sub-folder tables. No user input necessary.
4) *Residual moisture:* this function will create a residual soil moisture map based on the lookup table residual.tbl in the sub-folder tables. No user input necessary.
5) *Pore distribution index:* this function will create a pore distribution index map based on the lookup table poreindex.tbl in the sub-folder tables. No user input necessary.
6) *Wilting point:* this function will create a wilting point map based on the lookup table wiltingpoint.tbl in the sub-folder tables. No user input necessary.
7) *Initial soil moisture:* this function will create an initial relative saturation grid of the soil using the method of the Topographical Wetness Index. A minimum ration reflecting the driest cells can be set in the UserInput.py script.

Land use based parameters in dynamic_preproc.py

1) *Root depth:* this function will create a root depth grid based on the lookup table rootdepth.tbl in the sub-folder tables. No user input necessary.

2) *Interception capacity:* this function will create a yearly maximum and minimum interception capacity grids respectively based on the lookup tables interc_max.tbl and interc_min.tbl in the sub-folder tables. No user input necessary.
3) *Manning coefficient:* this function will create a Manning's coefficient grid for both land surface and river channel based on the geographical information of land use and stream order. Three methods are available to calculate Manning's coefficient in WetSpa:
   (1) Interpolation among different stream orders: this method defines channel Manning's n based on the stream orders with the lower values downstream and higher value upstream. A maximum and a minimum Manning's n value are asked to determine corresponding to the lowest and highest stream order. Default values are:
   Minimum Manning's value: 0.025
   Maximum Manning's value: 0.055
   (2) Remain the default constant as in the lookup table : this method defines a constant Manning's n for the river channel using the values assigned in the lookup table manning.tbl
   (3) Change to another constant: this method defines a constant Manning's n for the river channel using a user input value.

   The user can choose the proper method for its application by selecting 1, 2 or 3 in the UserInput.py file.

Potential runoff coefficient and depression storage capacity in dynamic_preproc.py

1) *Runoff coefficient:* this function will create a potential runoff coefficient map based on the combination of slope, land use and soil type maps. The user needs to set an impervious percentage for urban cells depending upon the main urban land use category and the cell size. The default value is 30% for a grid with cell size 100X100 m. The impervious percentage can be selected in the UserInput.py. An imperviousness map of the study area can be used instead, if available. This option has to be selected in the UserInput.py file and the map of imperviousness has to be saved as a PCRaster map (.../Project_name/Preprocess/basic_maps/imperviousness_start).
2) *Depression storage capacity:* this function will create a depression storage capacity map based on the combination of slope, land use, soil type maps and impervious percentage of urban cells (same as previous port for spatially distributed imperviousenss).

Flow routing parameters in dynamic_preproc

1) *Velocity:* this function will create a flow velocity grid based on Manning's n, hydraulic radius and slope map. Velocity threshold values can be selected by the user in the UserInput.py, the default values:

Minimum velocity value: 0.005 m/s
Maximum velocity value: 3.0 m/s
The flow velocity is set to the upper limit when the calculated velocity is higher than this limit and to the lower limit when the calculated velocity is higher than this lower limit.

2) *T0_h:* this function will create a mean flow travel grid in hours from each cell to the catchment outlet using the weighted ldddist() function (for more information check the PCRaster manual). No user input necessary.
3) *Delta_h:* this function will create a standard deviation grid of the flow travel times from each cell to the catchment outlet. No user input necessary.
4) *T0_s:* this function will create a mean flow travel time grid in hours from each cell to its sub-catchment outlet. No user input necessary.
5) *Delta_s:* this function will create a standard deviation grid of the flow travel time from each cell to its sub-catchment outlet. No user input necessary.

Thiessen polygons in static_preproc.py

Thiessen polygons maps will be created for precipitation, evapotranspiration and optionally for temperature in an automatic way based on the catchment mask and the coordinates of the measurement stations.

If one or more of the hydro-meteorological station is located outside the basin, WetSpa pre-process will create a temporary clone map to include those stations. Before the creation of this clone map, the number of necessary rows and columns is calculated and is shown on the python shell. The user has the choice to continue or abort the process.

Reason of this control: if an error is present in the table of the coordinates the temporary clone map could reach unwanted huge dimensions, exciding the storage capacity of your computer. So check carefully the number of rows and columns before continue!

3.3 Main.py

The final steps for running WetSpa PCRaster-Python model is (1) to modify your *Main.py* according to your case study, and (2) if necessary adapt the process components.

To run the model you can simply type in you command prompt window:

`python Main.py` (if Python 2.5 is properly listed in your environmental variables);

`C:\Python25\python.exe Main.py` (if you have other versions of Python installed on your computer, or if you could not modify the order of elements in your environmental variables).

If you want to run the model from the Python Shall:

right click on Main.py → open with IDLE →F5

An example of Main.py file:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# libraries to import
import sys
import os
import datetime

import numpy
import PCRaster
import random

sys.path.insert(0, os.path.abspath("../fw"))
sys.path.insert(0, os.path.abspath("rain"))
sys.path.insert(0, os.path.abspath("interception"))
sys.path.insert(0, os.path.abspath("RainExc_Infil"))
#[...]
sys.path.insert(0, os.path.abspath("gw_routing"))

import GlobalStorage
import scheduler
import modelSpecification

import static_preproc
import landuse
import dynamic_preproc
import IUH_standard
import Rain
import PET
import Interception
import RainExcess_Infiltration
#[...]
import outputs
```

```
29 s1 = datetime.datetime(1995, 1, 1, 0)
30 start = datetime.datetime(1995, 1, 1, 1)
31 end   = datetime.datetime(2005, 1, 1)
32 deltaTHyd     = datetime.timedelta(days=1)
33 deltaTLanduse = datetime.timedelta(days=365)
34 deltaTStatic  = datetime.timedelta(days=10065)

35 myGlobal = GlobalStorage.Storage()

36 WetSpaOO_model = modelSpecification.ModelSpecification()
37 StaticPreprocComponent = static_preproc.s_preproc(s1,end,
   deltaTStatic)
38 LuseComponent = landuse.landuse_change(s1,end, deltaTLanduse)
39 DynamicPreprocComponent = dynamic_preproc.d_preproc(s1,end,
   deltaTLanduse)
40 IUHComponent = IUH_standard.IUH(s1,end, deltaTLanduse)
41 RainComponent = Rain.Rain_dis(start,end, deltaTHyd, False)
42 #[...]
43 OutputComponent = outputs.output(start,end, deltaTHyd)

44 WetSpaOO_model.addComponent(StaticPreprocComponent)
45 WetSpaOO_model.addComponent(LuseComponent)
46 WetSpaOO_model.addComponent(RainComponent)
47 WetSpaOO_model.addComponent(IntercComponent)
48 WetSpaOO_model.addComponent(RainExcInfilComponent)
49 #[...]
50 WetSpaOO_model.addComponent(OutputComponent)

51 WetSpaOO_model.link(StaticPreprocComponent, "ma_mask", IUHComponent,
   "mask")
52 WetSpaOO_model.link(RainComponent, "v_rain", IntercComponent,
   "v_rain")
53 WetSpaOO_model.link(IntercComponent, "v_netp",
   RainExcInfilComponent, "v_netp")
54 #[...]
55 WetSpaOO_model.link(TotaldisComponent, "q_tot", OutputComponent,
   "q_tot")

56 single = scheduler.SingleRun(WetSpaOO_model)
57 single.run()
```

Lines 4-9: import external libraries

Lines 10-15: insert paths to WetSpa PCRaster-Python modules

Lines: 16-28: import WetSpa modules

Lines 29-34: set start, end and time step of WetSpa PCRaster-Python modules

Line 35: soil moisture storage

Line 36: creation of the model

Lines 37-43: creation of model components (processes)

Lines 44-50: adding model components to the model

Lines 51-55: creation of the links between components

Lines 56 and 57: model run

In theory every process (model component) can have different start, end and time step, however the coupling of processes with different time step resolution should be done carefully otherwise errors might arise.

Model components are created by defining their start, end and time step (lines 33-38). Some components require one extra Boolean parameter. This parameter is responsible for the generation of spatially distributed outputs. When the parameter is set to True, the model component will generate one or more stocks of maps in the correspondent folder. For example, the component interception can generate maps, for every time step, of net precipitation ('netP'), interception flux ('interc'), evaporation from interception storage ('v_ei') and interception storage content ('s_int'). Values are expressed in [mm].

If you want to remove certain model components make sure that the variables requests of other components are satisfied. Exchange of variables is done in lines 55-59 by creating links within components. For example:

```
WetSpaOO_model.link(IntercComponent, "v_netp", RainExcInfilComponent, "v_netp")
```

In this line of code the component Interception is exchanging the net precipitation to the runoff calculation component. Links can be established if the variables are properly initialized in the respective component. In this case, in the file Interception.py you will find this lines of code:

```
self.v_netp = ma.array(np.zeros(np.shape(self.mask)), mask = self.mask)
```

`self.addOutputValue(self.v_netp)`

Line a initialize the variable as a matrix filled with zeros, the second line tells the model v_netp will be an output for Interception.py to some components.

In RainExcess_Infiltration.py file you will find that this variable is set as an input value:

```
self.v_netp = ma.array(np.zeros(np.shape(self.mask)), mask = self.mask)
```

`self.addInputValue(self.v_netp)`

3.4 Outputs

Standard outputs can be found in the Output folder. Two outputs files are created in this folder: balance.txt and discharge.txt.

In balance.txt you can find the catchment average of several water fluxes or storages per time step expressed in [mm]: (1) rain, (2) interc = interception, (3) s_mo = soil moisture of the unsaturated zone, (4) infil = infiltration, (5) evap = total evapotranspiration, (6) perc= percolation from unsaturated zone to the saturated zone, (7) RS = surface runoff, (8) RI = interflow, (9) RG = base flow, (10) R = RS + RI + RG.

In discharge.txt you can find the components of the flow hydrograph at the outlet of the catchment expressed in [m^3/s]: (1) qs = direct surface runoff, (2) qi = discharge from interflow, (3) qs = baseflow.

If requested by the user, spatially-distributed outputs can be found in the correspondent component folders, see page 6.

4. Test cases

Three test cases are provided:

1) 5by5: a simple theoretical example of a catchment composed by 5 row and 5 columns (cell size = 1 km). All the standard processes are included in this example with a time step of 1 hour.

2) V_catchment: Two-dimensional areal surface flow using the rainfall-runoff example proposed by Di Giammarco [Di Giammarco et al., 1996].

Problem characteristics:

- Two-dimensional surface flow from a totally impervious V-catchment (see Figure 3 for a graphical representation of the catchment);
- Precipitation intensity: $3 \cdot 10^{-6}$ m/s for 90 minutes followed by 90 minutes of no rain;
- Due to symmetry only half of the catchment is simulated with cell size of 10 by 10 m (simulation domain 1000 m by 800 m);
- Surface slope: 0.05 and 0.02 perpendicular to, and parallel to the channel respectively;
- Manning's roughness coefficient of 0.15 and 0.015 are applied to the slopes and the channel, respectively;
- Simulation time step: 1 min.

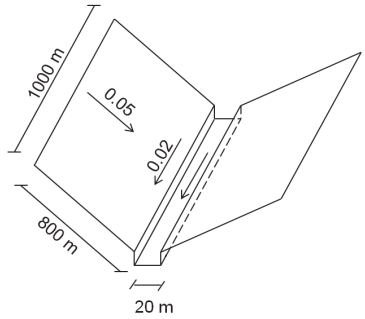

Figure 5: V-cathcment, modified from Di Giammarco et al., [1996].

In the V-catchment, sub-surface processes do not occur (fully impervious), thus to perform WetSpa PCRaster-Python simulations some pre-process parameters are set to zero (soil moisture, porosity, conductivity, and depression storage capacity), the processes of depression storage, interflow, evapotranspiration from soil, groundwater recharge and groundwater flow routing are deactivated.

3) Kleine_Nete: real case study. The Kleine Nete is a catchment located in Belgium with a surface of about 600 km^2. Data are provided with spatial resolution of 50 m and temporal resolution of 1 day. This example serves to demonstrate the coupling of components with different time steps. A test land use change component is provided with a time step of 1 year. The component

reads land use maps but it could easily be substituted with a model generating dynamic land use maps.

Bibliography

Batelaan, O., Wang, Z. M. & De Smedt, F., 1996. An adaptive GIS toolbox for hydrological modelling. *Application of geographic information systems in hydrology and water resources management, IAHS Publ.,* pp. 3-9.

Berezowki, T. et al., 2012. Impact of remotely sensed land-cover proportions on urban runoff production. *International Journal of Applied Earth Observation and Geoinformation,* Volume 16, pp. 54-65.

Chormanski, J. et al., 2008. Improving Distributed Runoff Prediction in Urbanized Catchments with Remote Sensing based Estimates of Impervious Surface Cover. *Sensors,* 8(2), pp. 910-932.

De Smedt, F., Liu, Y. B. & Gebremeskel, S., 2000. Hydrological modeling on a catchment scale using GIS and remote sensed land use information. In: C. A. Brebbia, ed. *Risk Analysis II.* Southampton, Boston: s.n., pp. 295-304.

Di Giammarco, P., Todini, E., and Lamberti, P., A conservative finite elements approach to overland flow: the control volume finite element formulation. Journal of Hydrology, 175, 267-291, 1996.

Liu, Y. B. et al., 2005. Automated calibration applied to a GIS-based flood simulation model using PEST. *Floods, from defence to management,* pp. 317-326.

Liu, Y. B. et al., 2003. A diffusive transport approach for flow routing in GIS-based flood modeling. *Journal of Hydrology,* 283(1-4), pp. 91-106.

Safari, A., De Smedt, F. & Moreda, F., 2012. WetSpa model application in the Distributed Model Intercomparison Project (DMIP2). *Jornal of Hydrology,* Volume 418, pp. 78-89.

Shafii, M. & De Smedt, F., 2009. Multi-objective calibration of a distributed hydrological model (WetSpa) using a genetic algorithm. *Journal of Hydrology and Earth Sciences,* 13(11), pp. 2137-2149.

Verbeiren, B. et al., 2013. Assessing urbanisation effects on rainfall-runoff using a remote sensing supported modelling strategy. *International Journal of Applied Earth Observation and Geoinformation,* Volume 21, pp. 92-102.