wflow Documentation

Jaap Schellekens

Sep 28, 2022

CONTENTS

Note: There will be no further developments in the Python wflow framework (bugfixes are possible), and the documentation is no longer updated. Developments continue in the Julia package Wflow, available here, including documentation.

Note: This documentation was generated on Sep 28, 2022

Documentation for the development version: https://wflow.readthedocs.org/en/latest/

Documentation for the stable version: https://wflow.readthedocs.org/en/stable/

CHAPTER

ONE

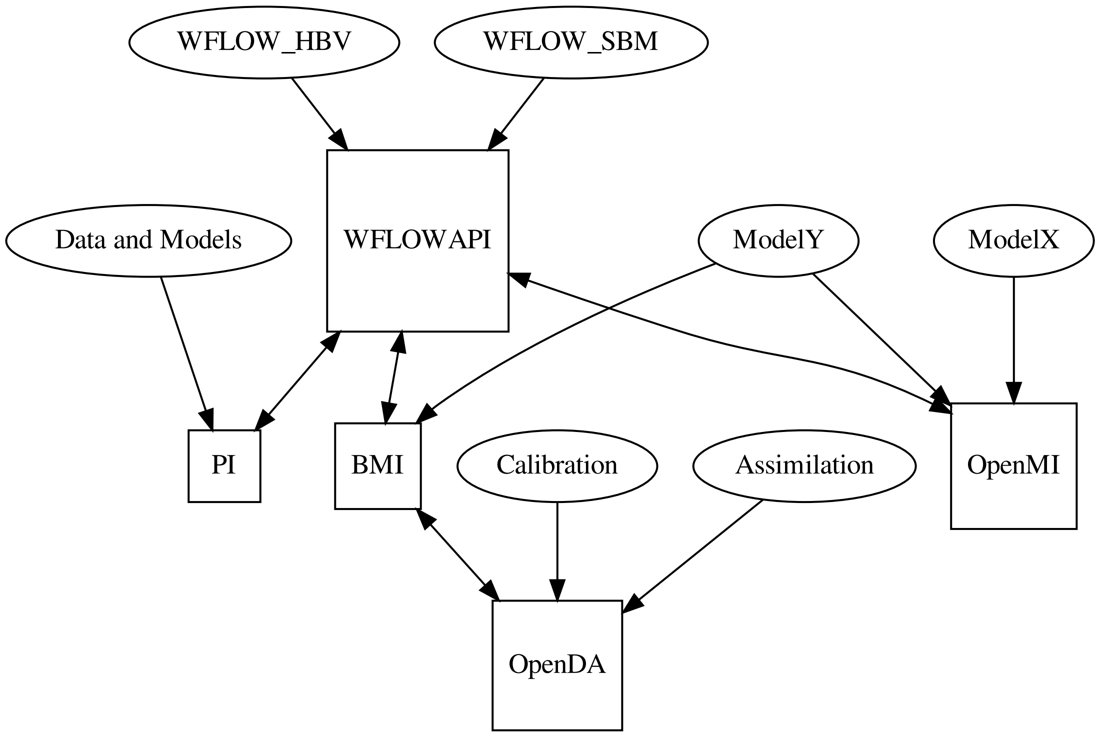
INTRODUCTION

This document describes the wflow distributed hydrological modelling platform. wflow is part of the Deltares' OpenStreams project. Wflow consists of a set of python programs that can be run on the command line and perform hydrological simulations. The models are based on the PCRaster python framework (www.pcraster.eu). In wflow this framework is extended (the `wf_DynamicFramework` class) so that models build using the framework can be controlled using the API. Links to BMI and OpenDA (www.openda.org) have been established. All code is available at github (https://github.com/openstreams/wflow/) and distributed under the GPL version 3.0.

The wflow distributed hydrological model platform currently includes the following models:

- the wflow_sbm model (derived from topog_sbm)
- the wflow_hbv model (a distributed version of the HBV96 model).
- the wflow_gr4 model (a distributed version of the gr4h/d models).
- the wflow_W3RA and wflow_w3 models (implementations and adaptations of the Australian Water Resources Assessment Landscape model (AWRA-L))
- the wflow_topoflex model (a distributed version of the FLEX-Topo model)
- the wflow_pcrglobwb model (PCR-GLOBWB (PCRaster Global Water Balance, v2.1.0_beta_1))
- the wflow_sphy model (SPHY (Spatial Processes in HYdrology, version 2.1))
- the wflow_stream model (STREAM (Spatial Tools for River Basins and Environment and Analysis of Management Options))
- the wflow_routing model (a kinematic wave model that can run on the output of one of the hydrological models optionally including a floodplain for more realistic simulations in areas that flood).
- the wflow_wave model (a dynamic wave model that can run on the output of the wflow_routing model).
- the wflow_floodmap model (a flood mapping model that can use the output of the wflow_wave model or de wflow_routing model).
- the wflow_sediment model (an experimental erosion and sediment dynamics model that uses the output of the wflow_sbm model).
- the wflow_lintul model (rice crop growth model LINTUL (Light Interception and Utilization))

The low level api and links to other frameworks allow the models to be linked as part of larger modelling systems:



Note: wflow is part of the Deltares OpenStreams project (http://www.openstreams.nl). The OpenStreams project is a work in progress. Wflow functions as a toolkit for distributed hydrological models within OpenStreams.

Note: As part of the eartH2Observe project global dataset of forcing data has been compiled that can also be used with the wflow models. A set of tools is available that can work with wflow (the wflow_dem.map file) to extract data from the server and downscale these for your wflow model. Check https://github.com/earth2observe/downscaling-tools for the tools. A description of the project can be found at http://www.earth2observe.eu and the data server can be access via http://wci.earth2observe.eu

The different wflow models share the same structure but are fairly different with respect to the conceptualisation. The shared software framework includes the basic maps (dem, landuse, soil etc) and the hydrological routing via the kinematic wave. The Python class framework also exposes the models as an API and is based on PCRaster/Python.

The wflow_sbm model maximises the use of available spatial data. Soil depth, for example, is estimated from the DEM using a topographic wetness index . The model is derived from the CQflow model (Köhler et al., 2006) that has been applied in various countries, most notably in Central America. The wflow_hbv model is derived from the HBV-96 model but does not include the routing functions, instead it uses the same kinematic wave routine as the wflow_sbm model to route the water downstream.

The models are programmed in Python using the PCRaster Python extension. As such, the structure of the model is transparent, can be changed by other modellers easily, and the system allows for rapid development.

1.1 Installation

1.1.1 Install as a conda package

By far the easiest way to install wflow, is using the `conda` package manager. This package manager comes with the Anaconda Python distribution. `wflow` is available in the conda-forge channel. To install you can use the following command:

- `conda install -c conda-forge wflow`

If this works it will install wflow with all dependencies including Python and PCRaster, and you skip the rest of the installation instructions.

1.1.2 Installing Python and PCRaster dependencies

The main dependencies for wflow are an installation of Python 3.6+, and PCRaster 4.2+. Only 64 bit OS/Python is supported.

Installing Python

For Python we recommend using the Anaconda Distribution for Python 3, which is available for download from https://www.anaconda.com/download/. The installer gives the option to add `python` to your `PATH` environment variable. We will assume in the instructions below that it is available in the path, such that `python`, `pip`, and `conda` are all available from the command line.

Note that there is no hard requirement specifically for Anaconda's Python, but often it makes installation of required dependencies easier using the conda package manager.

Installing pcraster

- If you are using conda, pcraster will be installed automatically in the section below, otherwise:
- Download pcraster from http://pcraster.geo.uu.nl/ website (version 4.2+)
- Follow the installation instructions at http://pcraster.geo.uu.nl/quick-start-guide/

1.1.3 Install as a conda environment

The easiest and most robust way to install wflow is by installing it in a separate conda environment. In the root repository directory there is an `environment.yml` file. This file lists all dependencies. Either use the `environment.yml` file from the master branch (please note that the master branch can change rapidly and break functionality without warning), or from one of the releases {release}.

Run this command to start installing all wflow dependencies:

- `conda env create -f environment.yml`

This creates a new environment with the name `wflow`. To activate this environment in a session, run:

- `activate wflow`

For the installation of wflow there are two options (from the Python Package Index (PyPI) or from Github). To install a release of wflow from the PyPI (available from release 2018.1):

- `pip install wflow=={release}`

To install directly from GitHub (from the HEAD of the master branch):

- `pip install git+https://github.com/openstreams/wflow.git` or from Github from a specific release:

- `pip install git+https://github.com/openstreams/wflow.git@{release}`

Now you should be able to start this environment's Python with `python`, try `import wflow` to see if the package is installed.

More details on how to work with conda environments can be found here: https://conda.io/docs/user-guide/tasks/manage-environments.html

If you are planning to make changes and contribute to the development of wflow, it is best to make a git clone of the repository, and do a editable install in the location of you clone. This will not move a copy to your Python installation directory, but instead create a link in your Python installation pointing to the folder you installed it from, such that any changes you make there are directly reflected in your install.

- `git clone https://github.com/openstreams/wflow.git`
- `cd wflow`
- `activate wflow`
- `pip install -e .`

Alternatively, if you want to avoid using `git` and simply want to test the latest version from the `master` branch, you can replace the first line with downloading a zip archive from GitHub: https://github.com/openstreams/wflow/archive/master.zip

1.1.4 Install using pip

Besides the recommended conda environment setup described above, you can also install wflow with `pip`. For the more difficult to install Python dependencies, it is best to use the conda package manager:

- `conda install numpy scipy gdal netcdf4 cftime xarray pyproj numba python-dateutil`

Then install a release {release} of wflow (available from release 2018.1) with pip:

- `pip install wflow=={release}`

If you want to avoid using `conda`, an example of a PCRaster build and pip install on Ubuntu Linux can be found in issue #36.

1.1.5 Check if the installation is successful

To check it the install is successful, go to the examples directory and run the following command:

- `python -m wflow.wflow_sbm -C wflow_rhine_sbm -R testing`

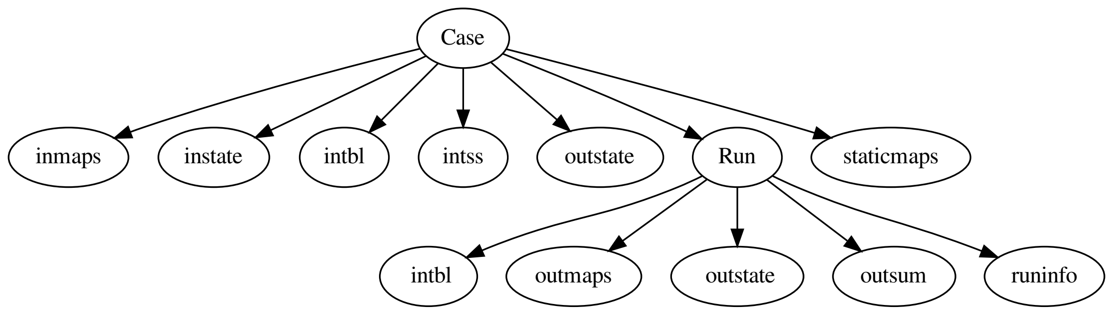
This should run without errors.

1.2 How to use the models

1.2.1 Using the models

Directory structure: cases and runs

A case is a directory holding all the data needed to run the model. Multiple cases may exist next to each other in separate directories. The model will only work with one case at the time. If no case is specified when starting the model a default case (default_sbm or default_hbv) is assumed. Within a case the model output (the results) are stored in a separate directory. This directory is called the run, indicated with a runId. This structure is indicated in the figure below:



If you want to save the results and not overwrite the results from a previous run a new runId must be specified.

inmaps
: Directory holding the dynamic input data. Maps of Precipitation, potential evapotranspiration and (optionally) temperature in pcraster mapstack format.

instate
: Directory holding the input initial conditions. Can be used to hotstart the model. Alternatively the model can start with default initial conditions but in that case a long spinup procedure may be needed. This is done using the -I command-line option.

intbl
: Directory holding the lookup tables. These hold the model parameters specified per landuse/soiltype class. Note that you can use the -i option to specify an alternative name (e.g. to support an alternative model calibration). Optionally a .tbl.mult file can be given for each parameter. This file is used after loading the .tbl file or .map file to multiply the results with. Can be used for calibration etc.

intss
: Directory holding the scalar input timeseries. Scalar input data is only assumed if the ScalarInput entry in the ini file is set to 1 (True).

outstate
: Directory holding the stat variable at the end of the run. These can be copied back to the instate directory to have the model start from these conditions. These are also saves in the runId/outstate directory

run_default
: The default name for a run. if no runId is given all output data is saved in this directory.

staticmaps
: Static maps (DEM, etc) as prepared by the wflow_prep script.

wflow_sbm|hbv.ini
: The default settings file for wflow_sbm of wflow_hbv

Running the model

Overview

In general the model is run from the dos/windows/linux command line. Based on the system settings you can call the wflow_[sbm|hbv].py file directly or you need to call python with the script as the first argument e.g.:

```
python -m wflow.wflow_sbm -C myCase -R calib_run -T 365 -f
```

In the example above the wflow_sbm model is run using the information in case myCase storing the results in runId calib_run. A total to 365 timesteps is performed and the the model will overwrite existing output in the calib_run directory. The default .ini file wflow_sbm.ini located in the myCase directory is read at startup.

Command-line options

The command line options for wflow_sbm are summarized below, use wflow_sbm -h to view them at the command line (option for other models may be different, see their respective documentation to see the options):

```
wflow_sbm [-h][-v level][-L logfile][-C casename][-R runId]
      [-c configfile][-T last_step][-S first_step][-s seconds]
      [-P parameter multiplication][-X][-f][-I][-i tbl_dir][-x subcatchId]
      [-p inputparameter multiplication][-l loglevel][--version]
```

```
-X: save state at the end of the run over the initial conditions at the start
-f: Force overwrite of existing results
-T: Set end time of the run: yyyy-mm-dd hh:mm:ss
-S: Set start time of the run: yyyy-mm-dd hh:mm:ss
-s: Set the model timesteps in seconds
-I: re-initialize the initial model conditions with default
-i: Set input table directory (default is intbl)
-x: Apply multipliers (-P/-p ) for subcatchment only (e.g. -x 1)
-C: set the name  of the case (directory) to run
-R: set the name runId within the current case
-L: set the logfile
-c: name of wflow configuration file (default: Casename/wflow_sbm.ini).
-h: print usage information
-P: set parameter change string (e.g: -P "self.FC = self.FC * 1.6") for non-dynamic␣
↪variables
-p: set parameter change string (e.g: -P "self.Precipitation = self.Precipitation * 1.11
↪") for
    dynamic variables
-l: loglevel (most be one of DEBUG, WARNING, ERROR)
```

wflow_sbm|hbv.ini file

The wflow_sbm|hbv.ini file holds a number of settings that determine how the model is operated. The files consists of sections that hold entries. A section is defined using a keyword in square brackets (e.g. [model]). Variables can be set in each section using a `keyword = value` combination (e.g. `ScalarInput = 1`). The default settings for the ini file are given in the subsections below.

[model] Options for all models:

ModelSnow = 0
: Set to 1 to model snow using a simple degree day model (in that case temperature data is needed).

WIMaxScale = 0.8
: Scaling for the topographical wetness vs soil depth method.

nrivermethod = 1
: Link N values to land cover (col 1), sub-catchment (col 2) and soil type (col 3) through the N_River.tbl file. Set to 2 to link N values in N_River.tbl file to streamorder (col 1).

MassWasting = 0
: Set to 1 to transport snow downhill using the local drainage network.

kinwaveIters = 0
: Set to 1 to enable iterations of the kinematic wave (time).

sCatch = 0
: If set to another value than 0 the model will only use the specified subcatchment

timestepsecs = 86400
: timestep of the model in seconds

Alpha = 60
: Alpha term in the river width estimation function

AnnualDischarge = 300
: Average annual discharge at the outlet of the catchment for the river wiidth estimation function

intbl = intbl
: directory from which to read the lookup tables (relative to the case directory)

Specific options for wflow_hbv :

updating = 0
: Set to 1 to switch on Q updating.

updateFile
: If updating is set to 1 specify a file

UpdMaxDist = 100
: Maximum distance from the gauge used in updating for which to update the kinematic wave reservoir (in model units, metres or degree lat lon)

The options below should normally not be needed. Here you can change the location of some of the input maps.

wflow_subcatch=staticmaps/wflow_subcatch.map
: map with the subcatchments

wflow_dem=staticmaps/wflow_dem.map
: the digital elevation map

wflow_ldd=staticmaps/wflow_ldd.map
: the local drainage network

wflow_river=staticmaps/wflow_river.map
all the cells marked as river

wflow_riverlength=staticmaps/wflow_riverlength.map
the length of the 'river' in each cell

wflow_riverlength_fact=staticmaps/wflow_riverlength_fact.map
factor to multiply the river length with

wflow_landuse=staticmaps/wflow_landuse.map
landuse map

wflow_soil=staticmaps/wflow_soil.map
soil map

wflow_gauges=staticmaps/wflow_gauges.map
map with the gauge locations

wflow_inflow=staticmaps/wflow_inflow.map
map with forced inflow points (optional)

wflow_mgauges=staticmaps/wflow_mgauges.map
map with locations of the meteorological gauges (only needed if you use scalar timeseries as input)

wflow_riverwidth=staticmaps/wflow_riverwidth.map
map with the width of the river

[layout]

sizeinmetres = 0
If set to zero the cell-size is given in lat/long (the default), otherwise the size is assumed to be in metres.

[outputmaps]

Outputmaps to save per timestep. Valid options for the keys in the wflow_sbm model are all variables visible the dynamic section of the model (see the code). A few useful variables are listed below.

```
[outputmaps]
self.RiverRunoff=run
self.SnowMelt=sno
self.Transfer=tr
self.SatWaterDepth=swd
```

Tip: NB See the wflow_sbm.py code for all the available variables as this list is incomplete. Also check the framwework documentation for the [run] section

The values on the right side of the equal sign can be choosen freely.

Example content:

```
Self.RiverRunoff=run
self.Transfer=tr
self.SatWaterDepth=swd
```

[outputcsv_0-n] [outputtss_0-n]

Number of sections to define output timeseries in csv format. Each section should at lears contain one samplemap item and one or more variables to save. The samplemap is the map that determines how the timesries are averaged/sampled. All other items are variabale filename pairs. The filename is given relative to the case directory.

Example:

```
[outputcsv_0]
samplemap=staticmaps/wflow_subcatch.map
self.RiverRunoffMM=Qsubcatch_avg.csv

[outputcsv_1]
samplemap=staticmaps/wflow_gauges.map
self.RiverRunoffMM=Qgauge.csv

[outputtss_0]
samplemap=staticmaps/wflow_landuse.map
self.RiverRunoffMM=Qlu.tss
```

In the above example the river discharge of this model (self.RiverRunoffMM) is saved as an average per subcatchment, a sample at the gauge locations and as an average per landuse.

[inputmapstacks]

This section can be used to overwrite the default names of the input mapstacks

Precipitation = /inmaps/P
timeseries for rainfall

EvapoTranspiration = /inmaps/PET
potential evapotranspiration

Temperature = /inmaps/TEMP
temperature time series

Inflow = /inmaps/IF
in/outflow locations (abstractions)

Updating using measured data

Note: Updating is only supported in the wflow_hbv model.

If a file (in .tss format) with measured discharge is specified using the -U command-line option the model will try to update (match) the flow at the outlet to the measured discharge. In that case the -u option should also be specified to indicate which of the columns must be used. When updating is switched on the following steps are taken:

- the difference at the outlet between measured and simulated Q (in mm) is determined
- this difference is added to the unsaturated store for all cells
- the ratio of measured Q divided by simulated Q at the outlet is used to multiply the kinematic wave store with. This ratio is scaled according to a maximum distance from the gauge.

Please note the following points when using updating:

- The tss file should have as many columns as there are gauges defined in the model
- The tss file should have enough data points to cover the simulation time
- The -U options should be used to specify which columns to actually use and in which order to use them. For example: -u '[1,3,2]' indicates to use column 1,2 and 3 in that order.

All possible options in wflow_sbm.ini file

```
[layout]
sizeinmetres = 1

[fit]
areamap = staticmaps/wflow_subcatch.map
areacode = 1
Q = testing.tss
WarmUpSteps = 1
ColMeas = 0
parameter_1 = RootingDepth
parameter_0 = M
ColSim = 0

[misc]

[outputmaps]
self.RiverRunoff = run

[framework]
debug = 0
outputformat = 1

[inputmapstacks]
Inflow = /inmaps/IF
Precipitation = /inmaps/P
Temperature = /inmaps/TEMP
EvapoTranspiration = /inmaps/PET

[model]
wflow_river = staticmaps/wflow_river.map
InterpolationMethod = inv
reinit = 1
WIMaxScale = 0.6
wflow_riverlength_fact = staticmaps/wflow_riverlength_fact.map
OverWriteInit = 0
intbl = intbl
wflow_riverwidth = staticmaps/wflow_riverwidth.map
wflow_soil = staticmaps/wflow_soil.map
sCatch = 0
Alpha = 120
wflow_subcatch = staticmaps/wflow_subcatch.map
wflow_mgauges = staticmaps/wflow_mgauges.map
timestepsecs = 86400
ScalarInput = 0
ModelSnow = 0
AnnualDischarge = 2290
wflow_landuse = staticmaps/wflow_landuse.map
TemperatureCorrectionMap = staticmaps/wflow_tempcor.map
wflow_inflow = staticmaps/wflow_inflow.map
wflow_riverlength = staticmaps/wflow_riverlength.map
wflow_ldd = staticmaps/wflow_ldd.map
```

(continues on next page)

(continued from previous page)

```
wflow_gauges = staticmaps/wflow_gauges.map
wflow_dem = staticmaps/wflow_dem.map
```

1.3 Building a model

Note: The information below is incomplete. Deltares is working on a tutorial.

1.3.1 Data requirements

The actual data requirements depend on the application of the model. The following list summarizes the data requirements:

- Static data
  - Digital Elevation Model (DEM)
  - A Land Cover map
  - A map representing Soil physical parameters (the Land Cover map can also be used)
- Dynamic data (spatial time series, map-stacks)
  - Precipitation
  - Potential evapotranspiration
  - Temperature (optional, only needed for snow pack modelling)
- Model parameters (per land use/soil type)
  - Soil Depth
  - etc… (see *Input parameters (lookup tables or maps)*)

The module can be linked to the Delft-FEWS system using the general adapter. The model itself comes with the necessary reading/writing routines for the Delft-FEWS pi XML files. An example of the link to Delft-FEWS is given in section *wflow_adapt Module*

In addition, each model should have an ini file (given the same basename as the model) that contains models specific information but also information for the framework (see wf_DynamicFramework)

1.3.2 Setting-up a new model

Setting-up a new model first starts with making a number of decisions and gathering the required data:

1. Do I have the static input maps in pcraster format (DEM ,land-use map, soil map)?
2. What resolution do I want to run the model on?
3. Do I need to define multiple sub-catchments to report totals/flows for seperately?
4. What forcing data do I have available for the model (P, Temp, ET)?
5. Do I have gridded forcing data or scalar time-series?

Note: Quantum Gis (QGIS) can read and write pcraster maps (via gdal) and is a very handy tool to support data preparation.

Note: Within the earth2observe project tools are being made to automatically download and downscale reanalysis date to be used as forcing to the wflow models. See https://github.com/earth2observe/downscaling-tools

Depending on the formats of the data some converting of data may be needed. The procedure described below assumes you have the main maps available in pcraster format. If that is not the case free tools like Qgis (www.qgis.org) and gdal can be used to convert the maps to the required format. Qgis is also very handy to see if the results of the scripts match reality by overlaying it with a google maps or OpenStreetMap layer using the qgis openlayers plugin.

When all data is available setting up the model requires the following steps:

1. Run the wflow_prepare_step1 and 2 scripts or prepare the input maps by hand (see *Preparing static input maps*)
2. Setup the wflow model directory structure (Setup a case) and copy the files (results from step2 of the prepare scripts) there (see *Setting Up a Case*)
3. Setup the .ini file
4. Test run the model
5. Supply all the .tbl files (or complete maps) for the model parameters (see *Input parameters (lookup tables or maps)*)
6. Calibrate the model

1.3.3 Preparing static input maps

Introduction

Preparing the input maps for a distributed model is not always trivial. wflow comes with two scripts that help in this process. The scripts are made with the assumption that the base DEM you have is a higher resolution as the DEM you want to use for the final model. When upscaling the scripts try to maintain as much information from the high resolution DEM as possible. The procedure described here can be used for all wflow models (wflow_sbm or wflow_hbv).

Using the scripts

The scripts assume you have a DEM, landuse and soil map available in pcraster format. If you do not have a soil or landuse map the you can generate a uniform map. The resolution and domain of these maps does not need to be the same, the scripts will take care of resampling. The process is devided in two scripts, wflow_prepare_step1.py and wflow_prepare_step2.py. In order to run the scripts the following maps/files need to be prepared.

Note: Both scripts need pcraster and gdal executables (version >= 1.10) to be available in your computers search path

- a DEM in pcraster format
- a land use map in pcraster format. If the resolution is different from the DEM the scripts will resample this map to match the DEM (or the DEM cutout). If no landuse map is found a uniform map will be created.
- a soil map in pcraster format. If no soil map is found a uniform map will be created.
- a configuration file for the prepare scripts that defines how they operate (.ini format) file (see below)

- an optional shape file with a river network (you can usually get one out of OpenStreetMap)
- an optional catchment mask file

The scripts work in two steps, each script need to be given at least one command-line option, the configuration file. The first script performs the following tasks:

- wflow_prepare_step1.py
  1. Performs an initial upscaling of the DEM if required (set in the configuration file). This initial upscaling may be needed if the processing steps (such as determining the drainage network) take a very long time or if the amount of available memory is not sufficient. The latter may be the case on 32bit systems. For example a 90x90 m grid for the Rhine/Meuse catchment could not be handled on a 32 bit system.
  2. Create the local drainage network. If the ldd is already present if will use the existing ldd. Use the force option to overwrite an existing ldd.
  3. Optionally use a shape file with a river network to "burn-in" this network and force the ldd to follow the river. In flat areas wher the river can be higher than the surrounding area having a river shape is crucial.

     Tip: Another option is to prepare a "pseudo dem" from a shape file with already defined catchment boundaries and outlets. Here all non boundary points would get a value of 1, all boundaries a value of 2 and all outlets a value of -10. This helps in generating a ldd for polder areas or other areas where the topography is not the major factor in determining the drainage network.

  4. Determine various statistics and also the largest catchment present in the DEM. This area will be used later on to make sure the catchments derived in the second step will match the catchment derived from the high resolution DEM
- wflow_prepare_step2.py
  1. Create a map with the extend and resolution defined in the configuration file and resample all maps from the first step to this resolution
  2. Create a new LDD using the following approach:
     – Construct a new dem to derive the ldd from suing the minimum dem from the first step for all the pixels that are located on a river and the maximum dem from the first step for all other pixels.
     – In addition raise all cells outside of the largest catchment defined in the first step with 1000 meter divided by the distance of each cell to the largest catchment.
     – Derive the ldd and determine the catchments

Once the script is finished successfully the following maps should have been created, the data type is shown between brackets:

- wflow_catchment.map (ordinal)
- wflow_dem.map (scalar)
- wflow_demmax.map (scalar)
- wflow_demmin.map (scalar)
- wflow-dem*percentile* - (10,25,33,50,66,75,90) (scalar)
- wflow_gauges.map (ordinal)
- wflow_landuse.map (nominal)
- wflow_soil.map (nominal)

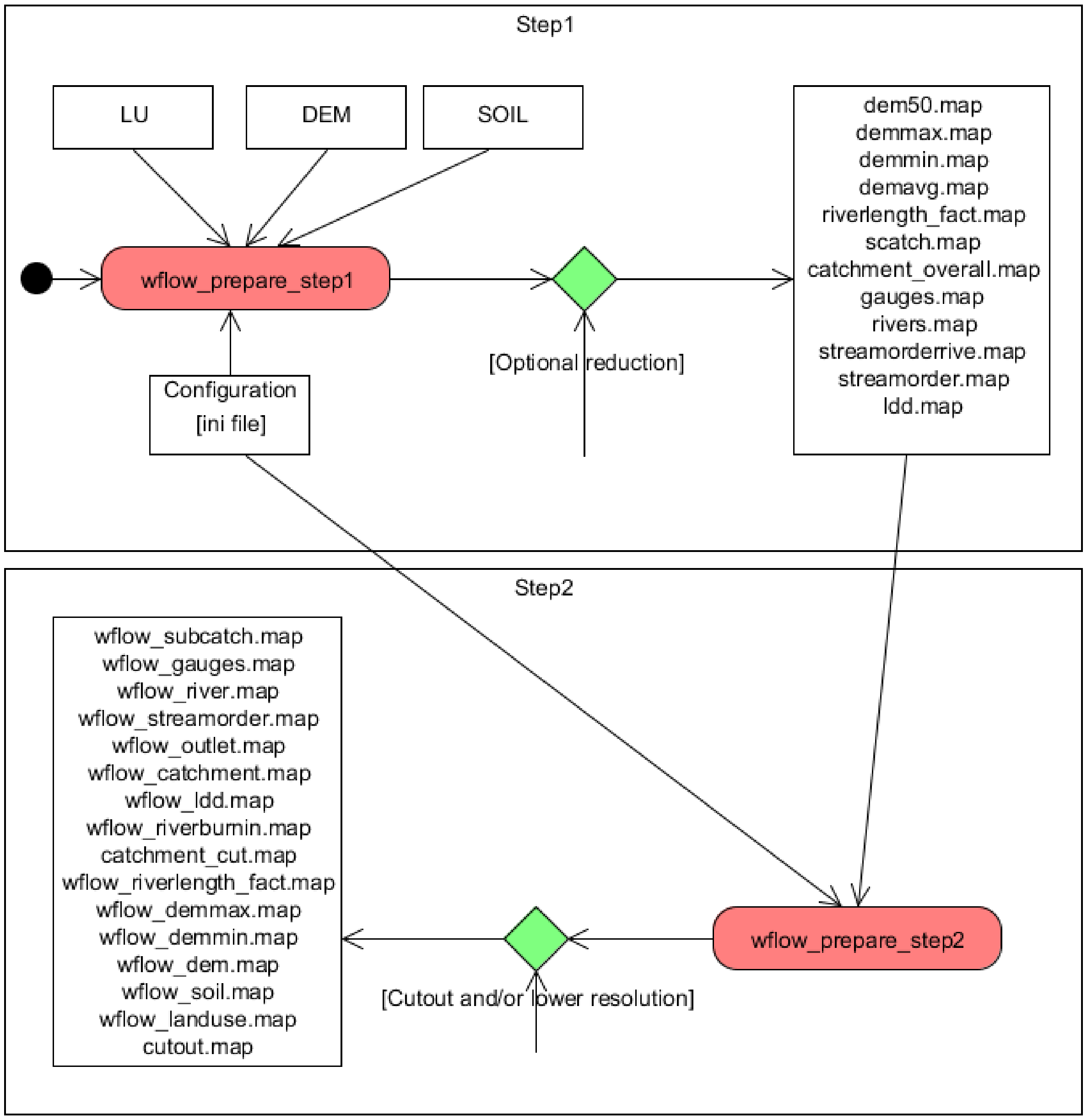


Fig. 1: Steps in creating the wflow model input

- wflow_ldd.map (ldd)
- wflow_outlet.map (scalar)
- wflow_riverburnin.map (boolean)
- wflow_riverlength_fact.map (scalar)
- wflow_river.map (ordinal)
- wflow_streamorder.map (ordinal)
- wflow_subcatch.map (ordinal)

The maps are created in the data processing directory. To use the maps in the model copy them to the staticmaps directory of the case you have created.

Note: Getting the subcatchment right can be a bit of a problem. In order for the subcatchment calculations to succeed the gauges that determine the outlets must be on a river grid cell. If the subcatchment creation causes problems the best way to check what is going on is to import both wflow_gauges,map en wflow_streamorder.map in qgis so you can check if the gauges are on a river cell. In the ini file you define the order above which a grid cell is regarded as a river.

Note: If the cellsize of the output maps is identical to the input DEM the second script shoudl NOT be run. All data will be produced by the first script.

Command line parameters

Both scripts take the same command-line parameters:

```
wflow_prepare_step1 -I inifile [-W workdir][-f][-h]

    -f force recreation of ldd if it already exists
    -h show this information
    -W set the working directory, default is current dir
    -I name of the ini file with settings
```

contents of the configuration file for the preprocessing

An example can be found `here`.

```
[directories]
# all paths are relative to the workdir set on the command line
# The directories in which the scripts store the output:
step1dir = step1
step2dir = step2

[files]
# Name of the DEM to use
masterdem=srtm_58_14.map
# name of the lad-use map to use
landuse=globcover_javabali.map
soil=soil.map
```

(continues on next page)

(continued from previous page)

```
# Shape file with river/drain network. Use to "burn in" into the dem.
river=river.shp
riverattr=river
# The riverattr above should be the shapefile-name without the .shp extension

[settings]
# Nr to reduce the initial map with in step 1. This means that all work is done
# on an upscaled version of the initial DEM. May be usefull for very
# large maps. If set to 1 (default) no scaling is taking place
initialscale=1

# Set lddmethod to dem (other methods are not working at the moment)
lddmethod=dem

# If set to 1 the gauge points are moved to the neares river point on a river
# with a strahler order higher of identical as defined in this ini file
snapgaugestoriver=1

# The strahler order above (and including) a pixel is defined as a river cell
riverorder=4

# X and y cooordinates of gauges (subcatchments). Please note the the locations
# are based on the river network of the DEM used in step2 (the lower resuolution
# DEM). This may need some experimenting... is most case the snap function
# will work by ymmv. To set multiple gauges use x_gauge_1, x_gauge_2

gauges_y = -6.1037
gauges_x = 107.4357


# settings for subgrid to create. This also determines how the
# original dem is (up)scaled. If the cellsize is the same
# as the original dem no scaling is performed. This grid will
# be the grid the final model runs on
Yul = -6.07
Xul = 106.9
Ylr = -7.30271
Xlr = 107.992
cellsize = 0.009166666663

# tweak ldd creation. Default should be fine most of the time
lddoutflowdepth=1E35
lddglobaloption=lddout
# use lddin to get rid of small catchments on the border of the dem
```

Problems

In many cases the scripts will not produce the maps the way you want them in the first try. The most common problems are:

1. The gauges do not coincide with a river and thus the subcatchment is not correct
   - Move the gauges to a location on the rivers as determined by the scripts. The best way to do this is to load the wflow_streamorder.map in qgis and use the cursor to find the nearest river cell for a gauge.
2. The delimited catchment is not correct even if the gauges is at the proper location
   - Get a better DEM or fix the current DEM.
   - Use a river shape file to fix the river locations
   - Use a catchment mask to force the catchment delineated to use that. Or just clip the DEM with the catchment mask. In the latter case use the lddin option to make sure you use the entire catchment.

If you still run into problems you can adjust the scripts yourself to get better results.

1.3.4 Script documentation

wflow_prepare_step1

wflow data preparation script. Data preparation can be done by hand or using the two scripts. This script does the first step. The second script does the resampling. This scripts need the pcraster and gdal executables to be available in you search path.

Usage:

```
wflow_prepare_step1 [-W workdir][-f][-h] -I inifile

-f force recreation of ldd if it already exists
-h show this information
-W set the working directory, default is current dir
-I name of the ini file with settings
```

$Id: $

`wflow_prepare_step1.`**`OpenConf`**`(`*fn*`)`

`wflow_prepare_step1.`**`configget`**`(`*config*, *section*, *var*, *default*`)`

gets parameter from config file and returns a default value if the parameter is not found

`wflow_prepare_step1.`**`main`**`()`

Variables

- **`masterdem`** – digital elevation model
- **`dem`** – digital elevation model
- **`river`** – optional river map

`wflow_prepare_step1.`**`usage`**`(`**args*`)`

wflow_prepare_step2

wflow data preparation script. Data preparation can be done by hand or using the two scripts. This script does the resampling. This scripts need the pcraster and gdal executables to be available in you search path.

Usage:

```
wflow_prepare_step2 [-W workdir][-f][-h] -I inifile
```

```
-f force recreation of ldd if it already exists
-h show this information
-W set the working directory, default is current dir
-I name of the ini file with settings
```

$Id: $

`wflow_prepare_step2.`**`OpenConf`**`(`*fn*`)`

`wflow_prepare_step2.`**`configget`**`(`*config*, *section*, *var*, *default*`)`

`wflow_prepare_step2.`**`main`**`()`

`wflow_prepare_step2.`**`resamplemaps`**`(`*step1dir*, *step2dir*`)`

Resample the maps from step1 and rename them in the process

`wflow_prepare_step2.`**`usage`**`(`**args*`)`

1.3.5 Setting Up a Case

PM

Note: Describes how to setup a model case structure. Probably need to write a script that does it automatically.

See wf_DynamicFramework for information on the settings in the ini file. The model specific settings are described seperately for each model.

1.3.6 Input parameters (lookup tables or maps)

The PCRaster lookup tables listed below are used by the model to create input parameter maps. Each table should have at least four columns. The first column is used to identify the land-use class in the wflow_landuse map, the second column indicates the subcatchment (wflow_subcatch), the third column the soil type (wflow_soil.map) and the last column list the value that will be assigned based on the first three columns.

Alternatively the lookup table can be replaced by a PCRaster map (in the staticmaps directory) with the same name as the tbl file (but with a .map extension).

Note: Note that the list of model parameters is (always) out of date. Getting the .tbl files from the example models (default_sbm and default_hbv) is probably the best way to start. In any case wflow will use default values for the tbl files that are missing. (shown in the log messages).

Below the contents of an example .tbl file is shown. In this case the parameters are identical for each subcatchment (and soil type) but is different for each landuse type. See the pcraster documentation (http://www.pcraster.eu) for details on how to create .tbl files.

```
1   <,14]   1 0.11
2   <,14]   1 0.11
3   <,14]   1 0.15
4   <,14]   1 0.11
5   <,14]   1 0.11
6   <,14]   1 0.11
```

Note: please note that if the rules in the tbl file do not cover all cells used in the model you will get missing values in the output. Check the maps in the runid/outsum directory to see if this is the case. Also, the model will generate a error message in the log file if this is the case so be sure to check the log file if you encounter problems. The message will read something like: "ERROR: Not all catchment cells have a value for…"

1.4 Questions and answers

1.4.1 Questions

[1] The discharge in the timeseries output gives weird numbers (1E31) what is going wrong?

[2] How do a setup a wflow model?

[3] Why do I have missing values in my model output?

[4] wflow stops and complains about types not matching

[5] wflow complains about missing initial state maps

[6] in some areas the mass balance error seems large

[1] *The discharge in the timeseries output gives weird numbers (1E31) what is going wrong?* The 1E31 values indicate missing values. This probably means that at least one of the cells in the part upstreasm of the discharge points has a missing value. Missing values are routed downstream so any missing values upstreams of a discharge will cause the discharge to eventually become a missing value. To resolve this check the following:

- Check if the .tbl files are correct (do they cover all values in the landuse soil and subcatchment maps)
- check for missing values in the input maps
- check of model parameters are within the working range: e.g. you have set a parameter (e.g. the canopy gap fraction in the interception model > 1) to an unrealistic value
- check all maps in the runId/outsum directory so see at which stage the missing values starts
- the soil/landuse/catchment maps does not cover the whole domain

Note: note that missing values in upstreams cells are routed down and will eventually make all downstreams values missing. Check the maps in the runid/outsum directory to see if the tbl files are correct

[2] *How do a setup a wflow model?* First read the section on *Setting-up a new model*. Next check one of the supplied example models

[3] *Why do I have missing values in my model output?* See question[1]

[4]

wflow stops and complains about types not matching
The underlying pcraster framework is very picky about data types. As such the maps must all be of the expected type. e.g. your landuse map MUST be nominal. See the pcraster documentation at pcraster.eu for more information

Note: If you create maps with qgis (or gdal) specify the right output type (e.g. Float32 for scalar maps)

[5] *wflow complains about missing initial state maps* run the model with the -I option first and copy the resulting files in runid/outstate back to the instate directory

[6] *in some areas the mass balance error seems large* The simple explicit solution of most models can cuase this, especially when parameter values

1.4.2 Answers

1.5 Available models

1.5.1 The wflow_hbv model

Introduction

The Hydrologiska Byrans Vattenbalansavdelning (HBV) model was introduced back in 1972 by the Swedisch Meteological and Hydrological Institute (SMHI). The HBV model is mainly used for runoff simulation and hydrological forecasting. The model is particularly useful for catchments where snow fall and snow melt are dominant factors, but application of the model is by no means restricted to these type of catchments.

Description

The model is based on the HBV-96 model. However, the hydrological routing represent in HBV by a triangular function controlled by the MAXBAS parameter has been removed. Instead, the kinematic wave function is used to route the water downstream. All runoff that is generated in a cell in one of the HBV reservoirs is added to the kinematic wave reservoir at the end of a timestep. There is no connection between the different HBV cells within the model. Wherever possible all functions that describe the distribution of parameters within a subbasin have been removed as this is not needed in a distributed application/

A catchment is divided into a number of grid cells. For each of the cells individually, daily runoff is computed through application of the HBV-96 of the HBV model. The use of the grid cells offers the possibility to turn the HBV modelling concept, which is originally lumped, into a distributed model.

The figure above shows a schematic view of hydrological response simulation with the HBV-modelling concept. The land-phase of the hydrological cycle is represented by three different components: a snow routine, a soil routine and a runoff response routine. Each component is discussed separately below.

The snow routine

Precipitation enters the model via the snow routine. If the air temperature, T_a, is below a user-defined threshold $TT(\approx 0^oC)$ precipitation occurs as snowfall, whereas it occurs as rainfall if $T_a \geq TT$. A another parameter TTI defines how precipitation can occur partly as rain of snowfall (see the figure below). If precipitation occurs as snowfall, it is added to the dry snow component within the snow pack. Otherwise it ends up in the free water reservoir, which represents the liquid water content of the snow pack. Between the two components of the snow pack, interactions take place, either through snow melt (if temperatures are above a threshold TT) or through snow refreezing (if temperatures are below threshold TT). The respective rates of snow melt and refreezing are:

$$\begin{aligned} Q_m &= cfmax(T_a - TT) \ \ ;T_a > TT \\ Q_r &= cfmax * cfr(TT - T_a) \ ;T_a < TT \end{aligned}$$

where Q_m is the rate of snow melt, Q_r is the rate of snow refreezing, and $cfmax$ and $cfr$ are user defined model parameters (the melting factor $mm/(^oC * day)$ and the refreezing factor respectively)

Note: The FoCFMAX parameter from the original HBV version is not used. instead the CFMAX is presumed to be for the landuse per pixel. Normally for forested pixels the CFMAX is 0.6 {*} CFMAX

are outside the nomally used range and with large timsteps. For example, setting the soil depth to zero will usually cause large errors. The solution is usually to check the parameters throughout the model.

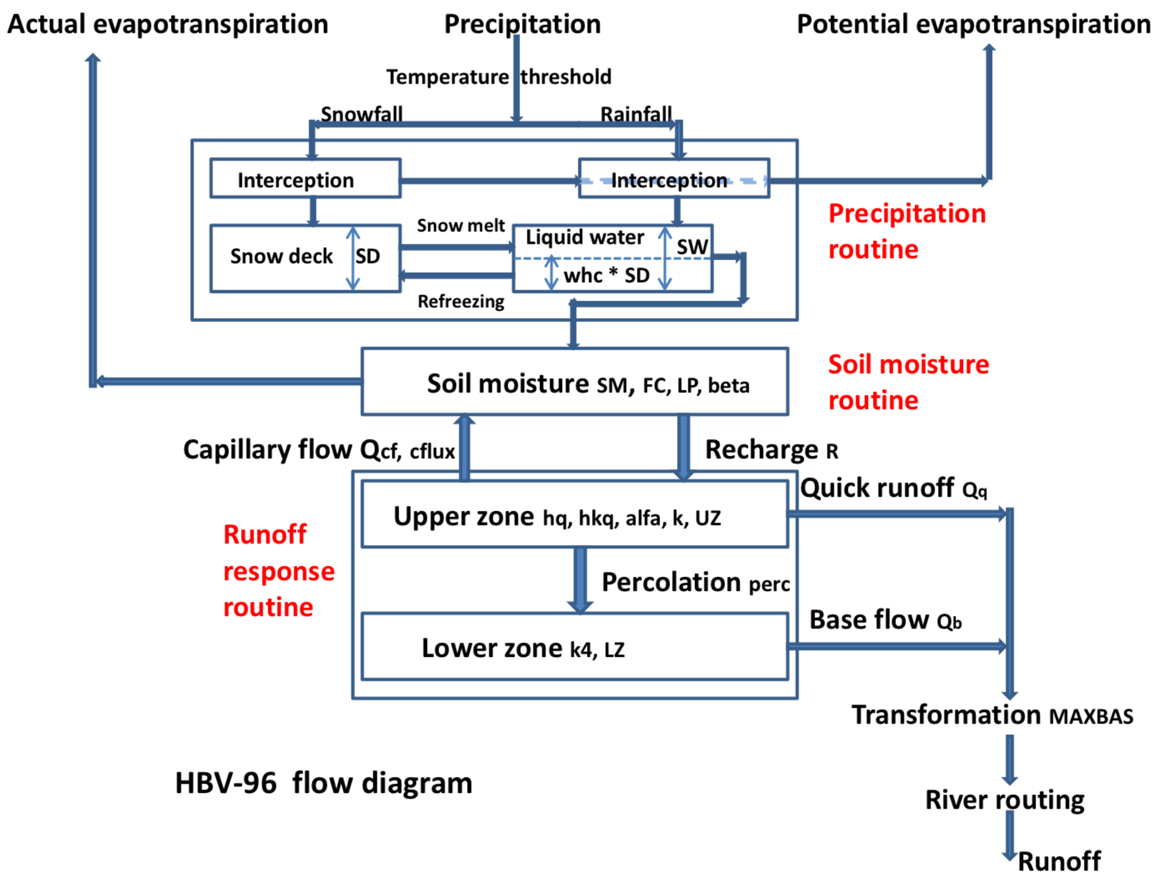


Fig. 2: Schematic view of the relevant components of the HBV model

The air temperature, T_a, is related to measured daily average temperatures. In the original HBV-concept, elevation differences within the catchment are represented through a distribution function (i.e. a hypsographic curve) which makes the snow module semi-distributed. In the modified version that is applied here, the temperature, T_a, is represented in a fully distributed manner, which means for each grid cell the temperature is related to the grid elevation.

The fraction of liquid water in the snow pack (free water) is at most equal to a user defined fraction, WHC, of the water equivalent of the dry snow content. If the liquid water concentration exceeds WHC, either through snow melt or incoming rainfall, the surpluss water becomes available for infiltration into the soil:

$$Q_{in} = max\{(SW - WHC * SD);\ 0.0\}$$

where Q_{in} is the volume of water added to the soil module, SW is the free water content of the snow pack and SD is the dry snow content of the snow pack.

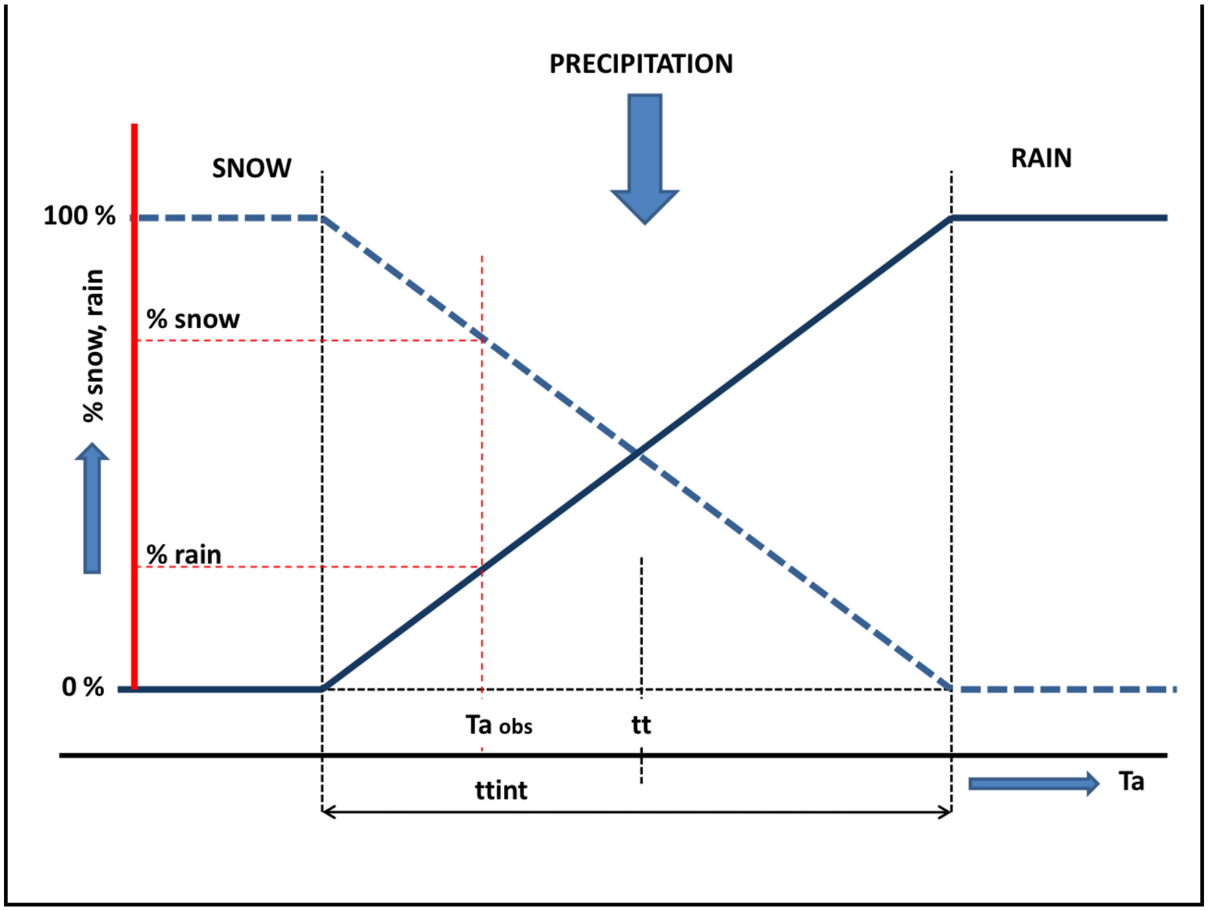


Fig. 3: Schematic view of the snow routine

The snow model als has an optional (experimental) 'mass-wasting' routine. This transports snow downhill using the local drainage network. To use it set the variable MassWasting in the model section to 1.

```
# Masswasting of snow
# 5.67 = tan 80 graden
SnowFluxFrac = min(0.5,self.Slope/5.67) * min(1.0,self.DrySnow/MaxSnowPack)
MaxFlux = SnowFluxFrac * self.DrySnow
self.DrySnow = accucapacitystate(self.TopoLdd,self.DrySnow, MaxFlux)
self.FreeWater = accucapacitystate(self.TopoLdd,self.FreeWater,SnowFluxFrac * self.
↪FreeWater )
```

Glaciers

Glacier processes are described in the wflow_funcs Module *Glacier modelling*

Potential Evaporation

The original HBV version includes both a multiplication factor for potential evaporation and a exponential reduction factor for potential evapotranspiration during rain events. The $CEVPF$ factor is used to connect potential evapotranspiration per landuse. In the original version the $CEVPFO$ is used and it is used for forest landuse only.

Interception

The parameters $ICF0$ and $ICFI$ introduce interception storage for forested and non-forested zones respectively in the original model. Within our application this is replaced by a single $ICF$ parameter assuming the parameter is set for each grid cell according to the land-use. In the original application it is not clear if interception evaporation is subtracted from the potential evaporation. In this implementation we dos subtract the interception evaporation to ensure total evaporation does not exceed potential evaporation. From this storage evaporation equal to the potential rate ET_p will occur as long as water is available, even if it is stored as snow. All water enters this store first, there is no concept of free throughfall (e.g. through gaps in the canopy). In the model a running water budget is kept of the interception store:

- The available storage (ICF-Actual storage) is filled with the water coming from the snow routine (Q_{in})
- Any surplus water now becomes the new Q_{in}
- Interception evaporation is determined as the minimum of the current interception storage and the potential evaporation

The soil routine

The incoming water from the snow and interception routines, Q_{in}, is available for infiltration in the soil routine. The soil layer has a limited capacity, F_c, to hold soil water, which means if F_c is exceeded the abundant water cannot infiltrate and, consequently, becomes directly available for runoff.

$$Q_{dr} = max\{(SM + Q_{in} - F_c);\ 0.0\}$$

where Q_{dr} is the abundant soil water (also referred to as direct runoff) and SM is the soil moisture content. Consequently, the net amount of water that infiltrates into the soil, I_{net}, equals:

$$I_{net} = Q_{in} - Q_{dr}$$

Part of the infiltrating water, I_{net}, will runoff through the soil layer (seepage). This runoff volume, SP, is related to the soil moisture content, SM, through the following power relation:

$$SP = \left(\frac{SM}{F_c}\right)^{\beta} I_{net}$$

where β is an empirically based parameter. Application of this equation implies that the amount of seepage water increases with increasing soil moisture content. The fraction of the infiltrating water which doesn't runoff, $I_{net} - SP$, is added to the available amount of soil moisture, SM. The β parameter affects the amount of supply to the soil moisture reservoir that is transferred to the quick response reservoir. Values of β vary generally between 1 and 3. Larger values of β reduce runoff and indicate a higher absorption capacity of the soil (see Figure ref{fig:HBV-Beta}).

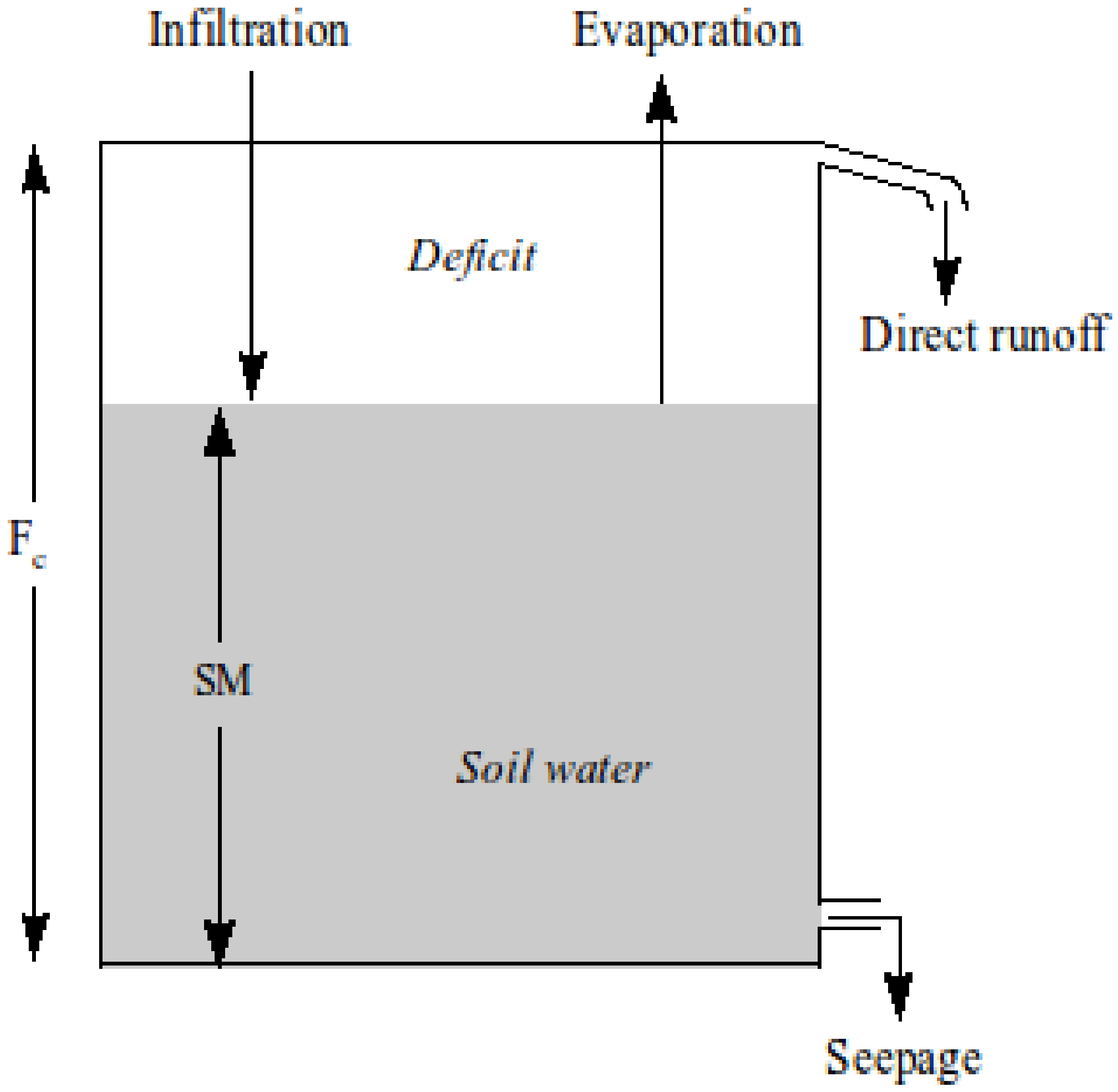


Fig. 4: Schematic view of the soil moisture routine

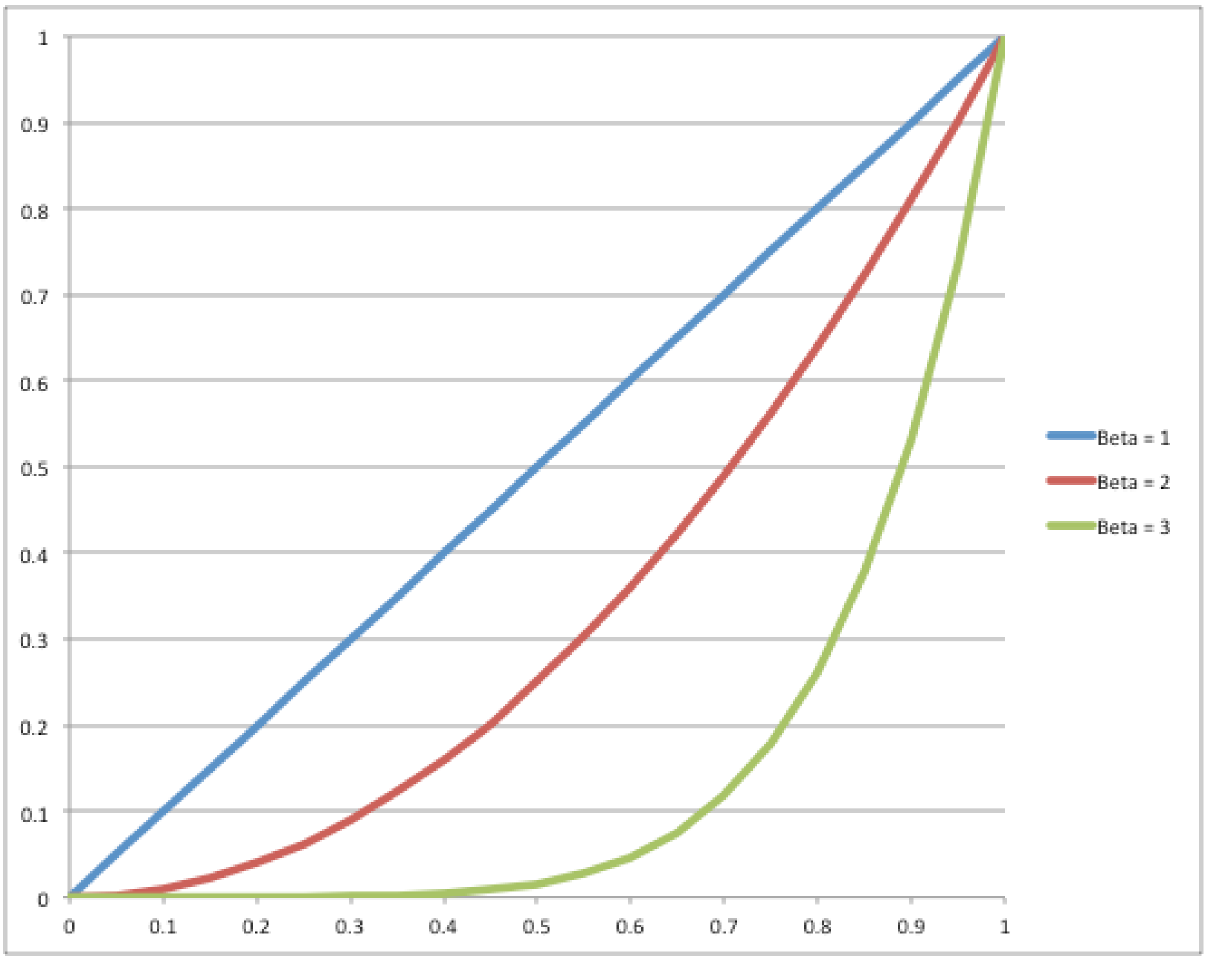


Fig. 5: Figure showing the relation between SM/F_c (x-axis) and the fraction of water running off (y-axis) for three values of β :1, 2 and 3

A percentage of the soil moisture will evaporate. This percentage is related to the measured potential evaporation and the available amount of soil moisture:

$$E_a = \frac{SM}{T_m} E_p \ ;SM < T_m$$

$$E_a = E_p \quad ;SM \geq T_m$$

where E_a is the actual evaporation, E_p is the potential evaporation and T_m ($\leq F_c$) is a user defined threshold, above which the actual evaporation equals the potential evaporation. T_m is defined as $LP{*}F_c$ in which LP is a soil dependent evaporation factor $(LP \leq 1)$.

In the original model (Berglov, 2009 XX), a correction to Ea is applied in case of interception. If Ea from the soil moisture storage plus Ei exceeds $ETp - Ei$ (Ei = interception evaporation) then the exceeding part is multiplied by a factor (1-ered), where the parameter ered varies between 0 and 1. This correction is presently not present in the wflow_hbv model.

The runoff response routine

The volume of water which becomes available for runoff, $S_{dr} + SP$, is transferred to the runoff response routine. In this routine the runoff delay is simulated through the use of a number of linear reservoirs.

Two linear reservoirs are defined to simulate the different runoff processes: the upper zone (generating quick runoff and interflow) and the lower zone (generating slow runoff). The available runoff water from the soil routine (i.e. direct runoff, S_{dr}, and seepage, SP) in principle ends up in the lower zone, unless the percolation threshold, $PERC$, is exceeded, in which case the redundant water ends up in the upper zone:

$$\triangle V_{LZ} = min\{PERC; (S_{dr} + SP)\}$$

$$\triangle V_{UZ} = max\{0.0; (S_{dr} + SP - PERC)\}$$

where V_{UZ} is the content of the upper zone, V_{LZ} is the content of the lower zone and $\triangle$ means increase of.

Capillary flow from the upper zone to the soil moisture reservoir is modeled according to:

$$Q_{cf} = cflux * (F_c - SM)/F_c$$

where $cflux$ is the maximum capilary flux in mm/day.

The Upper zone generates quick runoff (Q_q) using:

$$Q_q = K * UZ^{(1+alpha)}$$

here K is the upper zone recession coefficient, and α determines the amount of non-linearity. Within HBV-96, the value of K is determined from three other parameters: α, KHQ, and HQ (mm/day). The value of HQ represents an outflow rate of the upper zone for which the recession rate is equal to KHQ. if we define UZ_{HQ} to be the content of the upper zone at outflow rate HQ we can write the following equation:

$$HQ = K * UZ_{HQ}^{(1+\alpha)} = KHQ * UZ_{HQ}$$

If we eliminate UZ_{HQ} we obtain:

$$HQ = K * \left(\frac{HQ}{KHQ}\right)^{(1+\alpha)}$$

Rewriting for K results in:

$$K = KQH^{(1-alpha)} HQ^{-alpha}$$

Note: Note that the HBV-96 manual mentions that for a recession rate larger than 1 the timestap in the model will be adjusted.

The lower zone is a linear reservoir, which means the rate of slow runoff, Q_{LZ}, which leaves this zone during one time step equals:

$$Q_{LZ} = K_{LZ} * V_{LZ}$$

where K_{LZ} is the reservoir constant.

The upper zone is also a linear reservoir, but it is slightly more complicated than the lower zone because it is divided into two zones: A lower part in which interflow is generated and an upper part in which quick flow is generated (see Figure ref{fig:upper}).

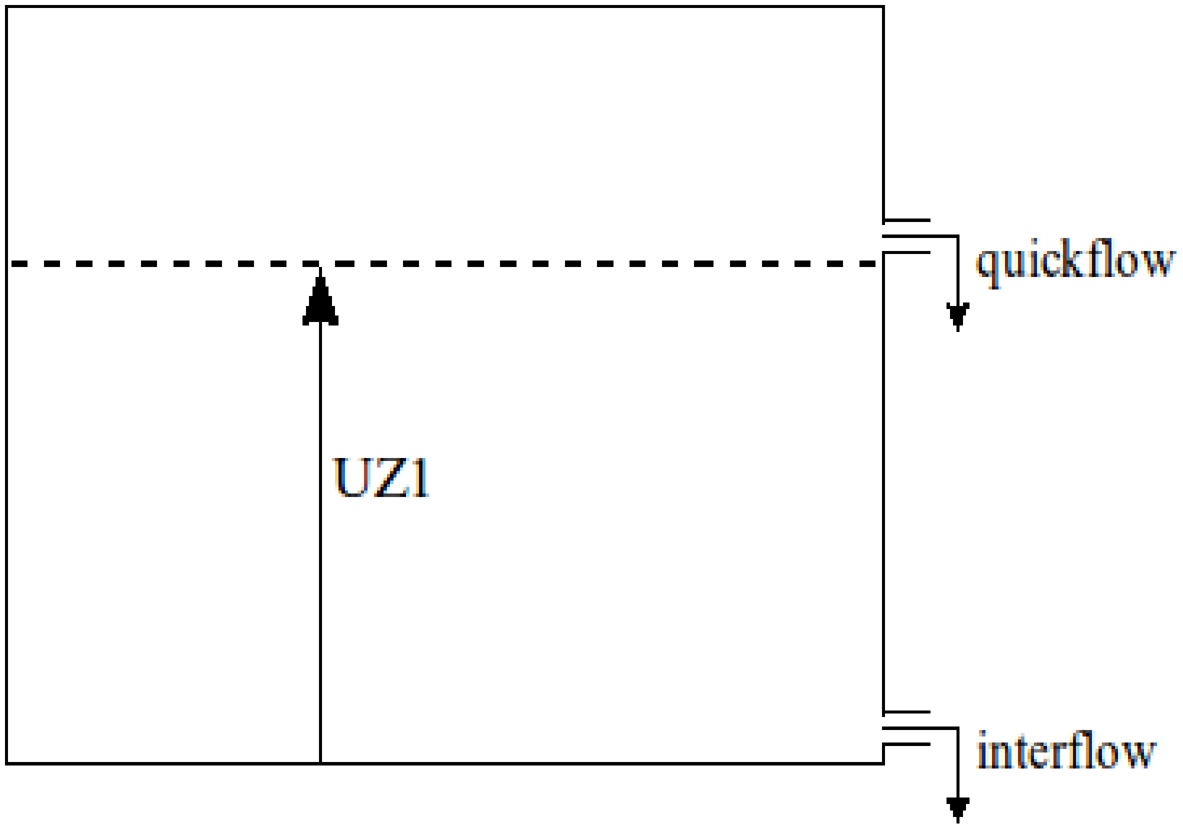


Fig. 6: Schematic view of the Upper zone

If the total water content of the upper zone, V_{UZ}, is lower than a threshold $UZ1$, the upper zone only generates interflow. On the other hand, if V_{UZ} exceeds $UZ1$, part of the upper zone water will runoff as quick flow:

$$Q_i = K_i * min\{UZ1; V_{uz}\}$$
$$Q_q = K_q * max\{(V_{UZ} - UZ1); 0.0\}$$

Where Q_i is the amount of generated interflow in one time step, Q_q is the amount of generated quick flow in one time step and K_i and K_q are reservoir constants for interflow and quick flow respectively.

The total runoff rate, Q, is equal to the sum of the three different runoff components:

$$Q = Q_{LZ} + Q_i + Q_q$$

The runoff behaviour in the runoff response routine is controlled by two threshold values P_m and $UZ1$ in combination with three reservoir parameters, K_{LZ}, K_i and K_q. In order to represent the differences in delay times between the three runoff components, the reservoir constants have to meet the following requirement:

$$K_{LZ} < K_i < K_q$$

Subcatchment flow

Normally the the kinematic wave is continuous throughout the model. By using the the SubCatchFlowOnly entry in the model section of the ini file all flow is at the subcatchment only and no flow is transferred from one subcatchment to another. This can be handy when connecting the result of the model to a water allocation model such as Ribasim.

Example:

```
[model]
SubCatchFlowOnly = 1
```

Description of the python module

1.5.2 The wflow_sbm Model

Introduction

The soil part of wflow_sbm model has its roots in the topog_sbm model but has had considerable changes over time. topog_sbm is specifically designed to simulate fast runoff processes in small catchments while wflow_sbm can be applied more widely. The main differences are:

- The unsaturated zone can be split-up in different layers
- The addition of evapotranspiration losses
- The addition of a capilary rise
- Wflow routes water over a D8 network while topog uses an element network based on contour lines and trajectories.

The sections below describe the working of the model in more detail.

Limitations

The wflow_sbm concept uses the kinematic wave approach for channel, overland and lateral subsurface flow, assuming that the topography controls water flow mostly. This assumption holds for steep terrain, but in less steep terrain the hydraulic gradient is likely not equal to the surface slope (subsurface flow), or pressure differences and inertial momentum cannot be neglected (channel and overland flow). In addition, while the kinemative wave equations are solved with a nonlinear scheme using Newton's method (Chow, 1988), other model equations are solved through a simple explicit scheme. In summary the following limitations apply:

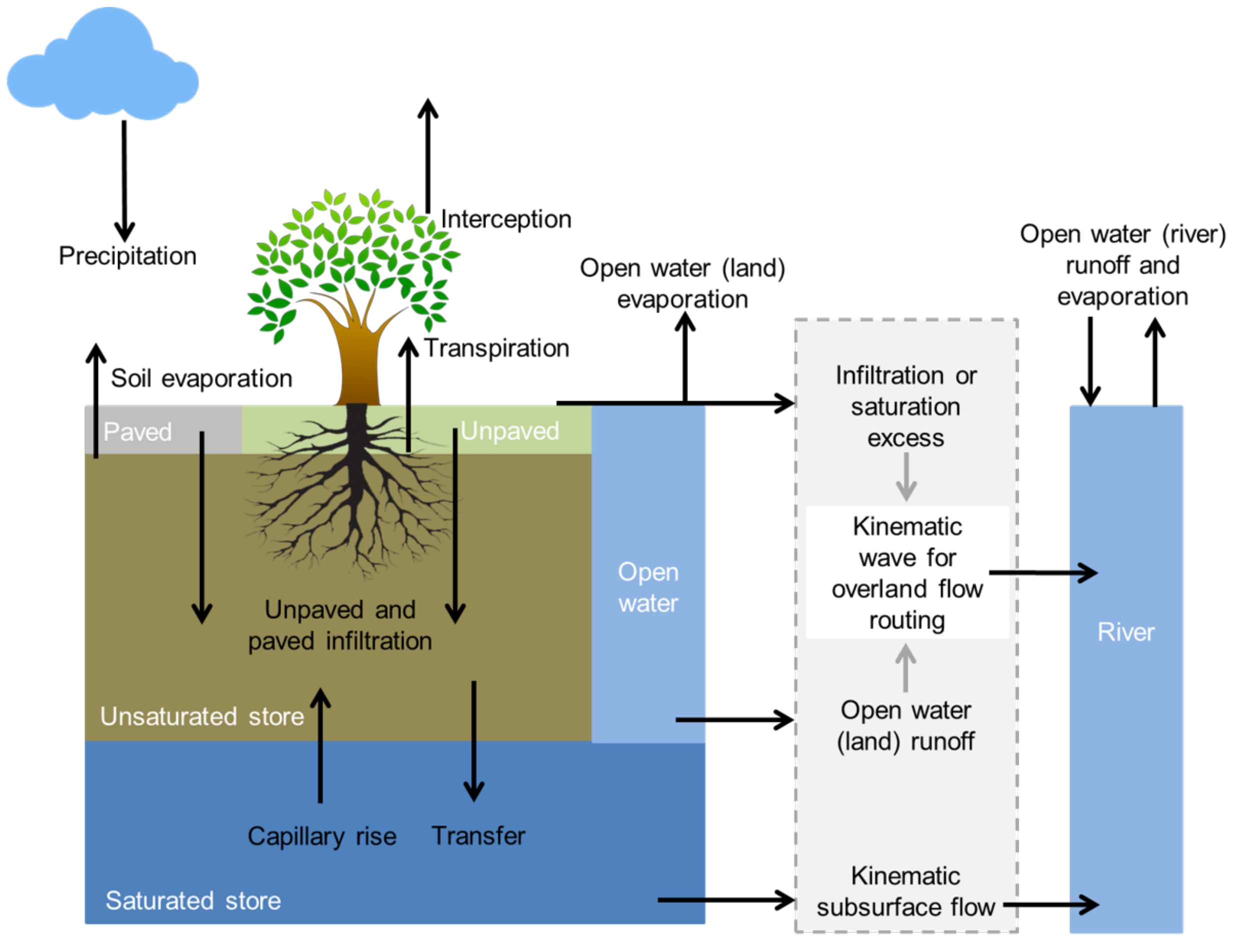

Precipitation
Interception
Open water (land) evaporation
Open water (river) runoff and evaporation
Soil evaporation
Transpiration
Paved
Unpaved
Infiltration or saturation excess
Kinematic wave for overland flow routing
Open water
River
Unpaved and paved infiltration
Unsaturated store
Open water (land) runoff
Capillary rise
Transfer
Saturated store
Kinematic subsurface flow

Fig. 7: Overview of the different processes and fluxes in the wflow_sbm model.

- Channel flow, and to a lesser degree overland flow, may be unrealistic in terrain that is not steep, and where pressure forces and inertial momentum are important.
- The lateral movement of subsurface flow may be very wrong in terrain that is not steep.
- The simple numerical solution means that results from a daily timestep model may be different from those with an hourly timestep.

Potential and Reference evaporation

The wflow_sbm model assumes the input to be potential evaporation. In many cases the evaporation will be a reference evaporation for a different land cover. In that case you can use the et_reftopot.tbl file to set the mutiplication per landuse to go from the supplied evaporation to the potential evaporation for each land cover. By default al is set to 1.0 assuming the evaporation to be potential.

Snow

Snow modelling is enabled by specifying the following in the ini file:

```
[model]
ModelSnow = 1
```

The snow model is described in the wflow_funcs Module *Snow modelling*

The snow model als has an optional (experimental) 'mass-wasting' routine. This transports snow downhill using the local drainage network. To use it set the variable MassWasting in the model section to 1.

```
# Masswasting of snow
# 5.67 = tan 80 graden
SnowFluxFrac = min(0.5,self.Slope/5.67) * min(1.0,self.DrySnow/MaxSnowPack)
MaxFlux = SnowFluxFrac * self.DrySnow
self.DrySnow = accucapacitystate(self.TopoLdd,self.DrySnow, MaxFlux)
self.FreeWater = accucapacitystate(self.TopoLdd,self.FreeWater,SnowFluxFrac * self.
↪FreeWater )
```

Glaciers

Glacier processes are described in the wflow_funcs Module *Glacier modelling*

The rainfall interception model

This section is described in the wflow_funcs Module *Rainfall Interception*

The soil model

Infiltration

If the surface is (partly) saturated the throughfall and stemflow that falls onto the saturated area is added to the river runoff component (based on fraction rivers, self.RiverFrac) and to the overland runoff component (based on open water fraction (self.WaterFrac) minus self.RiverFrac). Infiltration of the remaining water is determined as follows:

The soil infiltration capacity can be adjusted in case the soil is frozen, this is optional and can be set in the ini file as follows:

```
[model]
soilInfRedu = 1
```

The remaining storage capacity of the unsaturated store is determined. The infiltrating water is split in two parts, the part that falls on compacted areas and the part that falls on non-compacted areas. The maximum amount of water that can infiltrate in these areas is calculated by taking the minimum of the maximum infiltration rate (InfiltCapsoil for non-compacted areas and InfiltCapPath for compacted areas) and the water on these areas. The water that can actual infiltrate is calculated by taking the minimum of the total maximum infiltration rate (compacted and non-compacted areas) and the remaining storage capacity.

Infiltration excess occurs when the infiltration capacity is smaller then the throughfall and stemflow rate. This amount of water (self.InfiltExcess) becomes overland flow (infiltration excess overland flow). Saturation excess occurs when the (upper) soil becomes saturated and water cannot infiltrate anymore. This amount of water (self.ExcessWater and self.ExfiltWater) becomes overland flow (saturation excess overland flow).

The wflow_sbm soil water accounting scheme

A detailed description of the Topog_SBM model has been given by Vertessy (1999). Briefly: the soil is considered as a bucket with a certain depth (z_t), divided into a saturated store (S) and an unsaturated store (U), the magnitudes of which are expressed in units of depth. The top of the S store forms a pseudo-water table at depth z_i such that the value of S at any time is given by:

$$S = (z_t - z_i)(\theta_s - \theta_r)$$

where:

θ_s and θ_r are the saturated and residual soil water contents, respectively.

The unsaturated store (U) is subdivided into storage (U_s) and deficit (U_d) which are again expressed in units of depth:

$$\begin{aligned} U_d &= (\theta_s - \theta_r)z_i - U \\ U_s &= U - U_d \end{aligned}$$

The saturation deficit (S_d) for the soil profile as a whole is defined as:

$$S_d = (\theta_s - \theta_r)z_t - S$$

All infiltrating water enters the U store first. The unsaturated layer can be split-up in different layers, by providing the thickness [mm] of the layers in the ini file. The following example specifies three layers (from top to bottom) of 100, 300 and 800 mm:

```
[model]
UStoreLayerThickness = 100,300,800
```

The code checks for each grid cell the specified layers against the SoilThickness, and adds or removes (partly) layer(s) based on the SoilThickness.

Assuming a unit head gradient, the transfer of water (st) from a U store layer is controlled by the saturated hydraulic conductivity K_{sat} at depth z (bottom layer) or z_i, the effective saturation degree of the layer, and a Brooks-Corey power coefficient (parameter c) based on the pore size distribution index λ (Brooks and Corey (1964)):

$$st = K_{sat} \left(\frac{\theta - \theta_r}{\theta_s - \theta_r} \right)^c$$

$$c = \frac{2 + 3\lambda}{\lambda}$$

When the unsaturated layer is not split-up into different layers, it is possible to use the original Topog_SBM vertical transfer formulation, by specifying in the ini file:

```
[model]
transfermethod = 1
```

The transfer of water from the U store to the S store (st) is in that case controlled by the saturated hydraulic conductivity K_{sat} at depth z_i and the ratio between U and S_d:

$$st = K_{sat} \frac{U_s}{S_d}$$

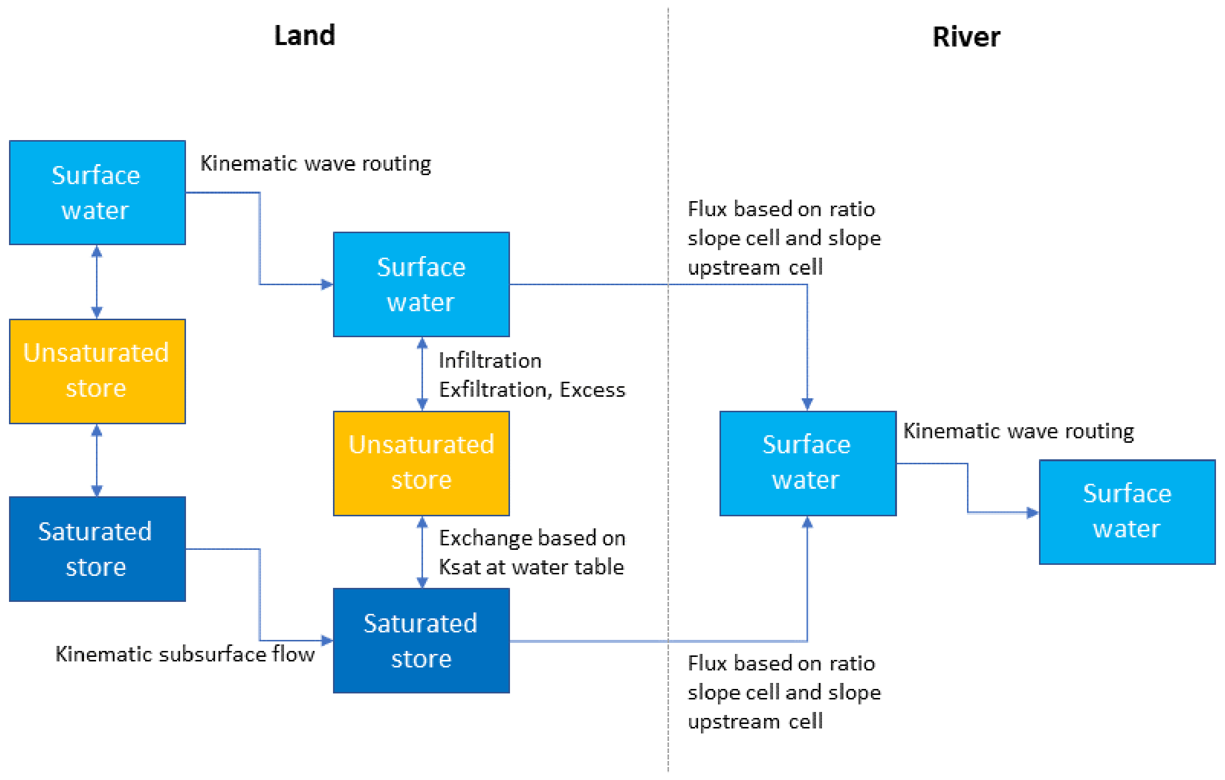


Fig. 8: Schematisation of the soil and the connection to the river within the wflow_sbm model

Saturated conductivity (K_{sat}) declines with soil depth (z) in the model according to:

$$K_{sat} = K_0 e^{(-fz)}$$

where:

K_0 is the saturated conductivity at the soil surface and

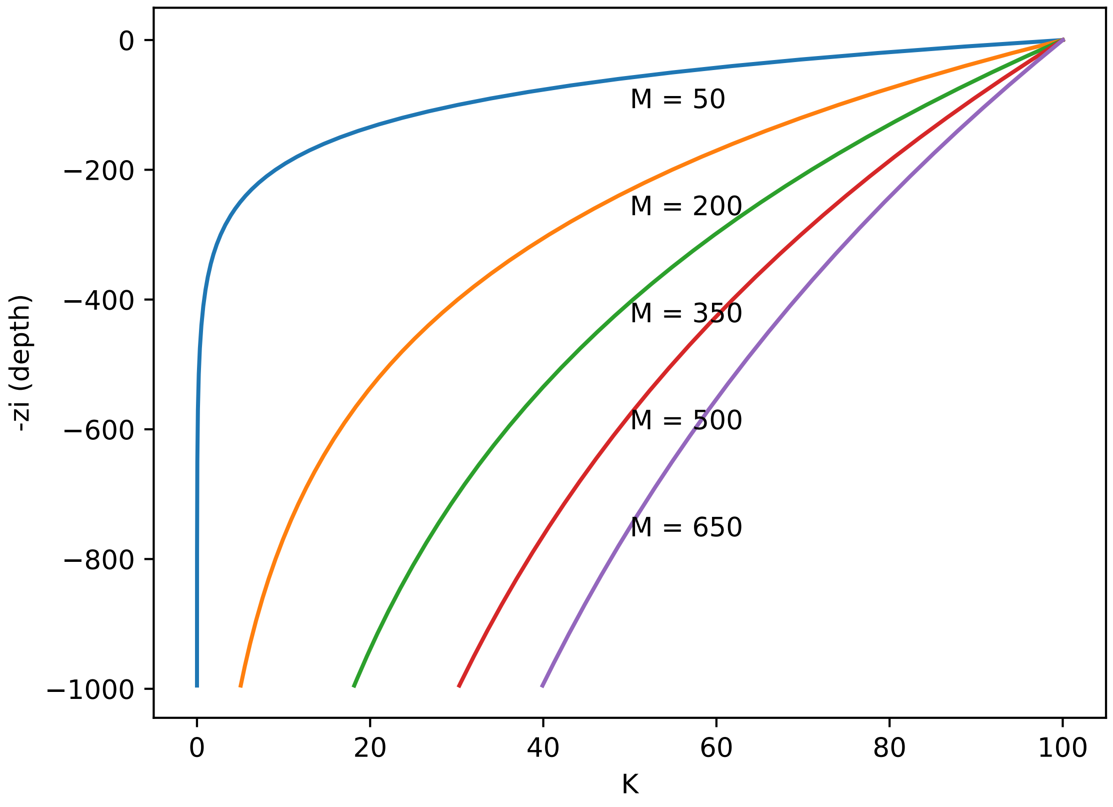
f is a scaling parameter $[mm^{-1}]$

The scaling parameter f is defined by:

$f = \frac{\theta_s - \theta_r}{M}$

with θ_s and θ_r as defined previously and M representing a model parameter (expressed in millimeter).

Figure: Plot of the relation between depth and conductivity for different values of M



The kinematic wave approach for lateral subsurface flow is described in the wflow_funcs Module *Subsurface flow routing*

Transpiration and soil evaporation

The potential eveporation left over after interception and open water evaporation (rivers and water bodies) is split in potential soil evaporation and potential transpiration based on the canopy gap fraction (assumed to be identical to the amount of bare soil).

For the case of one single soil layer, soil evaporation is scaled according to:

$soilevap = potensoilevap \frac{SaturationDeficit}{SoilWaterCapacity}$

As such, evaporation will be potential if the soil is fully wetted and it decreases linear with increasing soil moisture deficit.

For more than one soil layer, soil evaporation is only provided from the upper soil layer (often 100 mm) and soil evaporation is split in evaporation from the unsaturated store and evaporation from the saturated store. First water is evaporated water from the unsaturated store. Then the remaining potential soil evaporation can be used for evaporation from the saturated store. This is only possible, when the water table is present in the upper soil layer (very wet conditions). Both the evaporation from the unsaturated store and the evaporation from the saturated store are limited by the minimum of the remaining potential soil evaporation and the available water in the unsaturated/saturated zone of the upper soil layer. Also for multiple soil layers, the evaporation (both unsaturated and saturated) decreases linearly with decreasing water availability.

The original Topog_SBM model does not include transpiration or a notion of capilary rise. In wflow_sbm transpiration is first taken from the S store if the roots reach the water table z_i. If the S store cannot satisfy the demand the U store is used next. First the number of wet roots is determined (going from 1 to 0) using a sigmoid function as follows:

$$WetRoots = 1.0/(1.0 + e^{-SharpNess(WaterTable - RootingDepth)})$$

Here the sharpness parameter (by default a large negative value, -80000.0) parameter determines if there is a stepwise output or a more gradual output (default is stepwise). WaterTable is the level of the water table in the grid cell in mm below the surface, RootingDepth is the maximum depth of the roots also in mm below the surface. For all values of WaterTable smaller that RootingDepth a value of 1 is returned if they are equal a value of 0.5 is returned if the WaterTable is larger than the RootingDepth a value of 0 is returned. The returned WetRoots fraction is multiplied by the potential evaporation (and limited by the available water in saturated zone) to get the transpiration from the saturated part of the soil:

```
# evaporation from saturated store
wetroots = _sCurve(dyn['zi'][idx], a=static['ActRootingDepth'][idx], c=static[
↪'rootdistpar'][idx])
dyn['ActEvapSat'][idx] = min(PotTrans * wetroots, dyn['SatWaterDepth'][idx])
dyn['SatWaterDepth'][idx] = dyn['SatWaterDepth'][idx] - dyn['ActEvapSat'][idx]
RestPotEvap = PotTrans - dyn['ActEvapSat'][idx]
```

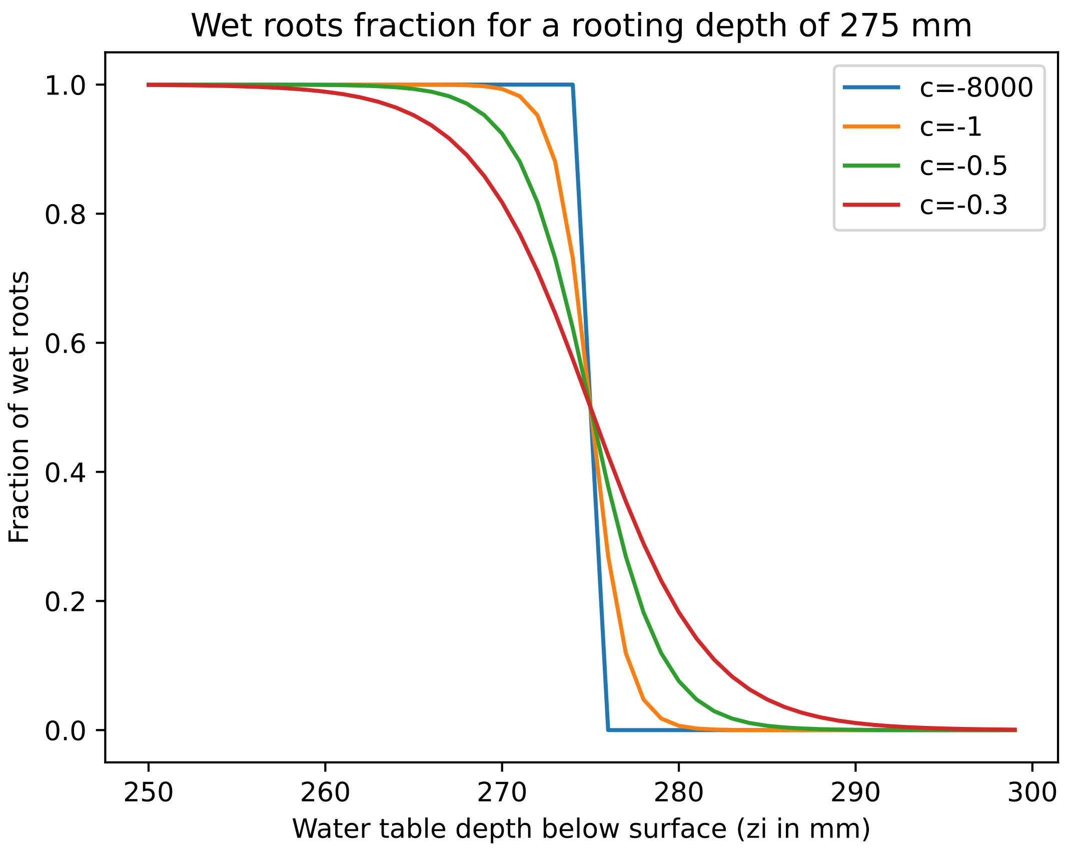
Figure: Plot showing the fraction of wet roots for different values of c for a RootingDepth of 275 mm

Next the remaining potential evaporation is used to extract water from the unsaturated store. The fraction of roots (AvailCap) that cover the unsaturated zone for each soil layer is used to calculate the potential root water extraction rate (MaxExtr):

```
MaxExtr = AvailCap * UstoreLayerDepth
```

When setting Whole_UST_Avail to 1 in the ini file as follows, the complete unsaturated storage is available for transpiration:

```
[model]
Whole_UST_Avail = 1
```


Wet roots fraction for a rooting depth of 275 mm
Fraction of wet roots
Water table depth below surface (zi in mm)
c=-8000
c=-1
c=-0.5
c=-0.3
1.0
0.8
0.6
0.4
0.2
0.0
250
260
270
280
290
300

Next, the Feddes root water uptake reduction model (Feddes et al. (1978)) is used to calculate a reduction coefficient as a function of soil water pressure. Soil water pressure is calculated following Brooks and Corey (1964):

$$\frac{(\theta - \theta_r)}{(\theta_s - \theta_r)} = \begin{cases} \left(\frac{h_b}{h}\right)^{\lambda}, h > h_b \\ 1, h \leq h_b \end{cases}$$

where:

h is the pressure head (cm), h_b is the air entry pressure head, and θ, θ_s, θ_r and λ as previously defined.

Feddes (1978) described a transpiration reduction-curve for the reduction coefficient α, as a function of h. Below, the function used in wflow_sbm, that calculates actual transpiration from the unsaturated zone layer(s).

```
def actTransp_unsat_SBM(RootingDepth, UStoreLayerDepth, sumLayer, RestPotEvap,␣
↪sumActEvapUStore, c, L,
thetaS, thetaR, hb, ust=0):

"""
Actual transpiration function for unsaturated zone:

  if ust is True, all ustore is available for transpiration

Input:

    - RootingDepth, UStoreLayerDepth, sumLayer (depth (z) of upper boundary unsaturated␣
↪layer),
      RestPotEvap (remaining evaporation), sumActEvapUStore (cumulative actual␣
↪transpiration (more than one UStore layers))
      c (Brooks-Corey coefficient), L (thickness of unsaturated zone), thetaS, thetaR,␣
↪hb (air entry pressure), ust

Output:

    - UStoreLayerDepth,  sumActEvapUStore, ActEvapUStore
"""

# AvailCap is fraction of unsat zone containing roots
if ust >= 1:
    AvailCap = UStoreLayerDepth * 0.99
else:
    if L > 0:
        AvailCap = min(1.0, max(0.0, (RootingDepth - sumLayer) / L))
    else:
        AvailCap = 0.0

MaxExtr = AvailCap * UStoreLayerDepth

# Next step is to make use of the Feddes curve in order to decrease ActEvapUstore when␣
↪soil moisture values
# occur above or below ideal plant growing conditions (see also Feddes et al., 1978). h1-
↪h4 values are
# actually negative, but all values are made positive for simplicity.
h1 = hb  # cm (air entry pressure)
h2 = 100  # cm (pF 2 for field capacity)
h3 = 400  # cm (pF 3, critical pF value)
```

(continues on next page)

(continued from previous page)

```
h4 = 15849  # cm (pF 4.2, wilting point)

# According to Brooks-Corey
par_lambda = 2 / (c - 3)
if L > 0.0:
    vwc = UStoreLayerDepth / L
else:
    vwc = 0.0
vwc = max(vwc, 0.0000001)
head = hb / (
    ((vwc) / (thetaS - thetaR)) ** (1 / par_lambda)
)  # Note that in the original formula, thetaR is extracted from vwc, but thetaR is not␣
↪part of the numerical vwc calculation
head = max(head,hb)

# Transform h to a reduction coefficient value according to Feddes et al. (1978).
# For now: no reduction for head < h2 until following improvement (todo):
#       - reduction only applied to crops
if(head <= h1):
    alpha = 1
elif(head >= h4):
    alpha = 0
elif((head < h2) & (head > h1)):
    alpha = 1
elif((head > h3) & (head < h4)):
    alpha = 1 - (head - h3) / (h4 - h3)
else:
    alpha = 1

ActEvapUStore = (min(MaxExtr, RestPotEvap, UStoreLayerDepth)) * alpha

UStoreLayerDepth = UStoreLayerDepth - ActEvapUStore

RestPotEvap = RestPotEvap - ActEvapUStore
sumActEvapUStore = ActEvapUStore + sumActEvapUStore

return UStoreLayerDepth, sumActEvapUStore, RestPotEvap
```

Capilary rise is determined using the following approach: first the K_{sat} is determined at the water table z_i; next a potential capilary rise is determined from the minimum of the K_{sat}, the actual transpiration taken from the U store, the available water in the S store and the deficit of the U store. Finally the potential rise is scaled using the distance between the roots and the water table using:

$$CSF = CS/(CS + z_i - RT)$$

in which CSF is the scaling factor to multiply the potential rise with, CS is a model parameter (default = 100, use CapScale.tbl to set differently) and RT the rooting depth. If the roots reach the water table ($RT > z_i$) CS is set to zero thus setting the capilary rise to zero.

Leakage

If the MaxLeakage parameter is set > 0, water is lost from the saturated zone and runs out of the model.

Soil temperature

The near surface soil temperature is modelled using a simple equation (Wigmosta et al., 2009):

$$T_s^t = T_s^{t-1} + w(T_a - T_s^{t-1})$$

where T_s^t is the near-surface soil temperature at time t, T_a is air temperature and w is a weighting coefficient determined through calibration (default is 0.1125 for daily timesteps).

A reduction factor (cf_soil, default is 0.038) is applied to the maximum infiltration rate (InfiltCapSoil and InfiltCap-Path), when the following model settings are specified in the ini file:

```
[model]
soilInfRedu = 1
ModelSnow = 1
```

A S-curve (see plot below) is used to make a smooth transition (a c-factor (c) of 8 is used):

$$\begin{aligned} b &= \frac{1.0}{(1.0 - cf\_soil)} \\ soilInfRedu &= \frac{1.0}{b + exp(-c(T_s - a))} + cf\_soil \\ a &= 0.0 \\ c &= 8.0 \end{aligned}$$

Irrigation and water demand

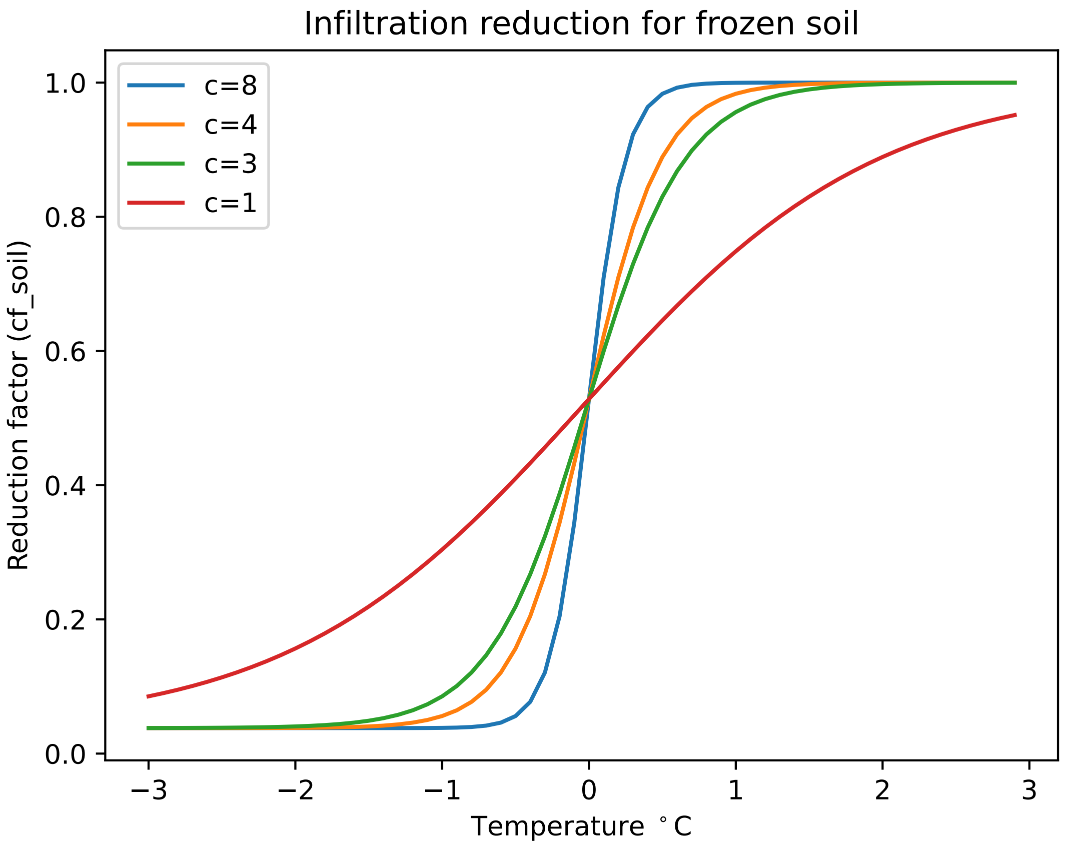
Water demand (surface water only) by irrigation can be configured in two ways:

1. By specifying the water demand externally (as a lookup table, series of maps etc)
2. By defining irrigation areas. Within those areas the demand is calculated as the difference between potential ET and actual transpiration

For both options a fraction of the supplied water can be put back into the river at specified locations

The following maps and variables can be defined:

- **wflow_irrigationareas.map**: Map of areas where irrigation is applied. Each area has a unique id. The areas do not need to be continuous& but all cells with the same id are assumed to belong to the same irrigation area.
- **wflow_irrisurfaceintake.map**: Map of intake points at the river(s). The id of each point should correspond to the id of an area in the wflow_irrigationareas map.
- **wflow_irrisurfacereturns.map**: Map of water return points at the river(s). The id of each point should correspond to the id of an area in the wflow_irrigationareas map or/and the wflow_irrisurfaceintake.map.
- **IrriDemandExternal**: Irrigation demand supplied to the model. This can be doen by adding an entry to the modelparameters section. if this is doen the irrigation demand supplied here is used and it is NOT determined by the model. Water demand should be given with a negative sign! See below for and example entry in the modelparameters section:


Infiltration reduction for frozen soil
c=8
c=4
c=3
c=1
1.0
0.8
0.6
0.4
0.2
0.0
Reduction factor (cf_soil)
−3
−2
−1
0
1
2
3
Temperature °C

```
IrriDemandExternal=intbl/IrriDemandExternal.tbl,tbl,-34.0,0,staticmaps/wflow_
↪irrisurfaceintakes.map
```

In this example the default demand is -34 m^3 s^{-1}. The demand must be linked to the map wflow_irrisurfaceintakes.map. Alternatively we can define this as a timeseries of maps:

```
IrriDemandExternal=/inmaps/IRD,timeseries,-34.0,0
```

- **DemandReturnFlowFraction**: Fraction of the supplied water the returns back into the river system (between 0 and 1). This fraction must be supplied at the wflow_irrisurfaceintakes.map locations but the water that is returned to the river will be returned at the wflow_irrisurfacereturns.map locations. If this variable is not defined the default is 0.0. See below for an example entry in the modelparameters section:

```
DemandReturnFlowFraction=intbl/IrriDemandReturn.tbl,tbl,0.0,0,staticmaps/wflow_
↪irrisurfaceintakes.map
```

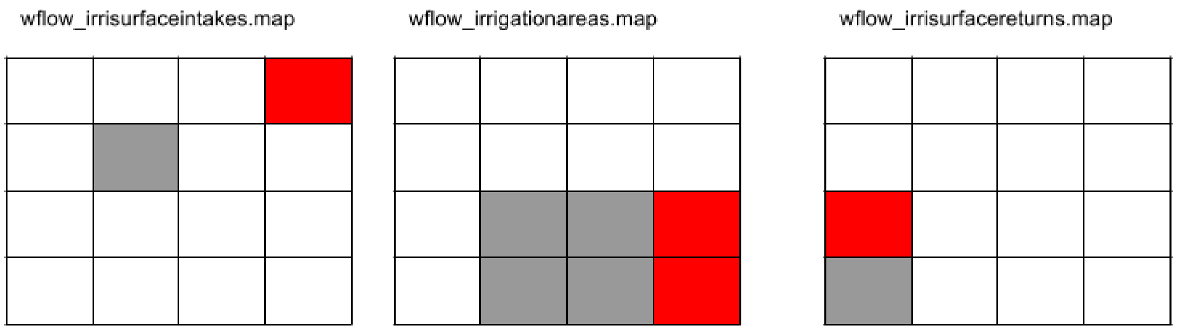

wflow_irrisurfaceintakes.map
wflow_irrigationareas.map
wflow_irrisurfacereturns.map

Fig. 9: Figure showing the three maps that define the irrigation intake points areas and return flow locations.

The irrigation model can be used in the following two modes:

1. An external water demand is given (the user has specified the IrriDemandExternal variable). In this case the demand is enforced. If a matching irrigation area is found the supplied water is converted to an amount in mm over the irrigation area. The supply is converted in the *next timestep* as extra water available for infiltration in the irrigation area. If a DemandReturnFlowFraction is defined this fraction is the supply is returned to the river at the wflow_irrisurfacereturns.map points.

2. Irrigation areas have been defined and no IrriDemandExternal has been defined. In this case the model will estimate the irrigation water demand. The irrigation algorithim works as follows: For each of the areas the difference between potential transpiration and actual transpiration is determined. Next, this is converted to a demand in m^3 s^{-1} at the corresponding intake point at the river. The demand is converted to a supply (taking into account the available water in the river) and converted to an amount in mm over the irrigation area. The supply is converted in the *next timestep* as extra water available for infiltration in the irrigation area. This option has only be tested in combination with a monthly LAI climatology as input. If a DemandReturnFlowFraction is defined this fraction is the supply is returned to the river at the wflow_irrisurfacereturns.map points.

Paddy areas and irrigation

Paddy areas (irrigated rice fields) can be defined by including the following maps:

- **wflow_irrigationpaddyareas.map**: Map of areas where irrigated rice fields are located.
- **wflow_hmax.map**: Map with the optimal water height [mm] in the irrigated rice fields.
- **wflow_hp.map**: Map of the water height [mm] when rice field starts spilling water (overflow).
- **wflow_hmin.map**: Map with the minimum required water height in the irrigated rice fields.
- **wflow_irrisurfaceintake.map**: Map of intake points at the river(s). The id of each point should correspond to the id of an area in the wflow_irrigationpaddyareas map.

Furthermore, gridded timeseries whether rice crop growth occurs (value = 1), or not (value = 0), are required. These timeseries can be included as follows:

```
[modelparameters]
CRPST=inmaps/CRPSTART,timeseries,0.0,1
```

Wflow_sbm will estimate the irrigation water demand as follows, a ponding depth (self.PondingDepth) is simulated in the grid cells with a rice crop. Potential evaporation left after interception and open water evaporation (self.RestEvap), is subtracted from the ponding depth as follows:

```
if self.nrpaddyirri > 0:
    self.ActEvapPond = pcr.min(self.PondingDepth, self.RestEvap)
    self.PondingDepth = self.PondingDepth - self.ActEvapPond
    self.RestEvap = self.RestEvap - self.ActEvapPond
```

Infiltration excess and saturation excess water are added to the ponding depth in grid cells with a rice crop, to the maximum water height when the rice field starts spilling water.

The irrigation depth is then determined as follows:

```
if self.nrpaddyirri > 0:
    irr_depth = (
        pcr.ifthenelse(
            self.PondingDepth < self.h_min, self.h_max - self.PondingDepth, 0.0
        )
        * self.CRPST
    )
```

The irrigation depth is converted to m^3 s^{-1} for each irrigated paddy area, at the corresponding intake point at the river. The demand is converted to a supply (taking into account the available water in the river) and converted to an amount in mm over the irrigation area. The supply is converted in the *next timestep* as extra water available for infiltration in the irrigation area.

This functionality was added to simulate rice crop production when coupled (through Basic Model Interface (BMI)) to the *The wflow_lintul Model*.

Kinematic wave, Length, Width and Slope

Both overland flow and river flow are routed through the catchment using the kinematic wave equation. For overland flow, width (self.SW) and length (self.DL) characteristics are based on the grid cell dimensions and flow direction, and in case of a river cell, the river width is subtracted from the overland flow width. For river cells, both width and length can either be supplied by separate maps:

- wflow_riverwidth.map
- wflow_riverlength.map

or determined from the grid cell dimension and flow direction for river length and from the DEM, the upstream area and yearly average discharge for the river width (Finnegan et al., 2005):

The yearly average Q at outlet is scaled for each point in the drainage network with the upstream area. α ranges from 5 to > 60. Here 5 is used for hardrock, large values are used for sediments.

When the river length is calculated based on grid dimensions and flow direction in wflow_sbm, it is possible to provide a map with factors to multiply the calculated river lenght with (wflow_riverlength_fact.map, default is 1.0).

The slope for kinematic overland flow and river flow can either be provided by maps:

- Slope.map (overland flow)
- RiverSlope.map (river flow)

or calculated by wflow_sbm based on the provided DEM and Slope function of PCRaster.

Note: If a river slope is available as map, then we recommend to also provide a river lenght map to avoid possible inconsistencies between datasets. If a river lenght map is not provided, the river length is calculated based on grid dimensions and flow direction, and if available the wflow_riverlength_fact.map.

Implementation of river width calculation:

```
if (self.nrresSimple + self.nrlake) > 0:
    upstr = pcr.catchmenttotal(1, self.TopoLddOrg)
else:
    upstr = pcr.catchmenttotal(1, self.TopoLdd)
Qscale = upstr / pcr.mapmaximum(upstr) * Qmax
W = (
    (alf * (alf + 2.0) ** (0.6666666667)) ** (0.375)
    * Qscale ** (0.375)
    * (pcr.max(0.0001, pcr.windowaverage(self.riverSlope, pcr.celllength() * 4.0)))
    ** (-0.1875)
    * self.NRiver ** (0.375)
)
# Use supplied riverwidth if possible, else calulate
self.RiverWidth = pcr.ifthenelse(self.RiverWidth <= 0.0, W, self.RiverWidth)
```

The table below list commonly used Manning's N values (in the N_River.tbl file). Please note that the values for non river cells may arguably be set significantly higher. (Use N.tbl for non-river cells and N_River.tbl for river cells)

Table 1: Manning's N values

Type of Channel and Description	Minimum	Normal	Maximum
Main Channels			
clean, straight, full stage, no rifts or deep pools	0.025	0.03	0.033
same as above, but more stones and weeds	0.03	0.035	0.04
clean, winding, some pools and shoals	0.033	0.04	0.045
same as above, but some weeds and stones	0.035	0.045	0.05
same as above, lower stages, more ineffective slopes and sections	0.04	0.048	0.055
same as second with more stones	0.045	0.05	0.06
sluggish reaches, weedy, deep pools	0.05	0.07	0.08
very weedy reaches, deep pools, or floodways with heavy stand of timber and underbrush	0.075	0.1	0.15
Mountain streams			
bottom: gravels, cobbles, and few boulders	0.03	0.04	0.05
bottom: cobbles with large boulders	0.04	0.05	0.07

Natural lakes and reservoirs can also be added to the model and taken into account during the routing process. For more information, see the documentation of the wflow_funcs module.

Subcatchment flow

Normally the the kinematic wave is continuous throughout the model. By using the the SubCatchFlowOnly entry in the model section of the ini file all flow is at the subcatchment only and no flow is transferred from one subcatchment to another. This can be handy when connecting the result of the model to a water allocation model such as Ribasim.

Example:

```
[model]
SubCatchFlowOnly = 1
```

Model variables stores and fluxes

The figure below shows the stores and fluxes in the model in terms of internal variable names.

Processing of meteorological forcing data

Although the model has been setup to do as little data processing as possible it includes an option to apply an altitude correction to the temperature inputs. The three squares below demonstrate the principle.

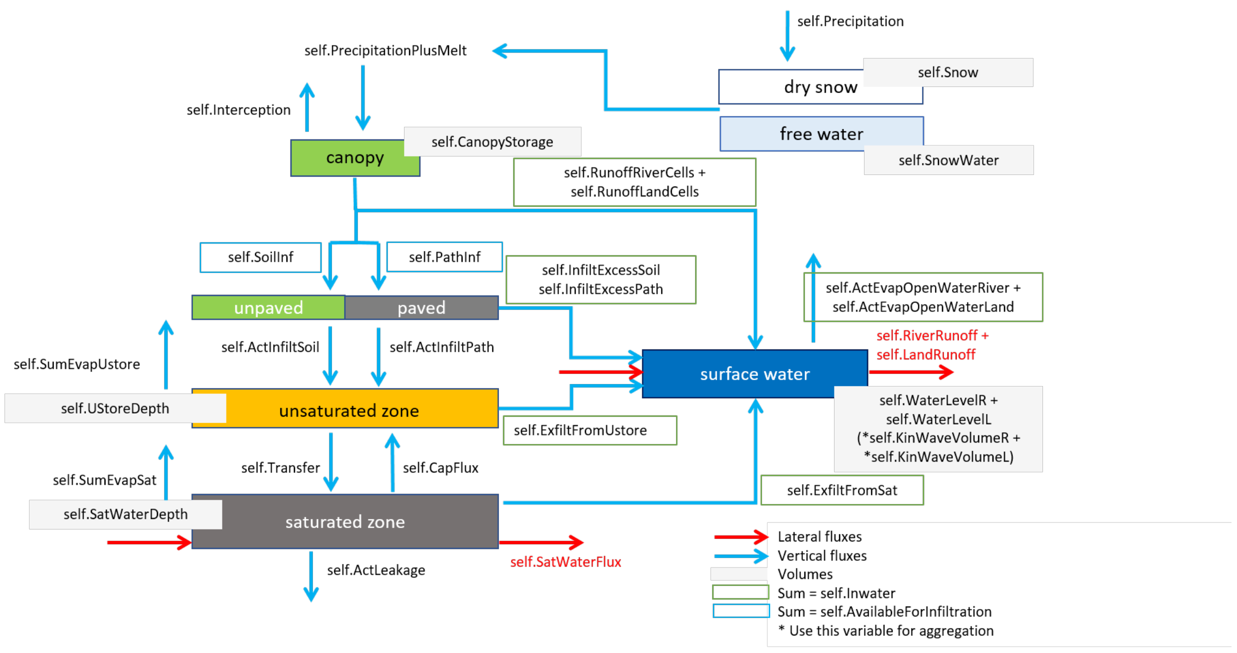
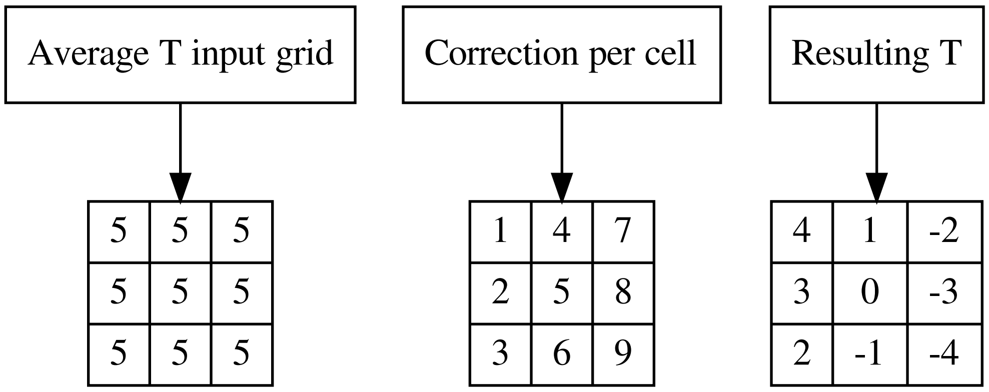


Fig. 10: Complete wflow scheme.



wflow_sbm takes the correction grid as input and applies this to the input temperature. The correction grid has to be made outside of the model. The correction grid is optional.

The temperature correction map is specified in the model section of the ini file:

```
[model]
TemperatureCorrectionMap=NameOfTheMap
```

If the entry is not in the file the correction will not be applied

Wflow_sbm model parameters

The list below shows the most important model parameters

CanopyGapFraction
Gash interception model parameter [-]: the free throughfall coefficient.

EoverR
Gash interception model parameter. Ratio of average wet canopy evaporation rate over average precipitation rate.

SoilThickness
Maximum soil depth [mm]

SoilMinThickness
Minimum soil depth [mm]

MaxLeakage
Maximum leakage [mm/day]. Leakage is lost to the model. Usually only used for i.e. linking to a dedicated groundwater model. Normally set to zero in all other cases.

KsatVer
Vertical saturated conductivity [mm/day] of the store at the surface. The M parameter determines how this decreases with depth.

KsatHorFrac
A multiplication factor [-] applied to KsatVer for the horizontal saturated conductivity used for computing lateral subsurface flow. This parameter compensates for anisotropy, small scale KsatVer measurement (small soil core) that do not represent larger scale hydraulic conductivity, and model resolution (in reality smaller (hillslope) flow length scales).

InfiltCapPath
Infiltration capacity [mm/day] of the compacted soil (or paved area) fraction of each gridcell.

InfiltCapSoil
Infiltration capacity [mm/day] of the non-compacted soil fraction (unpaved area) of each gridcell.

M
Soil parameter M [mm] determines the decrease of vertical saturated conductivity with depth. Usually between 20 and 2000.

MaxCanopyStorage
Canopy storage [mm]. Used in the Gash interception model.

N and N_River
Manning N parameter for the kinematic wave function for overland and river flow. Higher valuesdampen the discharge peak.

PathFrac
Fraction of compacted area per gridcell [-].

RootingDepth
Rooting depth of the vegetation [mm].

thetaR
Residual water content [mm/mm].

thetaS
Water content at saturation [mm/mm].

c
Brooks-Corey power coefficient [-] based on the pore size distribution index λ, used for computing vertical unsaturated flow.

Note: If SoilThickness and SoilMinThickness are not equal, wflow_sbm will scale SoilThickness based on the topographic wetness index.

Implementation of SoilThickness scaling:

```
# soil thickness based on topographic wetness index (see Environmental modelling:␣
↪finding simplicity in complexity)
# 1: calculate wetness index
# 2: Scale the soil thickness (now actually a max) based on the index, also apply a␣
↪minimum soil thickness
WI = pcr.ln(
    pcr.accuflux(self.TopoLdd, 1) / self.landSlope
)  # Topographic wetness index. Scale WI by zone/subcatchment assuming these are also␣
↪geological units
WIMax = pcr.areamaximum(WI, self.TopoId) * WIMaxScale
self.SoilThickness = pcr.max(
    pcr.min(self.SoilThickness, (WI / WIMax) * self.SoilThickness),
    self.SoilMinThickness,
)
```

Calibrating the wflow_sbm model

Introduction

As with all hydrological models calibration is needed for optimal performance. We have calibrated different wflow_sbm models using simple shell scripts and command-line parameters to multiply selected model parameters and evaluate the results later.

Parameters

SoilThickness
Increasing the soil depth and thus the storage capacity of the soil will decrease the outflow.

M
Once the depth of the soil has been set (e.g. for different land-use types) the M parameter is the most important variable in calibrating the model. The decay of the conductivity with depth controls the baseflow resession and part of the stormflow curve.

N and N_River
The Manning N parameter controls the shape of the hydrograph (the peak parts). In general it is advised to set N to realistic values for the rivers, for the land phase higher values are usually needed.

KsatVer and KsatHorFrac
Increasing KsatVer and or KsatHorFrac will lower the hydrograph (baseflow) and flatten the peaks. The latter also depend on the shape of the catchment.

References

- Vertessy, R.A. and Elsenbeer, H., 1999, Distributed modelling of storm flow generation in an Amazonian rainforest catchment: effects of model parameterization, Water Resources Research, vol. 35, no. 7, pp. 2173–2187.
- Brooks, R., and Corey, T., 1964, Hydraulic properties of porous media, Hydrology Papers, Colorado State University, 24, doi:10.13031/2013. 40684.
- Chow, V., Maidment, D. and Mays, L., 1988, Applied Hydrology. McGraw-Hill Book Company, New York.
- Finnegan, N.J., Roe, G., Montgomery, D.R., and Hallet, B., 2005, Controls on the channel width of rivers: Implications for modeling fluvial incision of bedrock, Geology, v. 33; no. 3; p. 229–232; doi: 10.1130/G21171.1.
- Wigmosta, M. S., L. J. Lane, J. D. Tagestad, and A. M. Coleman, 2009, Hydrologic and erosion models to assess land use and management practices affecting soil erosion, Journal of Hydrologic Engineering, 14(1), 27-41.

wflow_sbm module documentation

1.5.3 The wflow_gr4 model

Warning: The documentation is incomplete

Introduction

An *experimental* implementation of the gr4 model. It is based on the hourly (gr4h) version

Dependencies

[PM]

Configuration

The model needs a number of settings in the ini file. The default name for the ini file is wflow_gr4.ini.

See below for an example:

```
[model]

Tslice=1
# Maximum upstream distance to update the flow in metres


[gr4]
dt = 1
B = 0.9
D = 1.25
X4 = 32.83
# X1,X2 and X3 are given as .tbl files or maps

[layout]
# if set to zero the cell-size is given in lat/long (the default)
sizeinmetres = 1
```

(continues on next page)

(continued from previous page)

```
[outputmaps]
# Add maps here

# List all timeseries in tss format to save in this section. Timeseries are
# produced as averages per subcatchment. The discharge (run) timeseries
# is always saved (as samples at the gauge location)s.
[outputtss]
self.S_X1=S_X1
self.R_X3=R_X3
self.Pr=Pr
self.Q=Q
```

wflow_gr4 module documentation

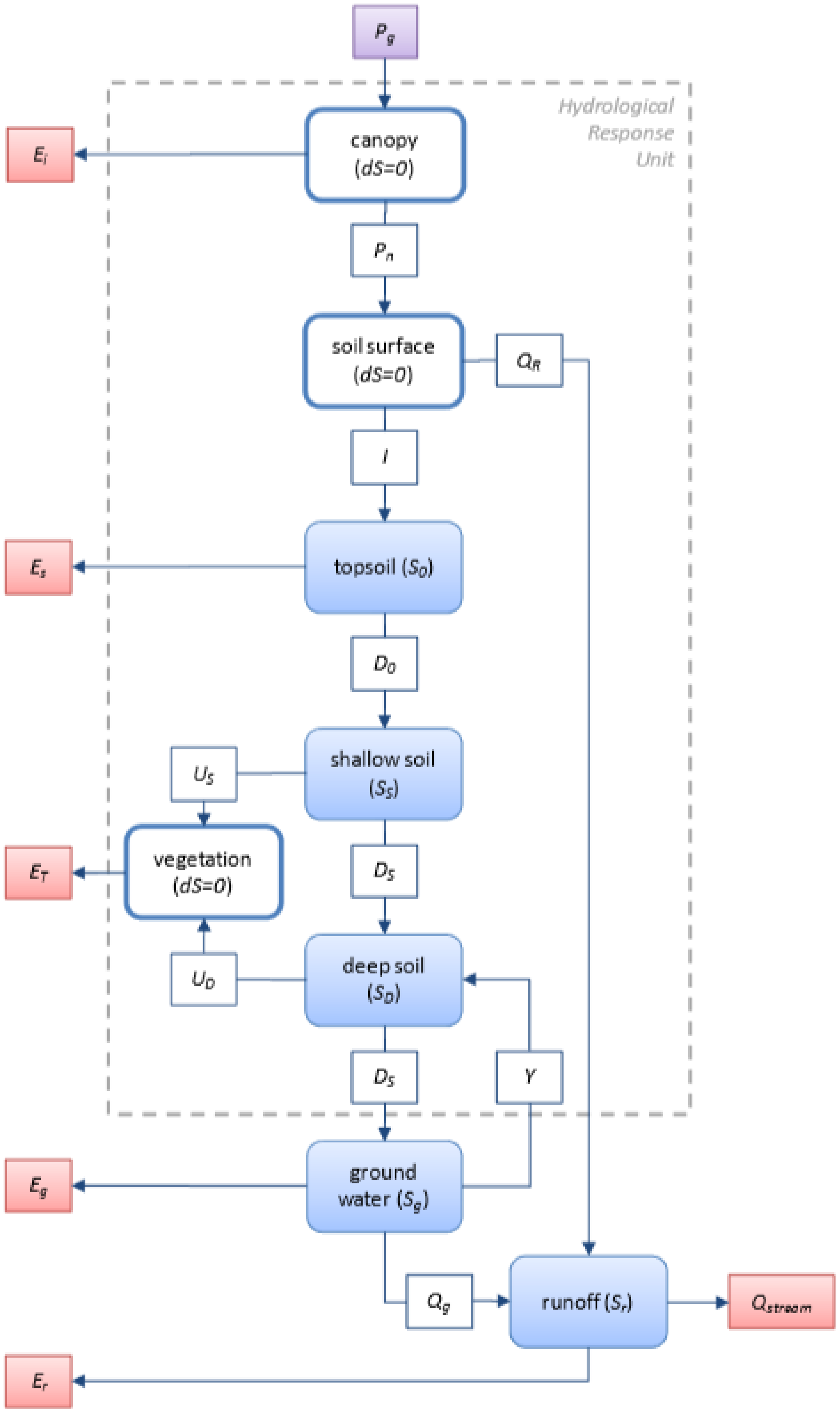
1.5.4 The wflow_W3RA and wflow_w3 Models

Warning: The documentation of this model is incomplete

Introduction

The wflow_w3ra and wflow_w3 models are adaptations of the Australian Water Resources Assessment Landscape model (AWRA-L). The AWRA-L model is developped through the Water Information Research and Development Alliance, the Bureau of Meteorology and CSIRO. It an hydrologic model that produces water balance component estimates for surface runoff, root water uptake, soil water drainage, groundwater discharge, capillary rise and streamflow. Radiation and energy balance, Vapour fluxes and vegetation phenology are also included.

The following Figure describes the water stores and fluxes considered in AWRA-L:



For more information, please refer to the dedicated model documentation: Van Dijk, A. I. J. M. 2010. The Australian Water Resources Assessment System. Technical Report 3. Landscape Model (version 0.5) Technical Description.

CSIRO: Water for a Healthy Country National Research Flagship.

Dependencies

Configuration

It needs a number of settings in the ini file. The default name for the file is wflow_w3ra.ini or wflow_w3.ini.

Examples are available in \wflow\examples\wflow_w3ra\ and \wflow\examples\wflow_rhine_w3\.

wflow_w3ra module documentation

1.5.5 The wflow_topoflex Model

Introduction

Topoflex applies the concept of using different model configurations for different hydrological response units (HRUs) . These HRUs can for a large part be derived from topography ([savenije]); however, additional data such as land use and geology can further optimize the selection of hydrological response units. In contrast to other models, topoflex generally uses a minimum amount of HRUs, which are defined manually, i.e. 2-5 depending on the size of and the variablity within the catchment. Nevertheless, the model code is written such that it can handle an infinte number of classes. The individual HRUs are treated as parallel model structures and only connected via the groundwater reservoir and the stream network.

The model code is written in a modular setup: different conceptualisations can be selected for each reservoir for each HRU. The code currently contains some reservoir conceptualisations (see below), but new conceptualisations can easily be added.

Examples of the application of topoflex can be found in Gharari et al. (2014), Gao et al. (2014) and Euser et al. (2015).

Figure 1 shows a possible model conceptualistion: one that used two HRUs (wetland (W) and hillslope (H), adapted from [euser])

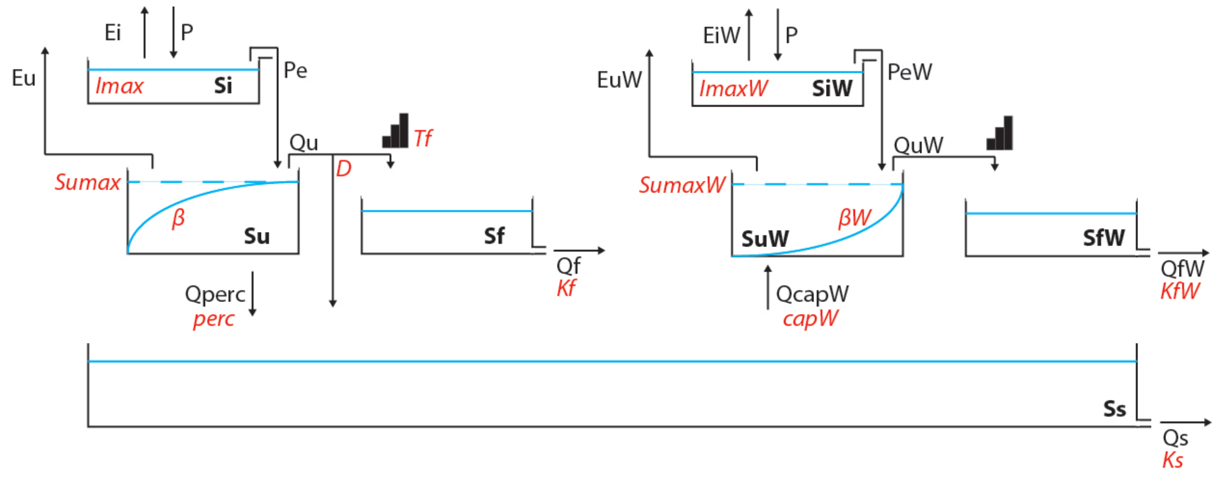


Fig. 11: Schematic view of the relevant components of the topoflex model

Limitations

- Using a set of HRUs introduces additional complexity (structural and computational) in the model. Therefore, calibration of the model should be carried out carefully (see below for some tips and tricks that might help).
- The selection and deliniation of the HRUs is a relatively subjective exercise: different data sources and preferably some expert knowledge migth help to construct meaningful HRUs.

Spatial resolution

The model uses a grid based on the available input or required output data. The combination with the HRUs has to be done in the preparation step: for each cell the percentage of each class needs to be determined and stored in a staticmap.

The cell sizes do not have to have the same size: the size of individual cells can be stored in a staticmap, which is used to calculate the contribution of each cell to the generated discharge.

Input data

The required input data consists of timeseries of precipitation, temperature potential evaporation. In case the Jarvis equations are used to determine transpiration more input variable are required, such as humidity, radiation, wind speed and LAI.

Different HRUs

Currently there are conceptualisations for three main HRUs: wetland, hillslope and (agricultural) plateau. These conceptualisations are simply a set of reservoirs, which match the perception of different landscape elements (in western Europe). Dependent on the area or interest the HRUs can be adapted, was well as the set of reservoirs per HRU.

wetland

Wetlands are areas close to the stream, where the groundwater levels are assumed to be shallow and assumed to rise quickly during an event. The dominant runoff process in wetlands is assumed to be saturation overland flow, leading to quick and sharp responses to rainfall.

hillslope

Hillslopes are steep areas in an catchment, generally covered with forest. The dominant runoff process is assumed to be quick subsurface flow: the hillslopes mainly contribute to the discharge during the winter period.

(agricultural) plateau

Plateus are flat areas high above the stream, thus with deep ground water levels. Depending on the specific conditions, the dominant runoff processes are ground water recharge, quick subsurface flow and hortonian overland flow. The latter is especially important in agricultural areas.

Other modelling options

Routing of generated discharge

The routing of generated discharge is based on the average velocity of water through the river, which is currently set to 1 m/s. For each cell the average distance to the outlet is calculated and multiplied with the selected flow velocity to determine the delay at the outlet.

There are currently two options to apply this routing:

1. only calculating the delay relevant for the discharge at the outlet
2. calculating the delay (and thus discharge) over the stream network. This option is mainly relevant for calculations with a finer grid

Calibrating the wflow_topoflex model

Including more HRUs in the model leads to an increase in parameters. To make it still feasible to calibrate the model, a set of constraints is introduced: parameter and process constraints. These constraints are assumed relations

> between parameters and fluxes of different HRUs and prevent the selection of unreaslistic parameters. The constraints are an important part of the perceptual model, but are not (yet) included in de wflow code. Below some examples of constraints are given, more examples of constraints can be found in Gharari et al. (2014), Gao et al. (2014) and Euser et al. (2015).

Parameter constraints

Parameter constraints are relations between parameters of different HRUs, for example the root zone storage capacity (S_{u,max}), which is assumed to be larger on hillslopes than in wetlands. As in the latter groundwater levels quicky rise during a rain event, reducing the root zone storage capacity. Parameter constraints are calculated before the model runs.

Process constraints

Process constraints are comparable with parameter constraints, but focus on fluxes from different HRUs, for example the fast response from the wetlands is assumed to be larger than the fast response of the hillslopes in the summer period. As on the hillslopes generally more storage is required before a runoff generation threshold is exceeded. Process constraints are calculated after the model runs.

References

- Euser, T., Hrachowitz, M., Winsemius, H. C. and Savenije, H. H. G.: The effect of forcing and landscape distribution on performance and consistency of model structures. Hydrol. Process., doi: 10.1002/hyp.10445, 2015.
- Gao, H., Hrachowitz, M., Fenicia, F., Gharari, S., and Savenije, H. H. G.: Testing the realism of a topography-driven model (FLEX-Topo) in the nested catchments of the Upper Heihe, China, Hydrol. Earth Syst. Sci., 18, 1895-1915, doi:10.5194/hess-18-1895-2014, 2014.
- Gharari, S., Hrachowitz, M., Fenicia, F., Gao, H., and Savenije, H. H. G.: Using expert knowledge to increase realism in environmental system models can dramatically reduce the need for calibration, Hydrol. Earth Syst. Sci., 18, 4839-4859, doi:10.5194/hess-18-4839-2014, 2014.
- Savenije, H. H. G.: HESS Opinions "Topography driven conceptual modelling (FLEX-Topo)", Hydrol. Earth Syst. Sci., 14, 2681-2692, doi:10.5194/hess-14-2681-2010, 2010.

example ini file

The .ini file below shows the available options

```
[framework]
outputformat = 1

# Model parameters and settings
[model]
## The settings below define the way input data is handeled
# ScalarInput: 0 = input from mapstacks (.map); 1 = input from timeseries (.tss)
ScalarInput=1
# InputSeries can be used if multiple sets of input series are available, TE: maybe␣
↪better to remove this option?
InputSeries=1
# InterpolationMethod: inv = inverse distance,  pol = Thiessen polygons, this option is␣
↪not relevant if the number of cells is equal to the number of meteo gauges
InterpolationMethod=inv

##  Remaining model settings
# L_IR(UR(FR)) indicates with reservoirs are treated lumped with regard to moisture␣
↪states. 0 = distributed; 1 = lumped. IR = interception reservoir; UR = unsaturated␣
↪zone reservoir; FR = fast runoff reservoir
L_IRURFR = 0
L_URFR = 0
L_FR = 0
# maxTransitTime is the travel time (same time resolution as model calculations) through␣
↪the stream of the most upstream point, rounded up
maxTransitTime = 9
# DistForcing is the number of used rainfall gauges
DistForcing = 3
# maxGaugeId is the id of the raingauge with the highest id-number, this setting has to␣
↪do with writing the output data for the correct stations
maxGaugeId = 10
# timestepsecs is the number of seconds per time step
timestepsecs = 3600

## the settings below deal with the selection of classes and conceptualisations of␣
↪reservoirs
## classes indicates the number of classes, the specific characters are not important␣
↪(will be used for writing output??), but the number of sets of characters is important
classes = ['W','H']

## selection of reservoir configuration - 'None' means reservoir should not be modelled␣
↪for this class
selectSi = interception_overflow2, interception_overflow2
selectSu= unsatZone_LP_beta, unsatZone_LP_beta
selectSus=None,None
selectSf=fastRunoff_lag2, fastRunoff_lag2
selectSr=None,None

## selection of Ss (lumped over entire catchment, so only one!)
# selectSs = <some other reservoir than groundWaterCombined3>
```

(continues on next page)

(continued from previous page)

```
## wflow maps with percentages, the numbers correspond to the indices of the characters␣
↪in classes.
wflow_percent_0 = staticmaps/wflow_percentW4.map
wflow_percent_1 = staticmaps/wflow_percentHPPPA4.map

##constant model parameters - some are catchment dependent
Ks = 0.0004
lamda = 2.45e6
Cp = 1.01e-3
rhoA = 1.29
rhoW = 1000
gamma = 0.066
JC_Topt = 301

#parameters for fluxes and storages
sumax = [140, 300]
beta = [0.1, 0.1]
D = [0, 0.10]
Kf = [0.1, 0.006]
Tf = [1, 3]
imax = [1.2, 2]
perc = [0, 0.000]
cap = [0.09, 0]
LP = [0.5, 0.8]
Ce = [1, 1]

#parameters for Jarvis stressfunctions
JC_D05 = [1.5,1.5]
JC_cd1 = [3,3]
JC_cd2 = [0.1,0.1]
JC_cr = [100,100]
JC_cuz = [0.07,0.07]
SuFC = [0.98,0.98]
SuWP = [0.1,0.1]
JC_rstmin = [150,250]

[layout]
# if set to zero the cell-size is given in lat/long (the default)
sizeinmetres = 0

[outputmaps]
#self.Si_diff=sidiff
#self.Pe=pe
#self.Ei=Ei

# List all timeseries in tss format to save in this section. Timeseries are
# produced per subcatchment.

[outputtss_0]
samplemap=staticmaps/wflow_mgauges.map
#states
```

(continues on next page)

(continued from previous page)

```
self.Si[0]=SiW.tss
self.Si[1]=SiH.tss
self.Sf[1]=SfH.tss
self.Sf[0]=SfW.tss
self.Su[1]=SuH.tss
self.Su[0]=SuW.tss
#fluxen
self.Precipitation=Prec.tss
self.Qu_[0]=QuW.tss
self.Qu_[1]=QuH.tss
self.Ei_[0]=EiW.tss
self.Ei_[1]=EiH.tss
self.Eu_[0]=EuW.tss
self.Eu_[1]=EuH.tss
self.Pe_[0]=peW.tss
self.Pe_[1]=peH.tss
self.Perc_[1]=PercH.tss
self.Cap_[0]=CapW.tss
self.Qf_[1] = QfH.tss
self.Qf_[0] = QfW.tss
self.Qfcub = Qfcub.tss
self.Qtlag = Qtlag.tss

[outputtss_1]
samplemap = staticmaps/wflow_gauges.map
#states
self.Ss = Ss.tss
#fluxen
self.Qs = Qs.tss
self.QLagTot = runLag.tss
self.WBtot = WB.tss

[modelparameters]
# Format:
# name=stack,type,default,verbose[lookupmap_1],[lookupmap_2],lookupmap_n]
# example:
# RootingDepth=monthlyclim/ROOT,monthyclim,100,1

# - name - Name of the parameter (internal variable)
# - stack - Name of the mapstack (representation on disk or in mem) relative to case
# - type - Type of parameter (default = static)
# - default - Default value if map/tbl is not present
# - set to 1 to be verbose if a map is missing
# - lookupmap - maps to be used in the lookuptable in the case the type is statictbl

#Possible types are::
# - staticmap: Read at startup from map
# - statictbl: Read at startup from tbl
# - tbl: Read each timestep from tbl and at startup
# - timeseries: read map for each timestep
# - monthlyclim: read a map corresponding to the current month (12 maps in total)
# - dailyclim: read a map corresponding to the current day of the year
```

(continues on next page)

(continued from previous page)

```
# - hourlyclim: read a map corresponding to the current hour of the day (24 in total)␣
↪(not implemented yet)
# - tss: read a tss file and link to lookupmap (only one allowed) a map using␣
↪timeinputscalar
Precipitation=intss/1_P.tss,tss,0.0,1,staticmaps/wflow_mgauges.map
Temperature=intss/1_T.tss,tss,10.5,1,staticmaps/wflow_mgauges.map
PotEvaporation=intss/1_PET.tss,tss,0.0,1,staticmaps/wflow_mgauges.map
```

An example ini file be found `here`.

wflow_topoflex module documentation

1.5.6 The wflow_pcrglobwb Model

Introduction

In 2018 the following PCR-GLOBWB ((PCRaster Global Water Balance) version was added to the wflow framework:

https://github.com/UU-Hydro/PCR-GLOBWB_model/tree/v2.1.0_beta_1

Changes made to the PCR-GLOBWB code

- The original code was converted from Python 2 to Python 3.
- For the different modules available in PCR-GLOBWB:
  - groundwater
  - landCover
  - landSurface
  - ncConverter
  - parameterSoilAndTopo
  - routing
  - virtualOS
  - waterBodies

  only the initialization function (including the states) was changed, to initialize these classes from wflow_pcrglobwb properly, so PCR-GLOBWB complies with the directory structure of wflow. Furthermore the checks in virtualOS whether the clone map and input maps (e.g. forcing) have the same attributes (cell size and domain), are switched off, including possible cropping and resampling of input maps when clone map and input maps don't have the same attributes.
- The wflow framework takes care of saving output, and the Reporting module of PCR-GLOBWB is not included. Also, the spinUp module of PCR-GLOBWB is not included in wflow. Initial conditions (cold state) can be set in the wflow_pcrglobwb.ini file as follows:

  ```
  [forestOptions]
  # initial conditions:
  interceptStorIni  = 0.0
  ```

(continues on next page)

(continued from previous page)

```
snowCoverSWEIni   = 0.0
snowFreeWaterIni  = 0.0
topWaterLayerIni  = 0.0
storUpp000005Ini  = 0.0
storUpp005030Ini  = 0.0
storLow030150Ini  = 0.0
interflowIni      = 0.0
```

or default initial conditions (set in the code) are used when these are not set in the ini file. Warm states can be set in the ini file as follows:

```
[forestOptions]
# initial conditions:
interceptStorIni    = landSurface.interceptStor_forest.map
snowCoverSWEIni     = landSurface.snowCoverSWE_forest.map
snowFreeWaterIni    = landSurface.snowFreeWater_forest.map
topWaterLayerIni    = landSurface.topWaterLayer_forest.map
storUppIni          = landSurface.storUpp005030_forest.map
storLowIni          = landSurface.storLow030150_forest.map
interflowIni        = landSurface.interflow_forest.map
```

and should be available in the instate directory of the Case directory.

Ini file settings

Below an example of a wflow_pcrglobwb.ini file:

```
[framework]
netcdfoutput = outmaps.nc
netcdfinput = inmaps/forcing.nc
netcdfwritebuffer=20
EPSG = EPSG:4326

[run]
# either a runinfo file or a start and end-time are required
starttime= 2002-01-01 00:00:00
endtime= 2002-01-30 00:00:00
reinit = 0
timestepsecs = 86400
runlengthdetermination=steps

[model]
modeltype = wflow_pcrglobwb

[layout]
# if set to zero the cell-size is given in lat/long (the default)
sizeinmetres = 0

[outputmaps]
self.routing.subDischarge = Qro
self.routing.waterBodyStorage = wbs
self.landSurface.storUpp = su1
```

(continues on next page)

(continued from previous page)

```
self.landSurface.storLow = slo
self.landSurface.actualET = aet
self.landsurface.swAbstractionFractionData = swAbsF
self.landSurface.totalPotET = pet
self.landSurface.gwRecharge = gwr
self.landSurface.snowCoverSWE = swe
self.groundwater.nonFossilGroundwaterAbs = nFAbs
self.landSurface.fossilGroundwaterAbstr = FAbs
self.landSurface.irrGrossDemand = IrrGD
self.landSurface.nonIrrGrossDemand = nIrrGD

[globalOptions]

# Map of clone (must be provided in PCRaster maps)
# - Spatial resolution and coverage are based on this map:
cloneMap = wflow_clone.map

# The area/landmask of interest
landmask = mask.map
# If None, area/landmask is limited for cells with ldd value.

[landSurfaceOptions]
debugWaterBalance = True

numberOfUpperSoilLayers = 2

topographyNC     = topoProperties5ArcMin.nc
soilPropertiesNC = soilProperties5ArcMin.nc

includeIrrigation = True

# a pcraster map/value defining irrigation efficiency (dimensionless) - optional
irrigationEfficiency = efficiency.map

# netcdf time series for historical expansion of irrigation areas (unit: hectares).
# Note: The resolution of this map must be consisten with the resolution of cellArea.
historicalIrrigationArea = irrigationArea05ArcMin.nc

includeDomesticWaterDemand = True
includeIndustryWaterDemand = True
includeLivestockWaterDemand = True

# domestic and industrial water demand data (unit must be in m.day-1)
domesticWaterDemandFile = domestic_water_demand_version_april_2015.nc
industryWaterDemandFile = industry_water_demand_version_april_2015.nc
livestockWaterDemandFile = livestock_water_demand_version_april_2015.nc

# desalination water supply (maximum/potential/capacity)
#desalinationWater = desalination_water_version_april_2015.nc # should be included
# zone IDs (scale) at which allocations of groundwater and surface water (as well as␣
↪desalinated water) are performed
allocationSegmentsForGroundSurfaceWater = uniqueIds60min.nom_5min.map
```

(continues on next page)

(continued from previous page)

```
# predefined surface water - groundwater partitioning for irrigation demand (based on␣
↪Siebert, 2010/2013: Global Map of Irrigation Areas version 5):
irrigationSurfaceWaterAbstractionFractionData        = AEI_SWFRAC_5min.map
irrigationSurfaceWaterAbstractionFractionDataQuality = AEI_QUAL_5min.map


# predefined surface water - groundwater partitioning for irrigation demand (based on␣
↪McDonald, 2014):
maximumNonIrrigationSurfaceWaterAbstractionFractionData = max_city_sw_fraction_5min.map

[forestOptions]
name = forest
debugWaterBalance = True

# snow module properties
snowModuleType      =  Simple
freezingT           = 0.0
degreeDayFactor     =  0.0025
snowWaterHoldingCap =  0.1
refreezingCoeff     =  0.05

# other paramater values
minTopWaterLayer = 0.0
minCropKC        = 0.2
minInterceptCap  = 0.0002

landCoverMapsNC = forestProperties5ArcMin.nc

# Parameters for the Arno's scheme:
arnoBeta = None
# If arnoBeta is defined, the soil water capacity distribution is based on this.
# If arnoBeta is NOT defined, maxSoilDepthFrac must be defined such that arnoBeta will␣
↪be calculated based on maxSoilDepthFrac and minSoilDepthFrac.

cropCoefficientNC = cropKC_forest_daily366.nc
interceptCapNC    = interceptCap_forest_daily366.nc
coverFractionNC   = coverFraction_forest_daily366.nc

# initial conditions:
interceptStorIni  = landSurface.interceptStor_forest.map
snowCoverSWEIni   = landSurface.snowCoverSWE_forest.map
snowFreeWaterIni  = landSurface.snowFreeWater_forest.map
topWaterLayerIni  = landSurface.topWaterLayer_forest.map
storUppIni  = landSurface.storUpp005030_forest.map
storLowIni  = landSurface.storLow030150_forest.map
interflowIni      = landSurface.interflow_forest.map

[grasslandOptions]
name = grassland
debugWaterBalance = True
```

(continues on next page)

(continued from previous page)

```
# snow module properties
snowModuleType      =  Simple
freezingT           = 0.0
degreeDayFactor     =  0.0025
snowWaterHoldingCap =  0.1
refreezingCoeff     =  0.05

# other paramater values
minTopWaterLayer = 0.0
minCropKC        = 0.2
minInterceptCap  = 0.0002

landCoverMapsNC = grasslandProperties5ArcMin.nc
#
# Parameters for the Arno's scheme:
arnoBeta = None
# If arnoBeta is defined, the soil water capacity distribution is based on this.
# If arnoBeta is NOT defined, maxSoilDepthFrac must be defined such that arnoBeta will␣
↪be calculated based on maxSoilDepthFrac and minSoilDepthFrac.

cropCoefficientNC = cropKC_grassland_daily366.nc
interceptCapNC    = interceptCap_grassland_daily366.nc
coverFractionNC   = coverFraction_grassland_daily366.nc

# initial conditions:
interceptStorIni  = landSurface.interceptStor_grassland.map
snowCoverSWEIni   = landSurface.snowCoverSWE_grassland.map
snowFreeWaterIni  = landSurface.snowFreeWater_grassland.map
topWaterLayerIni  = landSurface.topWaterLayer_grassland.map
#storUpp000005Ini  = landSurface.storUpp000005_grassland.map
storUppIni  = landSurface.storUpp005030_grassland.map
storLowIni  = landSurface.storLow030150_grassland.map
interflowIni      = landSurface.interflow_grassland.map

[irrPaddyOptions]
name = irrPaddy
debugWaterBalance = True

# snow module properties
snowModuleType      =  Simple
freezingT           = -0.0
degreeDayFactor     =  0.0025
snowWaterHoldingCap =  0.1
refreezingCoeff     =  0.05
#
landCoverMapsNC = irrPaddyProperties30min.nc
maxRootDepth     = 0.5
#
# Parameters for the Arno's scheme:
arnoBeta = None
# If arnoBeta is defined, the soil water capacity distribution is based on this.
# If arnoBeta is NOT defined, maxSoilDepthFrac must be defined such that arnoBeta will␣
↪be calculated based on maxSoilDepthFrac and minSoilDepthFrac.
```

(continues on next page)

(continued from previous page)

```
#
# other paramater values
minTopWaterLayer = 0.05
minCropKC         = 0.2
minInterceptCap  = 0.0002
cropDeplFactor   = 0.2

cropCoefficientNC = cropKC_irrPaddy_daily366.nc

# initial conditions:
interceptStorIni  = landSurface.interceptStor_irrPaddy.map
snowCoverSWEIni   = landSurface.snowCoverSWE_irrPaddy.map
snowFreeWaterIni  = landSurface.snowFreeWater_irrPaddy.map
topWaterLayerIni  = landSurface.topWaterLayer_irrPaddy.map
#storUpp000005Ini  = landSurface.storUpp000005_irrPaddy.map
storUppIni  = landSurface.storUpp005030_irrPaddy.map
storLowIni  = landSurface.storLow030150_irrPaddy.map
interflowIni      = landSurface.interflow_irrPaddy.map

[irrNonPaddyOptions]
name = irrNonPaddy
debugWaterBalance = True

# snow module properties
snowModuleType      =  Simple
freezingT           = -0.0
degreeDayFactor     =  0.0025
snowWaterHoldingCap =  0.1
refreezingCoeff     =  0.05
#
landCoverMapsNC  = irrNonPaddyProperties30min.nc
maxRootDepth     = 1.0
#
# Parameters for the Arno's scheme:
arnoBeta = None
# If arnoBeta is defined, the soil water capacity distribution is based on this.
# If arnoBeta is NOT defined, maxSoilDepthFrac must be defined such that arnoBeta will␣
↪be calculated based on maxSoilDepthFrac and minSoilDepthFrac.
#
# other paramater values
minTopWaterLayer = 0.0
minCropKC         = 0.2
minInterceptCap  = 0.0002
cropDeplFactor   = 0.5

cropCoefficientNC = cropKC_irrNonPaddy_daily366.nc

# initial conditions:
interceptStorIni  = landSurface.interceptStor_irrNonPaddy.map
snowCoverSWEIni   = landSurface.snowCoverSWE_irrNonPaddy.map
snowFreeWaterIni  = landSurface.snowFreeWater_irrNonPaddy.map
topWaterLayerIni  = landSurface.topWaterLayer_irrNonPaddy.map
```

(continues on next page)

(continued from previous page)

```
#storUpp000005Ini  = landSurface.storUpp000005_irrNonPaddy.map
storUppIni  = landSurface.storUpp005030_irrNonPaddy.map
storLowIni  = landSurface.storLow030150_irrNonPaddy.map
interflowIni      = landSurface.interflow_irrNonPaddy.map

[groundwaterOptions]

debugWaterBalance = True
groundwaterPropertiesNC = groundwaterProperties5ArcMin_5min.nc

# minimum value for groundwater recession coefficient (day-1)
minRecessionCoeff = 1.0e-4

limitFossilGroundWaterAbstraction = True
minimumTotalGroundwaterThickness       = 0.000
estimateOfTotalGroundwaterThickness    = thickness_05min_5min.map
estimateOfRenewableGroundwaterCapacity = 0.0

# annual pumping capacity for each region (unit: billion cubic meter per year), should␣
↪be given in a netcdf file
pumpingCapacityNC = regional_abstraction_limit_5min.nc


# initial conditions:
storGroundwaterIni = groundwater.storGroundwater.map
storGroundwaterFossilIni = groundwater.storGroundwaterFossil.map
#
avgNonFossilGroundwaterAllocationLongIni  = groundwater.avgNonFossilAllocation.map
avgNonFossilGroundwaterAllocationShortIni = groundwater.avgNonFossilAllocationShort.map
avgTotalGroundwaterAbstractionIni         = groundwater.avgAbstraction.map
avgTotalGroundwaterAllocationLongIni      = groundwater.avgAllocation.map
avgTotalGroundwaterAllocationShortIni     = groundwater.avgAllocationShort.map

allocationSegmentsForGroundwater = uniqueIds30min.nom_5min.map
#~ allocationSegmentsForGroundwater = None

[routingOptions]
debugWaterBalance = True

lddMap      = lddsound_05min.map
cellAreaMap = cellarea05min.map
gradient    = ChannelGradient_05min.map

# manning coefficient
manningsN   = 0.04

routingMethod = accuTravelTime
# TODO: including kinematicWave
#~ # Maximum length of a sub time step in seconds (optional and only used if either␣
↪kinematicWave or simplifiedKinematicWave is used)
#~ # - Note that too long sub time step may create water balance errors.
#~ # - Default values: 3600 seconds for 30 arcmin ; 720 seconds for 5 arcmin
```

(continues on next page)

(continued from previous page)

```
#~ maxiumLengthOfSubTimeStep = 3600.
#~ maxiumLengthOfSubTimeStep = 720.

# dynamic flood plain options
dynamicFloodPlain = False

# lake and reservoir parameters
waterBodyInputNC = waterBodies5ArcMin.nc
onlyNaturalWaterBodies = False

# composite crop factors for WaterBodies:
cropCoefficientWaterNC = cropCoefficientForOpenWater.nc
minCropWaterKC         = 0.20

# number of days (timesteps) that have been performed for spinning up initial conditions␣
↪in the routing module (i.e. channelStorageIni, avgDischargeLongIni,␣
↪avgDischargeShortIni, etc.)
timestepsToAvgDischargeIni     = routing.timestepsToAvgDischarge.map
# Note that:
# - maximum number of days (timesteps) to calculate long term average flow values␣
↪(default: 5 years = 5 * 365 days = 1825)
# - maximum number of days (timesteps) to calculate short term average values (default:␣
↪1 month = 1 * 30 days = 30)

# initial conditions:
waterBodyStorageIni        = routing.waterBodyStorage.map
avgLakeReservoirInflowShortIni = routing.avgInflow.map
avgLakeReservoirOutflowLongIni = routing.avgOutflow.map
channelStorageIni              = routing.channelStorage.map
readAvlChannelStorageIni       = routing.readAvlChannelStorage.map
avgDischargeLongIni            = routing.avgDischarge.map
m2tDischargeLongIni            = routing.m2tDischarge.map
avgBaseflowLongIni             = routing.avgBaseflow.map
riverbedExchangeIni            = routing.riverbedExchange.map
avgDischargeShortIni           = routing.avgDischargeShort.map
subDischargeIni                = routing.subDischarge.map
```

An example model is available in \wflow\examples\wflow_RhineMeuse_pcrglobwb\.

wflow_pcrglobwb module documentation

1.5.7 The wflow_sphy Model

Introduction

In 2017 the hydrological model SPHY (Spatial Processes in HYdrology) version 2.1 was added to the wflow framework:

https://github.com/FutureWater/SPHY/tree/v2.1

An example model is available in \wflow\examples\wflow_ganga_sphy\.

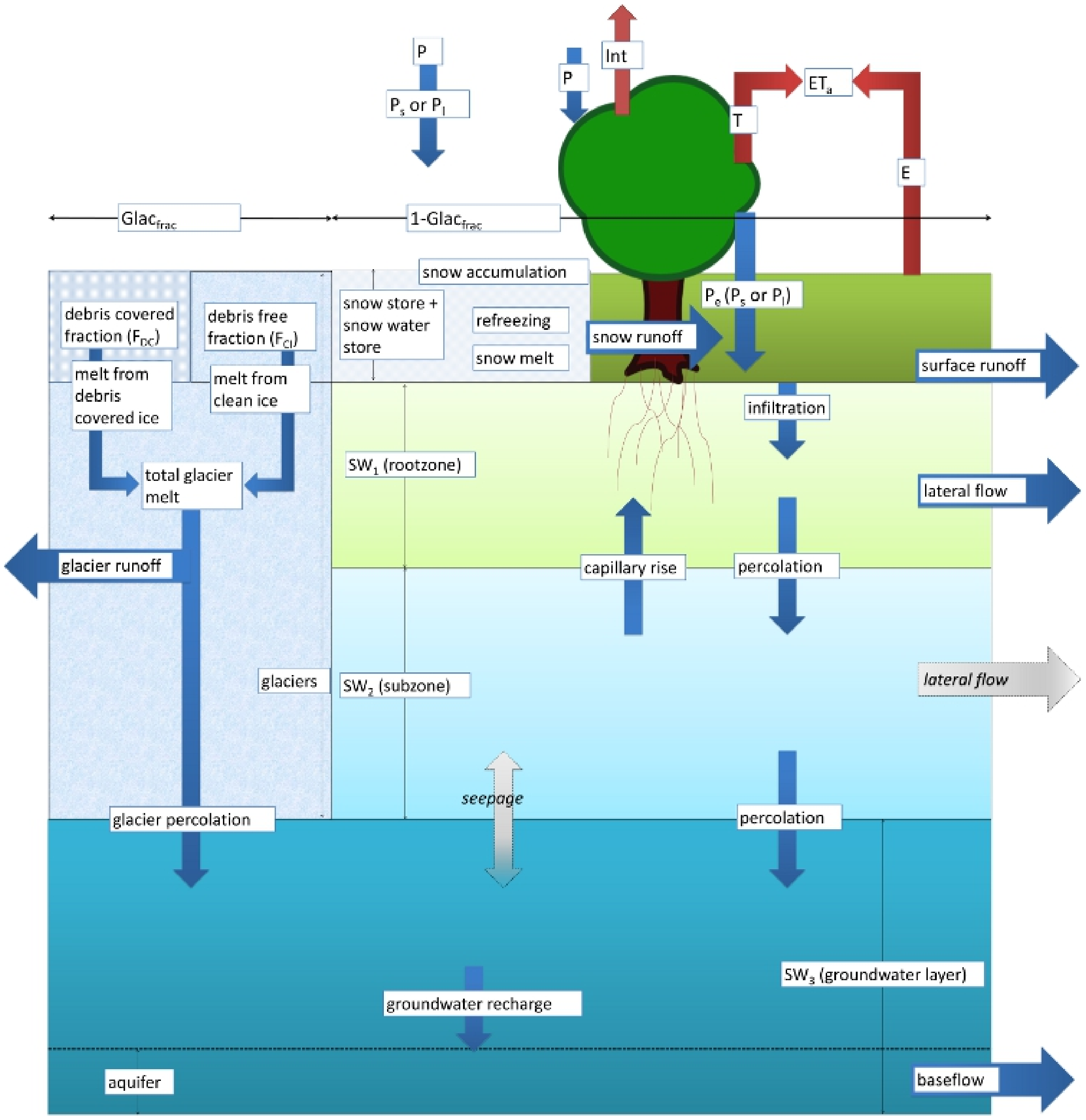


Fig. 12: Overview of the concepts of the wflow_sphy model (futurewater.eu).

1.5.8 The wflow_stream Model

Introduction

STREAM (Spatial Tools for River Basins and Environment and Analysis of Management Options) has been added to the wflow framework as part of the MSc. thesis work "Routing and calibration of distributed hydrological models" by Alkemade (2019).

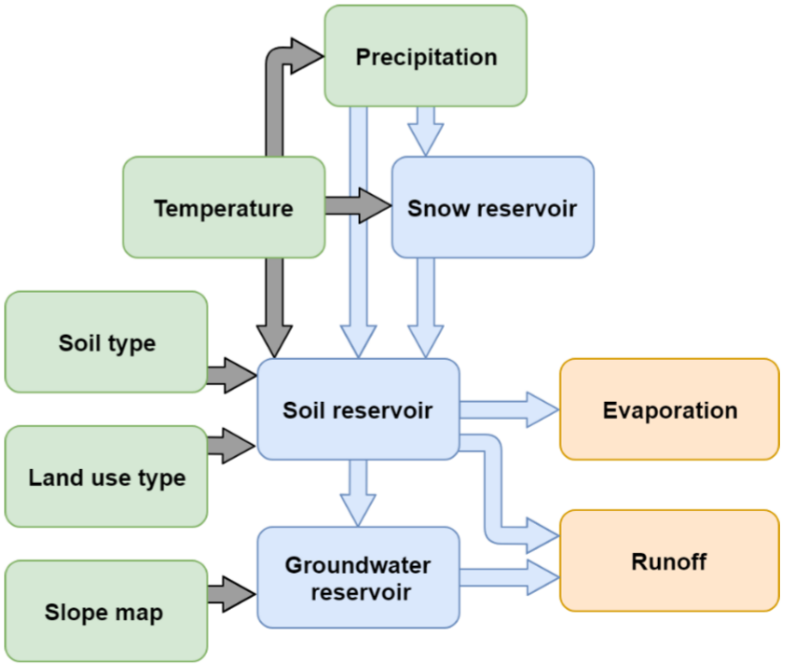


Fig. 13: Conceptual flow chart of STREAM.

STREAM is a hydrological model that was developed by Aerts et al. (1999). It is a distributed grid-based water-balance model, and as shown in the flow chart of STREAM it consists of a snow, soil and groundwater reservoir, and has a fast and slow runoff component. The soil moisture storage affects the fast component and ground water storage affects the slow component. Both behave as linear reservoirs. Potential evapotranspiration is derived from temperature, based on the Thornthwaite and Mather (1957) approach. Snow accumulation is equal to precipitation when temperature is below a threshold (e.g. zero degrees Celsius), and snow melts linearly depending on temperature. The fast and slow flows are routed to the catchment outlet by flow accumulation (based on a DEM) and assuming that all water moves through the system in one model time step (monthly).

Implementation in wflow

To add STREAM to the wflow framework, the model was re-written in the PCRaster Python framework. STREAM has until now mainly been used on monthly time step and calculates discharge by flow accumulation of upstream runoff. The following adjustment was made to run STREAM at a daily timestep:

- The original STREAM does not require any evaporation data and instead estimates it from temperature data on a monthly basis using the Thornthwaite and Mather (1957) approach. For wflow the calculation of evapotranspiration via temperature was taken out and potential evapotranspiration data is used as forcing to STREAM. This means that the wflow version of STREAM now requires not two but three forcings (precipitation, temperature and potential evapotranspiration).

Wflow_stream requires the following static maps, for the following parameters:

- whc [mm]
- C [day]
- cropf [-]
- meltf [-]
- togwf [-]

The parameter C is a recession coefficient and used in the draining of the groundwater reservoir. In the original STREAM model, this parameter is based on a combination of a categorical variable C with values 1, 2, 3 and 4 from the average slope in a cell (from a DEM), with higher slopes getting lower values and therefore a faster draining of the groundwater reservoir, and a global parameter Ccal that steers how fast groundwater flows (C * Ccal). The parameter whc represents the maximum soil water holding capacity. The parameter cropf represents the crop factor, to determine the actual evapotranspiration from the potential evapotranspiration. Parameter meltf is a melt factor that controls the rate of snow melt. Parameter togwf seperates the fraction of excess water going to groundwater and direct runoff.

Snow modelling

Snow accumulates below a surface temperature of 3.0 ^oC. Snow melts linearly based on surface temperature and a melt factor self.MELTcal.

```python
# snow routine
snowfall = self.precipitation
snowfall = scalar(self.temperature < 3.0) * snowfall
self.snow = self.snow + snowfall
melt = (self.MELTcal * self.temperature)
melt = scalar(self.temperature > 3.0) * melt
melt = max(0.0, min(self.snow, melt))
self.snow = self.snow - melt
self.precipitation = self.precipitation - snowfall + melt
```

Soil water balance

The Thornthwaite-Mather procedure (Thornthwaite and Mather, 1955) is used for modelling the soil water balance:

1. When $P - PET < 0$ (the soil is drying), available water soil water and excess water is:

$$AW_t = AW_{t-1} exp(\frac{(P - PET)}{WHC})$$
$$Excess = 0$$

where AW is available soil water, P is precipitation, PET is potential evapotranspiration, WHC is the soil water holding capacity and $Excess$ is all the water above WHC.

2. When the soil is wetting, $P - PET > 0$, but stays below WHC:

$$AW_{t-1} + (P - PET) < WHC$$

available water soil water and excess water is:

$$AW_t = AW_{t-1} + (P - PET)$$
$$Excess = 0$$

3. When the soil is wetting, $P - PET > 0$, above WHC:

$$AW_{t-1} + (P - PET) > WHC$$

available water soil water and excess water is:

$$AW_t = WHC$$
$$Excess = AW_{t-1} + (P - PET) - WHC$$

Groundwater and runoff

Excess water (self.excess) is seperated by a fraction (self.TOGWcal) into seepage (self.togw) to groundwater (self.Ground_water) and direct runoff. Seepage is added to the groundwater reservoir, and flow from the groundwater reservoir (self.sloflo) is modelled as a linear reservoir. Total runoff (self.runoff) consists of direct runoff and groundwater flow.

```
# seepage to groundwater
self.runoff = self.togwf * self.excess
self.togw = self.excess - self.runoff

# adding water to groundwater and taking water from groundwater
self.Ground_water = self.Ground_water + self.togw
self.sloflo = (self.Groundwater/ self.C )
self.Ground_water = self.Ground_water - self.sloflo

# adding water from groundwater to runoff
self.runoff = self.runoff + self.sloflo
```

References

- Aerts, J.C.J.H., Kriek, M., Schepel, M., 1999, STREAM (Spatial tools for river basins and environment and analysis of management options): ‘set up and requirements’, Phys. Chem. Earth, Part B Hydrol. Ocean. Atmos., 24(6), 591–595.
- Alkemade, G.I., 2019, Routing and calibration of distributed hydrological models, MSc. Thesis, VU Amsterdam, Faculty of Science, Hydrology.
- Thornthwaite, C.W., and Mather, J.R., 1955, The water balance, Publ. Climatol., 8(1), 1–104.
- Thornthwaite, C.W., and Mather, J.R., 1957, Instructions and tables for computing potential evapotranspiration and the water balance: Centerton, N.J., Laboratory of Climatology, Publ. Climatol., 10(3), 85–311.

wflow_stream module documentation

1.5.9 The wflow_routing Model

Introduction

The wflow routing module uses the pcraster kinematic wave to route water over a DEM. By adding a bankfull level and a floodplainwidth to the configuration the model can also include estimated flow over a floodplain. In addition, simple reservoirs can be configured.

Method

A simple river profile is defined using a river width a bankfull heigth and a floodplainwidth. A schematic drawing is shown in the figure below.

First the maximum bankfull flow for each cell is determined using:

$$Q_b = (\frac{H_b}{\alpha_{ch}} * Bw)^{1/\beta}$$

Next the channel flow is determined by taking the mimumm value of the total flow and the maximum banfull flow and the floodplain flow is determined by subtracting the bankfull flow from the total flow.

In normal conditions (below bankfull), the waterlevel in the river is determined as follows:

$$H_{ch} = \alpha_{ch} {Q_{ch}}^{\beta} / Bw$$

Where H_{ch} is the water level for the channel, α_{ch} is the kinematic wave coefficient for the channel, Q_{ch} is the discharge in the channel and Bw is the width of the river.

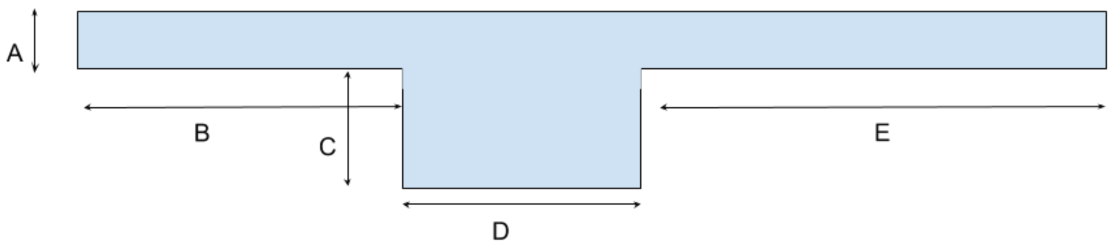
If the water level is above bankfull the water level on the floodplains is calculated as follows:

$$H_{fp} = \alpha_{fp} {Q_{fp}}^{\beta} / (Bw + P_{fp})$$

where H_{ch} is the water level on the floodplain, Q_{fp} is the discharge on the floodplain and P_{fp} is the wetted perimiter of the floodplain and α_{fp} is the kinematic wave coefficient for the floodplain,

The wetted perimiter of the channel, P_{ch}, is determined by:

$$P_{ch} = 2.0 H_{ch} + Bw$$



A = WaterLevelFP -> Depth of water in the floodplain
B+E = FloodPlainWidth -> maximum width of the floodplain
D = RiverWidth -> Width of the river
C = bankfulldepth -> Level at which the river starts to flood

The wetted perimiter of the floodplain is defined as follows:

$$N = max(0.0001, 1.0/(1.0 + exp(-c * H_{fp}) - 0.5) * 2.0$$
$$P_{fp} = NW_{fp}$$

This first equation defines the upper half of an S or sigmoid curve and will return values between 0.001 and 1.0. The c parameter defines the sharpness of the function, a high value of c will turn this into a step-wise function while a low value will make the function more smooth. The default value for c = 0.5. For example, with this default value a floodplain level of 1m will result in an N value of 0.25 and 2m will return 0.46. In the second equation this fraction is multiplied with the maximum floodplain width W_{fp}.

The α for the channel and floodplain are calculated as follows:

$$\alpha_{ch} = (n_{ch}/\sqrt{slope})^{\beta} P_{ch}^{(2.0/3.0)\beta}$$
$$\alpha_{fp} = (n_{fp}/\sqrt{slope})^{\beta} P_{fp}^{(2.0/3.0)\beta}$$

In which slope is the slope of the river bed and floodplain and n_{ch} and n_{fp} represent the manning's n for the channel and floodplain respectively.

A compound α_{total} is estimated by first calculating a compound n value n_{total}:

$$n_{total} = (P_{ch}/P_{total} n_{ch}^{3/2} + P_{fp}/P_{total} n_{fp}^{3/2})^{2/3}$$
$$\alpha_{total} = (n_{total}/\sqrt{slope})^{\beta} (P_{fp} + P_{ch})^{(2.0/3.0)\beta}$$

The α_{total} is used in the pcraster kinematic function to get the discharge for the next timestep.

The code is implemented in the updateRunoff attribute of the model class as follows:

```
self.Qbankfull = pow(self.bankFull/self.AlphaCh * self.Bw,1.0/self.Beta)
self.Qchannel = min(self.SurfaceRunoff,self.Qbankfull)
self.floodcells  = boolean(ifthenelse(self.WaterLevelCH > self.bankFull, boolean(1),␣
↪boolean(0)))
self.Qfloodplain = max(0.0,self.SurfaceRunoff - self.Qbankfull)

self.WaterLevelCH = self.AlphaCh * pow(self.Qchannel, self.Beta) / (self.Bw)
self.WaterLevelFP = ifthenelse(self.River,self.AlphaFP * pow(self.Qfloodplain, self.
↪Beta) / (self.Bw + self.Pfp),0.0)
self.WaterLevel = self.WaterLevelCH + self.WaterLevelFP

# Determine Qtot as a check
self.Qtot = pow(self.WaterLevelCH/self.AlphaCh * self.Bw,1.0/self.Beta) + pow(self.
↪WaterLevelFP/self.AlphaFP * (self.Pfp + self.Bw),1.0/self.Beta)
# wetted perimeter (m)
self.Pch = self.wetPerimiterCH(self.WaterLevelCH,self.Bw)
self.Pfp = ifthenelse(self.River,self.wetPerimiterFP(self.WaterLevelFP,self.
↪floodPlainWidth,sharpness=self.floodPlainDist),0.0)

# Alpha
self.WetPComb = self.Pch + self.Pfp
self.Ncombined = (self.Pch/self.WetPComb*self.N**1.5 + self.Pfp/self.WetPComb*self.
↪NFloodPlain**1.5)**(2./3.)
self.AlpTermFP = pow((self.NFloodPlain / (sqrt(self.SlopeDCL))), self.Beta)
self.AlpTermComb = pow((self.Ncombined / (sqrt(self.SlopeDCL))), self.Beta)
self.AlphaFP = self.AlpTermFP * pow(self.Pfp, self.AlpPow)
self.AlphaCh = self.AlpTerm * pow(self.Pch, self.AlpPow)
```

(continues on next page)

(continued from previous page)

```
self.Alpha = ifthenelse(self.River,self.AlpTermComb * pow(self.Pch + self.Pfp, self.
↪AlpPow),self.AlphaCh)
self.OldKinWaveVolume = self.KinWaveVolume
self.KinWaveVolume = (self.WaterLevelCH * self.Bw * self.DCL) + (self.WaterLevelFP *␣
↪(self.Pfp + self.Bw) * self.DCL)
```

Reservoirs

Simple reservoirs can be included within the kinematic wave routing by supplying a map with the locations of the reservoirs in which each reservoir has a unique id. Furthermore a set of lookuptables must be defined linking the reservoir-id's to reservoir characteristics:

- ResTargetFullFrac.tbl - Target fraction full (of max storage) for the reservoir: number between 0 and 1
- ResTargetMinFrac.tbl - Target minimum full fraction (of max storage). Number between 01 and 1 < ResTarget-FullFrac
- ResMaxVolume.tbl - Maximum reservoir storage (above which water is spilled) [m^3]
- ResDemand.tbl - Water demand on the reservoir (all combined) m^3/s
- ResMaxRelease.tbl - Maximum Q that can be released if below spillway [m^3/s]

By default the reservoirs are not included in the model. To include them put the following lines in the .ini file of the model.

```
[modelparameters]
# Add this if you want to model reservoirs
ReserVoirLocs=staticmaps/wflow_reservoirlocs.map,staticmap,0.0,0
ResTargetFullFrac=intbl/ResTargetFullFrac.tbl,tbl,0.8,0,staticmaps/wflow_reservoirlocs.
↪map
ResTargetMinFrac=intbl/ResTargetMinFrac.tbl,tbl,0.4,0,staticmaps/wflow_reservoirlocs.map
ResMaxVolume=intbl/ResMaxVolume.tbl,tbl,0.0,0,staticmaps/wflow_reservoirlocs.map
ResMaxRelease=intbl/ResMaxRelease.tbl,tbl,1.0,0,staticmaps/wflow_reservoirlocs.map
ResDemand=intbl/ResDemand.tbl,tblmonthlyclim,1.0,0,staticmaps/wflow_reservoirlocs.map
```

In the above example most values are fixed thought the year, only the demand is given per month of the year.

Configuration

The default name for the file is wflow_routing.ini.

Subcatchment flow

Normally the the kinematic wave is continuous throughout the model. By using the the SubCatchFlowOnly entry in the model section of the ini file all flow is at the subcatchment only and no flow is transferred from one subcatchment to another. This can be handy when connecting the result of the model to a water allocation model such as Ribasim.

Example:

```
[model]
SubCatchFlowOnly = 1
```

Forcing data

The model needs one set of forcing data: IW (water entering the model for each cell in mm). The name of the mapstack is can be defined in the ini file. By default it is inmaps/IW

See below for an example:

```
[inputmapstacks]
Inwater = /run_default/outmaps/IW
Inflow = /inmaps/IF

[run]
starttime = 1995-01-31 00:00:00
endtime = 1995-02-28 00:00:00
timestepsecs = 86400
reinit = 0

[model]
modeltype = routing
AnnualDischarge = 2290
Alpha = 120
WIMaxScale = 0.8
Tslice = 1
UpdMaxDist = 300000.0
reinit = 1
fewsrun = 0
OverWriteInit = 0
updating = 0
updateFile = no_set
sCatch = 0
intbl = intbl
timestepsecs = 86400
MaxUpdMult = 1.3
MinUpdMult = 0.7
UpFrac = 0.8
SubCatchFlowOnly = 0
wflow_subcatch = staticmaps/wflow_subcatch.map
wflow_dem = staticmaps/wflow_dem.map
wflow_ldd = staticmaps/wflow_ldd.map
wflow_river = staticmaps/wflow_river.map
wflow_riverlength = staticmaps/wflow_riverlength.map
wflow_riverlength_fact = staticmaps/wflow_riverlength_fact.map
wflow_gauges = staticmaps/wflow_gauges.map
wflow_inflow = staticmaps/wflow_inflow.map
wflow_riverwidth = staticmaps/wflow_riverwidth.map
wflow_floodplainwidth = staticmaps/wflow_floodplainwidth.map
wflow_bankfulldepth = staticmaps/wflow_bankfulldepth.map
wflow_landuse = staticmaps/wflow_landuse.map
wflow_soil = staticmaps/wflow_soil.map

[framework]
outputformat = 1
debug = 0
```

(continues on next page)

(continued from previous page)

```
netcdfinput = None
netcdfoutput = None
netcdfstaticoutput = None
netcdfstaticinput = None
EPSG = EPSG:4326

[layout]
sizeinmetres = 0

[outputmaps]
self.SurfaceRunoff = _run
self.Qfloodplain = _qfp
self.Qchannel = _qch
self.Qbankfull = _qbnk
self.WaterLevelFP = _levfp
self.WaterLevelCH = _levch
self.InwaterMM = _IW
self.floodcells = fcel
self.Qtot = QQQ
self.Pch = ch
self.Pfp = fp
self.Alpha = al
self.AlphaCh = alch
self.AlphaFP = alfp
self.Ncombined = nc
self.MassBalKinWave = wat

[outputcsv_0]
samplemap = None

[outputtss_0]
samplemap = None
```

A description of the implementation of the kinematic wave is given on the pcraster website at http://pcraster.geo.uu.nl/pcraster/4.0.2/doc/manual/op_kinematic.html

In addition to the settings in the ini file you need to give the model additional maps or lookuptables in the staticmaps or intbl directories:

Lookup tables

N.tbl
Manning's N for all no-river cells. Defaults to 0.072

N_River.tbl
Manning's N for the river cells. Defaults to 0.036

N_FloodPlain.tbl
Manning's N for the floodplain. A floodplain is always linked to a river cell. Defaults to 2* N of the river

ResDemand.tbl
Lookup table of demand (m^3/s) for reservoir locations.

ResMaxRelease.tbl
  Lookup table of maximum release capcity (m^3/s) for reservoir locations.

ResMaxVolume.tbl
  Lookup table of reservoi maximum volume (m^3) for reservoir locations.

ResTargetFullFrac
  Lookup table of the target maximum full fraction (0-1) (m^3/s) for reservoir locations.

ResTargetMinFrac
  Lookup table of the target minimum full fraction (0-1) (m^3/s) for reservoir locations.

As with all models the lookup tables can be replaced by a map with the same name (but with the .map extension) in the staticmaps directory.

staticmaps

wflow_subcatch.map
  Map of the subcatchment in the area. Usually shared with the hydrological model

wflow_dem.map
  The digital elevation model. Usually shared with the hydrological model

wflow_ldd.map
  The D8 local drainage network.

wflow_river.map
  Definition of the river cells.

wflow_riverlength.map
  Optional map that defines the actual legth of the river in each cell.

wflow_riverlength_fact.map
  Optional map that defines a multiplication factor for the river length in each cell.

wflow_gauges.map
  Map of river gauges that can be used in outputting timeseries

wflow_inflow.map
  Optional map of inflow points into the surface water. Limited testing.

wflow_riverwidth.map
  Optional map of the width of the river for each river cell.

wflow_floodplainwidth.map
  Optional map of the width of the floodplain for each river cell.

wflow_bankfulldepth.map
  Optional map of the level at which the river starts to flood and water will also be conducted over the floodplain.

wflow_floodplaindist.map
  Optional map that defines the the relation between bankfulldepth and the floodplaindepth. Default = 0.5

wflow_landuse.map
  Required map of landuse/land cover. This map is used in the lookup tables to relate parameters to landuse/landcover. Usually shared with the hydrological model

wflow_soil.map
  Required map of soil type. Usually shared with the hydrological model

initial conditions

The model needs the following files with initial conditions:

WaterLevelCH
Water level in the channel or the grid-cell water level for non-river cells.

WaterLevelFP
Water level in the floodplain

SurfaceRunoff
Discharge in each grid-cell

ReservoirVolume
Volume of each reservoir in m^3

The total waterlevel is obtained by adding the two water levels.

wflow_routing module documentation

1.5.10 The wflow_wave Model

Warning: The documentation of this model is incomplete

Introduction

An *experimental* implementation of the full dynamic wave equations has been implemented. The current implementation is fairly unstable and *very* slow.

However, in flat of tidal areas and areas that flood the dynamic wave can provide much better results. The plot below is from the Rio Mamore in Bolivia in the lower partes of the river with extensive wetlands that flood nearly each year.

Dependencies

This module is setup to be run in *an existing case and runid* of a wflow_sbm or wflow_hbv model. In order for wflow_wave to run they must have saved discharge and waterlevel for each timesteps. This output will be used as a forcing for the dynamic wave module. The wflow_wave module will also use the existing ldd and DEM

Configuration

It needs anumber of settings in the ini file. The default name for the file is wflow_wave.ini. it is also possible to insert this section in the wflow_sbm or wflow_hbv ini file and point to that file.

See below for an example:

```
[inputmapstacks]
# Name of the mapstack with discharge (output from the hydrological model)
Q = run
# Name of the mapstack with waterlevel (output from the hydrological model)
H = lev

[dynamicwave]

```

(continues on next page)

(continued from previous page)

```
# Number of timeslices per dynamic wave substep
TsliceDyn=100

# number of substeps for the dynamic wave with respect to the model timesteps
dynsubsteps=24

# map with level boundary points
wflow_hboun = staticmaps/wflow_outlet.map

# Optional river map for the dynamic wave that must be the same size or smaller as that␣
↪of the
# kinematic wave
wflow_dynriver = staticmaps/wflow_dynriver.map

# a fixed water level for each non-zero point in the wflow_hboun map
# level > 0.0 use that level
# level == 0.0 use supplied timeseries (see levelTss)
# level < 0.0 use upstream water level
fixedLevel = 3.0

# if this is set the program will try to keep the volume at the pits at
# a constant value
lowerflowbound = 1

# instead of a fixed level a tss file with levels for each timesteps and each
# non-zero value in the wflow_hboun map
#levelTss=intss/Hboun.tss

# If set to 1 the program will try to optimise the timestep
# Experimental, mintimestep is the smallest to be used
AdaptiveTimeStepping = 1
mintimestep =0.1
```

A description of the implementation of the dynamicwave is given on the pcraster website.

In addition to the settings in the ini file you need to give the model additional maps or lookuptables in the staticmaps or intbl directories:

- ChannelDepth.[map|tbl] - depth of the main channel in metres
- ChannelRoughness.[map|tbl] - 1/n n = manning roughness coefficient (default = 1/0.03)
- ChannelForm.[map|tbl] - form of the channel (default = 1.0)
- FloodplainWidth.[map|tbl] - width of the floodplain in metres (default = 0)

The following variables for the dynamicwave function are set as follows and are taken from the hydrological model run by default:

- ChannelBottomLevel - Taken from the dem (wflow_dem.map in the staticmaps dir)
- ChannelLength - Taken from the length in the kinematic wave (DCL.map from the outsum dir)
- ChannelBottomWidth - taken from the wflow_riverwidth map from the outsum dir

wflow_wave module documentation

1.5.11 The wflow_floodmap model

Introduction

The wflow_floodmap module can generate flood maps from output of a wflow_sbm|hbv| or wflow_routing model.

At the moment there are two approaches for flood mapping 1. wflow_floodmap.py – this is a regular wflow-type model, running at the same resolution as the wflow model used to establish flood maps. The benefit is that it produces dynamic flood maps. The down side is that the results are in the same resolution as the original model. The method is also not volume conservative as it only does a planar inundation and bound to lead to large overestimations in very flat areas.

2. wflow_flood.py (see Scripts folder). This is a postprocessor to results of a wflow model and transforms low-resolution model results into a high-resolution flood map using a (possibly much) higher resolution terrain dataset as input. We recommend to retrieve high resolution terrain from the Bare-Earth SRTM dataset by Bristol University. See https://data.bris.ac.uk/data/dataset/10tv0p32gizt01nh9edcjzd6wa

Method

PM

Configuration

PM

Description of the wflow_floodmap model

Description of the wflow_flood post processor

Definition of the wflow_flood post processor.

Performs a planar volume spreading on outputs of a wflow_sbm|hbv|routing model run. The module can be used to post-process model outputs into a flood map that has a (much) higher resolution than the model resolution.

The routine aggregates flooded water volumes occurring across all river pixels across a user-defined strahler order basin scale (typically a quite small subcatchment) and then spreads this volume over a high resolution terrain model. To ensure that the flood volume is not spread in a rice-field kind of way (first filling the lowest cell in the occurring subbasin), the terrain data is first normalised to a Height-Above-Nearest-Drain (HAND) map of the associated typically flooding rivers (to be provided by user through a catchment order) and flooding is estimated from this HAND map.

A sequential flood mapping is performed starting from the user defined Strahler order basin scale to the highest Strahler orders. A HAND map is derived for each river order starting from the lowest order. These maps are then used sequentially to spread flood volumes over the high resolution terrain model starting from the lowest catchment order provided by the user (using the corresponding HAND map) to the highest stream order. Backwater effects from flooding of higher orders catchments to lower order catchments are taken into account by taking the maximum flood level of both.

Preferrably a user should use from the outputs of a wflow_routing model because then the user can use the floodplain water level only (usually saved in a variable name levfp). If estimates from a HBV or SBM (or other wflow) model are used we recommend that the user also provides a “bank-full” water level in the command line arguments. If not provided, wflow_flood will also spread water volumes occuring within the banks of the river, probably leading to an overestimation of flooding.

The wflow_sbm|hbv model must have a saved mapstacks in NetCDF containing (over-bank) water levels The module selects the maximum occurring value in the map stacks provided (NetCDF) and spreads this out over a high resolution terrain dataset using the high resolution terrain, associated ldd and stream order map.

TODO:: enable selection of a time step.

Ini-file settings

The module uses an ini file and a number of command line arguments to run. The ini-file contains inputs that are typically the same across a number of runs with the module for a given study area (e.g. the used DEM, LDD, and some run parameters). For instance, a user can prepare flood maps from different flood events computed with one WFLOW model, using the same .ini file for each event.

The .ini file sections are treated below:

```
[HighResMaps]
dem_file =SRTM 90m merged/BEST90m_WGS_UTM42N.tif
ldd_file = SRTM 90m merged/LDD/ldd_SRTM0090m_WGS_UTM42N.map
stream_file = Processed DEMs/SRTM 90m merged/stream.map
[wflowResMaps]
riv_length_fact_file = floodhazardsimulations/Stepf_output/river_length_fact.map
riv_width_file = floodhazardsimulations/Stepf_output/wflow_floodplainwidth.map
ldd_wflow = floodhazardsimulations/Stepf_output/wflow_ldd.map
[file_settings]
latlon = 0
file_format = 0
```

The dem_file contains a file with the high-res terrain data. It MUST be in .tif format. This is because .tif files can contain projection information. At the moment the .tif file must have the same projection as the WFLOW model (can be WGS84, but also any local projection in meters), but we intend to also facilitate projections in the future.

The ldd_file contains the ldd, derived from the dem_file (PCRaster format)

The stream_file contains a stream order file (made with the PCRaster stream order file) derived from the LDD in ldd_file.

riv_length_fact_file and riv_width_file contain the dimensions of the channels within the WFLOW pixels (unit meters) and are therefore in the resolution of the WFLOW model. The riv_length_fact_file is used to derive a riv_length by multiplying the LDD length from cell to cell within the LDD network with the wflow_riverlength_fact.map map, typically located in the staticmaps folder of the used WFLOW model. The width map is also in meters, and should contain the flood plain width in case the wflow_routing model is used (typical name is wflow_floodplainwidth.map). If a HBV or SBM model is used, you should use the river width map instead (typical name wflow_riverwidth.map).

ldd_wflow is the ldd, derived at wflow resolution (typically wflow_ldd.map)

If latlon is 0, the cell-size is given in meters (the default)

If file_format is set to 0, the flood map is expected to be given in netCDF format (in the command line after -F) if set to 1, format is expected to be PCRaster format

```
[metadata_global]
source=WFLOW model XXX
institution=Deltares
title=fluvial flood hazard from a wflow model
references=http://www.deltares.nl/
Conventions=CF-1.6
project=Afhanistan multi-peril country wide risk assessment
```

In the metadata_global section the user can enter typical project details as metadata. These are used in the outcoming .tif file. We recommend to follow Climate and Forecast conventions for typical metadata entries (see http://cfconventions.org/). You can insert as many key/value pairs as you like. Some examples are given above.

```
[tiling]
x_tile=2000
y_tile=2000
x_overlap=500
y_overlap=500
```

When very large domains are processed, the complete rasters will not fit into memory. In this case, the routine will break the domain into several tiles and process these separately. The x_tile and y_tile parameters are used to set the tile size. If you are confident that the whole domain will fit into memory (typically when the size is smaller than about 5,000 x 5,000 rows and columns) then just enter a number larger than the total amount of rows and columns. The x_overlap and y_overlap parameters should be large enough to prevent edge effects at the edges of each tile where averaging subbasins are cut off from the edge. Slightly larger tiles (defined by the overlap) are therefore processed and the edges are consequently cut off after processing one tile to get a seamless product.

Some trial and error may be required to yield the right tile sizes and overlaps.

```
[inundation]
iterations=20
initial_level=32
```

The inundation section contains a number of settings for the flood fill algorithm. The number of iterations can be changed, we recommend to set it to 20 for an accurate results. The initial_level is the largest water level that can occur during flooding. Make sure it is set to a level (much) higher than anticipated to occur but not to a value close to infinity. If you set it orders too high, the solution will not converge to a reasonable estimate.

Command line arguments

When wflow_flood.py is run with the -h argument, you will receive the following feedback:

```
python wflow_flood.py -h
Usage: wflow_flood.py [options]

Options:
  -h, --help            show this help message and exit
  -q, --quiet           do not print status messages to stdout
  -i INIFILE, --ini=INIFILE
                        ini configuration file
  -f FLOOD_MAP, --flood_map=FLOOD_MAP
                        Flood map file (NetCDF point time series file
  -v FLOOD_VARIABLE, --flood_variable=FLOOD_VARIABLE
                        variable name of flood water level
  -b BANKFULL_MAP, --bankfull_map=BANKFULL_MAP
                        Map containing bank full level (is subtracted from
                        flood map, in NetCDF)
  -c CATCHMENT_STRAHLER, --catchment=CATCHMENT_STRAHLER
                        Smallest Strahler order threshold over which flooding
                        may occur
  -m MAX_CATCHMENT_STRAHLER, --max_catchment=MAX_CATCHMENT_STRAHLER
                        Largest Strahler order over which flooding may occur
```

(continues on next page)

(continued from previous page)

```
  -d DEST_PATH, --destination=DEST_PATH
                        Destination path
  -H HAND_FILE_PREFIX, --hand_file=HAND_FILE_PREFIX
                        optional HAND file prefix of already generated HAND maps
                        for different Strahler orders
  -n neg_HAND, --negHAND=0
            optional functionality to allow HAND maps to become
            negative (if set to 1, default=0)
```

Further explanation:

-i	the .ini file described in the previous section
-f	The NetCDF output time series or a GeoTIFF containing the flood event to be downscaled. In case of NetCDF, this is a typical NetCDF output file from a WFLOW model. Alternatively, you can provide a GeoTIFF file that contains the exact flood depths for a given time step user defined statistic (for example the maximum value across all time steps)
-v	Variable within the aforementioned file that contains the depth within the flood plain (typical levfp)
-b	Similar file as -f but providing the bank full water level. Can e provided in case you know that a certain water depth is blocked, or remains within banks. In cae a NetCDF is provided, the maximum values are used, alternatively, you can provide a GeoTIFF.
-c	starting point of catchment strahler order over which flood volumes are averaged, before spreading. The Height-Above-Nearest-Drain maps are derived from this Strahler order on (until max Strahler order). NB: This is the strahler order of the high resolution stream order map
-m	maximum strahler order over which flooding may occur (default value is the highest order in high res stream map)
-d	path where the file is stored
-H	HAND file prefix. As interim product, the module produces HAND files. This is a very time consuming process and therefore the user can also supply previously generated HAND files here (GeoTIFF format) The names of the HAND files should be constructed as follows: hand_prefix_{:02d}.format{hand_strahler}, so for example hand_prefix_03 for HAND map with minimum Strahler order 3. (in this case -H hand_prefix should be given) Maps should be made for Strahler orders from -c to -m (or maximum strahler order in the stream map)
-n	allow for negative HAND maps - if this option is set to 1, the user allows the HAND maps to become negative. This can be useful when there are natural embankments which result in a lower elevation than the river bed. However, this option leads to artifacts when the used SRTM is not corrected for elevation and when the river shapefile is not entirely correct (for example if the burned in river is following an old meander. Therefore the user must be very confident about

the used data sources (river shape file and digital elevation model corrected for vegetation) when this option is set to 1!

Outputs

wflow_flood produces the following outputs:

Table: outputs of the wflow_flood module

hand_contour_inun.log	log file of the module, contains info and error messages
inun_<-f>_catch_<-c>.tif	resulting inundation map (GeoTIFF)
<dem_file>_hand_strahler_<-c>.tif	HAND file based upon strahler order given with -c (only without -H

Questions can be directed to hessel.winsemius@deltares.nl

$Id: wflow_flood.py $ $Date: 2016-04-07 12:05:38 +0200 (Thu, 7 Apr 2016) $ $Author: winsemi $ $Revision: $ $HeadURL: $ $Keywords: $

1.5.12 The wflow_sediment Model

Warning: Experimental version

Introduction

The processes and fate of many particles and pollutants impacting water quality at the catchment level are intricately linked to the processes governing sediment dynamics. Both nutrients such as phosphorus, carbon or other pollutants such as metals are influenced by sediment properties in processes such as mobilization, flocculation or deposition. To better assert and model water quality in inland systems, a better comprehension and modelling of sediment sources and fate in the river is needed at a spatial and time scale relevant to such issues.

The wflow_sediment model was developed to answer such issues. It is a distributed physics-based model, based on the distributed hydrologic wflow_sbm model. It is able to simulate both land and in-stream processes, and relies on available global datasets, parameter estimation and small calibration effort.

In order to model the exports of terrestrial sediment to the coast through the Land Ocean Aquatic Continuum or LOAC (inland waters network such as streams, lakes…), two different modelling parts were considered (see Figure below). The first part is the modelling and estimation of soil loss and sediment yield to the river system by land erosion, called the soil loss model. The second part is the transport and processes of the sediment in the river system, called the river model. The two models together constitute the wflow_sediment model.

Method

The wflow_sediment model was developed using the same framework and tool as the wflow_sbm hydrologic model. It uses the results from the hydrology to then deduce soil erosion and delivery to the river system, in what is called the soil loss part of the model. It finally follows the fate and transport of the incoming sediments in the stream network, in what is called the river part of the model. To keep the consistency with wflow_sbm, the model is also developed in Python using PCRaster functions and should use the same datasets.

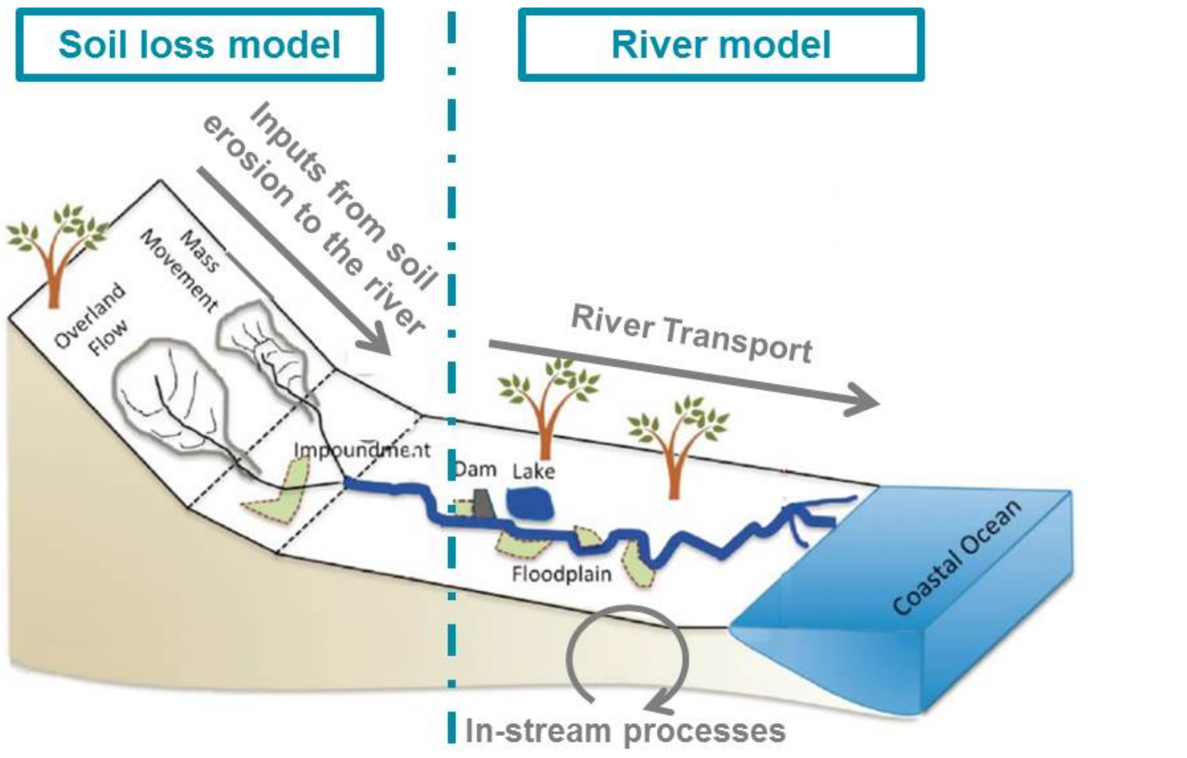


Fig. 14: Overview of the concepts of the wflow_sediment model.

Soil Loss part

The first process to consider in sediment dynamics is the generation of sediments by land erosion. The main processes behind soil loss are rainfall erosion and overland flow erosion. In order to model such processes at a fine time and space scale, physics-based models such as ANSWERS and EUROSEM were chosen here.

Rainfall erosion

In wflow_sediment, rainfall erosion can both be modelled using EUROSEM or ANSWERS equation. The main difference between the models is that EUROSEM uses a more physics-based approach using the kinetic energy of the rain drops impacting the soil (Morgan et al, 1998), while ANSWERS is more empirical and uses parameters from the USLE model (Beasley et al, 1991).

In EUROSEM, rainfall erosion is modelled according to rainfall intensity and its kinetic energy while it reaches the soil according to equations developed by Brandt (1990). As the intensity of the rain kinetic energy depends on the length of the fall, rainfall intercepted by vegetation will then be reduced compared to direct throughfall. The kinetic energy of direct throughfall is estimated by (Morgan et al, 1998):

$$KE_{direct} = 8.95 + 8.44 * log_{10} R_i$$

where KE_{direct} is kinetic energy of direct throughfall (J m^{-2} mm^{-1}) and R_i is rainfall intensity (mm h^{-1}). If the rainfall is intercepted by vegetation and falls as leaf drainage, its kinetic energy is then reduced according to (Brandt, 1990):

$$KE_{leaf} = 15.8 * H_p^{0.5} - 5.87$$

where KE_{leaf} is kinetic energy of leaf drainage (J m^{-2} mm^{-1}) and H_p is the effective canopy height (half of plant height in m). In the global version of wflow_sediment, canopy height can be derived from the global map from Simard & al. (2011) or user input depending on the land use.

Kinetic energies from both direct throughfall and leaf drainage are then multiplied by the respective depths of direct throughfall and leaf drainage (mm) and added to get the total rainfall kinetic energy KE. The soil detached by rainfall D_R (g m^{-2}) is then:

$$D_R = k * KE * e^{-\varphi h}$$

where k is an index of the detachability of the soil (g J^{-1}), KE is the total rainfall kinetic energy (J m^{-2}), h is the surface runoff depth on the soil (m) and φ is an exponent varying between 0.9 and 3.1 used to reduce rainfall impact if the soil is already covered by water. As a simplification, Torri (1987) has shown that a value of 2.0 for φ is representative enough for a wide range of soil conditions. The detachability of the soil k depends on the soil texture (proportion of clay, silt and sand content) and correponding values are defined in EUROSEM user guide (Morgan et al, 1998). As a simplification, in wflow_sediment, the mean value of the detachability shown in the table below are used. Soil texture is derived from the topsoil clay and silt content from SoilGrids (Hengl et al, 2017).

Table: Mean detachability of soil depending on its texture (Morgan et al, 1998).

Texture (USDA system)	Mean detachability k (g/J)
Clay	2.0
Clay Loam	1.7
Silt	1.2
Silt Loam	1.5
Loam	2.0
Sandy Loam	2.6
Loamy Sand	3.0
Fine Sand	3.5
Sand	1.9

Rainfall erosion is handled differently in ANSWERS. There, the impacts of vegetation and soil properties are handled through the USLE coefficients in the equation (Beasley et al, 1991):

$$D_R = 0.108 * C_{USLE} * K_{USLE} * A_i * R_i^2$$

where D_R is the soil detachment by rainfall (here in kg min^{-1}), C_{USLE} is the soil cover-management factor from the USLE equation, K_{USLE} is the soil erodibility factor from the USLE equation, A_i is the area of the cell (m ^{2}) and R_i is the rainfall intensity (here in mm min^{-1}). In wflow_sediment, there are several methods available to estimate the C and K factors from the USLE. They can come from user input maps, for example maps resulting from Panagos & al.'s recent studies for Europe (Panagos et al, 2015) (Ballabio et al, 2016). To get an estimate of the C factor globally, the other method is to estimate C values for the different land use type in GlobCover. These values, summed up in the table below, come from a literature study including Panagos & al.'s review (2015), Gericke & al. (2015), Mansoor & al. (2013), Chadli & al. (2016), de Vente & al. (2009), Borrelli & al. (2014), Yang & al. (2003) and Bosco & al. (2015).

The other methods to estimate the USLE K factor are to use either topsoil composition or topsoil geometric mean diameter. K estimation from topsoil composition is estimated with the equation developed in the EPIC model (Williams et al, 1983):

$$K_{USLE} = \left\{0.2 + 0.3exp\left[-0.0256SAN\frac{(1-SIL)}{100}\right]\right\}\left(\frac{SIL}{CLA+SIL}\right)^{0.3}$$
$$*\left(1-\frac{0.25OC}{OC+exp(3.72-2.95OC)}\right)*\left(1-\frac{0.75SN}{SN+exp(-5.51+22.9SN)}\right)$$

where CLA, SIL, SAN are respectively the clay, silt and sand fractions of the topsoil (%) from SoilGrids, OC is the topsoil organic carbon content (%) from SoilGrids and SN is $1 - SAN/100$. The K factor can also be estimated from the soil mean geometric diameter using the formulation from the RUSLE guide by Renard & al. (1997):

$$K_{USLE} = 0.0034 + 0.0405exp\left[-\frac{1}{2}\left(\frac{log_{10}(D_g) + 1.659}{0.7101}\right)^2\right]$$

where D_g is the soil geometric mean diameter (mm) estimated from topsoil clay, silt, sand fraction.

Table: Estimation of USLE C factor per Globcover land use type

GlobCover Value	Globcover label	C_{USLE}
11	Post-flooding or irrigated croplands (or aquatic)	0.2
14	Rainfed croplands	0.35
20	Mosaic cropland (50-70%) / vegetation (grassland/shrubland/forest) (20-50%)	0.27
30	Mosaic vegetation (grassland/shrubland/forest) (50-70%) / cropland (20-50%)	0.25
40	Closed to open (>15%) broadleaved evergreen or semi-deciduous forest (>5m)	0.0065
50	Closed (>40%) broadleaved deciduous forest (>5m)	0.001
60	Open (15-40%) broadleaved deciduous forest/woodland (>5m)	0.01
70	Closed (>40%) needleleaved evergreen forest (>5m)	0.001
90	Open (15-40%) needleleaved deciduous or evergreen forest (>5m)	0.01
100	Closed to open (>15%) mixed broadleaved and needleleaved forest (>5m)	0.02
110	Mosaic forest or shrubland (50-70%) / grassland (20-50%)	0.015
120	Mosaic grassland (50-70%) / forest or shrubland (20-50%)	0.03
130	Closed to open (>15%) (broadleaved or needleleaved, evergreen or deciduous) shrubland (<5m)	0.035
140	Closed to open (>15%) herbaceous vegetation (grassland, savannas or lichens/mosses)	0.05
150	Sparse (<15%) vegetation	0.35
160	Closed to open (>15%) broadleaved forest regularly flooded (semi-permanently or temporarily) - Fresh or brackish water	0.001
170	Closed (>40%) broadleaved forest or shrubland permanently flooded - Saline or brackish water	0.0005
180	Closed to open (>15%) grassland or woody vegetation on regularly flooded or waterlogged soil - Fresh, brackish or saline water	0.04
190	Artificial surfaces and associated areas (Urban areas >50%)	0.0
200	Bare areas	0.0
210	Water bodies	0.0
220	Permanent snow and ice	0.0
230	No data (burnt areas, clouds,…)	0.0

Overland flow erosion

Overland flow (or surface runoff) erosion is induced by the strength of the shear stress of the surface water on the soil. As in rainfall erosion, the effect of the flow shear stress can be reduced by the soil vegetation or by the soil properties. In wflow_sediment, soil detachment by overland flow is modelled as in ANSWERS with (Beasley et al, 1991):

$$D_F = 0.90 * C_{USLE} * K_{USLE} * A_i * S * q$$

where D_F is soil detachment by flow (kg min^{-1}), C_{USLE} and K_{USLE} are the USLE cover and soil erodibility factors, A_i is the cell area (m^2), S is the slope gradient and q is the overland flow rate per unit width (m^2 min^{-1}). The USLE C and K factors can be estimated with the same methods as for rainfall erosion and here the slope gradient is obtained from the sinus rather than the tangent of the slope angle.

Delivery to the river system

Once the amount of soil detached by both rainfall and overland flow has been estimated, it has then to be routed and delivered to the river network. Inland routing in sediment models is usually done by comparing the amount of detached sediment with the transport capacity of the flow, which is the maximum amount of sediment than the flow can carry downslope. There are several existing formulas available in the literature. For a wide range of slope and for overland flow, the Govers equation (1990) seems the most appropriate choice (Hessel et al, 2007). However, as the wflow_sediment model was developed to be linked to water quality issues, the Yalin transport equation was chosen as it can handle particle differentiation (Govers equation can still be used if wflow_sediment is used to only model inland processes with no particle differentiation). For land cells, wflow_sediment assumes that erosion can mobilize 5 classes of sediment:

- Clay (mean diameter of 2 μ m)
- Silt (mean diameter of 10 μ m)
- Sand (mean diameter of 200 μ m)
- Small aggregates (mean diameter of 30 μ m)
- Large aggregates (mean diameter of 500 μ m).

To deduce the amount of small and large aggregates from topsoil clay, silt and sand contents, the following equations from the SWAT model are used (Neitsch et al, 2011):

$$PSA = SAN * (1 - CLA)^{2.4}$$

$$PSI = 0.13SIL$$

$$PCL = 0.20CLA$$

$$SAG = 2.0CLAforCLA < 0.25$$

$$SAG = 0.28(CLA - 0.25) + 0.5for0.25 \leq CLA \leq 0.5$$

$$SAG = 0.57forCLA > 0.5$$

$$LAG = 1 - PSA - PSI - PCL - SAG$$

where CLA, SIL and SAN are the primary clay, silt, sand fractions of the topsoil and PCL, PSI, PSA, SAG and LAG are the clay, silt, sand, small and large aggregates fractions of the detached sediment respectively. The transport capacity of the flow using Yalin's equation with particle differentiation, developed by Foster (1982), is:

$$TC_i = (P_e)_i * (S_g)_i * \rho_w * g * d_i * V_*$$

where TC_i is the transport capacity of the flow for the particle class i, $(P_e)_i$ is the effective number of particles of class i, $(S_g)_i$ is the specific gravity for the particle class i (kg m^{-3}), ρ_w is the mass density of the fluid (kg m^{-3}), g is the acceleration due to gravity (m s^{-2}), d_i is the diameter of the particle of class i (m) and $V_* = (g * R * S)^{0.5}$ is the shear velocity of the flow (m s^{-1}) with S the slope gradient and R the hydraulic radius of the flow (m). The detached sediment are then routed downslope until the river network using the accucapacityflux, accupacitystate functions from the PCRaster Python framework depending on the transport capacity from Yalin.

Finally, the different processes happening for a land cell in the soil loss part of wflow_sediment are summarized in the figure below:

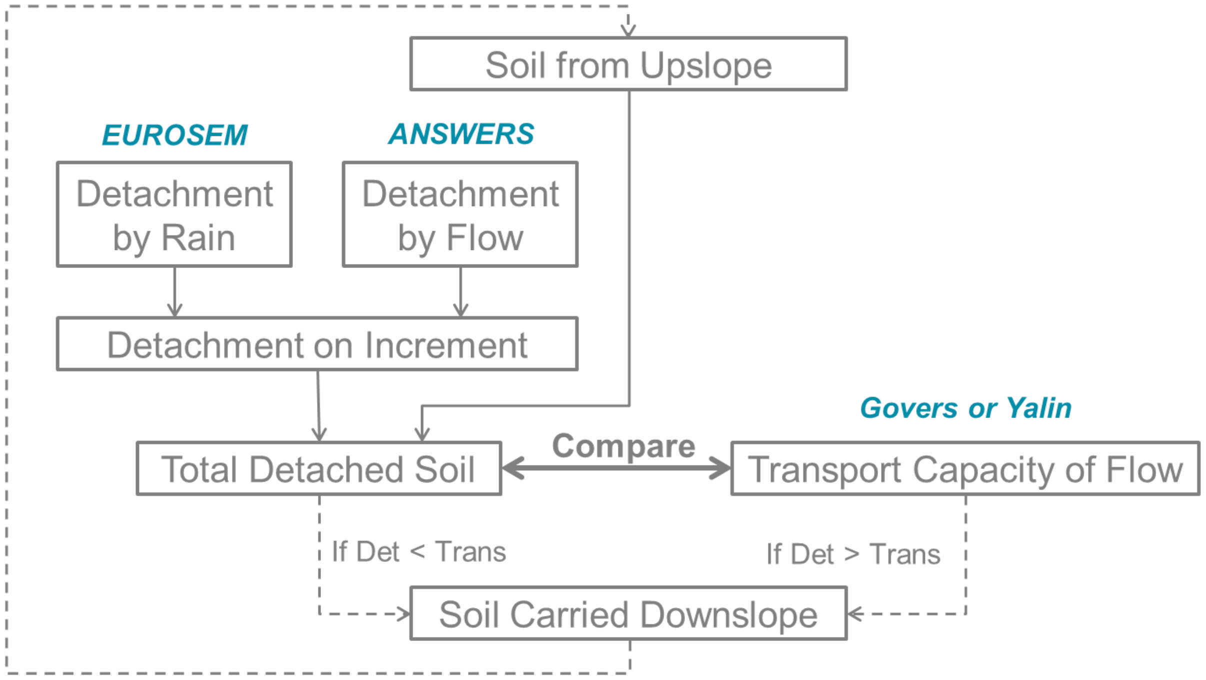


Fig. 15: Overview of the different processes for a land cell in wflow_sediment (adapted from Kinnell et al, 2010).

River part

As wflow_sediment was developed for applications across Europe, it must be able to simulate sediment dynamics both for small and large catchments. Which is why, for large catchments, it needed to model more precisely processes happening in the stream network. Thus once sediments coming from land erosion reach a river cell in the model, processes and equations change. There are not so many models available to model in-stream sediment dynamics with only hydrology. In the end, the more physics-based approach of the SWAT model was chosen as it requires little or no calibration and it can separate both the suspended from the bed load (Neitsch et al, 2011). As in SWAT, in the river part of wflow_sediment, 5 particles class are modelled: clay, silt, sand, small and large aggregates and gravel. Small and large aggregates are assumed to only come from land erosion, gravel only from river erosion, while clay, silt and sand can both come from either land or river erosion. In the river, suspended sediment load is assumed to be the sum of clay and silt and the bed sediment load is assumed to be composed of sand, gravel, small and large aggregates.

Sediment inputs in a river cell

The first part of the river model assesses how much detached sediment are in the river cell at the beginning of the timestep t. Sources of detached sediment are sediments coming from land erosion, estimated with the soil loss part of wflow_sediment model, the sediment coming from upstream river cells and the detached sediment that were left in the cell at the end of the previous timestep (t-1):

$$(sed_{in})_t = (sed_{land})_t + upstream\left[(sed_{out})_{t-1}\right] + (sed_{riv})_{t-1}$$

Sediment coming from upstream river cells is estimated using the PCRaster upstream function and the local drainage direction map to spot the upstream river cells.

River transport and erosion

Once the amount of sediment inputs at the beginning of the timestep is known, the model then estimates transport, and river erosion if there is a deficit of sediments. Transport in the river system is estimated via a transport capacity formula. There are several transport capacity formulas available in wflow_sediment, some requiring calibration and some not. Choosing a transport capacity equation depends on the river characteristics (some equation are more suited for narrow or wider rivers), and on the reliability of the required river parameters (such as slope, width or mean particle diameter of the river channel). Available transport capacity equations are:

- **Simplified Bagnold**: originally more valid for intermediate to large rivers, this simplified version of the Bagnold equation relates sediment transport to flow velocity with two simple calibration parameters (Neitsch et al, 2011):

$$C_{max} = c_{sp} * \left(\frac{prf * Q}{h * W}\right)^{sp_{exp}}$$

  where C_{max} is the sediment concentration (ton m^{-3} or kg/L), Q is the surface runoff in the river cell (m:math:^{3}/s), h is the river water level (m), W is the river width (m) and c_{sp}, prf and sp_{exp} are calibration parameters. The prf coefficient is usually used to deduce the peak velocity of the flow, but for simplification in wflow_sediment, the equation was simplified to only get two parameters to calibrate: sp_{exp} and $c_{Bagnold} = c_{sp} * prf^{sp_{exp}}$. The coefficient sp_{exp} usually varies between 1 and 2 while prf and c_{sp} have a wider range of variation. The table below summarizes ranges and values of the three Bagnold coefficients used by other studies:

Table 2: Range of the simplified Bagnold coefficients (and calibrated value)

Study	River	prf range	c_{sp} range	sp_{exp} range
Vigiak 2015	Danube	0.5-2 (/)	0.0001-0.01 (0.003-0.006)	1-2 (1.4)
Vigiak 2017	Danube	/	0.0001-0.01 (0.0015)	1-2 (1.4)
Abbaspour 2007	Thur (CH)	0.2-0.25 (/)	0.001-0.002 (/)	1.35-1.47 (/)
Oeurng 2011	Save (FR)	0-2 (0.58)	0.0001-0.01 (0.01)	1-2 (2)

- **Engelund and Hansen**: not present in SWAT but used in many models such as Delft3D-WAQ, Engelund and Hansen calculates the total sediment load as (Engelund and Hansen, 1967):

$$C_w = 0.05 \left(\frac{\rho_s}{\rho_s - \rho}\right) \left(\frac{u * S}{\sqrt{\left(\frac{\rho_s}{\rho_s - \rho}\right) * g * D_{50}}}\right) \theta^{1/2}$$

  where C_w is the sediment concentration by weight, ρ and ρ_s are the fluid and sediment density (here equal to 1000 and 2650 g m^{-3}), u is the water mean velocity (m/s), S is the river slope, g is the acceleration due to gravity, D_{50} is the river mean diameter (m) and θ is the Shields parameter.

- **Kodatie**: Kodatie (1999) developped the power relationships from Posada (1995) using field data and linear optimization so that they would be applicable for a wider range of riverbed sediment size. The resulting equation, for a rectangular channel, is (Neitsch et al, 2011):

$$C_{max} = \left(\frac{a * u^b * h^c * S^d}{V_{in}}\right) * W$$

  where V_{in} in the volume of water entering the river cell during the timestep (m:math:^{3}) and a, b, c and d are coefficients depending on the riverbed sediment size. Values of these coefficients are summarized in the table below:

Table 3: Range of the simplified Bagnold coefficients (and calibrated value)

River sediment diameter	a	b	c	d
$D_{50} \leq 0.05$mm	281.4	2.622	0.182	0
$0.05 < D_{50} \leq 0.25$mm	2 829.6	3.646	0.406	0.412
$0.25 < D_{50} \leq 2$mm	2 123.4	3.300	0.468	0.613
$D_{50} > 2$mm	431 884.8	1.000	1.000	2.000

- **Yang**: Yang (1996) developed a set of two equations giving transport of sediments for sand-bed or gravel-bed rivers. The sand equation ($D_{50} < 2mm$) is:

$$log\left(C_{ppm}\right) = 5.435 - 0.286 log\frac{\omega_{s,50}D_{50}}{\nu} - 0.457 log\frac{u_*}{\omega_{s,50}} + \left(1.799 - 0.409 log\frac{\omega_{s,50}D_{50}}{\nu} - 0.314 log\frac{u_*}{\omega_{s,50}}\right) log\left(\frac{uS}{\omega_{s,50}} - \frac{u_{cr}S}{\omega_{s,50}}\right)$$

  And the gravel equation ($2 \leq D_{50} < 10mm$) is:

$$log\left(C_{ppm}\right) = 6.681 - 0.633 log\frac{\omega_{s,50}D_{50}}{\nu} - 4.816 log\frac{u_*}{\omega_{s,50}} + \left(2.784 - 0.305 log\frac{\omega_{s,50}D_{50}}{\nu} - 0.282 log\frac{u_*}{\omega_{s,50}}\right) log\left(\frac{uS}{\omega_{s,50}} - \frac{u_{cr}S}{\omega_{s,50}}\right)$$

  where C_{ppm} is sediment concentration in parts per million by weight, $\omega_{s,50}$ is the settling velocity of a particle with the median riverbed diameter estimated with Stokes (m/s), ν is the kinematic viscosity of the fluid (m^2/s), u_* is the shear velocity ($\sqrt{gR_HS}$ in m/s with R_H the hydraulic radius of the river) and u_{cr} is the critical velocity (m/s, equation can be found in Hessel, 2007).

- **Molinas and Wu**: The Molinas and Wu (2001) transport equation was developed for large sand-bed rivers based on the universal stream power ψ. The corresponding equation is (Neitsch et al, 2011):

$$C_w = \frac{1430 * (0.86 + \sqrt{\psi}) * \psi^{1.5}}{0.016 + \psi} * 10^{-6}$$

  where ψ is the universal stream power given by:

$$\psi = \frac{\psi^3}{\left(\frac{\rho_s}{\rho} - 1\right) * g * h * \omega_{s,50} * \left[log_{10}\left(\frac{h}{D_{50}}\right)\right]^2}$$

Once the maximum concentration C_{max} is established with one of the above transport formula, the model then determines if there is erosion of the river bed and bank. In order to do that, the difference sed_{ex} between the maximum amount of sediment estimated with transport ($sed_{max} = C_{max} * V_{in}$) and the sediment inputs to the river cell (sed_{in} calculated above) is calculated. If too much sediment is coming in and sed_{ex} is negative, then there is no river bed and bank erosion. And if the river has not reach its maximum transport capacity, then erosion of the river happens.

First, the sediments stored in the cell from deposition in previous timesteps sed_{stor} are eroded from clay to gravel. If this amount is not enough to cover sed_{ex}, then erosion of the local river bed and bank material starts.

Instead of just setting river erosion amount to just cover the remaining difference sed_{exeff} between sed_{ex} and sed_{stor}, actual erosion potential is adjusted using river characteristics and is separated between the bed and bank of the river using the physics-based approach of Knight (1984).

The bed and bank of the river are supposed to only be able to erode a maximum amount of their material $E_{R,bed}$ for the bed and $E_{R,bank}$ for the river bank. For a rectangular channel, assuming it is meandering and thus only one bank is prone to erosion, they are calculated from the equations(Neitsch et al, 2011):

$$E_{R,bed} = k_{d,bed} * (\tau_{e,bed} - \tau_{cr,bed}) * 10^{-6} * L * W * \rho_{b,bed} * \Delta t$$

$$E_{R,bank} = k_{d,bank} * (\tau_{e,bank} - \tau_{cr,bank}) * 10^{-6} * L * h * \rho_{b,bank} * \Delta t$$

where E_R is the potential bed/bank erosion rates (tons), k_d is the erodibility of the bed/bank material (cm^3 N^{-1} s^{-1}), τ_e is the effective shear stress from the flow on the bed/bank (N/m^2), τ_{cr} is the critical shear stress for erosion to happen (N/m^2), L, W and h are the channel length, width and water height (m), ρ_b is the bulk density of the bed/bank of the river (g/cm^3) and Δt is the model timestep (s).

In wflow_sediment, the erodibility of the bed/bank are approximated using the formula from Hanson and Simon (2001):

$$k_d = 0.2 * \tau_{cr}^{-0.5}$$

Normally erodibilities are evaluated using jet test in the field and there are several reviews and some adjustments possible to this equation (Simon et al, 2011). However, to avoid too heavy calibration and for the scale considered, this equation is supposed to be efficient enough. The critical shear stress τ_{cr} is evaluated differently for the bed and bank. For the bed, the most common formula from Shields initiation of movement is used. For the bank, a more recent approach from Julian and Torres (2006) is used :

$$\tau_{cr,bank} = (0.1 + 0.1779 * SC + 0.0028 * SC^2 - 2.34 * 10^{-5} * SC^3) * C_{ch}$$

where SC is the percent clay and silt content of the river bank and C_{ch} is a coefficient taking into account the positive impact of vegetation on erosion reduction. This coefficient is then dependent on the land use and classical values are shown in the table below. These values where then adapted for use with the GlobCover land use map. Percent of clay and silt (along with sand and gravel) for the channel is estimated from the river median particle diameter assuming the same values as SWAT shown in the table below. Median particle diameter is here estimated depending on the Strahler river order. The higher the order, the smaller the diameter is. As the median diameter is only used in wflow_sediment for the estimation of the river bed/bank sediment composition, this supposition should be enough. Actual refined data or calibration may however be needed if the median diameter is also required for the transport formula. In a similar way, the bulk densities of river bed and bank are also just assumed to be of respectively 1.5 and 1.4 g/cm^3.

Table 4: Classical values of the channel cover vegetation coefficent (Julian and Torres, 2006)

Bank vegetation	C_{ch}
None	1.00
Grassy	1.97
Sparse trees	5.40
Dense trees	19.20

Table : Composition of the river bed/bank depending on the median diameter (Neitsch et al, 2011)

Sediment Fraction	d_{50} (μm)			
	≤ 5	5 to 50	50 to 2000	>2000
Sand	0.15	0.15	0.65	0.15
Silt	0.15	0.65	0.15	0.15
Clay	0.65	0.15	0.15	0.05
Gravel	0.05	0.05	0.05	0.65

Then, the repartition of the flow shear stress is refined into the effective shear stress and the bed and bank of the river using the equations developed by Knight (1984) for a rectangular channel:

$$\tau_{e,bed} = \rho g R_H S * \left(1 - \frac{SF_{bank}}{100}\right) * \left(1 + \frac{2h}{W}\right)$$

$$\tau_{e,bank} = \rho g R_H S * (SF_{bank}) * \left(1 + \frac{W}{2h}\right)$$

where ρg is the fluid specific weight (9800 N/m^3 for water), R_H is the hydraulic radius of the channel (m), h and W are the water level and river width (m). SF_{bank} is the proportion of shear stress acting on the bank (%) and is estimated from (Knight, 1984):

$$SF_{bank} = exp\left(-3.230 * log_{10}\left(\frac{W}{h} + 3\right) + 6.146\right)$$

Finally the relative erosion potential of the bank and bed of the river is calculated by:

$$RTE_{bed} = \frac{E_{R,bed}}{E_{R,bed} + E_{R,bank}}$$

$$RTE_{bank} = 1 - RTE_{bed}$$

And the final actual eroded amount for the bed and bank is the maximum between $RTE * sed_{exeff}$ and the erosion potential E_R. Total eroded amount of sediment sed_{erod} is then the sum of the eroded sediment coming from the storage of previously deposited sediment and the river bed/bank erosion.

River deposition

As sediments have a higher density than water, moving sediments in water can be deposited in the river bed. The deposition process depends on the mass of the sediment, but also on flow characteristics such as velocity. In wflow_sediment, as in SWAT, deposition is modelled with Einstein's equation (Neitsch et al, 2011):

$$P_{dep} = \left(1 - \frac{1}{e^x}\right) * 100$$

where P_{dep} is the percentage of sediments that is deposited on the river bed and x is a parameter calculated with:

$$x = \frac{1.055 * L * \omega_s}{u * h}$$

where L and h are channel length and water height (m), ω_s is the particle settling velocity calculated with Stokes formula (m/s) and u is the mean flow velocity (m/s). The calculated percentage is then subtracted from the amount of sediment input and eroded river sediment for each particle size class ($sed_{dep} = P_{dep}/100 * (sed_{in} + sed_{erod})$). Resulting deposited sediment are then stored in the river bed and can be re-mobilized in future time steps by erosion.

Mass balance and sediment concentration

Finally after estimating inputs, deposition and erosion with the transport capacity of the flow, the amount of sediment actually leaving the river cell to go downstream is estimated using:

$$sed_{out} = (sed_{in} + sed_{erod} - sed_{dep}) * \frac{V_{out}}{V}$$

where sed_{out} is the amount of sediment leaving the river cell (tons), sed_{in} is the amount of sediment coming into the river cell (storage from previous timestep, land erosion and sediment flux from upstream river cells in tons), sed_{erod} is the amount of sediment coming from river erosion (tons), sed_{dep} is the amount of deposited sediments (tons), V_{out} is the volume of water leaving the river cell (surface runoff Q times timestep Δt in m^3) and V is the total volume of water in the river cell (V_{out} plus storage $h * W * L$ in m^3).

A mass balance is then used to calculate the amount of sediment remaining in the cell at the end of the timestep $(sed_{riv})_t$:

$$\begin{aligned}(sed_{riv})_t = (sed_{riv})_{t-1} + (sed_{land})_t + upstream\left[(sed_{out})_{t-1}\right] \\ + (sed_{erod})_t - (sed_{dep})_t - (sed_{out})_t\end{aligned}$$

Finally, the different processes happening for a land cell in the river part of wflow_sediment are summarized in the figure below:

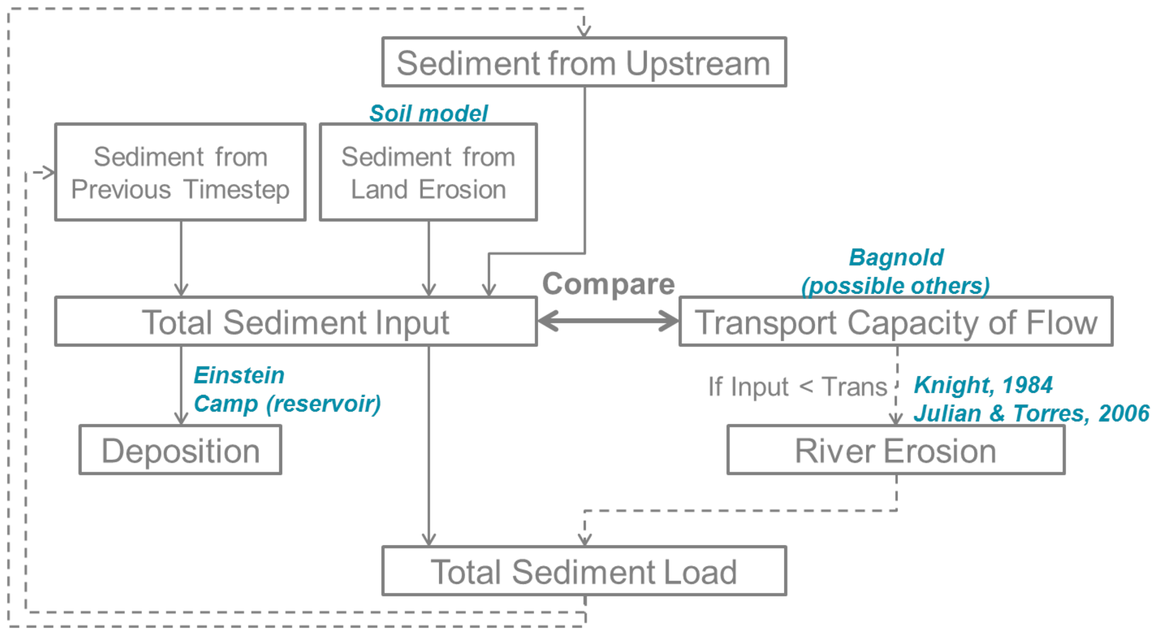


Fig. 16: Overview of the different processes for a river cell in wflow_sediment.

Lake modelling

Apart from land and river, the hydrologic wflow_sbm model also handles lakes and reservoirs modelling. In wflow_sbm, lakes and large reservoirs are modelled using a 1D bucket model at the cell corresponding to the outlet. For the other cells belonging to the lake/reservoir which are not the outlet, processes such as precipitation and evaporation are filtered out and shifted to the outlet cell. wflow_sediment then handles the lakes in the same way. If a cell belongs to a lake/reservoir and is not the outlet then the model assumes that no erosion/deposition of sediments is happening and the sediments are only all transported to the lake/reservoir outlet. Once the sediments reach the outlet, then sediments are deposited in the lake/reservoir according to Camp's model (1945) (Verstraeten et al, 2000):

$$TE = \frac{\omega_s}{u_{cr,res}} = \frac{A_{res}}{Q_{out,res}} * \omega_s$$

where TE is the trapping efficiency of the lake/reservoir (or the fraction of particles trapped), ω_s is the particle velocity from Stokes (m/s), $u_{cr,res}$ is the reservoir's critical settling velocity (m/s) which is equal to the reservoir's outflow $Q_{out,res}$ (m^3/s) divided by the reservoir's surface area A_{res} (m^2).

Configuration

The wflow_sediment model was developed as part of the wflow hydrologic platform and is therefore another wflow module, developed in Python, and using the same framework than wflow_sbm. First, the model case is set up and run normally with wflow_sbm. Then wflow_sediment is run using the outputs of the hydrologic model. As settings for wflow_sbm are explained in the corresponding part of this documentation, only specific details regarding the run of wflow_sediment are developed here.

Running wflow_sbm

To model sediment dynamics, the first step is to build a wflow_sbm model and to run it for the catchment considered. Apart from the usual settings for the wflow_sbm model, additional ones for a run with wflow_sediment are to save the following variables in the outputmaps section of the wflow_sbm.ini file:

- Precipitation "self.Precipitation" (can also be taken directly from the wflow_sbm forcings)
- Land runoff (overland flow) from the kinematic wave "self.LandRunoff"
- River runoff from the kinematic wave "self.RiverRunoff"
- Land water level in the kinematic wave "self.WaterLevelL"
- River water level in the kinematic wave "self.WaterLevelR"
- Rainfall interception by the vegetation "self.Interception".

wflow_sediment also needs some static output maps which are saved by default by wflow_sbm. These maps are the map of the actual width and length of the flow volume (Bw and DCL.map). After the set up, wflow_sbm is run normally either via a batch file or via the command line.

Running wflow_sediment

As wflow_sediment is built in the same way as wflow_sbm, its settings and use are very similar. First, some additional data must be downloaded. Then the corresponding ini file that summarizes all inputs and outputs of the model run is completed and the model can finally be run.

Additional data needed

Apart from many data, such as landuse, catchment map, ldd map etc, that are already needed for the wflow_sbm run, wflow_sediment requires some extra additional data which are:

- Map with topsoil percent of clay: this can be download, as for wflow_sbm other soil data, from the SoilGrids database (Hengl et al, 2017). Values then needs to be resampled and adjusted to the model grid size (for the global version of wflow by averaging). This data is mandatory for the sediment model to run.
- Map with topsoil percent of silt: this can also be downloaded from SoilGrids and processed in the same way as the topsoil clay map. This data is mandatory for the sediment model to run.
- Map with topsoil percent of organic carbon: this data can be downloaded from SoilGrids. Units should be in percent (SoilGrids gives it in per-mille) and adjusted to the model grid cells. This data is only needed if the user wishes to calculate the USLE K parameter of soil erosion using the EPIC formula.
- Map of vegetation height: this is available globally using the map published by Simard & al (2011). Other sources can however be used. Units should be in meters. Vegetation height is only needed if the EUROSEM model is used to calculate rainfall erosion.

Setting the ini file

As for wflow_sbm, the setting up of the wflow_sediment model is also done via an ini file and its different sections. A complete example is given in the wflow examples folder. The main sections and options needed are:

- **inputmapstacks**: Links to the dynamic outputs of the wflow_sbm run either stored as maps in the outmaps folder of the sbm run or in the netcdf file. Dynamic data needed are Precipitation, SurfaceRunoff, WaterLevel and Interception.

```
[inputmapstacks]
# Outputs from wflow_sbm
Precipitation           = /inmaps/P
Interception            = /inmaps/int
RiverRunoff = /inmaps/runR
LandRunoff = /inmaps/runL
WaterLevelR = /inmaps/levKinR
WaterLevelL = /inmaps/levKinL
```

- **framework**: As for wflow_sbm, specifies if the inputs or outputs of the model are in netcdf format or PCRaster maps. If the results of wflow_sbm are saved in a netcdf file, link to this file is precised in the *netcdfinput* argument.
- **run**: Info on the run parameters of the model. The start time, end time and timesteps of the model are written in this section. The *reinit* argument precise if the model should start from cold states (all the states maps of the model are set to zero if reinit = 1) or from the states maps given in the instate folder of the model (reinit = 0).
- **modelparameters**: Other parameters used by the model. This section should include the same inputs as the wflow_sbm.ini file for reservoir / lake modelling and Leaf Area Index data.
- **model**: Parameters and settings for the sediment model. It contains both links to the staticmaps of the model (DEM, LDD etc.) and settings to decide which equations to use. These switches are used to choose if both the

soil loss and river part of the model should be run (*runrivermodel* = 1 or 0 for just the soil loss part) and how the model should compute the USLE K factor (*uslekmethod* = 1 for a staticmap, 2 for geometric mean equation and 3 for EPIC equation), the USLE C factor (*uslecmethod* = 1 for a staticmap, 2 for a table based on land use), rainfall erosion (*rainerodmethod* = 1 for EUROSEM and 2 for ANSWERS), inland transport (*landtransportmethod* = 1 for Yalin with particle differentiation, 2 for Govers total transport and 3 for Yalin total transport) and the river transport (*rivtransportmethod* = 1 for Engelund and Hansen, 2 for simplified Bagnold, 3 for Kodatie, 4 for Yang and 5 for Molinas and Wu).

```
# Model parameters and settings
[model]
modeltype= sediment
configfile = wflow_sediment.ini
intbl = intbl
# Run only the soil erosion model (0) or also the river transport model (1)
runrivermodel = 1
#USLE K computation method
#1=map ; 2=geometric mean ; 3=EPIC
uslekmethod = 2
#Rainfall erosion
#1=EUROSEM ; 2=ANSWERS
rainerodmethod = 1
#Inland sediment transport formula
#1=Yalin (particle differentiation) ; 2=Govers (total) ; 3=Yalin (total)
#If the river transport model is run, will be set to 1
landtransportmethod = 1
#River sediment transport formula for erosion threshold
#1=Engelund and Hansen ; 2=Bagnold ; 3=Kodatie ; 4=Yang ; 5=Molinas and Wu
rivtransportmethod = 2

#Iterations of the transport equation
transportIters = 1
transportRiverTstep = 3600

#Model maps from wflow_sbm
wflow_dem = staticmaps/wflow_dem.map
wflow_landuse = staticmaps/wflow_landuse.map
wflow_soil = staticmaps/wflow_soil.map
wflow_subcatch = staticmaps/wflow_subcatch.map
wflow_ldd = staticmaps/wflow_ldd.map
wflow_river = staticmaps/wflow_river.map
wflow_riverwidth = staticmaps/wflow_riverwidth.map
wflow_dcl = staticmaps/DCL.map
wflow_streamorder = staticmaps/wflow_streamorder.map
#Additional model maps for wflow_sediment
wflow_clay = staticmaps/percent_clay.map
wflow_silt = staticmaps/percent_silt.map
wflow_oc = staticmaps/percent_oc.map
wflow_canopyheight = staticmaps/canopy_height.map
```

- **layout**: Specifies if the cell size is given in lat-lon (*sizeinmetres* = 0) or in meters (1). Should be set as in wflow_sbm.
- **outputmaps**: As in wflow_sbm, this section is used to choose which dynamic data to save from the wflow_sediment run. These are:

```
##### Output grids #####
[outputmaps]
#Gross precipitation [mm] (input)
self.Precipitation=P
#Surface runoff in the kinematic wave [m^3/s]
self.SurfaceRunoff=run
#Water level in the kinematic wave [m] (above the bottom)
self.WaterLevel=levKin
#Overland flow [m3/s]
self.OvRun = ovRun
#Soil loss by surface runoff erosion [ton/timestep/cell]
self.SedOv=sedov
#Soil loss by splash erosion [ton/timestep/cell]
self.SedSpl=sedspl
#Total soil loss [ton/timestep/cell]
self.SoilLoss=soilloss
#Sediment from land erosion entering the river [ton]
self.InLandSed = landsed
#Total river inputs [ton]
self.InSedLoad = insed
#River erosion [ton]
self.RivErodSed = erodsed
#Deposition in rivers [ton]
self.DepSedLoad = depsed
#Sediment stored on the river bed [ton]
self.RivStoreSed = rivstore
#Sediment output [ton]
self.OutSedLoad = outsed
#Final sediment load in river cells [ton]
self.SedLoad = sedload
#Total sediment concentration [mg/L]
self.SedConc = sedconc
#Suspended sediment concentration [mg/L]
self.SSConc = ssconc
```

- **summary**: Used to save summary maps of wflow_sediment outputs such as yearly average or yearly sum etc. It works in the same way than for wflow_sbm (see wflow documentation for more details).
- **outputcsv** and **outputtss**: Used to save the evolution of wflow_sediment outputs for specific points or areas of interest in csv or tss format. It works in the same way than for wflow_sbm (see wflow documentation for more details).

Running the model

Once all the settings are ready, the wflow_sediment model is run similarly to wflow_sbm via the command line or a batch file. The minimum command line requires:

- The link to the wflow_sediment script.
- -C option stating the name of the wflow case directory.
- -R option stating the name of the directory of wflow_sediment outputs.
- -c option stating the link to the wflow_sediment ini file.

As in wflow_sbm, the outputs of the wflow_sediment model can both be dynamic netcdf/pcraster maps data, static data, or dynamic data for points/areas of interest. The main outputs variables are soil loss by rainfall and overland flow erosion ("self.SedSpl" + "self.SedOv" = "self.soilloss" in ton/timestep/cell), all the arguments from the sediment mass balance in the river and the total, suspended and bed sediment concentration ("self.SedConc", "self.SSConc" and "self.BedConc" in mg/L). The other outputs are some default summary maps, saved in the outsum folder, and are the USLE C and K factors, the topsoil percent sand (deduced from clay and silt content) and the median river sediment diameter d_{50} in mm. The last outputs are the final states of the model, stored in the outstate folder. In the river model, wflow_sediment need to use the sediment load in the river at the beginning of the timestep, the sediment load coming from upstream river cells at the beginning of the timestep and the amount of deposited sediment that are stored in the river bed and are available for erosion. The three variables are stored in state map files for all particle size class (clay, silt, sand, small and large aggregates, gravel) and total sediment resulting in 21 state maps. If the model is run for the first time and the states are not available, then, as in wflow_sbm, a cold run with initial zero maps for the states can be done first for a year (at least 6 months depending on the size of the model).

References

- K.C. Abbaspour, J. Yang, I. Maximov, R. Siber, K. Bogner, J. Mieleitner, J. Zobrist, and R.Srinivasan. Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT. Journal of Hydrology, 333(2-4):413-430, 2007. 10.1016/j.jhydrol.2006.09.014
- C. Ballabio, P. Panagos, and L. Monatanarella. Mapping topsoil physical properties at European scale using the LUCAS database. Geoderma, 261:110-123, 2016. 10.1016/j.geoderma.2015.07.006
- D.B Beasley and L.F Huggins. ANSWERS - Users Manual. Technical report, EPA, 1991.
- P. Borrelli, M. Marker, P. Panagos, and B. Schutt. Modeling soil erosion and river sediment yield for an intermountain drainage basin of the Central Apennines, Italy. Catena, 114:45-58, 2014. 10.1016/j.catena.2013.10.007
- C. Bosco, D. De Rigo, O. Dewitte, J. Poesen, and P. Panagos. Modelling soil erosion at European scale: Towards harmonization and reproducibility. Natural Hazards and Earth System Sciences, 15(2):225-245, 2015. 10.5194/nhess-15-225-2015
- C.J Brandt. Simulation of the size distribution and erosivity of raindrops and throughfall drops. Earth Surface Processes and Landforms, 15(8):687-698, dec 1990.
- K. Chadli. Estimation of soil loss using RUSLE model for Sebou watershed (Morocco). Modeling Earth Systems and Environment, 2(2):51, 2016. 10.1007/s40808-016-0105-y
- F. Engelund and E. Hansen. A monograph on sediment transport in alluvial streams. Technical University of Denmark 0stervoldgade 10, Copenhagen K., 1967.
- G R Foster. Modeling the erosion process. Hydrologic modeling of small watersheds, pages 295-380, 1982.
- A. Gericke. Soil loss estimation and empirical relationships for sediment delivery ratios of European river catchments. International Journal of River Basin Management, 2015. 10.1080/15715124.2014.1003302
- G. Govers. Empirical relationships for the transport capacity of overland flow. IAHS Publication, (January 1990):45-63 ST, 1990.
- G.J Hanson and A Simon. Erodibility of cohesive streambeds in the loess area of the midwestern USA. Hydrological Processes, 15(May 1999):23-38, 2001.
- T. Hengl, J. Mendes De Jesus, G.B.M. Heuvelink, M. Ruiperez Gonzalez, M. Kilibarda, A. Blagotic, W. Shangguan, M. N. Wright, X. Geng, B. Bauer- Marschallinger, M.A. Guevara, R. Vargas, R.A. MacMillan, N.H. Batjes, J.G.B. Leenaars, E. Ribeiro, I. Wheeler, S. Mantel, and B. Kempen. SoilGrids250m: Global gridded soil information based on machine learning. PLoS ONE, 12(2), 2017. 10.1371/journal.pone.0169748
- R Hessel and V Jetten. Suitability of transport equations in modelling soil erosion for a small Loess Plateau catchment. Engineering Geology, 91(1):56-71, 2007. 10.1016/j.enggeo.2006.12.013

- J.P Julian, and R. Torres. Hydraulic erosion of cohesive riverbanks. Geomorphology, 76:193-206, 2006. 10.1016/j.geomorph.2005.11.003
- D.W. Knight, J.D. Demetriou, and M.E. Hamed. Boundary Shear in Smooth Rectangular Channels. J. Hydraul. Eng., 110(4):405-422, 1984. 10.1061/(ASCE)0733-9429(1987)113:1(120)
- L.D.K. Mansoor, M.D. Matlock, E.C. Cummings, and L.L. Nalley. Quantifying and mapping multiple ecosystem services change in West Africa. Agriculture, Ecosystems and Environment, 165:6-18, 2013. 10.1016/j.agee.2012.12.001
- Q Morgan, J.N Smith, R.E Govers, G Poesen, J.W.A Auerswald, K Chisci, G Torri, D Styczen, and M E Folly. The European soil erosion model (EUROSEM): documentation and user guide. Technical report, 1998.
- S.L Neitsch, J.G Arnold, J.R Kiniry, and J.R Williams. SWAT Theoretical Documentation Version 2009. Texas Water Resources Institute, pages 1-647, 2011. 10.1016/j.scitotenv.2015.11.063
- C. Oeurng, S. Sauvage, and J.M. Sanchez-Perez. Assessment of hydrology, sediment and particulate organic carbon yield in a large agricultural catchment using the SWAT model. Journal of Hydrology, 401:145-153, 2011. 10.1016/j.hydrol.2011.02.017
- P. Panagos, P. Borrelli, K. Meusburger, C. Alewell, E. Lugato, and L. Montanarella. Estimating the soil erosion cover-management factor at the European scale. Land Use Policy, 48:38-50, 2015. 10.1016/j.landusepol.2015.05.021
- K Renard, Gr Foster, Ga Weesies, Dk McCool, and Dc Yoder. Predicting soil erosion by water: a guide to conservation planning with the Revised Universal Soil Loss Equation (RUSLE). Washington, 1997.
- M. Simard, N.Pinto, J. B. Fisher, and A. Baccini. Mapping forest canopy height globally with spaceborne lidar. Journal of Geophysical Research: Biogeosciences, 2011. 10.1029/2011JG001708
- A. Simon, N. Pollen-Bankhead, and R.E Thomas. Development and application of a deterministic bank stability and toe erosion model for stream restoration. Geophysical Monograph Series, 194:453-474, 2011. 10.1029/2010GM001006
- D. Torri, M. Sfalanga, and M. Del Sette. Splash detachment: Runoff depth and soil cohesion. Catena, 14(1-3):149-155, 1987. 10.1016/S0341-8162(87)80013-9
- J. de Vente, J. Poesen, G. Govers, and C. Boix-Fayos. The implications of data selection for regional erosion and sediment yield modelling. Earth Surface Processes and Landforms, 34(15):1994-2007, 2009. 10.1002/esp.1884
- G. Verstraeten and J. Poesen. Estimating trap efficiency of small reservoirs and ponds: methods and implications for the assessment of sediment yield. Progress in Physical Geography, 24(2):219-251, 2000. 10.1177/030913330002400204
- O. Vigiak, A. Malago, F. Bouraoui, M. Vanmaercke, and J. Poesen. Adapting SWAT hillslope erosion model to predict sediment concentrations and yields in large Basins. Science of the Total Environment, 538:855-875, 2015. 10.1016/j.scitotenv.2015.08.095
- O. Vigiak, A. Malago, F. Bouraoui, M. Vanmaercke, F. Obreja, J. Poesen, H. Habersack, J. Feher, and S. Groselj. Modelling sediment fluxes in the Danube River Basin with SWAT. Science of the Total Environment, 2017. 10.1016/j.scitotenv.2017.04.236
- J.R. Williams, K.G. Renard, and P.T. Dyke. EPIC A new method for assessing erosion's effect on soil productivity. Journal of Soil and Water Conservation, 38(5):381-383, sep 1983.
- D. Yang, S. Kanae, T. Oki, T. Koike, and K. Musiake. Global potential soil erosion with reference to land use and climate changes. Hydrological Processes, 17(14):2913-2928, 2003. 10.1002/hyp.1441

wflow_sediment module documentation

1.5.13 The wflow_lintul Model

Introduction

Wflow_lintul, a raster-based crop growth model for rice, is based on LINTUL3, a point-based model for simulating nitrogen-limited rice growth (Shibu et al., 2010). LINTUL3 was parameterized and calibrated for rice by Shibu et al. (2010), based on experimental data from Southeast Asia (Drenth et al., 1994) and drawing on the more complex rice model ORYZA2000 (Bouman et al., 2001) and the preceding versions LINTUL1 (Spitters, 1990) and LINTUL2 (Spitters and Schapendonk, 1990). In contrast to LINTUL3, wflow_lintul is primarily intended for simulation of rice production under water-limited conditions, rather than under nitrogen-limited conditions. To that end, it was designed to function in close cooperation with the spatial hydrological model wflow_sbm, which operates on a watershed-scale.

The LINTUL (Light Interception and Utilization) models were the first deviation from the more complex, photosynthesis-based models of the "De Wit school" of crop modelling, also called the "Wageningen school" (Bouman et al., 1996). In the LINTUL models, total dry matter production is calculated in a comparatively simple way, using the Monteith approach (Monteith, 1969; 1990). In this approach, crop growth is calculated as the product of interception of (solar) radiation by the canopy and a fixed light-use efficiency (LUE; Russell et al., 1989). This way of estimating (daily) biomass production in the LINTUL models is reflected in the equation below and may be considered the core of the wflow_lintul model:

```
GTOTAL = self.LUE * PARINT * TRANRF
```

with GTOTAL the overall daily growth rate of the crop (g m^{-2} d^{-1}), self.LUE a constant light use efficiency (g MJ^{-1}), PARINT the daily intercepted photosynthetically active radiation (MJ m^{-2} d^{-1}), TRANRF (-) the 'transpiration reduction factor', i.e. the reducing impact of water shortage on biomass production.

For regional studies, LINTUL-type models have the advantage that data input requirements are drastically reduced and model parameterization is facilitated (Bouman et al., 1996). LINTUL was first developed for potential crop growth (i.e. perfect growing conditions without any water or nutrient shortages and in absence of pests, diseases and adverse soil conditions) as "LINTUL1" (Spitters, 1990). Later, it was extended to simulate water-limited conditions ("LINTUL2"; Spitters and Schapendonk, 1990) and nitrogen-limited conditions ("LINTUL3"; Shibu et al., 2010). Under water-limited conditions, all growth factors except water are assumed non-limiting, i.e. ample nutrient availability, a pest-, disease- and weed-free environment and no adverse soil conditions. Under nitrogen-limited conditions, only nitrogen availability may limit crop growth. LINTUL has been successfully applied to different crops such as potato (Spitters and Schapendonk, 1990), grassland (LINGRA) (Schapendonk et al., 1998), maize (Farré et al., 2000), oilseed rape (Habekotté, 1997) and rice (Shibu et al., 2010), in potential, water-limited or nitrogen-limited situations.

Differences between wflow_lintul and LINTUL3/other LINTUL versions

- **Potential and water-limited simulations**: Wflow_lintul presently (spring 2018) simulates potential and water-limited crop growth (the latter in conjunction with wflow_sbm). First preparations to add simulation of nitrogen-limited rice growth (LINTUL3) have been made in the present version of the model.
- **Water balance outsourced to wflow_sbm**: The simple water balance, based on Stroosnijder (1982) and Penning de Vries et al. (1989), which was present in LINTUL2 and LINTUL3, is no longer present in wflow_lintul. All water balance-related simulation tasks are outsourced to the wflow_sbm model. On the other hand, several crop growth related tasks in the hydrology model wflow_sbm, such as the simulation of LAI (leaf area index) are now outsourced to wflow_lintul. In the wflow framework, wflow_lintul and wflow_sbm communicate with each other via the Basic Model Interface (BMI) implementation for wflow (wflow_bmi).
- **Written in PCRaster Python**: Whereas the original LINTUL1, LINTUL2 and LINTUL3 models were implemented in the Fortran Simulation Translator (FST) software (Rappoldt and Van Kraalingen, 1996), wflow_lintul is written in PCRaster Python and fully integrated into the wflow hydrologic modelling framework.

- **Written in PCRaster Python**: Integration of wflow_lintul in the wflow framework means that all timer and input-output related tasks for the model are handled by that framework. The original LINTUL1, LINTUL2 and LINTUL3 models were implemented in the Fortran Simulation Translator (FST) software (Rappoldt and Van Kraalingen, 1996). In FST, timer and input-output related tasks were handled by the TTUTIL library (a collection UTILities originating from the former "Theoretische Teeltkunde" research group of Wageningen University and Research; Rappoldt and Van Kraalingen, 1996).
- **Raster based**: In contrast to LINTUL1, 2 and 3, most input and output variables in wflow_lintul are represented by maps or arrays, instead of single values (unless a base-map with a dimension of only one grid cell is specified). Each value in such a map or array represents a grid cell on the map of the study area/catchment; wflow-lintul simulates crop growth for all grid cells simultaneously and can therefore be characterized as a grid-based model. The original LINTUL1, LINTUL2 and LINTUL3 models, in contrast, were point-based and only able to simulate crop growth for one grid cell at a time.
- **Runs for multiple consecutive seasons or years**: As opposed to the original LINTUL models, which run for one season/year at a time, wflow_lintul can continuously run over multiple consecutive cropping cycles and years. At the end of each cropping season, the crop is then harvested automatically, meaning that all variables are reset. At the beginning of each new cropping season, crop growth is automatically re-initiated.
- **Advanced and remotely sensed run control**: Wflow_lintul offers various novel options for controlling the onset and termination of crop growth:

  1. Crop growth can be initiated automatically, on a pixel-by-pixel basis, based on remotely sensed data. To that end, satellite imagery needs to be processed in such a way that for each pixel/grid cell, a Boolean value indicates whether there is (rice) crop growth, predominantly (value = 1), or not (value = 0). The model starts/terminates crop growth if changes 0->1 or vice versa occur.

  2. Crop growth can be (automatically) initiated on a pixel-by-pixel basis if a pre-specified minimum amount of rainfall requirement has accumulated – this amount may be considered necessary for land preparation (soil puddling). In the implementation for central Java, this threshold is normally set at 200 mm rainfall from November 1 on (approximate start of the rainy season), based on Naylor et al. (2007).

  3. Similar to the original point-based LINTUL models, the user can enter a single fixed start date and/or a single fixed end date; these dates will then be applied across all grid cells of the simulated catchment area simultaneously. To simulate growth of transplanted rice, which has already emerged and grown for some time in the nursery before being planted out in the main field, a default initial development stage is specified in the model, together with initial weights of leaves, stems and roots.

Run options

Running wflow_lintul standalone (potential production)

For runs, i.e. without linking to the hydrology model wflow_sbm, the "WATERLIMITED" option in the wflow_lintul.ini should be set to "False". Water limited simulation presently requires the presence of a water balance/hydrology model (i.e. wflow_sbm).

Running wflow_lintul in conjunction with wflow_sbm

For running wflow_lintul in conjunction with the hydrological model wflow_sbm, which takes care of all water-balance related tasks and which is what the model is really intended for, a Python BMI runner module, exchanging data between wflow_sbm and wflow_lintul, needs to be run.

An example of a coupled wflow_sbm and wflow_lintul model is available in \wflow\examples\wflow_brantas_sbm_lintul

To run the coupled models:

- `activate wflow`
- `python bmi2runner.py bmi2runner.ini`

Settings for wflow_lintul

List of run control parameters

Wflow_lintul model runs can be configured by editing the values of a number of parameters in the [model] section of the wflow_lintul.ini file. Wflow_lintul reads the values from wflow_lintul.ini with its parameters function; if a certain parameter or value is not found in wflow_lintul.ini, a default value (provided for each individual parameter in the parameters function) is returned. The following nine variables in wflow_lintul.ini have a run control function:

```
[model]
CropStartDOY = 0
HarvestDAP = 90
WATERLIMITED = True
AutoStartStop = True
RainSumStart_Month = 11
RainSumStart_Day = 1
RainSumReq = 200
Pause = 13
Sim3rdSeason = True
```

For each of these control variables, additional explanation is explained below:

CropStartDOY (integer) - needs to be set to 0 if set from outside. Sets the Julian Day Of the Year (DOY) on which crop growth is initiated – for all grid cells simultaneously. If set to 0, crop growth is initiated automatically for each grid cell individually, based on a certain minimum accumulated rainfall quantity (AutoStartStop is set to True) or based on remote sensing information (if AutoStartStop is set to False; see also Run Control variable 4, AutoStartStop).

HarvestDAP (integer): sets the number of Days After Planting (DAP) on which crop growth is terminated and all crop-related states are reset to 0 – for all grid cells simultaneously. It is advisable to set HarvestDAP = 0, so that crop growth is terminated when the crop is physiologically mature, on a pixel by pixel basis. For rice in Indonesia, this is generally the case after 80-110 DAP. If a positive HarvestDAP value is set and the crop is not yet physiologically mature after the specified number of days, the crop will be harvested immaturely (simultaneously in all pixels). If the crop had already reached physiological maturity before the specified number of DAP, the last simulated values of the state variable (final rice yield, LAI, etc.) will be maintained until the specified number of DAP is reached and then be reset to 0 in all pixels simultaneously.

WATERLIMITED (Boolean): if WATERLIMITED = False, potential crop growth will be simulated, i.e. crop growth without any water limitation (definition: see Introduction). With this option, wflow_lintul can also be run standalone (i.e. without coupling to the hydrological model wflow_sbm). If WATERLIMITED

= True (default), evapotranspiration and soil water status will be simulated by the wflow_sbm model and crop growth will be reduced in case of less-than-perfect water supply (water-limited production).

AutoStartStop (Boolean): if set to True, rice crop growth in the rainy season will be initiated automatically for each grid cell (unless CropStartDOY is set > 0), based on the cumulative amount of precipitation that falls in that grid cell from November 1 on (this date can be changed by adjusting RainSumStart_Month and RainSumStart_Day in wflow_lintul.ini, see Section 4, run control variables 5 and 6). If a pre-specified threshold value (RainReqStartSeason_1, default value 200 mm; see Section 4, run control variable 7) is reached, rice crop growth will be initiated.

The second (dry season) crop will be initiated at a specified number of days after termination of the first (wet season) crop. This period allows for harvesting operations and preparation of the new field to take place and can be changed by modifying the value of the variable "Pause". Similarly, the third (dry season) crop will be initiated at the specified number of days after termination of the second (dry season) crop.

Crop growth during the second season will, similar to the first season, be terminated when the accumulated temperature sum (model internal variable self. TSUM) reaches TTSUM, or at the specified number of days after crop establishment (HarvestDAP) - whichever occurs first.

For all three simulated seasons, crop growth in a certain grid cell will cease - and a final rice yield estimate will be obtained - when the accumulated temperature sum (model internal variable self.TSUM) reaches TTSUM, i.e. the total temperature sum required to reach crop maturity. However, if HarvestDAP is set > 0, crop growth will cease after the specified number of DAP is reached, or when TTSUM is reached – whichever occurs first.

RainSumStart_Month (integer): the month of the year from which the accumulated amount of precipitation (required for initiating the first, rainy, season) is starting to be calculated. Crop simulation starts if the amount (in mm) meets RainReqStartSeason_1.

RainSumStart_Day (integer): the day of the planting month (RainSumStart_Month; parameter 5., above) from which the accumulated amount of precipitation (required for initiating the first, rainy, season) is calculated. Crop simulation starts if the amount (in mm) meets RainReqStartSeason_1.

RainSumReq (real, mm): the accumulated amount of precipitation (in mm) that is required for establishment of the irrigated rice crop at the start of the first (rainy) season.

Pause (integer, days): the number of days between the first (rainy) and second (dry) season and between the second and the third season (if applicable); this period is required for harvesting, land preparation and planting of the second and/or third crop.

Sim3rdSeason (Boolean): if set to True, a third rice crop will be simulated in each simulated year. In case it has not matured on the date specified by Run Control variables 5 and 6, the third crop will be terminated on that date, assuming that farmers will prioritize the first (main) rainy season crop and that in such situations (cooler places), planting 3 rice crops per year is probably not worthwhile.

List of model parameters

```
[model]
LAT = 3.16
LUE = 2.47
TSUMI = 362
K = 0.6
SLAC = 0.02
TSUMAN = 1420
TSUMMT = 580.
TBASE = 8.
```

(continues on next page)

(continued from previous page)

```
RGRL = 0.009
WLVGI = 0.86
WSTI = 0.71
WRTLI = 1.58
WSOI = 0.00
DVSDR = 0.8
RDRRT = 0.03
RDRSHM = 0.03
LAICR = 4.0
ROOTDM_mm = 1000.
RRDMAX_mm = 10.
ROOTDI_mm = 50.
RDRTB = [0.0,0.0, 0.6,0.00, 1.0,.015, 1.6,0.025, 2.1,0.05, "RDRTB"]
PHOTTB = [0.0,0.0, 8.0,0.0, 10.0,1.0, 12.0,1.0, 13.0,0.8, 14.0,0.6, 18.,0.0, "PHOTTB"]
SLACF = [0.0,1.72, 0.21,1.72, 0.24,1.72, 0.33,1.32, 0.7,1.20,1.01,1.00, 2.0,0.75, 2.1,0.
↪75, "SLACF"]
FRTTB = [0.0,0.300, 0.48,0.30, 0.62,0.12, 0.69,0.11, 0.84,0.1,.92,0.10, 1.00,0.08, 1.38,
↪0.00, 2.11,0.0, "FRTTB"]
FLVTB = [0.0,0.315, 0.48,0.35, 0.62,0.44, 0.69,0.463, 0.84,0.463, 0.92,0.45, 1.00,0.00,␣
↪1.38,0.00, 2.10,0.0, "FLVTB"]
FSTTB = [0.0,0.385, 0.48,0.35, 0.62,0.44, 0.69,0.427, 0.84,0.427, 0.92,0.27, 1.00,0.00,␣
↪1.38,0.00, 2.10,0.0, "FSTTB"]
FSOTB = [0.0,0.00, 0.48,0.00, 0.62,0.00, 0.69,0.00, 0.84,0.00, 0.92,0.18, 1.00,0.92, 1.
↪38,1.00, 2.10,1.00, "FSOTB"]
```

For each of the above parameters, additional explanation is explained below:

LAT (real, degrees): the geographic latitude, required for calculation of the correct astronomic day length for each day of the simulation; day length may influence crop-physiological processes such as the allocation of carbohydrates, the initialization of flowering, etc. It assumed that day length is identical throughout the simulated catchment area; in very large catchments it might, theoretically, be preferable to calculate the day length for each individual grid cell. Actual day length calculations are based on the FORTRAN subroutine SASTRO (Van Kraalingen, 1995) which is, in turn, based on Goudriaan and van Laar (1994, around p.30).

LUE (real, g biomass MJ:math:`^{-1}` of intercepted solar radiation): the light use efficiency of the crop or variety that is simulated. For rice, the default value is 3.0 (Shibu et al., 2010). To account for different varieties, management imperfections or adverse conditions, a different value may be obtained by local calibration with observed rice yields, preferably of well-managed experiments that approach potential or water limited growing conditions, i.e. in absence of nutrient shortages, toxic elements or pests and diseases. The potential yield level is determined by the growth-defining factors, i.e. incoming solar radiation, temperature and characteristics of the crop when the crop is optimally supplied with water and nutrients and is completely protected against growth-reducing factors. Water-limited and nutrient-limited yield levels are lower than the potential, due to a suboptimal supply of water and/or nutrients, respectively. The actual production level is determined by actual supplies of water and nutrients, and by the degree that the crop is protected against growth-reducing factors or escapes their effects (Van Ittersum and Rabbinge, 1997). Alternatively, if the LUE of wflow_lintul is calibrated against actual farm yields, the model may be used to predict actual crop yields, instead of water-limited or potential crop yields, taking into the account the average occurrence nutrient shortages, toxic elements or pests and diseases in the relevant region and over the relevant period of time.

TSUMI (real, degree days): the initial temperature sum (at the moment of transplanting, only relevant for transplanted rice).

K (real, dimensionless): the light extinction coefficient (for rice leaves). Light interception in LINTUL increases with LAI according to a negative exponential pattern, characterized by a crop-specific light extinction coefficient (K, m^2 (ground) m^{-2} (leaf)): an implementation of Lambert-Beer's law.

SLAC (real, m:math:`^{2}` (leaf) g:math:`^{-1}` (leaf)): Specific Leaf Area Constant, introduced by Shibu et al. (2010). SLAC is multiplied by SLACF, a Leaf area correction function as a function of development stage to obtain SLA, the specific leaf area (real, m^2 (leaf) g^{-1} (leaf)).

TSUMAN (real, degree days): the temperature sum required for the crop to reach anthesis.

TSUMMT (real, degree days): the temperature sum required for the crop to develop from anthesis to maturity (rice ready for harvest).

TBASE (real, degrees centigrade): since many growth processes are temperature dependent above a certain threshold temperature, only temperatures above a certain minimum temperature (TBASE, real, degrees centigrade) lead to increase in temperature sum/crop phenological development. For rice, TBASE is 8 °C (Shibu et al., 2010) hence below that temperature, no crop development takes place.

RGRL (real, dimensionless): the relative (daily) growth rate of leaf area during the exponential growth phase, expressed per degree-day.

WLVGI, WSTI, WRTLI, WSOI: Initial weight (at transplanting time) of green leaves, stems, roots and storage organs (i.e. rice grains), respectively, in g DM m2.

DVSDR (real, dimensionless): value of the Crop Development Stage, DVS) above which the death of leaves and roots sets in.

RDRRT (real, dimensionless): relative (daily) death rate of roots.

RDRSHM (real, dimensionless): maximum relative (daily) death rate of leaves due to shading.

LAICR (real, dimensionless): value of the crop Leaf Area Index (LAI) above which mutual shading of leaves occurs which may in turn influence the death rate of leaves.

ROOTDM_mm (real, mm): maximum rooting depth for a rice crop.

RRDMAX_mm (real, mm d:math:`^{-1}`): maximum daily increase in rooting depth (m) for a rice crop.

ROOTDI_mm (real, mm): initial rooting depth (after transplanting).

RDRTB (real, dimensionless): interpolation table defining the relative (daily) death rate of leaves (RDRTMP) as a function of Developmental Stage. Adopted by Shibu et al. (2010) from Bouman et al. (2001).

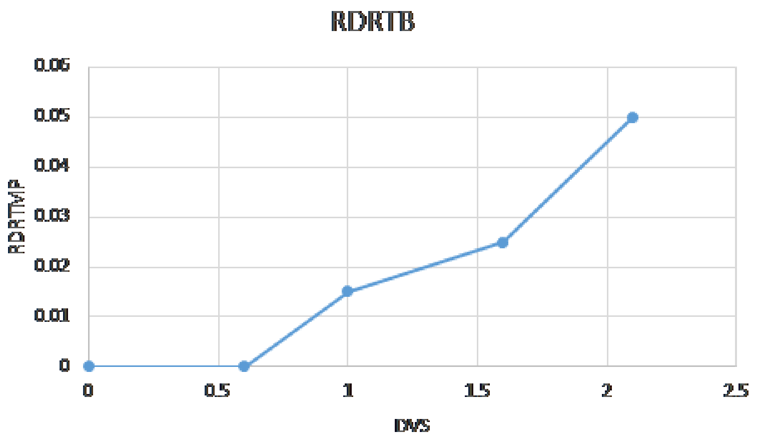


Fig. 17: The RDRTB parameter (list) defines the relative (daily) death rate of leaves (RDRTMP) as a function of crop Developmental Stage (DVS)

PHOTTB (real, dimensionless): interpolation table defining the modifying effect of photoperiodicity on the phenological development rate (from LINTUL3/Shibu et al. 2010 - original source unclear) as a function of daylength (DAYL, hours).

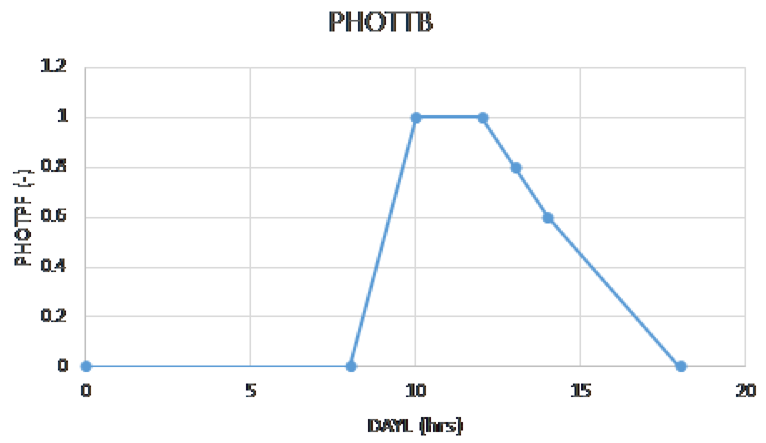


Fig. 18: The PHOTTB parameter (table) defines the modifying effect of photoperiodicity on the phenological development rate as a function of day length (DAYL)

SLACF (real, dimensionless): interpolation table defining the leaf area correction factor (SLACF) as a function of development stage (DVS; Drenth et al., 1994). The Specific Leaf Area Constant (SLAC) is multiplied with a correction factor to obtain the specific leaf area (SLA); this correction factor, in turn, is obtained by linear interpolation in the SLACF table (Figure 3), using the relevant value of DVS as independent variable.

```
SLA = self.SLAC * self.SLACF.lookup_linear(self.DVS)
```

where self.SLACF.lookup_linear(self.DVS) results in the relevant value of the correction factor by linear interpolation based on DVS.

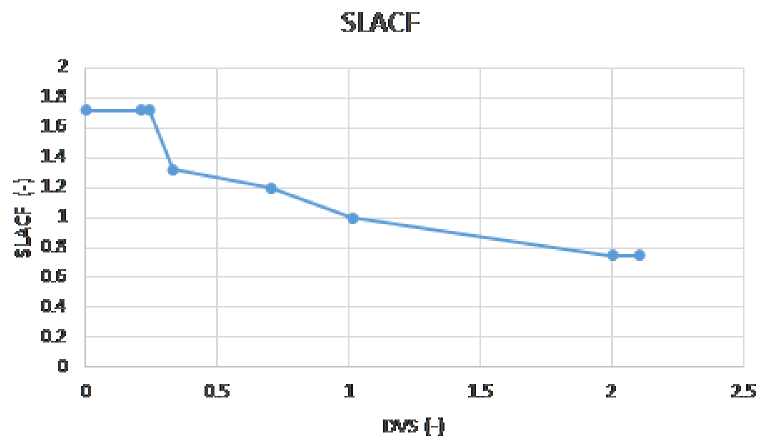


Fig. 19: The SLACF interpolation table defines the leaf area correction factor (SLACF) as a function of development stage (DVS)

FRTTB (real, dimensionless): interpolation table defining the fraction of daily dry matter production allocated to root growth (FRTWET), in absence of water shortage, as a function of development stage (DVS). FRTWET is obtained by linear interpolation in FRTTB, with DVS as the independent variable.

```
FRTWET = self.FRTTB.lookup_linear(self.DVS)
```

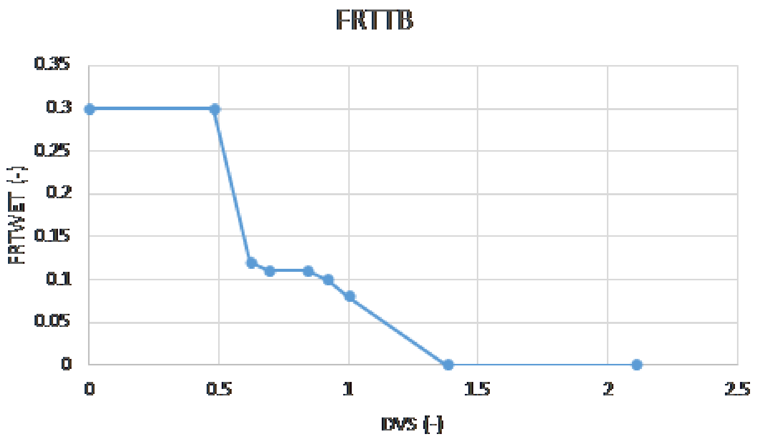


Fig. 20: The FRTTB interpolation table defines the fraction of daily dry matter production allocated to root growth (FRTWET) in absence of water shortage as a function of the crop development stage (DVS).

FLVTB (real, dimensionless): interpolation table defining the fraction of daily dry matter production allocated to growth of leaves (FLVT), in absence of water shortage, as a function of development stage (DVS). FLVT is obtained by linear interpolation in FLVTB, with DVS as the independent variable:

```
FLVT = self.FLVTB.lookup_linear(self.DVS)
```

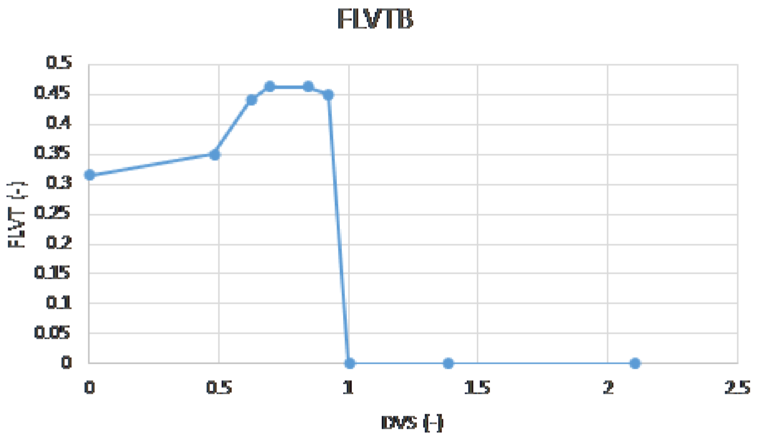


Fig. 21: The FLVTB interpolation table defines the fraction of daily dry matter production allocated to growth of leaves (FLVT) in absence of water shortage, as a function of development stage.

FSTTB (real, dimensionless): interpolation table defining the fraction of daily dry matter production allocated to growth of stems (FSTT), in absence of water shortage, as a function of development stage (DVS). FSTT is obtained by linear interpolation in FSTTB, with DVS as the independent variable:

```
FSTT = self.FSTTB.lookup_linear(self.DVS)
```

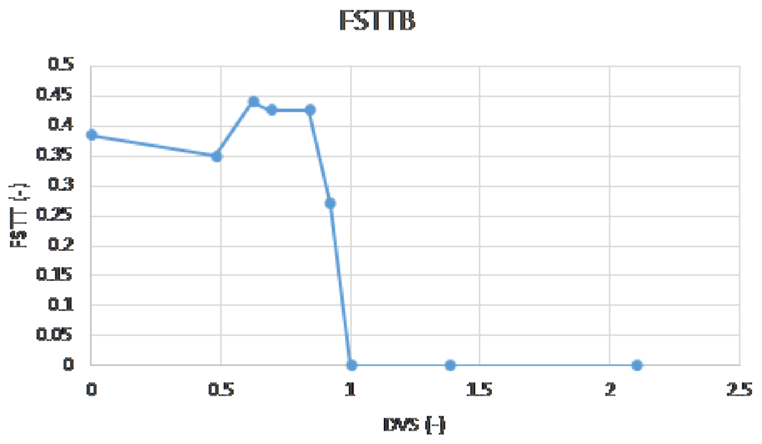


Fig. 22: The FSTTB interpolation table defines the fraction of daily dry matter production allocated to growth of stems (FSTT) in absence of water shortage, as a function of the crop development stage (DVS).

FSOTB (real, dimensionless): interpolation table defining the fraction of daily dry matter production allocated to growth of stems (FSOT), in absence of water shortage, as a function of development stage (DVS; Figure 7). FSOT is obtained by linear interpolation in FSOTB, with DVS as the independent variable:

```
FSOT = self.FSOTB.lookup_linear(self.DVS)
```

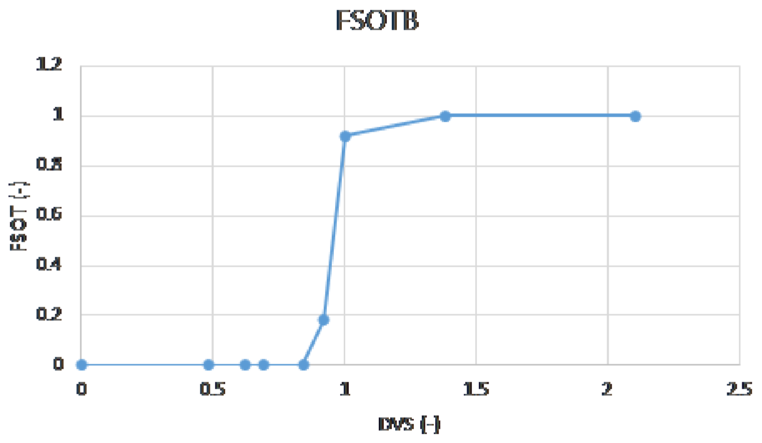


Fig. 23: The FSOTB interpolation table defines the fraction of daily dry matter production allocated to growth of stems (FSOT), in absence of water shortage, as a function of the crop development stage (DVS).

Model forcing data

In terms of meteorological data, wflow_lintul requires daily values of:

- Solar radiation (IRRAD; kJ m^{-2} d^{-1})
- Average temperature (T; °C)
- Precipitation (RAIN, mm)

For water-limited simulations, LINTUL-type models normally also require data of **vapour pressure** and **wind speed** for calculating evapotranspiration. However, in the case of wflow_lintul, actual (rice) crop transpiration (Transpiration; mm d^{-1}) and potential (rice) crop transpiration (PotTrans; mm d^{-1}) under water-limited crop growth are calculated by the wflow_sbm hydrologic model which then supplies these variables to wflow_lintul via the BMI. They are then used for calculating TRANRF, a factor that describes the reducing effect of water shortage on biomass production.

Wflow_sbm, in turn, requires potential evaporation as a forcing variable. Similarly, water content in the rooted zone (self.WA) is simulated by wflow_sbm and then transferred to wflow_lintul via the BMI.

In turn, for wflow_sbm being able to perform these calculations, wflow_lintul supplies it with daily values of LAI (m^2 leaf m^{-2} ground) and rooting depth to wflow_sbm, again via the BMI.

Description of core wflow_lintul model code

Phenology

Phenological development of crops is generally closely related to thermal time, i.e. the accumulated number of degree-days after emergence. Instead of simply adding up degree-days, a daily effective temperature (DTEFF, °C d) is used in wflow_lintul however, since many growth processes are only temperature dependent, or only occur, above a certain threshold temperature. DTEFF is calculated according to:

```
DTEFF = ifthenelse(Warm_Enough, DegreeDay, 0.)
```

with DTEFF the daily effective temperature (°C d), Warm_Enough a Boolean variable that equals True when the daily average temperature T is larger than or equal to TBASE, the base temperature for a rice crop (°C). DegreeDay is the daily average temperature reduced with TBASE:

```
DegreeDay = self.T - self.TBASE
```

Thus, if T is great than TBASE, DTEFF is set equal to DegreeDay; in all other cases it is set to 0. In addition, before DTEFF is added to TSUM (state variable), it is corrected for the potentially modifying effect of day length (photoperiodicity) resulting in the daily increase in temperature sum RTSUMP (°C d), according to:

```
RTSUMP = DTEFF * PHOTPF
```

where RTSUMP is the daily increase in temperature sum (°C d), modified by the influence of photoperiod DTEFF the daily effective temperature (°C d), PHOTPF is a correction factor that accounts for the modifying influence of day length on crop development via DTEFF; it is defined as a function of day length (DAYL). PHOTPF is less than 1 if day length (DAYL, hours) is shorter than 10 hours or longer than 12 hours. Day lengths between 10-12 hours have no modifying influence on DTEFF; phenological development is thus slowed down in such cases. DAYL in wflow_lintul is calculated by the function astro_py , a Python implementation of a FORTRAN subroutine from the Wageningen school of models, dating back to Spitters et al. (1989) and likely even further.

Calculation of TSUM (model state variable) by accumulating daily values of RTSUMP is then done according to:

```
self.TSUM = (self.TSUM + ifthenelse(CropStartNow, scalar(self.TSUMI), 0.) +
ifthenelse(EMERG, RTSUMP, 0.)) * ifthenelse(CropHarvNow, scalar(0.), 1.)
```

with self.TSUM the temperature sum (a crop state variable, °C d), CropStartNow: a Boolean variable that equals True on the moment of crop growth initiation TSUMI (real, degree days): the initial temperature sum (at the moment of transplanting, only relevant for transplanted rice), EMERG is a Boolean variable that indicates whether crop growth occurs; it equals True, when three conditions are met:

- crop growth has been initiated
- the water content of the soil is larger than the water content at permanent wilting point
- LAI is greater than 0.
- CropHarvNow a Boolean variable that only equals True on the day that the crop is being harvested

Thus, the state of TSUM of the previous day is increased (on the current day) with the initial temperature at the moment of transplanting/initiation of crop growth, and increased with the daily increase in (effective) temperature during the subsequent period of crop growth. At the moment of harvest (when :math:CropHarvNow = True), the third ifthenelse statement will return a zero value, hence multiplying TSUM with zero (effectively resetting it). Regarding the influence of TSUM on rice crop development two crop-specific parameters are of paramount importance, since assimilates are partitioned differently over the different plant organs before flowering (vegetative growth) than after flowering (generative growth):

- TSUMAN, the value of TSUM at which crop anthesis is initiated (1420 °C d for rice variety IR72)
- TSUMMT, the change in TSUM required for the crop to develop from anthesis to maturity (hence ripened rice grain can be harvested) - 580 °C d for rice variety IR72.

In LINTUL1 and LINTUL2, TSUM controls all processes that are influenced by crop phenological development. However, Shibu et al. (2010) derived important parts of LINTUL3 (for rice) from ORYZA2000 (Bouman et al., 2001). In ORYZA2000, most processes are steered by a variable called DVS (development stage; -). The development stage of a plant defines its physiological age and is characterized by the formation of the various organs and their appearance. The key development stages for rice distinguished in ORYZA2000 are emergence (DVS = 0), panicle initiation (DVS = 0.65), flowering (DVS = 1), and physiological maturity (DVS = 2; Bouman et al., 2001). Shibu et al. incorporated DVS into LINTUL which is why some processes are controlled directly by TSUM and others are controlled by its derived variable DVS – a situation that seems to offer scope for future improvement. In wflow_lintul, the key development stages emergence, flowering and physiological maturity coincide with TSUM = 0, TSUM = 1420 (flowering) and TSUM = 2000 (140 + 580; crop maturity), respectively. Panicle initiation (at DVS = 0.65) is not a significant event in wflow_lintul. DVS is calculated from :math:TSUM according to the following three equations:

```
DVS_veg  = self.TSUM / self.TSUMAN * ifthenelse(CropHarvNow, scalar(0.), 1.)

DVS_gen  = (1. + (self.TSUM - self.TSUMAN) / self.TSUMMT) *␣
↪ifthenelse(CropHarvNow, scalar(0.), 1.)

self.DVS  = ifthenelse(Vegetative, DVS_veg, 0.) + ifthenelse(Generative, DVS_
↪gen, 0.)
```

where: DVS_veg: DVS during the vegetative crop stage, CropHarvNow a Boolean variable that equals True when the crop is mature or has reached a fixed pre-defined harvest date. DVS_veg: DVS during the generative crop stage, TSUMAN, the value of TSUM at which crop anthesis is initiated (1420 °C d for rice variety IR72) TSUMMT, the change in TSUM required for the crop to develop from anthesis to maturity (hence ripened rice grain can be harvested) - 580 °C d for rice variety IR72. DVS, the crop (phenological) development stage.

Hence, DVS is calculated as the sum of DVS_veg and DVS_gen; it equals 1 (flowering) if TSUM reaches TSUMAN and equals 2 (maturity) if TSUM reaches the sum TSUMAN + TSUMMT. DVS_veg and DVS_gen are both reset (multiplied with zero) when the crop is harvested.

Photosynthesis and crop growth

As outlined in the Introduction, overall assimilate production rate in wflow_lintul is calculated as the product of the Photosynthetically Active (solar) Radiation (PAR) that is intercepted by the crop canopy and a fixed Light-Use Efficiency (LUE; Russell et al., 1989). This way of estimating (daily) biomass production in the LINTUL models is reflected as follows and may be considered the core of the wflow_lintul model:

```
GTOTAL = self.LUE * PARINT * TRANRF
```

with GTOTAL the overall daily growth rate of the crop (g m^{-2} d^{-1}), self.LUE a constant light use efficiency (g MJ^{-1}), PARINT the daily intercepted photosynthetically active radiation (MJ m^{-2} d^{-1}), TRANRF (-) the 'transpiration reduction factor', i.e. the reducing impact of water shortage on biomass production.

The daily intercepted PAR (PARINT,MJ m^{-2} d^{-1}), is calculated as:

```
PARINT = ifthenelse(Not_Finished, 0.5 * self.IRRAD * 0.001 * (1. - exp(-self.K␣
↪* self.LAI)), 0.)
```

with: Not_Finished a Boolean variable that indicates whether the crop is still growing and developing (True) or not (False). 0.5 a factor to account for the fraction of photosynthetically active radiation in the incident solar radiation. About 50% (in terms of energy) of the frequency spectrum of incident solar radiation can be used for photosynthesis by green plants. IRRAD: incident solar radiation (m^{-2} d^{-1}) as measured by e.g. a weather station. K a crop and variety-specific light extinction coefficient (-) and LAI the leaf area index (m^2 leaf m^{-2} ground).

Calculation of growth rates

The overall daily growth of the crop (GTOTAL, g m^2 d^1) is partitioned over growth of leaves, stems, storage organs (grains in the case of rice) and roots:

```
RWLVG = ifthenelse(EMERG, GTOTAL * FLV - DLV, scalar(0.))
RWSO = ifthenelse(EMERG, GTOTAL * FSO, scalar(0.))
RWRT = ifthenelse(EMERG, GTOTAL * FRT - DRRT, scalar(0.))
```

where RWLVG, RWST, RWSO and RWRT are the daily growth rates of leaves, stems, storage organs (i.e. rice grains) and roots, respectively (g m^2 d^1), GTOTAL is the overall daily growth rate of the crop (g m^2 d^1), EMERG is a Boolean variable that indicates whether crop growth occurs; it equals True, when three conditions are met:

- crop growth has been initiated
- the water content of the soil is larger than the water content at permanent wilting point
- LAI is greater than 0.

FLV, FST, FSO and FRT are the fractions (-) of the overall biomass growth rate (GTOTAL; g m^2 d^1) allocated to leaves, stems, storage organs (rice grains) and roots, respectively. They are related to the phenological development stage of the crop (DVS), following relationships defined in the parameter interpolation tables FLVTB, FSTTB, FSOTB, and FRTTB. If water shortage occurs, growth of belowground and aboveground crop parts is modified with the factors FRTMOD and FSHMOD, respectively:

::

FLV = FLVT * FSHMOD FST = FSTT * FSHMOD FSO = FSOT * FSHMOD FRT = FRTWET * FRTMOD

where FLVT, FSTT, FSOT and FRTWET are the (raw) interpolated allocation fractions for leaves, stems, storage organs (rice grains) and roots, respectively and FLV, FST, FSO and FRT are the final allocated fractions, after modification for water shortage (if any). DLV and DRRT are the death rates of leaves and roots, respectively (g m^2 d^1).

Summarizing: as long as crop growth occurs (i.e. if EMERG = True), growth rates of the different plant organs are calculated by multiplying overall crop growth with certain organ-specific fractions, depending on crop development stage via pre-defined interpolation functions, and on water availability. Net growth rates are obtained by diminishing the growth with the weight of the foliage that has died. The daily change in leaf weight as a consequence of the dying of foliage is calculated according to:

```
DLV = self.WLVG * RDR
```

with self.WLVG the total green leaf biomass (g m^2), RDR the relative (daily) death rate of leaves (-).

The daily change in leaf weight as a consequence of the dying of foliage is calculated according to:

```
DRRT = ifthenelse(Roots_Dying, self.WRT * self.RDRRT,scalar(0.))
```

with self.WLVG the total roots biomass (g m^2), RDRRT the relative (daily) death rate of roots.

Now that the net (mass) growth rates of leaves, stems, storage organs (rice grain) and roots are known, their respective masses (model state variables) can also be calculated, according to:

```
self.WLVG = (self.WLVG + ifthenelse(CropStartNow, self.WLVGI, scalar(0.)) +␣
↪RWLVG) * (1. - scalar(CropHarvNow))
self.WST = (self.WST + ifthenelse(CropStartNow, self.WSTI, scalar(0.)) + RWST)␣
↪* (1. - scalar(CropHarvNow))
self.WSO = (self.WSO + ifthenelse(CropStartNow, self.WSOI, scalar(0.)) + RWSO)␣
↪* (1. - scalar(CropHarvNow))
self.WRT = (self.WRT + ifthenelse(CropStartNow, self.WRTLI, scalar(0.)) +␣
↪RWRT) * (1. - scalar(CropHarvNow))
```

with:

- WLVG, WST, WSO and WRT the weights of green leaves, dead leaves, stem, storage organs and roots (g m^2)
- CropStartNow: a Boolean variable that equals True on the moment of crop growth initiation
- WLVGI, WSTI, WSOI, WRTLI the initial weights of green leaves, dead leaves, stem, storage organs and roots (g m^2), i.e. at the moment of transplanting
- RWLVG, RWST, RWSO and RWRT the daily growth rates of leaves, stems, storage organs (i.e. rice grains) and roots, respectively (g m^2 d^1)
- CropHarvNow a Boolean variable that only equals True on the day that the crop is being harvested

Daily change in Leaf Area Index (LAI)

Whereas the daily increase in leaf biomass was already explained above, the daily increase in leaf area (Leaf Area Index, LAI, m^2 m^2) is simulated as:

```
GLAI = ifthenelse(LetsGro, self.LAII, scalar(0.)) + ifthenelse(Juv_or_Harv,␣
↪self.LAI * (exp(self.RGRL * DTEFF)-1.) * TRANRF
+ ifthenelse(Adt_or_Harv, SLA * GLV, scalar(0.))
```

with:

- GLAI the (daily) growth in LAI (m^2 m^2 d^1)
- LetsGro a Boolean variable that, as soon as it becomes “True”, triggers the initiation of crop growth;
- LAII (m^2 m^2) the initial value of LAI (leaf area index)

- Juv_or_Harv a Boolean variable indicating that the crop is juvenile or is being harvested
- RGRL the relative (daily) growth rate of leaf area (-), expressed per degree-day ((°Cd)1)
- DTEFF the daily effective temperature (°C d).
- TRANRF the 'transpiration reduction factor', i.e. the relative impact of water shortage on biomass production
- Adt_or_Harv a Boolean variable indicating whether the crop is adult or is being harvested (True) or none of those two (False).
- SLA the specific Leaf area (m2/g). The model calculates the growth of leaf area by multiplying the simulated increase in leaf weight (GLV, g m^2 d^1) by the specific leaf area of new leaves (SLA, m^2 g^1).
- GLV the simulated increase in leaf weight (GLV, g m^2 d^1)

LAI is calculated as follows:

```
self.LAI   = (self.LAI + GLAI - DLAI) * ifthenelse(CropHarvNow, scalar(0.), 1.)
```

with LAI the leaf area index ($m^2 m : math : `^2), GLAI the (daily) growth in LAI (m : math : `^2$ m^2 d^1), DLAI the daily decrease in LAI (DLAI, m^2 m^2 d^1) from dying of leaves, CropHarvNow a Boolean variable that only equals True on the day that the crop is being harvested.

The daily decrease in LAI (DLAI, $m^2 m : math : `^2$ d^1) from dying of leaves is, analogous to the calculation of DLV (the death rate of leaves in terms of mass), calculated as:

```
DLAI = self.LAI * RDR
```

with LAI the leaf area index (m^2 m^2; state variable), RDR the relative (daily) decline in LAI due dying of leaves (-). RDR in turn depends on two terms, the relative (daily) death rate of leaves due to aging (RDRDV, -) and the relative (daily) death rate of leaves due to mutual shading (RDRSH, -), according to:

```
RDR = max(RDRDV, RDRSH)
```

The relative (daily) death rate of leaves due to aging (RDRDV, -), in turn, is calculated following a number of steps, starting with:

```
RDRDV = ifthenelse(AtAndAfterAnthesis, RDRTMP, scalar(0.))
```

with AtAndAfterAnthesis a Boolean variable that equals True if the crop has reached anthesis. Hence, if AtAndAfterAnthesis equals True, RDRDV is set equal to RDRTMP; in all other cases it is set to 0. RDRTMP is the relative (daily) death rate obtained by interpolation in the RDRTB table, with DVS as the independent variable.

The relative (daily) death rate of leaves due to mutual shading (RDRSH, -) is calculated according to:

```
RDRSH = max(0., self.RDRSHM * (self.LAI - self.LAICR)/self.LAICR)
```

with RDRSHM is a fixed daily relative death rate due to shading (-), LAICR is the critical LAI above which mutual shading starts to occur.

Root (depth) growth

Root depth growth can only occur if a number of (logical) conditions are simultaneously met:

```
RootGrowth = Enough_water & BeforeAnthesis & EMERG & CanGrowDownward
```

where:

- RootGrowth is a Boolean variable indicating whether root depth growth occurs (True) or not (False).
- Enough_water is a Boolean variable indicating whether there is water for crop growth (True) or not (False, this is the case when soil moisture content is at permanent wilting point).
- BeforeAnthesis is a Boolean variable indicating whether the crop growth has reached the anthesis stage in its phenological development (True) or not yet (False). It is calculated as:
- BeforeAnthesis = self.TSUM < self.TSUMAN with self.TSUM the accumulated temperature sum of the crop and self.TSUMAN the temperature sum required for the crop to reach anthesis. Hence, BeforeAnthesis remains True as long as the crop has not yet accumulated the temperature sum required for it to reach the anthesis development stage.
- EMERG is a Boolean variable that indicates whether crop growth occurs; it equals 'True', when three conditions are met:
  - crop growth has been initiated
  - the water content of the soil is larger than the water content at permanent wilting point
  - LAI is greater than 0.
- CanGrowDownward is a Boolean variable indicating whether roots can grow further downward (True) or not (False). The latter is the case is the maximum rooting depth has been reached. It is calculated as:

  ```
  CanGrowDownward = self.ROOTD_mm <= self.ROOTDM_mm
  ```

  with self.ROOTD_mm the rooting depth in mm and self.ROOTDM_mm is the maximum rooting depth in mm

Hence, as long as the roots have not yet reached their maximum depth, CanGroDownward remains True. Actual root growth only occurs where RootGrowth equals True:

```
RROOTD_mm = ifthenelse(RootGrowth, self.RRDMAX_mm, scalar(0.))
```

where RROOTD_mm is the daily increase in rooting depth (mm), RootGrowth is a Boolean variable indicating whether root depth growth occurs (True) or not (False). self.RRDMAX_mm is the maximum daily increase in rooting depth (mm)

So, if RootGrowth is True, the daily increase in rooting depth (mm) will be set equal to the maximum daily increase in rooting depth (mm); in all other cases it will be set to zero. There is no simulation of a reducing effect of root death on rooting depth. Root death only impacts the weight of (living) roots as they are diminished with the (daily) death rate of roots.

The influence of water stress on crop growth

As wflow_lintul is a much simpler rice model than e.g. ORYZA2000 (Bouman et al., 2001), the effect of water stress on crop growth is also modelled in a simpler way - there is no mechanistic simulation of crop drought responses such as leaf rolling, or of events such as spikelet sterility. These events can only be taken into account indirectly, by calibrating model drought response to match observed yield and biomass data – a process that has presently (June 2018) not yet been entirely completed.

A central parameter in modelling the response to water shortage in wflow_lintul is TRANRF, a factor that describes the reducing effect of water shortage on biomass production, having a direct impact on overall crop production and leaf area growth. It is calculated according to:

```
TRANRF = self.Transpiration/NOTNUL_pcr(self.PotTrans)
```

with self.Transpiration the actual (rice) crop transpiration (mm d-1), self.PotTrans the potential (rice) crop transpiration (mm d-1), NOTNUL_pcr a Python implementation of an FST intrinsic function (Rappoldt and Van Kraalingen, 1996), returning 1 if the value between parentheses equals 0, to prevent zero division errors. In all others cases, it returns the unchanged value.

Both self.Transpiration and NOTNUL_pcr(self.PotTrans) are calculated by the wflow_sbm hydrologic model which then supplies these variables to wflow_lintul via the BMI.

Indirectly, TRANRF also impacts crop growth through influencing the crop's root/shoot ratio. If crops are subjected to drought or nutrient stress, it is commonly observed that they are inclined to invest more in root growth (in search of water or nutrients, arguably). In wflow_lintul, TRANRF modifies root growth as soon as it drops below the value of 0.5 (i.e. as soon as crop transpiration is half or less its potential, under the given circumstances). Root growth is modified with a factor FRTMOD. FRTMOD is calculated according to:

```
FRTMOD = max(1., 1./(TRANRF + 0.5))
```

with TRANRF the 'transpiration reduction factor', i.e. the reducing impact of water shortage on biomass production

Similarly, shoot growth is modified with a factor FSHMOD. FSHMOD is calculated according to:

```
FSHMOD = (1. -FRT)/(1 - FRT/FRTMOD)
```

with FRT the final allocated fraction of total crop growth allocated to the roots, after modification for water shortage. FRTMOD a factor describing the modifying effect of drought stress on root growth.

References

- Bouman, B.A.M., van Keulen, H., van Laar, H.H., Rabbinge, R., 1996. The 'School of de Wit' crop growth simulation models: a pedigree and historical overview. Agric. Syst. 52, 171/198.
- Bouman, B.A.M., M.J. Kropff, T.P. Tuong, M.C.S. Wopereis, H.F.M. ten Berge, and H.H. van Laar. 2001. ORYZA2000: Modelling lowland rice. 235 p. International Rice Research Institute, Los Baños, Philippines, Wageningen University and Research Centre, Wageningen, The Netherlands.
- Drenth, H., Ten Berge, H.F.M. and Riethoven, J.J.M.(Editors). ORYZA simulation modules for potential and nitrogen limited rice production. SARP Research Proceedings-December, 1994. DLO-Research Institute for Agrobiology and Soil fertility, Wageningen, WAU-Department of Theoretical Production Ecology, Wageningen, IRRI-International Rice Research Institute, Los Banos, Pages: 197-210.
- Farré, I., Van Oijen, M., Leffelaar, P.A., Faci, J.M., 2000. Analysis of maize growth for different irrigation strategies in northeastern Spain. Eur. J. Agron. 12, 225–238.
- Goudriaan, J. and van Laar, H. H., 1994. Modelling Potential Crop Growth Processes. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994. pp. 238.

- Habekotté, B., 1997. Description, parameterization and user guide of LINTULBRASNAP 1.1. A crop growth model of winter oilseed rape (Brassica napus L.). In: Quantitative Approaches in Systems Analysis No. 9. Wageningen Agricultural University, Wageningen, The Netherlands, 40 pp.
- Van Ittersum, M.K., Rabbinge, R., 1997. Concepts in production ecology for analysis and quantification of agricultural input-output combinations. Field Crops Research 52 (1997) 197-208
- Van Kraalingen, D.W.G., 1995. The FSE system for crop simulation, version 2.1. Quantitative Approaches in Systems Analysis, No. 1. C.T. de Wit Graduate School for Production Ecology and Resource Conservation, Wageningen University, The Netherlands, pp. 58.
- Monteith, J. L. (1969). Light interception and radiative exchange in crop stands. In Physiological aspects of crop y,ie/d, eds J. D. Easton, F. A. Haskins, C. Y. 194 B. A. M. Bouman, H. van Keulen, H. H. van Laar, R. Rabbinge Sullivan & C. H. M. van Bavel. American Society of Agronomy, Madison, Wisconsin. pp. 89-l 11.
- Monteith, J. L. (1990). Conservative behaviour in the response of crops to water and light. In Theoretical Production Ecology: reflection and prospects, eds. R. Rabbinge, J. Goudriaan, H. van Keulen, F. W. T. Penning de Vries & H. H. van Laar. Simulation Monographs, PUDOC, Wageningen, The Netherlands. pp. 3-16.
- Naylor, R. L. Battisti, D.S., Vimont, D.J., Falcon, W.P., Burke, M.B., 2007. Assessing risks of climate variability and climate change for Indonesian rice agriculture. PNAS 104-19, p. 7752-7757
- Penning de Vries, F.W.T., D.M. Jansen, H.F.M. ten Berge & A. Bakema, 1989. Simulation of ecophysiological processes of growth of several annual crops. Simulation Monographs 29, Pudoc, Wageningen, 271 pp.
- Rappoldt C., Van Kraalingen D.W.G. (1996) The FORTRAN Simulation Translator. FST version 2.0. Introduction and Reference Manual, in: B. A. M. Bouman and M. K. Van Ittersum (Eds.), Quantitative Approaches in Systems Analysis No. 5, Wageningen University and Research Center, Wageningen. pp. 178.
- Russell, G., Jarvis, P. G. & Monteith, J. L. (1989). Absorption of solar radiation and stand growth. In Plant canopies, their growth form and function. eds G. Russell, P. G. Jarvis & B. Marshall. Cambridge University Press. Cambridge. UK. pp. 21-39.
- Shibu, M.E., Leffelaar, P.A., van Keulen, H., Aggarwal, P.K., 2010. LINTUL3, a simulation model for nitrogen-limited situations: application to rice. European Journal of Agronomy 32, 255–271.
- Schapendonk, A.H.C.M., Stol, W., Van Kraalingen, D.W.G., Bouman, B.A.M., 1998. LINGRA, a sink/source model to simulate grassland productivity in Europe. Eur. J. Agron. 9, 87–100.
- Spitters C.J.T., Van Keulen H., Van Kraalingen, D.W.G., 1989. A simple and universal crop growth simulator:SUCROS87, in: Rabbinge R., Ward S.A., van Laar H.H.(Eds.), Simulation and Systems Management in Crop Protection, III, Simulation Monographs 32 (1) (1989)47-177.
- Spitters, C.J.T., 1990. Crop growth models: their usefulness and limitations. Acta Hort. 267, 349–368.
- Spitters, C.J.T., Schapendonk, A.H.C.M., 1990. Evaluation of breeding strategies for drought tolerance in potato by means of crop growth simulation. Plant Soil 123, 193–203.
- Stroosnijder, L., 1982. Simulation of the soil water balance. In: Eds F.W.T. Penning de Vries & H.H. van Laar, Simulation of plant growth and crop production. Simulation Monographs, Pudoc, Wageningen, pp. 175-193.
- Van Rossum, G., 1995. Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.
- Wesseling, C.G., Karssenberg, D.-J., Burrough, P.A., Van Deursen, W.P.A., 1996. Integrated dynamic environmental models inGIS: the development of a Dynamic Modelling language. Transactions in GIS, 11, 4048.

wflow_lintul module documentation

1.6 The framework and settings for the framework

1.6.1 Using the .ini file

A number of settings of the framework can be set in the ini file for each model. The settings are explained in the section below.

Settings in the run section

Information for the current run can be given in the run section. Here the start and end-time of the run as well as the timestep can be given. Alternatively a link to a Delft-FEWS runinfo.xml file can be given. An example is shown below.

```
[run]
#either a runinfo file or a start and end-time are required
#runinfofile=runinfo.xml
starttime= 1995-01-31 00:00:00
endtime= 1995-02-28 00:00:00
# required, base timestep of the model
timestepsecs = 86400
#Indicates the model is running from Delft-FEWS or not
```

If this section is not present and a runinfo.xml is also not used you will need to specify the number of timesteps using the -T option on the command line (for most models).

Settings in the framework section

The in/output file formats can be specified in the framework section. At present only pcraster mapstacks and netcdf are available fot input. See the supplied pcr2netcdf.py script for information on the layout of the netcdf files. If netcdf files are used the name of the mapstack is used as the standardname in the netcdf file.

```
[framework]
# outputformat for the *dynamic* mapstacks (not the states and summary maps)
# 1: pcraster
# 2: numpy
# 3: matlab
outputformat=1

# netcdfoutput requires also outputformat = 1 (default) and additionally the name of the␣
↪file
# read the mapsstacks from a netcdf
netcdfinput= inmaps.nc
#Write to netcdf:
netcdfoutput = outmaps.nc
# Write summary maps to netcdf
netcdfstaticoutput = staticoutmaps.nc
# Write states to netcdf
netcdfstatesoutput = states.nc
#Read states from netcdf
```

(continues on next page)

(continued from previous page)

```
netcdfstatesinput = instates.nc
#netcdfwritebuffer=100
```

As can be seen from the example above a number of input/ouput streams can be switch on to work with netcdf files. These are:

- netcdfinput. Time dependant input. This does not work for climatology files at the moment
- netcdfoutput. Time dependant output.
- netcdfstaticoutput. Summary output at the end of a run, those that normally end up in the outsum directory
- netcdfstatesoutput. The model's state variables at the end of a run.
- netcdfstatesinput. The model's input state variables at the start of a run.

To enhance performance when writing netcdf files a netcdfwritebuffer can be set. The number indicates the number of timesteps to keep in memory before flusing the buffer. Setting the buffer to a large value may induce memory problems.

Settings in the API section

In the ini file example below several variables are configured to be available via the API. For most settings this only defines what the API will expose to the outside world. However, if you specify 0 (input) as a role for one of the forcing variables the `wf_readmap` function will no longer read maps from disk for that variable but will return the contents of that variable via the API.

The API section specifies variables that are exposed via the api. Use the following convention:

```
variable_name_in_model=variable_role,variable_unit
```

```
role: 0 = input (to the model)
      1 = is output (from the model)
      2 = input/output (state information)
      3 = model parameter
unit: 0 = mm/timestep
      1 = m^3/sec
      2 = m
      3 = degree Celcius
      4 = mm
      5 = -
```

Use role 0 for input maps to the model (forcing data that are normally read from disk) only, role 1 for outputs, role 2 for state variables and role 3 for model parameters. The units may be choose freely and be strings also.

example:

```
[API]
FreeWater=2,4
SoilMoisture=2,4
UpperZoneStorage=2,4
LowerZoneStorage=2,4
InterceptionStorage=2,4
SurfaceRunoff=2,m^3/sec
WaterLevel=2,2
DrySnow=2,4
```

(continues on next page)

(continued from previous page)

```
Percolation=1,0
ForecQ_qmec=0,1
PERC=3,5
FC=3,4
# Below are the forcing variables. By putting these here you MUST
# supply them via the API, if not these will default to 0.0
#P=0,0
PET=0,0
TEMP=0,3
```

Settings in the modelparameters section

Most of the time this section is not needed as this will mostly be configured in the python code by the model developer. However, in some case this section can be used alter the model for example force the model to read RootingDepth from an external data source. Not all models support this. You can check if the model you uses support this by looking for the wf_updateparameters() function in de model code.

The format of entries in this section is as follows:

```
name=stack,type,default,verbose,[lookupmap_1],[lookupmap_2],lookupmap_n]
```

- name - Name of the parameter (internal variable, without the self.)
- stack - Name of the mapstack (representation on disk or in mem) relative to case
- type - Type of parameter (default = static)
- default - Default value if map/tbl is not present
- **verbose - If set to 1 (True) the maps a log message will be generated is a default values is used**
  instead of a map
- lookupmap - map(s) [0 to n] to be used in the lookuptable in the case the type is tbl

Possible parameter types (the second option)are:

- staticmap: Read at startup from map
- statictbl: [deprecated] Read at startup from tbl, fallback to map (need Landuse, Soil and TopoId (subcatch) maps)!
- tbl: Read at startup from tbl.
- tblts: Lookup tables for each timestep, including initial section
- tblsparse: Lookup tables for each timestep, including initial section. Fills in missing timestep using previous timestep
- tblmonthlyclim: read a tbl file corresponding to the current month (12 maps in total)
- timeseries: read map for each timestep
- monthlyclim: read a map corresponding to the current month (12 maps in total)
- dailyclim: read a map corresponding to the current day of the year (366 maps in total)
- hourlyclim: [not implemented yet] read a map corresponding to the current hour of the day (24 maps in total)
- tss: read a tss file and link to lookupmap (only one allowed) a map using timeinputscalar

Example:

```
[modelparameters]
RootingDepth=monthlyclim/ROOTS,monthlyclim,75,0
# Force the model to read monthly climatology of P
Precipitation=inmaps/P,monthlyclim,0.0,1
```

Example:

```
[modelparameters]
RootingDepth=monthlyclim/ROOT,monthyclim,100,1
Sl=inmaps/clim/LCtoSpecificLeafStorage.tbl,tbl,0.5,1,inmaps/clim/LC.map
Kext=inmaps/clim/LCtoSpecificLeafStorage.tbl,tbl,0.5,1,inmaps/clim/LC.map
Swood=inmaps/clim/LCtoBranchTrunkStorage.tbl,tbl,0.5,1,inmaps/clim/LC.map
LAI=inmaps/clim/LAI,monthlyclim,1.0,1
```

Settings in the variable_change_timestep/once section

In the two sections "variable_change_timestep" and "variable_change_once" you can set operations on parameters and variable that are executed at the start of each timestep or once in the initialisation of the model respectively. What you specify here should be valid python code and include variable that exists in the model you are using. This only works if the actual model you are using includes the wf_multparameters() function. At the moment wflow_hbv, wflow_sbm, wflow_w3ra and wflow_routing include this. See below for a configuration example. Some models may also support this via the -P and -p command-line options.

```
[variable_change_timestep]
self.Precipitation = self.Precipitation * 1.2
# Mutiplies the precipitation input times 1.2 every timestep

[variable_change_once]
self.PathFrac = self.PathFrac * 1.1
# Increases the paved area fraction of the model by 10 %
```

Settings in the [rollingmean] section

The rollingmean section allows you to define a rolling mean for each variable in the model. This variable can be used by other applications (e.g. data assimilation) or you can report it as output. Example:

```
[rollingmean]
self.Surfacerunoff=12
```

The above will make a 12 timestep rollingmean and store this in the variable self.Surfacerunoff_mean_12

Settings in the summary_* sections

By adding variable in one or several of these sectiosn the framework will save these variables to disk (using the value at the end, sum, min, max or avg) at the end of a run.

the available sections are:

- summary - Saves the actual value of the variable
- summary_avg - Saves the average value over all timesteps of the variable
- summary_sum - Saves the sum over all timesteps of the variable

- summary_min - Saves the minimum value over all timesteps of the variable
- summary_max - Saves the maximum value over all timesteps of the variable

All maps are saved in the outsum directory of the current runid.

Example:

```
[summary]
self.MaxLeakage=MaxLeakage.map
# Save and average these per LU type

[summary_sum]
self.Precipitation=Sumprecip.map

[summary_max]
self.Precipitation=maxprecip.map

[summary_min]
self.Temperature=mintemp.map

[summary_avg]
self.Precipitation=avgprecip.map
```

Settings in the outputtss/outputcsv sections

[outputcsv_0-n] [outputtss_0-n]

Number of sections to define output timeseries in csv format. Each section should at least contain one samplemap item and one or more variables to save. The samplemap is the map that determines how the timeseries are averaged/sampled. The function key specifies how the data is sample: average(default), minimum, maximum, total, majority. The timeformat key can either be steps or datetime.

All other items are variable=filename pairs. The filename is given relative to the case directory.

Example:

```
[outputcsv_0]
samplemap=staticmaps/wflow_subcatch.map
self.SurfaceRunoffMM=Qsubcatch_avg.csv
function=average
# average is the default
timeformat = datetime
# steps is the default

[outputcsv_1]
samplemap=staticmaps/wflow_gauges.map
self.SurfaceRunoffMM=Qgauge.csv
self.WaterLevel=Hgauge.csv

[outputtss_0]
samplemap=staticmaps/wflow_landuse.map
self.SurfaceRunoffMM=Qlu.tss
function=total
```

In the above example the discharge of this model (self.SurfaceRunoffMM) is saved as an average per subcatchment, a sample at the gauge locations and as an average per landuse.

[run] section: The use of date and time

Available options in the [run] section

The run section can contain information about the model timesteps, the date/time range, how to initialize the model and how interpret the forcing data.

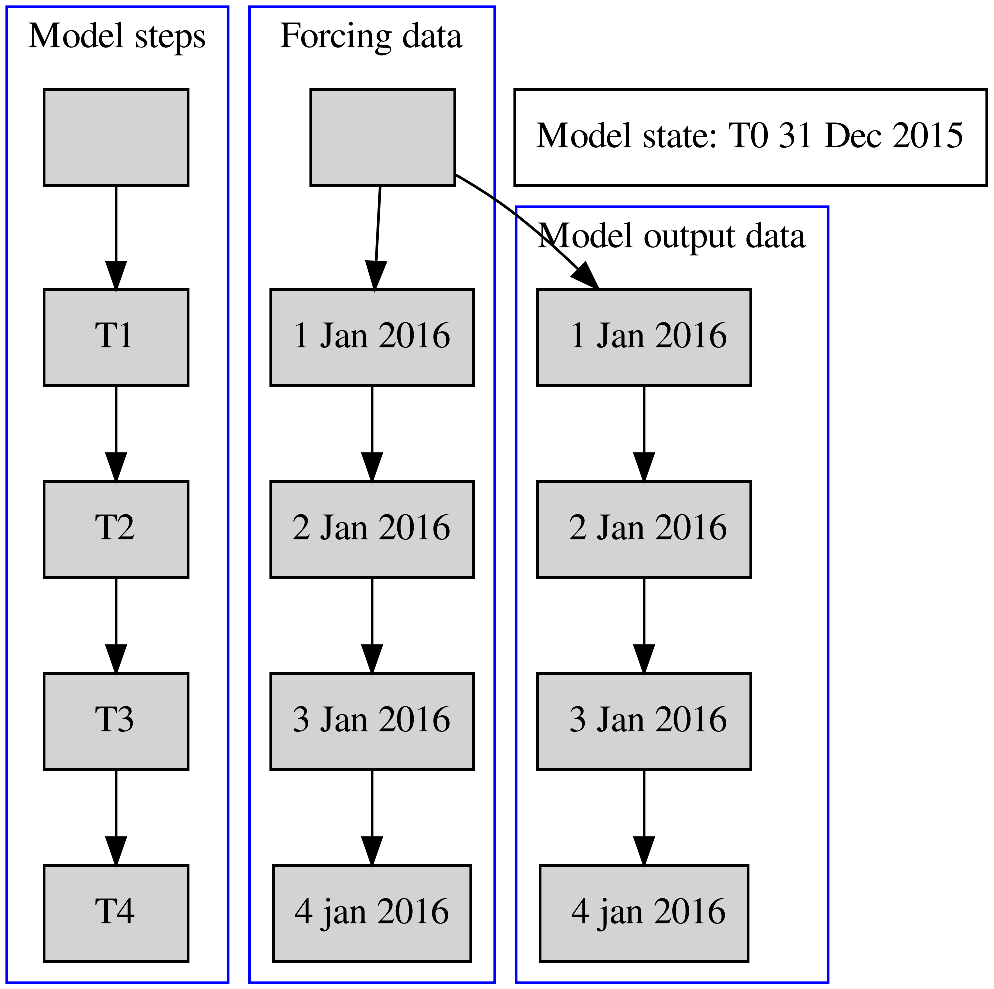
```
[run]
# either a runinfo file or a start and end-time are required
#runinfo=runinfo.xml
starttime=2016-01-01 00:00:00
endtime=2016-01-04 00:00:00
# Base timestep of the model in seconds, default = 86400
timestepsecs = 86400
#start model with cold state
reinit=1
# Default behaviour: steps
runlengthdetermination=steps
```

Date and time and timesteps

The original pcraster framework has no notion of date and time, only timesteps that are used to propagate a model forward. However, to be able to support the BMI and netcdf files date and time functionality has been inserted into the framework.

As most of the forcing in hydrological modelling is an accumulation the date/time of an input time series is assumed to represent the total (or average) of that variable one timestep length back in time from the timestamp.

For example, if the forcing data has the following four timestamps the model will run four timesteps. The first timesteps will be to propagate the state from T0 to T1 (in that case the date/time of the state files is assumed to be T0). As such the state going into the model should be valid for the T0, see graph below:



Here the first column shows the model steps and the second column the timestamp of the input/output data for that timestep (a empty box means there is nothing for that step). The first empty-row is regarded as the timestamp of the initial conditions. The first forcing data is used to propagate the model from T0 (the state, 31 Dec 2015) to T1 (1 Jan 2016) which will also be the first output the model writes.

The above corresponds to the following date/time in the [run] section:

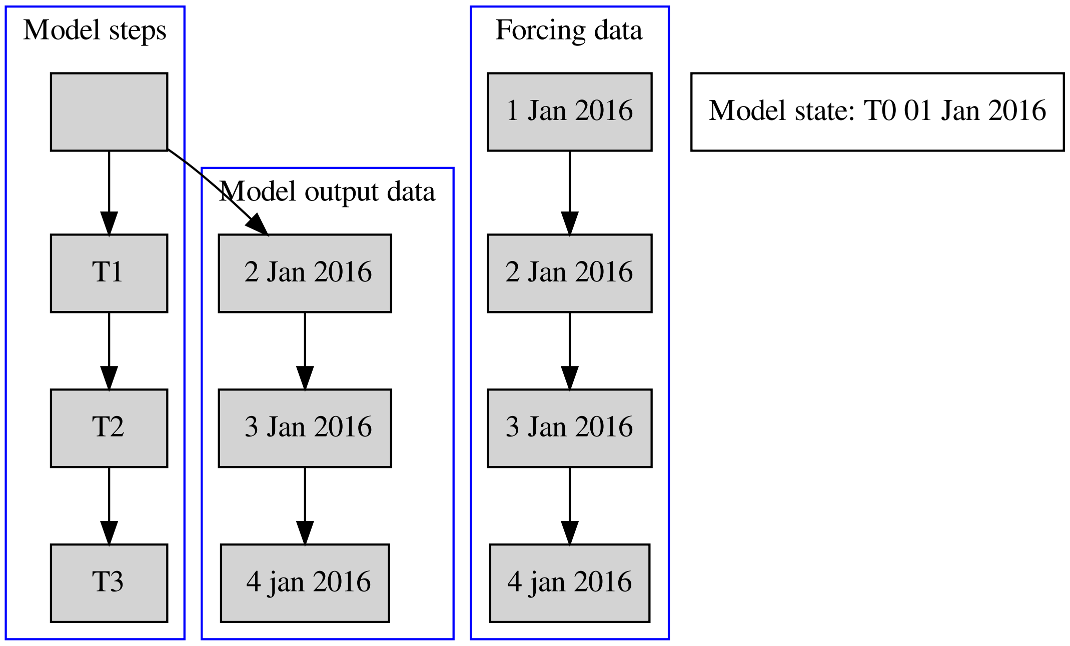
```
[run]
# either a runinfo file or a start and end-time are required
#runinfo=runinfo.xml
starttime=2016-01-01 00:00:00
endtime=2016-01-04 00:00:00
# required, base timestep of the model
timestepsecs = 86400
#start model with cold state
```

(continues on next page)

(continued from previous page)

```
reinit=1
# Default behaviour: steps
runlengthdetermination=steps
```

The above shows the default behaviour of the framework. For each data point in the input forcing a model steps is performed. The ‘runlengthdetermination’ variable in the run section can also be set to ‘intervals’. In that case the number of steps is determined from the number of intervals in the forcing data. Hence, the following run will be performed:



```
[run]
# either a runinfo file or a start and end-time are required
#runinfo=runinfo.xml
starttime=2016-01-01 00:00:00
endtime=2016-01-04 00:00:00
# required, base timestep of the model
timestepsecs = 86400
#start model with cold state
reinit=1
# Default behaviour: steps
runlengthdetermination=intervals
```

In this case the forcing data has the same input as in the previous case (4 timestamps) but in this case the first timestamp is regarded as the time of the initial conditions. As such, one timestep less (now three in total) is performed and the forcing data from 01 Jan 2016 is NOT used as it is regarded as the initial state. The first output point will be 02 Jan 2016 generated from the forcing marked as 2 Jan 2016.

The same applies when the start and end time of the model run are supplied via the bmi interface.

Settings in the netcdfmetadata section

All items in this section are copied as global attributes into the netcdf output file. Example:

```
[netcdfmetadata]
license=https://opendatacommons.org/licenses/odbl/
note=Test runs, results are not final
```

1.6.2 wf_DynamicFramework Module

1.7 The wflow Delft-FEWS adapter

1.7.1 wflow_adapt Module

Introduction

wflow_adapt is an adapter that links wflow to Delft-FEWS (http://publicwiki.deltares.nl/display/FEWSDOC/Home). it is typically run from the Delft-FEWS general adapter.

Linking wflow models to Delft-FEWS

To run the model from Delft-FEWS the following actions need to be performed:

- The runinfo.xml file should be specified in the [run]section of the ini file
- The use of netcdf input and output should be switched on
- The postadapter (wflow_adapt.py) needs to be run after the wflow run

The postadapter also converts the log messages of the model into Delft-FEWS diagnostics XML format.

- Casenamerunidwflow.log is converted to wflow_diag.xml
- Also the adapter log files is converted to wflow_adapt_diag.xml

Command line arguments:

An example of executing wflow from the Delft-FEWS general adapter is shown below:

```
<executeActivities>
<executeActivity>
<description>Run wflow</description>
<command><executable>bin-wflow\wflow_sbm.exe</executable></command>
    <arguments>
    <argument>-C</argument>
    <argument>rhine</argument>
    <argument>-f</argument>
</arguments>
<timeOut>7200000</timeOut>
</executeActivity>
<executeActivity>
```

(continues on next page)

(continued from previous page)

```
<description>Run wflow post</description>
<command> <executable>bin-wflow\wflow_adapt.exe</executable> </command> <arguments>
    <argument>-M</argument>
    <argument>Post</argument>
    <argument>-s</argument>
    <argument>rhine/instate/state.xml</argument>
    <argument>-o</argument>
    <argument>rhine/instate/outstate.xml</argument>
    <argument>-w</argument>
    <argument>./</argument>
    <argument>-C</argument>
    <argument>rhine</argument>
    <argument>-I</argument>
    <argument>wflow_sbm.ini</argument>
</arguments>
<timeOut>1200000</timeOut>
<overrulingDiagnosticFile>wflow_diag.xml</overrulingDiagnosticFile>
</executeActivity>
</executeActivities>
```

The wflow_adapt module can also be used by other programs to convert .tss files to pi-xml vv. Below the API documentation of the module is given.

In the above example the state files belonging to the model should be configed as per below in the General Adapter XML. In Fews the read and write locations are as viewed from the model's point of view:

```
<stateLocation>
    <readLocation>WaterLevel.map</readLocation>
    <writeLocation>../run_default/outstate/WaterLevel.map</writeLocation>
</stateLocation>
# Repeat for all state variables
```

Module function documentation

wflow_adapt.py: Simple wflow Delft-FEWS adapter in python. This file can be run as a script from the command-line or be used as a module that provides (limited) functionality for converting PI-XML files to .tss and back.

Usage pre adapter:

wflow_adapt -M Pre -t InputTimeseriesXml -I inifile

Usage postadapter:

wflow_adapt-M Post -t InputTimeseriesXml -s inputStateFile -I inifile
-o outputStateFile -r runinfofile -w workdir -C case [-R runId]

Issues:

- Delft-Fews exports data from 0 to timestep. PCraster starts to count at 1. Renaming the files is not desireable. The solution is the add a delay of 1 timestep in the GA run that exports the mapstacks to wflow.
- Not tested very well.
- There is a considerable amount of duplication (e.g. info in the runinfo.xml and the .ini file that you need to specify again :-())

Todo: rewrite and simplify

$Author: schelle $ $Id: wflow_adapt.py 915 2014-02-10 07:33:56Z schelle $ $Rev: 915 $

`wflow_adapt.`**`getEndTimefromRuninfo`**`(`*xmlfile*`)`

Gets the endtime of the run from the FEWS runinfo file

`wflow_adapt.`**`getMapStacksFromRuninfo`**`(`*xmlfile*`)`

Gets the list of mapstacks fews expect from the runinfo file and create those

`wflow_adapt.`**`getStartTimefromRuninfo`**`(`*xmlfile*`)`

Gets the starttime from the FEWS runinfo file

`wflow_adapt.`**`getTimeStepsfromRuninfo`**`(`*xmlfile*, *timestepsecs*`)`

Gets the number of timesteps from the FEWS runinfo file.

`wflow_adapt.`**`log2xml`**`(`*logfile*, *xmldiag*`)`

Converts a wflow log file to a Delft-Fews XML diag file

`wflow_adapt.`**`main`**`()`

Main entry for using the module as a command line program (e.g. from the Delft-FEWS GA)

`wflow_adapt.`**`mapstackxml`**`(`*mapstackxml*, *mapstackname*, *locationname*, *parametername*, *Sdate*, *Edate*, *timestepsecs*`)`

writes a mapstack xml file

`wflow_adapt.`**`pixml_state_updateTime`**`(`*inxml*, *outxml*, *DT*`)`

Reads the pi-state xml file inxml and updates the data/time of the state using datetime. Writes updated file to outxml

- Can be use in scripts to set the date.time of the output state.xml that Delft-FEWS writes.

Warning:

This function does not fully parse the xml file and will only work properly

if the xml files date the dateTime element written on one line.

`wflow_adapt.`**`pixml_totss`**`(`*nname*, *outputdir*`)`

Converts and PI xml timeseries file to a number of tss files.

The tss files are created using the following rules:

- tss filename determined by the content of the parameter element with a ".tss" postfix
- files are created in "outputdir"
- multiple locations will be multiple columns in the tss file written in order of appearance in the XML file

`wflow_adapt.`**`pixml_totss_dates`**`(`*nname*, *outputdir*`)`

Gets Date/time info from XML file and creates .tss files with:

- Day of year
- Hour of day
- Others may follow

`wflow_adapt.setlogger`(*logfilename*, *loggername*, *thelevel=20*)

Set-up the logging system and return a logger object. Exit if this fails

`wflow_adapt.tss_topixml`(*tssfile*, *xmlfile*, *locationname*, *parametername*, *Sdate*, *timestep*)

Converts a .tss file to a PI-xml file

1.8 Wflow modules and libraries

1.8.1 The wflow_fit module

Introduction

The wflow_fit module provides simple automated least square fitting for the wflow models. It uses the scipy.optimize function to perform the fitting.

The program works mu multipling the fit parameter with a factor and optimise this factor. To get the new optimised parameters for your model you have to multiply your original parameters with the optimised factor. You can specify measured and simulated Q pairs to use and which area of the model you wan to adjust for each Simulated/Measured pair

In order to use the fit module you must have a:

- A working wflow model
- a tss file with measured discharge
- an [fit] section in the ini file

The ini file

To be able to use the fit module you must add a [fit] section to the .ini file of the wflow model you want to fit.

```
[fit]
  # The parameters are name parameter_0 to parameter_n
parameter_0 = M
parameter_1 = RootingDepth
  # Q specifies the tss file with measure discharge data
  # the path is relative to the case directory
Q = testing.tss
  # The columns in the measured Q you want to fit to
ColMeas = [1,5]
  # The columns in the measured Q you want to fit
ColSim = [1,5]
  # Number of warup timesteps. This are not used in fitting
WarmUpSteps = 1
  # The map defining the areas you want to adjust
areamap=staticmaps/wflow_catchment.map
  # The areas you want to adjust for each Qmeas/Qsim combination
areacode=[1,5]
```

Fitting results

Results are saved in the wflow_fit.res file in the case/runid directory. In addition, the program saves a graph of modelled and observed data in the file fit.png and maps of the original and fitted parameters are also saved.

If you specify the -U option the resulting maps are saved in the staticmaps directory after each steps. As such, next steps (if you calibrate multiple subcatchments/areas) also include the results of the previous steps. Note that this will overwrite your maps if you already have those!

How to fit

Although wflow_sbm has a fairly large number of parameters most should not be fitted automatically. The parameters that are most suited for fitting are:

- M
- FirstZoneKsatVer
- RunoffGeneratingGWPerc (if this is switched on. It is usually best to first setup the model without this parameter!)
- RootingDepth

It is recommended to only fit one or two parameters at one time.

The wflow_rhine_sbm example can be used to test the fitting procedure.

```
wflow_fit.py -M wflow\_sbm -T 300 -C wflow\rhine\_sbm
```

Description of the python module

Fit a wflow_ hydrological model using scipy.leastsq.

usage

```
wflow_fit -M ModelName [-h][-F runinfofile][-C casename]
      [-c configfile][-T last_step][-S first_step][-s seconds]


-M: model to fit (e.g. wflow_sbm, wflow_hbv, wflow_cqf)

-T: Set last timestep

-S: Set the start timestep (default = 1)

-C: set the name  of the case (directory) to run

-R: set the name runId within the current case

-U: save the map after each step ti the input (staticmaps) dir so
    that next steps (colums) use the previous results

-c: name of wflow the configuration file (default: Casename/wflow_sbm.ini).

-h: print usage information
```

For this program to work you must add a [fit] section to the ini file of the program to fit (e.g. the wflow_hbv program)

$Author: schelle $ $Id: wflow_sbm.py 669 2013-05-16 05:25:48Z schelle $ $Rev: 669 $

`wflow_fit.`**`configget`**(*config*, *section*, *var*, *default*)

gets parameter from config file and returns a default value if the parameter is not found

`class` `wflow_fit.`**`wfmodel_fit_API`**(*startTime*, *stopTime*, *casename*, *runId='_fitrun'*, *modeltofit='wflow_sbm'*, *config='wflow_sbm.ini'*, *clonemap='wflow_subcatch.map'*)

Class that initializes and runs a wflow model

`multVarWithPar`(*pars*)

Multiply a parameter in the model with the fit parameters. Use a map to limit the area to adjust

`run`(*pars*)

Run the model for the number of timesteps.

`savemaps`(*pars*, *savetoinput=False*)

Ssave the adjusted (and original) parameter maps

`shutdown`(*pars*)

Shutdown the model

1.8.2 wflow_lib Module

wflow_lib - terrain analysis and hydrological library

The goal of this module is to make a series functions to upscale maps (DEM) and to maintain as much of the information in a detailled dem when upscaling to a coarser DEM. These include:

- river length (per cell)
- river network location
- elevation distribution
- other terrain analysis

the wflow_prepare scripts use this library extensively.

$Author: schelle $ $Id: wflow_lib.py 808 2013-10-04 19:42:43Z schelle $ $Rev: 808 $

`wflow_lib.`**`Gzip`**(*fileName*, *storePath=False*, *chunkSize=1048576*)

Usage: Gzip(fileName, storePath=False, chunksize=1024*1024) Gzip the given file to the given storePath and then remove the file. A chunk size may be selected. Default is 1 megabyte Input:

fileName: file to be GZipped storePath: destination folder. Default is False, meaning the file will be zipped to its own folder chunkSize: size of chunks to write. If set too large, GZip will fail with memory problems

`wflow_lib.`**`area_percentile`**(*inmap*, *area*, *n*, *order*, *percentile*)

calculates percentile of inmap per area n is the number of points in each area, order, the sorter order of inmap per area (output of areaorder(inmap,area)) n is the output of pcr.areatotal(pcr.spatial(pcr.scalar(1.0)),area)

Input:

- inmap
- area map
- n

- order (riverorder)
- percentile

Output:

- percentile map

`wflow_lib.`**`area_river_burnin`**`(`*ldd*, *dem*, *order*, *Area*`)`

Calculates the lowest values in as DEM for each erea in an area map for river of order *order*

Input:

- ldd
- dem
- order
- Area map

Output:

- dem

`wflow_lib.`**`area_riverlength_factor`**`(`*ldd*, *Area*, *Clength*`)`

ceates correction factors for riverlength for the largest streamorder in each area

Input:

- ldd
- Area
- Clength (1d length of a cell (pcr.sqrt(Area))

Output:

- distance per area

`wflow_lib.`**`areastat`**`(`*Var*, *Area*`)`

Calculate several statistics of *Var* for each unique id in *Area*

Input:

- Var
- Area

Output:

- Standard_Deviation,Average,Max,Min

`wflow_lib.`**`checkerboard`**`(`*mapin*, *fcc*`)`

checkerboard create a checkerboard map with unique id's in a fcc*fcc cells area. The resulting map can be used to derive statistics for (later) upscaling of maps (using the fcc factor)

Input:

- map (used to determine coordinates)
- fcc (size of the areas in cells)

Output:

- checkerboard type map

`wflow_lib.`**`classify`**(*inmap*, *lower=[0, 10, 20, 30]*, *upper=[10, 20, 30, 40]*, *classes=[2, 2, 3, 4]*)

classify a scaler maps accroding to the boundaries given in classes.

`wflow_lib.`**`configget`**(*config*, *section*, *var*, *default*)

Gets a string from a config file (.ini) and returns a default value if the key is not found. If the key is not found it also sets the value with the default in the config-file

Input:

- config - python ConfigParser object
- section - section in the file
- var - variable (key) to get
- default - default string

Returns:

- string - either the value from the config file or the default value

`wflow_lib.`**`configsection`**(*config*, *section*)

gets the list of keys in a section

Input:

- config
- section

Output:

- list of keys in the section

`wflow_lib.`**`configset`**(*config*, *section*, *var*, *value*, *overwrite=False*)

Sets a string in the in memory representation of the config object Deos NOT overwrite existing values if overwrite is set to False (default)

Input:

- config - python ConfigParser object
- section - section in the file
- var - variable (key) to set
- value - the value to set
- overwrite (optional, default is False)

Returns:

- nothing

`wflow_lib.`**`cutMapById`**(*data*, *subcatchmap*, *id*, *x*, *y*, *FillVal*)

Parameters

- **`data`** – 2d numpy array to cut
- **`subcatchmap`** – 2d numpy array with subcatch
- **`id`** – id (value in the array) to cut by
- **`x`** – array with x values
- **`y`** – array with y values

Returns
x,y, data

wflow_lib.**derive_HAND**(*dem*, *ldd*, *accuThreshold*, *rivers=None*, *basin=None*)

Function derives Height-Above-Nearest-Drain. See http://www.sciencedirect.com/science/article/pii/S003442570800120X Input:

dem – pcraster object float32, elevation data ldd – pcraster object direction, local drain directions accuThreshold – upstream amount of cells as threshold for river

delineation

rivers=None – you can provide a rivers layer here. Pixels that are
identified as river should have a value > 0, other pixels a value of zero.

basin=None – set a boolean pcraster map where areas with True are estimated using the nearest drain in ldd distance
and areas with False by means of the nearest friction distance. Friction distance estimated using the upstream area as weight (i.e. drains with a bigger upstream area have a lower friction) the spreadzone operator is used in this case.

Output:
hand – pcraster bject float32, height, normalised to nearest stream dist – distance to nearest stream measured in cell lengths

according to D8 directions

wflow_lib.**detdrainlength**(*ldd*, *xl*, *yl*)

Determines the drainaige length (DCL) for a non square grid

Input:

- ldd - drainage network
- xl - length of cells in x direction
- yl - length of cells in y direction

Output:

- DCL

wflow_lib.**detdrainwidth**(*ldd*, *xl*, *yl*)

Determines width of drainage over DEM for a non square grid

Input:

- ldd - drainage network
- xl - length of cells in x direction
- yl - length of cells in y direction

Output:

- DCL

wflow_lib.**find_outlet**(*ldd*)

Tries to find the outlet of the largest catchment in the Ldd

Input:

- Ldd

Output:

- outlet map (single point in the map)

`wflow_lib.`**`getRowColPoint`**`(`*in_map*, *xcor*, *ycor*`)`

returns the row and col in a map at the point given. Works but is rather slow.

Input:

- in_map - map to determine coordinates from
- xcor - x coordinate
- ycor - y coordinate

Output:

- row, column

`wflow_lib.`**`getValAtPoint`**`(`*in_map*, *xcor*, *ycor*`)`

returns the value in a map at the point given. works but is rather slow.

Input:

- in_map - map to determine coordinates from
- xcor - x coordinate
- ycor - y coordinate

Output:

- value

`wflow_lib.`**`getcols`**`()`

returns the number of columns in the current map

Input:

- –

Output:

- nr of columns in the current clonemap as a scalar

`wflow_lib.`**`getgridparams`**`()`

return grid parameters in a python friendly way

Output:

[Xul, Yul, xsize, ysize, rows, cols]

- xul - x upper left centre
- yul - y upper left centre
- xsize - size of a cell in x direction
- ysize - size of a cell in y direction
- cols - number of columns
- rows - number of rows
- xlr - x lower right centre
- ylr - y lower right centre

`wflow_lib.`**`getrows`**`()`

returns the number of rows in the current map

Input:

- –

Output:

- nr of rows in the current clonemap as a scalar

`wflow_lib.`**`idtoid`**`(`*sourceidmap*, *targetidmap*, *valuemap*`)`

tranfer the values from valuemap at the point id's in sourceidmap to the areas in targetidmap.

Parameters

- **`pointmap`** –
- **`areamap`** –
- **`valuemap`** –

Returns

`wflow_lib.`**`lddcreate_save`**`(`*lddname*, *dem*, *force*, *corevolume=1e+35*, *catchmentprecipitation=1e+35*, *corearea=1e+35*, *outflowdepth=1e+35*`)`

Creates an ldd if a file does not exists or if the force flag is used

input:

- lddname (name of the ldd to create)
- dem (actual dem)
- force (boolean to force recreation of the ldd)
- outflowdepth (set to 10.0E35 normally but smaller if needed)

Output:

- the LDD

`wflow_lib.`**`points_to_map`**`(`*in_map*, *xcor*, *ycor*, *tolerance*`)`

Returns a map with non zero values at the points defined in X, Y pairs. It's goal is to replace the pcraster col2map program.

tolerance should be 0.5 to select single points Performance is not very good and scales linear with the number of points

Input:

- in_map - map to determine coordinates from
- xcor - x coordinate (array or single value)
- ycor - y coordinate (array or single value)
- tolerance - tolerance in cell units. 0.5 selects a single cell 10 would select a 10x10 block of cells

Output:

- Map with values burned in. 1 for first point, 2 for second and so on

`wflow_lib.`**`pt_flow_in_river`**`(`*ldd*, *river*`)`

Returns all points (True) that flow into the mak river (boolean map with river set to True)

Parameters

- `ldd` – Drainage network
- `river` – Map of river (True River, False non-river)

Return ifmap
map with infrlo points into the river (True)

Return ctach
catchment of each of the inflow points

`wflow_lib.`**`readMap`**`(`*fileName*, *fileFormat*`)`

Read geographical file into memory

Parameters

- `fileName` –
- `fileFormat` –

Return x, y, data, FillVal

`wflow_lib.`**`riverlength`**`(`*ldd*, *order*`)`

Determines the length of a river using the ldd. only determined for order and higher.

Input:

- ldd, order (streamorder)

Returns:

- totallength,lengthpercell, streamorder

`wflow_lib.`**`sCurve`**`(`*X*, *a=0.0*, *b=1.0*, *c=1.0*`)`

sCurve function:

Input:

- X input map
- C determines the steepness or "stepwiseness" of the curve. The higher C the sharper the function. A negative C reverses the function.
- b determines the amplitude of the curve
- a determines the centre level (default = 0)

Output:

- result

`wflow_lib.`**`sCurveSlope`**`(`*X*, *a=0.0*, *b=1.0*, *c=1.0*`)`

First derivative of the sCurve defined by a,b,c at point X

Input:

- X - value to calculate for
- a
- b
- c

Output:

- first derivative (slope) of the curve at point X `wflow_lib.`**`snaptomap`**`(`*points*, *mmap*`)`

Snap the points in _points_ to nearest non missing values in _mmap_. Can be used to move gauge locations to the nearest rivers.

Input:

- points - map with points to move
- mmap - map with points to move to

Return:

- map with shifted points

`wflow_lib.`**`subcatch`**`(`*ldd*, *outlet*`)`

Determines a subcatchment map using LDD and outlet(s). In the resulting subcatchment map the i's of the catchment are determiend by the id's of the outlets.

Input:

- ldd
- Outlet - maps with points for each outlet.

Output:

- map of subcatchments

`wflow_lib.`**`subcatch_order_a`**`(`*ldd*, *oorder*`)`

Determines subcatchments using the catchment order

This version uses the last cell BELOW order to derive the catchments. In general you want the _b version

Input:

- ldd
- order - order to use

Output:

- map with catchment for the given streamorder

`wflow_lib.`**`subcatch_order_b`**`(`*ldd*, *oorder*, *sizelimit=0*, *fill=False*, *fillcomplete=False*, *stoporder=0*`)`

Determines subcatchments using the catchment order

This version tries to keep the number op upstream/downstream catchment the small by first dederivingatchment connected to the major river(the order) given, and fill up from there.

Input:

- ldd
- oorder - order to use
- sizelimit - smallest catchments to include, default is all (sizelimit=0) in number of cells
- if fill is set to True the higer order catchment are filled also
- if fillcomplete is set to True the whole ldd is filled with catchments.

:returns sc, dif, nldd; Subcatchment, Points, subcatchldd `wflow_lib.`**`subcatch_stream`**(*ldd*, *threshold*, *min_strahler=- 999*, *max_strahler=999*, *assign_edge=False*, *assign_existing=False*, *up_area=None*)

(From Deltares Hydrotools)

Derive catchments based upon strahler threshold Input:

ldd – pcraster object direction, local drain directions threshold – integer, strahler threshold, subcatchments ge threshold

are derived

min_strahler – integer, minimum strahler threshold of river catchments
to return

max_strahler – integer, maximum strahler threshold of river catchments
to return

assign_unique=False – if set to True, unassigned connected areas at
the edges of the domain are assigned a unique id as well. If set to False, edges are not assigned

assign_existing=False == if set to True, unassigned edges are assigned
to existing basins with an upstream weighting. If set to False, edges are assigned to unique IDs, or not assigned

output:
stream_ge – pcraster object, streams of strahler order ge threshold subcatch – pcraster object, subcatchments of strahler order ge threshold

`wflow_lib.`**`sum_list_cover`**(*list_of_maps*, *covermap*)

Sums a list of pcrastermap using cover to fill in missing values

Parameters

- **`list_of_maps`** – list of maps to sum
- **`covermap`** – maps/ value to use fro cover

Returns
sum of list of maps (single map)

`wflow_lib.`**`upscale_riverlength`**(*ldd*, *order*, *factor*)

Upscales the riverlength using 'factor' The resulting maps can be resampled (e.g. using resample.exe) by factor and should include the accurate length as determined with the original higher resolution maps. This function is **depricated**, use are_riverlength instead as this version is very slow for large maps

Input:

- ldd
- minimum streamorder to include

Output:

- distance per factor cells

`wflow_lib.`**`writeMap`**(*fileName*, *fileFormat*, *x*, *y*, *data*, *FillVal*)

Write geographical data into file

`wflow_lib.`**`zipFiles`**(*fileList*, *fileTarget*)

Usage: zipFiles(fileList, fileTarget) zip the given list of files to the given target file Input:

fileList: list of files to be zipped fileTarget: target zip-file

1.8.3 wflow_funcs Module

Introduction

wflow_funcs is a library of hydrological modules that can be used by any of the wflow models. It includes modules related to:

- the kinematic wave routing for surface and subsurface flow
- rainfall interception by the vegetation
- snow and glaciers modelling
- reservoirs and lakes modelling.

Kinematic Wave

Surface flow routing

The main flow routing scheme used by the wflow models (wflow_sbm and wflow_hbv) is the kinematic wave approach for channel and overland flow, assuming that the topography controls water flow mostly. The kinemative wave equations are (Chow, 1988): $\frac{dQ}{dx} + \frac{dA}{dt} = q$ and $A = \alpha * Q^{\beta}$. These equations can then be combined as a function of streamflow only:

$$\frac{dQ}{dx} + \alpha * \beta * Q^{\beta-1} * \frac{dQ}{dt} = q$$

where Q is the surface runoff in the kinematic wave [m^3/s], x is the length of the runoff pathway [m], A is the cross-section area of the runoff pathway [m^2], t is the integration timestep [s] and alpha and beta are coefficients.

These equations are solved with a nonlinear scheme using Newton's method and can also be iterated depending on the wflow models space and time resolutions. By default, the iterations are performed until a stable solution is reached (epsilon < 10^{-12}). For larger models, the number of iterations can also be fixed for wflow_sbm to a specific sub-timestep (in seconds) for both overland and channel flows to improve simulation time. To enable (fixed or not) iterations of the kinematic wave the following lines can be inserted in the ini files of the related models:

```
[model]
# Enable iterations of the kinematic wave
kinwaveIters = 1
# Fixed sub-timestep for iterations of channel flow (river cells)
kinwaveRiverTstep = 900
# Fixed sub-timestep for iterations of overland flow (land cells)
kinwaveLandTstep = 3600
```

For wflow_sbm and wflow_hbv Manning's N values for the river can be specified through a N_River.tbl file in two ways:

- N values are linked to land cover (col 1), sub catchment (col 2) and soil type (col 3) (default (NRiverMethod == 1))
- N values are linked to streamorder (col 1) (NRiverMethod == 2)

To link N values to streamorder insert the following in the ini file:

```
[model]
nrivermethod = 2
```

Subsurface flow routing

In wflow_sbm the kinematic wave approach is used to route subsurface flow laterally. The saturated store S can be drained laterally by saturated downslope subsurface flow per unit width of slope w [mm] according to:

$$q = \frac{K_0 tan(\beta)}{f}(e^{(-fz_i)} - e^{(-fz_t)})$$

where β is element slope angle [deg.], q is subsurface flow $[mm^2/t]$, K_0 is the saturated hydraulic conductivity at the soil surface [mm/t], z_i is the water table depth [mm], z_t is total soil depth [mm], and f is a scaling parameter $[mm^{-1}]$:

$$f = \frac{\theta_s - \theta_r}{M}$$

where θ_s is saturated water content [mm/mm] and θ_r is residual water content [mm/mm] and M represents a model parameter [mm], that determines the decrease of vertical saturated conductivity with depth.

Combining with the following continuity equation:

$$(\theta_s - \theta_r)\frac{\partial h}{\partial t} = -w\frac{\partial q}{\partial x} + wr$$

where h is the water table height [mm], x is the distance downslope [mm], and r is the netto input rate [mm/t] to the saturated store.

and substituting for $h(\frac{\partial q}{\partial h})$, gives:

$$w\frac{\partial q}{\partial t} = -cw\frac{\partial q}{\partial x} + cwr$$

where celerity $c = \frac{K_0 tan(\beta)}{(\theta_s - \theta_r)} e^{(-fz_i)}$

The kinematic wave equation for lateral subsurface flow is solved iteratively using Newton's method.

Note: For the lateral subsurface flow kinematic wave the model timestep is not adjusted. For certain model timestep and model grid size combinations this may result in loss of accuracy.

Rainfall Interception

Both the Gash and Rutter models are available in the wflow framework to estimate rainfall interception by the vegetation. The selection of an interception model depends on the simulation timestep. These modules are used by the wflow_sbm model.

The analytical (Gash) model

The analytical model of rainfall interception is based on Rutter's numerical model. The simplifications that introduced allow the model to be applied on a daily basis, although a storm-based approach will yield better results in situations with more than one storm per day. The amount of water needed to completely saturate the canopy is defined as:

$$P' = \frac{-\overline{R}S}{\overline{E}_w} ln\left[1 - \frac{\overline{E}_w}{\overline{R}}(1 - p - p_t)^{-1}\right]$$

where $\overline{R}$ is the average precipitation intensity on a saturated canopy and $\overline{E}_w$ the average evaporation from the wet canopy and with the vegetation parameters S, p and p_t as defined previously. The model uses a series of expressions to calculate the interception loss during different phases of a storm. An analytical integration of the total evaporation and rainfall under saturated canopy conditions is then done for each storm to determine average values of $\overline{E}_w$ and $\overline{R}$. The total evaporation from the canopy (the total interception loss) is calculated as the sum of the components listed in the table below. Interception losses from the stems are calculated for days with $P \geq S_t/p_t$. p_t and S_t are small and neglected in the wflow_sbm model.

Table: Formulation of the components of interception loss according to Gash:

For m small storms ($P_g < {P'}_g$)	$(1 - p - p_t) \sum_{j=1}^{m} P_{g,j}$
Wetting up the canopy in n large storms ($P_g \geq {P'}_g$)	$n(1 - p - p_t){P'}_g - nS$
Evaporation from saturated canopy during rainfall	$\overline{E}/\overline{R} \sum_{j=1}^{n}(P_{g,j} - {P'}_g)$
Evaporation after rainfall ceases for n large storms	nS
Evaporation from trunks in q storms that fill the trunk storage	qS_t
Evaporation from trunks in $(m + n - q)$ storms that do not fill the trunk storage	$p_t \sum_{j=1}^{m+n-q} P_{g,j}$

In applying the analytical model, saturated conditions are assumed to occur when the hourly rainfall exceeds a certain threshold. Often a threshold of 0.5 mm/hr is used. $\overline{R}$ is calculated for all hours when the rainfall exceeds the threshold to give an estimate of the mean rainfall rate onto a saturated canopy.

Gash (1979) has shown that in a regression of interception loss on rainfall (on a storm basis) the regression coefficient should equal to $\overline{E}_w/\overline{R}$. Assuming that neither $\overline{E}_w$ nor $\overline{R}$ vary considerably in time, $\overline{E}_w$ can be estimated in this way from $\overline{R}$ in the absence of above-canopy climatic observations. Values derived in this way generally tend to be (much) higher than those calculated with the penman-monteith equation.

Running with parameters derived from LAI

The model can determine the Gash parameters from an LAI maps. In order to switch this on you must define the LAI variable to the model (as in the example below).

```
[modelparameters]
LAI=inmaps/clim/LAI,monthlyclim,1.0,1
Sl=inmaps/clim/LCtoSpecificLeafStorage.tbl,tbl,0.5,1,inmaps/clim/LC.map
Kext=inmaps/clim/LCtoExtinctionCoefficient.tbl,tbl,0.5,1,inmaps/clim/LC.map
Swood=inmaps/clim/LCtoBranchTrunkStorage.tbl,tbl,0.5,1,inmaps/clim/LC.map
```

Here LAI refers to a MAP with LAI (in this case one per month), Sl to a lookuptable result of land cover to specific leaf storage, Kext to a lookuptable result of land cover to extinction coefficient and Swood to a lookuptable result of "canopy" capacity of the vegetation woody fraction.

Here it is assumed that Cmax(leaves) (Gash' canopy capacity for the leaves only) relates linearly with LAI (c.f. Van Dijk and Bruijnzeel 2001). This done via the Sl (specific leaf storage). Sl is determined via a lookup table with land cover. Next the Cmax(leaves) is determined using:

$$Cmax(leaves) = Sl * LAI$$

The table below shows lookup table for Sl (as determined from Pitman 1986, Lui 1998) and GlobCover land cover map.

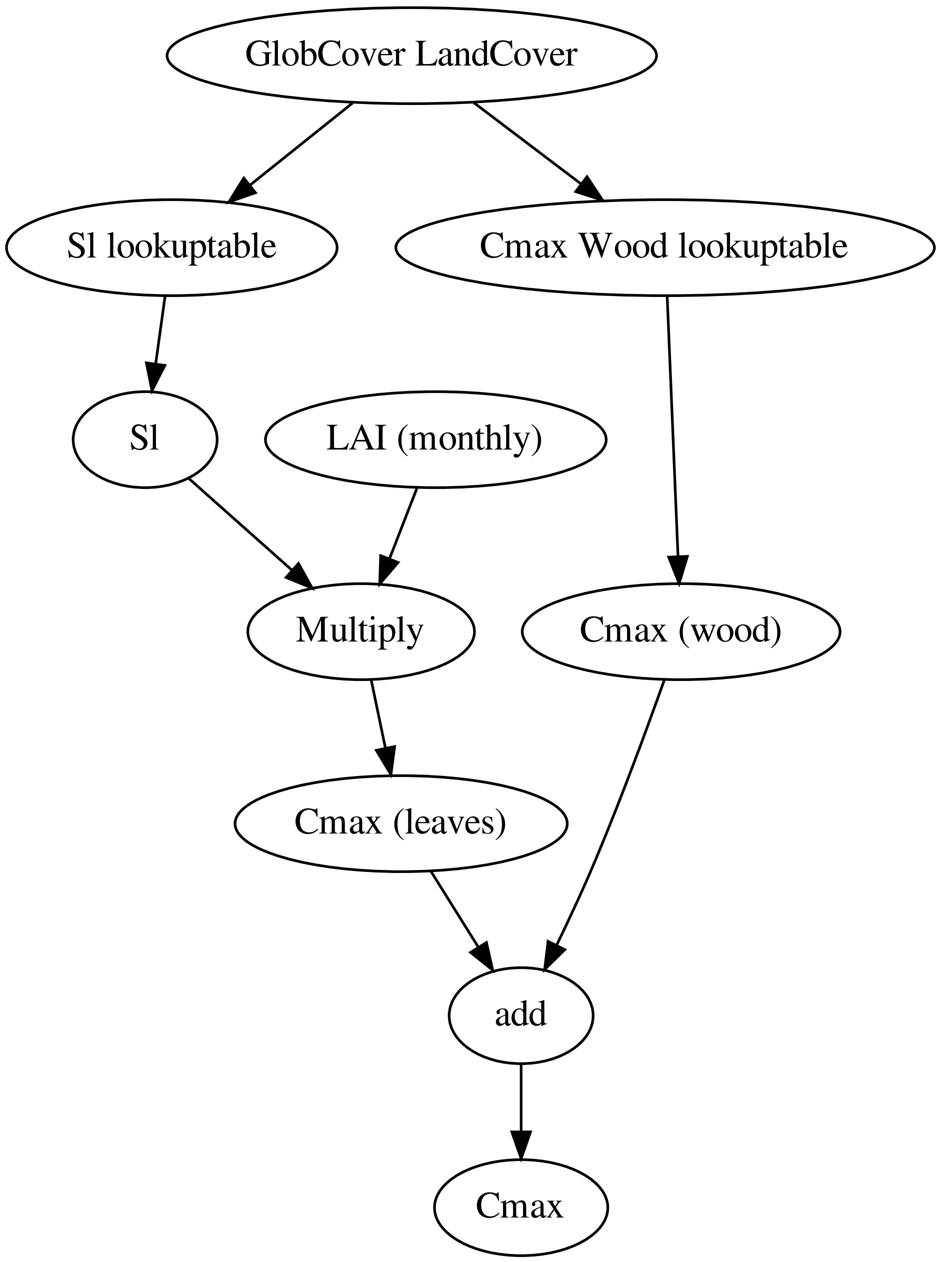
```
11   0.1272    Post-flooding or irrigated croplands (or aquatic)
14   0.1272    Rainfed croplands
20   0.1272    Mosaic cropland (50-70%) / vegetation (grassland/shrubland/forest) (20-50
↪%)
30   0.1272    Mosaic vegetation (grassland/shrubland/forest) (50-70%) / cropland (20-50
↪%)
```

(continues on next page)

(continued from previous page)

```
40   0.03926   Closed to open (>15%) broadleaved evergreen or semi-deciduous forest (>5m)
50   0.036     Closed (>40%) broadleaved deciduous forest (>5m)
60   0.036     Open (15-40%) broadleaved deciduous forest/woodland (>5m)
70   0.045     Closed (>40%) needleleaved evergreen forest (>5m)
90   0.045     Open (15-40%) needleleaved deciduous or evergreen forest (>5m)
100  0.03926   Closed to open (>15%) mixed broadleaved and needleleaved forest (>5m)
110  0.07      Mosaic forest or shrubland (50-70%) / grassland (20-50%)
120  0.1272    Mosaic grassland (50-70%) / forest or shrubland (20-50%)
130  0.07      Closed to open (>15%) (broadleaved or needleleaved, evergreen or␣
↪deciduous) shrubland (<5m)
140  0.09      Closed to open (>15%) herbaceous vegetation (grassland, savannas or␣
↪lichens/mosses)
150  0.04      Sparse (<15%) vegetation
160  0.04      Closed to open (>15%) broadleaved forest regularly flooded (semi-
↪permanently or temporarily) - Fresh or brackish water
170  0.036     Closed (>40%) broadleaved forest or shrubland permanently flooded -␣
↪Saline or brackish water
180  0.1272    Closed to open (>15%) grassland or woody vegetation on regularly flooded␣
↪or waterlogged soil - Fresh, brackish or saline water
190  0.04      Artificial surfaces and associated areas (Urban areas >50%)
200  0.04      Bare areas
210  0.04      Water bodies
220  0.04      Permanent snow and ice
230   -        No data (burnt areas, clouds,...)
```

To get to total storage (Cmax) the woody part of the vegetation also needs to be added. This is done via a simple lookup table between land cover the Cmax(wood):



The table below relates the land cover map to the woody part of the Cmax.

```
11   0.01    Post-flooding or irrigated croplands (or aquatic)
14   0.0     Rainfed croplands
20   0.01    Mosaic cropland (50-70%) / vegetation (grassland/shrubland/forest) (20-50%)
30   0.01    Mosaic vegetation (grassland/shrubland/forest) (50-70%) / cropland (20-50%)
40   0.5     Closed to open (>15%) broadleaved evergreen or semi-deciduous forest (>5m)
```

(continues on next page)

(continued from previous page)

```
50   0.5     Closed (>40%) broadleaved deciduous forest (>5m)
60   0.5     Open (15-40%) broadleaved deciduous forest/woodland (>5m)
70   0.5     Closed (>40%) needleleaved evergreen forest (>5m)
90   0.5     Open (15-40%) needleleaved deciduous or evergreen forest (>5m)
100  0.5     Closed to open (>15%) mixed broadleaved and needleleaved forest (>5m)
110  0.2     Mosaic forest or shrubland (50-70%) / grassland (20-50%)
120  0.05    Mosaic grassland (50-70%) / forest or shrubland (20-50%)
130  0.1     Closed to open (>15%) (broadleaved or needleleaved, evergreen or deciduous)␣
↪shrubland (<5m)
140  0.0     Closed to open (>15%) herbaceous vegetation (grassland, savannas or lichens/
↪mosses)
150  0.04    Sparse (<15%) vegetation
160  0.1     Closed to open (>15%) broadleaved forest regularly flooded (semi-
↪permanently or temporarily) - Fresh or brackish water
170  0.2     Closed (>40%) broadleaved forest or shrubland permanently flooded - Saline␣
↪or brackish water
180  0.01    Closed to open (>15%) grassland or woody vegetation on regularly flooded or␣
↪waterlogged soil - Fresh, brackish or saline water
190  0.01    Artificial surfaces and associated areas (Urban areas >50%)
200  0.0     Bare areas
210  0.0     Water bodies
220  0.0     Permanent snow and ice
230   -      No data (burnt areas, clouds,...)
```

The canopy gap fraction is determined using the k: extinction coefficient (van Dijk and Bruijnzeel 2001):

$$CanopyGapFraction = exp(-k * LAI)$$

The table below show how k is related to land cover:

```
11   0.6    Post-flooding or irrigated croplands (or aquatic)
14   0.6    Rainfed croplands
20   0.6    Mosaic cropland (50-70%) / vegetation (grassland/shrubland/forest) (20-50%)
30   0.6    Mosaic vegetation (grassland/shrubland/forest) (50-70%) / cropland (20-50%)
40   0.6    Closed to open (>15%) broadleaved evergreen or semi-deciduous forest (>5m)
50   0.8    Closed (>40%) broadleaved deciduous forest (>5m)
60   0.8    Open (15-40%) broadleaved deciduous forest/woodland (>5m)
70   0.8    Closed (>40%) needleleaved evergreen forest (>5m)
90   0.8    Open (15-40%) needleleaved deciduous or evergreen forest (>5m)
100  0.8    Closed to open (>15%) mixed broadleaved and needleleaved forest (>5m)
110  0.6    Mosaic forest or shrubland (50-70%) / grassland (20-50%)
120  0.6    Mosaic grassland (50-70%) / forest or shrubland (20-50%)
130  0.6    Closed to open (>15%) (broadleaved or needleleaved, evergreen or deciduous)␣
↪shrubland (<5m)
140  0.6    Closed to open (>15%) herbaceous vegetation (grassland, savannas or lichens/
↪mosses)
150  0.6    Sparse (<15%) vegetation
160  0.6    Closed to open (>15%) broadleaved forest regularly flooded (semi-permanently␣
↪or temporarily) - Fresh or brackish water
170  0.8    Closed (>40%) broadleaved forest or shrubland permanently flooded - Saline␣
↪or brackish water
180  0.6    Closed to open (>15%) grassland or woody vegetation on regularly flooded or␣
↪waterlogged soil - Fresh, brackish or saline water
```

(continues on next page)

(continued from previous page)

```
190  0.6    Artificial surfaces and associated areas (Urban areas >50%)
200  0.6    Bare areas
210  0.7    Water bodies
220  0.6    Permanent snow and ice
230   -     No data (burnt areas, clouds,...)d
```

The modified rutter model

For subdaily timesteps the model uses a simplification of the Rutter model. The simplified model is solved explicitly and does not take drainage from the canopy into account.

```
def rainfall_interception_modrut(Precipitation, PotEvap, CanopyStorage,␣
↪CanopyGapFraction, Cmax):
"""
Interception according to a modified Rutter model. The model is solved
explicitly and there is no drainage below Cmax.

Returns:
    - NetInterception: P - TF - SF (may be different from the actual wet canopy␣
↪evaporation)
    - ThroughFall:
    - StemFlow:
    - LeftOver: Amount of potential eveporation not used
    - Interception: Actual wet canopy evaporation in this thimestep
    - CanopyStorage: Canopy storage at the end of the timestep

"""

##########################################################################
# Interception according to a modified Rutter model with hourly timesteps#
##########################################################################

p = CanopyGapFraction
pt = 0.1 * p

# Amount of P that falls on the canopy
Pfrac = pcr.max((1 - p - pt), 0) * Precipitation

# S cannot be larger than Cmax, no gravity drainage below that
DD = pcr.ifthenelse(CanopyStorage > Cmax, CanopyStorage - Cmax, 0.0)
CanopyStorage = CanopyStorage - DD

# Add the precipitation that falls on the canopy to the store
CanopyStorage = CanopyStorage + Pfrac

# Now do the Evap, make sure the store does not get negative
dC = -1 * pcr.min(CanopyStorage, PotEvap)
CanopyStorage = CanopyStorage + dC

LeftOver = PotEvap + dC
# Amount of evap not used
```

(continues on next page)

(continued from previous page)

```
# Now drain the canopy storage again if needed...
D = pcr.ifthenelse(CanopyStorage > Cmax, CanopyStorage - Cmax, 0.0)
CanopyStorage = CanopyStorage - D

# Calculate throughfall
ThroughFall = DD + D + p * Precipitation
StemFlow = Precipitation * pt

# Calculate interception, this is NET Interception
NetInterception = Precipitation - ThroughFall - StemFlow
Interception = -dC

return NetInterception, ThroughFall, StemFlow, LeftOver, Interception, CanopyStorage
```

Snow and Glaciers

Snow and glaciers processes, from the HBV model, are available in the wflow framework. The snow and glacier functions are used by the wflow_sbm model, and the glacier function is used by the wflow_hbv model.

Snow modelling

If the air temperature, T_a, is below a user-defined threshold $TT(\approx 0^oC)$ precipitation occurs as snowfall, whereas it occurs as rainfall if $T_a \geq TT$. A another parameter TTI defines how precipitation can occur partly as rain of snowfall (see the figure below). If precipitation occurs as snowfall, it is added to the dry snow component within the snow pack. Otherwise it ends up in the free water reservoir, which represents the liquid water content of the snow pack. Between the two components of the snow pack, interactions take place, either through snow melt (if temperatures are above a threshold TT) or through snow refreezing (if temperatures are below threshold TT). The respective rates of snow melt and refreezing are:

$$Q_m = cfmax(T_a - TT) \;\; ;T_a > TT$$
$$Q_r = cfmax * cfr(TT - T_a) \; ;T_a < TT$$

where Q_m is the rate of snow melt, Q_r is the rate of snow refreezing, and $cfmax$ and $cfr$ are user defined model parameters (the melting factor $mm/(^oC * day)$ and the refreezing factor respectively)

Note: The FoCFMAX parameter from the original HBV version is not used. instead the CFMAX is presumed to be for the landuse per pixel. Normally for forested pixels the CFMAX is 0.6 {*} CFMAX

The air temperature, T_a, is related to measured daily average temperatures. In the original HBV-concept, elevation differences within the catchment are represented through a distribution function (i.e. a hypsographic curve) which makes the snow module semi-distributed. In the modified version that is applied here, the temperature, T_a, is represented in a fully distributed manner, which means for each grid cell the temperature is related to the grid elevation.

The fraction of liquid water in the snow pack (free water) is at most equal to a user defined fraction, WHC, of the water equivalent of the dry snow content. If the liquid water concentration exceeds WHC, either through snow melt or incoming rainfall, the surpluss water becomes available for infiltration into the soil:

$$Q_{in} = max\{(SW - WHC * SD);\; 0.0\}$$

where Q_{in} is the volume of water added to the soil module, SW is the free water content of the snow pack and SD is the dry snow content of the snow pack.

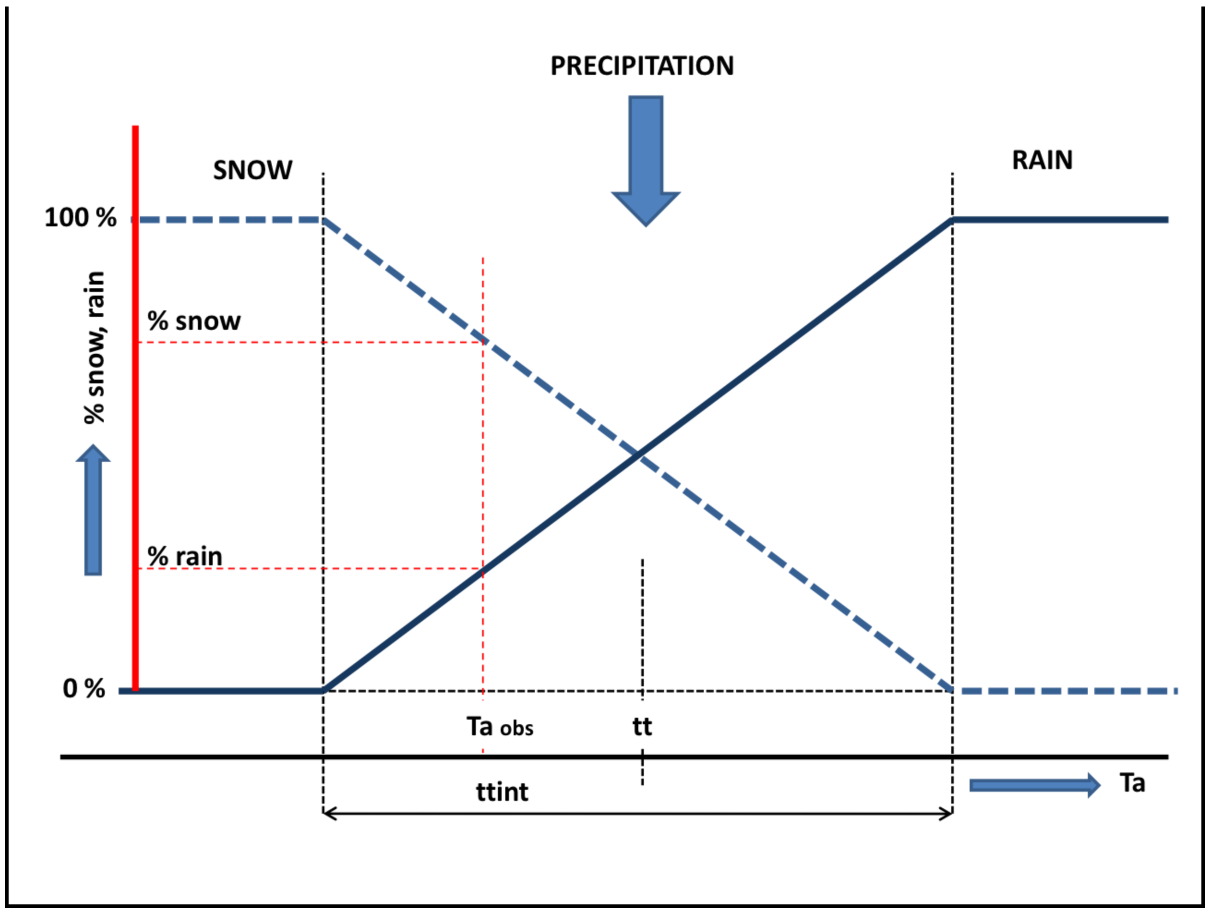


Fig. 24: Schematic view of the snow routine

Glacier modelling

Glacier processes can be modelled if the snow model is enabled in wflow_sbm. For wflow_hbv snow modelling is not optional. Glacier modelling is very close to snow modelling and considers two main processes: glacier build-up from snow turning into firn/ice (using the HBV-light model) and glacier melt (using a temperature degree-day model).

The definition of glacier boundaries and initial volume is defined in three staticmaps. *GlacierAreas* is a map containing the ID of the glacier present in the wflow cell. *GlacierFrac* is a map that gives the fraction of each grid cell covered by a glacier as a number between zero and one. *GlacierStore* is a state map that gives the amount of water (in mm w.e.) within the glaciers at each gridcell. Because the glacier store (GlacierStore.map) cannot be initialized by running the model for a couple of years, a default initial state map should be supplied by placing a GlacierStore.map file in the staticmaps directory. These three maps are prepared from available glacier datasets.

First, a fixed fraction of the snowpack on top of the glacier is converted into ice for each timestep and added to the glacier store using the HBV-light model (Seibert et al.,2017). This fraction, defined in the lookup table *G_SIfrac*, typically ranges from 0.001 to 0.006.

Then, when the snowpack on top of the glacier is almost all melted (snow cover < 10 mm), glacier melt is enabled and estimated with a degree-day model. If the air temperature, T_a, is below a certain threshold $G\_TT(\approx 0^oC)$ precipitation occurs as snowfall, whereas it occurs as rainfall if $T_a \geq G\_TT$.

With this the rate of glacier melt in mm is estimated as:

$$Q_m = G\_Cfmax(T_a - G\_TT) \ \ ;T_a > G\_TT$$

where Q_m is the rate of glacier melt and G_Cfmax is the melting factor in $mm/(^oC * day)$. Parameters *G_TT* and *G_Cfmax* are defined in two lookup tables. *G_TT* can be taken as equal to the snow TT parameter. Values of the melting factor normally varies from one glacier to another and some values are reported in the literature. *G_Cfmax* can also be estimated by multiplying snow Cfmax by a factor between 1 and 2, to take into account the higher albedo of ice compared to snow.

Glacier modelling can be enabled by including the following four entries in the modelparameters section:

```
[modelparameters]
GlacierAreas = staticmaps/wflow_glacierareas.map,staticmap,0.0,0
GlacierFrac = staticmaps/wflow_glacierfrac.map,staticmap,0.0,0
G_TT = intbl/G_TT.tbl,tbl,0.0,1,staticmaps/wflow_glacierareas.map
G_Cfmax = intbl/G_Cfmax.tbl,tbl,3.0,1,staticmaps/wflow_glacierareas.map
G_SIfrac = intbl/G_SIfrac.tbl,tbl,0.001,1,staticmaps/wflow_glacierareas.map
```

The initial glacier volume wflow_glacierstore.map should also be added in the staticmaps folder.

Reservoirs and Lakes

Simplified reservoirs and lakes models are included in the framwork and used by the wflow_sbm, wflow_hbv and wflow_routing models.

Reservoirs

Simple reservoirs can be included within the kinematic wave routing by supplying two maps, one map with the outlet of the reservoirs in which each reservoir has a unique id (ReserVoirSimpleLocs), and one map with the extent of reservoir (ReservoirSimpleAreas). Furthermore a set of lookuptables must be defined linking the reservoir-id's to reservoir characteristics:

- ResTargetFullFrac.tbl - Target fraction full (of max storage) for the reservoir: number between 0 and 1
- ResTargetMinFrac.tbl - Target minimum full fraction (of max storage). Number between 0 and 1 < ResTargetFullFrac
- ResMaxVolume.tbl - Maximum reservoir storage (above which water is spilled) [m^3]
- ResDemand.tbl - Minimum (environmental) flow requirement downstream of the reservoir m^3/s
- ResMaxRelease.tbl - Maximum Q that can be released if below spillway [m^3/s]
- ResSimpleArea.tbl - Surface area of the reservoir [m^2]

By default the reservoirs are not included in the model. To include them put the following lines in the .ini file of the model.

```
[modelparameters]
# Add this if you want to model reservoirs
ReserVoirSimpleLocs=staticmaps/wflow_reservoirlocs.map,staticmap,0.0,0
ReservoirSimpleAreas=staticmaps/wflow_reservoirareas.map,staticmap,0.0,0
ResSimpleArea = intbl/ResSimpleArea.tbl,tbl,0,0,staticmaps/wflow_reservoirlocs.map
ResTargetFullFrac=intbl/ResTargetFullFrac.tbl,tbl,0.8,0,staticmaps/wflow_reservoirlocs.
↪map
ResTargetMinFrac=intbl/ResTargetMinFrac.tbl,tbl,0.4,0,staticmaps/wflow_reservoirlocs.map
ResMaxVolume=intbl/ResMaxVolume.tbl,tbl,0.0,0,staticmaps/wflow_reservoirlocs.map
ResMaxRelease=intbl/ResMaxRelease.tbl,tbl,1.0,0,staticmaps/wflow_reservoirlocs.map
ResDemand=intbl/ResDemand.tbl,tblmonthlyclim,1.0,0,staticmaps/wflow_reservoirlocs.map
```

In the above example most values are fixed thoughout the year, only the ResDemand is in given per month of the year.

Natural Lakes

Natural (uncontrolled) lakes can be modelled in wflow using a mass balance approach:

$$\frac{S(t+\Delta t)}{\Delta t} = \frac{S(t)}{\Delta t} + Q_{in} + \frac{(P-E)*A}{\Delta t} - Q_{out}$$

where S is lake storage [m^3], Δt is the model timestep [seconds], Q_{in} is the sum of inflows (surface runoff and seepage) [m^3/s], Q_{out} is the lake outflow at the outlet [m^3/s], P is precipitation [m], E is lake evaporation [m] and A is lake surface.

Most of the variables in this equation are already known or coming from previous timestep, apart from $S(t+\Delta t)$ and Q_{out} which can both be linked to the water level H in the lake using a storage curve $S = f(H)$ and a rating curve $Q = f(H)$. In wflow, several options are available to select storage and rating curves, and in most cases, the mass balance is then solved by linearization and iteration or using the Modified Puls Approach from Maniak (Burek et al., 2013). Storage curves in wflow can either:

- Come from the interpolation of field data linking volume and lake height,
- Be computed from the simple relationship $S = A * H$.

Rating curves in wlow can either:

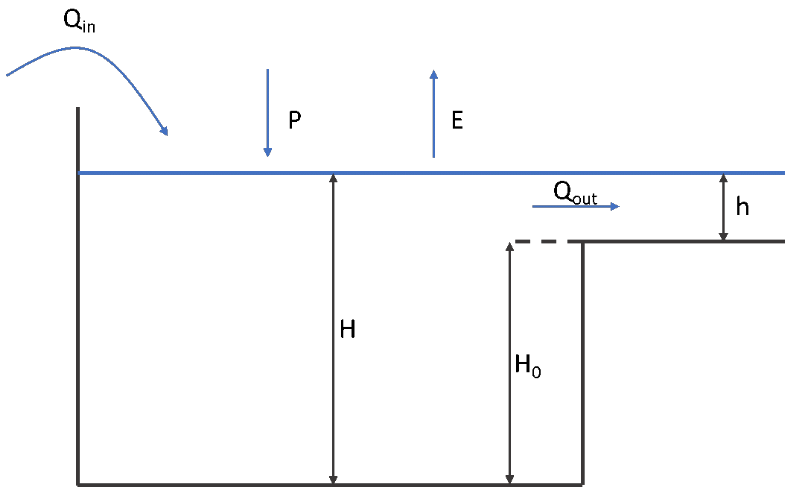


Fig. 25: Lake schematisation.

- Come from the interpolation of field data linking lake outflow and water height,
- Be computed from a rating curve of the form $Q_{out} = \alpha * (H - H_0)^\beta$, where H_0 is the minimum water level under which the outflow is zero. Usual values for β are 3/2 for a rectangular weir or 2 for a parabolic weir (Bos, 1989).

Modified Puls Approach

The Modified Puls Approach is a resolution method of the lake balance that uses an explicit relationship between storage and outflow. Storage is assumed to be equal to A*H and the rating curve for a parabolic weir (beta =2):

$$S = A * H = A * (h + H_0) = \frac{A}{\sqrt{\alpha}}\sqrt{Q} + A * H_0$$

Inserting this equation in the mass balance gives:

$$\frac{A}{\Delta t\sqrt{\alpha}}\sqrt{Q} + Q = \frac{S(t)}{\Delta t} + Q_{in} + \frac{(P-E)*A}{\Delta t} - \frac{A*H_0}{\Delta t} = SI - \frac{A*H_0}{\Delta t}$$

The solution for Q is then:

- $Q = \left(-LF + \sqrt{LF^2 + 2 * \left(SI - \frac{A*H_0}{\Delta t}\right)}\right)^2$ for $SI > \frac{A*H_0}{\Delta t}$ and where $LF = \frac{A}{\Delta t\sqrt{\alpha}}$
- $Q = 0$ for $SI \leq \frac{A*H_0}{\Delta t}$

Setup and input files

Natural lakes can be included within the kinematic wave routing in wflow, by supplying two maps with the locations of the lakes (one map for the extents, LakeAreas.map, and one for the outlets, LakeLocs.map) and in which each lake has a unique id. Furthermore a set of lookuptables must be dened linking the lake id's to lake characteristics:

- LakeArea: surface area of the lakes [m^2]
- LakeAvgLevel: average lake water level [m], used to reinitiate model states [m].
- LakeThreshold: water level threshold H_0 under which outflow is zero [m].
- LakeStorFunc: type of lake storage curve ; 1 for S = AH (default) and 2 for S = f(H) from lake data and interpolation.
- LakeOutflowFunc: type of lake rating curve ; 1 for Q = f(H) from lake data and interpolation, 2 for general $Q = b(H - H_0)^e$ and 3 in the case of Puls Approach $Q = b(H - H_0)^2$ (default).
- Lake_b: rating curve coefficient.
- Lake_e: rating curve exponent.

By default, the lakes are not included in the model. To include them, put the following lines in the .ini le of the model:

```
LakeLocs=staticmaps/wflow_lakelocs.map,staticmap,0.0,0
LakeAreasMap=staticmaps/wflow_lakeareas.map,staticmap,0.0,0
LinkedLakeLocs=intbl/LinkedLakeLocs.tbl,tbl,0,0,staticmaps/wflow_lakelocs.map
LakeStorFunc = intbl/LakeStorFunc.tbl,tbl,1,0,staticmaps/wflow_lakelocs.map
LakeOutflowFunc = intbl/LakeOutflowFunc.tbl,tbl,3,0,staticmaps/wflow_lakelocs.map
LakeArea = intbl/LakeArea.tbl,tbl,1,0,staticmaps/wflow_lakelocs.map
LakeAvgLevel = intbl/LakeAvgLevel.tbl,tbl,1,0,staticmaps/wflow_lakelocs.map
LakeAvgOut = intbl/LakeAvgOut.tbl,tbl,1,0,staticmaps/wflow_lakelocs.map
LakeThreshold = intbl/LakeThreshold.tbl,tbl,0,0,staticmaps/wflow_lakelocs.map
Lake_b = intbl/Lake_b.tbl,tbl,50,0,staticmaps/wflow_lakelocs.map
Lake_e = intbl/Lake_e.tbl,tbl,2.0,0,staticmaps/wflow_lakelocs.map
```

Additional settings

- Storage / rating curve from data

Storage and rating curves coming from field measurement can be supplied to wflow via tbl files supplied in the intbl folder. Naming of the files uses the ID of the lakes where data are available and is of the form Lake_SH_1.tbl and Lake_HQ_1.tbl for respectively the storage and rating curves of lake with ID 1.

The storage curve is a tbl with lake level [m] in the first column and corresponding lake storage [m^3] in the second column.

The rating curve uses level and discharge data depending on the Julian day of the year (JDOY). The first line contains the minimum level and corresponding average yearly discharge, and the second line the maximum level and corresponding average yearly discharge. The third line contains first a blank cell and numbers for the days of the year (from 1 to 365). The other lines contain the water level and the corresponding discharges for the different JDOY (example below).

```
394.00   43.000
398.00 1260.000
                1          2           3 ...       364         365
394.00      43.000     43.000      43.000 ...    43.000      43.000
394.01      44.838     44.838      44.838 ...    44.838      44.838
```

(continues on next page)

(continued from previous page)

```
...
397.99 1256.229 1256.229 1256.229 ... 1256.229 1256.229
398.00 1260.000 1260.000 1260.000 ... 1260.000 1260.000
```

- Estimating the lake water level threshold for outflow

By default, the lake water level threshold (for which lake ouflow is zero) is set to zero. For some more specific studies, this threshold can be estimated using the lake average outflow and level as well as the river characteristics at the outlet (relationship between discharge and water level from the kinematic wave). After estimation of the threshold, the parameter α from the lake rating curve is also adjusted.

To enable this option, the line estimatelakethresh = 1 should be added to the [model] section of the ini file (default value is 0), and the tbls LakeAvgOut [m^3/s] and LakeAvgLevel [m] with the lake average ouflow and level should be supplied.

- Linked lakes

In some cases, close lakes can be linked and return flow can be allowed from the downstream to the upstream lake. The linked lakes are defined in the LinkedLakeLocs.tbl file by linking the lake ID of the upstream lake (first column) to the ID of the linked downstream lake (second column).

Note: in every file, level units are meters [m] and NOT meters above sea level [m asl]. Especially with storage/rating curves coming from data, please be careful and convert units if needed.

References

- Burek P., Van der Knijf J.M., Ad de Roo, 2013. LISFLOOD – Distributed Water Balance and flood Simulation Model – Revised User Manual. DOI: http://dx.doi.org/10.2788/24719.
- Bos M.G., 1989. Discharge measurement structures. Third revised edition, International Institute for Land Reclamation and Improvement ILRI, Wageningen, The Netherlands.

wflow_funcs module documentation

wflow_funcs - hydrological functions library

In addition this library contain a number of hydrological functions that may be used within the wflow models

It contains both the kinematic wave, interception, snow/glaciers and reservoirs/lakes modules.

`wflow_funcs.`**`SnowPackHBV`**`(`*Snow*, *SnowWater*, *Precipitation*, *Temperature*, *TTI*, *TT*, *TTM*, *Cfmax*, *WHC*`)`

HBV Type snowpack modelling using a Temperature degree factor. All correction factors (RFCF and SFCF) are set to 1. The refreezing efficiency factor is set to 0.05.

Parameters

- **`Snow`** –
- **`SnowWater`** –
- **`Precipitation`** –
- **`Temperature`** –
- **`TTI`** –
- **`TT`** –

- **TTM** –
- **Cfmax** –
- **WHC** –

Returns
Snow,SnowMelt,Precipitation

wflow_funcs.**bf_oneparam**(*discharge*, *k*)

wflow_funcs.**bf_threeparam**(*discharge*, *k*, *C*, *a*)

wflow_funcs.**bf_twoparam**(*discharge*, *k*, *C*)

wflow_funcs.**estimate_iterations_kin_wave**(*Q*, *Beta*, *alpha*, *timestepsecs*, *dx*, *mv*)

wflow_funcs.**glacierHBV**(*GlacierFrac*, *GlacierStore*, *Snow*, *Temperature*, *TT*, *Cfmax*, *G_SIfrac*, *timestepsecs*, *basetimestep*)

Run Glacier module and add the snowpack on-top of it. First, a fraction of the snowpack is converted into ice using the HBV-light model (fraction between 0.001-0.005 per day). Glacier melting is modelled using a Temperature degree factor and only occurs if the snow cover < 10 mm.

Variables

- **GlacierFrac** – Fraction of wflow cell covered by glaciers
- **GlacierStore** – Volume of the galcier in the cell in mm w.e.
- **Snow** – Snow pack on top of Glacier
- **Temperature** – Air temperature
- **TT** – Temperature threshold for ice melting
- **Cfmax** – Ice degree-day factor in mm/(°C/day)
- **G_SIfrac** – Fraction of the snow part turned into ice each daily timestep
- **timestepsecs** – Model timestep in seconds
- **basetimestep** – Model base timestep (86 400 seconds)

Returns
Snow,Snow2Glacier,GlacierStore,GlacierMelt,

wflow_funcs.**kin_wave**(*rnodes*, *rnodes_up*, *Qold*, *q*, *Alpha*, *Beta*, *DCL*, *River*, *Bw*, *AlpTermR*, *AlpPow*, *deltaT*, *it=1*)

wflow_funcs.**kinematic_wave**(*Qin*, *Qold*, *q*, *alpha*, *beta*, *deltaT*, *deltaX*)

wflow_funcs.**kinematic_wave_ssf**(*ssf_in*, *ssf_old*, *zi_old*, *r*, *Ks_hor*, *Ks*, *slope*, *neff*, *f*, *D*, *dt*, *dx*, *w*, *ssf_max*)

wflow_funcs.**lookupResFunc**(*ReserVoirLocs*, *values*, *sh*, *dirLookup*)

wflow_funcs.**lookupResRegMatr**(*ReserVoirLocs*, *values*, *hq*, *JDOY*)

wflow_funcs.**naturalLake**(*waterlevel*, *LakeLocs*, *LinkedLakeLocs*, *LakeArea*, *LakeThreshold*, *LakeStorFunc*, *LakeOutflowFunc*, *sh*, *hq*, *lake_b*, *lake_e*, *inflow*, *precip*, *pet*, *LakeAreasMap*, *JDOY*, *timestepsecs=86400*)

Run Natural Lake module to compute the new waterlevel and outflow. Solves lake water balance with linearisation and iteration procedure, for any rating and storage curve. For the case where storage curve is S = AH and Q=b(H-Ho)^2, uses the direct solution from the Modified Puls Approach (LISFLOOD).

Variables

- `waterlevel` – water level H in the lake
- `LakeLocs` – location of lake's outlet
- `LinkedLakeLocs` – ID of linked lakes
- `LakeArea` – total lake area
- `LakeThreshold` – water level threshold Ho under which outflow is zero
- `LakeStorFunc` – type of lake storage curve 1: S = AH 2: S = f(H) from lake data and interpolation
- `LakeOutflowFunc` – type of lake rating curve 1: Q = f(H) from lake data and interpolation 2: General Q = b(H - Ho)^e 3: Case of Puls Approach Q = b(H - Ho)^2
- `sh` – data for storage curve
- `hq` – data for rating curve
- `lake_b` – rating curve coefficient
- `lake_e` – rating curve exponent
- `inflow` – inflow to the lake (surface runoff + river discharge + seepage)
- `precip` – precipitation map
- `pet` – PET map
- `LakeAreasMap` – lake extent map (for filtering P and PET)
- `JDOY` – Julian Day of Year to read storage/rating curve from data
- `timestepsecs` – model timestep in seconds

Returns

waterlevel, outflow, prec_av, pet_av, storage

`wflow_funcs.`**`propagate_downstream`**(*rnodes*, *rnodes_up*, *material*)

`wflow_funcs.`**`rainfall_interception_gash`**(*Cmax*, *EoverR*, *CanopyGapFraction*, *Precipitation*, *CanopyStorage*, *maxevap=9999*)

Interception according to the Gash model (For daily timesteps).

`wflow_funcs.`**`rainfall_interception_hbv`**(*Rainfall*, *PotEvaporation*, *Cmax*, *InterceptionStorage*)

Returns: TF, Interception, IntEvap,InterceptionStorage

`wflow_funcs.`**`rainfall_interception_modrut`**(*Precipitation*, *PotEvap*, *CanopyStorage*, *CanopyGapFraction*, *Cmax*)

Interception according to a modified Rutter model. The model is solved explicitly and there is no drainage below Cmax.

Returns:

- NetInterception: P - TF - SF (may be different from the actual wet canopy evaporation)
- ThroughFall:
- StemFlow:
- LeftOver: Amount of potential eveporation not used
- Interception: Actual wet canopy evaporation in this thimestep

- CanopyStorage: Canopy storage at the end of the timestep

`wflow_funcs.`**`sCurve`**`(`*X*, *a=0.0*, *b=1.0*, *c=1.0*`)`

sCurve function:

Input:

- X input map
- C determines the steepness or "stepwiseness" of the curve. The higher C the sharper the function. A negative C reverses the function.
- b determines the amplitude of the curve
- a determines the centre level (default = 0)

Output:

- result

`wflow_funcs.`**`set_dd`**`(`*ldd*, *_ldd_us=array([[3, 2, 1], [6, 5, 4], [9, 8, 7]])*, *pit_value=5*`)`

set drainage direction network from downstream to upstream

`wflow_funcs.`**`simplereservoir`**`(`*storage*, *inflow*, *ResArea*, *maxstorage*, *target_perc_full*, *maximum_Q*, *demand*, *minimum_full_perc*, *ReserVoirLocs*, *precip*, *pet*, *ReservoirSimpleAreas*, *timestepsecs=86400*`)`

Parameters

- **`storage`** – initial storage m^3
- **`inflow`** – inflow m^3/s
- **`maxstorage`** – maximum storage (above which water is spilled) m^3
- **`target_perc_full`** – target fraction full (of max storage) -
- **`maximum_Q`** – maximum Q to release m^3/s if below spillway
- **`demand`** – water demand (all combined) m^3/s
- **`minimum_full_perc`** – target minimum full fraction (of max storage) -
- **`ReserVoirLocs`** – map with reservoir locations
- **`timestepsecs`** – timestep of the model in seconds (default = 86400)

Returns

storage (m^3), outflow (m^3/s), PercentageFull (0-1), Release (m^3/sec)

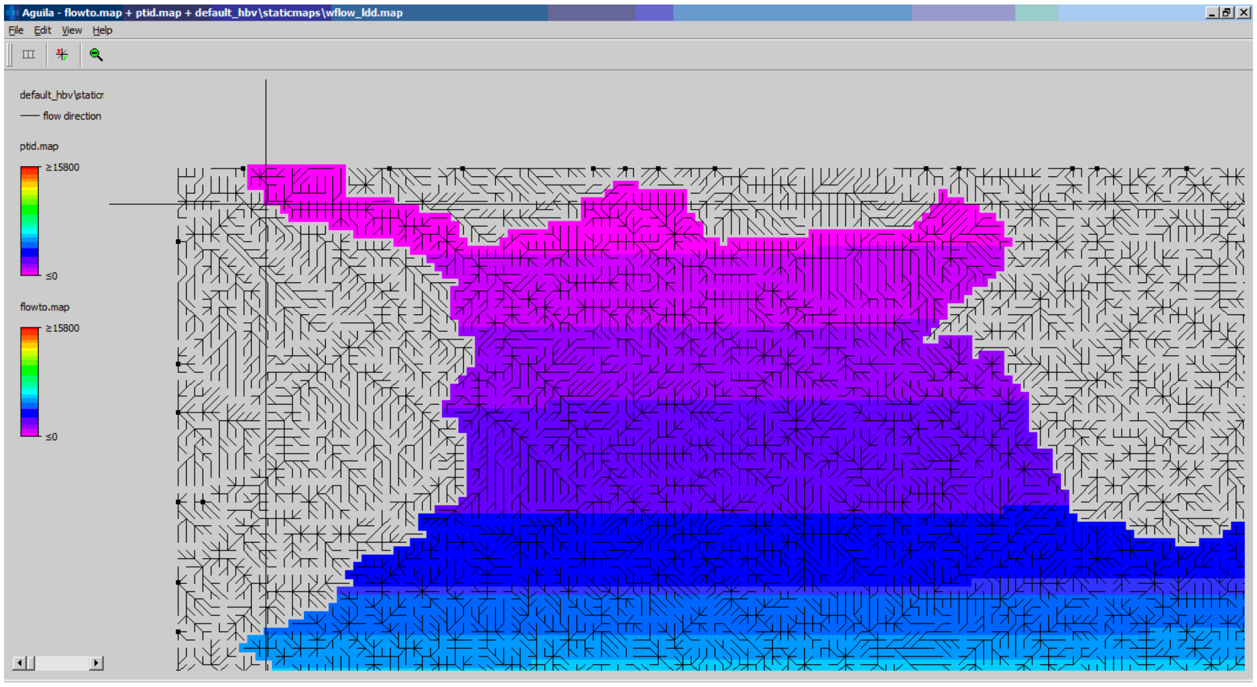
1.8.4 wflow_delwaq Module

The wflow_delwaq module provides a set of functions to construct a delwaq pointer file from a PCRaster local drainage network. A command-line interface is provide that allows you to create a delwaq model that can be linked to a wflow model.

The script sets-up a one-layer model (representing the kinematic wave reservoir). Water is labeled according to the area and flux where it enters the kinematic wave reservoir.

For the script to work a run of the wflow model must be available and a template directory in which the delwaq model is created should also be available. These are indicated by the -C -R and -D command line options. The -R and -C options indicate the wflow case and wflow run directories while the -D option indicates the delwaq template directory.

The template used is shown below:



```
debug/
fixed/
fixed/B2_numsettings.inc
fixed/B4_dispersion.inc
fixed/B4_dispx.inc
fixed/B9_Hisvar.inc
fixed/B9_Mapvar.inc
includes_deltashell/
includes_flow/
run.bat
dlwqlib.dll
libcta.dll
libiomp5md.dll
waq_plugin_wasteload.dll
delwaq1.exe
delwaq2.exe
deltashell.inp
```

The debug, includes_flow, and includes_deltashell directories are filled by the script. After that delwaq1.exe and delwaq2.exe programs may be run (the run.bat file shows how this is done):

```
delwaq1.exe deltashell.inp -np
delwaq2.exe deltashell.inp
```

the script sets up delwaq such that the result for the wflow gauges locations are stored in the deltashell.his file.

How the script works

The pointer file for delwaq is made using the following information:

1. The wflow_ldd.map files is used to create the internal flow network, it defines the segments and how water flows between the segments
2. The number of inflows into each segment determines is taken from the sources mapstacks (-S option). Together these sources should include all the water that enters the kinematic wave reservoir. These are the red and green arrows in the figure below
3. The delwaq network is generated for the area define in the wflow_catchment map. The included area is define by all cells were the catchment id in a cel is larger than 1.

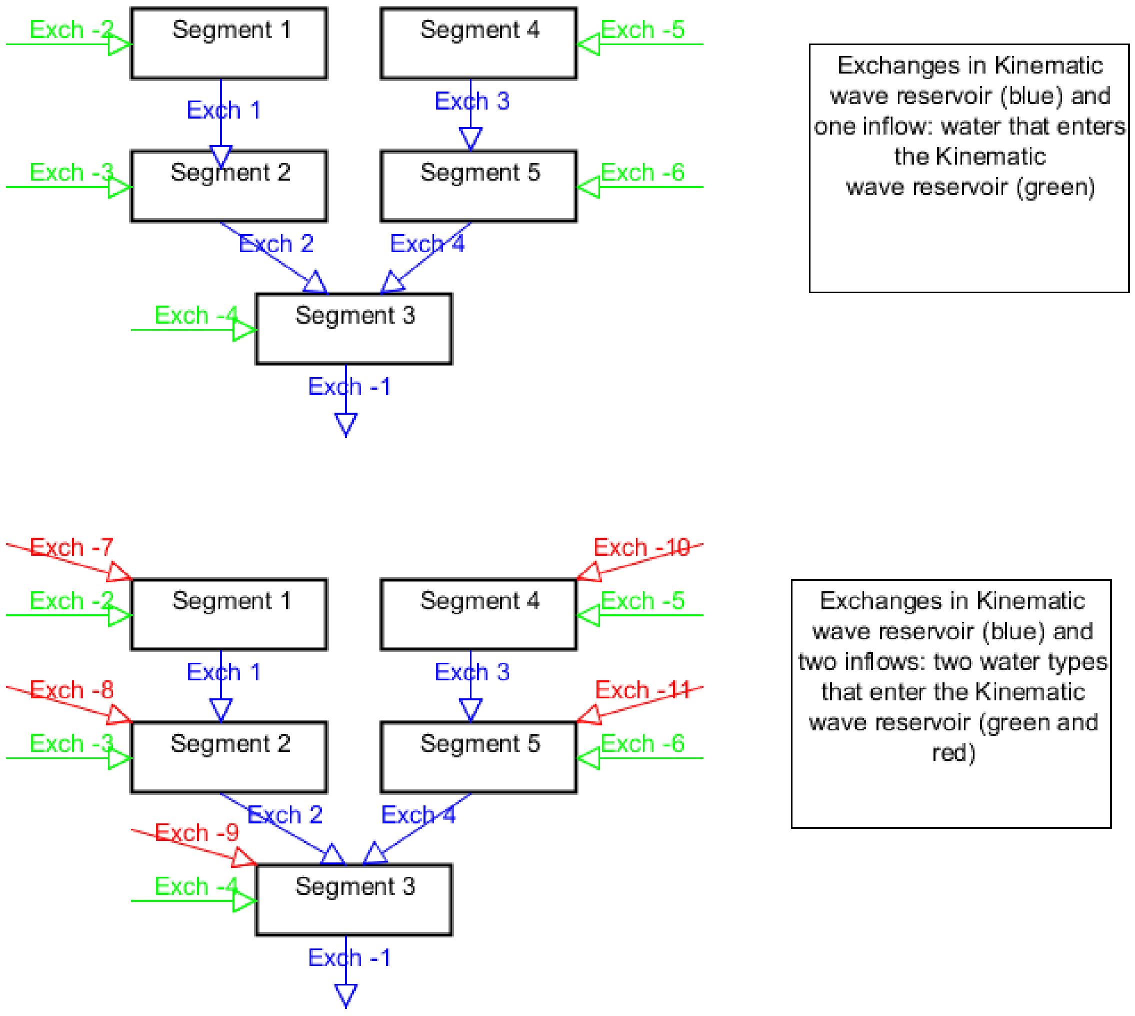


Fig. 26: Figure: How exchanges and inflows are connected

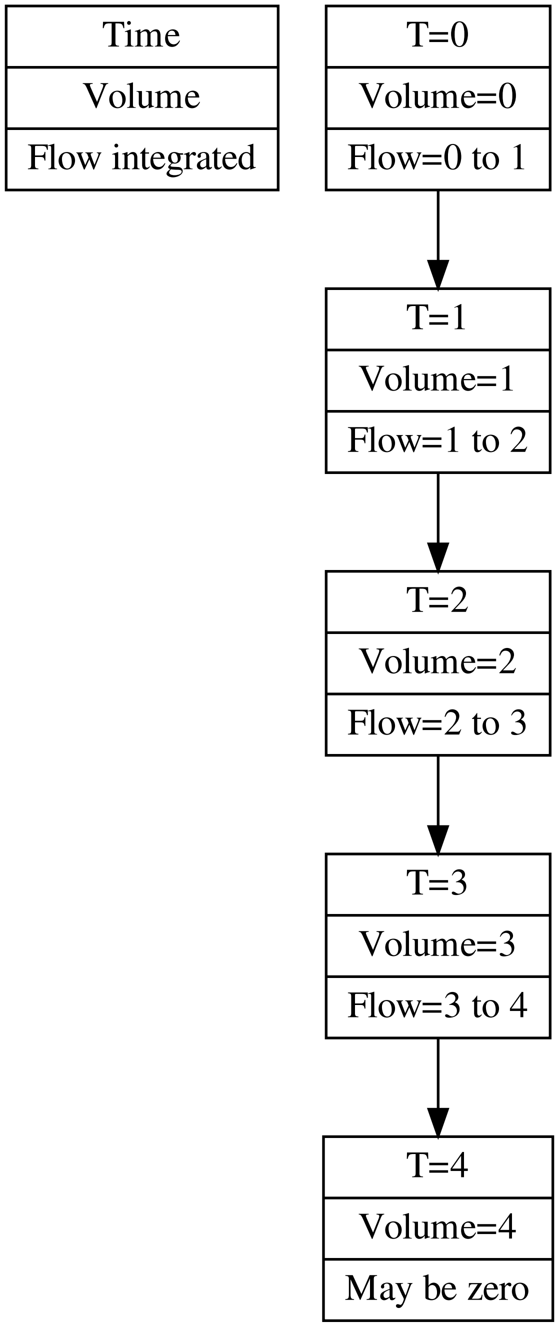
Within the includes_flow directory the following files are made:

- volume.dat - volumes (N+1) noseg

- flow.dat - flows (N). Contents is noq
- area.dat - N timesteps. Content is noq
- surface.dat - surface area of the water per segment (N+1), noseq
- length.dat - One timestep only (constant). Content is two times noq

Here nrseg is the number of segments (taken from the non-missing grid cell in the wflow_ldd.map file) and noq is the number of exchanges which is calculated as the number of segments plus number the of inflows (in each segment) times the number of segments

Delwaq expects volumes to be instantanious values at the start of a timestes while flows are integrated between tow timesteps. For volumes N+1 timesteps are needed, for flows N timesteps. The figure below demonstrates this principle for N=4.



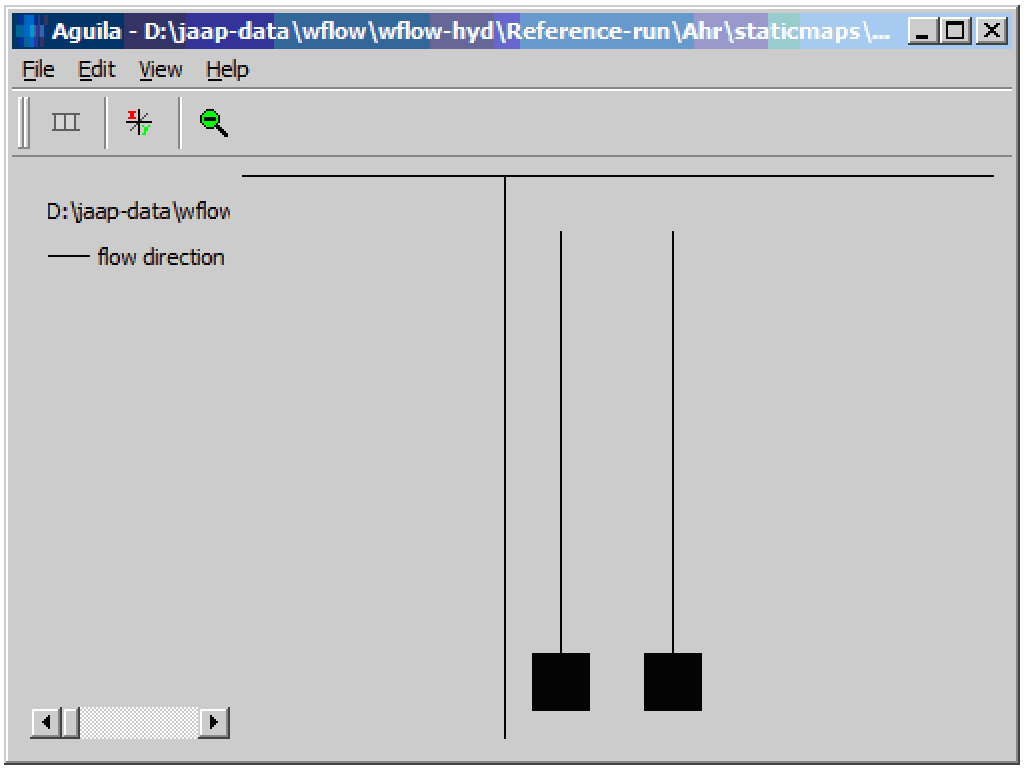
The volume.dat file is filled with N+1 steps of volumes of the wflow kinematic wave reservoir. To obtain the needed lag between the flows and the volumes the volumes are taken from the kinematic wave reservoir one timestep back (OldKinWaveVolume).

The flow.dat files is filled as follows. For each timestep internal flows (within the kinematic wave reservoir, i.e. flows from segment to segment) are written first (blue in the layout above). Next the flow into each segment are written.

Depending on how many inflow types are given to the script (sources). For one type, one set of flows is written, if there are two types two sets etc (green and red in the layout above).

Very simple example:

The following very simple example demonstrated how the pointer file is created. First the pcraster ldd:

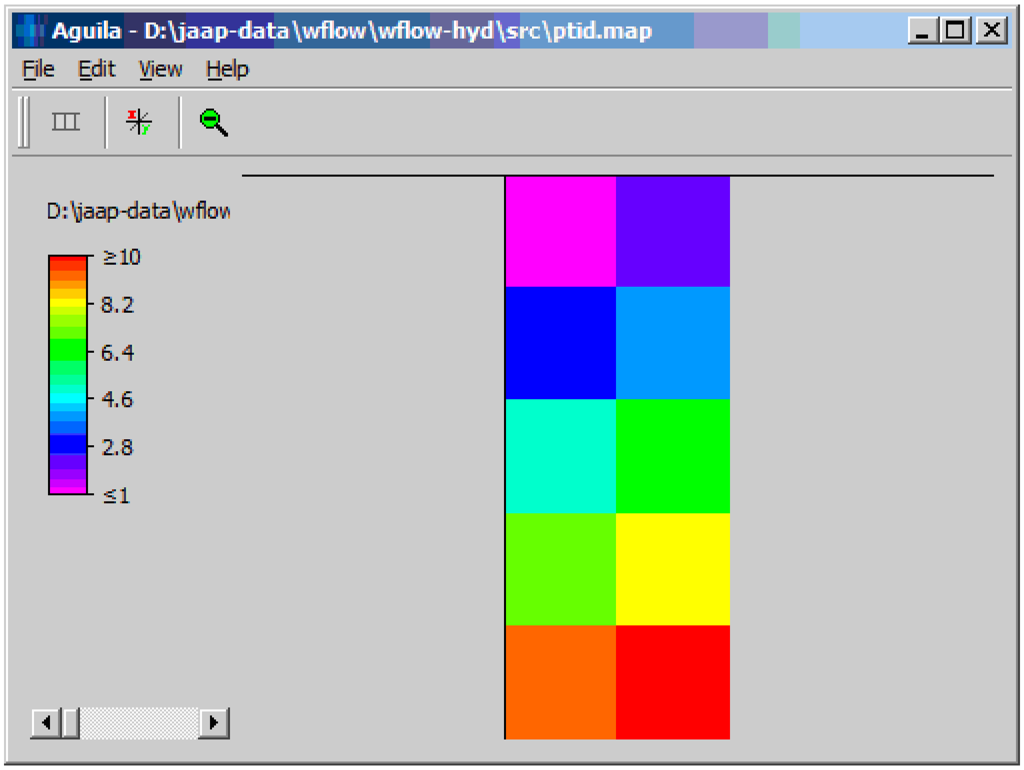

Aguila - D:\jaap-data\wflow\wflow-hyd\Reference-run\Ahr\staticmaps\...
File Edit View Help
D:\jaap-data\wflow
flow direction

The resulting network consist of 10 points:

As can be seen both 9 and 10 are bottom points. The generated pointer file is shown below:

```
;Written by dw_WritePointer
;nr of pointers is:  20
 1          3          0          0
 2          4          0          0
 3          5          0          0
 4          6          0          0
 5          7          0          0
 6          8          0          0
 7          9          0          0
 8         10          0          0
 9         -1          0          0
10         -2          0          0
-3          1          0          0
-4          2          0          0
-5          3          0          0
-6          4          0          0
-7          5          0          0
```

(continues on next page)



(continued from previous page)

```
 -8          6          0          0
 -9          7          0          0
-10          8          0          0
-11          9          0          0
-12         10          0          0
```

Case study for Malaysia and Singapore

To estimate load of different nutrients to Johor strait two wflow_sbm models have been setup. Next these models where linked to delwaq as follows:

1. A delwaq segment network similar to the wflow D8 ldd was made
2. The volumes in the delwaq segment are taken from the wflow_sbm kinematic wave volumes
3. For each segment two sources (inflows) are constructed, fast and slow each representing different runoff compartments from the wflow model. Fast represents SOF[1], HOF[2] and SSSF[3] while Slow represent groundwater flow.
4. Next the flow types are combined with the available land-use classes. As such a Luclass times flowtypes matrix of constituents is made. Each constituent (e.g. Slow flow of LU class 1) is traced throughout the system. All constituents are conservative and have a concentration of 1 as they flow in each segement.
5. To check for consistency an Initial water type and a Check water type are introduced. The Initial water will leave the system gradually after a cold start, the Check water type is added to each flow component and should be 1 at

[1] SOF: Saturation Overland Flow
[2] HOF: Hortonian Overland Flow (or infiltration excess Overland Flow)
[3] SSSF: SubSurface Storm Flow. Rapid lateral flow through the top part of the soil profile.

each location in the system (Mass Balance Check).

The above results in a system in which the different flow types (including the LU type where they have been generated) can be traced throughout the system. Each each gauge location the discharge and the flow components that make up the discharge are reported.

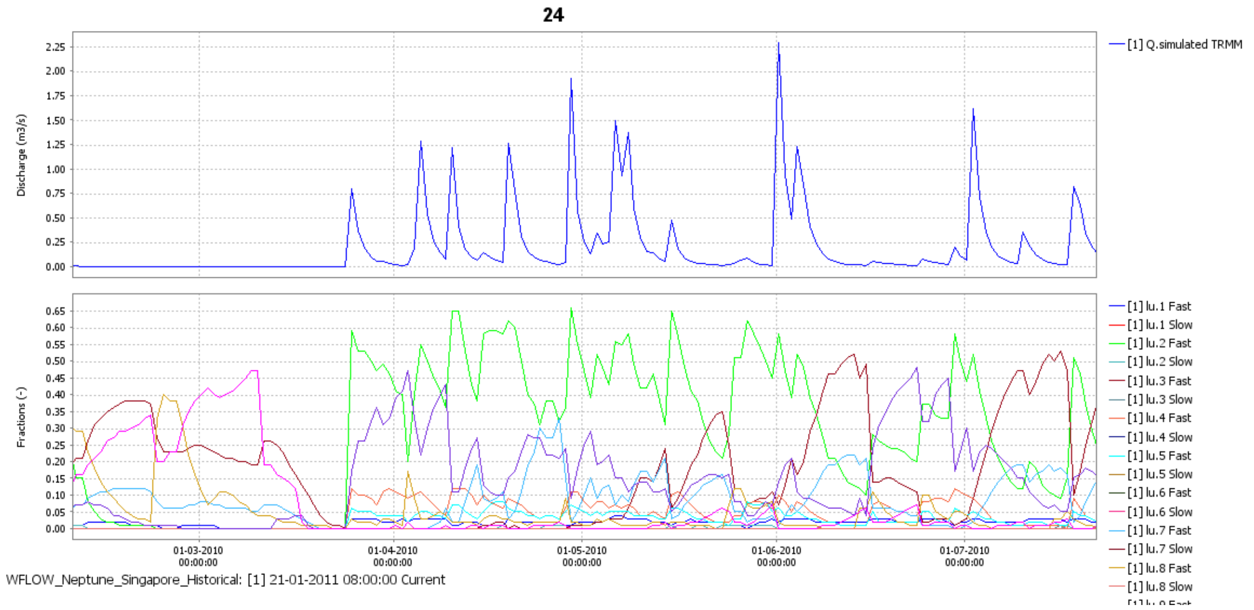


Fig. 27: Figure: Discharge and flow types for a small Singapore catchment. The Singapore catchment are dominated by fast flow types but during the end of the dry periods the slow flow types start to rise in importance.

By assuming each flow type is an end-member in a mixing model we can add fixed concentration of real parameters to the flow fractions and multiply those with the concentrations of the end-membesrt modelled concentration at the gauge locations can be obtained for each timestep.

The figure above shows the flow types in the models used in Singapore and Malaysia. Groundwater flow (both from completely saturated cell and subcell groundwater flow) makes up the Slow flow that is fed into the delwaq model while SOF and HOF make up the Fast flow to the delwaq model. In addition the water is also labelled according to the landuse type of the cell that it flows out off.

The whole procedure was setup in a Delft-FEWS configuration that can run the following steps operationally:

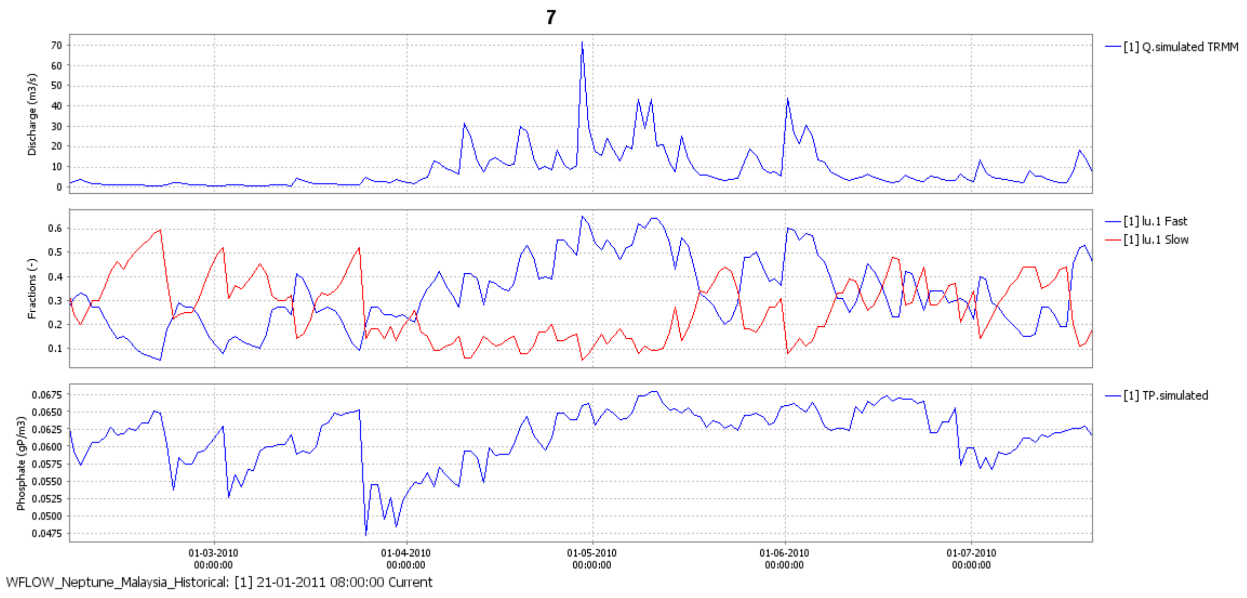


Fig. 28: Figure: Discharge, flow types and resulting total P for a catchment in Malaysia.

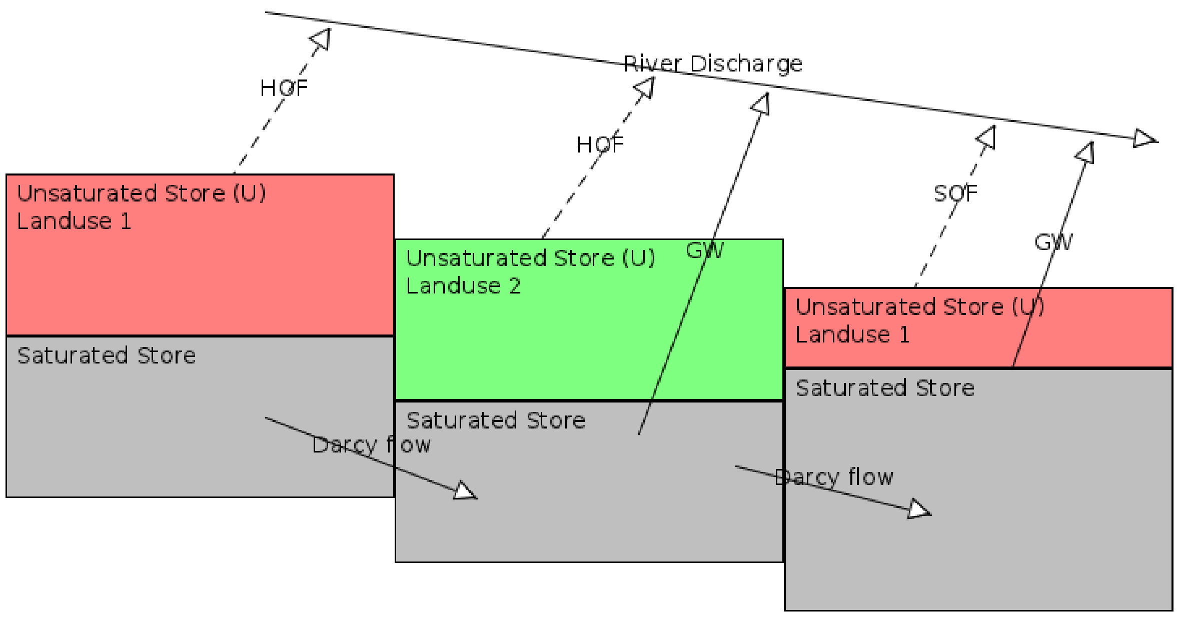
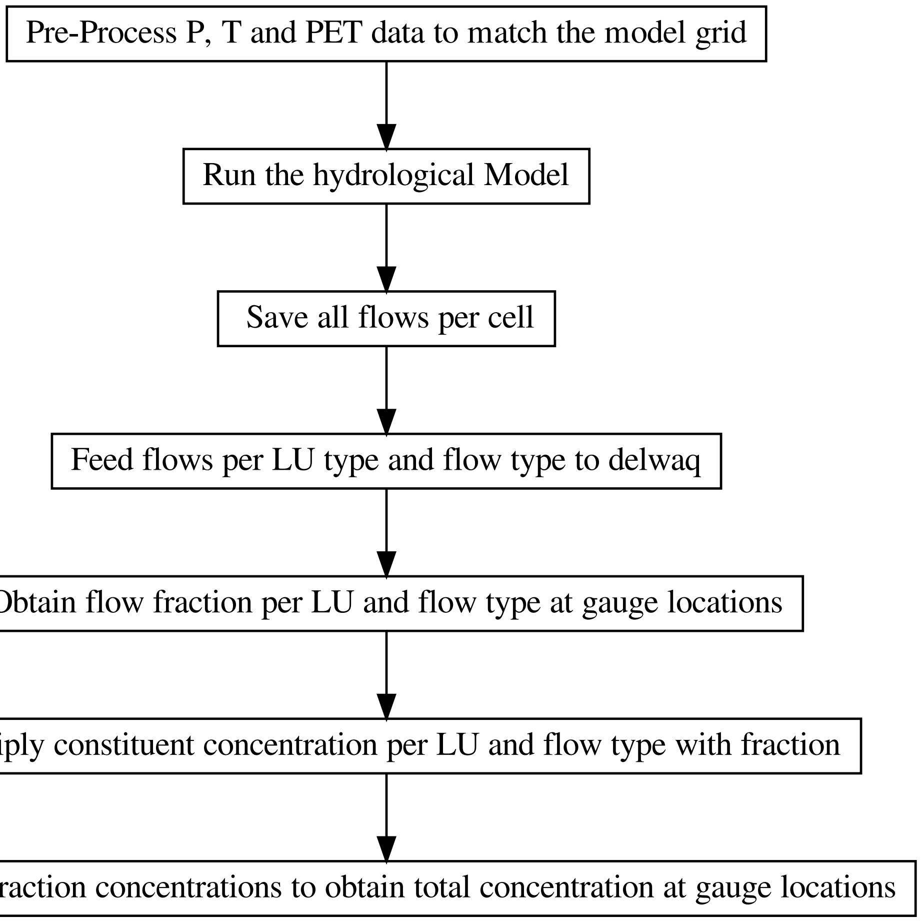

River Discharge
HOF
HOF
SOF
GW
GW
Unsaturated Store (U) Landuse 1
Unsaturated Store (U) Landuse 2
Unsaturated Store (U) Landuse 1
Saturated Store
Saturated Store
Saturated Store
Darcy flow
Darcy flow

Fig. 29: Figure: Flow types in the topog_sbm models used in the Singapore/Malaysia case. HOF = Hortonian or Infiltration excess overland flow. SOF = Saturation overland flow, GW = exfiltrating groundwater. Note that the subcell representation of saturated areas means that both SOF and GW can occur before a cell is completely saturated.


Pre-Process P, T and PET data to match the model grid
Run the hydrological Model
Save all flows per cell
Feed flows per LU type and flow type to delwaq
Obtain flow fraction per LU and flow type at gauge locations
Multiply constituent concentration per LU and flow type with fraction
Sum all fraction concentrations to obtain total concentration at gauge locations

wflow_delwaq module documentation

1.8.5 wflow_emwaq Module

The wflow_emwaq module provides a set of functions to create and link a Delft3Delwaq or D-Emission model to a wflow model. The module allows to both converts static and dynamic data from wflow_sbm to readable structure, flow and emission data used to set up an emission or a water quality model (later referred to as D-Emission and Delwaq).

It is an extension of the wflow_delwaq module that only handles the surface water for a Delwaq model. In addition, the script can aggregate results from wflow cells to build an emission/water quality model at the sub-catchment scale instead. The module has yet only been tested for wflow_sbm but could as well be applied to other wflow models.

Basic concepts on the structure of wflow and D-Emission/Delwaq

The wflow_sbm model is a fully distributed hydrologic model working on a regular grid of cells. Each cell is composed of several layers (or water buckets) such as open water, the unsaturated store or saturated store of the soil… (figure below) The link between each cell (or direction of the lateral water flow) is then defined according to the local drain direction map (ldd) which indicates which of the cell neighbours has the lowest elevation.

D-Emission or Delwaq are unstructured models. They are composed of unordered segments. Contrary to a wflow cell which contains several layers a D-Emission/WAQ segment only represent one layer, called a compartment. This means that one wflow cell is represented by several segments in D-Emission/WAQ: one for the open water, one for the unsaturated store, one for the saturated store… The direction of the flows in D-Emission/WAQ between each segment is defined in the pointer file. Contrary to wflow ldd which only needs to define the direction of lateral flows, the pointer file therefore also needs to indicate the direction of vertical flows (flows between the different layers/compartments of a same wflow cell). This also means that external flows coming to/out of a wflow cell (for example precipitation from the atmosphere) are defined in the pointer as flows between a segment and a boundary (for precipitation the boundary is the atmosphere).

The wflow_emwaq module main goal is then to convert wflow cells and ldd map into D-Emission/WAQ segments, compartments and pointer and to write the output flows from wflow into new flow files respecting the created cells/pointer framework.

As the cells grid from wflow_sbm can lead to a very large model (especially the 1km resolution of the global wflow_sbm model), it is possible with the coupling to aggregate the results from wflow cells to subcatchments. With this option, a D-Emission/WAQ segment represents then a specific compartment of a wflow subcatchment instead of a specific compartment for a wflow cell.

How to use the coupling module

Generalities

The wflow_emwaq module is a python script which has a similar structure as other wflow modules. First, a wflow_sbm model is set up and run normally via the ini file and input data (tbl, staticmaps, inmaps…). Then wflow_emwaq module is run to convert the outputs of wflow_sbm into several files that serves as inputs to D-Emission/WAQ models (see scheme below). The set-up of the wflow_emwaq run is also done via an ini file similar to wflow_sbm and via three csv tables that selects the compartments, fluxes and boundaries that the user wants to include in the D-Emission/WAQ models. Finally, the links to the different files created by the wflow_emwaq module are written in D-Emission/WAQ main input file and the emission and water quality model can be run.

Setting up the coupling

Defining the structure of the emission / water quality model

In order to set up an emission/water quality model, the user needs to choose which compartments, and which fluxes he wants to model. To aid that choice, the following figure sums up the different fluxes and the different layers/compartments of a wflow cell with their corresponding volumes.

For example, for a simple Delwaq run for the surface water with fraction calculation, only the surface water compartment is needed and all the fluxes coming in/out of it as shown in the left part of the following figure. In that case, the other compartments are turned into boundaries. In addition, as there is already a wflow variable that sums up all the in/outflows from the surface water (self.Inwater), the scheme can be simplified with just one boundary and flow (right part of the following figure).

Once the needed compartments and fluxes have been defined, they have to be translated into inputs for the wflow_emwaq module. This is done by editing the csv tables compartments.csv, fluxes.csv and boundaries.csv.

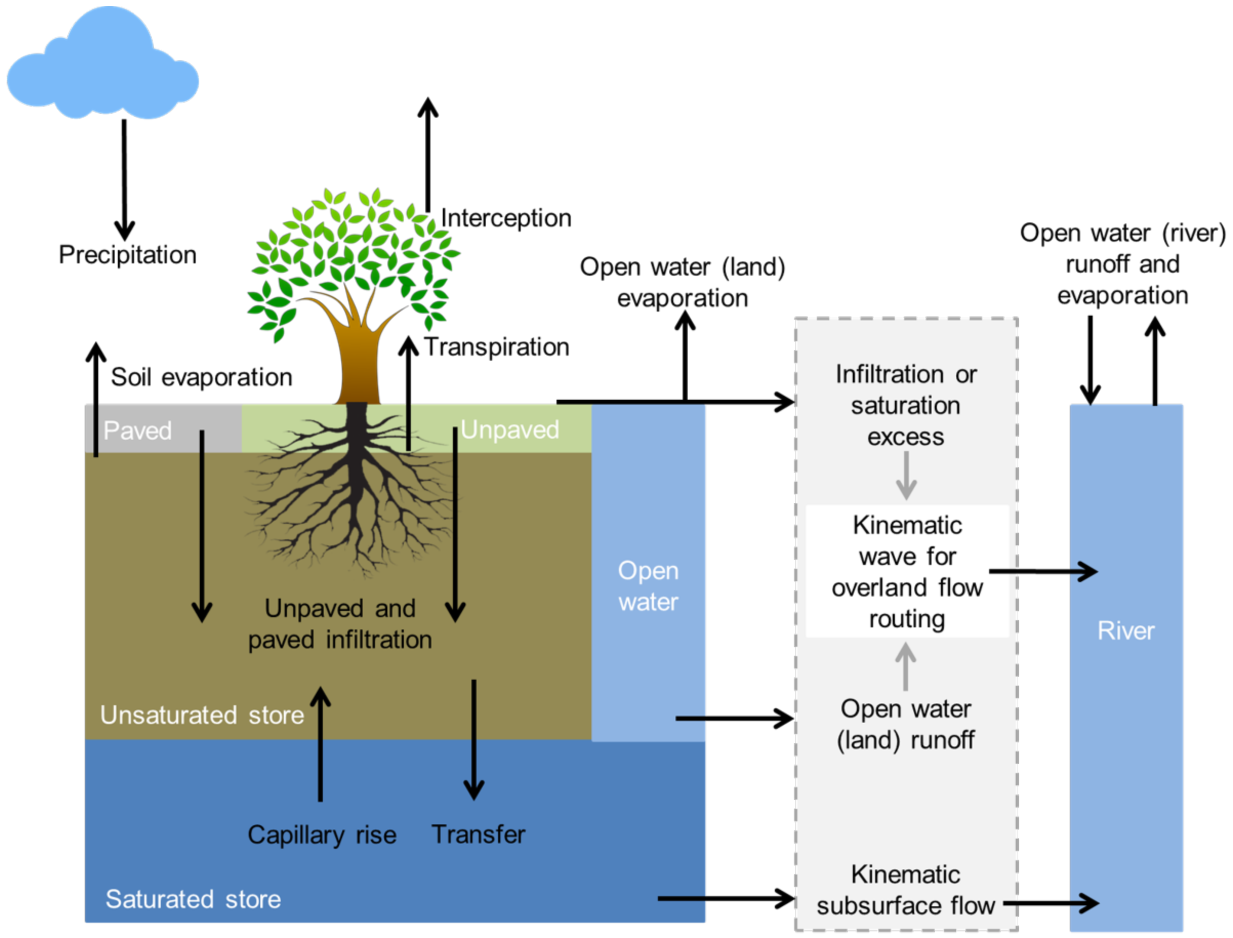

Precipitation
Interception
Open water (land) evaporation
Open water (river) runoff and evaporation
Soil evaporation
Transpiration
Paved
Unpaved
Infiltration or saturation excess
Kinematic wave for overland flow routing
Open water
River
Unpaved and paved infiltration
Unsaturated store
Open water (land) runoff
Capillary rise
Transfer
Saturated store
Kinematic subsurface flow

Fig. 30: Overview of the different processes and fluxes in the wflow_sbm model.

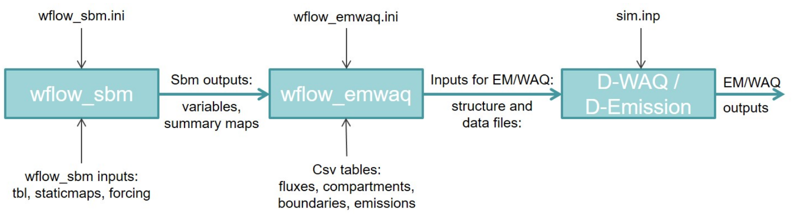


Fig. 31: Scheme of the coupling between wflow and D-Emission-Delwaq.

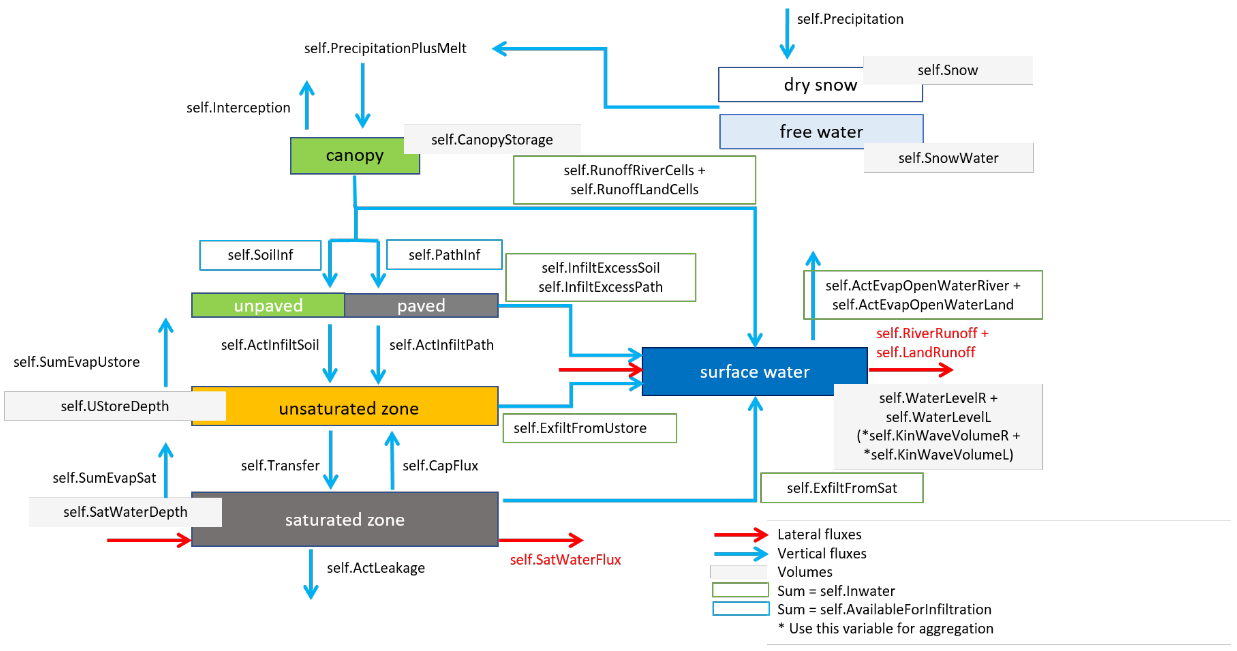


Fig. 32: Complete wflow scheme.

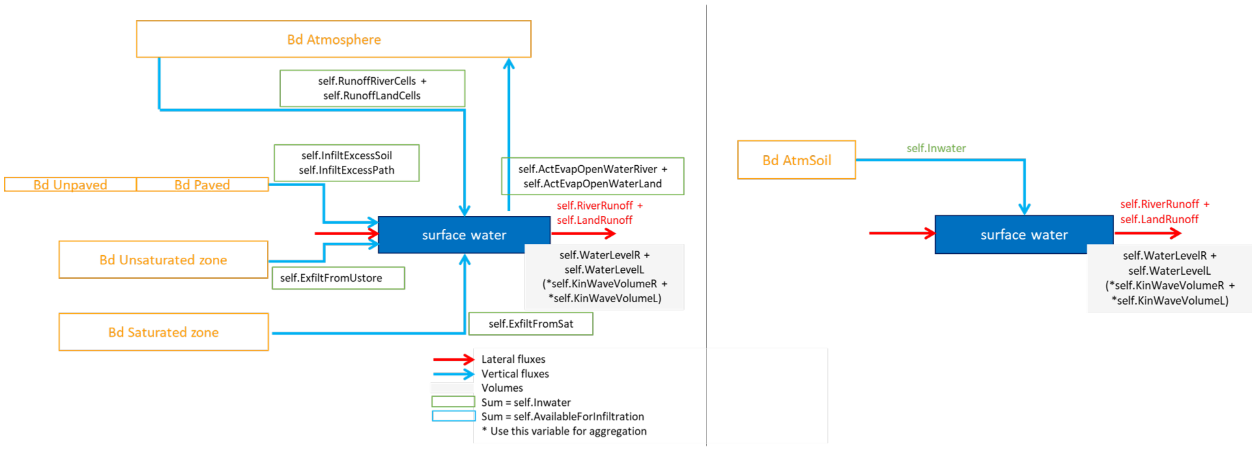


Fig. 33: Compartments and fluxes needed for a Delwaq fraction model (left: with all the fluxes, right: simplified).

The compartments.csv file is a table composed of eight columns (Table 1, columns in red need to be filled with precise keywords used in the coupling script, columns in blue only need to be consistent between the csv files):

- Nr: number of the compartment. This field is not used by the python script of the module but is defined for user information.
- ID: simplified identifier for the compartment, usually a few letters. The user can choose any name but IDs must be consistent with the ones in the fluxes.csv and boundaries.csv files.
- Name: name of the compartment. This field is not used by the python script of the module but is defined for user information.
- wflowVar: wflow_sbm variable representing the volume of the compartment (as in the complete scheme above). If the compartment isn't defined or doesn't have a volume defined in wflow_sbm, for example Sewage or Paved, the keyword ZeroMap can be used. Several wflow variables can be summed up to create one WAQ flux using '+' symbol between variables.
- mapstack: name of the PCRaster or NetCDF mapstack (storage of the output variables). Mapstack are defined in the wflow_sbm ini file by the user. Note that the maximum allowed number of characters is eight. If the compartment doesn't have a volume defined in wflow_sbm, for example Paved or Unpaved, the keyword ZeroMap can be used. Several wflow mapstacks can be summed up to create one WAQ flux using '+' symbol between mapstacks.
- Unit: unit of the compartment volume, usually either mm or m3 for wflow volumes. If the volume is in mm, the script will convert it into m3 which is the standard for Delwaq.
- At1, At2: D-Emission/WAQ attributes of the compartments. Usually attribute 1 specifies if the compartments is active or passive and attribute 2 specifies the type of the compartment (0 for surface water in 1D/2D models… see D-Emission/WAQ documentation). Here it is important that At2 is set to 0 for the Surface Water compartment and other numbers for the others.

Table 1: Example of the compartmemts.csv file for a Delwaq fraction model

Nr	Name	ID	wflowVar	mapstack	Unit	At1	At2
1	SurfaceWater	Sfw	WaterLevelR+WaterLevelL	levKinR+levKinL	m	1	0

The boundaries.csv file, which is not used by the script but for user information, is a table composed of three columns (Table 2, columns in blue only need to be consistent between the csv files):

- Nr: number of the boundary. This field is not used by the python script of the module but is defined for user information.
- ID: simplified identifier for the boundary, usually a few letters. The user can choose any names but IDs must be consistent with the ones in the fluxes.csv and compartments.csv files.
- Name: name of the boundary. This field is not used by the python script of the module but is defined for user information.

Table 2: Example of the boundaries.csv file for a Delwaq fraction model

The fluxes.csv file is a table composed of nine columns (Table 3, columns in red need to be filled with precise keywords used in the coupling script, columns in blue only need to be consistent between the csv files, columns in green are required only for a coupling with an emission model):

- Nr: number of the flux. This field is not used by the python script of the module but is defined for user information.
- Name: name of the flux. This field is not used by the python script of the module but is defined for user information. For a coupling with D-Emission, this field is used to create the parameters name of the hydrology file. An "f" will be added at the beginning of the name if the flux unit is in fraction.

Nr	Name	ID
1	Atmosphere	Atm
2	Unpaved	Unp
3	Paved	Pav
4	UnsaturatedStore	Ust
5	SauratedStore	Sst

- From: simplified identifier for the compartment or boundary where the flow comes from, usually a few letters. The user can choose any name but IDs must be consistent with the ones in the compartments.csv and boundaries.csv files.
- To: simplified identifier for the compartment or boundary where the flow goes to, usually a few letters. The user can choose any name but IDs must be consistent with the ones in the compartments.csv and boundaries.csv files.
- wflowVar: wflow_sbm variable representing the flux (as in Figure 3). If the flux is not defined in wflow_sbm, for example flux between the Sewage compartment and Surface Water in an emission model, the keyword ZeroMap can be used. Several wflow variables can be summed up to create one WAQ flux using '+' symbol between variables.
- mapstack: name of the PCRaster or NetCDF mapstack (storage of the output variables). Mapstack are defined in the wflow_sbm ini file by the user. Note that the maximum allowed number of characters is eight. If the flux is not defined in wflow_sbm, for example flux between the Sewage compartment and Surface Water in an emission model, the keyword ZeroMap can be used. Several wflow mapstacks can be summed up to create one WAQ flux using '+' symbol between mapstacks.
- Unit: unit of the flux, usually either mm or m3/s for wflow fluxes. If the flux is in mm, the script will convert it into m3/s which is the standard for Delwaq or D-Emission.
- EmPointerHyd: specificity of the emission model. In the coupling between wflow and D-Emission, both the pointer file between each segment and actual fluxes data (hydrology.bin file) are saved. While the two files are consistent in Delwaq, this is not the case for D-Emission where some fluxes may only be present in the pointer but not the hydrology file and vice-versa. To take that into account the following keywords are used: P for a flux only present in the pointer file, H for a flux only present in the hydrology file and PH for a flux present in both. Note: D-Emission also needs an additional flow recorded in the hydrology file which is the "TotalFlow" that sums up some of wflow_sbm fluxes to the surface water. If a wflow flux is needed for the calculation of the "TotalFlow", the keywords T, PT, HT or PHT are used depending if the flow is also needed for the pointer and/or hydrology file.
- EmFraction: for emission modelling some fluxes units have to be converted to fraction of the volume of the compartment from which they come from. For those fluxes the keyword 1 is used, else 0.

Table 3: Example of the fluxes.csv file for a Delwaq fraction model

Nr	Name	From	To	wflowVar	mapstack	Unit	EmPointerHyd	EmFraction
1	Precipitation	Atm	Sfw	RunoffOpenWater	POpen	mm	T	0
2	ExcessUnpaved	Unp	Sfw	InfiltExcessSoil	infExUnp	mm	PHT	0
3	ExcessPaved	Pav	Sfw	InfiltExcessPath	infExPav	mm	PHT	0
4	ExfiltrationUStore	Ust	Sfw	ExfiltFromUstore	exfUSt	mm	PHT	1
5	ExfiltrationSStore	Sst	Sfw	ExfiltFromSat	exfSSt	mm	PHT	1

Specificities of the coupling with D-Emission

Setting up an emission model from wflow_sbm is less straightforward than for Delwaq and the coupling script has therefore further requirements needed. For example, an emission model can contain compartments that are not present in wflow_sbm, for example the Sewage compartment (figure below). To add this compartment, a new compartment line in the compartments.csv is added and as for the Paved or Unpaved compartments that don't have any volume, the corresponding wflowVar and mapstack column values are ZeroMap. Then additional fluxes linked to the Sewage compartments are also added. There are two of them: a flux from the Sewage to the Surface Water compartment and a second flux from the Sewage to a Waste Water Treatment Plant boundary. As these fluxes are not defined in wflow_sbm, the corresponding wflowVar and mapstack values are again ZeroMap.

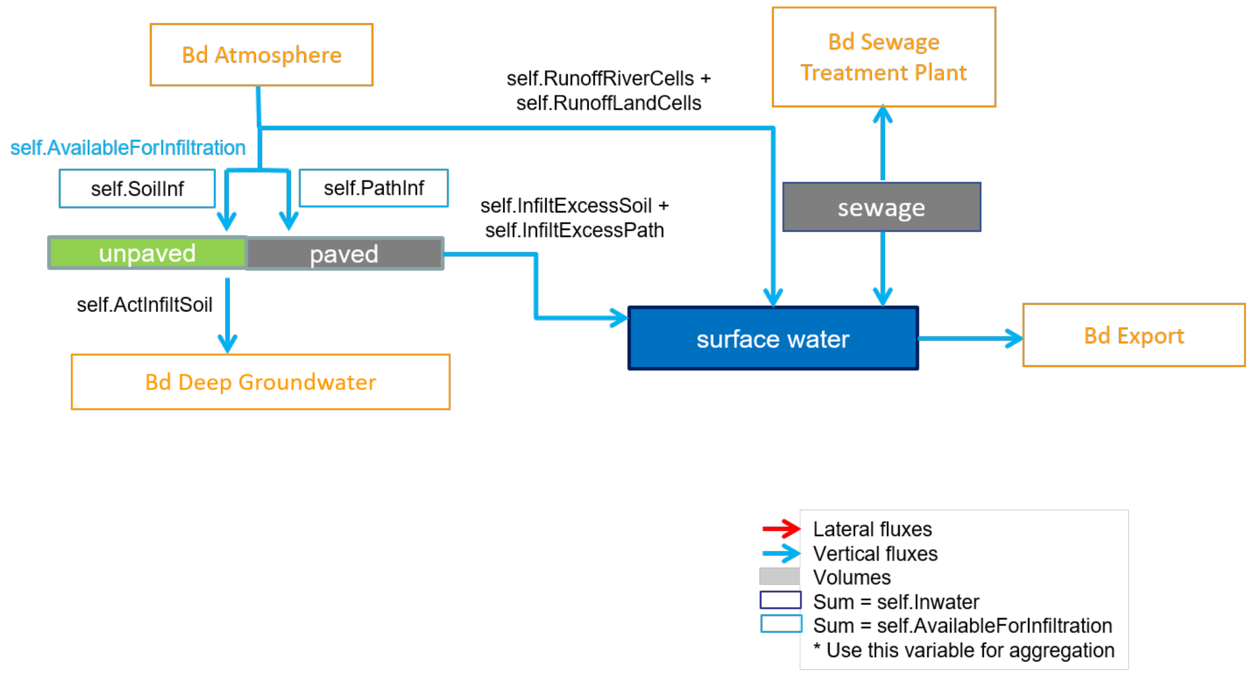

Bd Atmosphere
Bd Sewage Treatment Plant
self.RunoffRiverCells + self.RunoffLandCells
self.AvailableForInfiltration
self.SoilInf
self.PathInf
self.InfiltExcessSoil + self.InfiltExcessPath
sewage
unpaved
paved
self.ActInfiltSoil
surface water
Bd Export
Bd Deep Groundwater
Lateral fluxes
Vertical fluxes
Volumes
Sum = self.Inwater
Sum = self.AvailableForInfiltration
* Use this variable for aggregation

Fig. 34: Example of an emssion model including a Sewage compartment.

Then, contrary to Delwaq, wflow_sbm fluxes are not saved in a flow.dat file that represents exactly the pointer structure but rather in a hydrology.bin file that only saves some of the fluxes. This means that some wflow_sbm fluxes are needed just in the pointer or in the hydrology file. In addition, in the hydrology file, an additional flow recorded is the "TotalFlow" that sums up some of wflow_sbm fluxes to the surface water. These fluxes may already be needed for the pointer or the hydrology file or just for the "TotalFlow". In order to tell the coupling code which flux is needed where, the column EmPointerHyd is added in the fluxes.csv file. This column is filled with specific keywords:

- P: flux only needed for the pointer (for example the flux from the Sewage to the Waste Water Treatment Plant).
- H: flux only needed in the hydrology file.
- T: flux only needed for the calculation of the "TotalFlow" (for example the precipitation falling directly to the surface runoff compartment self.RunoffOpenWater).
- PH: flux needed for the pointer and the hydrology file.
- PT: flux needed for the pointer and "TotalFlow".
- HT: flux needed for the hydrology file and "TotalFlow".
- PHT: flux needed for the pointer, hydrology file and "TotalFlow".

Finally, the last difference is that for D-Emission, some of the saved fluxes are not in m3/s but in fraction of the compartment's volume. To identify such fluxes, the column EmFraction was added to the fluxes.csv table and is filled with 1 if the wflow_sbm flux needs to be converted in fraction of the volume or with 0 if the flux needs to be converted in m3/s.

Specificities of the aggregation option

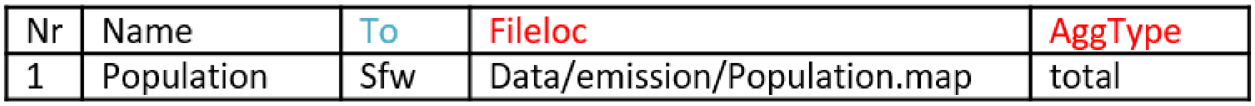
If aggregation of wflow_sbm results from the cell to the subcatchment scale is needed, then the variable representing the volume of the surface water in wflow_sbm must be self.KinWaveVolume. This is because the surface area of the surface water compartment is not always the cell area but can also be the river surface area for wflow river cells.

Adding emissions maps

The coupling can also prepare .inc emission data for D-Emission/WAQ, that are adapted to the wflow schematisation. In order to enable this option, emissions data must be prepared as wflow PCRaster map files and the links to these maps written in an emissions.csv file. The emssions.csv file contains 5 columns (Table 4, columns in red need to be filled with precise keywords used in the coupling script, columns in blue only need to be consistent between the csv files):

- Nr: number of the emission data.
- Name: name of the emission
- To: ID of the compartment where the emission are located.
- Fileloc: the link to the .map emission file. Path is relative to the wflow model casename.
- AggType: if data need to be aggregated to the subcatchment level, identify the aggregation operation (total, average, maximum, minimum).

Table 4: Example of the emissions.csv file



Nr	Name	To	Fileloc	AggType
1	Population	Sfw	Data/emission/Population.map	total

Setting up the wflow_sbm run

If the user wants to aggregate the results from wflow cells to subcatchments, then wflow outputs are not saved as classical map files (netCDF or PCRaster) but as CSV tables where all the different fluxes and volumes are already aggregated. To create the CSV tables, wflow_sbm variables are not saved in the outputmaps section of the ini file but in the different outputcsv sections. Complete settings of these sections are described in the wflow documentation. For a coupling with water quality and emission modelling, only two types of CSV files need to be saved: wflow variables that are sampled at the outlet of the subcatchments and wflow variables that are summed up over the entire subcatchment area.

The corresponding settings for variables summed over the entire catchment area are:

- Type of wflow variables concerned: all the vertical fluxes between compartments and or boundaries, plus the variables representing the compartments volume/storage.
- samplemap: link to the map of the subcatchments where results should be summed up.
- function: type of function to use for the sampling by subcathcments. For the coupling, the sum is needed and function is then total. (Other possible wflow_sbm functions are average, minimum, maximum…).
- List of wflow variables to save with the ".csv" extension.

And corresponding settings for variables sampled at the outlet of the subcatchments are:

- Type of wflow variables concerned: all the lateral fluxes between compartments ie SurfaceRunoff and/or SatWaterFlux. In addition, due to possible conversions of units, the compartments volume/storage are also needed. As two csv files cannot have the same name, the letter g is added to the mapstack name. For example, the csv tables of the Saturated Store water depth would be "SStore.csv" for the sum over subcatchments and "SStoreg.csv" for the sampling at the gauges/subcatchments outlets.

Warning: If the aggregation of wflow_sbm results from cells to subcatchments is used, the wflow variable used for the surface water compartment is self.KinWaveVolume instead of self.WaterLevel

Setting up the wflow_emwaq run

As for wflow_sbm, wflow_emwaq also has its own ini file wflow_emwaq.ini. The different sections are (see the template for examples):

- inputcsv: similar to the inputmapstacks section in wflow_sbm.ini, it lists the links to the csv tables. By defaults, the csv are placed in a new csv folder in the model directory of the wflow case. Required arguments are:
  - sepcsv: separator between each field of the csv file. Usually "," or ";"
  - compartments: link to the compartments.csv file.
  - boundaries: link to the boundaries.csv file.
  - fluxes: link to the fluxes.csv file.
  - emissions: link to the emissions.csv file. If no path is specified the coupling won't produce any emission data.
- run: similar than in wflow_sbm.ini. Required fields are:
  - starttime: beginning of the simulation.
  - endtime: end of the simulation.
  - timestepsecs: length of the time step in seconds.
  - runlengthdetermination: either intervals or steps depending on the format of the timestep.
- model: maps and options to use for the model run.

  Type of the results saved by wflow_sbm. Indicate which type of file the coupling should read from. Options are:
  - input_type: type of wflow outputs to read. Can be "netcdf" or PCRaster "map" files or "csv" tables. The csv tables option is the one to use if aggregation of results to subcatchments is chosen.
  - netcdfinput: name of the possible NetCDF output file of wflow_sbm is required. Can be left empty if input_type is either map or csv.

  Options for wflow_emwaq run (0 to turn off or 1 to turn on):
  - write_ascii: dynamic data for D-Emission/WAQ are saved in binary files. If this option is on, an ASCII copy will also be created. Default is 0.
  - write_structure: if on, structure data for D-Emission/WAQ are produced. Note that this option cn be used without previsouly running wflow_sbm. Default is 1.
  - write_dynamic: if on, dynamic data for D-Emission/WAQ are produced. Default is 1.
  - fraction: if on, produce additional structure files used for Delwaq fraction mode. Default is 0.
  - emission: if on, a coupling for D-Emission will be prepared. If off, a coupling for Delwaq will be prepared. Default is 0.

  - fews: if on, additional files to use for FEWS or deltashell GUI will be produced. Default is 0.
  - aggregation: if on, results from wflow_sbm will be aggregated from the cell to the subcatchment scale.

  Input or output staticmaps of wflow_sbm to use. The maps needed are:

  - wflow_subcatch: map of wflow subcatchments. Serves as basis to define the extent of the model grid. Map used for the aggregation of results option.
  - wflow_ldd: wflow local drain direction map used for pointer creation.
  - wflow_gauges: map with points of interest where D-Emission/WAQ results will be saved. Map used for the aggregation of results option.
  - wflow_reallength: map with the cell size (corrected with real latitude).
  - wflow_pathfrac: fraction of the land part of the cell that is paved.
  - wflow_waterfrac: fraction of the open water part of the cell.

  Settings for D-Emission/WAQ monitoring points or areas. Indicates the segments or zones where the emission water quality models sum and save the results.

  - mon_points: selection of segments where results will be saved. Option segments will save the results for every segment of the model. Option gauges will save it for the compartments belonging to the points selected in the wflow_gauge map. The gauges option is not handled if aggregation option is on.
  - mon_areas: selection of monitoring areas to aggregate results. Option subcatch will save results for the segments present in each subcatchment of the wflow_subcatch map. Option compartments will save it for the segments belonging in the same compartment type (e.g. by surface water). The subcatch option is not handled if aggregation option is on.

  Settings for the numbering of the model boundaries. Indicates if the boundary ID is set by boundary type (ex: one ID for atmosphere) or if it is set by boundary type and for each segment.

  - bd_id: the default numbering is one ID per boundary type. If bd_id is set to segments then one ID per boundary type and per segment will be created.

  Settings for the Delwaq run input file (.inp)

  - template_ini: the path to a template Delwaq input file to add to the coupling results. By default, no file is added.

  Additional parameters from wflow_sbm needing for data processing. Now only one parameter is needed:

  - UStoreLayerThickness: same as in wflow_sbm.ini. It is needed in order for the python script to be able to handle the instates files of all the layers of the unsaturated store.

- **outputstat_**: similar to the **outputcsv_** section of wflow_sbm ini file. This section prepare a list of files for Delwaq to generate statistics during its run (inputs for block 10). If several statistics have to be computed, several **outputstat_** sections can be created (outputstat_0, outputstat_1…).

  - name: name of the created file. Resulting file will be “B10_nametime.inc”
  - start-period: the start time for which statistics are calculated (YYYY/MM/DD-hh:mm:ss).
  - end-period: the end time for which statistics are calculated (YYYY/MM/DD-hh:mm:ss).
  - output-operation: type of operation (example: STADSC).
  - substance: substance for which statistics are calculated. Several substances can be added using the names substance1, substance2…

Running the coupling

Like wflow_sbm, the whole coupling process can be run either from the command line or batch file(s). First wflow_sbm is run. The minimum command line requires:

- The link to the wflow_sbm script.
- -C option stating the name of the wflow case directory.
- -R option stating the name of the directory of wflow_sbm outputs.

Then wflow_emwaq is run. The minimum command line requires:

- The link to the wflow_emwaq script.
- -C option stating the name of the wflow case directory (idem sbm).
- -R option stating the name of the directory containing wflow_sbm outputs (idem sbm).
- -D option stating the name of the directory of wflow_emwaq outputs.
- -c option stating the name of wflow_emwaq ini file.

Additional run options corresponding to the one set up in the model section of the ini file are:

- -i for writing binary and ASCII files.
- -u for writing structure files.
- -y for writing dynamic files.
- -f for writing additional fraction files.
- -e for an emission or water quality coupling.
- -F for writing additional files for FEWS or deltashell GUIs.
- -a for aggregation of results from cells to subcatchments.

Finally, all the files produced by wflow_emwaq need to be included in the main input file from D-Emission/WAQ and the emission or water quality model can be run. An example of the command lines needed to run both wflow python scripts is:

```
activate wflow
python wflow\wflow_sbm.py -C Rhine -R SBM
python wflow\wflow_emwaq.py -C Rhine -R SBM -D Rhine\WAQ -c wflow_emwaq.ini
```

Note: if only the schematisation is of interest and not the dynamic fluxes, the coupling can also be run without first running wflow_sbm. In that case, the writing structure files option should be turned on and the writing dynamic files option turned off.

Outputs of the wflow_emwaq module

The wflow_emwaq module produces several folders in the defined run directory. These are the debug folder which contains additional maps or files produced during the coupling process, the includes_deltashell folder which includes all the structure files listed in Table 4 and the includes_deltashell folder which includes all the data files listed in Table 5. The type of files produced depend if the coupling is made for an emission model or a water quality model with or without fraction calculation.

Table 4: List of structure files produced by wflow_emwaq

File name	Emission	Delwaq	Fraction mode
B1_sublist.inc			X
B1_timestamp.inc	X	X	
B2_monareas.inc	X	X	
B2_nrofmon.inc	X	X	
B2_outputtimes.inc	X	X	
B2_stations.inc	X	X	
B2_stations-balance.inc	X	X	
B2_timers.inc	X	X	
B3_attributes.inc	X	X	
B3_nrofseg.inc	X	X	
B4_nrofexch.inc	X	X	
B4_pointer.inc	X	X	
B5_bounddata.inc			X
B5_boundlist.inc	X	X	
B8_initials.inc			X
B10_simtime.inc	X	X	
geometry.inc	X		
hydrology.bin	X		
nrofsegl.inc	X		

Table 5: List of data files produced by wflow_emwaq

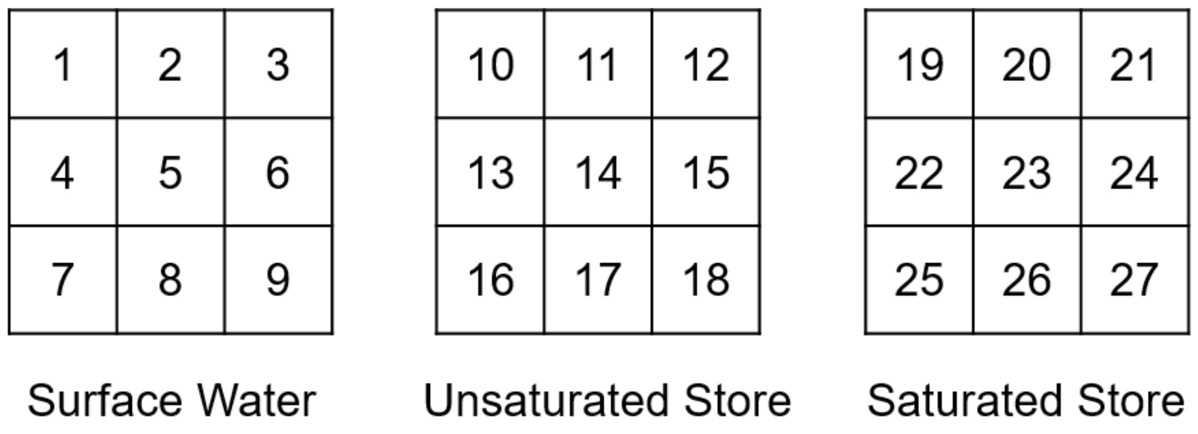
File name	Emission	Delwaq
area.dat		X
flow.dat		X
manning.dat		X
surface.dat		X
velocity.dat		X
volume.dat		X
hydrology.bin	X	

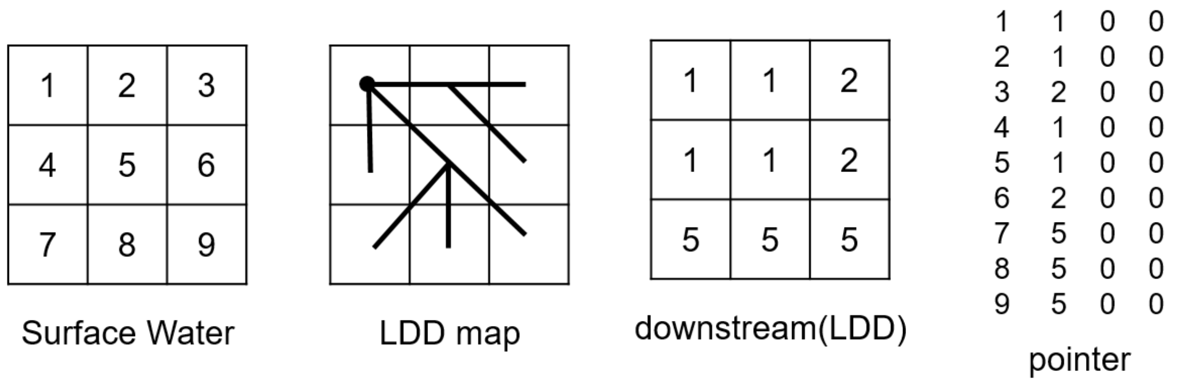
How the coupling script works

Creation of the pointer

To create the pointer file needed by the emission/water quality models, the wflow_emwaq scripts first gives a unique ID number to each of the active cell of the wflow_sbm grid (cells that are in the area modelled). Then the script reads the compartments.csv table. The unique IDs previously created are then assigned to the first compartment listed in the table. Then if there are other compartments listed in the table, the unique IDs are copied and shifted by the total number of cells. Compartments are therefore read one by one and the different segment IDs are created step by step accordingly. For example, for a wflow model with 9 active grid cells, if the compartments.csv table contains three lines starting from “Surface Water” to “Unsaturated Store” and finishing with “Saturated Store”, the corresponding IDs of the D-Emission/WAQ model would be:

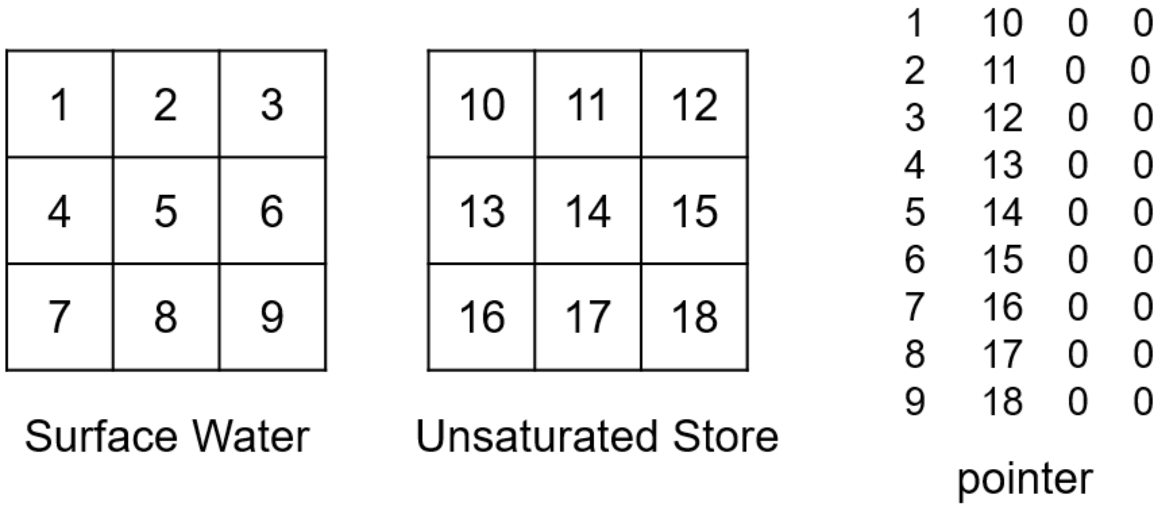
Then the fluxes and links to the fluxes are handled. Instead of reading the fluxes list defined in the corresponding csv table one by one like for the compartments, they are instead read by type starting with the lateral fluxes (equal name in the From and To column of the fluxes.csv table). The direction of the flow within the segment of the same compartment is defined with the local drain direction (LDD) map from wflow_sbm. With this map, the downstream cell ID is identified and the pointer file construction is starting. Back to our example, the first lines of the pointer would then be:


1
2
3
4
5
6
7
8
9
Surface Water
10
11
12
13
14
15
16
17
18
Unsaturated Store
19
20
21
22
23
24
25
26
27
Saturated Store


1
2
3
4
5
6
7
8
9
Surface Water
LDD map
1
1
2
1
1
2
5
5
5
downstream(LDD)
1 1 0 0
2 1 0 0
3 2 0 0
4 1 0 0
5 1 0 0
6 2 0 0
7 5 0 0
8 5 0 0
9 5 0 0
pointer

The next step is to spot the outflows of the model (flow staying in the same cell) and turn them into outflow boundaries in the pointer. In the pointer, boundaries are segment with negative values. Outflows IDs are then given a new ID starting from -1. In the example, the first line of the pointer will give: 1 -1 0 0.

The second type of fluxes treated are the vertical fluxes between compartments (names in the From and To columns of fluxes.csv both corresponds to names in the ID column of compartments.csv). Direction of the flow corresponds then to the IDs of the compartments of the same cell. Vertical fluxes between compartments are read in the same order of appearance in the fluxes.csv file. For example, if we take a flow going from the surface water to the unsaturated store, the corresponding lines of the pointer will be:

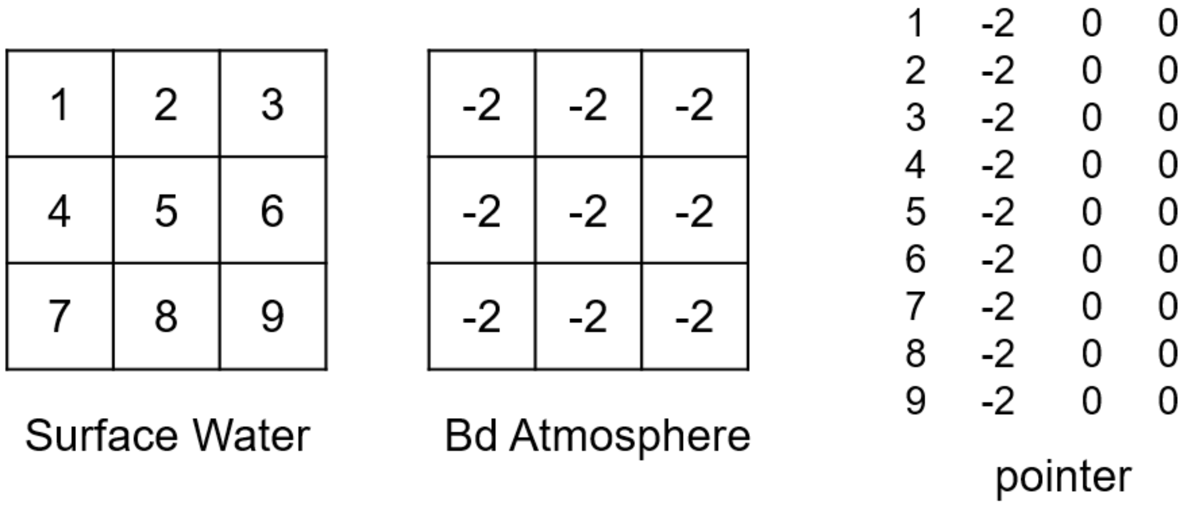


1	2	3
4	5	6
7	8	9

10	11	12
13	14	15
16	17	18

```
1  10  0  0
2  11  0  0
3  12  0  0
4  13  0  0
5  14  0  0
6  15  0  0
7  16  0  0
8  17  0  0
9  18  0  0
```

Finally, the vertical fluxes between compartments and boundaries are treated (names in either the From or To column of fluxes.csv corresponds to a name in the ID column of compartments.csv). By default, the coupling gives one ID per boundary type. If the bd_id option is set, IDs for boundaries are defined in the same way as for the compartments (one ID per boundary per active cell). Numbering starts with an offset corresponding to the number of outflows. Vertical fluxes between compartment and boundary are also read in the same order of appearance in the fluxes.csv file. For example, if we have a flux going from the boundary atmosphere to the surface water (precipitation), the corresponding IDs for the atmosphere boundary and pointer lines, by default, would be (there was already one outflow boundary):

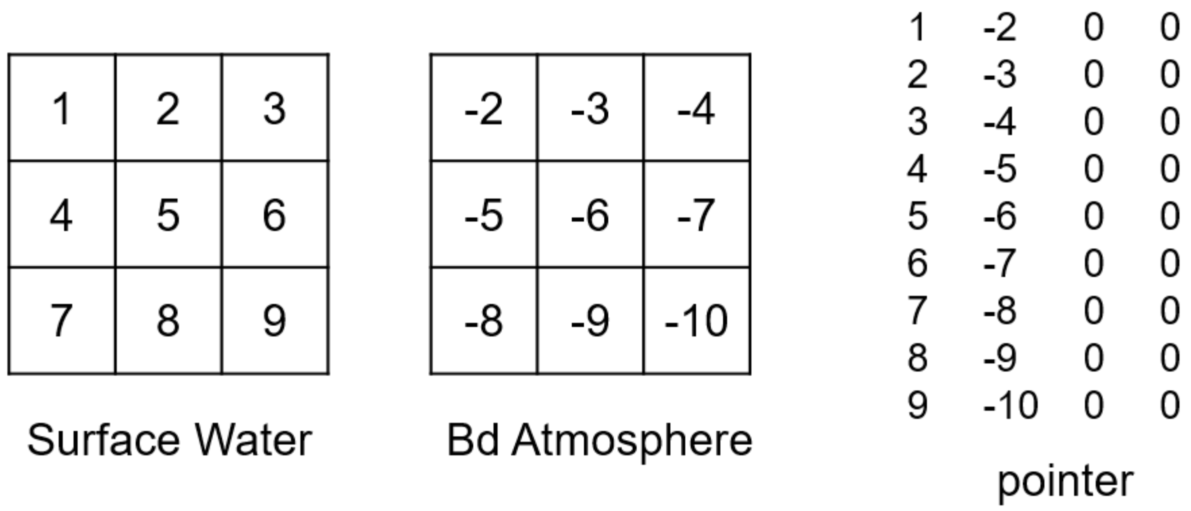


1	2	3
4	5	6
7	8	9

-2	-2	-2
-2	-2	-2
-2	-2	-2

```
1  -2  0  0
2  -2  0  0
3  -2  0  0
4  -2  0  0
5  -2  0  0
6  -2  0  0
7  -2  0  0
8  -2  0  0
9  -2  0  0
```

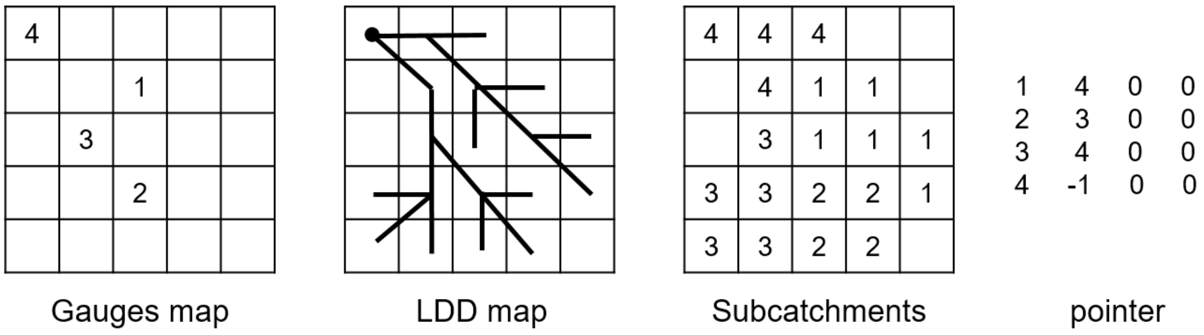
And if bd_id is set to segments:



1	-2	0	0
2	-3	0	0
3	-4	0	0
4	-5	0	0
5	-6	0	0
6	-7	0	0
7	-8	0	0
8	-9	0	0
9	-10	0	0

Specificities of the aggregation option

When wflow_sbm outputs are handled for all cells by the coupling, IDs are given to the cells row by row starting from the top left corner to the bottom right corner of the grid. If wflow results are aggregated by subcatchments IDs depends on the wflow subcatchment map that is used. Subcatchments ID are then sorted by ascending order rather than location on the map. For lateral flows, the downstream subcatchment is determined using the downstream cell of the outlet (corresponding to the wflow gauge map). If we take an example with four subcatchments, the pointer for the lateral flux (after replacing the outflow by a boundary) will then be:



1	4	0	0
2	3	0	0
3	4	0	0
4	-1	0	0

If there are several compartments, they are then handled in the same way than for cells. The fluxes are also treated in the same order, first the lateral fluxes, then fluxes between two compartments and finally fluxes with boundaries.

Creation of the other structure files

Once the pointer is created, the other structure files are created using a specific function from the coupling script. These functions usually use inputs coming from the pointer file creation.

Creation of the flow file (either flow.dat or hydrology.bin)

For a coupling with Delwaq for each wflow cells, the flow.dat file is constructed in the very same way as the pointer. Flows are read from lateral, to between 2 compartments to between boundaries and compartments and then in the order of their definition in the fluxes.csv table. For each timestep, the corresponding wflow netcdf or map output will be read, and fluxes values saved row by row from the top left corner to the bottom right corner, corresponding to the numbering of the wflow cells IDs.

For a coupling with D-Emission for each wflow cells, as some fluxes are present in the pointer but not the hydrology file, another definition file hydrology.inc is created to give the order of the fluxes saved in the hydrology.bin file and the corresponding segments affected in the pointer file. As the order of the fluxes definition is not important then, fluxes are saved one by one in the order of their definition in the fluxes.csv table (and not from lateral to vertical fluxes). The "TotalFlow" is then added at the end. As for a Delwaq coupling, for each timestep, the corresponding wflow netcdf or map output will be read and fluxes values saved row by row from the top left corner to the bottom right corner, corresponding to the numbering of the wflow cells IDs.

If the aggregation option is on, the entire aggregated CSV tables of the different wflow variables are read in the order of the flow definition in the pointer file for Delwaq (lateral then vertical) and in the order of their definition in the fluxes.csv table for D-Emission. As wflow already sorts the saved aggregated variables by ascending number of the subcatchment IDs, no restructuration of the variables CSV tables is needed.

wflow_emwaq module documentation

1.9 BMI: Basic modeling interface

1.9.1 The wflow_bmi interface

Introduction

In order to simplify conversion of an existing model to a reusable, plug-and-play model component, CSDMS has developed a simple interface called the Basic Model Interface or BMI that model developers are asked to implement. Recall that in this context an interface is a named set of functions with prescribed function names, argument types and return types. The BMI functions make the model self-describing and fully controllable by a modeling framework.

See also: http://csdms.colorado.edu/wiki/BMI_Description

This is the first implementation of the BMI for the wflow pcraster/python models

Configuration

Mapping of long_var_name to model variables not yet implemented. The long_var_name should be model for names for now

wflow_bmi module documentation

1.9.2 The wflow_bmi_combined interface

Introduction

In order to simplify conversion of an existing model to a reusable, plug-and-play model component, CSDMS has developed a simple interface called the Basic Model Interface or BMI that model developers are asked to implement. Recall that in this context an interface is a named set of functions with prescribed function names, argument types and return types. The BMI functions make the model self-describing and fully controllable by a modeling framework.

See also: http://csdms.colorado.edu/wiki/BMI_Description

The wflow_bmi_combined module implements a class that connects 2 or more python bmi modules and exports those to the outside as a single bmi model.

- A @ character is used to separate the module from the variable. For example, the variable Flow in module wflow_hbv becomes wflow_hbv@Flow in the combined model (it was Flow in the single interface)
- The individual models can run in a separate case dir or can be combined into one directory (and share maps that are identical)

The bmi2runner.py script can be used to run a set of combined models, it is documented seperately.

wflow_bmi_combined module documentation

1.9.3 bmi2runner

Introduction

bmi2runner.py is a simple script that runs two or more wflow modules connection via the BMI interface (the combined version). A configfile is used to control which models to start as well as the exchange of data between the models.

The config file contains a list of models configured with the name of the wflow modules to run and the ini file that is used by the model. Furthermore, in the exchanges section the data flows from model to model are configured.

```
[models]
# module name = path to config of module relative to the dir of this ini file
wflow_sbm=wflow_sbm/wflow_sbm_comb.ini
wflow_routing=wflow_routing/wflow_routing_comb.ini

[exchanges]
# From_model/var -> To_model/var
wflow_sbm@InwaterMM=wflow_routing@IW
```

To setup a combined model you should first configure and setup the individual models. They can be setup in separate case directories or they can be merged in one case directory. Each model should have it's own config/ini file. The following principles apply when using the bmi2runner script:

- the models are executed in the order they are listed in the models section
- the variables are get/set in the order they appear in the exchanges section
- the script runs explicitly, no iteration is performed

Example

In the examples directory the file bmirunner.ini is present. You can use this to run a combined wflow_sbm/wflow_routing model. Start this up using the following command (in the examples dir):

```
bmi2runner.py -c bmirunner.ini
```

A second example runs wflow_sbm next wflow_routing followed by the wflow_floodmap module:

```
bmi2runner.py -c bmirunner-floodmap.ini
```

The contents of the ini file (bmirunner-floodmap.ini) is given below:

```
[models]
wflow_sbm=wflow_rhine_sbm/wflow_sbm.ini
wflow_routing=wflow_routing/wflow_routing_BMI.ini
wflow_floodmap=wflow_routing/wflow_floodmap_BMI.ini

[exchanges]
# From_model.var -> To_model.var
wflow_sbm@InwaterMM=wflow_routing@IW
wflow_routing@WaterLevel=wflow_floodmap@H
```

In this case the floodmap module uses the same directory as the routing module (but a different config file).

bmi2runner script

1.10 Using the wflow modelbuilder

1.10.1 wflow modelbuilder tutorial

Note: UNDER CONSTRUCTION

The wflow modelbuilder is a new tool with which you can set up a wflow model in a few simple steps. The default setup of the wflow modelbuilder is fully based on global data sets. The modelbuilder uses a set of tools called hydro-engine (https://github.com/openearth/hydro-engine), which are built on top of Google Earth Engine (https://earthengine.google.com/).

Installation

The wflow modelbuilder is part of the standard wflow distribution.

You can download the wflow source code from https://github.com/openstreams/wflow (follow the wflow installation instructions). In due time the wflow modelbuilder will also be available as an executable and you will only have to download the wflow executables.

Set up a wflow model

Once you have downloaded and installed wflow, the modelbuilder is available in this location:

```
<your_wflow_folder>/Scripts/wtools_py/modelbuilder.py
```

You can run the modelbuilder script from the command line or in a batch file. The script uses the settings.json file for the location of the model. To run the modelbuilder script with python, use the following command:

```
python modelbuilder.py --geojson-path settings.json
```

The settings.json file must be present. The modelbuilder comes with a default settings.json file. Its contents look like this:

```
{
  "type": "FeatureCollection",
  "features": [
        {
          "type": "Feature",
          "properties": {},
          "geometry": {
                "type": "Point",
                "coordinates": [
                  7.466239929199218,
                  50.31565429419649
                ]
          }
        }
  ]
}
```

In this default settings.json a set of point coordinates is specified; the coordinates in the example are for the Moselle catchment in Germany, a tributary of the Rhine river. To make your own settings.json file, you can use the website geojson.io. This is a website that can be used to easily select a location and create your own settings.json file. Alternatively, you can change the coordinates in the default settings.json file. This settings.json must be present for the modelbuilder to know the location of the model.

Furthermore you can specify the following options:

```
--geojson-path          Path to a GeoJSON file with the geometry that needs to be path␣
↪of the model

--cellsize              Desired model cell size in decimal degrees (default=0.01)

--name                  Name of the wflow case (default=wflow_<modelconcept>_case)

--model                 Name of the wflow model concept (options: sbm, hbv,␣
↪w3ra)(default=sbm)

--timestep              Model time step for hbv (options: hourly, daily) (default=daily)

--case-template         Name of the template wflow case (default=wflow_<modelconcept>_
↪template)

--case-path             Path where both the template and created case reside(default is␣
↪the current directory)

--fews/--no-fews        Flag indicating whether the wflow case is part of a Delft-FEWS␣
↪setup (default=no-fews)

--fews-config-path      Path to the Delft-FEWS config directory (to save default FEWS␣
↪states) (default=Config)

--dem-path              Path to a local DEM if available

--river-path            Path to a local river shapefile if available

--region-filter         Tell hydro-engine which model area to pick, by default this is␣
↪everything upstream of the provided geometry, but it is also possible to get only the␣
↪current catchment (catchments-intersection), or just exactly the provided geometry␣
↪(region), like your own catchment polygon (options: catchments-upstream, catchments-
↪intersection, region)(default=catchments-upstream)
```

Example:

```
python modelbuilder.py --geojson-path settings.json --name wflow_moselle --cellsize 0.01
```

Run this command from the command line or in a batch file, and you will have your model.

The generated model structure looks like this:

```
data\
inmaps\
instate\
intbl\
mask\
```

(continues on next page)

(continued from previous page)

```
run_default\
staticmaps\
wflow_sbm.ini
```

To run the wlfow model, you need the staticmaps and intbl directories and the wflow_sbm.ini file. Also the inmaps and the instate directories are needed to run the model, but these are not filled yet. By default, results of your model run are stored in the run_default directory, and this directory including all its subfolders is required if you run the model within FEWS.

In the mask folder you will find the mask that is used to clip the model, and the grid definition in FEWS format (in grid.xml), which you can copy-paste into the Grids.xml file in your FEWS configuration. In the data folder you will find the data that was used to generate the model, after clipping it from the global data: geojson files for the catchments and rivers, and raster files for the DEM and the parameter maps.

The wflow_sbm.ini file is the file with configuration settings that is needed to run the wflow-sbm model. This is an example file – please change the settings in the ini file according to your specific model setup (see *wflow_sbm|hbv.ini file*).

Model data

Where does the data come from? This default setup of the wflow modelbuilder is fully based on global data sets. Below you find the specifications of the global data sets used.

Catchment delineation

The clipping of the global maps is done based on the model area. The model area is based on the HydroBASINS subcatchments, level 9 (http://hydrosheds.org/page/hydrobasins). The modelbuilder determines within which HydroBASINS subcatchment the coordinates are located that you specified in the settings.json file, and queries all upstream catchments as a single or multiple polygons. These subcatchments together define the area of your model. The data sets described below are all clipped based on this area.

Rivers

For the river network, the HydroSHEDS drainage network is queried as polylines (http://hydrosheds.org/).

Optionally, a local or improved river vector file (shapefile, geojson, etc.) can be provided to the modelbuilder with the option `--river-path`. If a local river vector file is specified, this will be used instead of the default global river file.

DEM

For the elevation data the digital elevation model (DEM) used is SRTM v4, 30m (https://www2.jpl.nasa.gov/srtm/)

Optionally, a local or improved Digital Elevation Model (DEM) can be provided to the modelbuilder with the option `--dem-path`. If a local DEM is specified, this will be used instead of the default global DEM.

Land use

For land use the 0.5 km MODIS-based Global Land Cover Climatology map by the USGS Land Cover Institute (LCI) is used (https://landcover.usgs.gov/global_climatology.php). This land cover dataset consists of 17 different classes for land cover types. The legend for this land cover map is also provided in the template case (and copied to your wflow model) in data/parameters/lulegend.txt

LAI

LAI (Leaf Area Index) maps for the wflow-sbm model are stored in the staticmaps/clim directory. These are twelve maps with monthly average LAI, based on combined AVHRR and MODIS data, derived from Liu et al. 2012 [Liu2012], calculated as averages over 1981-2011.

Soil type

A soil map indicating major soil texture types is also downloaded with the modelbuilder (wflow_soil.map), which is derived from the Harmonized World Soil Database (HWSD) (FAO et al. 2009 [FAO2009]). The legend for this soil dataset is also provided in the template case in data/parameters/wflow_soil.csv. In the current setup with global data, this soil map is not used, since all soil-based parameters are specified as rasters. It can however be useful if you want to differentiate parameters in the intbl directory based on soil type, or if you want add more parameters as .tbl files.

Model parameters

Parameters linked to LAI:

- Specific leaf storage: determined from Liu 1998 [Liu1998]
- Storage on the woody part of the vegetation (branch and trunk storage): determined from Liu 1998 [Liu1998]
- Extinction coefficient: Van Dijk & Bruijnzeel 2001 [VanDijk2001]

Parameters linked to soil and land use:

- Parameters provided as maps in the staticmaps directory: based on Dai et al. 2013 [Dai2013] and Shangguan et al. 2014 [Shangguan2014]
- Other parameters provided as intbl files: the parameters that are not specified as rasters, are given in the intbl directory as .tbl files, which can be linked to either land use, soil type or subcatchment (see *Input parameters (lookup tables or maps)*). For these parameters a default value or values have been established.

It is important to note that with the modelbuilder setup you can easily generate a functioning model, including the model structure and all the rasters and other files you need, resampled to your model resolution. However, this results by no means in a calibrated model. The parameter maps and tables are a best first estimate based on global datasets, but most likely need tweaking for application in a regional- or local-scale model.

Current limitations

At the moment it is only possible to set up a model with the modelbuilder in the WGS84 coordinate system (EPSG:4326).

References

1.11 Release notes

1.11.1 2019.1

- Wflow_sbm, redesign of lateral subsurface flow because of a bug that resulted in an overestimation of lateral subsurface flux:
  - The kinematic wave approach is used to route subsurface flow laterally.
  - Kinematic wave solution for surface (overland and river) and subsurface flow added to wflow_funcs, and used by wflow_sbm. Numba (open-source JIT compiler) is used to accelerate these functions. This replaces the PCRaster kinematic wave solution.
  - A seperate kinematic wave reservoir for river (self.RiverRunoff) and overland flow (self.LandRunoff)
  - Lateral subsurface flow and overland flow feed into drainage network (channel) based on ratio slopes (slope upstream cell / (slope upstream cell + slope cell))
  - Option to provide land slope and river slope (staticmaps)
  - Removed sub-grid runoff generation
  - Removed re-infiltration of surface water
- Wflow_hbv uses kinematic wave solution for surface flow available in wflow_funcs (replacement of PCRaster kinematic wave solution)
- Option to enable iteration (smaller time steps) for kinematic wave solution for surface runoff (wflow_hbv and wflow_sbm)
- Added hydrological model wflow_w3 to wflow framework, based on Australian Water Resources Assessment Landscape model (AWRA-L), an improved version compared to wflow_W3RA.
- Added hydrological model wflow_stream to wflow framework (STREAM (Spatial Tools for River Basins and Environment and Analysis of Management Options)).
- Added experimental version of wflow_sediment to simulate soil erosion and delivery to the river system.
- Added the wflow_emwaq module: provides a set of functions to create and link a Delft3D-WAQ or D-Emission model to a wflow model.
- Update of lake modelling for wflow_hbv and wflow_sbm (including also Modified Puls Approach) in wflow_funcs
- Update of glacier model (used by wflow_hbv and wflow_sbm)
- Parameter Manning N of river can be linked to streamorder for wflow_hbv
- setuptools_scm used for version wflow package

1.11.2 2018.1

- -T and -S options now use date/time strings to better support netcdf
- date/time handling improved
- fews adapter and support simplified:
  - fewsrun removed
  - adapter need less arguments
- netcdf handling improved:
  - check for date/time in input and determines offset if there is a problem.
  - netcdf states now work (checks for date/time)
- wflow_sbm:
  - Feddes reduction of root water uptake
  - Improved/fixed handling of open water and soil evaporation
- Added hydrological model PCR-GLOBWB ((PCRaster Global Water Balance) version v2.1.0_beta_1) (wflow_pcrglobwb) to wflow framework
- Added hydrological model SPHY (Spatial Processes in HYdrology) version 2.1 to the wflow framework
- Simple reservoirs: possible to include reservoir areas as static map
- Parameter Manning N of river can be linked to streamorder for wflow_sbm
- HBV lake functionality also available in wflow_sbm

1.11.3 2017.01

- wflow_sbm renamed to wflow_sbm_old
- wflow_sbm2 renamed to wflow_sbm:
  - This breaks older models. You will need to regenerate initial conditions. In the new model the unsaturated part of the soil can be split into several layers. With one layer the results should be the same but the name of the state file is now UStoreLayerDepth_0.map for the first layer (or for the whole layer if only one layer is defined). It used to be UStoreDepth.map
- Changes to wflow_hbv:
  - Quickflow calculation
  - Addition of lake modelling (based on HBV96, complex reservoirs)
  - Simple reservoirs functionality also available in wflow_hbv
- Added rice crop growth model LINTUL (wflow_lintul) to wflow framework and coupled to wflow_sbm (BMI)
- Added irrigation of rice fields to wflow_sbm

1.11.4 2016.04

Note: Several none-backwards compatible changes will be part of this release. Use 2016.03 for older models

- update soil names in sbm to the sbm2 names:
  - FirstZoneKsatVer -> KsatVer
  - FirstZoneMinCapacity -> SoilMinThickness
  - FirstZoneCapacity (FirstZoneThickness) -> SoilThickness
  - FirstZoneCapacity -> SoilWaterCapacity
  - FirstZoneFlux -> SatWaterFlux
  - FirstZoneDepth -> SatWaterDepth
- [model]reinit moved to [run]reinit (same for fewsrun) in all models
- added [rollingmean] section in framework
- updates to topoflex
- updates to BMI framework
- updates to netcdf reader. [model] modeltype= can be specified
- updated wflow_hbv to better resemble hbv-96 for the upper zone when kquickflow is determined (and not specified directly). This may break older calibrations

1.11.5 2016.03

- last tag before moving to new names in SBM
- irrigation added to SBM/SBM2

1.11.6 2016.02

- added better BMI support
- added bmi2runner and wflow_bmi_combined
- updated date/time framework
- added wflow_topoflex model
- added reservoir support to wflow_routing
- added support for other functions apart from averaging in output of time series
- wflow_delwaq support netCDF files
- wflow_sbm2 updated

1.11.7 2015.02

- Updated netcdf support -> now needs the pyproj module
- Updated wflow_routing with two-layer (floodplain)
- Redone multiplication of parameter framework. Now possible via ini and BMI
- Added .mult postfix for tbl files to apply multiplication (untested)
- Added bmi support
- wflow_sbm s-curve sharpness of height distribution now based on upper and lower half of the distribution (average)
- Added separate soil evap and RootZoneSoilMoisture to wflow_sbm
- Added support in wflow_SBM to get vegetation parameters for Gash from LAI
- Moved non-mature scripts to SandBox dir
- Added unit tests for wflow_sbm, hbv and bmi
- Started with wflow_sbm2 -> New version with new (more logical) variable names. This version will also have a better unsaturated zone (for G4INDO project). New Names:
  - FirstZoneKsatVer -> KsatVer
  - FirstZoneMinCapacity -> SoilMinThickness
  - FirstZoneCapacity (FirstZoneThickness) -> SoilThickness
  - FirstZoneCapacity -> SoilWaterCapacity
  - FirstZoneFlux -> SatWaterFlux
  - FirstZoneDepth -> SatWaterDepth

1.11.8 2015.01

- support for scalar input to wflow_sbm/hbv has been removed.
- added wf_updateparameters to framework. This allows the user to set parameters to update in the ini file but also list them in the parameters function in the model itself. This functionality should replace all manual reading of forcing data and static parameters

1.11.9 Version 1.0 RC7

unsupported interim release

- added HBV type lower zone to wflow_sbm. Use MaxPercolation > 0 to use this zone. MaxLeakege > 0 will send water outside of the model
- Test version of the wflow_W3RA model
- Made two lateral flow options for sbm
- Stopped support for pcraster version 3 and python 2.6
- removed all the try/except from importing wflow. Now you NEED to install wflow as a package
- Added seperate wflow_routing module that includes the kinematic wave routing. This part will be removed from the wflow_sbm and wflow_hbv models

- Added check in gash interception not to have more interception than available potential evap
- Fixed capillary rise calculation to include timestep. This means that sub-daily models may need to be recalibrated

1.11.10 Version 1.0 RC5

unsupported interim release

- netcdf reading and writing added (filename should be configured in ini file, framework section: netcdfoutput, netcdfwritebuffer, netcdfinput)
- summary sections (summary, summary_max, symmary_avg, ect) added to ini file to save maps at end of run
- added option to save flow per subcatchment by setting pits at the end of each subcatchment in the ldd
- added new tbl file for wflow_sbm (et_reftopot.tbl). Used to covert reference ET to potential ET. Set to 1 by default
- better representation of open water ET in wflow_sbm
- wflow_adapt can now convert log files to XML for Delft-FEWS

1.11.11 Version 1.0 RC4

unsupported interim release

- tss (and csv) output refactored. The ini file can now hold multiple outputtss sections each with a diffrent maps for extracting/averaging

1.12 Linking wflow to OpenDA

Warning: This is experimental and incomplete

1.12.1 Prerequisites

- download the Wflow openda kernel binary release.
- install openda

1.12.2 Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<bmiModelFactoryConfig xmlns="http://www.openda.org" xmlns:xsi="http://www.w3.org/2001/
↪XMLSchema-instance" xsi:schemaLocation="http://www.openda.org bmiModelFactoryConfig.xsd
↪">
    <pythonModel>
        <pythonPath>../../../wflow_bin</pythonPath>
        <moduleName>wflow.wflow_bmi</moduleName>
        <className>wflowbmi_csdms</className>
        <!-- You must give an absolute path to the py.exe of the binary distribution -->
        <pythonExecutable>d:\2015.02\GLOFFIS_SA\Modules\OpenStreams\wflow_bin\py.exe</
↪pythonExecutable>
```

(continues on next page)

(continued from previous page)

```
    </pythonModel>
    <modelTemplateDirectory>../../combined</modelTemplateDirectory>
    <modelConfigFile>wflow_wr3a.ini</modelConfigFile>
   <bmiModelForcingsConfig>
        <dataObject>
            <className>org.openda.exchange.dataobjects.NetcdfDataObject</className>
            <file>Precipitationmapstack.nc</file>
            <arg>true</arg>
            <arg>false</arg>
        </dataObject>
    </bmiModelForcingsConfig>
</bmiModelFactoryConfig>
```

CHAPTER TWO

REFERENCES

- Köhler, L., Mulligan, M., Schellekens, J., Schmid, S. and Tobón, C.: Final Technical Report DFID-FRP Project no. R7991 Hydrological impacts of converting tropical montane cloud forest to pasture, withinitial reference to northern Costa Rica. 2006.

CHAPTER

THREE

PAPERS/REPORTS USING WFLOW

Arnal, L., 2014. An intercomparison of flood forecasting models for the Meuse River basin (MSc Thesis). Vrije Universiteit, Amsterdam.

Azadeh Karami Fard, 2015. Modeling runoff of an Ethiopian catchment with WFLOW (MSc thesis). Vrije Universiteit, Amsterdam.

de Boer-Euser, T., Bouaziz, L., De Niel, J., Brauer, C., Dewals, B., Drogue, G., Fenicia, F., Grelier, B., Nossent, J., Pereira, F., Savenije, H., Thirel, G., Willems, P., 2017. Looking beyond general metrics for model comparison – lessons from an international model intercomparison study. Hydrol. Earth Syst. Sci. 21, 423–440. doi:10.5194/hess-21-423-2017

Emerton, R.E., Stephens, E.M., Pappenberger, F., Pagano, T.C., Weerts, A.H., Wood, A.W., Salamon, P., Brown, J.D., Hjerdt, N., Donnelly, C., Baugh, C.A., Cloke, H.L., 2016. Continental and global scale flood forecasting systems. WIREs Water 3, 391–418. doi:10.1002/wat2.1137

Hally, A., Caumont, O., Garrote, L., Richard, E., Weerts, A., Delogu, F., Fiori, E., Rebora, N., Parodi, A., Mihalović, A., Ivković, M., Dekić, L., van Verseveld, W., Nuissier, O., Ducrocq, V., D'Agostino, D., Galizia, A., Danovaro, E., Clematis, A., 2015. Hydrometeorological multi-model ensemble simulations of the 4 November 2011 flash flood event in Genoa, Italy, in the framework of the DRIHM project. Nat. Hazards Earth Syst. Sci. 15, 537–555. doi:10.5194/nhess-15-537-2015

Hassaballah, K., Mohamed, Y., Uhlenbrook, S., Biro, K., 2017. Analysis of streamflow response to land use land cover changes using satellite data and hydrological modelling: case study of Dinder and Rahad tributaries of the Blue Nile. Hydrol. Earth Syst. Sci. Discuss. 2017, 1–22. doi:10.5194/hess-2017-128

Jeuken, A., Bouaziz, L., Corzo, G., Alfonso, L., 2016. Analyzing Needs for Climate Change Adaptation in the Magdalena River Basin in Colombia, in: Filho, W.L., Musa, H., Cavan, G., O'Hare, P., Seixas, J. (Eds.), Climate Change Adaptation, Resilience and Hazards, Climate Change Management. Springer International Publishing, pp. 329–344.

López López, P., Wanders, N., Schellekens, J., Renzullo, L.J., Sutanudjaja, E.H., Bierkens, M.F.P., 2016. Improved large-scale hydrological modelling through the assimilation of streamflow and downscaled satellite soil moisture observations. Hydrol. Earth Syst. Sci. 20, 3059–3076. doi:10.5194/hess-20-3059-2016

Maat, W.H., 2015. Simulating discharges and forecasting floods using a conceptual rainfall-runoff model for the Bolivian Mamoré basin (MSc thesis). University of Twente, Enschede.

Research paper: HYDROLOGIC MODELING OF PRINCIPAL SUB-BASINS OF THE MAGDALENA-CAUCA LARGE BASIN USING WFLOW MODEL [WWW Document], n.d. . ResearchGate. URL https://www.researchgate.net/publication/280293861_HYDROLOGIC_MODELING_OF_PRINCIPAL_SUB-BASINS_OF_THE_MAGDALENA-CAUCA_LARGE_BASIN_USING_WFLOW_MODEL (accessed 4.4.17).

Tangdamrongsub, N., Steele-Dunne, S.C., Gunter, B.C., Ditmar, P.G., Weerts, A.H., 2015. Data assimilation of GRACE terrestrial water storage estimates into a regional hydrological model of the Rhine River basin. Hydrol. Earth Syst. Sci. 19, 2079–2100. doi:10.5194/hess-19-2079-2015

Tretjakova, D., 2015. Investigating the effect of using fully-distributed model and data assimilation on the performance of hydrological forecasting in the Karasu catchment, Turkey (MSc thesis). Wageningen University.

Wang, X., Zhang, J., Babovic, V., 2016. Improving real-time forecasting of water quality indicators with combination of process-based models and data assimilation technique. Ecological Indicators 66, 428–439. doi:10.1016/j.ecolind.2016.02.016

CHAPTER

FOUR

TODO

Todo: rewrite and simplify

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/wflow/checkouts/latest/wflow/wflow_adapt.py:docstr
of wflow_adapt, line 23.)

BIBLIOGRAPHY

[Dai2013] Dai, Y., W. Shangguan, Q. Duan, B. Liu, S. Fu, G. Niu, 2013. Development of a China Dataset of Soil Hydraulic Parameters Using Pedotransfer Functions for Land Surface Modeling. Journal of Hydrometeorology, 14:869-887.

[VanDijk2001] Dijk, A.I.J.M. van and L.A. Bruijnzeel (2001), Modelling rainfall interception by vegetation of variable density using an adapted analytical model. Part 1. Model description. Journal of Hydrology 247, 230-238.

[FAO2009] FAO/IIASA/ISRIC/ISS-CAS/JRC, 2009. Harmonized World Soil Database (version 1.1). FAO, Rome, Italy and IIASA, Laxenburg, Austria.

[Liu1998] Liu, S. (1998), Estimation of rainfall storage capacity in the canopies of cypress wetlands and slash pine uplands in North-Central Florida. Journal of Hydrology 207, 32-41.

[Liu2012] Liu, Y., R. Liu, and J. M. Chen (2012), Retrospective retrieval of long-term consistent global leaf area index (1981–2011) from combined AVHRR and MODIS data. J. Geophys. Res., 117, G04003, doi:10.1029/2012JG002084.

[Shangguan2014] Shangguan, W., Dai, Y., Duan, Q., Liu, B. and Yuan, H., 2014. A Global Soil Data Set for Earth System Modeling. Journal of Advances in Modeling Earth Systems, 6: 249-263.

INDEX