hmc Documentation

Release 1.0.0

Fabio Delogu

Oct 14, 2022

CONTENTS:

CHAPTER

ONE

OVERVIEW

The **Hydrological Model Continuum**, commonly named **HMC**, is a python package part of **FloodPROOFS Modelling System**. The FloodPROODS framework is a modelling system designed to assist decision makers during the operational phases of flood forecasting, nowcasting, mitigation and monitoring in various-sized catchments.

The system is operational in real time in different basins on the whole Italian territory for the National Civil Protection Department and in some basins of Bolivia, Albania and Lebanon. It is also used for water management purposed by Italian hydro-power companies (Compagnia Valdostana delle Acque, CVA, ERG Power Generation).

1.1 Components

In the Flood-PROOFS forecasting chain, the hydrological running part is performed by using the HMC package [1]. Written in python3 language, HMC manages all aspects concerning the execution of the Continuum Model for the FloodPROOFS modelling system. This package is able to run model using different configurations and using different datasets (weather stations, satellite, nwp ...).

For running model, HMC provides a Python3 scripts able to check input data, run model and save outcome data; all the actions are configured by using a configuration file usually named namelist. Once that the namelist file is properly filled in all its parts, HMC model will use it to computing results (discharges, soil moisture, evapotranspiration, land surface temperature ...) according with the information set by the users.

Particularly, the HMC library is organized in different parts, which are summarized as follows:

- **Applications**: tools to configure procedures able to processing different type of data, usually written in python3 programming language and available in the apps folder;
- **Executables**: tools and utilities to run procedures, to download datasets or to interface HyDE package with different environments, usually written in bash or python3 programming languages and available in the bin folder;
- **Docs**: documents to describe HyDE package, applications and executables, usually written in reStructured-Text (rst) and available in the docs folder;
- **Codes**: generic and specific classes and methods used by the applications, to perform processing data and to save results in the formats commonly used in FloodPROOFs modelling system, are written in python3 and available in the src folder;
- **Utilities**: common functionality required by the previous components.

The structure of the package is reported above:

```
hmc-master
├── apps
├── bin
├── docs
├── src
│   ├── common
│   └── hmc
├── test
├── AUTHORS.rst
├── CHANGELOG.rst
├── LICENSE.rst
└── README.rst
```

All codes and case studies are freely available and users can be get them from our github repository [1].

1.2 Prerequisites

In order to use the HyDE, users are strongly raccomanted to control if the following characteristics, libraries and packages are available and correctly installed on their machine.

Usually, HyDE is installed on **Linux Debian/Ubuntu 64bit** environment and all dependencies must be installed in according with this operative system.

All codes, subroutines and scripts are developed using both **Python** (version 3 and greater) [2] and **GNU Bash** [3]. QGIS geographic information system (version 2.18 and greater) [4] is used to develop tools to organize, create and control static and dynamic datasets.

The libraries and the packages are mainly divided in three categories:

- python3 packages and applications;
- GNU Bash applications;
- other software and applications (Jupyter Notebook, Panoply, cdo, nco, ncview . . .)

1.2.1 Python3 libraries

The python3 standard library is not sufficient to correctly install all HyDE applications; for this reason some extra libraries are needed to guarantee all functionalities. To install all python3 libraries a bash script named "setup_fp_env_python.sh" is provided [5]; basically, the script calls a **miniconda** [6] installation that allow to get all needed libraries and install them into "$HOME/user/fp_libs_python/" folder. During the installation, a virtual environment named "fp_env_python" is created too.

The users have to clone the github repository:

```
>> git clone git@github.com:c-hydro/fp-envs.git
```

and run the following bash script:

```
>> ./setup_fp_env_python.sh
|[take a moment ... ]
```

Once all libraries are correctly installed and configurated, to activate "fp_env_python" by command-line is necessary to execute the following:

```
>> export PATH="$HOME/fp_libs_python/bin:$PATH"
>> source activate fp_env_python
```

By default, the "fp_env_python" environment is shown in parentheses () or brackets [] at the beginning of your command prompt:

```
(fp_env_python) >>
```

Activating the virtual enviroment permits to use a correct configuration and all applications of HyDE package will work properly.

1.2.2 Other software and applications

As previously said, to perform analysis or check results, users can use some external and freely available softwares and applications.

Some of these are reported in the following list:

- PanoplyJ Data Viewer [7]
- CDO - Climate Data Operators [8]
- NCO - NetCDF Operators [9]
- NCView: a netCDF visual browser [10]
- Jupyter notebook web application [11]

More information is available on the homepage of the different software.

1.3 Potential Users

The HyDE package has been released to enable different applications (for example local/regional scenario assessment) and further development by external users.

Potential users are anticipated to predominately be interested in the ability to run the system with local data (including scenario modelling) and to modify the system with new capabilities. The potential collaborators have expressed a range of potential goals for their use of the modelling system, including performing comparisons with existing models, tailoring the hydrological performance to specific land uses and cropping types.

Broadly speaking, there are four potential user categories of the FloodPROOFS modelling system:

- **Data user**: who accessing the model outputs through the Bureau's website.
- **Case study user**: who work to evaluate his/her case using data over a selected time period.
- **Applying users**: who would primarily be interested in applying the current model to a region of interest using localised and/or scenario data where available.
- **Contributor users**: who will extend the capabilities of the model with new research and coding (modify the system with new capabilities)

It is expected that the majority of early adopters of the HyDE package will be Applying users looking to apply the system with local data/scenarios, with more Contributor users adopting the system as it becomes well known and established.

1.4 Contribute

We are happy if you want to contribute. Please raise an issue explaining what is missing or if you find a bug. We will also gladly accept pull requests against our master branch for new features or bug fixes.

1.4.1 Development setup

For Development we also recommend a "conda" environment. You can create one including test dependencies and debugger by running

```
>> conda env create -n fp_env_dev -c <list_of_packages>
```

or alternatively using a file:

```
>> "conda env create -n fp_env_dev  -f <file_of_packages.yml>
```

This will create a new "fp_env_dev" environment which you can activate by using "source activate fp_env_dev".

1.4.2 Guidelines

If you want to contribute please follow these steps:

- Fork the HyDE package to your account
- Clone the repository, make sure you use "git clone –recursive" to also get the test data repository.
- make a new feature branch from the repository master branch
- Add your feature

- Please include tests for your contributions in one of the test directories. We use py.test so a simple function called "test_my_feature" is enough
- submit a pull request to our master branch

1.5 Reference

[1] CIMA Hydrology and Hydraulics GitHub Repository
[2] Python programming language
[3] GNU Bash
[4] QGIS project
[5] FloodPROOFS virtual environment tools
[6] Conda environment manager
[7] Panoply netCDF, HDF and GRIB Data Viewer
[8] CDO - Climate Data Operators
[9] NCO - NetCDF Operators
[10] NCView: a netCDF visual browser
[11] Jupyter notebook web application
[12] Cron jobs scheduler

CHAPTER

TWO

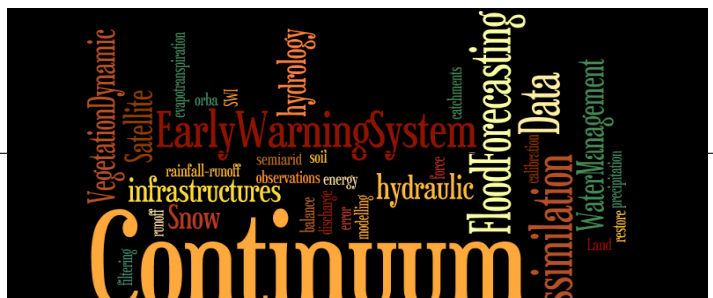
DESCRIPTION

2.1 Introduction

Continuum is a continuous distributed hydrological model, usually named HMC, that strongly relies on a morphological approach, based on a novel way for the drainage network components identification. The model is a compromise between models with a strong empirical connotation, which are easy to implement but far from reality, and complex physically based models which try to reproduce the hydrological processes with high detail but which introduce a hard parameterization and consequent uncertainty and lack of robust parameters identification.

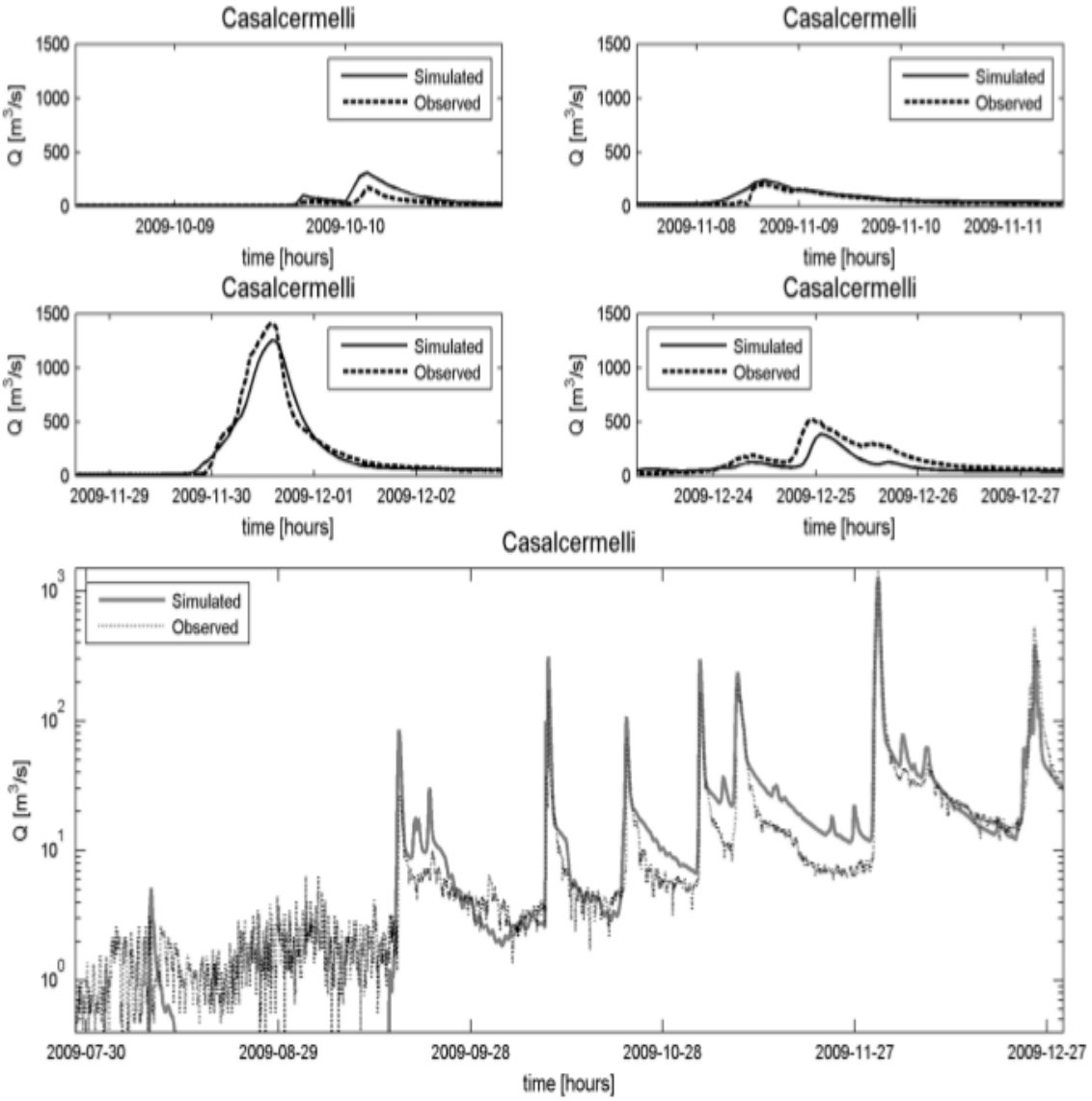
The HMC aims at an equilibrium between simplicity and rigorous physical modeling while maintaining comparable performances to existing models; the reduced complexity of the schematizations and the relatively small number of parameters leads to a considerably lower calibration effort, increasing model robustness and portability to data scarce environments. The resulting increased computational efficiency leads to an easier utilization of the model in ensemble mode so that a direct quantification of prediction uncertainties is possible. All the main hydrological phenomena are modeled in a distributed way. Nowadays satellite products of meteorological, hydrological and "vegetation" variables are becoming promising and widely available all over the world. Examples of meteorological satellite products are: TRMM evapotranspiration estimations, H-SAF soil moisture. These kind of data are now available for a high percentage of the earth's surface. This leads to a panorama where the possibility of running a hydrologic model using only satellite information is real. Along these lines develops one of the most important features of the HMC: it can be calibrated using only satellite data, e.g. surface temperature and/or soil moisture; results in reproducing observed discharge in this framework are quite promising. This makes the model suitable for application in data scarce environments.

The model is able



to
re-
pro-
duce
the
spatial-
temporal
evo-
lu-
tion
of
soil moisture, energy fluxes, surface soil temperature and evapotranspiration. Moreover, it can account for seasonal vegetation variability in terms of interception and evaporation. Deep flow and water table evolution are modelled with a simple scheme that reproduces the main physical characteristics of the process and a distributed interaction between water table and soil surface with a low level of parameterization. The introduction of the so-called "force-restore" equation for the surface energy balance allows the LST estimation and makes the model feasible to exploit remote sensing data. These latter can be used for calibration or, more appropriately, in a data assimilation framework. Referring to already tested calibration methodologies and making some basic assumptions, the calibration task can be reduced to just six parameters at catchment scale that are then spatially distributed by means of simple assumptions on the physical processes that they describe. Consequently, the parameter space is really small for a distributed continuous model, and HMC can be implemented with easily accessible data and territorial information (digital elevation model, basic soil and vegetation parameters). If more detailed territorial information were available, the parameterization methodology could be approached reducing the number of assumptions and linking the parameters more tightly to territorial characteristics. The sensitivity analysis has been carried out on five parameters: two parameters regulate the overland flow, the shape of the hydrograph and response time; two parameters are related to the soil characteristics and affect infiltration and relative soil humidity; and one parameter takes into account soil porosity and the ratio between vertical and horizontal saturated soil conductivity. A nice separation can be found between parameters in the fact that they influence quite distinct response of the model, with the result of further simplifying the calibration procedure. Standard calibration and validation based on streamflow data have been carried out on different periods and on different outlet sections (in terms of soil use, slope and response time). The model produces good results in terms of discharge. Further work is needed to introduce the modelling of the snow cover evolution and of the snowmelt in order to carry out multi-annual simulations where data are available. Model initialization influences the simulation for quite a long period; in particular the definition of initial water table level is a sensitive choice. The ideal situation is to extend as far as possible the warm-up period, including a long period without rainfall events.

The water table initialization methodology here is based on a possible spatial distribution of water table that exploits morphological constraints. Further validation analysis has been carried out comparing LST estimated by the model and satellite measurements at both pixel and basin scale. The results provide evidence that HMC reliably models the LST dynamics at various temporal scales, with some periods of overestimation, particularly during the warmer hours of summer. During the cold season the modelled LST has a lower variability with respect to the satellite estimates, but here the percentage of reliable data is quite scarce because of the more frequent cloud covering, and this makes the comparison more uncertain. The approach followed in the design of HMC proposes concentrating the efforts in augmenting the number of state variables that are predicted by the model and those that are also observables by using classical or remote instruments of measure. Specific attention is paid to distributed variables (e.g. LST fields) that offer very different information when compared to integral measures (e.g. discharge time series). The LST comparison showed potential for additional constraints to be used in the calibration phase or to be exploited in a more comprehensive assimilation framework. The distributed nature of the LST in comparison to traditional calibration time series (e.g. discharge data series) can add important information for a better estimation of state variables and parameter patterns. A demonstration of this potential is carried out by calibrating a sub-set of the parameters referring to LST satellite estimation and to morphologic information derived by the DEM. The results are comforting, and the proposed methodology led to a parameter set that well reproduces both satellite LST and streamflow observations, the latter used only in the validation phase. LST was successfully used to drive the calibration procedure. This was possible thanks to the model structure and the way parameters are treated and distributed in time and space. This could have a strong application impact in environments where reliable streamflow data are not available, given the worldwide availability of LST data.


Casalcermelli
Simulated
Observed
Q [m3/s]
time [hours]
2009-10-09
2009-10-10
Casalcermelli
2009-11-08
2009-11-09
2009-11-10
2009-11-11
Casalcermelli
2009-11-29
2009-11-30
2009-12-01
2009-12-02
Casalcermelli
2009-12-24
2009-12-25
2009-12-26
2009-12-27
Casalcermelli
2009-07-30
2009-08-29
2009-09-28
2009-10-28
2009-11-27
2009-12-27

2.2 Overland channel flow

The surface flow schematization distinguishes between channel and hillslope flow. In channels the momentum equation per unit of width is derived from the kinematic schematization with a nonlinear dependence between discharge and water velocity. The water depth for the i-th channel cell is evaluated combining the momentum equation and the mass balance equation:

$$\frac{dh_i}{dt} = I_i - \frac{1}{\Delta x} \cdot u_c \cdot \sqrt{tg(\beta_i)} \cdot h_i^{1.5}$$

where I_i represents the input per unit of area (the sumof runoff, saturation excess and inow discharge from upstream) to the grid cell [L T 1]. On the hillslopes the overland ow has a linear equation for the motion:

$$q = \Delta x \cdot u_h \cdot h_i$$

where u_h parameterizes the main morphologic characteristics of the hillslopes [T 1] (slope, roughness, etc.). The nal schematization is equivalent to a linear reservoir model. The parameters u_h and u_c need calibration at basin scale (i.e. one value for the entire catchment). In both hillslopes and channels, the re-infiltration process is accounted for: the input to the i-th cell must exceed its infiltration capacity; otherwise, it infiltrates the soil. Exfiltration is also possible.

2.3 Vegetation interception

Interception includes the portion of rainfall that is caught by tree leaves, grass and vegetation cover in general, and is evaporated before it touches the ground.

Ponding effects are also included in this initial abstraction. Interception is modelled by a simple empirical equation. A maximum retention capacity S_v is introduced, and it is estimated as a function of the leaf area index (LAI) by the relationship:

$$S_v = 0.95 + 0.5 \cdot LAI - 0.06 \cdot LAI^2$$

The water in the reservoir with capacity S_v is evaporated at the evaporation rate derived by the latent heat flux estimation without affecting the infiltration computation; the input is the precipitation. The advantage of using a LAI-dependent expression is that the model takes into account vegetation variability in space and time. LAI is usually updated every 15 days from satellite optical sensor data (e.g. from MODIS).

2.4 Infiltration and Subsurface Flow

The
in-
fil-
tration methodology is a modification of the Horton equation based on physically interpretable parameters. It accounts for soil moisture evolution even in condition of intermittent and low-intensity rainfall (namely lower than the infiltration capacity of the soil). The soil is schematized as a reservoir with capacity V_{max} [L], and a selective filter $g(t)$ [L T1] manages the inflow:

$$g(t) = f_0 + (f_1 - f_0) + \frac{V(t)}{V_{max}}$$

where f_0 is the maximum infiltration rate for completely dry soils and f_1 is the asymptotic minimum infiltration rate for saturated soils considered as a function of f_0:

$$f_1 = c_t \cdot f_0$$

The method proposed has been further modified by introducing the field capacity of the soil, defined as the water content that can be held by capillarity against the force of gravity:

$$V_{fc} = c_t \cdot V_{max}$$

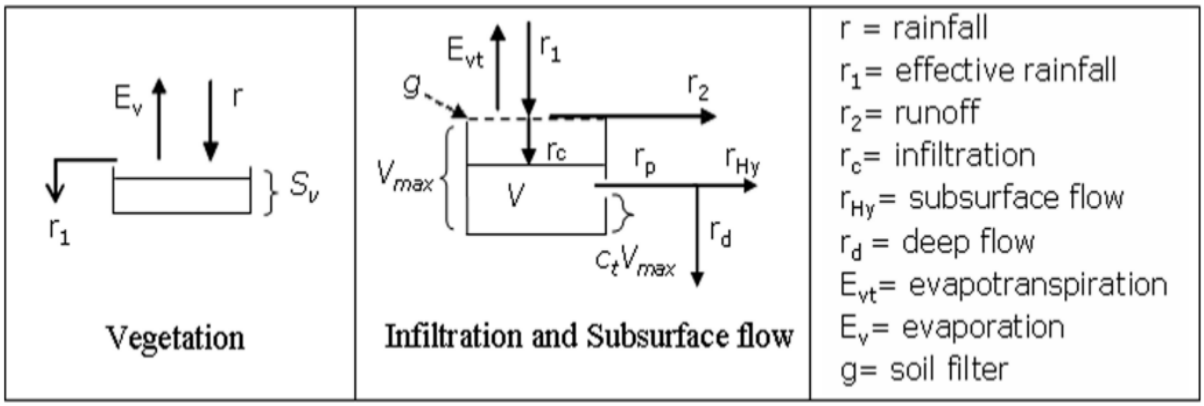
with the parameter c_t [0, 1]. In this configuration, the dynamic mass-balance equation for the soil can be written for each cell:

$$\frac{dV}{dt} = g(t) - r_p(t)$$

where:

$$r_p(t) = f_1 \cdot (V(t) - \frac{c_t \cdot V_{max}}{V_{max} \cdot (1 - c_t)})$$

The infiltration scheme has four parameters: the initial infiltration rate f_0, the maximum soil retention capacity V_{max}, and the parameters to define soil field capacity c_t and final infiltration rate c_f.



The parameters f_0 and V_{max} are related to the soil type and land use through the curve number (CN) parameter. They can be easily derived by soil use and soil type maps and they vary spatially in the catchment; c_t and c_f are calibration parameters and are assumed to be constant for the whole basin. In this way the pattern of f_1 and V_{fc} is spatially modulated by the pattern of V_{max}. The percolation rate separates into two components: a contribution to subsurface flow r_{Hy} and one to deep flow r_d or recharging water table defined as:

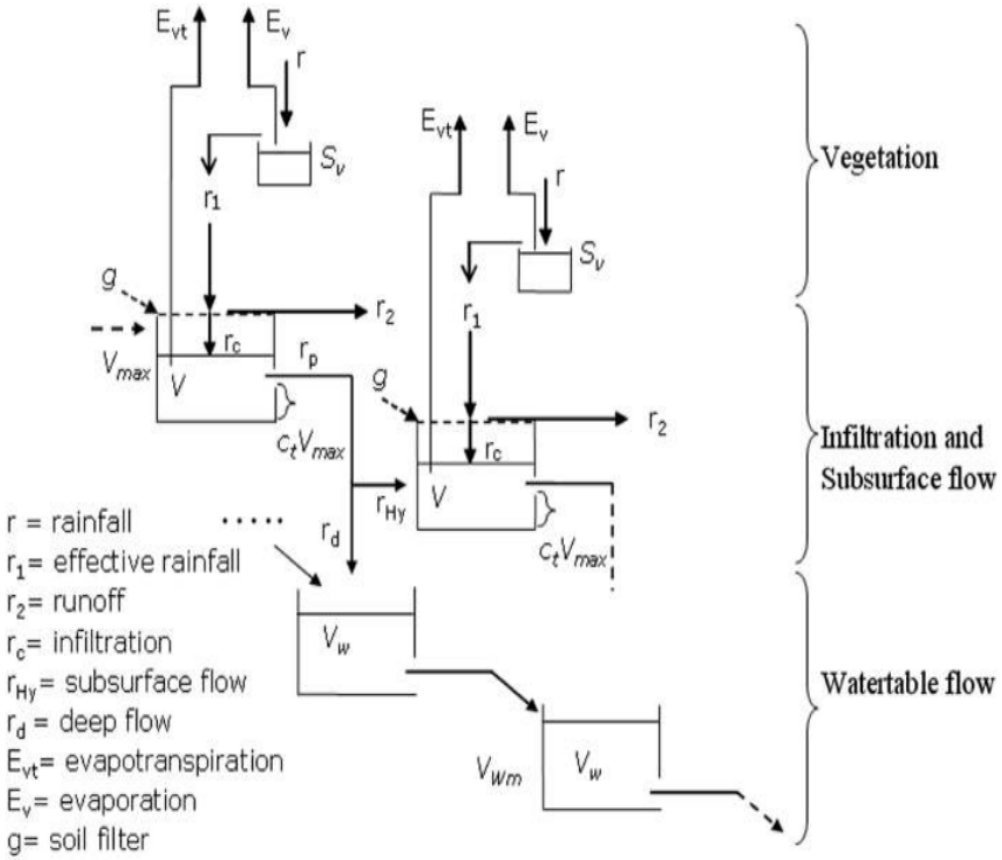
$$r_{Hy} = sin(\alpha) \cdot r_p(t)$$

$$r_d = (1 - sin(\alpha)) \cdot r_p(t)$$

where the angle α is such that $\tan(\alpha)$ is the downslope index; $sin(\alpha)$ is a decomposition term that increases with the terrain slope, reproducing the major proneness of the high slope areas to subsurface flow due to gravity. The downslope index seems to be less sensitive than local surface slope to changes in DEM resolution, and it is able to capture dominant controls on local drainage regimes, especially in cases where profile curvature exerts a strong control on the drainage pattern. The subsurface flow is propagated between cells following the surface drainage network directions, and the soil moisture state of each cell is updated by considering both the infiltration, estimated by the modified Horton method, and the inflow from the upper cells. Therefore a cell can reach saturation because of the percolation from upper cells causing saturation excess. The water in the reservoir with capacity S_v is evaporated at the evaporation rate derived by the latent heat flux estimation without affecting the infiltration computation; the input is the precipitation. The advantage of using a LAI-dependent expression is that the model takes into account vegetation variability in space and time.

2.5 Deep Flow and Water Table

Several approaches are possible to describe the dynamics of both the deep flow and the water table, with examples from Darcy's law applications to conceptual reservoir models. However, it is often difficult to have the data necessary for the correct implementation and parameterization of water table dynamics. In HMC the water table evolution is modelled with a simplified approach that maintains a physical and distributed description of the process.



Above all we are interested in the water table interaction with the subsurface flow and soil surface and in its effects on surface flow and soil moisture spatial pattern; the adopted scheme allows also the reproduction of the baseflow far from rainfall events with a parsimonious parameterization. The layer of soil containing the aquifer is schematized as a

unique homogeneous layer bounded by the lower impervious (bedrock) surface and the bottom of the root zone. The thickness of this layer is expressed in terms of maximum volume of water content of the aquifer, and it is estimated using the surface slope as a proxy. The maximum water content in every cell (i) of the basin is given by:

$$VW_{mi} = VW_{max} \cdot \frac{\tan(\alpha_{max}) - \tan(\alpha_i)}{\tan(\alpha_{max}) - \tan(\alpha_{min})}$$

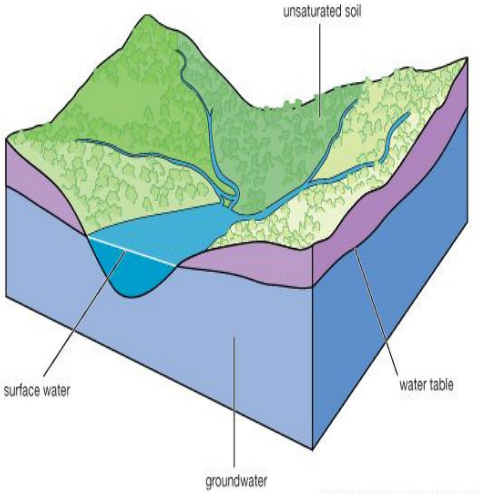
where VW_{max} is the absolute maximum water content of the aquifer on the whole investigated area; this sets a limit that is basically a calibration parameter. The reservoir is fed with rd (see previous section). The effect of porosity is considered as a multiplicative factor in the Darcy's equation used to estimate the flux per unit area between two contiguous cells (i and j):

$$q_{ij} = \frac{h_{Wi} - h_{Wj}}{\Delta x} \cdot R_f \cdot f_{1i}$$

where x is the DEM spatial resolution, f1i the final infiltration rate estimated, h_w the water table level and R_f a factor that also takes care of differentiating the saturated vertical and horizontal conductivity. Each cell can drain towards all the neighbouring cells following the 2-D water table gradient that depends on the elevation and on the water content of each cell. When the water table reaches the surface ($VW_i = VW_{mi}$), the deep percolation term is inhibited, while the condition $VW(t)0$ is a limit that can only be reached after a very long and anomalous dry period.

2.6 Energy Balance and Evapotranspiration

The representation of surface mass and energy turbulent fluxes requires the solution of a conservation equation for mass and energy driven by temperature and moisture content. Since the vertical gradient of such variables is quite large, a high-resolution multiple layer model would be required to estimate soil surface temperature and moisture content with accuracy. Such an approach demands substantial amounts of computing resources to solve the balance equations.



An alternative approach makes use of computationally efficient parameterization of soil heat and moisture flux terms. In other studies is showed that the heat flux into the soil could be parameterized by the sum of a temperature-derivative term and the difference between ground surface and deep soil temperature. This approach is known as the "force-restore" method, because the forcing by net radiation is modified by a restoring term that contains the deep soil temperature. Since then the "force-restore" method has been widely used in land surface modelling. Some studies demonstrated that the "force-restore" equation is the solution of the heat diffusion equation, with purely sinusoidal forcing assuming that the thermal properties are constant with depth and the surface forcing term is also nearly independent of air temperature and has a strong periodic behaviour in time. The HMC solves a complete and explicit energy balance at the interface between soil surface and atmosphere by using the "force-restore" approach for land surface temperature. Theoretically, the control volume to which the balance is applied is the unit area bounded vertically by the surface of the soil and the top of the canopy, assuming the thermal capacity of this volume

is negligible. The horizontal energy fluxes are neglected. In practice, the volume is extended to the unit cell of the numerical scheme used. This approximation is a fair trade-off between parsimony in parameterization and accuracy in the description of the processes.The conservation of energy at soil surface is given by:

$$G = R_n - H - LE$$

where R_n is the net radiation, H the sensible heat flux, LE the latent heat flux and G the ground flux (all [E t1 L2]). This latter term closes the budget, and it represents the heat propagated by diffusion towards the deep layers of the soil. The shortwave component of R_n is derived from radiometer observations when the density of observations is appropriate. Otherwise, it is estimated by combining the extraterrestrial component of the radiation computed, attenuated using meteorological variables and cloud cover. The terrain parameter characterizations that influence both direct and diffuse components of the radiation are computed. The longwave components are rarely available from observations, and they are therefore estimated using the Stefan–Boltzmann law as a function of air temperature and humidity. The daily cycle of LST has the implicit signature of the energy balance. Maximum amplitudes of LST diurnal cycle are usually reached in the presence of bare and dry soil. The presence of moisture on the surface and in the subsurface soil greatly moderates the daily range of LST. The vegetation cover has a similar effect. The "force-restore" approach leads to the following equation for LST:

$$\frac{dLST}{dt} = 2 \cdot \sqrt{\pi\omega}\frac{(R_n - H - LE)}{\varphi} - 2\pi\omega \cdot (LST - T_{deep})$$

where φ [E L2 T1 t(1/2)] is the effective thermal inertia and T_{deep} [T] is a "restoring" deep ground temperature. T_{deep} is evaluated by filtering data for air temperature at ground level; φ is the thermal inertia, and it is a function of conductivity, density and specific heat capacity of soil, and it is eventually related to soil moisture. The fluxes are estimated using bulk formulations. The equation input variables are commonly observed by ground-based micrometeorological networks. The soil parameters used in the estimation of the thermal inertia, usually constant at basin scale, can be estimated by a data assimilation process, or related to soil type when reliable maps are available. In HMC the evapotranspiration LE [m s1] is estimated as:

$$ET = \frac{LE}{\rho_w \lambda_{LE}}$$

where ρ_w [m L3] is the water density, and ET is deducted from the interception storage S_v if not empty, otherwise from the subsurface reservoir $V(t)$ adding the following:

$$\frac{dS_v}{dt} = ET$$

if S_v greater then 0 and:

$$\frac{dV}{dt} = ET$$

if S_v greater equal 0.

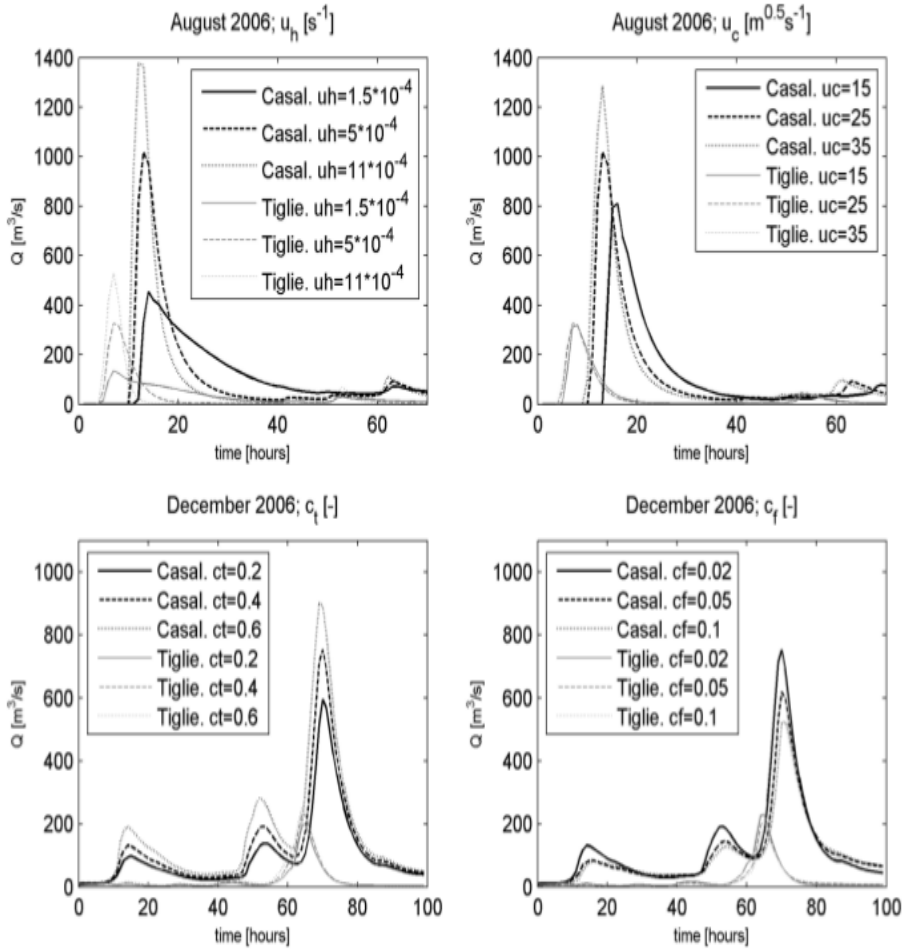
2.7 Parameters

Six model parameters need calibration on the basis of input–output time series: c_f, c_t, h_h, u_c, R_f, VW_{max}. The first two parameters c_f and c_t mainly rule the generation of runoff and the movement of water in the different soil layers, while u_h and u_c control the surface water motion. VW_{max} represents the maximum storage capacity of the aquifer, and R_f summarizes the effect of soil porosity as well as of the ratio between vertical and horizontal saturated soil conductivity. The range of variation of the parameters has been defined based on prior knowledge of the parameter meaning, which defines their mathematical and physical range of validity.

Table 1: Summary of the model parameters that need calibration with their brief description

Parameters	Units	Description
uh	[s-1]	Flow motion coefficient in hillslopes
uc	[m-0.5s-1]	Friction coefficient in channels
cf	[-]	Defines the infiltration capacity at saturation
ct	[-]	Defines the mean field capacity
Rf	[-]	Related to anisotropy between the vertical and horizontal saturated conductivity, and to porosity
VWmax	[mm]	Maximum water capacity of the aquifer on the whole investigated area

The parameters u_c and u_h impact the water flow on the surface. High values of these two parameters lead to narrow and highly peaked hydrographs; u_h has influence on the general shape of the hydrograph while u_c has an increasing influence with the increasing length of the channelled paths (e.g. large/elongated basins). It modifies the peak flow value as well as the peak arrival time. The impact estimation of parameters u_h and u_c has been made considering a short period of simulation (16 to 18 August 2006) since they influence directly the overland and channel flow. The first subplot in the following figure shows that u_h has a considerable influence on both the Tiglieto and Casalcermelli outlets.



The peak values and the hydrograph shape have quite a large range of variation, while peak times are not significantly affected by this parameter. The second subplot shows the influence of u_c. It mainly affects the shape and the peak times on the Casalcermelli outlet section, while hydrographs of the Tiglieto outlet show negligible differences. Note that Casalcermelli has a drainage area that is one order of magnitude larger in respect to Tiglieto. The parameter c_t is related to the soil field capacity and defines the fraction of water volume in the soil not available for percolation and subsurface flow. It has an impact on the dynamics of soil saturation between rain events: higher values of c_t reduce the soil drying time scale especially during the cold season, with consequently higher runoff coefficients for single rainfall events. However, the subsurface flow tends to vanish rapidly, because water level drops easily under the field capacity. The parameter c_f controls both the velocity of subsurface flow and the dynamics of saturation of the single cells. Low values of c_f (i.e. low values saturated hydraulic conductivity) tend to cause the rapid saturation during rainfall events associated with slow subsurface flow increasing runoff production. Higher values of c_f produce a rapid subsurface flow with saturated areas that quickly concentrate along the drainage network. The combination of the two soil parameters c_t and c_f controls the distribution of the volumes of soil and surface water in space and time, and it impacts soil humidity propagation; c_t and c_f influence the mass balance over long periods and regulate the exchanges between subsurface flow and runoff. The third and fourth panels in the previuous figure show how they affect the tails of the hydrographs and the values of peak flows in the period between 7 and 10 December 2006. The effect of the combination of these two parameters is quite complex, and it is only partially represented in the figures. They must be calibrated over long periods of time using, at best, external soil information when available.

The parameter R_f regulates the response of the deep flow and mainly influences low flow regimes, while for larger basins it also affects high discharges. In the figure below the period between 14 and 17 September 2006 is shown. The effects of R_f on the Tiglieto outlet are negligible during the flood while the influence on low flows is more relevant for both the outlet sections. Particular remarks need to be made about VW_{max} – a measure of the capacity of the basin for storing water in its aquifer and deep soil layer. All these simulations highlighted another important feature of the model.

Because of its internal structure, similarly to other complete hydrological models, it is possible to map different processes and therefore different parts of the hydrograph, onto the parameters, so that different parts of the hydrograph time series can be used separately to better identify model parameter values. Further analysis is needed to show sensitivity to spatial and temporal resolution.

It is not easy to define a value for VW_{max} that reproduces a correct or realistic distribution of the deep soil layer water storage throughout the basin due to the fact that this distribution is hard to observe. Tests made using different values of VW_{max} in a physically acceptable range and starting from the same initial condition show that the model has low sensitivity to this parameter when the period of simulation covers between 6–12 months. This is related to the slow temporal dynamic of the water table.

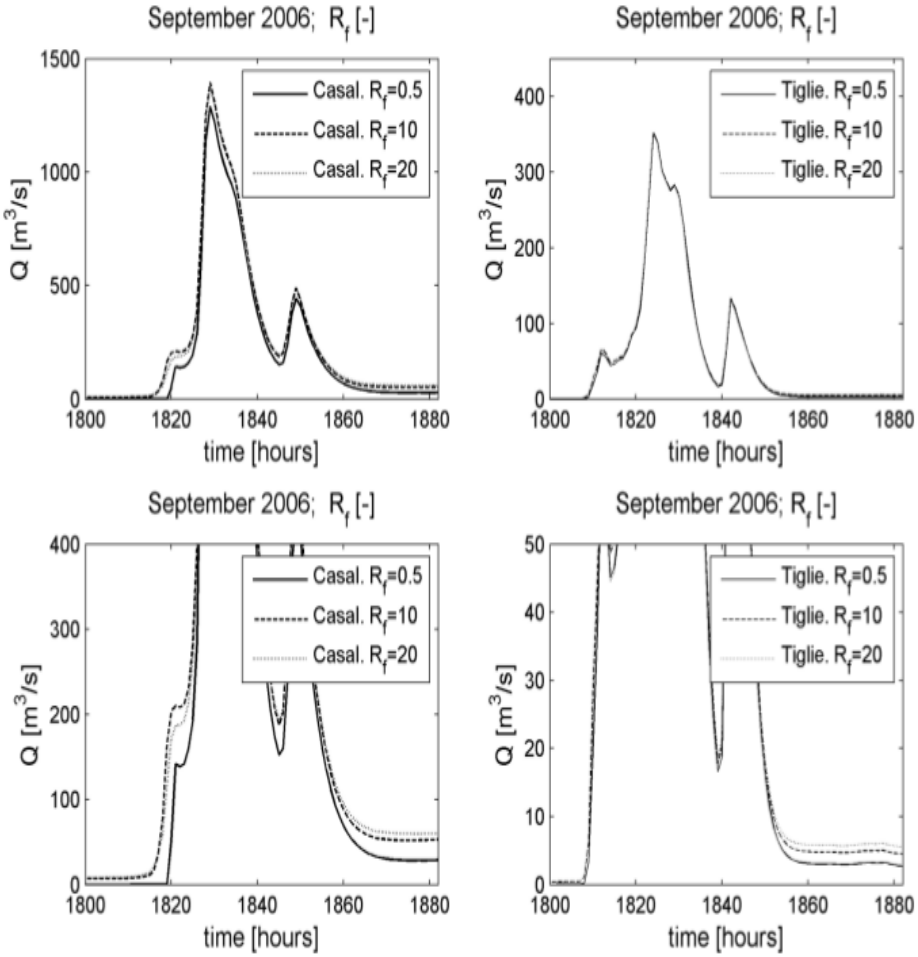
If data series for very long simulations (many years) are available, the parameter VW_{max} can be re-calibrated and adjusted. In the adopted scheme the initialization of the related state variable $VW(t)$ is more important than its upper limit. In fact, practice demonstrates that the definition of the water table initial condition $VW(t = 0)$ evidently influences simulated discharge. A reasonable initial condition produces a rapid stabilization of the water table with dynamics driven by the water input from upper soil layer. Two considerations are made in order to define these:

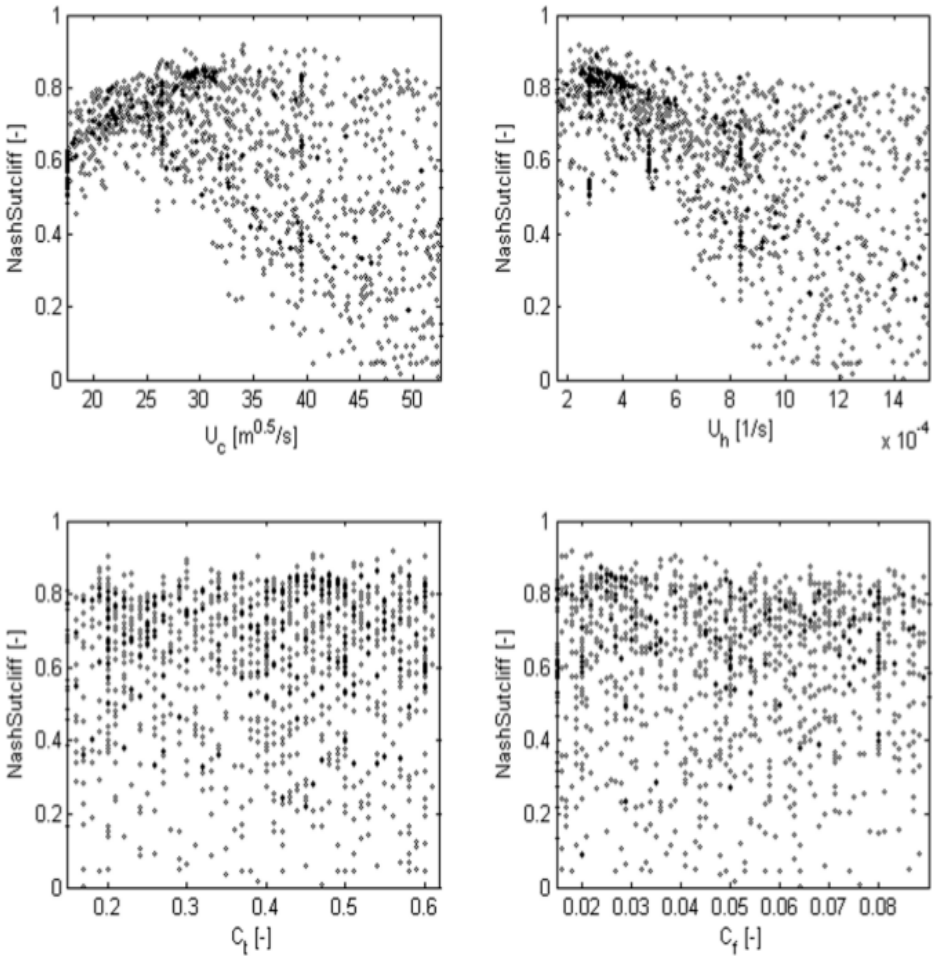
- in correspondence with the drainage network, the water table is generally next to soil surface, because it is continuously recharged by the upstream portions of catchment;
- the mountainous parts of the water table receive reduced contribution, because they drain small areas and are characterized by high gradients, and here the water table has lower levels.

Based on these considerations, water table initial conditions are set as follows; $VW(t = 0)$, in correspondence with channels, is set close to VW_{max}. In the hillslopes the level of $VW(t = 0)$ is estimated supposing it is inversely proportional to the downslope index α.

In order to carry out a basic sensitivity analysis, we considered what appear to be the most sensitive parameters (u_c, u_h, c_t, c_f) and a set of 2000 model runs has been generated using a Monte Carlo approach, sampling the parameters from a uniform distribution bounded by the parameter domain.

The runs have been carried out on a sub-period of the calibration period where two major flood events occurred (July


September 2006; R_f [-]
Casal. R_f=0.5
Casal. R_f=10
Casal. R_f=20
Q [m^3/s]
time [hours]
September 2006; R_f [-]
Tiglie. R_f=0.5
Tiglie. R_f=10
Tiglie. R_f=20
Q [m^3/s]
time [hours]
September 2006; R_f [-]
Casal. R_f=0.5
Casal. R_f=10
Casal. R_f=20
Q [m^3/s]
time [hours]
September 2006; R_f [-]
Tiglie. R_f=0.5
Tiglie. R_f=10
Tiglie. R_f=20
Q [m^3/s]
time [hours]


NashSutcliff [-]
U_c [m^0.5/s]
U_h [1/s]
x 10^-4
C_t [-]
C_f [-]

– September 2006). To show the results a dot plot representation has been drawn using the Nash–Sutcliffe coefficient (NS) as skill estimator. From figure above, it is possible to clearly identify a behavioural domain for u_h and u_c, while a lot of uncertainty in the streamflow simulation due to c_f and c_t is present. This could be due to the very nonlinear relationship that connects these last two parameters with streamflow.

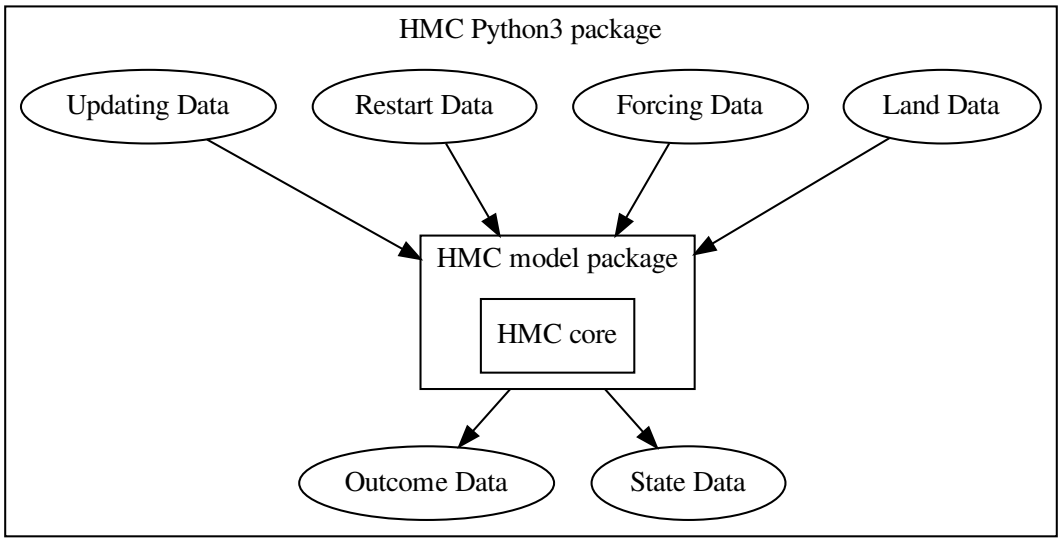
Parameter	Unit	Min	Max	ΔPar
c_t	[−]	0.15	0.65	0.01
c_f	[−]	0.01	0.1	0.01
u_c	$m^{0.5} s^{-1}$	15	60	1
u_h	s^{-1}	0.0002	0.0015	0.0001
R_f	[−]	0.5	50	0.5

CHAPTER

THREE

APPLICATIONS

The **HMC python3 package** is a library to wrap the HMC model into a python environment; generally, the python part performs all the actions that are needed to configure and run the model. All the model codes are available in the **HMC model package**; the routines about the model computational part are written in Fortran2003 programming language and they are linked and controlled by the python library. The workflow between **HMC python3 package** and **HMC model package** is reported below:



Generally, the applications for running HMC, available in the HMC python3 package, are able to:

- prepare the HMC namelist file;
- collect and control HMC land data;
- collect and control HMC forcing, restart and updating data;
- run HMC using namelist, land and forcing data;
- save and organize HMC state and outcome data.

The running applications require two default arguments to set model and time information:

1. -settings_file

2. -time

The first argument [-settings_file], for running the HMC model, is a **configuration file**; it is mandatory to run the HMC model because all the information, needed by the HMC simulations, are set in this configuration file; the second argoment [-time] is the **model simulation time** for applying model in a selected date.

The structure of HMC python3 package is shown below.

```
hmc-master
├── **apps**
│   ├── HMC_Model_Run_Manager.py
│   ├── configuration_run_app_1.json
│   ├── configuration_run_app_2.json
│   ├── configuration_data_app_1.json
│   ├── configuration_data_app_2.json
│   └── ...
├── bin
│   ├── local
│   ├── server
│   └── ...
├── docs
├── hmc
├── AUTHORS.rst
├── CHANGELOG.rst
├── LICENSE.rst
└── README.rst
```

The application for running HMC model and all its part is named "HMC_Model_Run_Manager.py" and, in the following line, the generic usage of this procedure is reported:

```
>> python3 HMC_Model_RUN_Manager.py -settings_file configuration.json -time "yyyymmddHHMM
↪"
```

3.1 Configuration file

As previously said, the HMC python3 package needs two arguments to run the HMC model over a domain during a selected period, The first arguments [-settings_file] is a **configuration file** in JSON format, usually named **hmc_configuration_{domain}_{run}.json**; this file is used to set all information needed by the model. The second arguments [-time] is the **model simulation time** in 'yyyymmddHHMM' format.

The available sections in the configuration file are: * General Info * Parameters Info * Geographical Info

The **General Info** section is used to store information about the version and the release date of the model, the conventions used by input/output files, the authors and their contacts, the reference project and so on. An example of GeneralInfo data structure is reported in the following block.

```
{
  "Conventions": "CF-1.7",
  "title": "HMC Run Manager",
  "institution": "CIMA Research Foundation - www.cimafoundation.org",
  "website": "",
  "source": "",
  "history": "Python settings algorithm for running hydrological model Continuum (HMC)",
  "references": "http://cf-pcmdi.llnl.gov/ ; http://cf-pcmdi.llnl.gov/documents/cf-
```

(continues on next page)

(continued from previous page)

```
↪standard-names/ecmwf-grib-mapping",
  "authors": "Fabio Delogu",
  "email": "",
  "project": "HMC Project",
  "version": "2.0.7",
  "date": "20180521"
}
```

The **Parameters Info** section is used to store information about the model configuration. This part present different subsections that users have to fill before running the HMC model.

- **Run_Params**

  In this subsection, users have to decide information about the run type:

  - RunDomain is the reference name for the domain.
  - RunName is the reference name for the run.
  - RunMode is the configuration of the run (deterministic or probabilistic); the flag EnsMode should be set using a boolean operator equal to "false" for deterministic run and equal to "true" for probabilistic run. If probabilistic mode is activated, the name of variable, used by the ensemble, and its limits have to be set.

  In the example, a nwp probabilistic run of 30 ensembles over the Italy domain is reported.

```
{
"RunDomain": "italy",
"RunName": "italy_nwp_probabilistic",
"RunMode":
        {"EnsMode": true,
         "EnsVar": {"VarName": "RFarm_ID", "VarMin": 1, "VarMax": 30,
↪"VarStep": 1} }
}
```

- **Run_VarExec**

  In this subsection, users have to fill the fields for setting model executable. Particularly:

  - RunModelExec is the name of the model executable (where $RUN is the name of the run).
  - RunModelNamelist is the name of model namelist (where $DOMAIN is the name of domain).
  - RunModelCLine is the command line to execute the model and $UC, $UH, $CT, $CF, $DOMAIN, $CPI, $KSATRATIO, $WTABLEHBR, $SLOPEMAX are the parameters requested by the model.

  In the example, generic executable and namelist files are set in configuration file.

```
{
"RunModelExec": "HMC_Model_V2_$RUN.x",
"RunModelNamelist": "$DOMAIN.info.txt",
"RunModelCLine": "$UC $UH $CT $CF $DOMAIN $CPI $KSATRATIO $WTABLEHBR
↪$SLOPEMAX"
}
```

- **Run_ConfigFile**

  In this subsection, users have to decide the following arguments:

  - FileData is the configuration file of the datasets that will be used by the model execution.

  - FileLog is the logging file that will be used by both HMC python3 package and HMC model package for saving log and debug information.

In the example, the configurations files are defined for a probabilistic run of Italy domain.

```
{
"FileData": "hmc_configuration_data_italy_nwp_probabilistic.json",
"FileLog": "hmc_logging_italy_nwp_probabilistic.txt"
}
```

- **Run_Path**

  In this subsection, users have to set useful paths needed by the model execution.

  - PathTemp is the temporary path.
  - PathCache is the cache path.
  - PathExec is the execution path (where copy executable and namelist files).
  - PathLibrary is the library path where HMC Fortran2003 codes are build.

  The example shows how to organize a model execution using a root path named "run"; all the pahs are generic and use $RUN and $MODE to specify the running paths.

```
{
"PathTemp": "/run/$RUN/temp/$MODE/",
"PathCache": "/run/$RUN/cache/$MODE/$yyyy/$mm/$dd/$HH/",
"PathExec": "/run/$RUN/exec/$MODE/",
"PathLibrary": "/library/"
}
```

- **Time_Params**

  In this subsection, users have to set the information about time. The meaning of each field is defined as follows:

  - TimeNow is the simulation time in "yyyymmddHHMM"; if the value is set to "null", TimeNow is defined according with the machine time.
  - TimeDelta is the time frequency of data (forcing, updating, restart, state and outcome) [seconds]
  - TimeStepObs is the running period of observed data steps.
  - TimeStepFor is the running period of forecast data steps.
  - TimeStepCheck is the checking period of observed data steps (in operational mode).
  - TimeRestart is used to set the restart time; starting from the previous "RestartStep" value until the time step hour is not equal to "RestartHH" value.
  - TimeWorldRef is defined by two values: 'gmt' or 'local'.
  - TimeTcMax is the maximum corrivation time; if the parameter is set to "-9999", the value is automatically computed as function of Digital Terrain Model static layer.

  In the example above, a configuration of operational run with data each 3600 [seconds] is reported; the execution takes care both the observed part (10 steps) and the forecast part (36 steps); the period for checking data is set (4 steps) and the restart part is configured (at least 24 steps in the past at 00.00). The reference time is "gmt" and the corrivation time is calculated using the information of the Digital Terrain Model.

```
{
"TimeNow": null,
```

(continues on next page)

(continued from previous page)

```
"TimeDelta": 3600,
"TimeStepObs": 10,
"TimeStepFor": 36,
"TimeStepCheck": 4,
"TimeRestart": {"RestartStep": 24, "RestartHH": "00"},
"TimeWorldRef": {"RefType": "gmt", "RefLoad": 0, "RefSave": 0},
"TimeTcMax": -9999
}
```

- **HMC_Params**

  In this subsection, users have to set the parameters of the HMC model; these values are avarage over the domain; in case of a distributed file for Ct, Cf, Uc and Uh is available the value will be overwritten during the model simulation. For further information of the meaning and validity range of each parameter see the *description* section.

  In the following example, the default value for each parameter.

```
{
"Ct": 0.5,
"Cf": 0.02,
"Uc": 20,
"Uh": 1.5,
"CPI": 0.3,
"KSatRatio": 1,
"WTableHbr": 500,
"SlopeMax": 70
}
```

- **HMC_Flag**

  In this subsection, users have to decide which physics or conditions have to be activated or not. The meaning of each of them is defined as follows:

  - **Flag_OS**
    - 1 Windows machine
    - 10 Linux Debian/Ubuntu machine
  - **Flag_Restart**
    - 1 to activate restart conditions.
    - 0 to deactivate restart condition (default starting condition).
  - **Flag_FlowDeep**
    - 1 to activate the flow deep flow
    - 0 to deactivate the flow deep flow
  - **Flag_DtPhysConv**
    - 1 to use a dynamic integration convolution step
    - 0 to use a static integration convolution step
  - **Flag_Snow**
    - 1 to activate snow physics

        * 0 to deactivate snow physics
    - **Flag_Snow_Assim**
        * 1 to activate assimilation of snow variables
        * 0 to deactivate assimilation of snow variables
    - **Flag_SM_Assim**
        * 1 to activate assimilation of soil moisture variable
        * 0 to deactivate assimilation of soil moisture variable
    - **Flag_LAI**
        * 1 to use a LAI datasets
        * 0 to use an empiric relationship
    - **Flag_Albedo**
        * 1 to use a dynamic monthly values
        * 0 to use a static value
    - **Flag_CoeffRes**
        * 1 to use an empiric relationship
        * 0 to not use an empiric relationship (null)
    - **Flag_WS**
        * 1 to use the water sources mode
        * 0 to not use the water sources mode
    - **Flag_ReleaseMass**
        * 1 to use the mass balance control
        * 0 to not use the mass balance control
    - **Flag_DebugSet**
        * 1 to activate debugging mode
        * 0 to deactivate debugging mode
    - **Flag_DebugLevel**
        * 0 to set debugging mode to BASIC info
        * 1 to set debugging mode to MAIN info
        * 2 to set debugging mode to VERBOS info
        * 3 to set debugging mode to EXTRA info

An example of flags configuration is reported below; in this case, the assimilation of soil moisture variable and the snow physics are activated. The debug is set to 0 for running in operational mode without extra information in logging file.

```
{
"Flag_OS": 10,
"Flag_Restart": 1,
"Flag_FlowDeep": 1,
```

(continues on next page)

(continued from previous page)

```
"Flag_DtPhysConv": 1,
"Flag_Snow": 1,
"Flag_Snow_Assim": 0,
"Flag_SM_Assim": 1,
"Flag_DebugSet": 0,
"Flag_DebugLevel": 3,
"Flag_CoeffRes": 0,
"Flag_WS": 0,
"Flag_ReleaseMass": 1,
"Flag_LAI": 0,
"Flag_Albedo": 0
}
```

- **HMC_Dt**

  In this subsection, users have to set the dt of the model. The meaning of the parameters is reported below:

  - Dt_Model is used to set the model step [seconds]
  - Dt_PhysConv is used to set the convolution integration step [seconds]

  For example in the following block, the Dt of the model is set to 3600 [seconds] and the convolution integration step is 50 [seconds].

```
{
"Dt_Model": 3600,
"Dt_PhysConv": 50
}
```

- **HMC_Data**

  In this subsection, users have to set data forcing attributes if data used by the model are in binary format (old versions). In the following list, the meaning of each variable is shown:

  - ForcingGridFactor is equal to 10, 100 or 1000 if binary data were saved, using a scale factor, in integer type
  - ForcingGeo is the LowerLeft corner of the reference Digital Terrain Model
  - ForcingRes is the resolution of the reference Digital Terrain Model
  - ForcingDataThr is the minimum threshold of valid data used during the model running [%]

The example above reported a condition with the data are not in binary format. The minimum threshold of valid data is set to 95 [%].

```
{
"ForcingGridSwitch": 0,
"ForcingScaleFactor": -9999,
"ForcingGeo": [-9999.0, -9999.0],
"ForcingRes": [-9999.0, -9999.0],
"ForcingDims": [-9999.0, -9999.0],
"ForcingDataThr": 95
}
```

The **GeoSysten Info** section is used to store information about the geographical reference of the data. Usually, in HMC model the reference epsg is 4326 (WGS 84 – WGS84 - World Geodetic System 1984).

In the example, all the information needed by the model to correctly georeferecing data in the epsg 4326 system.

```
{
"epsg_code": 4326,
"grid_mapping_name": "latitude_longitude",
"longitude_of_prime_meridian": 0.0,
"semi_major_axis": 6378137.0,
"inverse_flattening": 298.257223563
}
```

3.1.1 Model settings

3.1.2 Data settings

The data settings file is divided in different sections to set all the datasets needed by the model for performing simulations. The available sections are reported below:

- Data Land
- Data Forcing
- Data Updating
- Data Outcome
- Data State
- Data Restart

In all section we can defined name, location, type, time resolution and variables of each datasets.

Data Land

The Data Land part is used to specify the features of static data needed by HMC to initilize physics and parameters variables. Two data flags are set for configuring algorithm:

- AlgCheck: to check if file is available into data folder;
- AlgReq: to set if a variable is mandatory (true) or ancillary (false).

It is possible to specify two type of datasets; Gridded datasets are in ASCII format raster, Point datasets are defined using both a shapefile (section) and a generic ASCII file (dam, intake, joint, lake).

- **Gridded**
  - Alpha –> angle alpha map for defining watertable
  - Beta –> angle beta map for defining watertable
  - Cf –> Cf parameter map
  - Ct –> Ct parameter map
  - Uh –> Uh parameter map
  - Uc –> Uc parameter map
  - Drainage_Area –> drained area map [-]
  - Channels_Distinction –> hills and channels map [0, 1]
  - Cell_Area –> cell area map [km^2]
  - Coeff_Resolution –> coefficient resolution map [-]

  - Flow_Directions –> flow directions map [1, 2, 3, 4, 6, 7, 8, 9]
  - Partial_Distance –> partial distance map [-]
  - Vegetation_IA –> vegetation retention map [-]
  - Vegetation_Type –> curve number [-]
  - Terrain –> digital terrain model [m]
  - Mask –> domain mask [0, 1]
  - WaterSource –> water sources map [-]
  - Nature –> nature map [0, 100]

Gridded data are saved in ASCII format raster; the file begins with header information that defines the properties of the raster such as the cell size, the number of rows and columns, and the coordinates of the origin of the raster. The header information is followed by cell value information specified in space-delimited row-major order, with each row seperated by a carraige return. In order to convert an ASCII file to a raster, the data must be in this same format. The parameters in the header part of the file must match correctly with the structure of the data values. The basic structure of the ASCII raster has the header information at the beginning of the file followed by the cell value data.

- **Point**
  - Dam –> to define index positions and features of dam(s)
  - Intake –> to define index positions and features of intake(s)
  - Joint –> to define index positions and features of joint(s)
  - Lake –> to define index positions and features of lake(s)
  - Section –> to define index positions of outlet section(s)

```
{
"Gridded"     : {
  "FileName"    : "$DOMAIN.$VAR.txt",
  "FilePath"    : "$HOME/hmc-ws/data/static/land/",
  "FileType"    : 1,
  "FileTimeRes" : null,
  "FileVars"    : {
    "Alpha"                 : {"Name": "alpha",            "AlgCheck": true,   "AlgReq
↪": false },
    "Beta"                  : {"Name": "beta",             "AlgCheck": true,   "AlgReq
↪": false },
    "Cf"                    : {"Name": "cf",               "AlgCheck": true,   "AlgReq
↪": false },
    "Ct"                    : {"Name": "ct",               "AlgCheck": true,   "AlgReq
↪": false },
    "Uh"                    : {"Name": "uh",               "AlgCheck": true,   "AlgReq
↪": false },
    "Uc"                    : {"Name": "uc",               "AlgCheck": true,   "AlgReq
↪": false },
    "Drainage_Area"         : {"Name": "area",             "AlgCheck": true,   "AlgReq
↪": false },
    "Channels_Distinction"  : {"Name": "choice",           "AlgCheck": true,   "AlgReq
↪": false },
    "Cell_Area"             : {"Name": "areacell",         "AlgCheck": true,   "AlgReq
↪": false },
```

(continues on next page)

(continued from previous page)

```
    "Coeff_Resolution"      : {"Name": "coeffres",          "AlgCheck": true,   "AlgReq
↪": false },
    "Flow_Directions"       : {"Name": "pnt",               "AlgCheck": true,   "AlgReq
↪": false },
    "Partial_Distance"      : {"Name": "partial_distance",  "AlgCheck": true,   "AlgReq
↪": false },
    "Vegetation_IA"         : {"Name": "ia",                "AlgCheck": true,   "AlgReq
↪": false },
    "Vegetation_Type"       : {"Name": "cn",                "AlgCheck": true,   "AlgReq
↪": true  },
    "Terrain"               : {"Name": "dem",               "AlgCheck": true,   "AlgReq
↪": true  },
    "Mask"                  : {"Name": "mask",              "AlgCheck": true,   "AlgReq
↪": false },
    "WaterSource"           : {"Name": "ws",                "AlgCheck": true,   "AlgReq
↪": false },
    "Nature"                : {"Name": "nature",            "AlgCheck": true,   "AlgReq
↪": false }
  }
},
"Point": {
  "FileName"    : "$DOMAIN.info_$VAR.txt",
  "FilePath"    : "$HOME/hmc-ws/data/static/point/",
  "FileType"    : 1,
  "FileTimeRes" : null,
  "FileVars"    : {
    "Dam"                   : {"Name": "dam",               "AlgCheck": true,   "AlgReq
↪": true  },
    "Intake"                : {"Name": "intake",            "AlgCheck": true,   "AlgReq
↪": true  },
    "Joint"                 : {"Name": "joint",             "AlgCheck": true,   "AlgReq
↪": false },
    "Lake"                  : {"Name": "lake",              "AlgCheck": true,   "AlgReq
↪": false },
    "Section"               : {"Name": "section",           "AlgCheck": true,   "AlgReq
↪": true  }
  }
}
```

example of file(s) here

Data Forcing

The Data Forcing part is used to specify the features of the forcing data needed by HMC to properly run, compute physics variables and obtain related results. Usually, the forcing data used by the model can be divided in three main groups:

- Gridded
- Point
- Time Series

For each group, some features have to be set to configure its generic part:

- FileName: generic name of the forcing data file(s) [example: "hmc.forcing-grid.$yyyy$mmddHH$MM.nc"].
- FilePath: generic path of the forcing data file(s) [example: "$HOME/hmc-ws/run/$RUN/data/forcing/$MODE/gridded/$yyyy/$mm/$dd/"].
- FileType: flag for defining the format of the forcing data file(s) in netCDF format [2].
- FileTimeRes: resolution time of the forcing data file(s) in seconds [example: 3600].
- FileVars: variables available in the specified file(s).

The variables included in each group are defined by the attributes in the "FileVars" section; these attributes are used by the model routines to get and use data, read related information and prevent unexpected failures.

- Gridded

The forcing gridded files are in netCDF format; the forcing data are divided in two different types of variables:

- OBS: to define the observed dataset (if needed).
- FOR: to define the forecast dataset (if needed).

Each variable, in OBS and FOR section, is defined by a list of attributes in order to correctly parser the information; the attributes available are reported below:

- VarResolution: resolution time of the variable in seconds [example: 3600].
- **VarArrival: arrival time of the variable.**
  - Day: to define the variable arrival day [example: 0].
  - Hour: to define the variable arrival hour [example: null].
- **VarOp: ancillary defined operations of the variable.**
  - Merging: to merge variable from different sources in a common dataset [example: true].
  - Splitting: to split variable in different time steps [example: false].
- VarStep: step of the variable [example: 1].
- **VarDims: dimensions of the variable.**
  - X: dimension along X axis [example: "west_east"].
  - Y: dimension along Y axis [example: "south_north"].
  - Time: dimension along T axis [example: "time"].
- **VarName: name of the variable.**
  - FileName: the name of the source file [example: "ws.db.$yyyy$mmddHH$MM.nc.gz"]
  - FilePath: the path of the source file [example: "$HOME/hmc-ws/data/dynamic/outcome/observation/ws/$yyyy/$mm/$dd/"]
  - FileVar: the name of the variable in the source file [example: "Rain"]

An example of the forcing gridded data structure is reported below.

```
{
"Gridded"     : {
  "FileName"    : "hmc.forcing-grid.$yyyy$mm$dd$HH$MM.nc",
  "FilePath"    : "$HOME/hmc-ws/run/$RUN/data/forcing/$MODE/gridded/$yyyy/$mm/$dd/",
  "FileType"    : 2,
  "FileTimeRes" : 3600,
  "FileVars"    : {
```

(continues on next page)

(continued from previous page)

```
    "OBS"          : {
      "VarResolution"    : 3600,
      "VarArrival"       : {"Day": 0, "Hour": null},
      "VarOp"            : {"Merging": true, "Splitting": false},
      "VarStep"          : 1,
      "VarDims"          : {"X": "west_east", "Y": "south_north"},
      "VarName"          : {
        "Rain"             : {
          "FileName"           : "ws.db.$yyyy$mm$dd$HH$MM.nc.gz",
          "FilePath"           : "$HOME/hmc-ws/data/dynamic/outcome/observation/ws/$yyyy/
↪$mm/$dd/",
          "FileVar"            : "Rain"
                               },
        "AirTemperature"   : {
          "FileName"           : "ws.db.$yyyy$mm$dd$HH$MM.nc.gz",
          "FilePath"           : "$HOME/hmc-ws/data/dynamic/outcome/observation/ws/$yyyy/
↪$mm/$dd/",
          "FileVar"            : "AirTemperature"
                               },
                             },
                  },

    "FOR"            : {
      "VarResolution"    : 3600,
      "VarArrival"             : {"Day": 1, "Hour": ["00"]},
      "VarOp"            : {"Merging": false, "Splitting": true},
      "VarStep"          : 72,
      "VarDims"          : {"X": "west_east", "Y": "south_north", "time" :"time"},
      "VarName"          : {
        "Rain"                                    : {
          "FileName"           : "nwp.lami.$yyyy$mm$dd0000.nc.gz",
          "FilePath"           : "$HOME/hmc-ws/data/dynamic/outcome/nwp/lami-i7/$yyyy/$mm/
↪$dd/$RFarm_ID/",
          "FileVar"            : "Rain"
                               },
        "AirTemperature"       : {
          "FileName"           : "nwp.lami.$yyyy$mm$dd0000.nc.gz",
          "FilePath"           : "$HOME/hmc-ws/data/dynamic/outcome/nwp/lami-i7/$yyyy/$mm/
↪$dd/",
          "FileVar"            : "AirTemperature"
                               },
        "Wind"                                    : {
          "FileName"           : "nwp.lami.$yyyy$mm$dd0000.nc.gz",
          "FilePath"           : "$HOME/hmc-ws/data/dynamic/outcome/nwp/lami-i7/$yyyy/$mm/
↪$dd/",
          "FileVar"            : "Wind"
                               },
                  },

              },
          },
}
```

In the gridded forcing section, the following variables are mandatory:

- Rain [mm]
- Air Temperature [C]
- Wind Speed [m/s]
- Relative Humidity [%]
- Incoming Radiation [W/m^2]

Other variables are optional:

- Air Pressure [kPa]
- Albedo [0, 1]
- Leaf Area Index [0, 8]

- Point

The forcing point files are in ASCII format; the forcing data, as seen in the gridded section, are divided in two different types of variables:

- OBS: to define the observed dataset (if needed).
- FOR: to define the forecast dataset (if needed).

Each variable, in OBS and FOR section, is defined by a list of attributes in order to correctly parser the information; the attributes available are reported below:

- VarResolution: resolution time of the variable in seconds [example: 3600].
- **VarArrival: arrival time of the variable.**
  - Day: to define the variable arrival day [example: 0].
  - Hour: to define the variable arrival hour [example: null].
- **VarOp: ancillary defined operations of the variable.**
  - Merging: to merge variable from different sources in a common dataset [example: null].
  - Splitting: to split variable in different time steps [example: false].
- VarStep: step of the variable [example: 1].
- **VarDims: dimensions of the variable.**
  - Time: dimension along T axis [example: "time"].
- **VarName: name of the variable.**
  - FileName: the name of the source file [example: "rs.db.$yyyy$mmddHH$MM.txt"]
  - FilePath: the path of the source file [example: "$HOME/hmc-ws/data/dynamic/outcome/observation/rs/$yyyy/$mm/$dd/"]
  - FileVar: the name of the variable in the source file [example: "Discharge"]

An example of the forcing point data structure is reported below.

```
{
"Point"       : {
  "FileName"    : "hmc.$VAR.$yyyy$mm$dd$HH$MM.txt",
  "FilePath"    : "$HOME/hmc-ws/run/$RUN/data/forcing/point/$yyyy/$mm/$dd/",
  "FileType"    : 1,
```

(continues on next page)

(continued from previous page)

```
  "FileTimeRes"       : 3600,
  "FileVars"    : {
    "OBS"             : {
      "VarResolution"   : 3600,
      "VarArrival"      : {"Day": 0, "Hour": null},
      "VarOp"           : {"Merging": null, "Splitting": null},
      "VarStep"         : 1,
      "VarDims"         : {"T": "time"},
      "VarName"         : {
        "Discharge"                    : {
          "FileName"          : "rs.db.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"          : "$HOME/hmc-ws/data/dynamic/outcome/observation/rs/$yyyy/
↪$mm/$dd/",
          "FileVar"                  : "discharge"
                        },
        "DamV"                                  : {
          "FileName"          : "damv.db.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"          : "$HOME/hmc-ws/data/dynamic/outcome/observation/dp/$yyyy/
↪$mm/$dd/",
          "FileVar"           : "damv"
                        },
        "DamL"                                  : {
          "FileName"          : "daml.db.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"          : "$HOME/hmc-ws/data/dynamic/outcome/observation/dp/$yyyy/
↪$mm/$dd/",
          "FileVar"                  : "daml"
                        }
                      }
                },
      "FOR"       : {}
            }
}
```

In the point forcing section, all variables are optional:

- Discharge [m^3/s]
- Dam Volume [m^3]
- Dam Level [m]

- Time Series

The forcing time-series files are in ASCII format; the forcing data, as seen in the gridded section, are divided in two different types of variables:

- OBS: to define the observed dataset (if needed).
- FOR: to define the forecast dataset (if needed).

Each variable, in OBS and FOR section, is defined by a list of attributes in order to correctly parser the information; the attributes available are reported below:

- VarResolution: resolution time of the variable in seconds [example: 3600].
- **VarArrival: arrival time of the variable.**
  - Day: to define the variable arrival day [example: 0].

  - Hour: to define the variable arrival hour [example: null].
- **VarOp: ancillary defined operations of the variable.**
  - Merging: to merge variable from different sources in a common dataset [example: null].
  - Splitting: to split variable in different time steps [example: null].
- VarStep: step of the variable [example: null].
- **VarDims: dimensions of the variable.**
  - Time: dimension along T axis [example: "time"].
- **VarName: name of the variable.**
  - FileName: the name of the source file [example: "hnc.forcing-ts.plant_$NAME_PLANT.txt"]
  - FilePath: the path of the source file [example: "$HOME/hmc-ws/data/dynamic/outcome/observation/turbinate/$yyyy/$mm/$dd/"]
  - FileVar: the name of the variable in the source file [example: "DamQ"]

An example of the forcing time-series data structure is reported below.

```
{
"TimeSeries"  : {
  "FileName"    : "hmc.forcing-ts.plant_$NAME_PLANT.txt",
  "FilePath"    : "$HOME/hmc-ws/run/$RUN/data/forcing/timeseries/",
  "FileType"    : 1,
  "FileTimeRes" : 3600,
  "FileVars"    : {
    "OBS"           : {
      "VarResolution"   : 3600,
      "VarArrival"      : {"Day": 0, "Hour": null},
      "VarOp"           : {"Merging": null, "Splitting": null},
      "VarStep"         : null,
      "VarDims"         : {"T": "time"},
      "VarName"         : {
        "DamQ"                                  : {
          "FileName"          : "hmc.forcing-ts.plant_$NAME_PLANT.txt",
          "FilePath"          : "$HOME/hmc-ws/data/dynamic/outcome/observation/turbinate/
↪$yyyy/$mm/$dd/",
          "FileVar"                   : ""
                        },
          "IntakeQ"                     : {
            "FileName"        : "hmc.forcing-ts.plant_$NAME_PLANT.txt",
            "FilePath"        : "$HOME/hmc-ws/data/dynamic/outcome/observation/turbinate/
↪$yyyy/$mm/$dd/",
            "FileVar"               : ""
                        }
                    }
                },
      "FOR"           : {}
              }
            }
}
```

In the time-series forcing section, all variables are optional:

- Dam Discharge [m^3/s]
- Intake Discharge [m^3/s]

Data Updating

- Gridded

!——— ! Updating data (optional): ! a2dVarSnowCAL : snow cover area [-2,3] ! a2dVarSnowQAL : snow cover quality [0,1] ! a2dVarSnowMaskL : snow mask [0,1] ! ! a2dVarSMStarL : soil moisture value [0, 1] ! a2dVarSMGainL : soil moisture gain [0, 1] ! a2dVarSnowHeightF : snow height [cm] ! a2dVarSnowKernelF : snow kernel [0,1] !———

Data Outcome

- Gridded

The outcome gridded files are in netCDF format; the outcome data are defined using the ARCHIVE key word. This definition is used to specify and manage the results of the model.

Each variable, as seen in previous sections, is defined by a list of attributes in order to correctly parser the information; the attributes available are reported below:

- VarResolution: resolution time of the variable in seconds [example: 3600].
- **VarArrival: arrival time of the variable.**
  - Day: to define the variable arrival day [example: 0].
  - Hour: to define the variable arrival hour [example: null].
- **VarOp: ancillary defined operations of the variable.**
  - Merging: to merge variable from different sources in a common dataset [example: null].
  - Splitting: to split variable in different time steps [example: null].
- VarStep: step of the variable [example: 1].
- **VarDims: dimensions of the variable.**
  - X: dimension along X axis [example: “west_east”].
  - Y: dimension along Y axis [example: “south_north”].
- **VarName: name of the variable.**
  - FileName: the name of the source file [example: “hmc.output-grid.$yyyy$mmddHH$MM.nc.gz”]
  - FilePath: the path of the source file [example: “$HOME/hmc-ws/run/$RUN/data/outcome/$MODE/gridded/$yyyy/$mm/$dd/”]
  - FileVar: the name of the variable in the source file [example: “Discharge”]

An example of the outcome gridded data structure is reported below.

```
{
"Gridded"          : {
  "FileName"     : "hmc.output-grid.$yyyy$mm$dd$HH$MM.nc.gz",
  "FilePath"     : "$HOME/hmc-ws/run/$RUN/data/outcome/$MODE/gridded/$yyyy/$mm/$dd/",
```

(continues on next page)

(continued from previous page)

```
  "FileType"    : 2,
  "FileTimeRes"       : 3600,
  "FileVars"    : {
    "ARCHIVE"     : {
      "VarResolution"   : 3600,
      "VarArrival"      : {"Day": 0, "Hour": null},
      "VarOp"           : {"Merging": null, "Splitting": null},
      "VarStep"         : 1,
      "VarDims"         : {"X": "west_east", "Y": "south_north"},
      "VarName"         : {
        "ALL"                                 : {
          "FileName"          : "hmc.output-grid.$yyyy$mm$dd$HH$MM.nc.gz",
          "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/gridded/$MODE/",
          "FileVar"           : "ALL"
                        }
                      }
                    }
                  }
                }
}
```

In the gridded outcome files, the default configuration of outcome variables are reported below:

- Discharge: Discharge [m^3/s]
- Daily Accumulated Evapotranspiration: ETCum [mm]
- Sensible Heat: H [W/m^2]
- Latent Heat: LE [W/m^2]
- Land Surface Temperature: LST [K]
- Soil Moisture: SM [%]
- Total Soil Capacity: VTot [mm]

If the snow part is activated, the following variables are added in the outcome files:

- Snow Water Equivalent: SWE [mm]
- Snow Melting: MeltingS [mm]
- Snow Density: RhoS [kg/m^3]
- Snowfall: Snowfall [mm]
- Snow Albedo: AlbedoS [-]
- Snow Age: AgeS [steps]
- Daily Accumulated Snow Melting: MeltingSDayCum [mm]

- Point

The outcome point files are in ASCII format; the outcome data are defined using the ARCHIVE key word. This definition is used to specify and manage the results of the model.

Each variable, as seen in previous sections, is defined by a list of attributes in order to correctly parser the information; the attributes available are reported below:

- VarResolution: resolution time of the variable in seconds [example: 3600].
- **VarArrival: arrival time of the variable.**
  - Day: to define the variable arrival day [example: 0].
  - Hour: to define the variable arrival hour [example: null].
- **VarOp: ancillary defined operations of the variable.**
  - Merging: to merge variable from different sources in a common dataset [example: null].
  - Splitting: to split variable in different time steps [example: null].
- VarStep: step of the variable [example: 1].
- **VarDims: dimensions of the variable.**
  - T: dimension along t axis [example: "time"].
- **VarName: name of the variable.**
  - FileName: the name of the source file [example: "hmc.$VAR.$yyyymmddHHMM.txt"]
  - FilePath: the path of the source file [example: "$HOME/hmc-ws/run/$RUN/data/outcome/$MODE/point/$yyyy/$mm/$dd/"]
  - FileVar: the name of the variable in the source file [example: "DamV"]

An example of the outcome point data structure is reported below.

```
{
"Point"       : {
  "FileName"    : "hmc.$VAR.$yyyy$mm$dd$HH$MM.txt",
  "FilePath"    : "$HOME/hmc-ws/run/$RUN/data/outcome/$MODE/point/$yyyy/$mm/$dd/",
  "FileType"    : 1,
  "FileTimeRes" : 3600,
  "FileVars"    : {
    "ARCHIVE"     : {
      "VarResolution"   : 3600,
      "VarArrival"      : {"Day": 0, "Hour": null},
      "VarOp"           : {"Merging": null, "Splitting": null},
      "VarStep"         : 1,
      "VarDims"         : {"T": "time"},
      "VarName"         : {
        "Discharge"       : {
          "FileName"          : "hmc.discharge.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/point/$MODE/discharge/",
          "FileVar"           : "discharge"
                      },
        "DamV"            : {
          "FileName"          : "hmc.vdam.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/point/$MODE/dam_volume/",
          "FileVar"           : "vdam"
                      },
        "DamL"            : {
          "FileName"          : "hmc.ldam.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
```

(continues on next page)

(continued from previous page)

```
↪outcome/point/$MODE/dam_level/",
          "FileVar"            : "ldam"
                        },
        "VarAnalysis"      : {
          "FileName"           : "hmc.var-analysis.$yyyy$mm$dd$HH$MM.txt",
          "FilePath"           : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/point/$MODE/analysis/",
          "FileVar"            : "var-analysis"
                        }
                      }
                    }
                  }
              }
}
```

The default variables saved by the model in point format are reported:

- Discharge: discharge [m^3/s]

In addition, if dams are included in the model configuration, the following variables are added:

- Dam volume: vdam [m^3];
- Dam Level: ldam [m].

Moreover, a point analysis file, to control and check the run of the model, is saved, when the FlagDebug > 0, with information for some state variables:

- Average variables: var-analysis.

In the analysis file, the following fields are saved on three columns [average, max, min]:

1. Rain [mm]
2. AirTemperature [C]
3. IncomingRadiation [W/m^2]
4. Wind Speed [m/s]
5. Relative Humidity [%]
6. AirPressure [kPa]
7. LAI [-]
8. Albedo [0,1]
9. Land Surface Temperature [K]
10. Sensible Heat [W/m^2]
11. Laten Heat [W/m^2]
12. Evapotranspiration [mm]
13. Intensity [mm]
14. VTot [mm]
15. Volume Retention [mm]
16. Volume Subflow [mm]
17. Volume Losses [mm]

18. Volume Exfiltration [mm]
19. Flow Deep [mm]
20. Watertable [mm]

- Time Series

The outcome time-series files are in ASCII format; the outcome data are defined using the ARCHIVE key word. This definition is used to specify and manage the results of the model.

Each variable, as seen in previous sections, is defined by a list of attributes in order to correctly parser the information; the attributes available are reported below:

- VarResolution: resolution time of the variable in seconds [example: 3600].
- **VarArrival: arrival time of the variable.**
  - Day: to define the variable arrival day [example: 0].
  - Hour: to define the variable arrival hour [example: null].
- **VarOp: ancillary defined operations of the variable.**
  - Merging: to merge variable from different sources in a common dataset [example: null].
  - Splitting: to split variable in different time steps [example: null].
- VarStep: step of the variable [example: 1].
- **VarDims: dimensions of the variable.**
  - T: dimension along t axis [example: "time"].
- **VarName: name of the variable.**
  - FileName: the name of the source file [example: "hmc.$VAR.txt"]
  - FilePath: the path of the source file [example: "$HOME/hmc-ws/run/$RUN/data/outcome/time-series/$MODE/"]
  - FileVar: the name of the variable in the source file [example: "hydrograph"]

An example of the outcome time-series data structure is reported below.

```
{
"TimeSeries"  : {
  "FileName"    : "hmc.$VAR.txt",
  "FilePath"    : "$HOME/hmc-ws/run/$RUN/data/outcome/$MODE/timeseries/",
  "FileType"    : 1,
  "FileTimeRes"       : 3600,
  "FileVars"    : {
      "ARCHIVE"   : {
        "VarResolution"   : 3600,
        "VarArrival"      : {"Day": 0, "Hour": null},
        "VarOp"           : {"Merging": null, "Splitting": null},
        "VarStep"         : 1,
        "VarDims"         : {"T": "time"},
        "VarName"         : {
          "Discharge"       : {
            "FileName"          : "hmc.hydrograph.txt",
            "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/timeseries/$MODE/",
```

(continues on next page)

(continued from previous page)

```
            "FileVar"           : "hydrograph"
                              },
          "DamV"              : {
            "FileName"          : "hmc.vdam.txt",
            "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/timeseries/$MODE/",
            "FileVar"           : "vdam"
                              },
          "DamL"                                  : {
            "FileName"          : "hmc.ldam.txt",
            "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/timeseries/$MODE/",
            "FileVar"                 : "ldam"
                              },
          "VarAnalysis"                 : {
            "FileName"          : "hmc.var-analysis.txt",
            "FilePath"          : "$HOME/hmc-ws/data/archive/$RUN/$yyyy/$mm/$dd/$HH/
↪outcome/timeseries/$MODE/",
            "FileVar"                 : "var-analysis"
                              }
                            }
                  }
                }
              }
}
```

The default variables saved by the model in time-series format are reported:

- Discharge: discharge [m^3/s]

In addition, if dams are included in the model configuration, the following variables are added:

- Dam volume: vdam [m^3];
- Dam Level: ldam [m].

Moreover, a time-series analysis file, to control and check the run of the model, is saved, when the FlagDebug > 0, with information for some state variables:

- Average variables: var-analysis.

The time-series variables saved in the outcome file are the same as reported in the outcome point files.

Data State / Data Restart

- Gridded

The Hydrological Model Continuum saves, using the temporal resolution set in the namelist, the following gridded variables:

- Total volume [mm]
- Retention volume [mm]
- Hydro level [-]
- Routing [-]

- Flow deep and exfilration [-]
- Water table level [m]
- Land surface temperature [K]
- Air temperature marked [K]
- Air temperaure last 24 hours [K]
- Water sources [m^3/s]

If the snow part is activated, in addition the variables related to the snow physics will be saved:

- Snow water equivalent [mm]
- Snow density [kg/m^3]
- Snow albedo [-]
- Snow age [day]
- Air temperature last day [C]
- Air temperature last 5 day(s) [C]
- Point

The Hydrological Model Continuum saves, using the temporal resolution set in the namelist, the following point variables:

- **Dam(s)**
  - matrix coordinates of dams [-]
  - codes of dams [-]
  - dams volume max [m^3]
  - dams volume [m^3]
- Lake(s)
- **matrix coordinates of lakes [-]**
  - codes of lakes [-]
  - lakes volume min[m^3]
  - lakes volume [m^3]

3.2 Model execution

The Hydrological Model Continuum can be run in different configurations according with the use and the applications for which it is designed for. It is possible to configure model using the hmc python3 package or in stand-alone version; using the stand-alone version, the model can be easily integrated in different frameworks ensuring the dependencies of zlib, hdf5 and netCDF4 libraries. Usually, the model is distributed with a configure file to compile the source codes in a generic Linux Debian/Ubuntu system and to select different configuration flags. Generally, to properly build the model, the users have to follow the following steps:

1. build the model dependencies compiling zlib, hdf5 and netCDF4 libraries;
2. build the model codes against the libraries previouly installed

3.2.1 Wrapping mode

As said in the previous paragraph, the Hydrological Model Continuum can be configured using the hmc python3 package; it is the generic packege to use the model and in the previous sections all parameters and datasets are fully explained for avoiding mismatches and failures in setting parameters or preparing datasets.

The application for running the model is stored in the Apps folder in the main root of the package as reported in the following package tree:

```
hmc-master
├── apps
│   ├── **HMC_Model_Run_Manager.py**
│   ├── configuration_run_app_1.json
│   ├── configuration_run_app_2.json
│   ├── configuration_data_app_1.json
│   ├── configuration_data_app_2.json
│   └── ...
├── bin
├── docs
├── hmc
├── AUTHORS.rst
├── CHANGELOG.rst
├── LICENSE.rst
└── README.rst
```

As said at the beginning of this documentation, a generic execution of the model is performed using the following line in a terminal:

```
>> python3 HMC_Model_RUN_Manager.py -settings_file configuration.json -time "yyyymmddHHMM
↪"
```

3.2.2 Standalone mode

The Hydrological Model Continuum can be run in a stand-alone version too; the users have to be properly filled the namelist of the model and using the command-line to launch the following instructions to set the libraries:

```
>> export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/path_to_netcdf_folder/lib/
>> ulimit -s unlimited
```

EXECUTION TYPE 1: The executable of the model can ben launch using a list of seven parameters according with the following notation:

```
>> ./HMC_Model_V2_DomainName.x [parameters: uc, uh, ct, cf, domain, cpi, rf, vmax,␣
↪slopemax]
```

An filled example of the previous command-line, is reported:

```
>> ./HMC_Model_V2_DomainName.x 30 3 0.6 0.015 DomainName 0.3 500 1 70
```

EXECUTION TYPE 2: It is possible to run the Hydrological Model Continuum using a command-line in which only the model namelist is passed:

```
>> ./HMC_Model_V2_DomainName.x domainname.info.txt
```

This approach is generally preferred because the number of argument is fixed and never vary in different configuration and applications; in fact, all the changes in the model (e.g. flags, datasets, parameters) will be managed using the namelist file.

3.2.3 Debugging mode

The users could be interested in debugging codes to have a deeper knowledge of the Hydrological Model Continuum; usually, for doing this task, the common advice is used a IDE (e.g, CodeBlock, VS Code, Apache Netbeans) in order to easily analyze codes and memory usage. In this part, the configuration of a debug workspace in Apache NetBeans IDE will be presented.

First of all, the package for C, C++ and Fortran programming languages have to be installed in Apache NetBeans; to complete this step, the users have to install the package related with C/C++ language. Particuarly, following these instructions:

1) Tools –> Plugins –> Settings –> Tick "NetBeans 8.2 Plugin Portal" –> Close

and Reboot the Apache NetBeans IDE. Next step, users should create a New Project following these instructions:

2) File –> New Project –> Category :: Sample :: C/C++ –> Project :: Hello World Fortran Application –> Next –> Choose Name –> Close

After creating a folder project, users have to import all source code in the project folder; Using the left menu where the name of projects are visible, right click on selected project:

3) Source Files –> Add existing items ...

performing this action, a form to select all file will be opened. Finally, all source files will be available into source file folder of the selected project. Next steps cover the configuration of the dependencies in the project. Particularly, the main task is linking the NetCDF library against the project. For configuring the NetCDF4 in Apache NetBeans IDE, the users have to add in:

4) Project –> Properties –> Linker –> Libraries

   in /path_to_netcdf/ find the following files netcdff.a and netcdff.so and note "double f" for fortran libraries

5) Project –> Properties –> Linker –> Additional Options

   -I/path_to_netcdf/include/ -L/path_to_netcdf/lib/ -lnetcdff -lnetcdf

6) Project –> Properties –> Fortran Compiler –> Additional Options

   -I/path_to_netcdf/include/ -L/path_to_netcdf/lib/ -lnetcdff -lnetcdf

7) Project –> Properties –> Fortran Compiler –> Additional Options

   gfortran: -cpp -DLIB_NC ifort: -fpp -DLIB_NC

8) Project –> Properties –> Run –> Environment –> NewValue

   Name: LD_LIBRARY_PATH Value: $LD_LIBRARY_PATH:/path_to_necdf/lib/

Once the NetCDF4 are linked, it will be possible to compile each source file using the F9 key. After doing all these steps, the users have to set the debug command to run Hydrological Model Continuum using, for instance, a namelist file of a study case:

9) Debug –> Debug Command

   "${OUTPUT_PATH}" domainname.info.txt

After setting the environment and all needed options for running the model, the users will are able to get a deeper information using the options to execute code in a debugging mode using breakpoints and all the features available in **gdb** debugging library.

3.2.4 Profiling Mode

Another option for the users is the profiling of the model using the **gprof** command. This tool provides a detailed postmortem analysis of program timing at the subprogram level, including how many times a subprogram was called, who called it, whom it called, and how much time was spent in the routine and by the routines it called.

To enable gprof profiling, the users have to compile and link the program with the -pg option; in the configure program is a section for compiling the model in profiling model.

```
>> ./HMC_Model_V2_DomainName.x domainname.info.txt
```

The simulation must complete normally for gprof to obtain meaningful timing information. At program termination, the file **gmon.out** is automatically written in the working directory. This file contains the profiling data that will be interpreted by gprof. To obtain a ascii file for all information, the following command have to be execute ./$name_exec 30 3 0.6 0.015 marche 0.3 500 1 70

```
>> gprof HMC_Model_V2_DomainName.x gmon.out > hmc_model_analysis.txt
```

Alternatively, a python script to plot a scheme of model flow is provided in the hmc package:

```
>> gprof HMC_Model_V2_DomainName.x | ./gprof2dot.py | dot -Tpng -o hmc_model_analysis.png
```

CHAPTER

FOUR

EXECUTABLES

description here

CHAPTER

FIVE

DEVELOPERS

- Fabio Delogu <fabio.delogu@cimafoundation.org>
- Simone Gabellani <simone.gabellani@cimafoundation.org>
- Francesco Silvestro <francesco.silvestro@cimafoundation.org>
- Valerio Basso <valerio.basso@cimafoundation.org>
- Alessandro Masoero <alessandro.masoero@cimafoundation.org>
- Nicola Rebora <nicola.rebora@cimafoundation.org>

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search