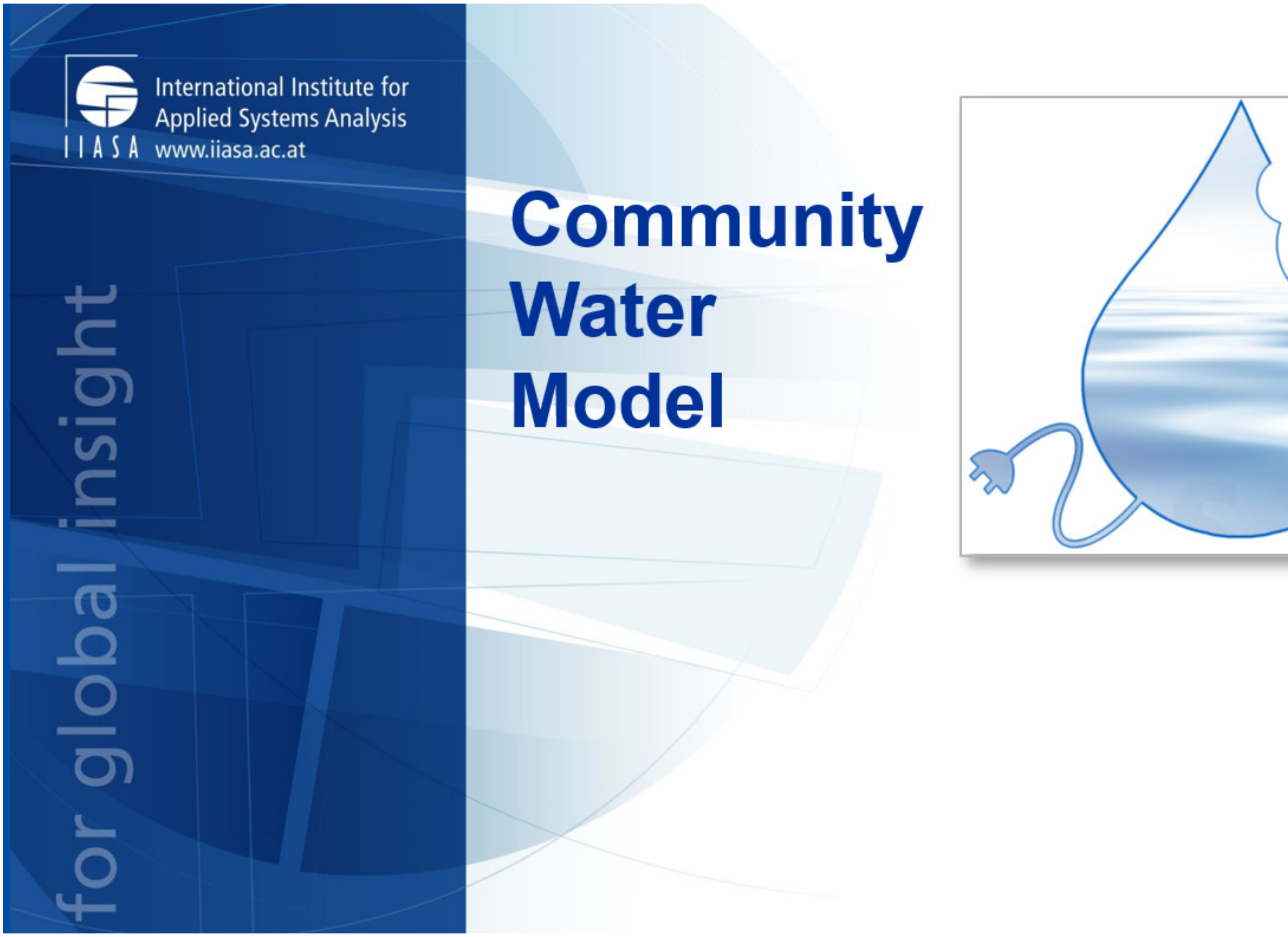
CWATM Documentation

Release 1

Peter Burek, IIASA WAT

Jan 07, 2019

Contents



Authors *Peter Burek, Yusuke Satoh, Peter Greve*

Version 1.02

Version Date Jan 07, 2019

Content:

CHAPTER 1

Introduction

1.1 Community Water Model - CWATM

With a growing population and economic development, it is expected that water demands will increase significantly in the future, especially in developing regions. At the same time, climate change is expected to alter spatial patterns of precipitation and temperature and will have regional to localized impacts on water availability. Thus, it is important to assess water demand, water supply and environmental needs over time to identify the populations and locations that will be most affected by these changes linked to water scarcity, droughts and floods. The Community Water Model will be designed for this purpose in that they include an accounting of how future water demands will evolve in response to socioeconomic change and how water availability will change in response to climate.

CWAT will represent one of the new key elements of the WAT program going forward and increasing the innovative niche of the work. We will use and develop the model to work at both global and regional (basin) level. The configuration of the model is open source and community-driven to promote our work amongst the wider water community and is flexible enough to introduce further planned developments such as water quality and hydro-economy.

Our vision for the short to medium term work of the group is to introduce water quality (i.e., salinization in deltas and eutrophication associated with mega cities) into the community model and to consider how to include a qualitative/quantitative measure of transboundary river and groundwater governance into a scenario and modelling framework.

Contact CWAT

www.iiasa.ac.at/cwatm
wfas.info@iiasa.ac.at

Download pdf

`CWATM_MANUAL.pdf`

CHAPTER 2

Model Design

Contents

2.1 Background

2.1.1 Water Futures and Solutions Initiatives (WFAS)

Water Futures and Solutions Initiatives is using a multi-model approach for global climatic, hydro-socioeconomic modeling in order to assess possible futures. We use three leading global hydrological models H08, WaterGAP and PCR-GLOBWB for estimating water demand and supply. This approach is used for a better understanding of the uncertainty and limitations of modeling. It provides a degree of confidence in the results an is in-line with the ISI-MIPS approach of multi-modeling

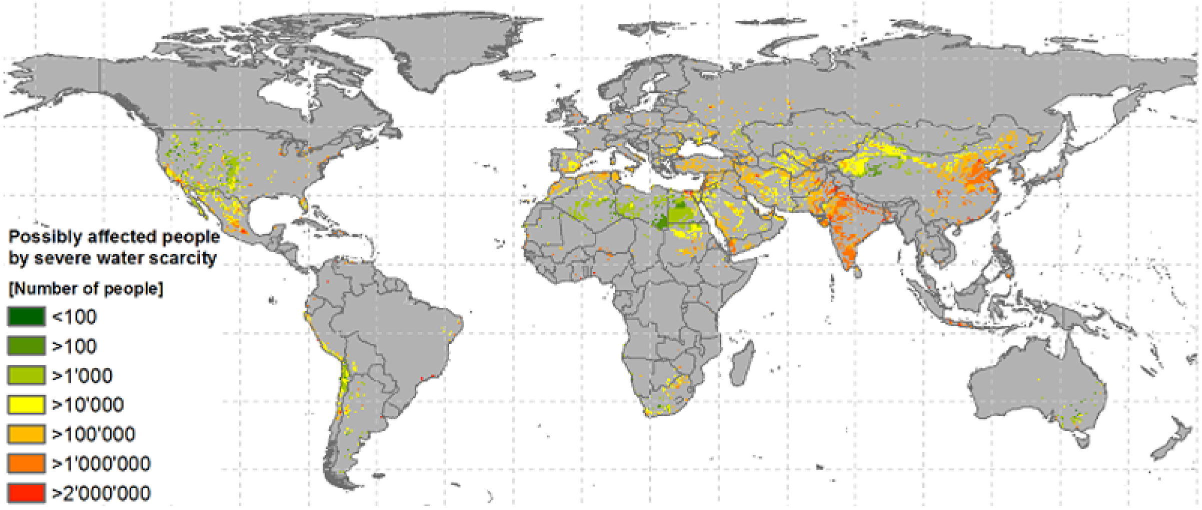


Figure 1: Potential population under severe water scarcity in 2050 - Middle of the Road Scenario - WFAS fast-track analysis

2.1.2 Nexus Integration - Water Energy Food Environment

In the framework of the Integrated Solution project
the Community Water Model (CWATM) will be coupled with the existing IIASA models MESSAGE and GLOBIOM
in order to do enhanced water assessments and an improved analysis feedback on water, energy, food and environmental aspects

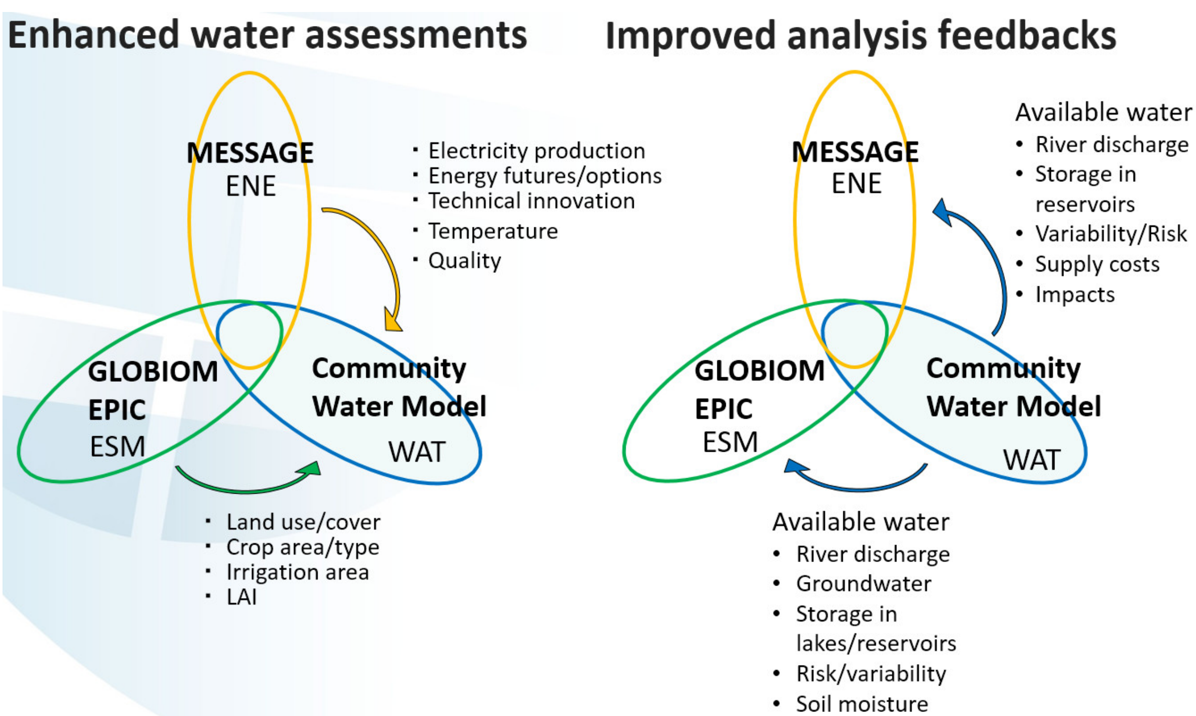


Figure 2: IIASA model interactions

2.1.3 CWAT and the IIASA global hydro-economic model

The Community Water Model will help to develop a next-generation hydro-economic modeling tool that represents the economic trade-offs among water supply technologies and demands. The tool will track water use from all sectors and will identify the least-cost solutions for meeting future water demands under policy constraints. In addition, the tool will track the energy requirements associated with the water supply system (e.g., desalination and water conveyance) to facilitate the linkage with the energy-economic tool. The tool will also incorporate environmental flow requirements to ensure sufficient water for environmental needs. The new hydro-economic model will be linked to CWATM by GAMS output and input files (gdx-files).

2.2 Features of the Model

2.2.1 Community Model

Feature	Description
Community driven	Open-source but lead by IIASA GitHub repository
Well documented	Documentation, automatic source code documentation GitHub Docu
Easy handling	Use of a setting file with all necessary information for the user *Complete settings file* and *Output Meta NetCDF information*
Multi-platform	Windows, Mac, Linux, Unix - to be used on different platforms (PC, clusters, super-computers)
Modular	Processes in subprograms, easy to adapt to the requirements of options/ solutions Modular structure

2.2.2 Water Model

Feature	Description
Flexible	different resolution, different processes for different needs, links to other models, across sectors and across scales
Adjustable	to be tailored to the needs at IIASA i.e. collaboration with other programs/models, including solutions and option as part of the model
Multi-disciplinary	including economics, environmental needs, social science perspectives
Sensitive	Sensitive to option / solution
Fast	Global to regional modeling – a mixture between conceptional and physical modeling – as complex as necessary but not more
Comparable	Part of the ISI-MIP community

2.2.3 Demo of first results

Here are some first demonstration of the model run:

Demo of the model

2.3 Model design and processes

2.3.1 Design

The Community Water Model (CWATM) will be designed for the purpose to assess water availability, water demand and environmental needs. It includes an accounting of how future water demands will evolve in response to socioeconomic change and how water availability will change in response to climate.

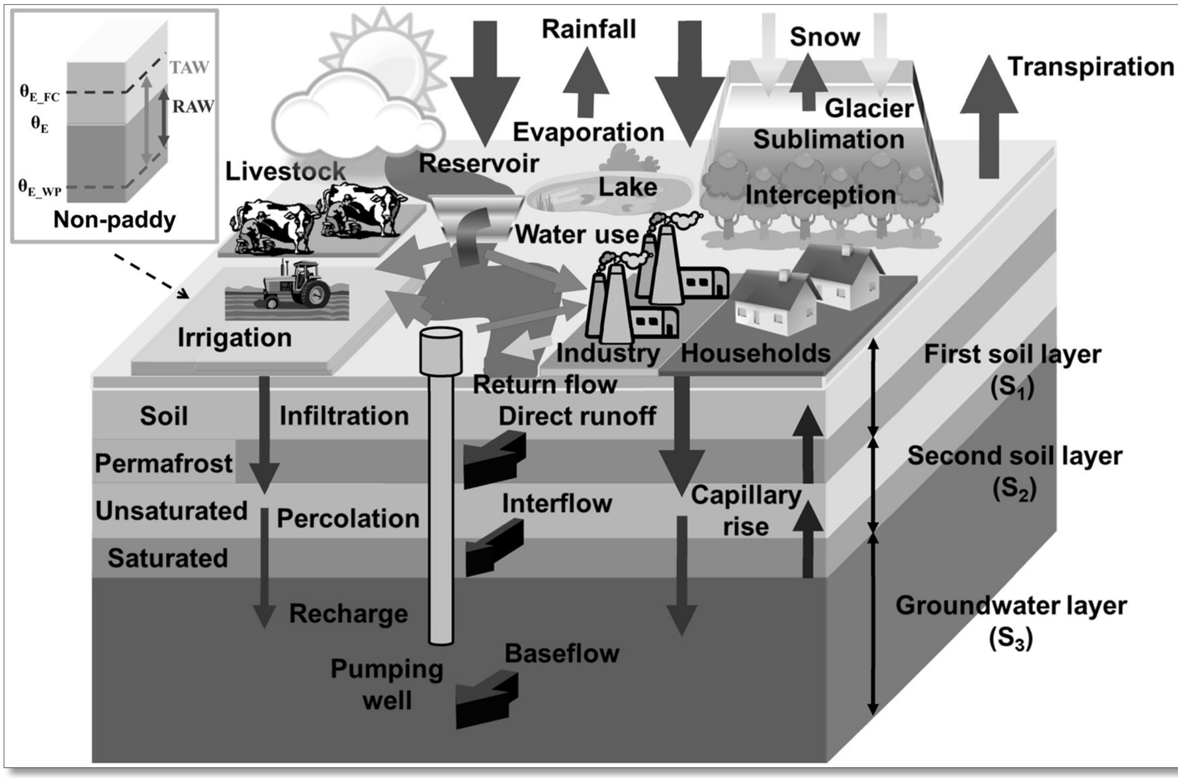


Figure 3: CWATM - Water related processes included in the model design

2.3.2 Processes

Calculation of potential Evaporation

Using Penman-Montheith equations based on FAO 56

Calculation of rain, snow, snowmelt

Using day-degree approach with up to 10 vertical layers Including snow- and glacier melt.

Land cover

using fraction of 6 different land cover types

- Forest
- Grassland

- Irrigated land
- Paddy irrigated land
- Sealed areas (urban)
- Water

Water demand

- including water demand from industry and domestic land use via precalculated monhly spatial maps
- including agricultural water use from calculation of plant water demand
- Return flows

Vegetation

Vegetation taken into account for calculating

- Albedo
- Transpiration
- Interception

Soil

Three soil layers for each land cover type including processes:

- Frost interupting soil processes
- Infiltration
- Preferential flow
- Capillary rise
- Surface runoff
- Interflow
- Percolation into groundwater

Groundwater

Groundwater storage is simulated as linear groundwater reservoir

Lakes & Reservoirs

Lakes are simulated with the weir function Reservoirs are simulated as outflow function between three storage limits (conservative, normal,flood) and three outflow functions (minimum, normal, non-damaging)

Routing

Routing is calculated using the kinematic wave approach

CHAPTER 3

Publication

Contents

3.1 Publication

1. Burek, P.; Y. Satoh; P. Greve; T. Tang; M.T> Kahil; X., He; Y. Wada et al. Development of the CWatM (Community Water Model) – A high resolution hydrological model for regional and global assessment of integrated water management options. In preparation
2. He, X., Poledna, S., Burek, P. Kahil, T, Y. Wada et al. Investigation of drought adaptation options using an integrated hydrological and agent-based model. In preparation
3. Satoh, Y., Kahil, T., Byers, E., Burek, P., Fischer, G., Tramberend, S., Greve, P., Flörke, M., Eisner, S., Hanasaki, N., Magnuszewski, P., Nava, L. F., Cosgrove, W., Langan, S. and Wada, Y. (2017), Multi-model and multi-scenario assessments of Asian water futures: The Water Futures and Solutions (WFaS) initiative. Earth's Future, 5, 823-852
4. Burek, P., Y. Satoh, G. Fischer, M.T. Kahil, A. Scherzer, S. Tramberend, L. F. Nava, Y. Wada, S. Eisner, M. Flörke, N. Hanasaki, P. Magnuszewski, B. Cosgrove, D. Wiberg and A. P. D. W. Bill Cosgrove (2016). Water Futures and Solution - Fast Track Initiative (Final Report). IIASA, Laxenburg, Austria.
5. Wada, Y., M. Flörke, N. Hanasaki, S. Eisner, G. Fischer, S. Tramberend, Y. Satoh, M. T. H. van Vliet, P. Yillia, C. Ringler, P. Burek and D. Wiberg (2016). "Modeling global water use for the 21st century: Water Futures and Solutions (WFaS) initiative and its approaches." Geosci. Model Dev. Discuss. 8(8): 6417-6521.

3.2 Presentations

Burek P, Satoh Y, Greve P, Kahil T, & Wada Y (2017). The Community Water Model (CWATM) / Development of a community driven global water model. In: European Geosciences Union (EGU) General Assembly 2017, 23–28 April 2017, Vienna, Austria - Poster

Event: 2017 AGU Fall Meeting, New Orleans, Louisiana

Presentation title: Improving Water Resources Management on Global and Region Scales – Evaluating Strategies for Water Futures with the IIASA's Community Water Model

When: Friday, 15 December 2017 11:50 - 12:05

Where: H52F: Progress in Large-Scale Modeling and Remote Sensing of the Water Cycle Toward Better Human Water

3.3 Developer

Research Scholars, Water Program, IIASA

3.3.1 Peter Burek, Yusuke Satoh, Peter Greve


BE SAFE

CHAPTER 4

Setup of the model

Contents

4.1 Setup

4.1.1 Requirements

Python version

NEW from 2019 on: Requirements are a 64 bit Python 3.7.x version

Reason for this step:

- Python 2.7 support ends in 2019

- We will be able to provide a better error handling
- We are able to provide an executable of CWATM for Windows

Warning: a 32 bit version is not able to handle the data requirements!

Warning: From 2019 on we are changing to Python 3.7. We do not provide further support for Python 2.7

Libraries

These external libraries are needed:

- Numpy
- Scipy
- netCDF4
- GDAL

Windows

The four libraries can be installed with pip or downloaded at Unofficial Windows Binaries for Python Extension Packages

Windows executeable Python version

A CWATM executable cwatm.exe can be used instead of the Python version

- ADVANTAGE: You can run it without installing or knowledge of Python
- DISADVANTAGE 1: You cannot see the source code or change it
- DISADVANTAGE 2: We do not update this version as often as the Python version
- It is done with cx_freeze library
- It includes all Python libraries

Note:

A cwatmexe.zip (around 300 MB with all Python libraries) is stored on:
Source code on Github repository of CWATM
Executable cwatmexe.zip on Github repository of CWATM

Note: We recommend using the Python 3.7.x version, but if you not experienced in Python or have problems installing CWATM, please use the executable version.

PCRaster

CWATM is not using anything from PCRaster
But the general idea of PCraster to split the hydrological modules in a initial | part and a dynamic part is still used

Anyway PCRaster is a great tool
PCRASTER from Faculty of Geosciences, Utrecht University, The Netherlands
Webpage of PCRaster

Reference:
Karssenberg, D., Schmitz, O., Salamon, P., de Jong, K., and Bierkens, M. F. P.: A software framework for construction of process-based stochastic spatio-temporal models and data assimilation, Environmental Modelling & Software 25(4), 489-502, 2010. doi: 10.1016/j.envsoft.2009.10.004

4.1.2 C++ libraries

For the computational time demanding parts e.g. routing, CWATM comes with a C++ library

Compiled versions

Windows and CYGWIN_NT-6.1
a compiled version is provided and CWATM is detecting automatically which system is running and which compiled version is needed

Linux
For Cygwin linux a compiled version *t5cyg.so* is provided in *../source/hydrological_modules/routing_reservoirs/* for version CYGWIN_NT-6.1.
If you use another cygwin version please compile it by yourself and name it *t5_linux.so*

For Linux Ubuntu a compiled version is provided as *t5_linux.so*. The file is in *../source/hydrological_modules/routing_reservoirs/*

Note: If you use another Linux version or the compiled version is not working or you have a compiler which produce faster executables please compile a version on your own.

Compiling a version

C++ sourcecode is in *../source/hydrological_modules/routing_reservoirs/t5.cpp*

Note: A compiled version is already provided for Windows and Linux.

Windows

A compiled version is provided, but maybe you have a faster compiler than the "Minimalist GNU for Windows" or "Microsoft Visual Studio 14.0" we used.

To compile with g++:

```
..\g++ -c -fPIC -Ofast t5.cpp -o t5.o
..\g++ -shared -Ofast -Wl,-soname,t5.so -o t5.so  t5.o
```

To compile with Microsoft Visual Studio 14.0:

```
call "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin\amd64/vcvars64.bat"
cl /LD /O2 t5.cpp
```

Note:

We used Visual Studio, because it seems to be computational faster

the libray used with Windows is named *t5.dll*, if you generate a libray *t5.so* the filename in **../source/management_modules/globals.py** has to be changed!

Linux

To compile with g++:

```
..\g++ -c -fPIC -Ofast t5.cpp -o t5_linux.o
..\g++ -shared -Ofast -Wl,-soname,t5_linux.so -o t5_linux.so  t5_linux.o

    or

    ..\g++ -c -Ofast t5.cpp -o t5_linux.o
..\g++ -shared -Ofast -Wl,-soname,t5_linux.so -o t5_linux.so  t5_linux.o
```

Warning: Please rename your compiled version to t5_linux.so! At the moment the file t5_linux.so is compiled with Ubuntu Linux

4.1.3 Test the model

Windows and Linux

python <modelpath>/cwatm.py

The output should be:

```
Running under platform:  Windows  **(or Linux etc)**
CWatM - Community Water Model
Authors: ...
Version: ...
Date: ...
```

Warning: If python is not set in the environment path, the full path of python has to be used

Warning: Please use the right version of CWATM with the right version of Python (either 2.7 or 3.7)

4.2 Running the model

4.2.1 Start the model

Warning: The model needs a settings file as an argument. See: *Settings file*

Windows

python <modelpath>/cwatm.py settingsfile flags

example:

```
python cwatm.py settings1.ini
or with more information and an overview of computational runtime
python cwatm.py settings1.ini -l -t
```

Warning: If python is not set in the environment path, the full path of python has to be used

Linux

<modelpath>/cwatm.py settingsfile flags

example:

```
cwatm.py settings1.ini -l -t
```

Flags

Flags can be used to change the runtime output on the screen

example:

```
-q --quiet       output progression given as .
-v --veryquiet   no output progression is given
-l --loud        output progression given as time step, date and discharge
-c --check       input maps and stack maps are checked, output for each input map BUT␣
↪no model run
-h --noheader    .tss file have no header and start immediately with the time series
-t --printtime   the computation time for hydrological modules are printed
    -w --warranty    copyright and warranty information
```

Settings file

The setup of the setings file is shown in the next chapter.

NetCDF meta data

The format for spatial data for output data is netCDF. In the meta data file information can be added e.g. a description of the parameter

Note: It is not necessary to change this file! This is an option to put additional information into output maps

4.3 Settings file

The settings file is controlling the CWATM run

```
######## ##          ##  ####  ######  ##    ##
##       ##          ## ##  ##   ##   ####  ####
##        ##        ##  ##  ##   ##   ## #### ##
##        ##   ##   ## ########  ##  ##   ##   ##
##         ## #### ##  ##    ##  ##  ##        ##
##         ####  #### ##      ## ## ##          ##
##########  ##    ##  ##      ## ## ##          ##

# Community Water Model Version 0.99
```

4.3.1 Components of the settings file

General flags

General flags are set in the first paragraph For example: If Temperature data are in unit ° Celsius ot Kelvin

```
[OPTIONS]
#-------------------------------------------------------
# OPTION - to switch on/off
#-------------------------------------------------------

# Data otions
# if temperature is stored in Kelvin instead Celsius
TemperatureInKelvin = True
# if lat/lon the area has to be user defined = precalculated
gridSizeUserDefined = True

#-----------------------------------------------
# Evaporation: calculate pot. evaporation (True) or use precalculated pot.evaporation␣
↪map stacks (False)
calc_evaporation = False

#-----------------------------------------------
# Irrigation and water demand

# if irrigation is included, otherwise paddy and non paddy is put into 'grassland'
includeIrrigation = True
# if water demand from irrigation, industry and domestic is included
includeWaterDemand = False
# Water allocation
# if water demand and availability is calculated for region to compare demand vs.␣
↪avail
```

(continues on next page)

(continued from previous page)

```
usingAllocSegments = False
# limit abstraction to available groundwater (True) include fossil groundwater (False)
limitAbstraction = False

# Environmental Flow
calc_environflow = False
use_environflow = False

#-----------------------------------------------
# Soil
# use preferential flow, that bypasses the soil matrix and drains directly to the groundwater (not for irrPaddy)
preferentialFlow = False
# Capillar rise
CapillarRise = True

#-----------------------------------------------
# Routing

# if runoff concentration to the edge of a cell is included
includeRunoffConcentration = True
# Waterbodies like lakes and reservoirs
includeWaterBodies = True
# kinematic wave routing, if False no routing is calculated
includeRouting = True

#-----------------------------------------------
# Inflow from outside of the modelled area
inflow = False

# --- Reporting & Debugging ------------------- ---------------------------------------------
# Reporting options
writeNetcdfStack = True
reportMap = True
reportTss = True
# Checking water balance (for debugging)
calcWaterBalance = False
sumWaterBalance = False
# use additional PCRaster GIS commands
PCRaster = False







#-------------------------------------------------------
# DEFINITIONS OF PARAMETERS
```

NetCDF meta data

The format for spatial data for input and output data is netCDF. For output data the basic information are given in the settingsfile

```
[NETCDF_ATTRIBUTES]
institution = IIASA
title = Global Water Model - WATCH WDFEI
metaNetcdfFile = $(FILE_PATHS:PathRoot)/source/metaNetcdf.xml
```

For each output file the specific information about units, variable name, displayed variable name is given in the metaNetcdf.xml. See: *Output Meta NetCDF information*

Path of data, output

Note: Further on the pathes can be used as placeholders

```
#-------------------------------------------------------
[FILE_PATHS]
#-------------------------------------------------------
PathRoot = E:/CWATM_rhine

PathOut = $(PathRoot)/output
PathMaps = $(PathRoot)/cwatm_input
PathMeteo = $(PathRoot)/climate
```

Defining the modeling area

In general the input data are stored and used at global scale. The modeling area can be defined by:

- a mask map
- coordinates

Note: The mask map can be a .tif, PCraster or a netCDF format | The coordinates have the format: Number of Cols, Number of rows, cellsize, upper left corner X, upper left corner Y

```
# AREA AND OUTLETS
#-------------------------------------------------------
[MASK_OUTLET]

# Area mask
# A pcraster map, tif or netcdf map e.g.  $(FILE_PATHS:PathRoot)/data/areamaps/area_indus.map
# or a retancle: Number of Cols, Number of rows, cellsize, upper left corner X, upper left corner Y
MaskMap = $(FILE_PATHS:PathRoot)/source/rhine30min.tif
#MaskMap = 14 12 0.5 5.0 52.0


#-------------------------------------------------------
# Station data
# either a map e.g. $(FILE_PATHS:PathRoot)/data/areamaps/area3.map
# or a location coordinates (X,Y) e.g. 5.75 52.25 9.25 49.75 )
# Lobith/Rhine
Gauges = 6.25 51.75
```

(continues on next page)

(continued from previous page)

```

# if .tif file for gauges, this is a flag if the file is global or local
# e.g. Gauges = $(FILE_PATHS:PathRoot)/data/areamaps/gaugesRhine.tif
GaugesLocal = True
```

Defining the time

The start and end time have to be defined. Spin-up time is the time for warming up (results will be stored after the spin-up time)

Note: The time can be given as date: dd/mm/yyyy or as relative date: number (but then CalendarDayStart has to be defined)

Note: Spin-up time can be given as date or number

```
#-------------------------------------------------------
[TIME-RELATED_CONSTANTS]
#-------------------------------------------------------

# StepStart has to be a date e.g. 01/06/1990
# SpinUp or StepEnd either date or numbers
# SpinUp: from this date output is generated (up to this day: warm up)

StepStart = 1/1/1990
SpinUp =  1/01/1995
StepEnd =  31/12/2010
```

Initial conditions

Initial conditions can be stored and be loaded in order to initialise a warm start of the model

Note: Initial conditions are store as one netCDF file with all necessary variables

```
#-------------------------------------------------------
[INITITIAL CONDITIONS]
#-------------------------------------------------------

# for a warm start initial variables a loaded
# e.g for a start on 01/01/2010 load variable from 31/12/2009
load_initial = False
initLoad = $(FILE_PATHS:PathRoot)/init/Rhine_19891231.nc

# saving variables from this run, to initiate a warm start next run
# StepInit = saving date, can be more than one: 10/01/1973 20/01/1973
save_initial = False
initSave = $(FILE_PATHS:PathRoot)/init/Rhine
StepInit = 31/12/1989 31/12/2010
```

StepInit indicate the date(s) when initial conditions are saved:

```
StepInit = 31/12/1989
StepInit = 31/12/1989 31/12/2010
StepInit = 31/12/1989 5y
here: second value in StepInit is indicating a repetition of year(y), month(m) or␣
↪day(d),
e.g. 2y for every 2 years or 6m for every 6 month
```

Output

Output can be spatial/time as netCDF4 map stacks

and/or time series at specified points

Note: For additional information see *Model Output*

Output can be as maps and time series:

- per day [Daily]
- total month [MonthTot], average month [MonthAvg], end of month [MonthEnd]
- total year [AnnualTot], average year [AnnualAvg], end of year [AnnualEnd]
- total sum [TotalTot], total average [TotalAvg]

For each of the following sections output can be defined for different variables:

- Meteo
- Snow
- Soil for different land cover (forest, grassland, irrigated land, paddy irrigated)
- Water demand
- Groundwater
- River routing
- Lakes and reservoirs

Or output can be defined in the section *[output]*

An output directory can be defined and for each sort of output the variable(s) can be set:

*OUT_* defines that this variable(s) are output
*MAP_* or *TSS_* defines if it is a spatial map or a time series of point(s)
Daily or *MonthAvg* or .. is specifying the time
The variable is given after the equal sign e.g. * = discharge*
If more than one variable should be used for output, split with **,**
E.g. OUT_MAP_Daily = discharge -> daily spatial map of discharge

As example output for precipitation, temperature and discharge is shown here:

```
# OUTPUT maps and timeseries
OUT_Dir = $(FILE_PATHS:PathOut)
OUT_MAP_Daily =
OUT_MAP_MonthEnd =
OUT_MAP_MonthTot = Precipitation, Tavg
OUT_MAP_MonthAvg =

OUT_TSS_MonthTot = Precipitation, Tavg
OUT_TSS_Daily = discharge
OUT_TSS_MonthEnd = discharge
OUT_TSS_AnnualEnd = discharge
```

Note: For each variable the meta data information can be defined in *Output Meta NetCDF information*

Reading information

Information will be read in from values in the settings file Here the value definitions for [SNOW] is shown:

```
#-------------------------------------------------------
[SNOW]
#-------------------------------------------------------

# Number of vertical Snow layers
NumberSnowLayers = 7
# up to which layer the ice melt is calculated with the middle temperature
GlacierTransportZone = 3

# Temperature lapse rate with altitude [deg C / m]
TemperatureLapseRate = 0.0065
# Multiplier applied to precipitation that falls as snow
SnowFactor = 1.0
# Range [m C-1 d-1] of the seasonal variation, SnowMeltCoef is the average value
SnowSeasonAdj = 0.001
# Average temperature at which snow melts
TempMelt =1.0
# Average temperature below which precipitation is snow
TempSnow = 1.0
# Snow melt coefficient: default: 4.0
# SRM: 0.0045 m/C/day ( = 4.50 mm/C/day), Kwadijk: 18 mm/C/month (= 0.59 mm/C/day)
# See also Martinec et al., 1998.

# use in CALIBRATION -> copied to CALIBRATION
#SnowMeltCoef = 0.004
IceMeltCoef  = 0.007

#---------------------------------------------------------------
# INITIAL CONDITIONS - Initial snow depth in snow zone 1-7 [mm]  - SnowCoverIni

[FROST]
# Snow water equivalent, (based on snow density of 450 kg/m3) (e.g. Tarboton and Luce, 1996)
SnowWaterEquivalent = 0.45
# Daily decay coefficient, (Handbook of Hydrology, p. 7.28)
Afrost = 0.97
```

(continues on next page)

(continued from previous page)

```
# Snow depth reduction coefficient, [cm-1], (HH, p. 7.28)
Kfrost = 0.57
# Degree Days Frost Threshold (stops infiltration, percolation and capillary rise)
# Molnau and Bissel found a value 56-85 for NW USA.
FrostIndexThreshold = 56
```

Note: TemperatureLapseRate = 0.0065 | for the variable TemperatureLapseRate the value of 0.0065 is set

Variables can also be defined by spatial maps or map stacks

```
tanslope = $(PathTopo)\tanslope.map
forest_coverFractionNC    = $(PathForest)\coverFractionInputForest366days.nc
```

Note: suffix can be .map, but if there is no PCraster map it will look automatically for netCDF .nc

Warning: in most cases values can be replaced by map

4.3.2 Sections of information

- Snow
- Frost
- General information on land cover types
- Soil
- **Information for each of the six land cover types**
  - Forest
  - Grassland
  - Paddy irrigated area
  - Irrigated area
  - Sealed area
  - Water covered area
- Interflow
- Groundwater
- Water demand
- Runoff concentration
- Routing

- Lakes and reservoirs
- Inflow

4.3.3 Complete settings file

Example of a settings file:

```
# ----------------------------------------------------

######## ##          ##  ####  ######  ##    ##
##       ##          ## ##  ##   ##   ####  ####
##        ##        ##  ##  ##   ##   ## #### ##
##        ##   ##   ## #########  ##  ##   ##   ##
##         ## #### ##  ##    ##  ##  ##        ##
##         ####  #### ##      ## ## ##          ##
##########  ##    ##  ##      ## ## ##          ##

# Community Water Model Version 0.99
# SETTINGS FILE
# ----------------------------------------------------


[OPTIONS]
#-----------------------------------------------------------
# OPTION - to switch on/off
#-----------------------------------------------------------

# Data otions
# if temperature is stored in Kelvin instead Celsius
TemperatureInKelvin = True
# if lat/lon the area has to be user defined = precalculated
gridSizeUserDefined = True

#--------------------------------------------------
# Evaporation: calculate pot. evaporation (True) or use precalculated pot.evaporation␣
↪map stacks (False)
calc_evaporation = False

#--------------------------------------------------
# Irrigation and water demand

# if irrigation is included, otherwise paddy and non paddy is put into 'grassland'
includeIrrigation = True
# if water demand from irrigation, industry and domestic is included
includeWaterDemand = False
# Water allocation
# if water demand and availability is calculated for region to compare demand vs.␣
↪avail
usingAllocSegments = False
# limit abstraction to available groundwater (True) include fossil groundwater (False)
limitAbstraction = False

# Environmental Flow
calc_environflow = False
use_environflow = False
```

(continues on next page)

(continued from previous page)

```
#-----------------------------------------------
# Soil
# use preferential flow, that bypasses the soil matrix and drains directly to the␣
↪groundwater (not for irrPaddy)
preferentialFlow = False
# Capillar rise
CapillarRise = True

#-----------------------------------------------
# Routing

# if runoff concentration to the edge of a cell is included
includeRunoffConcentration = True
# Waterbodies like lakes and reservoirs
includeWaterBodies = True
# kinematic wave routing, if False no routing is calculated
includeRouting = True

#-----------------------------------------------
# Inflow from outside of the modelled area
inflow = False

# --- Reporting & Debugging ------------------- --------------------------------------
↪--------
# Reporting options
writeNetcdfStack = True
reportMap = True
reportTss = True
# Checking water balance (for debugging)
calcWaterBalance = False
sumWaterBalance = False
# use additional PCRaster GIS commands
PCRaster = False







#-------------------------------------------------------
# DEFINITIONS OF PARAMETERS
#-------------------------------------------------------

#-------------------------------------------------------
[FILE_PATHS]
#-------------------------------------------------------
PathRoot = E:/CWATM_rhine

PathOut = $(PathRoot)/output
PathMaps = $(PathRoot)/cwatm_input
PathMeteo = $(PathRoot)/climate




#-------------------------------------------------------
[NETCDF_ATTRIBUTES]
```

(continues on next page)

(continued from previous page)

```
institution = IIASA
title = Global Water Model - WATCH WDFEI
metaNetcdfFile = $(FILE_PATHS:PathRoot)/source/metaNetcdf.xml

#-------------------------------------------------------
# AREA AND OUTLETS
#-------------------------------------------------------
[MASK_OUTLET]

# Area mask
# A pcraster map, tif or netcdf map e.g.  $(FILE_PATHS:PathRoot)/data/areamaps/area_
↪indus.map
# or a retancle: Number of Cols, Number of rows, cellsize, upper left corner X, upper␣
↪left corner Y
MaskMap = $(FILE_PATHS:PathRoot)/source/rhine30min.tif
#MaskMap = 14 12 0.5 5.0 52.0


#-------------------------------------------------------
# Station data
# either a map e.g. $(FILE_PATHS:PathRoot)/data/areamaps/area3.map
# or a location coordinates (X,Y) e.g. 5.75 52.25 9.25 49.75 )
# Lobith/Rhine
Gauges = 6.25 51.75

# if .tif file for gauges, this is a flag if the file is global or local
# e.g. Gauges = $(FILE_PATHS:PathRoot)/data/areamaps/gaugesRhine.tif
GaugesLocal = True

#-------------------------------------------------------
[TIME-RELATED_CONSTANTS]
#-------------------------------------------------------

# StepStart has to be a date e.g. 01/06/1990
# SpinUp or StepEnd either date or numbers
# SpinUp: from this date output is generated (up to this day: warm up)

StepStart = 1/1/1990
SpinUp =  1/01/1995
StepEnd =  31/12/2010




#-------------------------------------------------------
[INITITIAL CONDITIONS]
#-------------------------------------------------------

# for a warm start initial variables a loaded
# e.g for a start on 01/01/2010 load variable from 31/12/2009
load_initial = False
initLoad = $(FILE_PATHS:PathRoot)/init/Rhine_19891231.nc

# saving variables from this run, to initiate a warm start next run
# StepInit = saving date, can be more than one: 10/01/1973 20/01/1973
save_initial = False
initSave = $(FILE_PATHS:PathRoot)/init/Rhine
```

(continues on next page)

(continued from previous page)

```
StepInit = 31/12/1989 31/12/2010

#-------------------------------------------------------
# CALIBARTION PARAMETERS
#-------------------------------------------------------
[CALIBRATION]

# These are parameter which are used for calibration
# could be any parameter, but for an easier overview, tehey are collected here
# in the calibration template a placeholder (e.g. %arnoBeta) instead of value

# Snow
SnowMeltCoef = 0.0027
# Cropf factor correction
crop_correct =  1.11
#Soil
soildepth_factor = 1.28
#Soil preferentialFlowConstant = 4.0, arnoBeta_add = 0.1
preferentialFlowConstant = 4.5
arnoBeta_add = 0.19
# interflow part of recharge factor = 1.0
factor_interflow = 2.8
# groundwater recessionCoeff_factor = 1.0
recessionCoeff_factor = 5.278
# runoff concentration factor runoffConc_factor = 1.0
runoffConc_factor = 0.1
#Routing manningsN Factor to Manning's roughness = 1.0 [0.1-10.]
manningsN = 1.86
# reservoir  normal storage limit (fraction of total storage, [-]) [0.15 - 0.85]␣
↪default 0.5
normalStorageLimit = 0.44
# lake parameter - factor to alpha: parameter of of channel width and weir␣
↪coefficient  [0.33 - 3.] dafault 1.
lakeAFactor = 0.33
# lake parameter - factor for wind evaporation
lakeEvaFactor = 1.52
#-------------------------------------------------------
# TOPOGRAPHY MAPS
#-------------------------------------------------------
[TOPOP]
# local drain direction map (1-9)
Ldd = $(FILE_PATHS:PathMaps)/routing/ldd.map

# Elevation standard deviation [m], i.e. altitude difference elevation within pixel.
# Used for sub-pixel modelling of snow accumulation and melt
ElevationStD = $(FILE_PATHS:PathMaps)/landsurface/topo/elvstd.map

# Area of pixel [m2] (for lat/lon every cell has a different area)
CellArea = $(FILE_PATHS:PathMaps)/routing/cellarea.map

#-------------------------------------------------------
# INPUT METEOROLOGICAL TIMESERIES AS MAPS
#-------------------------------------------------------
[METEO]
# precipitation [kg m-2 s-1]
#PrecipitationMaps = $(FILE_PATHS:PathMeteo)/pr*
PrecipitationMaps = $(FILE_PATHS:PathMeteo)/30min/pr_rhine*
```

(continues on next page)

(continued from previous page)

```
# average daily temperature [K]
#TavgMaps = $(FILE_PATHS:PathMeteo)/tavg*
TavgMaps = $(FILE_PATHS:PathMeteo)/30min/tavg_rhine*

# ----------------------------------------
# This is used if calc_evaporation = False

# daily reference evaporation (free water)
E0Maps = $(FILE_PATHS:PathMeteo)/30min/EWRef_rhine.nc
#E0Maps = $(FILE_PATHS:PathMeteo)/EWRef_daily*
# daily reference evapotranspiration (crop)
ETMaps = $(FILE_PATHS:PathMeteo)/30min/ETRef_rhine.nc
#ETMaps = $(FILE_PATHS:PathMeteo)/ETRef_daily*

# --------------------------------
# from kg m-2s-1 to m : 86.4
precipitation_coversion = 86.4

# from MM to m : 0.001
#precipitation_coversion = 0.001

evaporation_coversion = 1.00

# OUTPUT maps and timeseries
#OUT_Dir = $(FILE_PATHS:PathOut)
#OUT_MAP_Daily = Precipitation, prec1

#-------------------------------------------------------
# CALCULATE EVAPORATION - PENMAN - MONTEITH
#-------------------------------------------------------
[EVAPORATION]

# This is used if calc_evaporation = True
# use albedo maps
albedo = True
albedoMaps = $(FILE_PATHS:PathMaps)/landsurface/albedo/albedo.nc

# if not albedo maps use fixed albedo
# Albedo of bare soil surface (Supit et. al.)
AlbedoSoil = 0.15
# Albedo of water surface (Supit et. al.)
AlbedoWater = 0.05
# Albedo of vegetation canopy (FAO,1998)
AlbedoCanopy = 0.23

# use specific humidity (TRUE) QAir,  or relative humidity (FALSE) - rhs
useHuss = False

# map stacks Temperature [K}]
TminMaps = $(FILE_PATHS:PathMeteo)/tmin*
TmaxMaps = $(FILE_PATHS:PathMeteo)/tmax*
# Instantaneous surface pressure[Pa]
PSurfMaps = $(FILE_PATHS:PathMeteo)/ps*
# 2 m istantaneous specific humidity[kg /kg] (QAir) or relative humidity [%] (rhs)
RhsMaps = $(FILE_PATHS:PathMeteo)/hurs*
# wind speed maps at 10m [m/s]
WindMaps = $(FILE_PATHS:PathMeteo)/wind*
```

(continues on next page)

(continued from previous page)

```
# radiation surface downwelling shortwave maps  [W/m2]
RSDSMaps = $(FILE_PATHS:PathMeteo)/rsds*
# radiation surface downwelling longwave maps [W/m2] [W/m2]
RSDLMaps = $(FILE_PATHS:PathMeteo)/rlds*

# OUTPUT maps and timeseries
#OUT_Dir = $(FILE_PATHS:PathOut)
#OUT_MAP_Daily = EWRef, ETRef, temp, prec

#-------------------------------------------------------
[SNOW]
#-------------------------------------------------------

# Number of vertical Snow layers
NumberSnowLayers = 7
# up to which layer the ice melt is calculated with the middle temperature
GlacierTransportZone = 3

# Temperature lapse rate with altitude [deg C / m]
TemperatureLapseRate = 0.0065
# Multiplier applied to precipitation that falls as snow
SnowFactor = 1.0
# Range [m C-1 d-1] of the seasonal variation, SnowMeltCoef is the average value
SnowSeasonAdj = 0.001
# Average temperature at which snow melts
TempMelt =1.0
# Average temperature below which precipitation is snow
TempSnow = 1.0
# Snow melt coefficient: default: 4.0
# SRM: 0.0045 m/C/day ( = 4.50 mm/C/day), Kwadijk: 18 mm/C/month (= 0.59 mm/C/day)
# See also Martinec et al., 1998.

# use in CALIBRATION -> copied to CALIBRATION
#SnowMeltCoef = 0.004
IceMeltCoef  = 0.007

#--------------------------------------------------------------
# INITIAL CONDITIONS - Initial snow depth in snow zone 1-7 [mm]  - SnowCoverIni

[FROST]
# Snow water equivalent, (based on snow density of 450 kg/m3) (e.g. Tarboton and Luce,
↪ 1996)
SnowWaterEquivalent = 0.45
# Daily decay coefficient, (Handbook of Hydrology, p. 7.28)
Afrost = 0.97
# Snow depth reduction coefficient, [cm-1], (HH, p. 7.28)
Kfrost = 0.57
# Degree Days Frost Threshold (stops infiltration, percolation and capillary rise)
# Molnau and Bissel found a value 56-85 for NW USA.
FrostIndexThreshold = 56

#--------------------------------------------------------------
# INITIAL CONDITIONS: FrostIndexIni

[VEGETATION]
cropgroupnumber = $(FILE_PATHS:PathMaps)/others/cropgrp.nc
# soil water depletion fraction, Van Diepen et al., 1988: WOFOST 6.0, p.86, Doorenbos␣
↪et. al 1978
```

(continues on next page)

(continued from previous page)

```
#-------------------------------------------------------
[SOIL]
#-------------------------------------------------------

PathTopo = $(FILE_PATHS:PathMaps)/landsurface/topo
PathSoil = $(FILE_PATHS:PathMaps)/landsurface/soil
PathSoil1 =  $(FILE_PATHS:PathMaps)/others

# Topography - tangent slope, slope length
tanslope = $(PathTopo)/tanslope.map
slopeLength = $(PathTopo)/slopeLength.map

# maps of relative elevation above flood plains
relativeElevation = $(PathTopo)/dzRel_hydro1k.nc

# Soil hydraulic properties

# soil (Hypres pedotransfer function - http://esdac.jrc.ec.europa.eu/ESDB_Archive/
↪ESDBv2/popup/hy_param.htm)
KSat1 = $(PathSoil1)/ksat1.map
KSat2 = $(PathSoil1)/ksat2.map
KSat3 = $(PathSoil1)/ksat3.map
# Alpha: an Genuchten’s shape parameter
alpha1 = $(PathSoil1)/alpha1.map
alpha2 = $(PathSoil1)/alpha2.map
alpha3 = $(PathSoil1)/alpha3.map
#Lambda: an Genuchten’s shape parameter = n-1-> n = lamda+1, m = 1 - (1/n)
lambda1 = $(PathSoil1)/lambda1.map
lambda2 = $(PathSoil1)/lambda2.map
lambda3 = $(PathSoil1)/lambda3.map
# thetas  is the volumetric water content θ saturated
thetas1 = $(PathSoil1)/thetas1.map
thetas2 = $(PathSoil1)/thetas2.map
thetas3 = $(PathSoil1)/thetas3.map
# thetar is the volumetric water content θ residual
thetar1 = $(PathSoil1)/thetar1.map
thetar2 = $(PathSoil1)/thetar2.map
thetar3 = $(PathSoil1)/thetar3.map

percolationImp = $(PathSoil)/percolationImp.map

maxGWCapRise    = 5.0

minCropKC        = 0.2
minTopWaterLayer = 0.0

# Soil depth
StorDepth1 = $(PathSoil)/storageDepth1.map
StorDepth2 = $(PathSoil)/storageDepth2.map

# preferential flow (between 1.0 and 8.0)
# used in CALIBRATION -> copied to CALIBRATION
#preferentialFlowConstant = 4.0

#-------------------------------------------------------
[LANDCOVER]
```

(continues on next page)

(continued from previous page)

```
PathLandcover = $(FILE_PATHS:PathMaps)/landsurface

coverTypes = forest, grassland, irrPaddy, irrNonPaddy, sealed, water
coverTypesShort = f, g, i, n, s, w
fractionLandcover = $(PathLandcover)/fractionLandcover.nc

# Landcover can vary from year to year
dynamicLandcover = True
# if landcover cannot vary, which year should be taken as fixed year
fixLandcoverYear = 1961

#-------------------------------------------------------

[__forest]
PathForest = $(FILE_PATHS:PathMaps)/landcover/forest
PathSoil1 =  $(FILE_PATHS:PathMaps)/others

# Parameters for the Arno's scheme
# arnoBeta is defined by orographic,+ land cover add + calibration add, the soil␣
↪water capacity distribution is based on this
# range [0.01 - 1.2]
forest_arnoBeta = 0.2

#forest_soil
forest_KSat1 = $(PathSoil1)/forest_ksat1.map
forest_KSat2 = $(PathSoil1)/forest_ksat2.map
forest_KSat3 = $(PathSoil1)/ksat3.map
forest_alpha1 = $(PathSoil1)/forest_alpha1.map
forest_alpha2 = $(PathSoil1)/forest_alpha2.map
forest_alpha3 = $(PathSoil1)/alpha3.map
forest_lambda1 = $(PathSoil1)/forest_lambda1.map
forest_lambda2 = $(PathSoil1)/forest_lambda2.map
forest_lambda3 = $(PathSoil1)/lambda3.map
forest_thetas1 = $(PathSoil1)/forest_thetas1.map
forest_thetas2 = $(PathSoil1)/forest_thetas2.map
forest_thetas3 = $(PathSoil1)/thetas3.map
forest_thetar1 = $(PathSoil1)/forest_thetar1.map
forest_thetar2 = $(PathSoil1)/forest_thetar2.map
forest_thetar3 = $(PathSoil1)/thetar3.map

# other paramater values
forest_minInterceptCap  = 0.001
forest_cropDeplFactor   = 0.0

forest_fracVegCover  = $(PathForest)/fracVegCover.map
forest_rootFraction1 = $(PathForest)/rootFraction1.map
forest_rootFraction2 = $(PathForest)/rootFraction2.map
#forest_maxRootDepth  = 2.0
forest_maxRootDepth  = $(PathForest)/maxRootDepth.map
forest_minSoilDepthFrac = $(PathForest)/minSoilDepthFrac.map


forest_cropCoefficientNC = $(PathForest)/CropCoefficientForest_10days.nc
forest_interceptCapNC    = $(PathForest)/interceptCapForest10days.nc

# initial conditions: forest_interceptStor, forest_w1, forest_w2, forest_w3,
```

(continues on next page)

(continued from previous page)

```
[__grassland]
PathGrassland = $(FILE_PATHS:PathMaps)/landcover/grassland

# Parameters for the Arno's scheme:
grassland_arnoBeta = 0.0
# arnoBeta is defined by orographic,+ land cover add + calibration add, the soil␣
↪water capacity distribution is based on this
# range [0.01 - 1.2]

# other paramater values

grassland_minInterceptCap  = 0.001
grassland_cropDeplFactor   = 0.0

grassland_fracVegCover  = $(PathGrassland)/fracVegCover.map
grassland_rootFraction1 = $(PathGrassland)/rootFraction1.map
grassland_rootFraction2 = $(PathGrassland)/rootFraction2.map
grassland_maxRootDepth  = $(PathGrassland)/maxRootDepth.map
grassland_minSoilDepthFrac = $(PathGrassland)/minSoilDepthFrac.map


grassland_cropCoefficientNC = $(PathGrassland)/CropCoefficientGrassland_10days.nc
grassland_interceptCapNC    = $(PathGrassland)/interceptCapGrassland10days.nc

# initial conditions: grassland_interceptSto, grassland_w1, grassland_w2, grassland_w3


[__irrPaddy]
PathIrrPaddy = $(FILE_PATHS:PathMaps)/landcover/irrPaddy

# Parameters for the Arno's scheme:
irrPaddy_arnoBeta = 0.2
# arnoBeta is defined by orographic,+ land cover add + calibration add, the soil␣
↪water capacity distribution is based on this
# range [0.01 - 1.2]

# other paramater values

irrPaddy_minInterceptCap  = 0.001
irrPaddy_cropDeplFactor   = 0.0

irrPaddy_fracVegCover  = $(PathIrrPaddy)/fracVegCover.map
irrPaddy_rootFraction1 = $(PathIrrPaddy)/rootFraction1.map
irrPaddy_rootFraction2 = $(PathIrrPaddy)/rootFraction2.map
irrPaddy_maxRootDepth  = $(PathIrrPaddy)/maxRootDepth.map
irrPaddy_minSoilDepthFrac = $(PathIrrPaddy)/minSoilDepthFrac.map

irrPaddy_cropCoefficientNC = $(PathIrrPaddy)/CropCoefficientirrPaddy_10days.nc

# maximum flooding depth for paddy
irrPaddy_maxtopwater = 0.05



# initial conditions: irrPaddy_interceptStor, irrPaddy_w1, irrPaddy_w2, irrPaddy_w3
```

(continues on next page)

(continued from previous page)

```
[__irrNonPaddy]
PathIrrNonPaddy = $(FILE_PATHS:PathMaps)/landcover/irrNonPaddy

# Parameters for the Arno's scheme:
irrNonPaddy_arnoBeta = 0.2
# arnoBeta is defined by orographic,+ land cover add + calibration add, the soil␣
↪water capacity distribution is based on this
# range [0.01 - 1.2]

# other paramater values


irrNonPaddy_minInterceptCap  = 0.001
irrNonPaddy_cropDeplFactor   = 0.0

irrNonPaddy_fracVegCover  = $(PathIrrNonPaddy)/fracVegCover.map
irrNonPaddy_rootFraction1 = $(PathIrrNonPaddy)/rootFraction1.map
irrNonPaddy_rootFraction2 = $(PathIrrNonPaddy)/rootFraction2.map
irrNonPaddy_maxRootDepth  = $(PathIrrNonPaddy)/maxRootDepth.map
irrNonPaddy_minSoilDepthFrac = $(PathIrrNonPaddy)/minSoilDepthFrac.map


irrNonPaddy_cropCoefficientNC = $(PathIrrNonPaddy)/CropCoefficientirrNonPaddy_10days.
↪nc


# initial conditions: irrNonPaddy_interceptStor, irrNonPaddy_w1, irrNonPaddy_w2,␣
↪irrNonPaddy_w3


[__sealed]
PathSealed = $(FILE_PATHS:PathMaps)/landcover/sealed

sealed_minInterceptCap  = 0.001

# initial conditions: sealed_interceptStor


[__open_water]
PathWater = $(FILE_PATHS:PathMaps)/landcover/water

water_minInterceptCap  = 0.0

#-------------------------------------------------------
[GROUNDWATER]
#-------------------------------------------------------

PathGroundwater = $(FILE_PATHS:PathMaps)/groundwater

recessionCoeff = $(PathGroundwater)/recessionCoeff.map
# baseflow = recessionCoeff * storage groundwater
specificYield = $(PathGroundwater)/specificYield.map
kSatAquifer = $(PathGroundwater)/kSatAquifer.map
# both not used at the moment in groundwater modul, but already loaded
```

(continues on next page)

(continued from previous page)

```
#-------------------------------------------------------------
# INITIAL CONDITIONS: storGroundwater


#-------------------------------------------------------
[WATERDEMAND]
#-------------------------------------------------------

PathWaterdemand = $(FILE_PATHS:PathMaps)/landsurface/waterDemand
# For water demand vs. availability: areas have to be aggregated
# Allocation map
allocSegments = $(PathWaterdemand)/catchx.nc

domesticWaterDemandFile = $(PathWaterdemand)/domesticWaterDemand.nc
industryWaterDemandFile = $(PathWaterdemand)/industryWaterDemand.nc

irrNonPaddy_efficiency = $(FILE_PATHS:PathMaps)/landsurface/waterDemand/efficiency.nc
irrPaddy_efficiency = $(FILE_PATHS:PathMaps)/landsurface/waterDemand/efficiency.nc

#irrNonPaddy_efficiency = 0.8
#irrPaddy_efficiency = 0.8
irrigation_returnfraction = 0.5

# -----------------------------------------------------------
# Estimate of fractions of groundwater and surface water abstractions
# Either a fixed fraction for surface water abstration
# based on fraction of average baseflow and upstream average discharge
# if swAbstractionFrac < 0: fraction is taken from baseflow / discharge
# if swAbstractionFrac > 0 this value is taken as a fixed value
swAbstractionFrac = 0.9
averageDischarge = $(FILE_PATHS:PathOut)/discharge_totalavg_rhine30min.nc
# in [m3/s]
averageBaseflow  = $(FILE_PATHS:PathOut)/baseflow_totalavg_rhine30min.nc
# in [m3/s]
baseflowInM = True
# if baseflow is in [m] instead of [m3/s] it will be converted


#-------------------------------------------------------
# RUNOFF CONCENTRATION
#-------------------------------------------------------
[RUNOFF_CONCENTRATION]

# using triagular weigthning method
# the bigger the factor, more lag time
forest_runoff_peaktime = 1.0
grassland_runoff_peaktime = 0.5
irrPaddy_runoff_peaktime = 0.5
irrNonPaddy_runoff_peaktime = 0.5
sealed_runoff_peaktime = 0.15
water_runoff_peaktime = 0.01

interflow_runoff_peaktime =1.0
baseflow_runoff_peaktime = 2.0

# initial conditions:
# here only 1 layer is shown, but there are up to 10: runoff_concIni
```

(continues on next page)

(continued from previous page)

```
#-------------------------------------------------------
# ROUTING MAPS and PARAMETERSD
#-------------------------------------------------------
[ROUTING]

PathRouting = $(FILE_PATHS:PathMaps)/routing

# Number of substep per day
# should be 10 for 0.5 deg but 24 for 0.1 deg

NoRoutingSteps = 10
#kinematic wave parameter: 0.6 is for broad sheet flow
chanBeta = 0.6

# Channel gradient (fraction, dy/dx)
chanGrad = $(PathRouting)/kinematic/changrad.nc
# Minimum channel gradient (for kin. wave: slope cannot be 0)
chanGradMin = 0.0001

#Channel Manning's n
chanMan = $(PathRouting)/kinematic/chanman.nc
#Channel length [meters]
chanLength = $(PathRouting)/kinematic/chanleng.nc
#Channel bottom width [meters]
chanWidth = $(PathRouting)/kinematic/chanbw.nc
#Bankfull channel depth [meters]
chanDepth = $(PathRouting)/kinematic/chanbnkf.nc

# initial conditions: channelStorageIni, riverbedExchangeIni, dischargeIni

#-------------------------------------------------------
# LAKES AND RESERVOIRS
#-------------------------------------------------------
[LAKES_RESERVOIRS]

PathLakesRes = $(FILE_PATHS:PathMaps)/routing/lakesreservoirs

# Use reservoirs and lakes (otherwise use only lakes Lake ID=1 and 3 => natural␣
↪conditions)
useResAndLakes = True
# Reservoirs do have a year of implementation
dynamicLakesRes = True
# if Reservoirs does not have a year of implemtation, which year should be taken as␣
↪fixed year
fixLakesResYear = 1950


#----------------------------
#Big lakes and Reservoirs

# ID of every lake, reservoir from HydroLakes database
waterBodyID = $(PathLakesRes)/lakesResID.nc
# 1 for lake, 2 for reservoir, 3 for lake and reservoir
waterBodyTyp = $(PathLakesRes)/lakesResType.nc
# Avergae discharge from HydroLakes Database
```

(continues on next page)

(continued from previous page)

```
waterBodyDis = $(PathLakesRes)/lakesResDis.nc

# Lakes surface area from HydroLakes Database
waterBodyArea = $(PathLakesRes)/lakesResArea.nc
# a factor to scale the outlet of a lake
#lakeAFactor = 1.0  -> calibration

#----------------------------
# Small lakes and reservoirs

useSmallLakes = True

smallLakesRes = $(PathLakesRes)/smallLakesRes.nc
smallwaterBodyDis = $(PathLakesRes)/smalllakesresDis.nc

# averageRunoff in [m] (if not given smallwaterBodyDis is taken instead)
#averageRunoff  = $(FILE_PATHS:PathOut)/runoff_totalavg_cali.nc

# for water demand
#min storage in [m3] (if not give it is calculated)
#minStorage  = $(FILE_PATHS:PathOut)/minsmalllakeStorage_cali.nc


# initial conditions: lakeInflowIni, lakeStorageIni, outLakeIni, lakeOutflowIni,␣
↪reservoirStorageIni

#------------------
# Reservoirs
# reservoir volume from HydroLakes database
waterBodyVolRes = $(PathLakesRes)/lakesResVolRes.nc
# reservoir starting year from HydroLakes database
waterBodyYear = $(PathLakesRes)/lakesResYear.nc

#  Conservative, normal and flood storage limit (fraction of total storage, [-])
conservativeStorageLimit = 0.1
#normalStorageLimit = 0.5   # --> put into calibration
floodStorageLimit = 0.9
# adjusting the balance between normal and flood storage
# [0 ..1]  0: NormalstorageLimit      1: (= closer to flood) results in keeping the␣
↪normal qoutflow longer constant
adjust_Normal_Flood = 0.5

# Minimum, Normal and Non-damaging reservoir outflow  (fraction of average discharge,␣
↪[-])
MinOutflowQ = 0.2
NormalOutflowQ = 1.0
NonDamagingOutflowQ = 4.0

# initial conditions: lakeInflowIni, lakeStorageIni, outLakeIni, lakeOutflowIni,␣
↪reservoirStorageIni


#-------------------------------------------------------
[INFLOW]
#-------------------------------------------------------

# if option inflow = true
```

(continues on next page)

(continued from previous page)

```
# the inflow from outside is added at inflowpoints
In_Dir = $(FILE_PATHS:PathRoot)/in

# nominal map with locations of (measured)inflow hydrographs [cu m / s]
InflowPoints = $(In_Dir)/in.map
#InflowPoints = 8.25 49.75 7.75 50.25

# if InflowPoints is a map, this flag is to identify if it is global (False) or local␣
↪(True)
# observed or simulated input hydrographs as time series [cu m / s]
# Note: that identifiers in time series have to correspond to InflowPoints
# can be several timeseries in one file or different files e.g. main.tss mosel.tss
#QInTS = main1.tss mosel1.tss
QInTS = mm.tss



#-------------------------------------------------------
[ENVIRONMENTALFLOW]
#-------------------------------------------------------

# Either calculate without run with predone discharge (set calc_ef_after = False)
calc_ef_after = True
# Or calculate after run (set calc_ef_after = False) and defining the file to be used
EFDis = $(FILE_PATHS:PathOut)/discharge_rhine.nc

# if predone discharge, do the maps need to be cut to fit to the mask?
cut_ef_map = False

EnvironmentalFlowFile = $(FILE_PATHS:PathOut)/MQ90_12month.nc

# MAF: Mean, Q90: percentile 90, MMF: monthly average, MQ90: monthly Q90 9averagwed␣
↪over al Jan, Feb..
# EF_VMF: Environmental flow - variable monthly flow, EF_VMF_LIH - EF- variable␣
↪monthly flow, high intermediate, low class
OUT_Dir = $(FILE_PATHS:PathOut)
#OUT_MAP_Once = MAF, Q90
#OUT_MAP_12month = MMF, MQ90, EF_VMF, EF_VMF_LIH
#OUT_MAP_12month = MQ90, EF_VMF



#+++++++++++++++++++++++++++++++++++++++++++++++++++++++
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++


[OUTPUT]

# OUTPUT maps and timeseries
OUT_Dir = $(FILE_PATHS:PathOut)

OUT_TSS_Daily = discharge
#OUT_TSS_MonthAvg = discharge
#OUT_TSS_AnnualAvg = discharge

#OUT_Map_Daily = discharge
#OUT_Map_MonthAvg = discharge, precipitation, runoff
```

(continues on next page)

(continued from previous page)

```
#OUT_Map_AnnualAvg = discharge
#OUT_MAP_TotalAvg = discharge, baseflow
```

4.4 NetCDF meta data

4.4.1 Output Meta NetCDF information

The metaNetcdf.xml includes information on the output netCDF files e.g. description of the parameter, unit ..

Example of a metaNetcdf.xml file:

```
<CWATM>
# METADATA for NETCDF OUTPUT DATA

# varname: name of the variable in the CWAT code
# unit: unit of the varibale
# long name# standard name

# Discharge maps
<metanetcdf varname="discharge" unit="m3/s"  standard_name="Discharge" long_name=
↪"Discharge in cubic meter per second"  title="1st Demo CWATM" author="PB" />

# others
<metanetcdf varname="soilmoisture" unit="mm"  standard_name="soil moisture" long_name=
↪"Soil moisture"  title ="1st Demo CWATM" author="PB" />

# Initial condition Files
<metanetcdf varname="initcondition"  purpose ="Initial Conditions CWATM" author="PB" /
↪>
<metanetcdf varname="SnowCover1" unit="mm"  standard_name="SnowCover1" long_name=
↪"Snow cover top layer"  />
<metanetcdf varname="SnowCover2" unit="mm"  standard_name="SnowCover2" long_name=
↪"Snow cover middle layer"  />
<metanetcdf varname="SnowCover3" unit="mm"  standard_name="SnowCover3" long_name=
↪"Snow cover lower layer"  />
<metanetcdf varname="FrostIndex" unit="degree/days"  standard_name="FrostIndex" long_
↪name="Frost index based on Molnau, Bissel (1983)"  />
</CWATM>
```

4.4.2 Name and location of the NetCDF meta data file

In the settings file the name and location of the metadata file is given.

```
#-------------------------------------------------------
[NETCDF_ATTRIBUTES]
institution = IIASA
title = Global Water Model - WATCH WDFEI
metaNetcdfFile = $(FILE_PATHS:PathRoot)/CWATM/source/metaNetcdf.xml
```

4.5 Initialisation

CWATM needs to have estimates of the initial state of the internal storage variables, e.g. the amount of water stored in snow, soil, groundwater etc.

There are two possibilities:

1. The initial state of the internal storage variables are unknown and a **first** guess has to be used e.g. all storage variables are half filled.
2. The initial state is known from a previous run, where the variables are stored at a certain time step. This is called **warm start**

The the **warm start** is usful for:

- using a long pre-run to find the steady-state storage of the groundwater storage and use it as initial value
- using the stored variables to shorten the warm-up period
- using the stored variables to restart every day with the values from the pre3vious day (forecasting mode)

4.5.1 Example of soil moisture

The next figure shows the impact of different initial condition on the soil moisture of the lower soil. In one of the simulations the soil is initially almost ompletely saturated. In another simulation the soil is completely dry and the third simulation starts with initial conditions in between the two extremes.

In the beginning the effect of different initial condition can be seen clearly. But after one year the three curves converge. The **memory** of the lower soil goes back for about one year.

For all the initial condition apart from groundwater the memory is about 12 month. That means practically a spin-up of one year is sufficient to habve enough warm-up time.

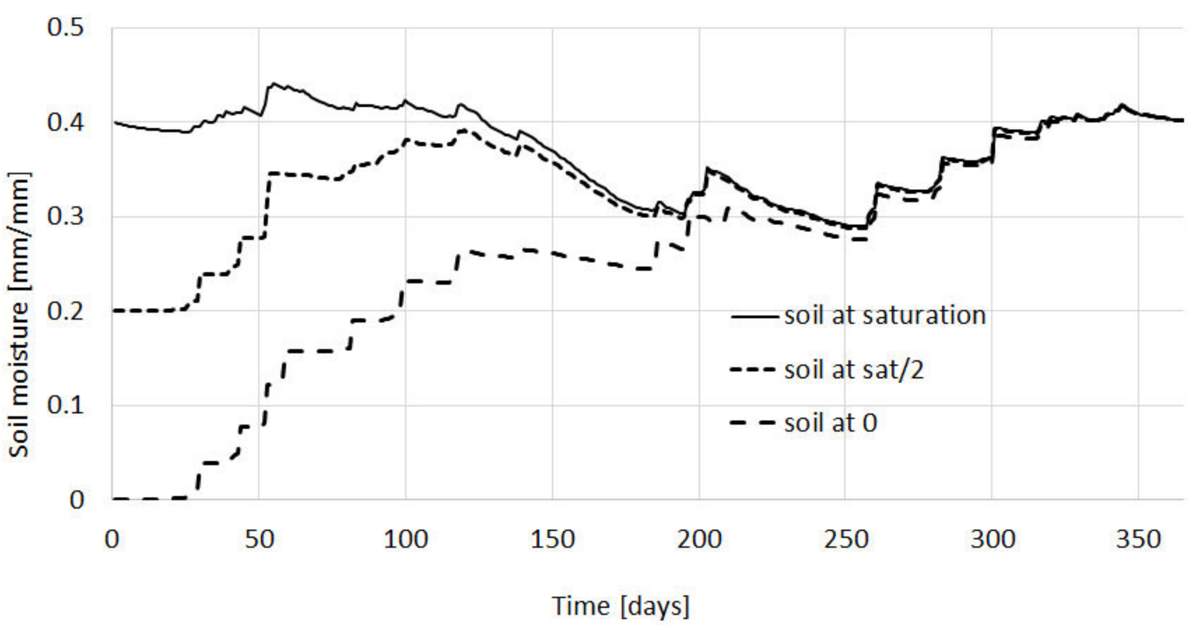


Figure: Simulation of soil moisture in the lower soil with different initial conditions

For the groundwater zone a longer warm-up period is needed, because of the slow response of groundwater. Here a rather fast reacting groundwater storage is shown with the three curves coverge after two years.

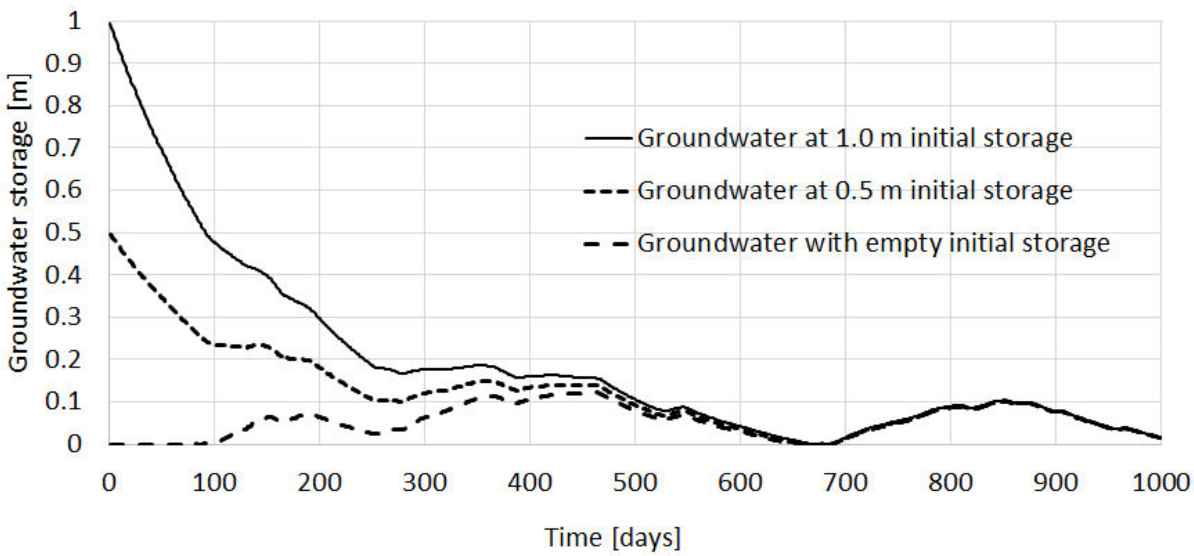


Figure: Simulation of groundwater storage with different initial conditions

4.5.2 Cold start

For a **cold start** the values of the storage variables are unknown and set to a "first" guess. A list of variables and their default value for a **cold start** is given below in: *Initial conditions*

Set up a cold start in the settingsfile

In the settings file the option: **load_initial** has to be set on **False**

```
#-------------------------------------------------------
[INITITIAL CONDITIONS]
#-------------------------------------------------------

# for a warm start initial variables a loaded
# e.g for a start on 01/01/2010 load variable from 31/12/2009
load_initial = False
initLoad = $(FILE_PATHS:PathRoot)/init/Rhine_19891231.nc
```

Note: It is possible to exclude the warming up period of your model run for further analysis of results by setting the **SpinUp** option

```
[TIME-RELATED_CONSTANTS]
SpinUp =  01/01/1995
```

4.5.3 Storing initial variables

In the settings file the option **save_intitisal** has to be set to **True**

The name of the initial netCDF4 file has to be put in **initsave** and one or more dates have to be specified in StepInit

```
# saving variables from this run, to initiate a warm start next run
# StepInit = saving date, can be more than one: 10/01/1973 20/01/1973
save_initial = False
initSave = $(FILE_PATHS:PathRoot)/init/Rhine
StepInit = 31/12/1989 31/12/2010
```

4.5.4 Warm start

CWATM can write internal variables to a netCDF file for choosen timesteps. These netCDF files can be used as the initial conditions for a suceeding simulation.

This is useful for establishing a steady-state with a long-term run and then using this steady-state for succeding simulations or for an every day run (forecasting mode)

Warning: If the parameters are changes after a run(especially the groundwater parameters) the stored initial values do not represent the conditions of the storage variables. Stored initial conditions should **not** be used as initial values for a model run with another set of parameters. If you do this during calibration, you will not be able to reproduce the calibration results!

Set up a cold start in the settingsfile

In the settings file the option: **load_initial** has to be set on **True** And define the name of the netcdf4 file in **initLoad**

Note: Use the initial values of the previous day here. E.g. if you run the model from 01/01/2006 use the inital condition from 31/12/2005

```
#-------------------------------------------------------
[INITITIAL CONDITIONS]
#-------------------------------------------------------

# for a warm start initial variables a loaded
# e.g for a start on 01/01/2010 load variable from 31/12/2009
load_initial = False
initLoad = $(FILE_PATHS:PathRoot)/init/Rhine_19891231.nc
```

4.5.5 Initial conditions

No.	Variable	Description	Default value	Number of maps
1	SnowCover	Snow cover for up to 7 zones	0	7
2	FrostIndex	Degree days frost threshold	0	1
3	Forest state	Interception storage	0	1
		Top water layer	0	1
		Soil storage for 3 soil layers	0	3
4	Grassland state	Interception storage	0	1
		Top water layer	0	1
		Soil storage for 3 soil layers	0	3
5	Paddy irrigation state	Interception storage	0	1
		Top water layer	0	1
		Soil storage for 3 soil layers	0	3
6	Irrigation state	Interception storage	0	1
		Top water layer	0	1
		Soil storage for 3 soil layers	0	3
7	Sealed area state	Interception storage	0	1
8	Groundwater	Groundwater storage	0	1
9	Runoff concentration	10 layers of runoff concentration	0	10
10	Routing	Channel storage	0.2 * total cross section	1
	Routing	Riverbed exchange	0	1
	Routing	Discharge	depending on ini channel stor.	1
11	Lakes and Reservoirs	Lake inflow	from HydroLakes database	1
		Lake outflow	same as lake inflow	1
		Lake&Res outflow to other lakes&res	same as lake inflow	1
		Lake storage	based on inflow and lake area	1
		Reservoir storage	0.5 * max. reservoir storage	1
		Small lake storage	based on inflow and lake area	1
		Small lake inflow	from HydroLakes database	1
		Small lake outflow	same as small lake inflow	1

4.6 Model Output

An advantage of **CWATM** is the full flexibility of the output variables.

- All parameters and variables can be used for output as maps or time series.
- Even if the model is run at daily timestep, output can be daily, monthly, annual, at the end of a run
- all variables maps are stored as netcdf and the meta data information can be added

4.6.1 Time depending and non depending output maps

Output maps will be produced as spatial maps, stack of spatial maps (over time)
Format: netCDF

The netCDF maps can be read with:

Windows

- Panoply

Linux

- ncview
- cdo

4.6.2 Or time series at specified points

Timeseries are procuded as ASCII files, which can be read with every text editor
or with PCRaster Aquila

The specific point where timeseries are provided are defined in the settings file as *Gauges*:

```
# Station data
# either a map e.g. $(FILE_PATHS:PathRoot)/data/areamaps/area3.map
# or a location coordinates (X,Y) e.g. 5.75 52.25 9.25 49.75 )
# Lobith/Rhine
Gauges = 6.25 51.75

# if .tif file for gauges, this is a flag if the file is global or local
# e.g. Gauges = $(FILE_PATHS:PathRoot)/data/areamaps/gaugesRhine.tif
GaugesLocal = True
```

4.6.3 Output variables

Output can be every global defined variable in the model Variable are e.g. Precipitation, runoff, baseflow

but also not so common variables as:

- reservoirStorage (amount of water in the reservoirs in [m3])
- nonIrrReturnFlowFraction (returnflow from domenstic and industrial water use [m3])
- actualET[1] (actual evapotranspiration from grassland [m/day])
- …

4.6.4 Daily, monthly - at the end or average

- per day
- total month, average month, end of month

- total year, average year, end of year
- total average, total at the end

for example

```
[OUTPUT]
# OUTPUT maps and timeseries
OUT_Dir = $(FILE_PATHS:PathOut)
OUT_MAP_Daily = discharge, runoff
OUT_MAP_MonthAvg = Precipitation
OUT_MAP_TotalEnd = lakeStorage
OUT_MAP_TotalAvg = Tavg

OUT_TSS_Daily = discharge
OUT_TSS_AnnualAvg = Precipitation
```

Note: For each variable the meta data information can be defined in *Output Meta NetCDF information*

Note: For information how to adjust the output in the settings file see *Output*

4.6.5 Most important output variables - a selection

```
#Variable name     : Description
discharge          : river discharge
runoff             : runoff
Precipitation      : rainfall + snow
Tavg               : average temperature
ETRef: potential   : evaporation from reference soil
sum_gwRecharge     : total groundwater recharge
totalET            : total actual evapotranspiration
baseflow           : baseflow from groundwater
... (to be continued)
```

4.6.6 Output variables - starting a list

A list of variables can be produced by using:
grep -d recurse 'self.var.' *.py
Every self.var.variable can be used as output variable
For a description of the variable please take a look at the python module itself.
.
As output variable please use without self.var.

```
#Python_modul             Variable_name
capillarRise.py           self.var.capRiseFrac
evaporationPot.py         self.var.AlbedoCanopy
evaporationPot.py         self.var.AlbedoSoil
evaporationPot.py         self.var.AlbedoWater
```

(continues on next page)

(continued from previous page)

```
evaporationPot.py        self.var.ETRef
evaporationPot.py        self.var.EWRef
evaporation.py           self.var.potBareSoilEvap
evaporation.py           self.var.snowEvap
evaporation.py           self.var.SnowMelt
evaporation.py           self.var.potBareSoilEvap
evaporation.py           self.var.cropKC[No]
evaporation.py           self.var.totalPotET[No]
evaporation.py           self.var.potTranspiration[No]
groundwater.py           self.var.recessionCoeff
groundwater.py           self.var.specificYield
groundwater.py           self.var.kSatAquifer
groundwater.py           self.var.storGroundwater
groundwater.py           self.var.baseflow
interception.py          self.var.interceptCap[No]
interception.py          self.var.interceptStor[No]
interception.py          self.var.availWaterInfiltration[No]
interception.py          self.var.potTranspiration[No]
interception.py          self.var.actualET[No]
lakes_reservoirs.py      self.var.waterBodyID
lakes_reservoirs.py      self.var.waterBodyOut
lakes_reservoirs.py      self.var.lakeArea
lakes_reservoirs.py      self.var.lakeDis0
lakes_reservoirs.py      self.var.lakeAC
lakes_reservoirs.py      self.var.lakeEvaFactor
lakes_reservoirs.py      self.var.reslakeoutflow
lakes_reservoirs.py      self.var.lakeVolume
lakes_reservoirs.py      self.var.lakeStorage
lakes_reservoirs.py      self.var.lakeInflow
lakes_reservoirs.py      self.var.lakeOutflow
lakes_reservoirs.py      self.var.reservoirStorage
lakes_reservoirs.py      self.var.lakeResStorage
lakes_reservoirs.py      self.var.sumlakeResInflow
lakes_reservoirs.py      self.var.sumlakeResOutflow
lakes_res_small.py       self.var.smalllakeArea
lakes_res_small.py       self.var.smalllakeDis0
lakes_res_small.py       self.var.smalllakeA
lakes_res_small.py       self.var.smalllakeFactor
lakes_res_small.py       self.var.smalllakeVolumeM3
lakes_res_small.py       self.var.smallevapWaterBodyStorage
landcoverType.py         self.var.coverTypes
landcoverType.py         self.var.totalET
landcoverType.py         self.var.actSurfaceWaterAbstract
landcoverType.py         self.var.minInterceptCap
landcoverType.py         self.var.interceptStor[No]
landcoverType.py         self.var.sum_interceptStor
landcoverType.py         self.var.minCropKC
landcoverType.py         self.var.maxGWCapRise
... (to be continued)
```

CHAPTER 5

Tutorial

Contents

5.1 Requirements

5.1.1 Requirements

Python version

NEW from 2019 on: Requirements are a 64 bit Python 3.7.x version

> **Warning:** a 32 bit version is not able to handle the data requirements!

> **Warning:** From 2019 on we are changing to Python37. We do not provide further support for Python 2.7

Libraries

These external libraries are needed:

- Numpy
- Scipy
- netCDF4
- GDAL

Windows

The four libraries can be installed with pip or downloaded at Unofficial Windows Binaries for Python Extension Packages

Windows executeable Python version

A CWATM executable cwatm.exe can be used instead of the Python version

5.2 Test the model

Windows and Linux

python <modelpath>/cwatm.py

The output should be:

```
Running under platform:  Windows  **(or Linux etc)**
CWatM - Community Water Model
Authors: ...
Version: ...
Date: ...
Arguments list:
settings.ini     settings file
-q --quiet       output progression given as .
-v --veryquiet   no output progression is given
```

(continues on next page)

(continued from previous page)

```
-l --loud         output progression given as time st
-c --check        input maps and stack maps are check
-h --noheader     .tss file have no header and start
-t --printtime    the computation time for hydrologic
-w --warranty     copyright and warranty information
```

Warning: If python is not set in the environment path, the full path of python has to be used

5.2.1 Error because the python libraries are installed incorrectly

If the model is causing an error at this stage, please check the python libraries:

```
python
import numpy
import scipy.ndimage
import gdal
import netCDF4
```

5.3 Running the model 1

Warning: The model needs a settings file as an argument. See: *Settings file*

python <modelpath>/cwatm.py settingsfile flags

example:

```
python cwatm.py settings_rhine.ini -l
```

The flag -l show the output on screen as date and discharge

At this point you should receive this eror message:

```
======================== CWATM FILE ERROR ===========================
Cannot find option file: d:/work/CWATM/source/metaNetcdf.xml In  "metaNetcdfFile"
searching: "d:/work/CWATM/source/metaNetcdf.xml"
path: d:/work/CWATM/source does not exists
```

5.4 Downloading and installing the spatial dataset

The spatial dataset contains:

- static data ie. data that does not change over time (a model assumption) e.g. soil data
- time dependend (inter annual) data that change periodical during a year e.g. crop coefficient of vegetation
- time dependend (intra annual) data that change by month or year e.g. fraction of landcover

These data are stored as global dataset:

- cwat_input.zip for the 30' global version
- cwat_input5min.zip for the 5' global version

As climate data different forcings can be used e.g:

- PGMFD v.2 (Princeton), GSWP3, etc.
- precipitation from e.g. MSWEP http://www.gloh2o.org/
- WATCH+WFDEI https://www.isimip.org/gettingstarted/details/5/

and as projection e.g.:

- ISI-MIP dataset https://www.isimip.org/gettingstarted/#input-data-bias-correction

For the tutorial we cut out Rhine basin and included the WATCH+WFDEI precipitation, average temperature and the calculated potential evaporation .
A 30' and a 5' version can be found on FTP in rhine/climate

Reference:

Weedon, G.P., S.S. Gomes, P.P. Viterbo, W.J. Shuttleworth, E.E. Blyth, H.H. Österle, J.C. Adam, N.N. Bellouin, O.O. Boucher, and M.M. Best, 2011: Creation of the WATCH Forcing Data and Its Use to Assess Global and Regional Reference Crop Evaporation over Land during the Twentieth Century. J. Hydrometeor., 12, 823–848, doi: 10.1175/2011JHM1369.1

Weedon, G. P., G. Balsamo, N. Bellouin, S. Gomes, M. J. Best, and P. Viterbo (2014), The WFDEI meteorological forcing data set: WATCH Forcing Data methodology applied to ERA-Interim reanalysis data, Water Resour. Res., 50, 7505–7514, doi:10.1002/2014WR015638.

Note:

Please copy and unpack the spatial dataset (either 30' or 5')in a folder
Please copy the the climate dataset 30min_meteo_rhine.zip or 5min_meteo_rhine.zip in a seperate folder
Please create a folder called output

5.5 Changing the Settings file

to run the model the pathes to data have to be set correctly: The information of pathes are stored in the settings file around line 80-100

[FILE_PATHS]:

```
PathRoot = E:/
PathOut = $(PathRoot)/output
PathMaps = E:/cwatm_input
PathMeteo = E:/climate
#-------------------------------------
[NETCDF_ATTRIBUTES]
institution = IIASA
title = Global Water Model - WATCH WDFEI
metaNetcdfFile = $(FILE_PATHS:PathRoot)/CWATM/source/metaNetcdf.xml
```

Note: Please change the pathes according to your file system

5.6 Running the model 2

If you type now:

```
python cwatm.py settings_rhine.ini -l
```

You should see:

```
E:\CWATM_rhine\source>python cwatm.py settings_rhine30min.ini -l
CWATM - Community Water Model  Version: 0.991  Date:  16/09/2017
International Institute of Applied Systems Analysis (IIASA)
Running under platform:  Windows
-----------------------------------------------------------
CWATM Simulation Information and Setting
The simulation output as specified in the settings file: settings_rhine30min.ini
can be found in E:/CWATM_rhine/output
Step         Date   Discharge
1      01/01/1961        4.20
2      02/01/1961        4.23
...
```

If you don't see this. Something went wrong and you mifght see this instead:

```
E:\CWATM_rhine\source>python cwatm.py settings_rhine30min.ini -l
CWATM - Community Water Model  Version: 0.991  Date:  16/09/2017
International Institute of Applied Systems Analysis (IIASA)
Running under platform:  Windows
-----------------------------------------------------------
ERROR 4: `E:/CWATM_rhine/cwatm_input/routing/ldd.map' does not exist in the file
↪system,
and is not recognised as a supported dataset name.
management_modules.messages.CWATMFileError:
======================== CWATM FILE ERROR ===========================
In  "Ldd"
searching: "E:/CWATM_rhine/cwatm_input/routing/ldd.map"
path: E:/CWATM_rhine/cwatm_input/routing does not exists
```

The model tries to help you on finding the error.
In this case it is looking for the river network map ldd.map or ldd.nc or ldd.tif
but it cannot find the file and not even the path to the file.

Here you might change:

```
[FILE_PATHS]
PathRoot = E:/CWATM_rhine
PathMaps = $(PathRoot)/cwatm_input
```

or:

```
[TOPOP]
# local drain direction map (1-9)
Ldd = $(FILE_PATHS:PathMaps)/routing/ldd.map
```

But many other error can occure too! Have fun.

5.7 Changing parameters of the model

Note: An overview of possibilities is given in see *Settings file*

5.8 Changing the Output

5.8.1 Output variables

Output can be every global defined variable in the model Variable are e.g. Precipitation, runoff, baseflow

but also not so common variables as:

- reservoirStorage (amount of water in the reservoirs in [m3])
- nonIrrReturnFlowFraction (returnflow from domenstic and industrial water use [m3])
- actualET[1] (actual evapotranspiration from grassland [m/day])
- ...

5.8.2 Daily, monthly - at the end or average

- per day
- total month, average month, end of month
- total year, average year, end of year
- total average, total at the end

for example

```
[OUTPUT]
# OUTPUT maps and timeseries
OUT_Dir = $(FILE_PATHS:PathOut)
OUT_MAP_Daily = discharge, runoff
OUT_MAP_MonthAvg = Precipitation
OUT_MAP_TotalEnd = lakeStorage
OUT_MAP_TotalAvg = Tavg

OUT_TSS_Daily = discharge
OUT_TSS_AnnualAvg = Precipitation
```

Note: For each variable the meta data information can be defined in *Output Meta NetCDF information*

Note: For information how to adjust the output in the settings file see *Output*

5.8.3 Most important output variables - a selection

```
#Variable name    : Description
discharge         : river discharge
runoff            : runoff
Precipitation     : rainfall + snow
Tavg              : average temperature
ETRef: potential  : evaporation from reference soil
sum_gwRecharge    : total groundwater recharge
totalET           : total actual evapotranspiration
baseflow          : baseflow from groundwater
... (to be continued)
```

5.8.4 Output variables - starting a list

A list of variables can be produced by using:
grep -d recurse 'self.var.' *.py
Every self.var.variable can be used as output variable
For a description of the variable please take a look at the python module itself.

As output variable please use without self.var.

```
#Python_modul            Variable_name
capillarRise.py          self.var.capRiseFrac
evaporationPot.py        self.var.AlbedoCanopy
evaporationPot.py        self.var.AlbedoSoil
evaporationPot.py        self.var.AlbedoWater
evaporationPot.py        self.var.ETRef
evaporationPot.py        self.var.EWRef
evaporation.py           self.var.potBareSoilEvap
evaporation.py           self.var.snowEvap
evaporation.py           self.var.SnowMelt
evaporation.py           self.var.potBareSoilEvap
evaporation.py           self.var.cropKC[No]
evaporation.py           self.var.totalPotET[No]
evaporation.py           self.var.potTranspiration[No]
groundwater.py           self.var.recessionCoeff
groundwater.py           self.var.specificYield
groundwater.py           self.var.kSatAquifer
groundwater.py           self.var.storGroundwater
groundwater.py           self.var.baseflow
interception.py          self.var.interceptCap[No]
interception.py          self.var.interceptStor[No]
interception.py          self.var.availWaterInfiltration[No]
interception.py          self.var.potTranspiration[No]
interception.py          self.var.actualET[No]
lakes_reservoirs.py      self.var.waterBodyID
lakes_reservoirs.py      self.var.waterBodyOut
```

(continues on next page)

(continued from previous page)

```
lakes_reservoirs.py      self.var.lakeArea
lakes_reservoirs.py      self.var.lakeDis0
lakes_reservoirs.py      self.var.lakeAC
lakes_reservoirs.py      self.var.lakeEvaFactor
lakes_reservoirs.py      self.var.reslakeoutflow
lakes_reservoirs.py      self.var.lakeVolume
lakes_reservoirs.py      self.var.lakeStorage
lakes_reservoirs.py      self.var.lakeInflow
lakes_reservoirs.py      self.var.lakeOutflow
lakes_reservoirs.py      self.var.reservoirStorage
lakes_reservoirs.py      self.var.lakeResStorage
lakes_reservoirs.py      self.var.sumlakeResInflow
lakes_reservoirs.py      self.var.sumlakeResOutflow
lakes_res_small.py       self.var.smalllakeArea
lakes_res_small.py       self.var.smalllakeDis0
lakes_res_small.py       self.var.smalllakeA
lakes_res_small.py       self.var.smalllakeFactor
lakes_res_small.py       self.var.smalllakeVolumeM3
lakes_res_small.py       self.var.smallevapWaterBodyStorage
landcoverType.py         self.var.coverTypes
landcoverType.py         self.var.totalET
landcoverType.py         self.var.actSurfaceWaterAbstract
landcoverType.py         self.var.minInterceptCap
landcoverType.py         self.var.interceptStor[No]
landcoverType.py         self.var.sum_interceptStor
landcoverType.py         self.var.minCropKC
landcoverType.py         self.var.maxGWCapRise
... (to be continued)
```

CHAPTER 6

Demo of the model

6.1 Resolution

CWATM can be run globally at 0.5° or separately for any basin or any clipping of a global map.
Depending on the data provided the model can also run for any other resolutions (e.g. 5 arcmin).
Timestep is daily, output of maps, time series can be daily, monthly, yearly

Here some outputs of the global run on 0.5° are shown:

6.2 Demo 1 - NetCDF videos

6.2.1 Global discharge

One year run example: 1/1/1991- 31/12/1992

6.2.2 Global potential evaporation [mm/day]

One year run example

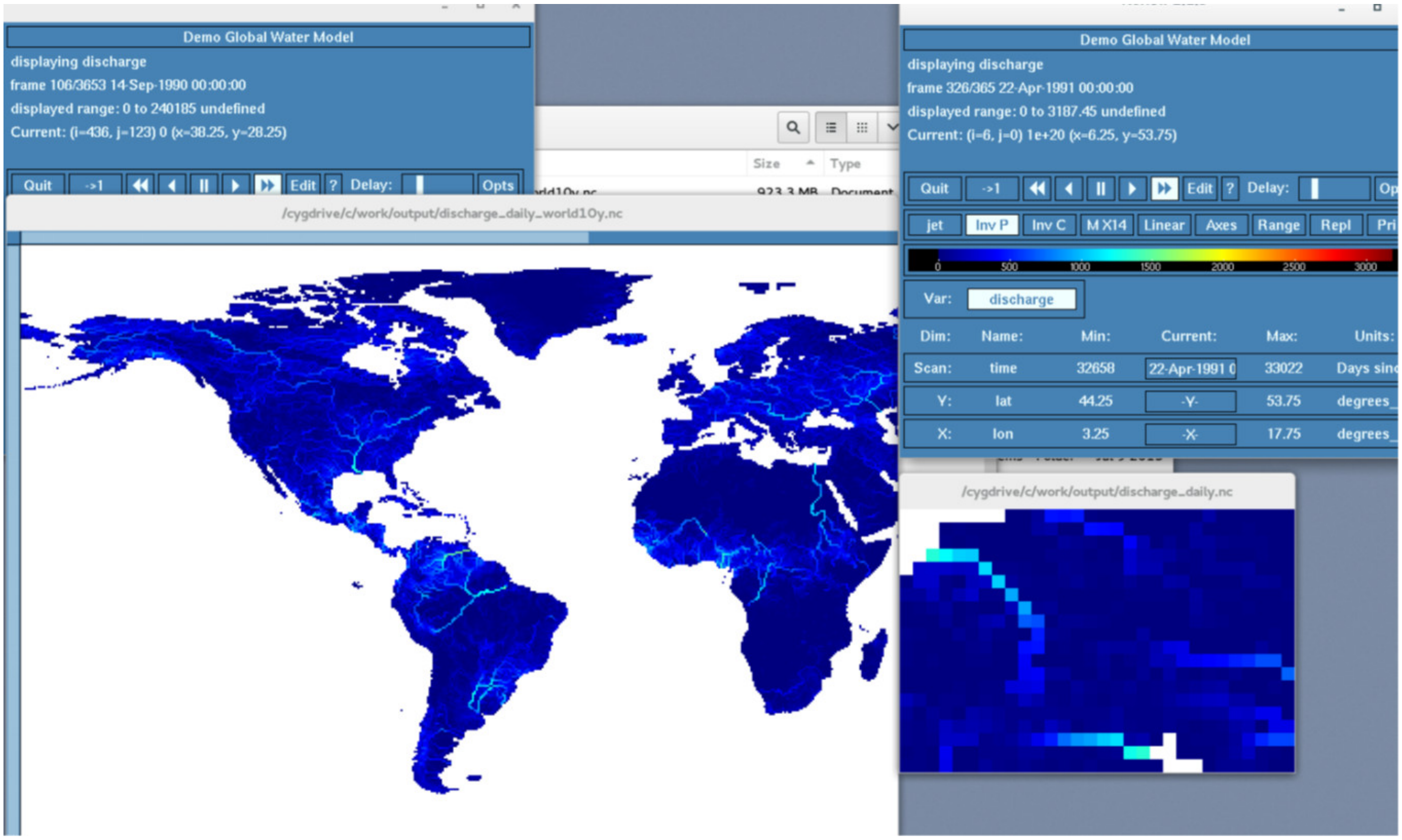
6.2.3 Global soil moisture [mm/mm]

One year run example

6.3 Demo 2 - NcView output

Global discharge as world map

Output from NcView



6.4 Demo 3 - NcView timeserie

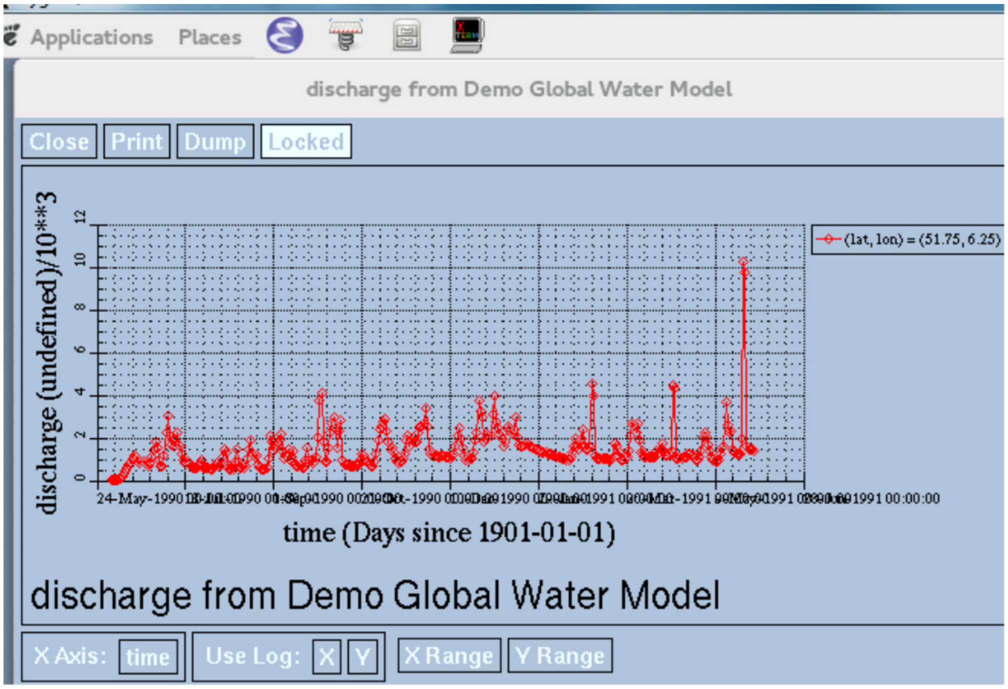
Discharge as timeseries
Output from NcView

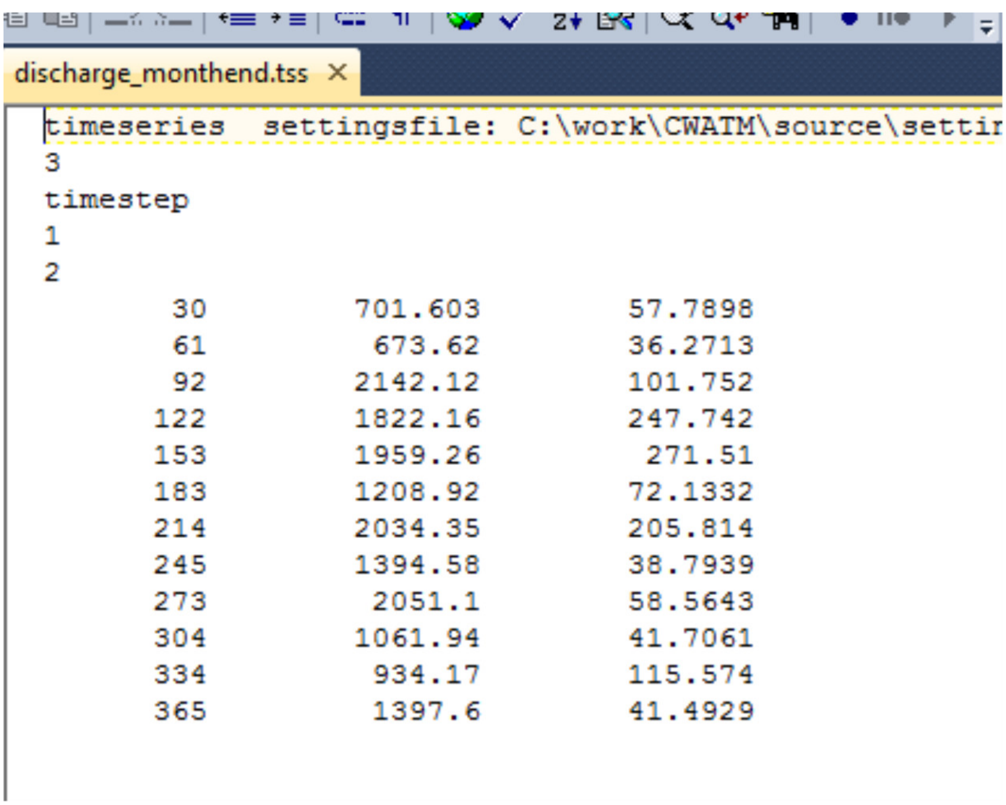
6.5 Demo 4 - Monthly timeserie

Discharge as monthly timeseries

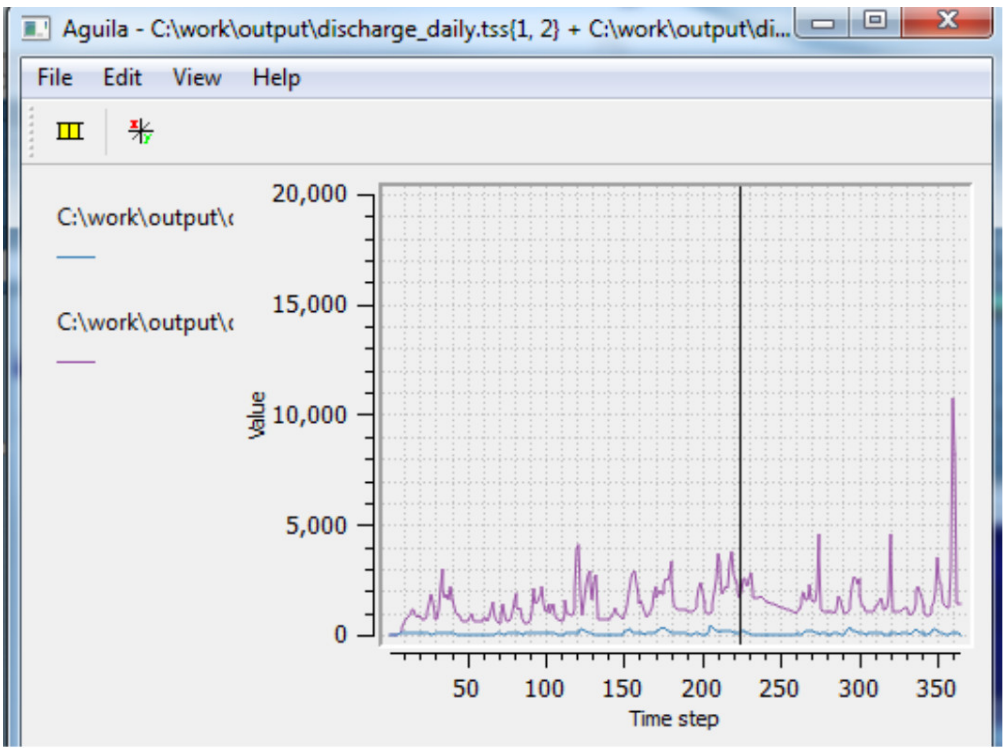
6.6 Demo 5 - PCRaster Aquila output

Discharge as timeseries Output from PCRaster Aquila




discharge_monthend.tss

```
timeseries  settingsfile: C:\work\CWATM\source\settir
3
timestep
1
2
         30         701.603        57.7898
         61          673.62        36.2713
         92         2142.12        101.752
        122         1822.16        247.742
        153         1959.26         271.51
        183         1208.92        72.1332
        214         2034.35        205.814
        245         1394.58        38.7939
        273          2051.1        58.5643
        304         1061.94        41.7061
        334          934.17        115.574
        365          1397.6        41.4929
```


Aguila - C:\work\output\discharge_daily.tss{1, 2} + C:\work\output\di...
File Edit View Help
C:\work\output\
C:\work\output\
20,000
15,000
10,000
5,000
0
Value
50 100 150 200 250 300 350
Time step

CHAPTER 7

The Model Itself

Contents

7.1 Performance

Computational run time (on a linux single node - 2400 MHz with Intel Xeon CPU E5- 2699A v4):

Daily timestep on 0.5 deg

Global: 100 years in appr. 12h = 7.2min per year

	Process	sum % runtime
1	Read Meteo Data	6.2
2	Et pot	7.6
3	Snow	8.8
4	Soil	59.4
5	Groundwater	59.5
6	Runoff conc	70.1
7	Lakes	70.4
8	Routing	95.5
9	Output	100

For the global setting, soil processes with 50% computing time is the most time consuming part, followed by routing with 25% and runoff concentration with 10%.

Rhine: 640 years in appr. 4.5h = 0.4min per year

	Process	sum % runtime
1	Read Meteo Data	79.4
2	Et pot	80.5
3	Snow	80.9
4	Soil	88.8
5	Groundwater	88.9
6	Runoff conc	89.6
7	Lakes	89.8
8	Routing	99.6
9	Output	100

For the Rhine basin reading input maps 79% is by far the most time consuming process, followed by routing (kinematic wave) 10% and the soil processes (8%)

7.2 Updates

Note:

Update history taken from github log

git log –pretty=format:"%ad - %an : %s" –date=short –graph > github.log

Most recent updates on top

```
* 2019-01-05 - CWatM : fix: corrected some warnings from PCCharm code inspector
* 2019-01-04 - CWatM : add: adding executable cwatm.exe
* 2019-01-04 - CWatM : Fix: new water demand changes did not use the same variable␣
↪name act_surfacewater in waterdemand and routing_kinematic. changed this in both␣
↪version 2.7 and 3.7 Add python: added a report command to report data as .map or .
↪tif for debugging
* 2018-12-20 - CWatM : Python3.7 New: replaced pcraster framework by own framework␣
↪Removed folder pcraster2 New: added save conditions for warmstart -> you can add a␣
↪10d or 6m or 2y after the first date -> the initial data will be saved every 10d␣
↪(or whatever number), or 6 month or 2 year
* 2018-12-17 - CWatM : New: Python 3 test code
*   2018-12-17 - CWatM : Merge branch 'develop' of https://github.com/iiasa/CWATM_
↪priv into develop
|\
| *   2018-12-12 - Community Water Model : Merge pull request #3 from iiasa/
↪waterdemand_update
| |\
| | *   2018-12-12 - Community Water Model : Merge branch 'develop' into waterdemand_
↪update
| | |\
| | |/
| |/|
| | * 2018-08-15 - Unknown : modify irrConsuption to act_irrConsumption in␣
↪landcoverType and soil modules
```

(continues on next page)

(continued from previous page)

```
| | * 2018-08-15 - Unknown : potential and actual values are explicitly written in␣
↪waterdemand module
| | * 2018-08-08 - Unknown : modified efficiency vaiables ;)
| | * 2018-08-07 - Unknown : modified read-netcdf for wateruse data
| | * 2018-07-24 - Yusuke : Added act_nonIrrConsumption conponents
| | * 2018-07-24 - Yusuke : Clean up before editing
* | | 2018-12-17 - CWatM : New: python 3 test version
* | | 2018-12-17 - CWatM : New: Added Python source code: Further test required, but␣
↪it seems to work. -> Plan in 2019 further development will use Python 3.7 coding␣
↪New: Building a executable .exe with Python 3  seems to work as well. Further␣
↪testing -> 2019 an installation setup will be produced using cx_freeze and Inno␣
↪setup to make an easy start on Windows (no Python background will be required for␣
↪CWATM users)
|/ /
* | 2018-12-12 - CWatM : Put Yusuke's version of waterdemand in (soil, landtypes,␣
↪waterdemand)
* | 2018-12-12 - CWatM : Fix: checkmap -c option now checks maps first (but can be␣
↪improved) new flag: usemeteodownscaling in [meteo] for using meteo downscaling Fix:␣
↪can now use rivernetwork as map or tif again (ldd.map) changes in initial and data_
↪handling
* | 2018-12-11 - CWatM : in sync with version on p drive
* | 2018-12-11 - CWatM : Small change in tutorial, added output variable added␣
↪calibration tutorial, to be extended
* | 2018-09-24 - CWatM : chk: waterdemand can use water demand netcdf with m/s or␣
↪million m3 per month/year
* | 2018-08-07 - CWatM : fix: reading meteo map with no leap year (365 day maps) new:␣
↪using a cover map to put addition values in
|/
* 2018-07-09 - CWatM : Fix: waterbalance for soil Chg: output of tss from 3-d␣
↪variable e.g actualET[1]
* 2018-06-27 - CWatM : fix: corrected storing initial values for the next warm start␣
↪chk: changed environmental flow (EF)settings file -      loading EF is now in water␣
↪demand
* 2018-06-07 - CWatM : chg: outcommented a library call in data_handling    #from␣
↪netcdftime import utime chg: added the sum of ET_actual again
* 2018-05-17 - CWatM : Changed waterbalance Changed waterbodies in large and small␣
↪lakes and reservoirs
* 2018-04-24 - CWatM : Fix: bugfix to read waterdemand map
* 2018-04-19 - CWatM : Change: meteo data can be clipped before and used. CWAT␣
↪detects if it is a global map or a regional one e.g using only meteo data set for␣
↪the Rhine.
* 2018-04-16 - CWatM : Change; in waterdemand, landcovertyp and soil cjhange variable␣
↪names     Gross = demand = withdrawal, netto = consumptiom      all vraibales names␣
↪now are ..demand or .. consumption
* 2018-04-13 - CWatM : test
* 2018-04-13 - CWatM : Change: netcdf output as monthly or annual map has now a␣
↪adequate monthly or yearly time step e.g. Months since 1901-01-01
* 2018-04-03 - CWatM : Change: CWATM can be used with a smaller meteo dataset e.g. to␣
↪use a demo dataset for the Rine with pr, tavg, ETRef, EWref
* 2018-04-03 - CWatM : Change: CWAT can be used with a smaller meteo dataset  e.g. to␣
↪download a smaller test meteo dataset for the Rhine
* 2018-04-03 - CWatM : Chg: running cwatm with a smaller meteo dataset in order to␣
↪make a test catchment (e.g. Rhine) with a small meteo dataset
* 2018-03-20 - CWatM : Added: - small lakes - calc environmental flow - 5 arcmin␣
↪version - downscale 30min meteo dataset to 5min
* 2017-11-20 - CWatM : fix: replace strftime with .year or .month etc fix: looks for >
↪ 1e20 and -1e20 in each map and change these to standard zero value (default =0)
```

(continues on next page)

(continued from previous page)

```
* 2017-10-30 - CWatM : Fix: bug fix to save maps with a SpinUp <> None
* 2017-10-27 - CWatM : Fix: reading meteo maps - every data > 1e12 is set to 0 Add:␣
↪maxtopwater in prg and settings.ini Fix: calibration routine
* 2017-09-21 - CWatM : bugfix: snow with more layers than 3
* 2017-09-20 - CWatM : chg: water demand, small lakes, land cover
* 2017-08-29 - CWatM : chg: water demand , soil add: error handling for output maps
* 2017-08-17 - CWatM : new: water demand is working chg: soil especially paddy and␣
↪non paddy irrigation bug: checked water balance
* 2017-07-13 - CWatM : fix: small bugfix, to run precipitation maps with the suffix .
↪nc4
* 2017-07-13 - CWatM : chg: soil part - using different maps -> map folder has to be␣
↪updated! chg: meteo maps do not have to be merge before -> stack of maps can  be␣
↪used add: inflow to a catchment (still to work on)
* 2017-05-23 - CWatM : chk: saving of netcdf with fixed number of time and with fixed␣
↪chunk size -> less diskspace used chk: a few more error handlings added
* 2017-05-19 - CWatM : chk: Chaznged soil calculation to Arno scheme and Mualem - van␣
↪Genuchten equation new: put in a lot of checks for the settingsfile e.g. check True␣
↪and false (not mispelled like ture). Check timing, check output variables chk: a␣
↪lot more error messages are given out if something is wrong chk: output netcdf time␣
↪is calculate in advanced in order to reduce size of output netcdf -> data_handling␣
↪line 789 sets it to this value
* 2017-05-10 - CWatM : chk: bugfix cropKC per land cover new: snow evaporation␣
↪included new: Calibration routine added
* 2017-04-20 - CWatM : fix: output to netcdf - in output and data_handling fix:␣
↪output as a time series without header with the option -h new: readme.md for github
* 2017-04-18 - CWatM : Transfer to new IIASA domain and making it private in branch␣
↪develop
* 2017-04-18 - CWatM : Transfer to new IIASA CWAT domain
* 2017-04-18 - CWatM : ready for transfer to iiasa
* 2017-04-13 - CWatM : data handling: faster read of meteo data
* 2017-04-06 - CWatM : soil - Copy (2).py- removed bug in calculation of soildepth -␣
↪change calc of arno beta
* 2017-04-06 - CWatM : Merge branch 'branch2' of https://github.com/CWatM/CWatM into␣
↪branch2
* 2017-04-06 - Community Water Model : Create LICENSE
* 2017-04-06 - CWatM : Updated soil, removed bug in calculating the soil depth␣
↪changed how arno beta is calculated
* 2017-02-03 - CWatM : - made CWATM run under cygwin   (for other linux version the␣
↪c++ code has to be compiled) - fixed reading maskmap from rectangle
* 2017-02-02 - CWatM : set realtive file path to c++ routine
* 2017-02-02 - CWatM : - New kinematic routing    - c++ routine include     TODO:␣
↪make it usable for linu/Unix - removed pcraster GIS commands - new output routine␣
↪for time series - Budyko output.html - corrected bug in snow modules - corrected␣
↪bug in init read/save module - WORKING on lakes/reservoirs TODO: bug in reading␣
↪maskmap from coordinates
* 2017-01-17 - CWatM : init condition - save more than 1 date
* 2017-01-16 - CWatM : Lake/reservoirs routing
* 2016-12-22 - CWatM : updated soil , initconditions etc
* 2016-12-16 - CWatM : runoff concentration
* 2016-12-08 - CWatM : With sphinx documentation making files
* 2016-12-07 - CWatM : Update
* 2016-12-07 - CWatM : Preferential flow, frost
* 2016-11-10 - CWatM : Cacluation Evaporation from climate data
* 2016-10-21 - CWatM : Changed soil + test
* 2016-10-18 - CWatM : Waterdemand included
* 2016-10-03 - CWatM : last August update - waterbalance
* 2016-08-26 - CWatM : water balance 7
```

(continues on next page)

(continued from previous page)

```
* 2016-08-26 - CWatM : water balance 6
* 2016-08-25 - CWatM : water Balance 5
* 2016-08-24 - CWatM : water balance 4 Checks ok : soil , groundwater, routing,␣
↪waterdemand Missing: reservoirs, sum up to catchments
* 2016-08-23 - CWatM : water balance 3
* 2016-08-23 - CWatM : water balance 2
* 2016-08-22 - CWatM : Water balance check 1 Output on screen
* 2016-08-19 - CWatM : initial condition
* 2016-08-17 - CWatM : Spin up
* 2016-08-17 - CWatM : output netcdf add attributes
* 2016-08-10 - CWatM : output + time
* 2016-08-10 - CWatM : date and time
* 2016-08-09 - CWatM : output 3
* 2016-08-09 - CWatM : output 2
* 2016-08-08 - CWatM : output timeseries
* 2016-08-03 - CWatM : waterbodies 1 Checked routing - working :)
* 2016-08-02 - CWatM : routing 3
* 2016-08-01 - CWatM : routing 2
* 2016-08-01 - CWatM : routing 1
* 2016-07-29 - CWatM : some changes I do not know anymore
* 2016-07-26 - CWatM : soil + groundwater
* 2016-07-26 - CWatM : soil check3
* 2016-07-25 - CWatM : soil check2
* 2016-07-25 - CWatM : check soil module
* 2016-07-24 - CWatM : soil update
* 2016-07-24 - Burek : Soil and groundwater
* 2016-07-22 - CWatM : soil
* 2016-07-21 - CWatM : till waterdemand - soil
* 2016-07-20 - CWatM : Next step interception
* 2016-07-19 - CWatM : changing irrigationarea part
* 2016-07-15 - CWatM : Initial procedure for soil, groundwater, waterdemand
* 2016-07-13 - CWatM : include: snow frost
```

7.3 TODO

7.3.1 Structural improvements

Note: This has to be done. Importance: 1 to be changed first .. 3 to be changed later

Topic	TODO	Description	Importance	DONE
Documentation	Documentation	start writing a user manual	1	.
Documentation	Source code documentation	Improve comment-lines in the code and include them in the autodocu sphinx	1	.
Documentation	Include log file/change log	document the changes in the code/settings	2	.
Output	GAMS output	output/input in GAMS (gdx - files)	2	.
Output	Extent output possibilities	Output as e.g. yearly areatotal, catchment total as maps, as time series	1	.
Handling	Improve error handling	more messages for users if something goes wrong	1	.
Handling	Checks maps	include a pre-run, where input data are checked for plausibility	2	.
Handling	Load multiple netcd files	read meteo input netcdf from split files	2	.

7.3.2 Model improvements

TODO	Description	Importance	DONE
Frost	include frost routine (no soil movement during strong frost)	1	X
Snow	include more than 3 vertical layers (make it flexible)	2	X
Runoff concentration	include a 1st routing to the edge of a grid cell	1	X
Include water & sealed land cover	include 2 more land cover types (water covered area, sealed area)	1	X
Preferential flow	include preferential flow to soil layers	1	X
Calculate Evaporation on PM	include Penman Monteith ET routine	1	X
Reduce reading of time series maps	e.g. interception maps only 1 per month	2	X
Kinematic wave	Add C++ kinematic wave procedure	2	X
soil depend on land cover	include hydropedo transfer function landcover -> soil	2	.
Improve lakes& reservoirs	Add another way of including lakes/reservoirs	2	X
Inflow points	add points where water can be added/substracted	1	X
Include Environmental flow	use environmental flow concept on the fly not only post-processing	2	X
Water allocation	include water demand <-> water supply functionality	2-3	.
Include EPIC approach	to be in line with ESM include the EPIC approach	3	.

CHAPTER 8

Data

Contents

8.1 Data requirements

8.2 Data format

In general data format is netCDF (version3 or version4)

For the mask map (to define the area of calculation) or the stations (to define the time series outputs) in can be either netCDF, Geotiff or PCRaster maps

8.3 Data storage structure

```
project
│-  README.txt
│
└--areamaps
│   └- maskmap, stationmap
│
└--landcover
│   └---forest
│   │    │-  CropCoefficientForest_10days
│   │    │-  interceptcapForest10days
│   │    │-  maxRootdepth, minSoilDepthFrac
│   │    └-  rootFraction1, rootFraction2
│   │
│   └---grassland (same var as forest)
│   │
│   └---irrNonPaddy (same var as forest)
│   │
│   └---irrPaddy (same var as forest)
│
└───landsurface
│   │- fractionlandcover, global_clone
│   │
│   └---albedo
│   │   └- albedo
│   │
│   └---topo
│   │   └- dz_Rel_hydro1k, elvstd , tanslope
│   │
│   └---waterDemand
│       └- domesticWaterDemand, industryWaterDemand, irrigationArea, efficiency
│
└───soil
│   └- alpha, forest_alpha, lamdba, forest_lambda, ksat, forest_ksat, thetas, forest_
↪thetas, thetar, forest_thetar
│   └-cropgrp
│
└---groundwater
│   └- kSatAquifer, recessionCoeff, specificYield
│
└---routing
    │- ldd, catchment, cellarea
    │
    └---kinematic
```

(continues on next page)

(continued from previous page)

```
    │     └─ chanbnkf, chanbw, changrad, chanleng, chanman
    │
    └---lakereservoirs
          └─ lakeResArea, lakeResDis,lakeResID, lakeResType, lakeResVolRes, lakeResYear,
          └─ smallLakesRes, smalllakesresArea, smalllakesresDis, smallwatershedarea
```

8.4 Static data

8.4.1 Mask map

- mask map or coordinates to model only regions or catchments
- maps or coordinates for station to print time series

8.4.2 Landsurface

Albedo

Global Albedo dataset from Muller et al., (2012) http://www.globalbedo.org

Digital elevation model and river channel network

The model uses a digital elevation model and its derivate (e.g. standards deviation, slope) as variables for the snow processes and for the routing of surface runoff. The Shuttle Radar Topography Mission - SRTM (Jarvis et al., 2008) is used for latitudes <= 60 deg North and DEM Hydro1k (US Geological Survey Center for Earth Resources Observation and Science)is used for latitudes > 60 deg North CWATM uses a local drainage direction map which defines the dominant flow direction in one of the eight neighboring grid cells (D8 flow model). This forms a river network from the springs to the mouth of a basin. To be compliant with the ISIMIP framework the 0.5 deg drainage direction map (DDM30) of (Döll and Lehner, 2002) is used. For higher resolution e.g. 5' different sources of river network maps are available e.g. HydroSheds (Lehner et al., 2008) – DRT (Wu et al., 2011) and CaMa-Flood (Yamazaki et al., 2009). These approaches uses the same hydrological sound digital elevation model but differ in the upscaling methods. Fang et al. (2017) shows the importance of routing schemes and river networks in peak discharge simulation.

8.4.3 Soil and soil hydraulic properties

Soil textural data were derived from the ISRIC SoilGrids1km database http://www.isric.org/explore/soilgrids (Hengl et al. 2014). Pedotransfer functions applied on 1km soil texture data - originating from the HYPRES database (Wösten et al. 1999) were used to obtain the Mualem - VanGenuchten soil hydraulic parameters for soil water transport modeling in the soil module.

8.4.4 Water demand

8.4.5 Groundwater

GLHYMPS—Global Hydrogeology Maps of permeability and porosity http://crustalpermeability.weebly.com/data-sources.html (Gleeson et al., 2014)

Lakes and Reservoirs

The HydroLakes database http://www.hydrosheds.org/page/hydrolakes (Bernhard Lehner et al., 2011; Messager, Lehner, Grill, Nedeva, & Schmitt, 2016) provides 1.4 million global lakes and reservoirs with a surface area of at least 10ha. CWATM differentiate between big lakes and reservoirs which are connected inside the river network and smaller lakes and reservoirs which are part of a single grid cell and part of the runoff concentration within a grid cell. Therefore the HydroLakes database is separated into "big" lakes and reservoirs with an area 100 km2 or a upstream area 5000 km2 and "small" lakes which represents the non-big lakes. All lakes and reservoirs are combined at grid cell level but big lakes can have the expansion of several grid cells. Lakes bigger than 10000 km2 are shifted according to the ISIMIP protocol.

8.5 Temporal data for each year

8.5.1 Crop coefficient

Based on: MIRCA2000—Global data set of monthly irrigated and rainfed crop areas around the year 2000. http://www.uni-frankfurt.de/45218023/MIRCA (Portmann et al., 2010)

8.5.2 Land cover

Land cover is used to calculate fraction of water, forest, irrigated area, rice irrigated area, sealed (impermeable area) and the remaining fraction for each cell. For each fraction the soil module runs separately. The total runoff of each cell is calculated by weighting the cell according to the different fractions.

Source: https://lta.cr.usgs.gov/GLCC (US Geological Survey Center for Earth Resources Observation and Science)

8.6 Continous temporal data

8.6.1 Meteorological data

- max, min, avg temperature [K]
- humidity (relative[%] or specific[%])
- surface pressure [Pa]
- radiation (short wave and long wave downwards) [W m-2]
- windspeed [m/s]

If potential evaporation is already calculated in a prerun or from external source

- Precipitation [Kg m-2 s-1] or [m] or [mm] (can be adjusted by a conversion factor in the settings file)
- Temperature (avg) [K]
- Potential evaporation [Kg m-2 s-1] or [m] or [mm] (can be adjusted by a conversion factor in the settings file)

From observation: (see ISI-MIP 2a)

- WFDEI.GPCC (Weedon et al. 2014) WFD—Watch forcing data set: 0.5 3/6 hourly meteorological forcing from ECMRWF reanalysis (ERA40) bias-corrected and extrapolated by CRU TS and GPCP (rainfall) and corrections for under catch
- PGMFD v.2 - Princeton (Sheffield et al. 2006),

- GSWP3 (Kim et al.)
- MSWEP (Beck et al. 2017) .

From Global Circulation models GCMs (see ISI-Mip 2b)

- HadGem2-ES (Met Office Hadley Centre, UK)
- IPSL-CM5A-LR (Institut Pierre-Simon Laplace, France)
- GFDL-ESM2M (NOAA, USA)
- MIROC-ESM-CHEM (JAMSTEC, AORI, University of Tokyo, NIES, Japan)
- NorESM1-M (Norwegian Climate Centre, Norway)

8.7 References

- Beck, H. E., A. I. J. M. Van Dijk, V. Levizzani, J. Schellekens, D. G. Miralles, B. Martens and A. De Roo (2017). "MSWEP: 3-hourly 0.25° global gridded precipitation (1979-2015) by merging gauge, satellite, and reanalysis data." Hydrology and Earth System Sciences 21(1): 589-615.
- Döll, P. and B. Lehner (2002). "Validation of a new global 30-min drainage direction map." Journal of Hydrology 258(1): 214-231.
- Döll, P. and S. Siebert (2002). "Global modeling of irrigation water requirements." Water Resources Research 38(4): 81-811.
- Gleeson, T., N. Moosdorf, J. Hartmann and L. P. H. Van Beek (2014). "A glimpse beneath earth's surface: GLobal HYdrogeology MaPS (GLHYMPS) of permeability and porosity." Geophysical Research Letters 41(11): 3891-3898.
- Hengl, T., J. M. de Jesus, R. A. MacMillan, N. H. Batjes, G. B. M. Heuvelink, E. Ribeiro, A. Samuel-Rosa, B. Kempen, J. G. B. Leenaars, M. G. Walsh and M. R. Gonzalez (2014). "SoilGrids1km — Global Soil Information Based on Automated Mapping." PLOS ONE 9(8): e105992.
- Jarvis, A., H. I. Reuter, A. Nelson and E. Guevara (2008). Hole-filled SRTM for the globe Version 4, available from the CGIAR-CSI SRTM 90m Database (http://srtm.csi.cgiar.org).
- Kim, H., S. Watanabe, E.-C. Chang, K. Yoshimura, Y. Hirabayashi, J. Famiglietti and T. Oki "Century long observation constrained global dynamic downscaling and hydrologic implication [in preparation]."
- Lehner, B., C. R. Liermann, C. Revenga, C. Vörösmarty, B. Fekete, P. Crouzet, P. Döll, M. Endejan, K. Frenken, J. Magome, C. Nilsson, J. C. Robertson, R. Rödel, N. Sindorf and D. Wisser (2011). "High-resolution mapping of the world's reservoirs and dams for sustainable river-flow management." Frontiers in Ecology and the Environment 9(9): 494-502.
- Lehner, B., K. Verdin and A. Jarvis (2008). "New global hydrography derived from spaceborne elevation data." Eos 89(10): 93-94.
- Messager, M. L., B. Lehner, G. Grill, I. Nedeva and O. Schmitt (2016). "Estimating the volume and age of water stored in global lakes using a geo-statistical approach." 7: 13603.
- Muller, P. J., P. Lewis, J. Fischer, P. North and U. Framer (2012). The ESA GlobAlbedo Project for mapping the Earth's land surface albedo for 15 Years from European Sensors., paper presented at IEEE Geoscience and Remote Sensing Symposium (IGARSS) IEEE Geoscience and Remote Sensing Symposium (IGARSS) 2012. Munich, Germany.
- Portmann, F. T., S. Siebert and P. Döll (2010). "MIRCA2000—Global monthly irrigated and rainfed crop areas around the year 2000: A new high-resolution data set for agricultural and hydrological modeling." Global Biogeochemical Cycles 24(1): n/a-n/a.

- Sheffield, J., G. Goteti and E. F. Wood (2006). "Development of a 50-year high-resolution global dataset of meteorological forcings for land surface modeling." Journal of Climate 19(13): 3088-3111.
- Siebert, S., P. Döll, J. Hoogeveen, J. M. Faures, K. Frenken and S. Feick (2005). "Development and validation of the global map of irrigation areas." Hydrology and Earth System Sciences 9(5): 535-547.
- US Geological Survey Center for Earth Resources Observation and Science Hydro1k. U. E. Land Processes Distributed Active Archive Center (LP DAAC), Sioux Falls, SD.
- Weedon, G. P., G. Balsamo, N. Bellouin, S. Gomes, M. J. Best and P. Viterbo (2014). "The WFDEI meteorological forcing data set: WATCH Forcing data methodology applied to ERA-Interim reanalysis data." Water Resources Research 50(9): 7505-7514.
- Wösten, J. H. M., A. Lilly, A. Nemes and C. Le Bas (1999). "Development and use of a database of hydraulic properties of European soils." Geoderma 90(3-4): 169-185.
- Wu, H., J. S. Kimball, N. Mantua and J. Stanford (2011). "Automated upscaling of river networks for macroscale hydrological modeling." Water Resources Research 47(3).
- Yamazaki, D., T. Oki and S. Kanae (2009). "Deriving a global river network map and its sub-grid topographic characteristics from a fine-resolution flow direction map." Hydrology and Earth System Sciences 13(11): 2241-2251.
- Zhao, F., Veldkamp, T. I. E., Frieler, K., Schewe, J., Ostberg, S., Willner, S., Schauberger, B., Gosling, S., N. , Müller Schmied, H., Portmann, F., T. , Leng, G., Huang, M., Liu, X., Tang, Q., Hanasaki, N., Biemans, H., Gerten, D., Satoh, Y., Pokhrel, Y., Stacke, T., Ciais, P., Chang, J., Ducharne, A., Guimberteau, M., Wada, Y., Kim, H., & Yamazaki, D. (2017). The critical role of the routing scheme in simulating peak river discharge in global hydrological models. Environmental Research Letters, 12(7), 075003

CHAPTER 9

Calibration tool

Calibration tool for hydrological models
in ../CWATM/calibration

using a distributed evolutionary algorithms in python: DEAP library
http://deap.readthedocs.io/en/master/
https://github.com/DEAP/deap/blob/master/README.md

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau and Christian Gagné, "DEAP: Evolutionary Algorithms Made Easy", Journal of Machine Learning Research, vol. 13, pp. 2171-2175

The calibration tool was created by Hylke Beck 2014 (JRC, Princeton) hylkeb@princeton.edu
Thanks Hylke for making it available for use and modification
Modified by Peter Burek

The submodule Hydrostats was created 2011 by:
Sat Kumar Tomer (modified by Hylke Beck)
Please see his book Python in Hydrology

9.1 Calibration method

Calibration is using an evolutionary computation framework in Python called DEAP (Fortin et al., 2012). We used the implemented evolutionary algorithm NSGA-II (Deb et al., 2002) for single objective optimization. As objective function we used the modified version of the Kling-Gupta Efficiency (Kling et al., 2012), 2012), with r as the correlation coefficient between simulated and observed discharge (dimensionless), β as the bias ratio (dimensionless) and γ as the variability ratio.

$KGE' = 1 - \sqrt{(r-1)^2) + (\beta-1)^2 + (\gamma-1)^2}$
where: $\beta = \frac{\mu_s}{\mu_o}$ and $\gamma = \frac{CV_s}{CV_o} = \frac{\sigma_s/\mu_s}{\sigma_o/\mu_o}$

Where CV is the coefficient of variation, μ is the mean streamflow [m3 s1] and σ is the standard deviation of the streamflow [m3 s1]. KGE', r, β and γ have their optimum at unity. The KGE' measures the Euclidean distance from the ideal point (unity) of the Pareto front and is therefore able to provide an optimal solution which is simultaneously good for bias, flow variability, and correlation. For a discussion of the KGE objective function and its advantages over the often used Nash–Sutcliffe Efficiency (NSE) or the related mean squared error see (Gupta et al., 2009). The calibration uses general a population size (µ) of 256, a recombination pool size (λ) of 32.The number of generations was set to 30, which we found was sufficient to achieve convergence for stations

9.1.1 Further ideas for calibration

- Regionalization see (Samaniego et al. 2017) and (Beck et al. 2016)
- Using Budyko see (Greve et al. 2016)

9.2 Calibration parameters

Snow
1. Snowmelt coefficient in [m/C deg/day] as a degree-day factor
Evapotranspiration
2. Crop factor as an adjustment to crop evapotranspiration
Soil
3. Soil depth factor: a factor for the overall soil depth of soil layer 1 and 2
4. Preferential bypass flow: empirical shape parameter of the preferential flow relation
5. Infiltration capacity parameter: empirical shape parameter b of the ARNO model
Groundwater
6. Interflow factor: factor to adjust the amount which percolates from interflow to groundwater
7. Recession coefficient factor: factor to adjust the base flow recession constant (the contribution from groundwater to baseflow)
Routing
8. Runoff concentration factor: a factor for the concentration time of run-off in each grid-cell
9. Channel Manning's n factor: a factor roughness in channel routing
10. Channel, lake and river evaporation factor: factor to adjust open water evaporation
Reservoir & lakes
11. Normal storage limit: the fraction of storage capacity used as normal storage limit
12. Lake A factor : factor to channel width and weir coefficient as a part of the Poleni weir equation

9.3 Calibration tool structure

```
calibration
|-  readme.txt
|-  readme.txt
|
```

(continues on next page)

(continued from previous page)

```
└--observed_data
|    └─ lobith2006.cvs, ...
|
└--templates
|    └-- runpy.bat, runpy.sh
|    └-- settings.ini
```

9.4 How it works

The calibration tool builds up a single-objective obtimization framework using the Python libray DEAP For each run it triggers the run of the hydrological model:

- using a template of the settings file
- replacing the output folder in this template file
- replace placeholders with the values of calibration parameters, the limit of the parameter range is given in the file: ParamRanges.csv

After each run the model run is compared to observed values (e.g. observed_data/lobith2006.csv)

After the calibration, statistics and the best run is printed output

9.5 What is needed

1. The template files in ../templates have to be adjusted

- runpy.bat: the path to cwatm.py have to be set correctly (for linux a .sh file has to be created)
- The actual version of a cwatm settings file has to modified:
- replacing the output folder with the placeholder: %run_rand_id

```
#-------------------------------------------------------
# CALIBARTION PARAMETERS
#-------------------------------------------------------
[CALIBRATION]

# These are parameter which are used for calibration
# could be any parameter, but for an easier overview, tehey are collected here
# in the calibration template a placeholder (e.g. %arnoBeta) instead of value

OUT_Dir = %run_rand_id
```

- putting the output variables in e.g. OUT_TSS_Daily = discharge or monthly average discharge OUT_TSS_MonthAvg = discharge

```
OUT_TSS_Daily = discharge
OUT_TSS_MonthAvg = discharge
```

- delete all the output variables in the template (mostly at the end of the file)
- replacing calibration parameter values with a placeholder: e.g. %SnowMelt

```
# Snow  SnowMeltCoef = 0.004
SnowMeltCoef = %SnowMelt
# Cropf factor correction
crop_correct =  %crop
#Soil
soildepth_factor = %soildepthF
#Soil preferentialFlowConstant = 4.0, arnoBeta_factor = 1.0
preferentialFlowConstant = %pref
arnoBeta_add = %arnoB
# interflow part of recharge factor = 1.0
factor_interflow = %interF
# groundwater recessionCoeff_factor = 1.0
recessionCoeff_factor = %reces
# runoff concentration factor runoffConc_factor = 1.0
runoffConc_factor = %runoff
#Routing manningsN factor [0.1 - 10.0] default  1.0
manningsN = %CCM
# reservoir  normal storage limit (fraction of total storage, [-]) [0.15 - 0.85] default 0.5
normalStorageLimit = %normalStorageLimit
# lake parameter - factor to alpha: parameter of of channel width and weir coefficient  [0.33 - 3.] dafault 1.
lakeAFactor = %lakeAFactor
# lake wind factor - factor to evaporation from lake [0.8 - 2.] dafault 1.
lakeEvaFactor = %lakeEvaFactor
```

2. the range of parameter space has to be defined in ParamRanges.csv

```
ParameterName,MinValue,MaxValue
SnowMelt,0.001,0.007
crop,0.8,3.0
soildepthF, 0.8,1.8
pref,0.5,8
arnoB,0.01,1.0
interF, 0.33,3.0
reces,0.1,10
runoff,0.1,5
CCM,0.1,10.0
normalStorageLimit,0.15,0.85
lakeAFactor,0.333,3.0
lakeEvaFactor,0.5,3.0
No,1,100
```

3. The observed discharge has to be provided in an .cvs file e.g. observed_data/lobith2006.csv

In the template settings the date has to be set, so that the period of observed discharge is between SpinUp and StepEnd

```
#-------------------------------------------------------
[TIME-RELATED_CONSTANTS]
```

(continues on next page)

(continued from previous page)

```
#-----------------------------------------------------------

# StepStart has to be a date e.g. 01/06/1990
# SpinUp or StepEnd either date or numbers
# SpinUp: from this date output is generated (up to this day: warm up)

StepStart = 1/1/1990
SpinUp = 1/1/1995
StepEnd =  31/12/2010
```

4. And empty ../catchments directory needs to be created

5. A few option in the settings.txt have to be adjusted (how many runs?, a first run with standard parameters? etc)

```
[DEFAULT]
Root = /c/watmodel/CWATM
RootPC = C:/watmodel/CWATM
Rootbasin = calibration_rhine

ForcingStart = 1/1/2000
ForcingEnd = 31/12/2010
timeperiod = daily

[ObservedData]
Qtss = observed_data/lobith.csv
Column = lobith
Header = River: Rhine  station: Lobith

[Validate]
Qtss = observed_data/lobith_val.csv
ValStart = 1/1/1990
ValEnd = 31/12/1999


[Path]
Templates = templates
SubCatchmentPath = catchments
ParamRanges = ParamRanges.csv

[Templates]
ModelSettings = settings.ini
RunModel = runpy.sh

[Option]
firstrun = False
para_first = [0.0022, 1.72, 1.24, 7.07, 0.55, 1.92, 2.81, 0.74,1.34,0.35,2.04,1.0, 1.]
# Snowmelt, crop KC, soil depth,pref. flow, arno beta, interflow factor, groundwater␣
↪recession,
# runoff conc., routing, manning factor, normalStorageLimit, lakeAFactor,
↪lakeEvaFactor,No of run
bestrun = True
```

(continues on next page)

(continued from previous page)

```
[DEAP]
maximize = True
use_multiprocessing = 1
ngen = 30
mu = 256
lambda_ = 32
```

6. run python calibration_single.py settings.txt

9.6 Recommondations

1. Run the model first to store the pot. evaporation results

Afterwards use the stored evaporation to run the calibration
calc_evaporation = False

2. Run the model and store the last day to be used as initial condition for the calibration runs

Best is to use a long term run for this.

```
[INITITIAL CONDITIONS]
#-------------------------------------------------------

# for a warm start initial variables a loaded
# e.g for a start on 01/01/2010 load variable from 31/12/2009
load_initial = False
initLoad = $(FILE_PATHS:PathRoot)/init/Rhine_19891231.nc

# saving variables from this run, to initiate a warm start next run
# StepInit = saving date, can be more than one: 10/01/1973 20/01/1973
save_initial = False
initSave = $(FILE_PATHS:PathRoot)/init/Rhine
StepInit = 31/12/1989 31/12/2010
```

load_initial = False
save_initial = True

During calibration use:
load_initial = True
save_initial = False

3. Use a long SpinUp time (> 5 years to give groundwater enough time)

9.7 References

- Beck, H. E., A. I. J. M. van Dijk, A. de Roo, D. G. Miralles, T. R. McVicar, J. Schellekens and L. A. Bruijnzeel (2016). “Global-scale regionalization of hydrologic model parameters.” Water Resources Research 52(5): 3599-3622.
- Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). “A fast and elitist multiobjective genetic algorithm: NSGA-II.” IEEE Transactions on Evolutionary Computation 6(2): 182-197.
- Fortin, F. A., F. M. De Rainville, M. A. Gardner, M. Parizeau and C. Gagńe (2012). “DEAP: Evolutionary algorithms made easy.” Journal of Machine Learning Research 13: 2171-2175.
- Greve, P., L. Gudmundsson, B. Orlowsky and S. I. Seneviratne (2016). “A two-parameter Budyko function to represent conditions under which evapotranspiration exceeds precipitation.” Hydrology and Earth System Sciences 20(6): 2195-2205.
- Gupta, H. V., H. Kling, K. K. Yilmaz and G. F. Martinez (2009). “Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling.” Journal of Hydrology 377(1-2): 80-91.
- Kling, H., M. Fuchs and M. Paulin (2012). “Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios.” Journal of Hydrology 424-425: 264-277.
- Samaniego, L., R. Kumar, S. Thober, O. Rakovec, M. Zink, N. Wanders, S. Eisner, H. Müller Schmied, E. Sutanudjaja, K. Warrach-Sagi and S. Attinger (2017). “Toward seamless hydrologic predictions across spatial scales.” Hydrology and Earth System Sciences 21(9): 4323-4346.

CHAPTER 10

Calibration tutorial

10.1 What you need

Python 2.7.x 64 bit and a running CWATM (libraries netCDF4, numpy, scipy, GDAL) In addition: **library deap**

Calibration is using a distributed evolutionary algorithms in python: DEAP library

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau and Christian Gagné, "DEAP: Evolutionary Algorithms Made Easy", Journal of Machine Learning Research, vol. 13, pp. 2171-2175

Note: Deap can also be used in Python 3.7 but at the moment we did not translate the scripts to Python3.7. it is on the TODO list

You can install it with: Pip install deap (you might change into the folder ../python/Scripts/)

- Make sure that python 2.7.x is working
- Make sure that CWATM is running in non calibration mode
- For some of the following steps it is easier to have PCRaster installed: http://pcraster.geo.uu.nl/

10.2 Running calibration

1. Look into the settings file of the calibration folder.
2. look into runCalibration.bat. If python is in your computer path everything should be ok, otherwise put in the path to python
3. look into templates/runpy.bat. Put the path to python in if necessary
4. look into templates/settings.ini. Put the pathes in a right way that it fits to your computer:

```
[FILE_PATHS]
#-------------------------------------------------------
PathRoot = P:/watmodel/CWATM/calibration_tutorial
PathOut = $(PathRoot)/output
PathMaps = $(PathRoot)/CWATM_data/cwatm_input
PathMeteo = $(PathRoot)/climate
```

5. in observed_data/yukon2001.cvs you find the observed data:

```
- make sure the name in the header is the same as in [ObservedData] Column
- make sure that there are enough data in (from ForcingStart to ForcingEnd)
```

6. make sure the folder catchments is empty! Before each try this folder has to be empty

10.3 Run runCalibration.bat

1. go for testing (see below)
2. go for testing again (see below)
3. Change use_multiprocessing =1 in settings.txt
4. Run runCalibration.bat and after some time something should appear on your window

10.4 For testing

- Change use_multiprocessing = 0 in settings.txt
- Delete catchments but keep the empty folder
- Run runCalibration.bat and wait till catchment/00_001 gets filled, then interrupt

1. Change to catchments/00_001
2. Run runpy00_001.bat
3. See what errors come up and change settings-Run00_001.ini
4. Change template/setting.ini in the same way
5. Do this again and again till no error

10.5 Running it on your computer

It will be really slow on Windows using data on the the server – next step run it on your PC

- copy the whole folder P:watmodelCWATMcalibration_tutorial to your PC (only 15 GB)
- (but maybe you have already parts of it on your computer – like the big climate input files)
- Make it work on your computer:

```
Changing file paths in templates/settings.ini, setting.txt
Changing the path for python in runCalibration.bat and templates/runpy.bat
```

10.6 Preparation for another catchment

10.6.1 Preparing the observed dataset – discharge

Calibration works by comparing simulated discharge with observed discharge using an objective function: Here we use the Kling-Gupta Efficiency but we can also use Nash-Sutcliffe Efficiency . Please find some more information on the objective function an on the evolutionary computation framework used for calibration on: https://cwatm.github.io/calibration.html

- The observed values can be stored as daily values or monthly values
- The observed values should be at least cover 5 years (best is 10-15 years)
- The observed discharge has to be stored as textfile in:

```
./observed_data/nameofstation.cvs
And has to look like this:
date,yukon_pilot_station
2001-04-01,1302.6
2001-04-02,1302.6
2001-04-03,1302.6
2001-04-04,1302.6
...
...
2013-12-31,2647.6
```

- Or:

```
date ,zhutuo
2002-01-01,3229.0
2002-02-01,2979.2
2002-03-01,3229.0
```

Format:

- Date format like this year-month-day [yyyy-mm-dd]
- Separated by a comma
- Discharge in [m3/s]
- If a value is missing that is not a problem (as long as the time series is long enough):

```
it should like this: (no value after the comma)
2002-01-12,
```

- For each day (or month) a line

Settings.txt

In the settings file the lines:

```
[ObservedData]
Qtss = observed_data/zhutuo_2002month.csv
Column = zhutuo
Header = River: Yangtze  station: Zhutuo
```

Should correspondent to the name and header in the observed discharge.cvs

The lines:

```
ForcingStart = 1/1/2002
ForcingEnd = 31/12/2013
```

Should correspondent to the amount of lines in the observed discharge.cvs

10.7 Creating an initial netcdf file for warm start

It is best to have a long warm up phase especially for groundwater: See also: https://cwatm.github.io/setup.html#initialisation

You can run CWATM for a couple of years (20 years or more) and store the last days storage values in a file. This file can be read in to enable a 'warm" start

- change use_multiprocessing = 0 in settings.txt
- Delete catchments but keep the empty folder
- Run runCalibration.bat and wait till catchment/00_001 gets filled, then interrupt
- Change to catchments/00_001

Open the settings-Run_001.init

- Change load_initial = True to load_initial = False
- save_initial = True
- initSave = $(FILE_PATHS:PathRoot)/CWATM_init/testx
- StepInit = 31/01/1996 (change it to a date 1 month after your StepStart)
- Run runpy00_001.bat

There should be a file ./CWATM_init/testx_19960131.nc

- Change to: load_initial = True
- initLoad = $(FILE_PATHS:PathRoot)/CWATM_init/testx_19960131.nc
- Run runpy00_001.bat

If it work then it used the initial file you generate before (that was just a test)

Now change to:

- StepStart = 1/1/1961
- StepEnd = 31/12/2013
- load_initial = False
- save_initial = True
- initSave = $(FILE_PATHS:PathRoot)/CWATM_init/station_name
- StepInit = 31/12/2013
- Run runpy00_001.bat

This should have generated a file ./CWATM_init/station_name_20131231.nc

And again:

- StepStart = 1/1/1961 (some 20 years or longer)
- StepEnd = 31/12/1995 (a day before your normal running day)

- load_initial = True
- initLoad = $(FILE_PATHS:PathRoot)/CWATM_init/station_name_20131231.nc
- save_initial = True
- initSave = $(FILE_PATHS:PathRoot)/CWATM_init/station_name
- StepInit = 31/12/1995 (a day before your running day)
- Run runpy00_001.bat

This should have generated a file ./CWATM_init/station_name_19951231.nc

And last part:

- Change StepStart and StepEnd back to original values
- load_initial = True
- initLoad = $(FILE_PATHS:PathRoot)/CWATM_init/station_name_19951231.nc
- save_initial = False
- Run runpy00_001.bat

If it works, do the same in the ./template/settings.ini

Note: You have now a "warm" start for every calibration run

10.8 Cutting out a catchment as mask map

See the .doc file in P:watmodelCWATMcalibration_tutorialcalibrationtoolscut_catchmentFor a description:

Requirements: PCRASTER:

We do no need the python version, I think downloading, extracting and setting of the paths in P:watmodelCWATMcalibration_tutorialcalibrationtoolscut_catchmentcatchconfig_win.ini Creating the 2 potential evaporation files in advance

Potential evaporation is Calculated with Penman-Monteith in CWATM, but it is not part of the calibration = there is no change in pot. Evaporation. In order to make the calibration computational faster the results of pot evaporation could be stored and used every time.

For the 30min this is done already as global map set, but for the 5min these files become too big. So they have to be produced for each basin separately

Same preparation as for **Creating an initial netcdf file for warm start** see above There should be a folder catchments00_001 with a working run for 001.

Open the settings-Run_001.init

Change:

```
[Option] calc_evaporation = True
[TIME-RELATED_CONSTANTS] SpinUp = None
[EVAPORATION]
OUT_Dir = $(FILE_PATHS:PathOut)
OUT_MAP_Daily = ETRef, EWRef
```

Run runpy00_001.bat There should be a file ETRef.nc and EWRef in the output directory

Rename the files e.g. ETRef.nc to ETRef_yangtze.nc, EWRef.nc to EWRef_yangtze.nc and copy it to PathMeteo (or somewhere else, you have to put the path in)

Open the settings-Run_001.init

Change:

```
[Option] calc_evaporation = False
[TIME-RELATED_CONSTANTS] SpinUp = -> to the time it was before
[Meteo]
daily reference evaporation (free water)
E0Maps = $(FILE_PATHS:PathMeteo)/EWRef_yangtze
daily reference evapotranspiration (crop)
ETMaps = $(FILE_PATHS:PathMeteo)/ETRef_yangtze
[EVAPORATION]
OUT_Dir = $(FILE_PATHS:PathOut)       !!! outcomment this again - important
OUT_MAP_Daily = ETRef, EWRef
```

Test it: Run runpy00_001.bat

And change the settings.ini in templates in the same way

10.9 Calibration of a downstream catchment

Calibration of a downstream catchment (upstream catchment is already calibrated) can be done using:

- The catchment area of the downstream catchment minus the upstream catchment
- The missing discharge from the upstream catchment is replaced by an inflow file

1. Cut the mask map, so that the upstream catchment is NOT in the mask map anymore
2. Detect the point(s) downstream of the inflow points
3. Run the best calibration scenario(s) of the upstream catchments again to produce long timeserie(s) of the outlet(s) point
4. Create an inflow file from the long timeseries of outlet(s)
5. Create a downstream calibration settings (directories, templates etc.)

Test the catchment!

6. Change the settings file of the downstream calibration so that it includes the inflow from upstream

Test it! 7. Create initial file for warm start

10.9.1 Cutting the mask map

Assuming you have a mask map of the whole catchment (e.g. Yangtze.map and the station points (here Zhutuo 105.75 28.75 and Yichang 111.25 30.75 1. Creating catchment for Zhutuo: catchment 105.75 28.75 ldd_yangtze.map zhu1.map 2. Creating catchment for Yichang: catchment 111.25 30.75 ldd_yangtze.map yi1.map 3. Creating Yichang without Zhutuo:

```
pcrcalc a2.map = cover(scalar(zhu1.map)*2,scalar(yi1.map))
pcrcalc yichang.map = boolean(if(a2.map eq 1,a2.map))
```

Result is a maskmap: Yichang.map

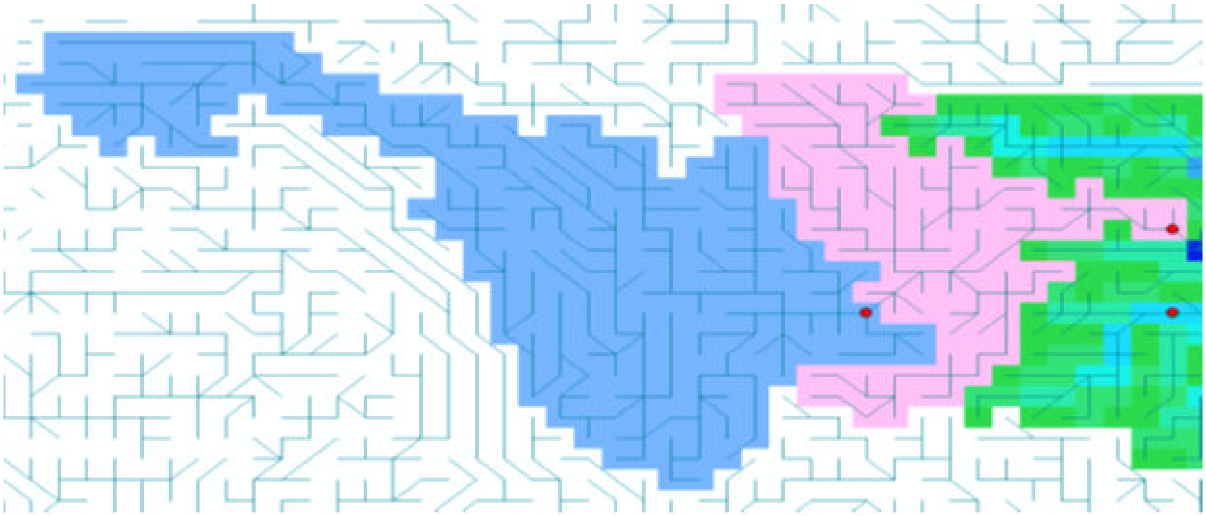


Figure 1: Upstream catchment (blue) and downstream catchment (red)

10.9.2 Detecting the downstream point

The inflow point of the new catchment has to be in the new mask and preferable one grid cell in flow direction below the upstream station e.g. 1 gridcell North East of Zhutuo (see purple circle in fig. 2)

The inflow point has the lon/lat 106.25 29.25

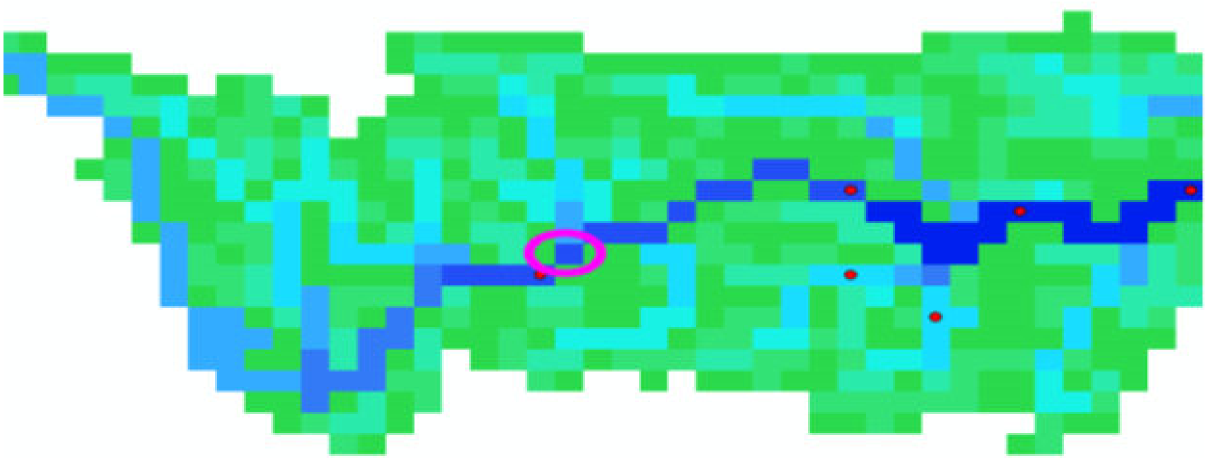


Figure 2: Downstream point

10.9.3 Run the best calibration scenario upstream

In order to get a long inflow timeserie for the inflow point (here: Zhutuo) you need to run the best scenario of the upstream catchment (here: 31_best)

- Change into the folder ../catchments/best
- Change settings file from:

```
StepStart = 1/1/1996
SpinUp = 1/1/2002
StepEnd =  31/12/2013
```

- To:

```
StepStart = 1/1/1990
SpinUp = 1/1/1996
StepEnd =  31/12/2013
```

Results is a time series from 1/1/1990 – 31/12/2013 in: discharge_daily.tss

10.9.4 Create an inflow file from the long timeseries of outlet(s)

- Create a folder ../inflow
- Copy the ../catchments/31_best/discharge_daily.tss to ../inflow/zhutuo.tss

10.9.5 Create a downstream calibration settings (directories, templates etc.)

Create downstream calibration settings as before

- Copy everything from upstream catchment (e.g. zhutuo) but not catchments
- Create empty catchments folder
- Create a observed discharge file in observed
- Change settings.txt accordingly
- Change settings.ini accordingly

Test the catchment setting!

But do not create an initial run yet!

10.9.6 Change the settings file

Change the settings file of the downstream calibration so that it includes the inflow from upstream Change the part of the settings.ini:

```
[Option]
inflow = True
[INFLOW]
#-------------------------------------------------------
# if option inflow = true
# the inflow from outside is added as inflowpoints
In_Dir = $(FILE_PATHS:PathRoot)/calibration/calibration_yichang/inflow
# nominal map with locations of (measured)inflow hydrographs [cu m / s]
InflowPoints = 106.25 29.25
InLocal = True
.
# if InflowPoints is a map, this flag is to identify if it is global (False) or local
↪(True)
# observed or simulated input hydrographs as time series [cu m / s]
# Note: that identifiers in time series have to correspond to InflowPoints
```

(continues on next page)

(continued from previous page)

```
# can be several timeseries in one file or different files e.g. main.tss mosel.tss
QInTS = zhutuo.tss
```

Test it!

Generate initial file for warm start Use initial file for calibration

10.10 Joining best sub-basin results to calibration maps

1. You need all runs done for all sub-basins
2. A region map

For each subbasin a unique number e.g. Zambezi basin

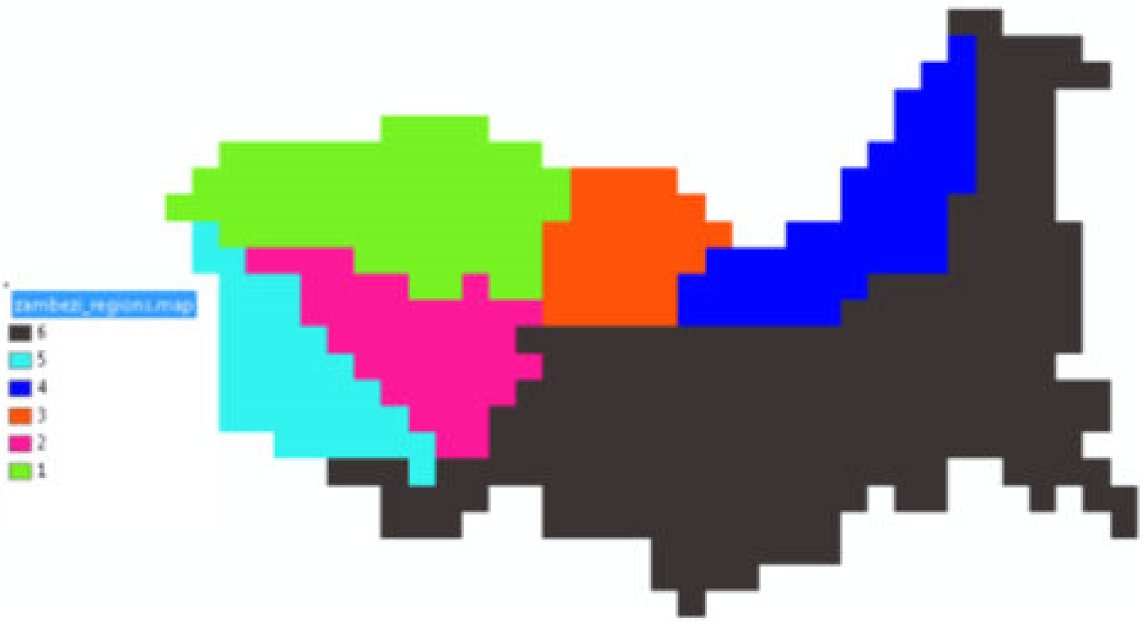


Figure 3 Sub-basin map with a unique identifier for each subbasin

3. You need a working PCRaster installation
4. The settings file settings.txt has to be changed:

```
[DEFAULT]
Root = P:/watmodel/CWATM/calibration/calibration_zambezi
# root directory where all subbasin are in
.
[Catchments]
catch = lukulu, katima, kafue, luangwa, kwando, tete
# name of the subbasin, has to be the same as the folder name in root
# the order has to be the same as in the region map
.
[region]
regionmap = P:/watmodel/CWATM/calibration_tutorial/calibration/
↪CreateCalibrationMaps/zambezi_regions.map
# region map, the order has to be the same a [Catchment]
```

(continues on next page)

(continued from previous page)

```
.
[Path]
Templates = %(Root)s/templates
SubCatchmentPath = %(Root)s/catchments
ParamRanges = %(Root)s/Join/ParamRanges.csv
.
Result = P:/watmodel/CWATM/calibration_tutorial/calibration/CreateCalibrationMaps/
↪results
# here are the results
.
PCRHOME = C:\PCRaster\bin
# Where is your PCraster installation?
```

5. Run python CAL_5_PARAMETER_MAPS.py

10.11 Calibration parameters

Snow
1. Snowmelt coefficient in [m/C deg/day] as a degree-day factor
Evapotranspiration
2. Crop factor as an adjustment to crop evapotranspiration
Soil
3. Soil depth factor: a factor for the overall soil depth of soil layer 1 and 2
4. Preferential bypass flow: empirical shape parameter of the preferential flow relation
5. Infiltration capacity parameter: empirical shape parameter b of the ARNO model
Groundwater
6. Interflow factor: factor to adjust the amount which percolates from interflow to groundwater
7. Recession coefficient factor: factor to adjust the base flow recession constant (the contribution from groundwater to baseflow)
Routing
8. Runoff concentration factor: a factor for the concentration time of run-off in each grid-cell
9. Channel Manning's n factor: a factor roughness in channel routing
10. Channel, lake and river evaporation factor: factor to adjust open water evaporation
Reservoir & lakes
11. Normal storage limit: the fraction of storage capacity used as normal storage limit
12. Lake A factor : factor to channel width and weir coefficient as a part of the Poleni weir equation

10.12 Calibration tool structure

10.13 References

- Beck, H. E., A. I. J. M. van Dijk, A. de Roo, D. G. Miralles, T. R. McVicar, J. Schellekens and L. A. Bruijnzeel (2016). "Global-scale regionalization of hydrologic model parameters." Water Resources Research 52(5): 3599-3622.
- Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE Transactions on Evolutionary Computation 6(2): 182-197.

- Fortin, F. A., F. M. De Rainville, M. A. Gardner, M. Parizeau and C. Gagńe (2012). "DEAP: Evolutionary algorithms made easy." Journal of Machine Learning Research 13: 2171-2175.
- Greve, P., L. Gudmundsson, B. Orlowsky and S. I. Seneviratne (2016). "A two-parameter Budyko function to represent conditions under which evapotranspiration exceeds precipitation." Hydrology and Earth System Sciences 20(6): 2195-2205.
- Gupta, H. V., H. Kling, K. K. Yilmaz and G. F. Martinez (2009). "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling." Journal of Hydrology 377(1-2): 80-91.
- Kling, H., M. Fuchs and M. Paulin (2012). "Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios." Journal of Hydrology 424-425: 264-277.
- Samaniego, L., R. Kumar, S. Thober, O. Rakovec, M. Zink, N. Wanders, S. Eisner, H. Müller Schmied, E. Sutanudjaja, K. Warrach-Sagi and S. Attinger (2017). "Toward seamless hydrologic predictions across spatial scales." Hydrology and Earth System Sciences 21(9): 4323-4346.

CHAPTER 11

Model download

Contents

11.1 GNU General public license

Version 3, 29 June 2007

11.1.1 Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

11.1.2 TERMS AND CONDITIONS

0. Definitions

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
>
> c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

> a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
>
> b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
>
> c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
>
> d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
>
> e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
>
> f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

11.1.3 Terms and Conditions of Use of the Community Water Model

Stipulations regarding the use of the Commumunity Water Model (hereinafter *CWATM*) provided here by the International Institute for Applied System Analysis, Laxenburg, Austria (IIASA) are as follows:

1. Copyright

The owners of the copyright of CWATM are Peter Burek, Yoshihide Wada, Yusuke Satoh and Peter Greve (hereinafter the "Developer") at IIASA.

2. Reprints and quotations

In the event of the publication of a document or scientific paper using CWATM or its modified version, the following paper must be referenced:

paper is still under work.

At the moment refer to:

Burek, P., Satoh, Y., Greve, P., Kahil, T. and Wada, Y. 2017: The Community Water Model (CWATM) / Development of a community driven global water model. Geophysical Research Abstracts. Vol. 19, EGU2017-9769

3. Usage

Usage is regulated by GNU General Public License V3 (see above)

4. Final Remarks

We as developers belief that CWATM should be utilize to encourage ideas and to advance hydrological, environmental science and stimulate integration into other science disciplines.

CWATM is based on existing knowledge of hydrology realized with Python and C++. Especially ideas from HBV, PCR-GLOBE, LISFLOOD, H08, Matsiro are used for inspiration.

Your support is more then welcome and highly appreciated

The developers of CWAT Model

11.2 Download

11.2.1 Download pdf

`CWATM_MANUAL.pdf`

11.2.2 Source code - Community Water Model

The source code of CWATM is freely available under the GNU General Public License.
Please see its *Terms and Conditions of Use of the Community Water Model*

Source code on Github repository of CWATM

Please use the actual Python 3.7 version
From 2019 we are not maintaining the Python 2.7 version
In case of trouble, try the executable version cwatmexe.zip

Warning: The source code is free, but we can give only limited support, due to limited person power!

11.2.3 Global dataset

If you are interested in obtaining the gloabl data set,
please send an email to wfas.info@iiasa.ac.at
We will give you access to our ftp server

11.2.4 Contact CWATM

www.iiasa.ac.at/cwatm
wfas.info@iiasa.ac.at

11.3 Source code

```
--------------------------------------------------
######## ##          ##  ####  ######  ##    ##
##       ##          ## ##  ##   ##   ####  ####
##        ##        ##  ##  ##   ##   ## #### ##
##        ##   ##   ## ########  ##  ##   ##   ##
##         ## #### ##  ##    ##  ##  ##        ##
##         ####  #### ##      ## ## ##          ##
##########  ##    ##  ##      ## ## ##          ##

Community WATer Model
--------------------------------------------------
```

11.3.1 cwatm module

Note: Base module: run with settings file e.g. python cwatm.py settings.ini

```
--------------------------------------------------
######## ##          ##  ####  ######  ##    ##
##       ##          ## ##  ##   ##   ####  ####
##        ##        ##  ##  ##   ##   ## #### ##
##        ##   ##   ## ########  ##  ##   ##   ##
##         ## #### ##  ##    ##  ##  ##        ##
##         ####  #### ##      ## ## ##          ##
##########  ##    ##  ##      ## ## ##          ##

Community WATer Model
```

CWATM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

———————————————————

`cwatm3.CWATMexe()`

Base subroutine of the CWATM model

- parses the settings file
- read the information for the netcdf files
- check if dates are alright
- check flags for screen output
- runs the model

`class cwatm3.CWATModel`

Bases: *cwatm_initial.CWATModel_ini*, *cwatm_dynamic.CWATModel_dyn*

Initial and dynamic part of the CWATM model

- initial part takes care of all the non temporal initialiation procedures
- dynamic part loops over time

`dynamic()`

Dynamic part of CWATM calls the dynamic part of the hydrological modules Looping through time and space

Note: if flags set the output on the screen can be changed e.g.

- v: no output at all
- l: time and first gauge discharge
- t: timing of different processes at the end

`i = 1`

`cwatm3.GNU()`

prints GNU General Public License information

`cwatm3.headerinfo()`

Print the information on top of each run

`cwatm3.usage()`

Prints some lines describing how to use this program which arguments and parameters it accepts, etc

- -q –quiet output progression given as .
- -v –veryquiet no output progression is given
- -l –loud output progression given as time step, date and discharge
- -c –check input maps and stack maps are checked, output for each input map BUT no model run
- -h –noheader .tss file have no header and start immediately with the time series
- -t –printtime the computation time for hydrological modules are printed

11.3.2 cwatm_dynamic module

class cwatm_dynamic.**CWATModel_dyn**

Bases: *management_modules.dynamicModel.DynamicModel*

dynamic()

Dynamic part of CWATM calls the dynamic part of the hydrological modules Looping through time and space

Note: if flags set the output on the screen can be changed e.g.

- v: no output at all
- l: time and first gauge discharge
- t: timing of different processes at the end

i = 1

11.3.3 cwatm_initial module

class cwatm_initial.**CWATModel_ini**

Bases: *management_modules.dynamicModel.DynamicModel*

CWATN initial part this part is to initialize the variables. It will call the initial part of the hydrological modules

i = 1

11.3.4 hydrological_modules package

Initialize

miscInitial module

Initializing some variables

class hydrological_modules.miscInitial.**miscInitial**(*misc_variable*)

Bases: object

Miscellaneous repeatedly used expressions Definition if cell area comes from regular grid e.g. 5x5km or from irregular lat/lon Conversion factors between m3 and mm etc.

Note: Only used in the initial phase.

initial()

Initialization of some basic parameters e.g. cellArea

- grid area, length definition
- conversion factors
- conversion factors for precipitation and pot evaporation

initcondition module

Load initial storage parameter maps

class hydrological_modules.initcondition.**initcondition**(*initcondition_variable*)

Bases: object

READ/WRITE INITIAL CONDITIONS all initial condition can be stored at the end of a run to be used as a **warm** start for a following up run

dynamic()

Dynamic part of the initcondition module write initital conditions into a single netcdf file

Note: Several dates can be stored in different netcdf files

initial()

initial part of the initcondition module Puts all the variables which has to be stored in 2 lists:

- initCondVar: the name of the variable in the init netcdf file
- initCondVarValue: the variable as it can be read with the 'eval' command

Reads the parameter *save_initial* and *save_initial* to know if to save or load initial values

load_initial(*name*, *default=0.0*, *number=None*)

First it is checked if the initial value is given in the settings file

- if it is <> None it is used directly
- if None it is loaded from the init netcdf file

Parameters

- **name** – Name of the init value
- **default** – default value -> default is 0.0
- **number** – in case of snow or runoff concentration several layers are included: number = no of the layer

Returns spatial map or value of initial condition

landcoverType module

Generate landcover types

class hydrological_modules.landcoverType.**landcoverType**(*landcoverType_variable*)

Bases: object

LAND COVER TYPE

runs the 6 land cover types through soil procedures

This routine calls the soil routine for each land cover type

dynamic()

Dynamic part of the land cover type module

Calculating soil for each of the 6 land cover class

- calls evaporation_module.dynamic

- calls interception_module.dynamic
- calls soil_module.dynamic
- calls sealed_water_module.dynamic

And sums every thing up depending on the land cover type fraction

dynamic_fracIrrigation (*init=False*, *dynamic=True*)
Dynamic part of the land cover type module

Calculating fraction of land cover

- loads the fraction of landcover for each year from netcdf maps
- calculate the fraction of 6 land cover types based on the maps

Parameters

- **init** – (optional) True: set for the first time of a run
- **dynamic** – used in the dynmic run not in the initial phase

Returns

-

initial ()
Initial part of the land cover type module Initialise the six land cover types

- Forest
- Grasland/non irrigated land
- Irrigation
- Paddy iirigation
- Sealed area
- Water covered area

And initialize the soil variables

Hydrology I - from rain to soil

readmeteo module

Read meteorological input data

class hydrological_modules.readmeteo.**readmeteo** (*readmeteo_variable*)
Bases: object

READ METEOROLOGICAL DATA

reads all meteorological data from netcdf4 files

downscaling1 (*input*, *downscale=0*)
Downscaling based on elevation correction for temperature and pressure

Parameters

- **input** –
- **downscale** – 0 for no change, 1: for temperature change 6 deg per 1km , 2 for psurf

Returns input - downscaled input data

downscaling2 (*input*, *downscaleName=''*, *wc2=0*, *wc4=0*, *downscale=0*)

Downscaling based on Delta method:

Note:

References

Moreno and Hasenauer 2015:

ftp://palantir.boku.ac.at/Public/ClimateData/Moreno_et_al-2015-International_Journal_of_Climatology.pdf

Mosier et al. 2018:

http://onlinelibrary.wiley.com/doi/10.1002/joc.5213/epdf

Parameters

- **input** – low input map
- **downscaleName** – High resolution monthly map from WorldClim
- **wc2** – High resolution WorldClim map
- **wc4** – upscaled to low resolution
- **downscale** – 0 for no change, 1: for temperature , 2 for pprecipitation, 3 for psurf

Returns input - downscaled input data

Returns wc2

Returns wc4

dynamic ()

Dynamic part of the readmeteo module

Read meteo input maps from netcdf files

Note: If option *calc_evaporation* is False only precipitation, avg. temp., and 2 evaporation vlaues are read Otherwise all the variable needed for Penman-Monteith

---

Note: If option *TemperatureInKelvin* = True temperature is assumed to be Kelvin instead of Celsius!

---

initial ()

Initial part of meteo

read multiple file of input

inflow module

Read river discharge time series as inflow data

class hydrological_modules.inflow.**inflow** (*inflow_variable*)

Bases: object

READ INFLOW HYDROGRAPHS (OPTIONAL) If option "inflow" is set to 1 the inflow hydrograph code is used otherwise dummy code is used

dynamic()
Dynamic part of the inflow module Use the inflow points to add inflow from time series file(s)

initial()
Initial part of the inflow module Get the inflow points

calls function `hydrological_modules.getlocOutpoints()` calls function `hydrological_modules.join_struct_arrays2()`

snow_frost module

Calculate snow and frost

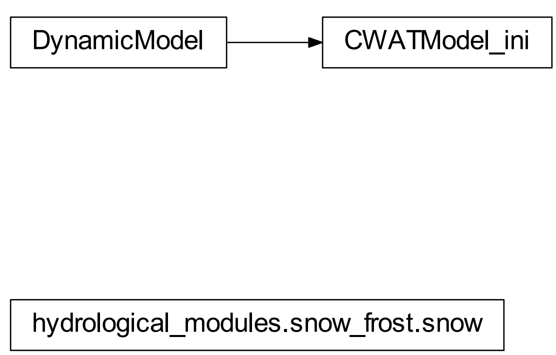
class `hydrological_modules.snow_frost.`**snow**(*snow_variable*)
Bases: `object`

RAIN AND SNOW

Domain: snow calculations evaluated for center points of up to 7 sub-pixel snow zones 1 -7 which each occupy a part of the pixel surface

Variables *snow* and *rain* at end of this module are the pixel-average snowfall and rain

Inheritance: `inheritance-diagram`



dynamic()
Dynamic part of the snow module

Distinguish between rain/snow and calculates snow melt and glacier melt The equation is a modification of:

References

Speers, D.D., Versteeg, J.D. (1979) Runoff forecasting for reservoir operations - the pastand the future. In: Proceedings 52nd Western Snow Conference, 149-156

Frost index in soil [degree days] based on:

References

Molnau and Bissel (1983, A Continuous Frozen Ground Index for Flood Forecasting. In: Maidment, Handbook of Hydrology, p. 7.28, 7.55)

Todo: calculate sinus shape function for the southern hemisspere

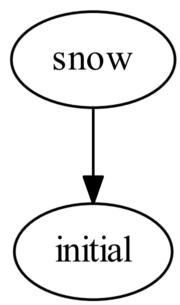
test of math1

$$a = \sqrt{2}$$

initial()

Initial part of the snow and frost module

- loads all the parameters for the day-degree approach for rain, snow and snowmelt
- loads the parameter for frost



evaporationPot module

Calculate potential Evaporation

class hydrological_modules.evaporationPot.**evaporationPot**(*evaporationPot_variable*)

Bases: `object`

POTENTIAL REFERENCE EVAPO(TRANSPI)RATION Calculate potential evapotranspiration from climate data mainly based on FAO 56 and LISVAP Based on Penman Monteith

References

http://www.fao.org/docrep/X0490E/x0490e08.htm#penman%20monteith%20equation

http://www.fao.org/docrep/X0490E/x0490e06.htm http://www.fao.org/docrep/X0490E/x0490e06.htm

https://ec.europa.eu/jrc/en/publication/eur-scientific-and-technical-research-reports/lisvap-evaporation-pre-processor-lisflood-water-balance-and-flood-simulation-model

dynamic()

Dynamic part of the potential evaporation module Based on Penman Monteith - FAO 56

Returns ETRef - potential reference evapotranspiration rate [m/day] EWRef - potential evaporation rate from water surface [m/day]

initial()

Initial part of evaporation type module Load inictial parameters

Note: Only run if *calc_evaporation* is True

evaporation module

Calculate actual evapotranspiration

class hydrological_modules.evaporation.**evaporation**(*evaporation_variable*)

Bases: `object`

Evaporation module Calculate potential evaporation and pot. transpiration

dynamic(*coverType*, *No*)

Dynamic part of the soil module

calculating potential Evaporation for each land cover class with kc factor get crop coefficient, use potential ET, calculate potential bare soil evaporation and transpiration

Parameters

- **coverType** – Land cover type: forest, grassland ...
- **No** – number of land cover type: forest = 0, grassland = 1 ...

Returns potential evaporation from bare soil, potential transpiration

interception module

Calculate interception

class hydrological_modules.interception.**interception**(*interception_variable*)

Bases: `object`

INTERCEPTION

dynamic(*coverType*, *No*)

Dynamic part of the interception module calculating interception for each land cover class

Parameters

- **coverType** – Land cover type: forest, grassland ...
- **No** – number of land cover type: forest = 0, grassland = 1 ...

Returns interception evaporation, interception storage, reduced pot. transpiration

sealed_water module

Calculate water runoff from impermeable surface

class hydrological_modules.sealed_water.**sealed_water**(*sealed_water_variable*)
Bases: object

Sealed and open water runoff

calculated runoff from impermeable surface (sealed) and into water bodies

dynamic(*coverType*, *No*)
Dynamic part of the sealed_water module

runoff calculation for open water and sealed areas

Parameters

- **coverType** – Land cover type: forest, grassland . . .
- **No** – number of land cover type: forest = 0, grassland = 1 . . .

Hydrology II - from soil to river

soil module

** Calculate fluxes in 3 layer soil**

class hydrological_modules.soil.**soil**(*soil_variable*)
Bases: object

SOIL

Caclulation vertical transfer of water based on Arno scheme

dynamic(*coverType*, *No*)
Dynamic part of the soil module

For each of the land cover classes the vertical water transport is simulated Distribution of water holding capiacity in 3 soil layers based on saturation excess overland flow, preferential flow Dependend on soil depth, soil hydraulic parameters

initial()
Initial part of the soil module

- Initialize all the hydraulic properties of soil
- Set soil depth

capillarRise module

Calculate capillar rise from groundwater

class hydrological_modules.capillarRise.**capillarRise**(*capillarRise_variable*)
Bases: object

CAPPILAR RISE calculate cell fraction influenced by capillary rise

dynamic()
Dynamic part of the capillar Rise module calculate cell fraction influenced by capillary rise depending on appr. height of groundwater and relative elevation of grid cell

Returns capRiseFrac = cell fraction influenced by capillary rise

groundwater module

Calculate groundwater

class hydrological_modules.groundwater.**groundwater**(*groundwater_variable*)

Bases: object

GROUNDWATER

dynamic()

Dynamic part of the groundwater module Calculate groundweater storage and baseflow

initial()

Initial part of the groundwater module

- load parameters from settings file
- initial groundwater storage

runoff_concentration module

Calculate runoff concentration - from grid cell to grid cell corner

class hydrological_modules.runoff_concentration.**runoff_concentration**(*runoff_concentration_variable*)

Bases: object

Runoff concentration

this is the part between runoff generation and routing for each gridcell and for each land cover class the generated runoff is concentrated at a corner of a gridcell this concentration needs some lag-time (and peak time) and leads to diffusion lag-time/ peak time is calculated using slope, length and land cover class diffusion is calculated using a triangular-weighting-function

dynamic()

Dynamic part of the runoff concentration module

For surface runoff for each land cover class and for interflow and for baseflow the runoff concentration time is calculated

Note: the time demanding part is calculated in a c++ library

initial()

Initial part of the runoff concentration module

Setting the peak time for:

- surface runoff = 3
- interflow = 4
- baseflow = 5

based on the slope the concentration time for each land cover type is calculated

Note: only if option **includeRunoffConcentration** is TRUE

Hydrology III - Socio-economic - Water demand

waterdemand module

Calculate water demand from different sectors

Naming convention:

-
-
-
-

class `hydrological_modules.waterdemand.`**`waterdemand`**(*waterdemand_variable*)

Bases: `object`

WATERDEMAND

calculating water demand - Industrial, domenstic based on precalculated maps Agricultural water demand based on water need by plants

`dynamic`()

Dynamic part of the water demand module

- calculate the fraction of water from surface water vs. groundwater
- get non-Irrigation water demand and its return flow fraction

`initial`()

Initial part of the water demand module

Set the water allocation

Hydrology IV - Lakes, reservoirs and river

lakes_reservoirs module

Calculate water retention in lakes

class `hydrological_modules.lakes_reservoirs.`**`lakes_reservoirs`**(*lakes_reservoirs_variable*)

Bases: `object`

LAKES AND RESERVOIRS

Note: Calculate water retention in lakes and reservoirs

Using the **Modified Puls approach** to calculate retention of a lake See also: LISFLOOD manual Annex 3 (Burek et al. 2013)

for Modified Puls Method the Q(inflow)1 has to be used. It is assumed that this is the same as Q(inflow)2 for the first timestep has to be checked if this works in forecasting mode!

Lake Routine using Modified Puls Method (see Maniak, p.331ff) $(Qin1 + Qin2)/2 - (Qout1 + Qout2)/2 = (S2 - S1)/dtime$

changed into:

$(S2/dtime + Qout2/2) = (S1/dtime + Qout1/2) - Qout1 + (Qin1 + Qin2)/2$

outgoing discharge (Qout) are linked to storage (S) by elevation.

Now some assumption to make life easier:

1.) storage volume is increase proportional to elevation: S = A * H where: H: elevation, A: area of lake

2.) $outgoingdischarge = c * b * H^2.0$ (c: weir constant, b: width)

2.0 because it fits to a parabolic cross section see (Aigner 2008) (and it is much easier to calculate (that's the main reason)

c: for a perfect weir with mu=0.577 and Poleni: $2/3\mu * \sqrt{2 * g} = 1.7$

c: for a parabolic weir: around 1.8

because it is a imperfect weir:$C = c * 0.85 = 1.5$

results in a formular: $Q = 1.5 * b * H * *2 = a * H * *2- > H = \sqrt{Q/a}$

Solving the equation:

$(S2/dtime + Qout2/2) = (S1/dtime + Qout1/2) - Qout1 + (Qin1 + Qin2)/2$

$SI = (S2/dtime + Qout2/2) = (A * H)/DtRouting + Q/2 = A/(DtRouting * \sqrt(a) * \sqrt(Q) + Q/2$

-> replacement: $A/(DtRouting * sqrt(a)) = Lakefactor, Y = \sqrt(Q)$

$Y^2 + 2 * Lakefactor * Y - 2 * SI = 0$

solution of this quadratic equation:

$Q = (-LakeFactor + sqrtLakeFactor^2 + 2 * SI)^2$

---

dynamic()

Dynamic part set lakes and reservoirs for each year

Note: Flow out of lake: solving the equation:

$(S2/dtime + Qout2/2) = (S1/dtime + Qout1/2) - Qout1 + (Qin1 + Qin2)/2$

$SI = (S2/dtime + Qout2/2) = (A * H)/DtSec + Q/2 = A/(DtSec * \sqrt{a} * \sqrt{Q} + Q/2$

-> replacement: $A/(DtSec * sqrt(a)) = Lakefactor, Y = sqrt(Q)$

$Y^2 + 2 * Lakefactor * Y - 2 * SI = 0$

solution of this quadratic equation:

$Q = (-LakeFactor + \sqrt{LakeFactor^2 + 2 * SI})^2$

---

dynamic_inloop(*NoRoutingExecuted*)

Dynamic part to calculate outflow from lakes and reservoirs

- lakes with modified Puls approach
- reservoirs with special filling levels

Parameters **NoRoutingExecuted** – actual number of routing substep

Returns outLdd: outflow in m3 to the network

Note: outflow to adjected lakes and reservoirs is calculated separately

---

initWaterbodies()

Initialize water bodies Read parameters from maps e.g area, location, initial average discharge, type 9reservoir or lake) etc.

Compress numpy array from mask map to the size of lakes+reservoirs (marked as capital C at the end of the variable name)

initial_lakes()

Initial part of the lakes module Using the **Modified Puls approach** to calculate retention of a lake

initial_reservoirs()

Initial part of the reservoir module Using the appraoch of LISFLOOD

See also:

LISFLOOD manual Annex 1: (Burek et al. 2013)

lakes_res_small module

Calculate water retention in small lakes

class hydrological_modules.lakes_res_small.**lakes_res_small**(*lakes_res_small_variable*)

Bases: object

Small LAKES AND RESERVOIRS

Note: Calculate water retention in lakes and reservoirs

Using the **Modified Puls approach** to calculate retention of a lake See also: LISFLOOD manual Annex 3 (Burek et al. 2013)

for Modified Puls Method the Q(inflow)1 has to be used. It is assumed that this is the same as Q(inflow)2 for the first timestep has to be checked if this works in forecasting mode!

Lake Routine using Modified Puls Method (see Maniak, p.331ff) $(Qin1 + Qin2)/2 - (Qout1 + Qout2)/2 = (S2 - S1)/dtime$

changed into:

$(S2/dtime + Qout2/2) = (S1/dtime + Qout1/2) - Qout1 + (Qin1 + Qin2)/2$

outgoing discharge (Qout) are linked to storage (S) by elevation.

Now some assumption to make life easier:

1.) storage volume is increase proportional to elevation: S = A * H where: H: elevation, A: area of lake

2.) $outgoingdischarge = c * b * H^2.0$ (c: weir constant, b: width)

2.0 because it fits to a parabolic cross section see (Aigner 2008) (and it is much easier to calculate (that's the main reason)

c: for a perfect weir with mu=0.577 and Poleni: $2/3\mu * \sqrt{2 * g} = 1.7$

c: for a parabolic weir: around 1.8

because it is a imperfect weir:$C = c * 0.85 = 1.5$

results in a formular: $Q = 1.5 * b * H * *2 = a * H * *2- > H = \sqrt{Q/a}$

Solving the equation:

$(S2/dtime + Qout2/2) = (S1/dtime + Qout1/2) - Qout1 + (Qin1 + Qin2)/2$

$SI = (S2/dtime + Qout2/2) = (A * H)/DtRouting + Q/2 = A/(DtRouting * \sqrt(a) * \sqrt(Q) + Q/2$

-> replacement: $A/(DtRouting * sqrt(a)) = Lakefactor, Y = \sqrt(Q)$

$Y^2 + 2 * Lakefactor * Y - 2 * SI = 0$

solution of this quadratic equation:

$Q = (-LakeFactor + sqrt LakeFactor^2 + 2 * SI)^2$

---

dynamic()

Dynamic part to calculate outflow from small lakes and reservoirs

- lakes with modified Puls approach
- reservoirs with special filling levels

Note: **Flow out of lake:** solving the equation:

$(S2/dtime + Qout2/2) = (S1/dtime + Qout1/2) - Qout1 + (Qin1 + Qin2)/2$

$SI = (S2/dtime + Qout2/2) = (A * H)/DtSec + Q/2 = A/(DtSec * \sqrt{a} * \sqrt{Q} + Q/2$

-> replacement: $A/(DtSec * sqrt(a)) = Lakefactor, Y = sqrt(Q)$

$Y^2 + 2 * Lakefactor * Y - 2 * SI = 0$

solution of this quadratic equation:

$Q = (-LakeFactor + \sqrt{LakeFactor^2 + 2 * SI})^2$

Returns outflow in m3 to the network

initial()

Initialize small lakes and reservoirs Read parameters from maps e.g area, location, initial average discharge, type: reservoir or lake) etc.

routing_reservoirs.routing_kinematic module

River routing - kinematic wave

class hydrological_modules.routing_reservoirs.routing_kinematic.**routing_kinematic**(*routing_kine*

Bases: object

ROUTING

routing using the kinematic wave

dynamic()

Dynamic part of the routing module

- calculate evaporation from channels
- calculate riverbed exchange between riverbed and groundwater
- if option **waterbodies** is true, calculate retention from water bodies
- calculate sideflow -> inflow to river
- calculate kinematic wave -> using C++ library for computational speed

initial()
Initial part of the routing module

- load and create a river network
- calculate river network parameter e.g. river length, width, depth, gradient etc.
- calculate initial filling
- calculate manning's roughness coefficient

routing_reservoirs.routing_sub module

Sub routines for river routing

hydrological_modules.routing_reservoirs.routing_sub.**Compress**(*map*, *mask*)
compressing map from 2D to 1D without missing values

Parameters

- **map** – input map
- **mask** – mask map

Returns compressed map

hydrological_modules.routing_reservoirs.routing_sub.**catchment1**(*dirUp*, *points*)
calculates all cells which belongs to a catchment from point onward

Parameters

- **dirUp** –
- **points** –

Returns subcatchment

hydrological_modules.routing_reservoirs.routing_sub.**decompress**(*map*)
Decompressing map from 1D to 2D with missing values

Parameters **map** – compressed map

Returns decompressed 2D map

hydrological_modules.routing_reservoirs.routing_sub.**defLdd2**(*ldd*)
defines river network

Parameters **ldd** – river network

Returns ldd variables

hydrological_modules.routing_reservoirs.routing_sub.**dirDownstream**(*dirUp*, *lddcomp*, *dirDown*)
runs the river network tree downstream - from source to outlet

Parameters

- **dirUp** –
- **lddcomp** –
- **dirDown** –

Returns direction downstream

`hydrological_modules.routing_reservoirs.routing_sub.`**`dirUpstream`**(*dirshort*)

runs the network tree upstream from outlet to source

Parameters **`dirshort`** –

Returns direction upstream

`hydrological_modules.routing_reservoirs.routing_sub.`**`downstream1`**(*dirUp*, *weight*)

calculated 1 cell downstream

Parameters

- **`dirUp`** –
- **`weight`** –

Returns dowmnstream 1 cell

`hydrological_modules.routing_reservoirs.routing_sub.`**`lddrepair`**(*lddnp*, *lddOrder*)

repairs a river network

- eliminate unsound parts
- add pits at points with no connections

Parameters

- **`lddnp`** – rivernetwork as 1D array
- **`lddOrder`** –

Returns repaired ldd

`hydrological_modules.routing_reservoirs.routing_sub.`**`postorder`**(*dirUp*, *catchment*, *node*, *catch*, *dirDown*)

Routine to run a postoder tree traversal

Parameters

- **`dirUp`** –
- **`catchment`** –
- **`node`** –
- **`catch`** –
- **`dirDown`** –

Returns dirDown and catchment

`hydrological_modules.routing_reservoirs.routing_sub.`**`subcatchment1`**(*dirUp*, *points*, *ups*)

calculates subcatchments of points

Parameters

- **`dirUp`** –
- **`points`** –
- **`ups`** –

Returns subcatchment

hydrological_modules.routing_reservoirs.routing_sub.**upstream1**(*downstruct*, *weight*)

Calculates 1 cell upstream

Parameters

- **downstruct** –
- **weight** –

Returns upstream 1cell

hydrological_modules.routing_reservoirs.routing_sub.**upstreamArea**(*dirDown*, *dirshort*, *area*)

calculates upstream area

Parameters

- **dirDown** – array which point from each cell to the next downstream cell
- **dirshort** –
- **area** – area in m2 for a single gridcell

Returns upstream area

Hydrology V - Water balance

waterbalance module

class hydrological_modules.waterbalance.**waterbalance**(*waterbalance_variable*)

Bases: object

WATER BALANCE

- check if water balnace per time step is ok (= 0)
- produce an annual overview - income, outcome storage

checkWaterSoilGround()

Check water balance of snow, vegetation, soil, groundwater

dynamic()

Dynamic part of the water balance module If option **sumWaterBalance** sum water balance for certain variables

initial()

Initial part of the water balance module

waterBalanceCheck(*fluxesIn*, *fluxesOut*, *preStorages*, *endStorages*, *processName*, *printTrue=False*)

Dynamic part of the water balance module

Returns the water balance for a list of input, output, and storage map files

Parameters

- **fluxesIn** – income
- **fluxesOut** – this goes out
- **preStorages** – this was in before
- **endStorages** – this was in afterwards

- **processName** – name of the process
- **printTrue** – calculate it?

Returns

-

waterBalanceCheckSum (*fluxesIn*, *fluxesOut*, *preStorages*, *endStorages*, *processName*, *printTrue=False*)
Returns the water balance for a list of input, output, and storage map files and sums it up for a catchment

Parameters

- **fluxesIn** – income
- **fluxesOut** – this goes out
- **preStorages** – this was in before
- **endStorages** – this was in afterwards
- **processName** – name of the process
- **printTrue** – calculate it?

Returns Water balance as output on the screen

11.3.5 management_modules package

Data management

data_handling module

Managing data and data handling

management_modules.data_handling.**cbinding** (*inBinding*)
Check if variable in settings file has a counterpart in the source code

Parameters **inBinding** – parameter in settings file

management_modules.data_handling.**checkOption** (*inBinding*)
Check if option in settings file has a counterpart in the source code

Parameters **inBinding** – parameter in settings file

management_modules.data_handling.**compressArray** (*map*, *name='None'*, *zeros=0.0*)
Compress 2D array with missing values to 1D array without missing values

Parameters

- **map** – in map
- **name** – filename of the map
- **zeros** – add zeros (default= 0) if values of map are to big or too small

Returns Compressed 1D array

management_modules.data_handling.**decompress** (*map*, *pcmap*)
Decompress 1D array without missing values to 2D array with missing values

Parameters

- **map** – numpy 1D array as input

- **pcmap** – if True map is used as .map format

Returns 2D array for displaying

`management_modules.data_handling.`**`divideValues`**(*x*, *y*, *default=0.0*)

returns the result of a division that possibly involves a zero

Parameters

- **x** –
- **y** – divisor
- **default** – return value if y =0

Returns result of x/y or default if y = 0

`management_modules.data_handling.`**`getmeta`**(*key*, *varname*, *alternative*)

get the meta data information for the netcdf output from the global variable metaNetcdfVar

Parameters

- **key** – key
- **varname** – variable name eg self.var.Precipitation

Returns metadata information

`management_modules.data_handling.`**`loadmap`**(*name*, *lddflag=False*, *compress=True*, *local=False*, *cut=True*)

load a static map either value or pc raster map or netcdf

Parameters

- **name** – name of map
- **lddflag** – if True the map is used as a ldd map
- **compress** – if True the return map will be compressed
- **local** – if True the map is local and will be not cut
- **cut** – if True the map will be not cut

Returns 1D numpy array of map

`management_modules.data_handling.`**`loadsetclone`**(*name*)

load the maskmap and set as clone

Parameters **name** – name of mask map, can be a file or - row col cellsize xupleft yupleft -

Returns new mask map

`management_modules.data_handling.`**`mapattrNetCDF`**(*name*, *check=True*)

get the 4 corners of a netcdf map to cut the map defines the rectangular of the mask map inside the netcdf map calls function *`management_modules.data_handling.readCoord()`*

Parameters

- **name** – name of the netcdf file
- **check** – checking if netcdffile exists

Returns cut1,cut2,cut3,cut4

Raises **`if cell size is different`** – *`management_modules.messages.CWATMError()`*

management_modules.data_handling.**mapattrNetCDFMeteo**(*name*, *check=True*)
get the map attributes like col, row etc from a netcdf map and define the rectangular of the mask map inside the netcdf map calls function *management_modules.data_handling.readCoordNetCDF()*

Parameters

- **name** – name of the netcdf file
- **check** – checking if netcdffile exists

Returns cut0,cut1,cut2,cut3,cut4,cut5,cut6,cut7

management_modules.data_handling.**mapattrTiff**(*nf2*)
map attributes of a geotiff file

Parameters **nf2** –

Returns cut0,cut1,cut2,cut3

management_modules.data_handling.**metaNetCDF**()
get the map metadata from precipitation netcdf maps

management_modules.data_handling.**multinetdf**(*meteomaps*, *startcheck='dateBegin'*)

Parameters

- **meteomaps** – list of meteomaps to define start and end time
- **startcheck** – date of beginning simulation

Returns

Raises **if no map stack in meteo map folder** – *management_modules.messages.CWATMFileError()*

management_modules.data_handling.**readCoord**(*name*)
get the meta data information for the netcdf output from the global variable metaNetcdfVar

Parameters **name** – name of the netcdf file

Returns latitude, longitude, cell size, inverse cell size

management_modules.data_handling.**readCoordNetCDF**(*name*, *check=True*)
reads the map attributes col, row etc from a netcdf map

Parameters

- **name** – name of the netcdf file
- **check** – checking if netcdffile exists

Returns latitude, longitude, cell size, inverse cell size

Raises **if no netcdf map can be found** – *management_modules.messages.CWATMFileError()*

management_modules.data_handling.**readmeteodata**(*name*, *date*, *value='None'*, *addZeros=False*, *zeros=0.0*, *mapsscale=True*)
load stack of maps 1 at each timestamp in netcdf format

Parameters

- **name** – file name
- **date** –
- **value** – if set the name of the parameter is defined
- **addZeros** –

- **zeros** – default value
- **mapsscale** – if meteo maps have the same extend as the other spatial static m

Returns Compressed 1D array of meteo data

Raises

- **if data is wrong** – *management_modules.messages.CWATMError()*
- **if meteo netcdf file cannot be opened** – *management_modules.messages.CWATMFileError()*

management_modules.data_handling.**readnetcdf2**(*namebinding*, *date*, *useDaily='daily'*, *value='None'*, *addZeros=False*, *cut=True*, *zeros=0.0*, *meteo=False*, *usefilename=False*, *compress=True*)

load stack of maps 1 at each timestamp in netcdf format

Parameters

- **namebinding** – file name in settings file
- **date** –
- **useDaily** – if True daily values are used
- **value** – if set the name of the parameter is defined
- **addZeros** –
- **cut** – if True the map is clipped to mask map
- **zeros** – default value
- **meteo** – if map are meteo maps
- **usefilename** – if True filename is given False: filename is in settings file
- **compress** – True - compress data to 1D

Returns Compressed 1D array of netcdf stored data

Raises

- **if netcdf file cannot be opened** – *management_modules.messages.CWATMFileError()*
- **if netcdf file is not of the size of mask map** – *management_modules.messages.CWATMWarning()*

management_modules.data_handling.**readnetcdfInitial**(*name*, *value*, *default=0.0*)

load initial condition from netcdf format

Parameters

- **name** – file name
- **value** – netcdf variable name
- **default** – (optional) if no variable is found a warning is given and value is set to default

Returns Compressed 1D array of netcdf stored data

Raises

- **if netcdf file is not of the size of mask map** – *management_modules.messages.CWATMError()*

- **if varibale name is not included in the netcdf file** – *management_modules.messages.CWATMWarning()*

management_modules.data_handling.**readnetcdfWithoutTime**(*name*, *value='None'*)
load maps in netcdf format (has no time format)

Parameters

- **namebinding** – file name in settings file
- **value** – (optional) netcdf variable name. If not given -> last variable is taken

Returns Compressed 1D array of netcdf stored data

management_modules.data_handling.**report**(*name*, *valueIn*, *compr=True*)
For debugging: Save the 2D array as .map or .tif

Parameters

- **name** – Filename of the map
- **valueIn** – 1D or 2D array in
- **compr** – (optional) array is 1D (default) or 2D

Returns

-

```
Example:
> report(c:/temp/ksat1.map, self.var.ksat1)
```

management_modules.data_handling.**returnBool**(*inBinding*)
Test if parameter is a boolean and return an error message if not, and the boolean if everything is ok

Parameters **inBinding** – parameter in settings file

Returns boolean of inBinding

management_modules.data_handling.**setmaskmapAttr**(*x*, *y*, *col*, *row*, *cell*)
Definition of cell size, coordinates of the meteo maps and maskmap

Parameters

- **x** – upper left corner x
- **y** – upper left corner y
- **col** – number of cols
- **row** – number of rows
- **cell** – cell size

Returns

-

management_modules.data_handling.**valuecell**(*coordx*, *coordstr*)
to put a value into a raster map -> invert of cellvalue, map is converted into a numpy array first

Parameters

- **coordx** – x,y or lon/lat coordinate
- **coordstr** – String of coordinates

Returns 1D array with new value

`management_modules.data_handling.`**`writeIniNetcdf`**`(`*netfile*, *varlist*, *inputlist*`)`

write variables to netcdf init file

Parameters

- **`netfile`** – file name
- **`varlist`** – list of variable to be written in the netcdf file
- **`inputlist`** – stack of 1D arrays

Returns

-

`management_modules.data_handling.`**`writenetcdf`**`(`*netfile*, *prename*, *addname*, *varunits*, *inputmap*, *timeStamp*, *posCnt*, *flag*, *flagTime*, *nrdays=None*, *dateunit='days'*`)`

write a netcdf stack

Parameters

- **`netfile`** – file name
- **`prename`** – 1st part of variable name with tell which variable e.g. discharge
- **`addname`** – part of the variable name with tells about the timestep e.g. daily, monthly
- **`varunits`** – unit of the variable
- **`inputmap`** – 1D array to be put as netcdf
- **`timeStamp`** – time
- **`posCnt`** – calculate nummer of the indece for time
- **`flag`** – to indicate if the file is new -> netcdf header has to be written,or simply appending data
- **`flagtime`** – to indicate the variable is time dependend (not a single array!)
- **`nrdays`** – (optional) if indicate number of days are set in the time variable (makes files smaller!)
- **`dateunit`** – (optional) dateunit indicate if the timestep in netcdf is days, month or years

Returns flag: to indicate if the file is set up

timestep module

Managing time

`management_modules.timestep.`**`Calendar`**`(`*input*, *errorNo=0*`)`

Get the date from CalendarDayStart in the settings xml Reformatting the date till it fits to datetime

Parameters

- **`input`** – string from the settingsfile should be somehow a date
- **`errorNo`** – 0: check startdate, enddate 1: check startinit

Returns a datetime date

`management_modules.timestep.`**`checkifDate`**`(`*start*, *end*, *spinup*`)`

Checks if start date is earlier than end date etc And set some date variables

Parameters

- **start** – start date
- **end** – end date
- **spinup** – date till no output is generated = warming up time

Returns a list of date variable in: dateVar

`management_modules.timestep.ctbinding(inBinding)`
Check if variable in settings file has a counterpart in source code

Parameters **x** – variable in settings file to be tested

Returns

-

Raises if variable is not found send an error: *management_modules.messages.CWATMError()*

`management_modules.timestep.date2indexNew(date, nctime, calendar, select='nearest', name='')`
The original netCDF4 library cannot handle month and years Replace: date2index This one checks for days, month and years And set some date variables

Parameters

- **date** – date
- **nctime** – time unit of the netcdf file
- **select** – (optional) which date is selected, default: nearest
- **name** – (optional) name of th dataset

Returns index

`management_modules.timestep.date2str(date)`
Convert date to string of date e.g. 27/12/2018

Parameters **x** – date as (datetime)

Returns date string

`management_modules.timestep.datetoInt(dateIn, begin, both=False)`
Calculates the integer of a date from a reference date

Parameters

- **dateIn** – date
- **begin** – reference date
- **both** – if set to True both the int and the string of the date are returned

Returns integer value of a date, starting from begin date

`management_modules.timestep.datetosaveInit(initdates, begin, end)`
Calculates the save init dates

Parameters

- **initdates** – one or several dates
- **begin** – reference date
- **end** – end date

Returns integer value of a dates, starting from begin date `management_modules.timestep.`**`timemeasure`**(*name*, *loops=0*, *update=False*, *sample=1*)

Measuring of the time for each subroutine

Parameters

- **`name`** – name of the subroutine
- **`loops`** – if it it called several times this is added to the name
- **`update`** –
- **`sample`** –

Returns add a string to the time measure string: timeMesString

`management_modules.timestep.`**`timestep_dynamic`**(*self*)

Dynamic part of setting the date Current date is increasing, checking if beginning of month, year

Returns a list of date variable in: dateVar

configuration module

Loading and parsing of the settings file

`class` `management_modules.configuration.`**`ExtParser`**(**args*, ***kwargs*)

Bases: `configparser.ConfigParser`

addition to the parser to replace placeholders

Example

PathRoot = C:/work MaskMap = $(FILE_PATHS:PathRoot)/data/areamaps/area.tif

`get`(*section*, *option*, *raw=False*, *vars=None*, ***kwargs*)

def get(self, section, option, raw=False, vars=None placeholder replacement

Parameters

- **`section`** – section part of the settings file
- **`option`** – option part of the settings file
- **`raw`** –
- **`vars`** –

Returns

`management_modules.configuration.`**`parse_configuration`**(*settingsFileName*)

Parse settings file

Parameters **`settingsFileName`** – name of the settings file

Returns parameters in list: binding, options in list: option

`management_modules.configuration.`**`read_metanetcdf`**(*metaxml*, *name*)

Read the metadata for netcdf output files unit, long name, standard name and additional information

Parameters

- **`metaxml`** – file mit information for netcdf files (metadata)
- **`name`** – file name information

Returns List with metadata information: metaNetcdfVar

management_modules.messages module

Error handling - giving out messages

exception management_modules.messages.**CWATMError**(*msg*)

Bases: Exception

The error handling class prints out an error

Parameters **Warning** – class CWATMError

Returns prints out a message about an error

exception management_modules.messages.**CWATMFileError**(*filename*, *msg=''*, *sname=''*)

Bases: *management_modules.messages.CWATMError*

The error handling class prints out an error

Parameters **Warning** – class CWATMError

Returns prints out a message about file error

exception management_modules.messages.**CWATMRunInfo**(*outputDir*, *Steps=1*, *ensMembers=1*, *Cores=1*)

Bases: Warning

prints out an error

Parameters **Warning** – class warning

Returns prints out a message

Warning warning given with a header and a message from the subroutine

exception management_modules.messages.**CWATMWarning**(*msg*)

Bases: Warning

the error handling class prints out an error

Parameters **Warning** – class warning

Returns prints out a message

Handling output of CWATM

management_modules.output module

class management_modules.output.**outputTssMap**(*out_variable*)

Bases: object

Class: Output of time series and map

dynamic(*ef=False*)

Dynamic part of the output module Output of maps and timeseries

Parameters **ef** – done with environmental flow

initial()

Initial part of the output module

management_modules.checks module

Checking maps if they fit in

`management_modules.checks.checkmap(*args, **kwargs)`

`management_modules.checks.counted(fn)`
count number of times a subroutine is called

Parameters `fn` –

Returns number of times the subroutine is called

Program management

Global definition of variables

globals module

Global definition of variables

`management_modules.globals.globalFlags(arg)`
Read flags - according to the flags the output is adjusted quiet,veryquiet, loud, checkfiles, noheader,printtime, warranty

Parameters `arg` – argument from calling cwatm

dynamicModel module

Framework of initial and dynamic modules

`class management_modules.dynamicModel.DynamicModel`
Bases: `object`

`i = 1`

`class management_modules.dynamicModel.ModelFrame(model, lastTimeStep=1, firstTimestep=1)`
Bases: `object`

Frame of the dynamic hydrological model

lastTimeStep: Last time step to run firstTimestep: Starting time step of the model

`run()`
Run the dynamic part of the model

Returns

-

replace_pcr module

Some pcr operation are done in numpy

`management_modules.replace_pcr.npareaaverage(values, areaclass)`
numpy area average procedure

Parameters

- **values** –
- **areaclass** –

Returns calculates the average area of a class

management_modules.replace_pcr.**npareamajority**(*values*, *areaclass*)

numpy area majority procedure

Parameters

- **values** –
- **areaclass** –

Returns calculates the majority of an area of a class

management_modules.replace_pcr.**npareamaximum**(*values*, *areaclass*)

numpy area maximum procedure

Parameters

- **values** –
- **areaclass** –

Returns calculates the maximum of an area of a class

management_modules.replace_pcr.**npareatotal**(*values*, *areaclass*)

numpy area total procedure

Parameters

- **values** –
- **areaclass** –

Returns calculates the total area of a class

- genindex

Python Module Index