hyde Documentation

Release 1.9.1

Fabio Delogu

Jul 14, 2023

CONTENTS:

ONE

OVERVIEW

The **Hydrological Data Engines**, commonly named **HyDE**, 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 data dynamic processing part is performed by using the HyDE package [1]. Written in python3 language, HyDE manages all aspects concerning the production of the datasets suitable for the FloodPROOFS modelling system. This package is able to read data both from various types of information (weather stations, satellite, nwp …) and in different formats (such as GRIB, BUFR, hdf, hdf5, netcdf4 …); once getting and reading the source data, HyDE converts them from their original struture to formats commonly used by FloodPROOFS modelling system (for instance netCDF4, JSON and so on) and by other freely available applications (such as cdo, panoply, ncview, MET …).

For adapting datasets to the requested format, HyDE provides a checking of names, units and physics limits of all variables used by the FloodPROOFS modelling system. The reprojection or the interpolation of datasets, where they

are needed, are performed by HyDE package too.

Particularly, the HyDE 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:

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

All codes and datasets 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 configured, 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

APPLICATIONS

In this section, applications for different data type are presented; each of them prepare data to suit the conventions used in HyDE package and in FloodPROOFS modelling system too. Once data are collected from different sources, they are available in a raw format and it is necessary to adapt them, for example changing format, resolutions and domain, to the outcome conventions. All applications are stored in the apps folder in the root of the package; an example of apps tree is reported:

```
hyde-master
├── **apps**
│   ├── ground_network
│   │   ├── rs
│   │   └── ws
│   ├── nwp
│   │   ├── gfs
│   │   ├── lami
│   │   └── wrf
│   ├── rfarm
│   │   ├── lami
│   │   └── wrf
│   ├── satellite
│   ├── utils
│   └── ...
├── bin
├── docs
├── src
│   ├── common
│   └── hyde
├── test
├── AUTHORS.rst
├── CHANGELOG.rst
├── LICENSE.rst
└── README.rst
```

2.1 Ground Network Applications

The procedure of **weather stations** data, available in HyDE package is divided in two parts:

1. extraction of weather stations datasets, both for all requested variables and every time step, from an local or worlwide database (using queries or functions developed specifically for that database) and storage of all variables data in a CSV file for each variable and for every timestep;

2. conversion of stations data, available in the CSV files created previously, in a spatial information over the selected domain (using, for instance, interpolation, regridding or warping methods) and saving all variables in a netcdf4 file for every timestep.

As previously said, the first step of procedure is about the downloading/getting data from a database or data repository. The extraction of weather stations data is performed by an python3 application available in HyDE package in "/hyde-master/bin/downloader/ws/" named "hyde_downloader_{database_name}_ws.py"; for configuring this procedure, a JSON file usually named "hyde_downloader_{database}_ws.json" has to be filled in all of its part. The name of used database has to be specified by the users according with their settings. At the end, a output CSV file will be created with all data for each variable and for every timestep.

General use of the weather stations downloader is reported below:

```
>> python3 hyde_downloader_ws.py -settings_file hyde_configuration_ws.json"
```

In the next example, the settings of downloading procedure for rain data is reported. Users can activate downloading of selected variable using attribute "download" equal to "true". The units of the variable can be specified in order to save this information in the CSV file.

```
"rain": {
    "download": true,
    "name": "Rain",
    "sensor": "Raingauge",
    "units": "mm"
},
```

According with the period of simulation or analysis, the time fields have to be updated. Frequency and expected data period have to be set by the users.

```
"time_info": {
    "time_get": "201809010500",
    "time_period": 6,
    "time_frequency": "H"
},
```

Once source data are configured, the weather stations data will be save in a CSV file.

The second step of procedure concerns the spatialization of weather stations data. Spatial datasets of weather stations data are realized using an python3 application available in HyDE package in "/hyde-master/apps/ground_network/ws/" named "HYDE_DynamicData_GroudNetwork_WS.py"; for configuring this procedure, a JSON file usually named "hyde_configuration_groundnetwork_ws.json" has to be filled in all of its part. Starting from CSV file previously created, the script will able to save an netCDF4 output file will be created with all variables data for every timestep.

General use of the weather stations application is reported below:

```
>> python3 HYDE_DynamicData_GroundNetwork_WS.py -settings_file hyde_configuration_
↪groundnetwork_ws.json -time "%Y-%m-%d %H:%M"
```

For example, if the application is set to 2016-03-13 22:00:

```
>> python3 HYDE_DynamicData_GroundNetwork_WS.py -settings_file hyde_configuration_
↪groundnetwork_ws.json -time "2016-03-13 22:00"
```

AirPressure		Geo2D
AirTemperature		Geo2D
crs	crs	—
IncRadiation		Geo2D
Latitude	latitude coordinate	2D
Longitude	longitude coordinate	2D
Rain		Geo2D
RelHumidity		Geo2D
SnowKernel		Geo2D
SnowLevel		Geo2D
Terrain	geometric height	Geo2D
time	time	—
time_bnds	time_bnds	1D
Wind		Geo2D

Fig. 1: Example of variable list for an weather stations output file.

For example to set the time information, users have to change a time fields according with their simulation settings:

```
"time": {
    "time_forecast_step": 0,
    "time_forecast_delta": 0,
    "time_observed_step": 10,
    "time_observed_delta": 3600,
    "time_reference_type": {
        "units": null,
        "rounding": null,
        "steps": null
    }
}
```

All the input variables needed for performing simulation have to be set in dynamic source and outcome fields available in the JSON configuration file. In the next example, rain configurarion options are reported. Users can change, for instance, the interpolation method (for example "nearest") and interpolation radius influence along x and y. If some variables are not available to avoid warnings and faults, users can be set "var_mode" to "false" in order to save time and skip uncessary elaborations.

```
"outcome": {
    "rain_data":{
        "id": {
            "var_type": ["var2d", "interpolated"],
            "var_mode": true,
            "var_name": ["Rain"],
            "var_file": "ws_product",
            "var_colormap": "rain_colormap",
            "var_method_save": "write2DVar"
        },
        "attributes": {
```

```
            "long_name": "Rain",
            "standard_name": "rain",
            "_FillValue": -9999.0,
            "ScaleFactor": 1,
            "units": "mm",
            "Valid_range": [0, null],
            "description": ""
        }
    },
}
```

Once source data are configured, outcome data need to be configureted too. Users, for example, can be set units, scale factor and valid range to properly filter weather stations data and define a spatial information that will be saved in a netCDF4 output file.
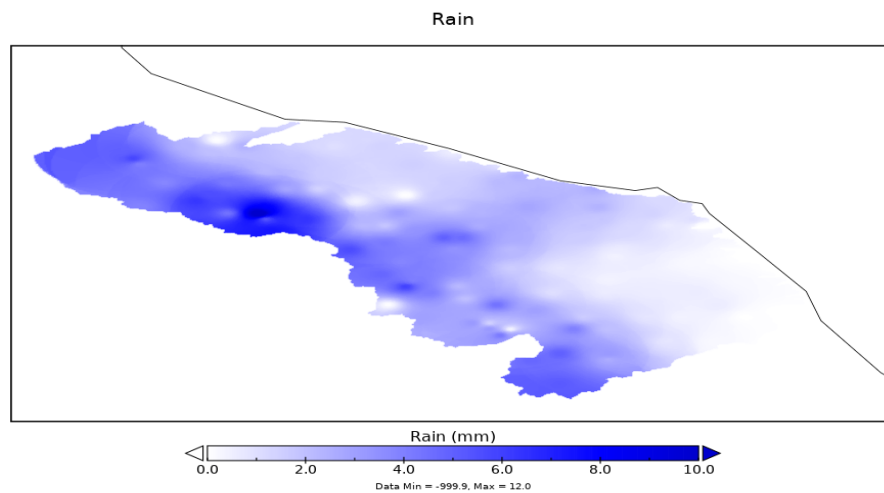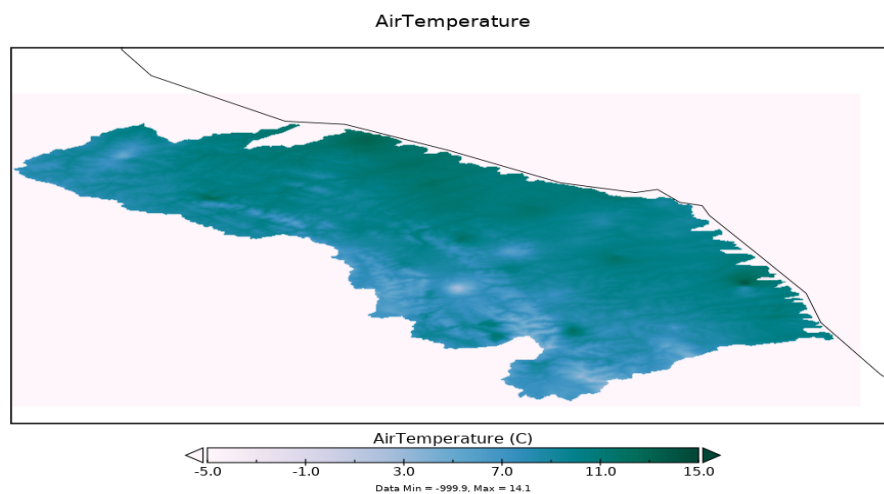

Fig. 2: Example of rain map for Marche Italian region.

Fig. 3: Example of air temperature map for Marche Italian region.

2.2 Satellite Applications

2.2.1 H-SAF

The **"EUMETSAT Satellite Application Facility on Support to Operational Hydrology and Water Management (H-SAF)"** started on 2005 as part of the EUMETSAT SAF Networks. The H-SAF generates and archives high-quality data sets and products for operational hydrological applications starting from the acquisition and processing of data from Earth observation satellites in geostationary and polar orbits operated both by EUMETSAT and other satellite organization. The retrieval of products uses data from microwave and infrared instruments and aims at reaching the best possible accuracy compatible with satellite systems as available today or in the near future. H-SAF applications fit with the objectives of other European and international programmes with special relevance to those initiatives which want to mitigate hazards and natural disasters such as flash floods, forest fires, landslides and drought conditions, and improve water management. All the information about the project and the product are available on HSAF WebSite.

In H-SAF project the products are divided in three categories:

- **precipitation** (liquid, solid, rate, accumulated;
- **soil moisture** (at large-scale, at local-scale, at surface, in the roots region);
- **snow parameters** (detection, cover, melting conditions, water equivalent).

In HyDE python package, the following applications are available to perform analysis over these products:

- precipitation –> **H03b**, **H05b**;
- soil moisture –> **H14**, **H16**, **H101**;
- snow parameters –> **H10**, **H12**, **H13**.

The procedures of H-SAF consist in some applications written in python3 available in HyDE package in "/hyde-master/apps/satellite/HSAF/".

The **product H03B** is based on the IR images from the SEVIRI instrument on-board Meteosat Second Generation (MSG) satellites. The spatial coverage of the product included the H-SAF area (Europe and Mediterranean basin) and the Africa and Southern Atlantic Ocean. Thus the new geographic region covers the MSG area correspondent to 60°S – 67.5°N, 80°W – 80°E.

The product is generated at the 15-min imaging rate of SEVIRI, and the spatial resolution is consistent with the SEVIRI pixel. The precipitation estimates are obtained by combining IR GEO equivalent blackbody temperatures at 10.8 m with rain rates from PMW measurements. The units of H03b istantaneous rain product are expressed in [mm/h].

General use of the **H03B application** is reported below:

```
>> python3 HYDE_DynamicData_HSAF_H03B.py -settings_file configuration.json -time "%Y-%m-
↪%d %H:%M"
```

The **h05b product** is derived from precipitation maps generated by merging MW images from operational sun-synchronous satellites and IR images from geostationary satellites. Integration is performed over 3, 6, 12 and 24 hours.

In order to reduce biases, the satellite-derived field is forced to match raingauge observations and, in future, the accumulated precipitation field outputted from a NWP model. The units of H05b accumulated rain product are expressed in [mm].

General use of the **H05B application** is reported below:

```
>> python3 HYDE_DynamicData_HSAF_H05B.py -settings_file configuration.json -time "%Y-%m-
↪%d %H:%M"
```
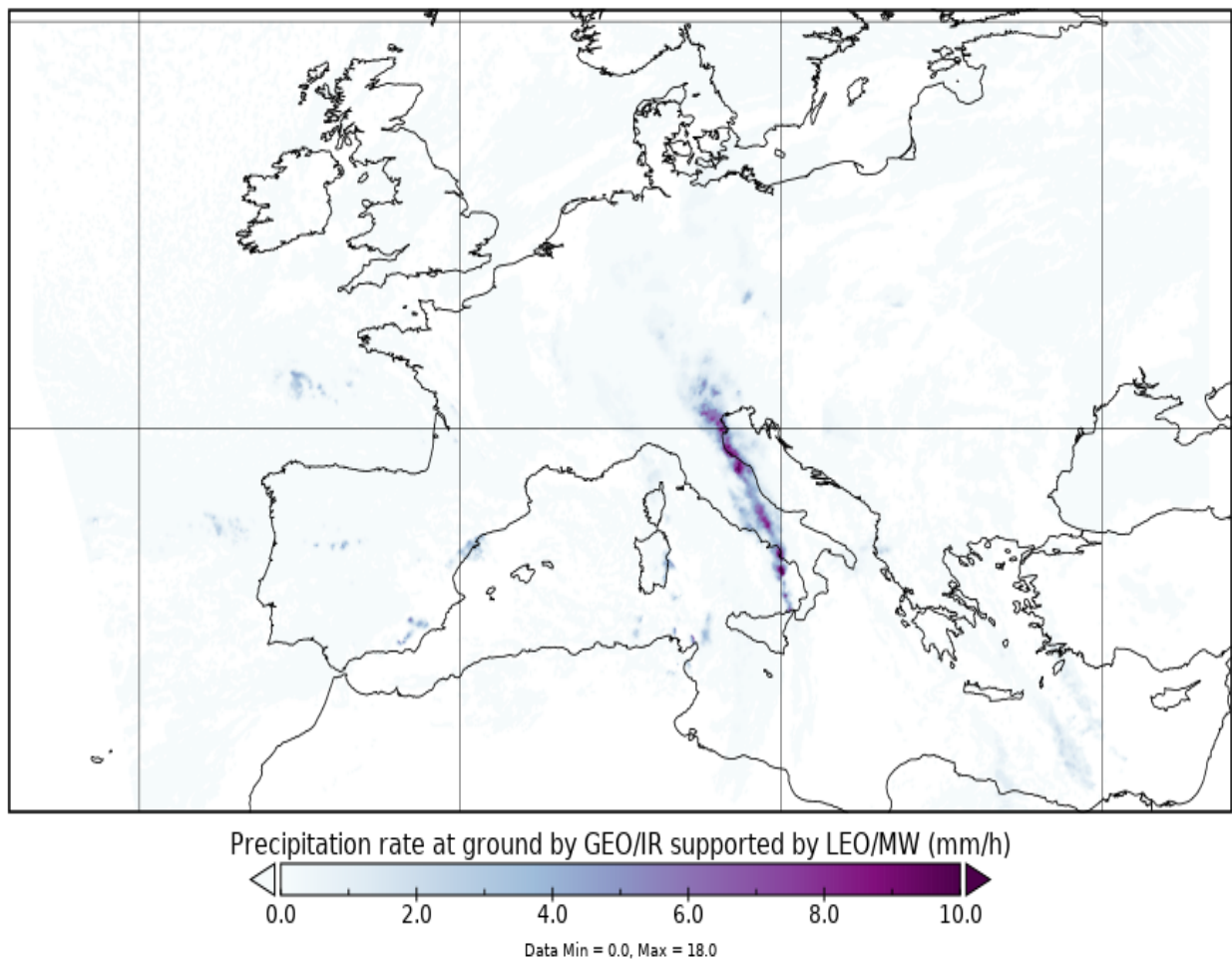
Fig. 4: Example of h03b istantaneous rain map for Europe domain.

Accumulated precipitation at ground by blended MW and IR (24 accumulated hours)
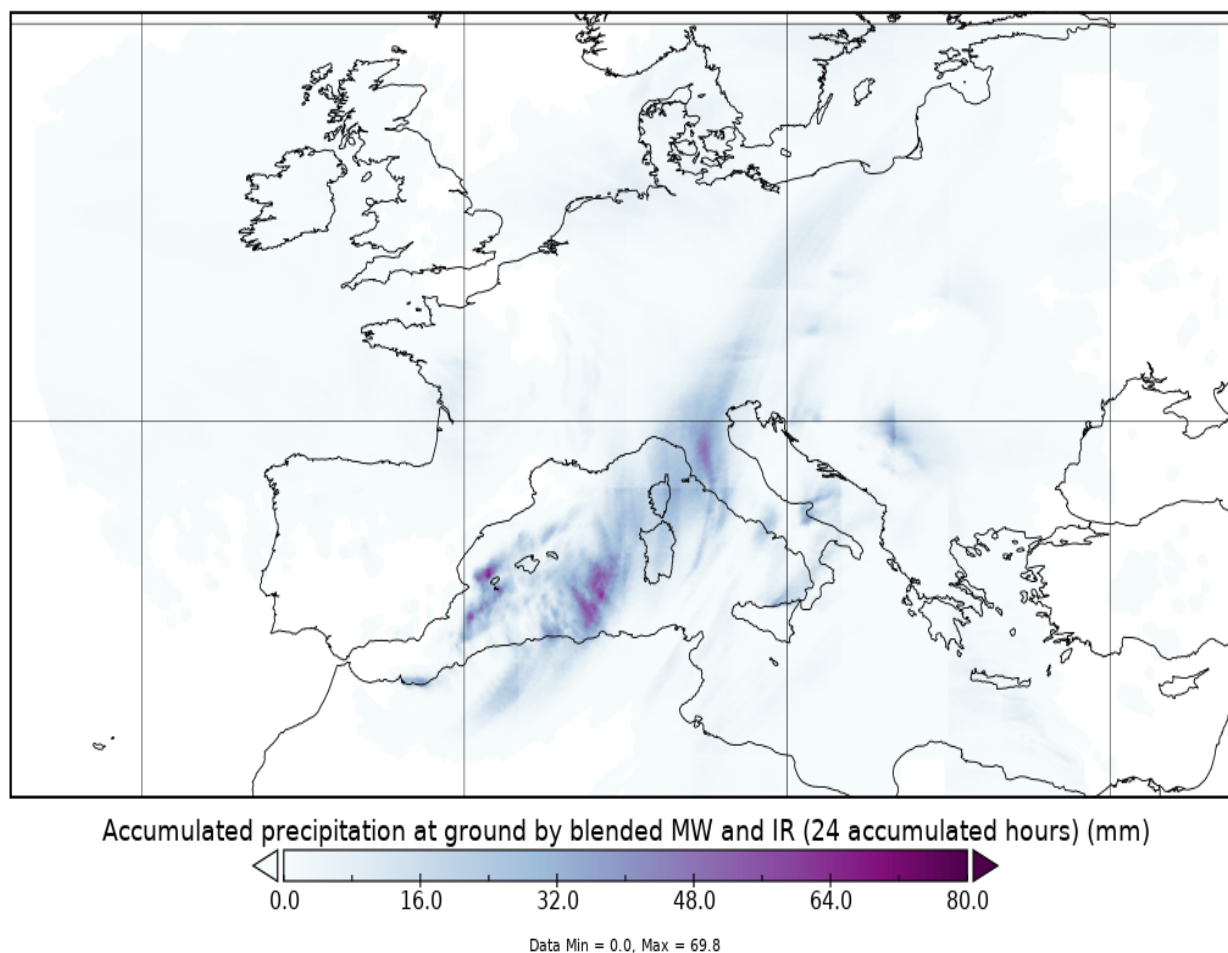

Fig. 5: Example of h05b accumulated rain map for Europe domain.

The **H10 product** is an output of image classification processing. The snow signature is recognised as differential brightness in more short-wave channels, intended to discriminate snow from no-snowed land and snow from clouds.
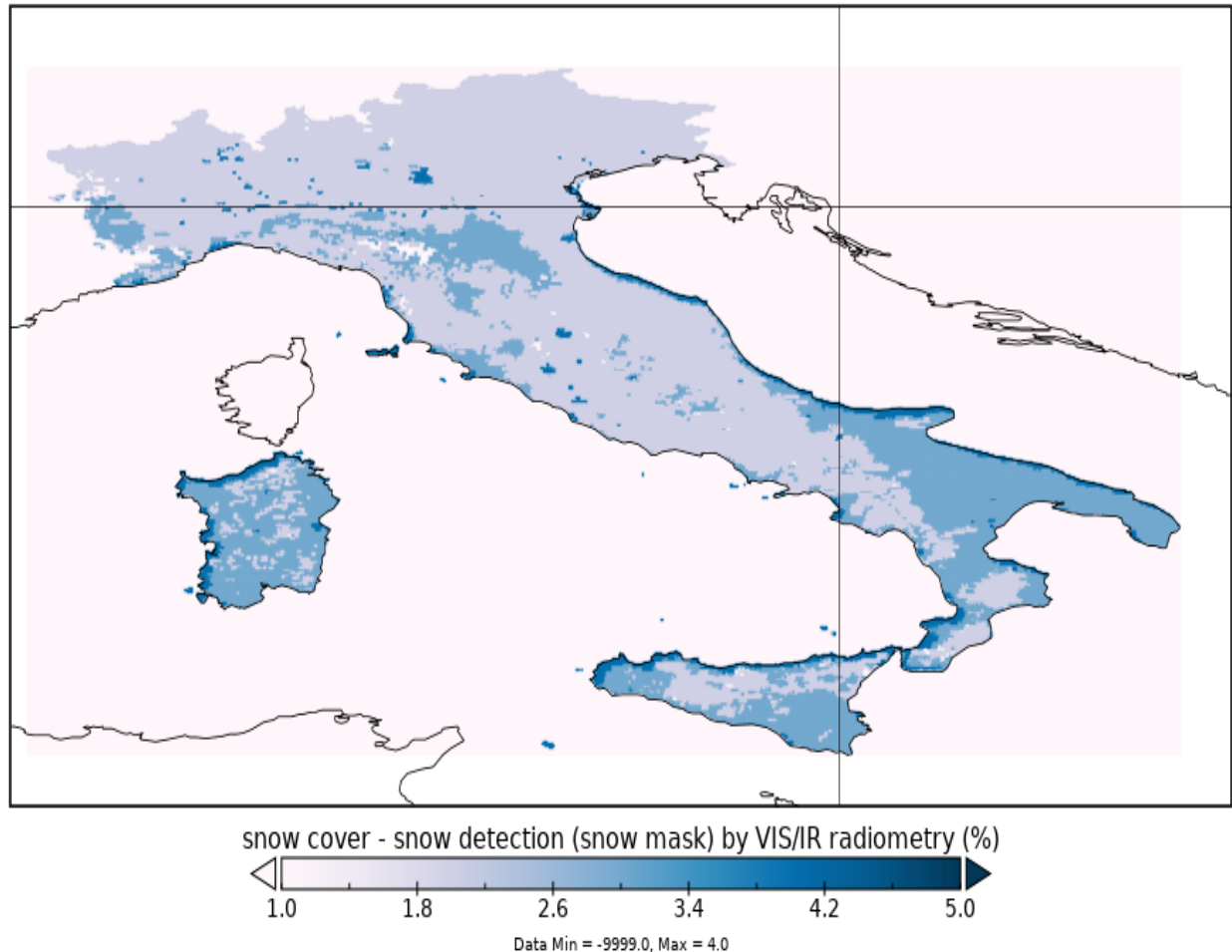

Fig. 6: Example of h10 snow cover map for Italy domain, where 1:snow, 2:cloud, 3:bare ground, 4:water, 5:dark/no_data.

Both radiometric signatures are used (specifically, the 1.6 micron channel as compared with others), and time-persistency (for cloud filtering by the "minimum brightness" technique applied over a sequence of images).

General use of the **H10 application** is reported below:

```
>> python3 HYDE_DynamicData_HSAF_H10.py -settings_file configuration.json -time "%Y-%m-
→%d %H:%M"
```

The **H12 product** differs from H10 in so far as, in the snow map, the resolution elements report the fractional snow coverage instead of being binary (snow/snow-free). The possibility of appreciating fractional coverage stems from the lack of observed brightness in respect of what would be if the pixel were fully filled by snow..

The forest canopy obscuring the full visibility to the ground is accounted for by applying certain a priori transmissivity information, which must be generated using satellite-borne reflectance data acquired under full dry snow cover conditions
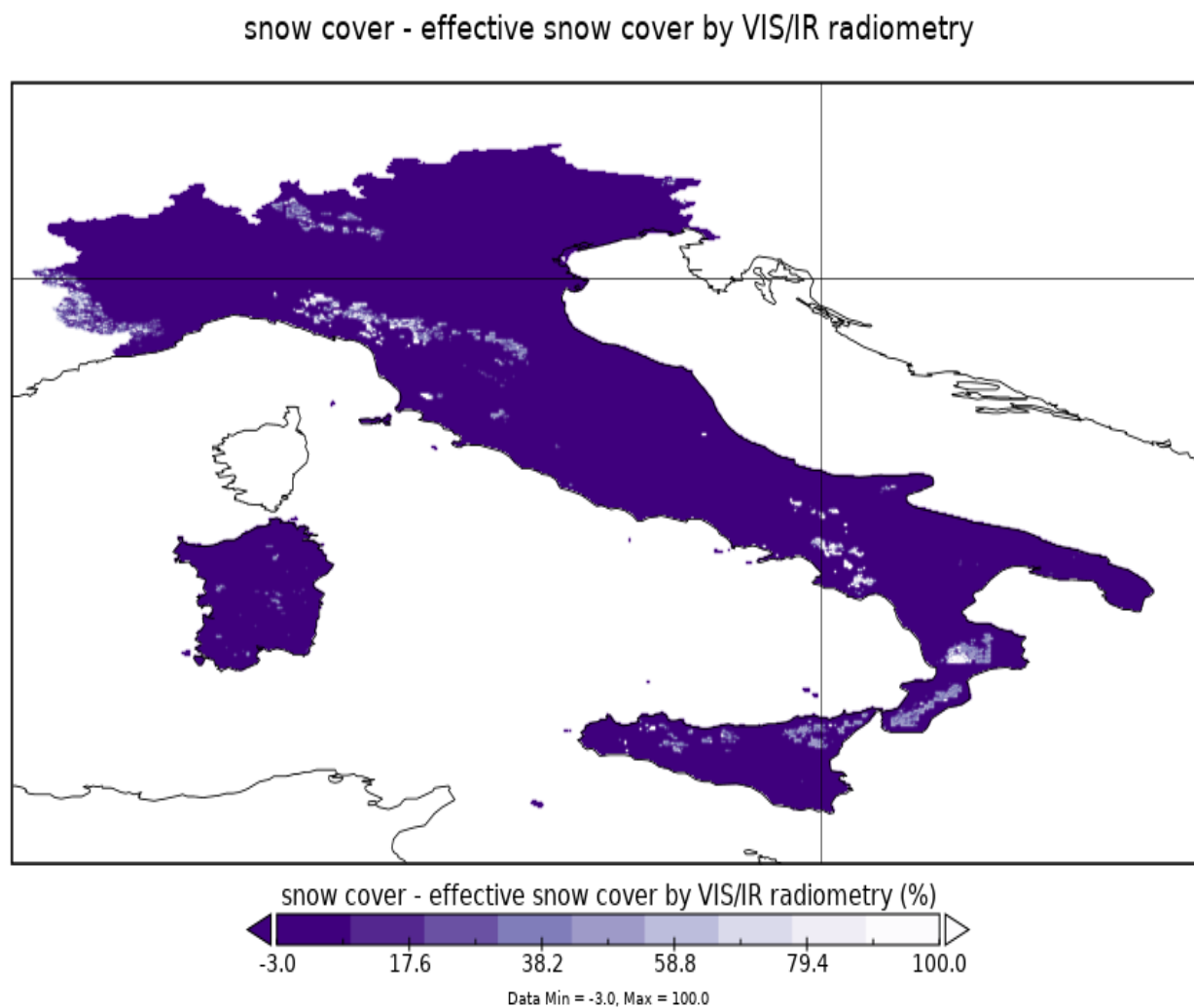
snow cover - effective snow cover by VIS/IR radiometry

Fig. 7: Example of h12 snow cover map for Italy domain, where 0:bare ground, 1-100:fractional cover -1:no_classified/no_data, -2:cloud, -3:water.

General use of the **H12 application** is reported below:

```
>> python3 HYDE_DynamicData_HSAF_H12.py -settings_file configuration.json -time "%Y-%m-
↪%d %H:%M"
```

The **H13 product** is generated by using microwaves that are sensitive to snow thickness and density, i.e. to the snow water equivalent. Depending on the snow being dry or wet, the penetration changes (dry snow is more transparent). High frequencies are required for dry snow, which is an advantage from the resolution viewpoint.
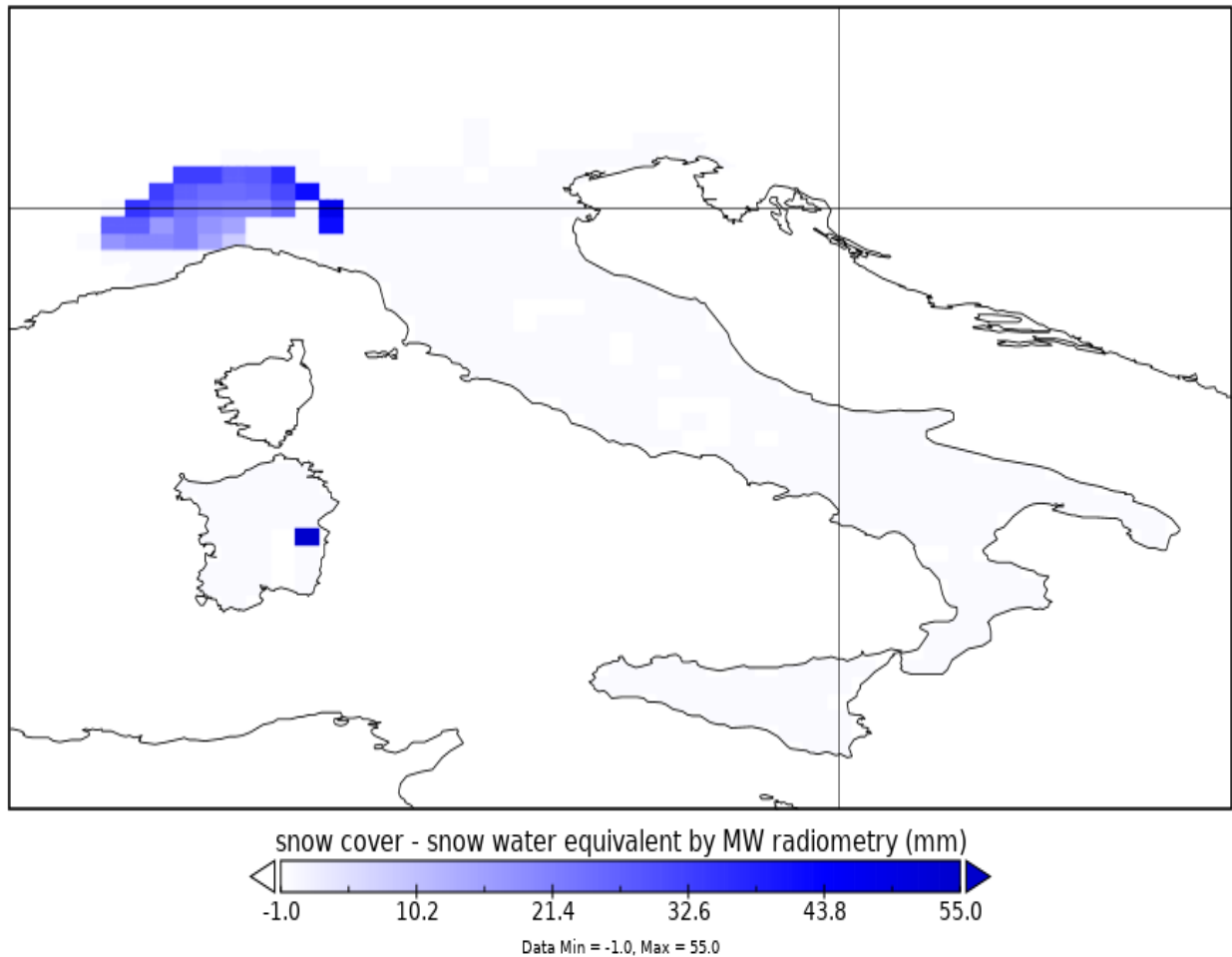
Fig. 8: Example of h13 snow water equivalent map for Italy domain, where 0:bare ground, 1-500:snow water equivalent, -1:no_classified/no_data

However, with increasing snow depth, lower frequencies are necessary for better penetration, thus a multifrequency approach is required. It is an all-weather, night-and-day measurement, whose processing requires considerable support from ancillary information. Method and performance may be different for flat/forested areas and mountainous regions.

General use of the **H13 application** is reported below:

```
>> python3 HYDE_DynamicData_HSAF_H13.py -settings_file configuration.json -time "%Y-%m-
↪%d %H:%M"
```

The Level 2 surface soil moisture products **H16** and **H101** are derived from the radar backscattering coefficients measured by the Advanced Scatterometer (ASCAT) on-board the series of Metop satellites using a change detection method, developed at the Research Group Remote Sensing, Department for Geodesy and Geoinformation (GEO), Vienna University of Technology (TU Wien).
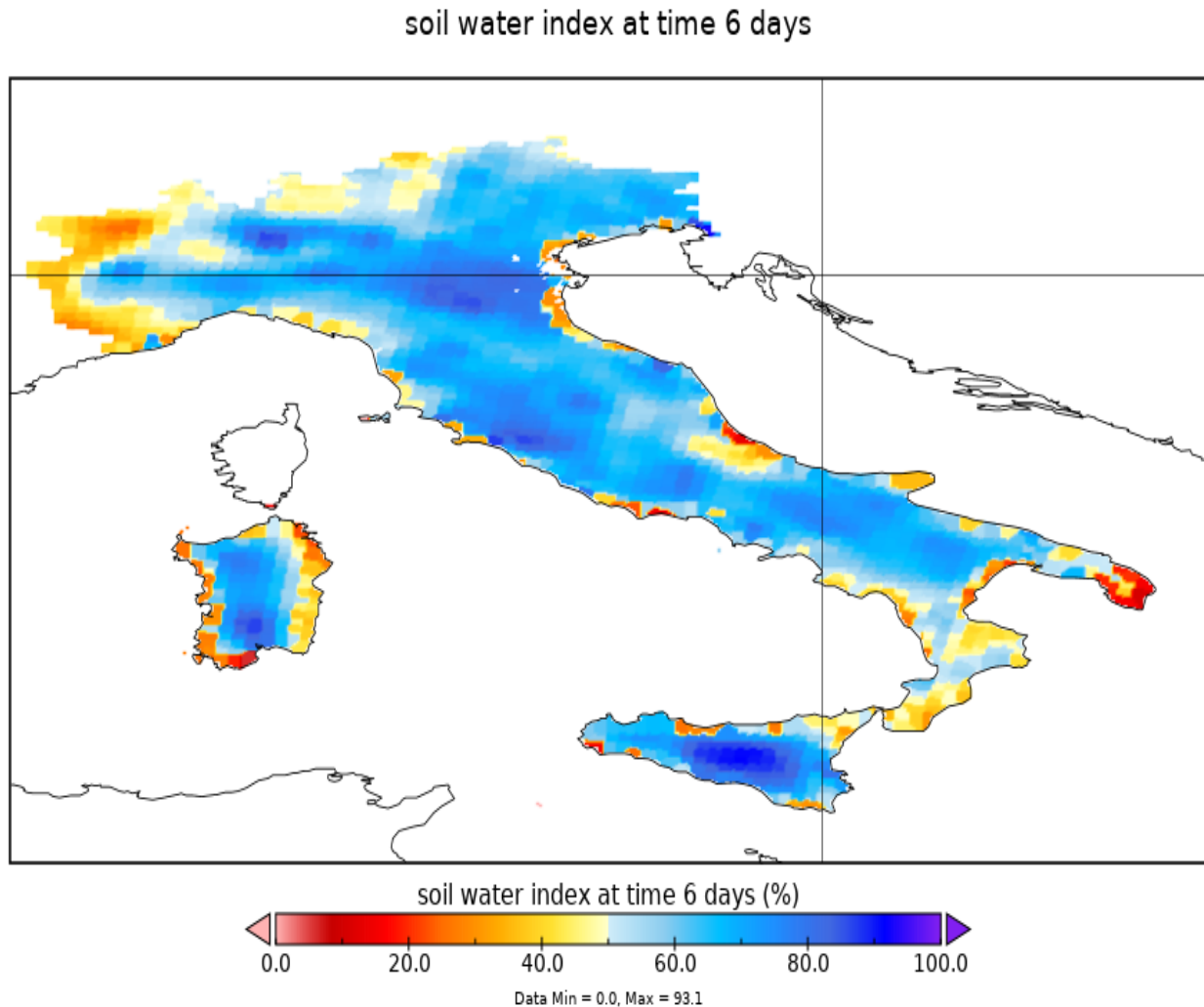

Fig. 9: Example of ASCAT H16/H101 soil water index (SWI) map, based on surface soil moisture (SSM) time-series.

In the TU Wien soil moisture retrieval algorithm, long-term Scatterometer data are used to model the incidence angle dependency of the radar backscattering signal. Knowing the incidence angle dependency, the backscattering coefficients are normalized to a reference incidence angle. Finally, the relative soil moisture data ranging between 0% and 100% are derived by scaling the normalized backscattering coefficients between the lowest/highest values corresponding to the driest/wettest soil conditions. Once SSM values are obtained, next step is how to extend the influnce of SSM at the different soil deepths. Starting from the SSM values, the Soil Water Index (SWI) is a method to estimate root zone soil moisture using an exponential Filter.

The **ASCAT OBS applications**, based on **H16** and **H101** products, are develop in order to compute the time-series of SSM and SWI both for the **data record** (DR) and the **near-real-time** (NRT) datasets.

General use of ASCAT OBS application for computing data record dataset is reported below:

```
>> python3 HYDE_DynamicData_HSAF_ASCAT_OBS_DR.py -settingfile configuration_product.json
```

For computing the near-real-time datasets, the genearal command-line is as follows:

```
>> python3 HYDE_DynamicData_HSAF_ASCAT_OBS_NRT.py -settingfile configuration_product.
↪json -time "%Y-%m-%d %H:%M"
```

The **product H14** is generated by the soil moisture assimilation system, using the surface observation from ASCAT that is propagated towards the roots region down to 2.89 m below surface, providing estimates for 4 layers (thicknesses 0.07, 0.21, 0.72 and 1.89 m). The ECMWF model generates soil moisture profile information according to the Hydrology Tiled ECMWF Scheme for Surface Exchanges over Land (HTESSEL). The product is available at a 24-hour time step, with a global daily coverage at 00:00 UTC.

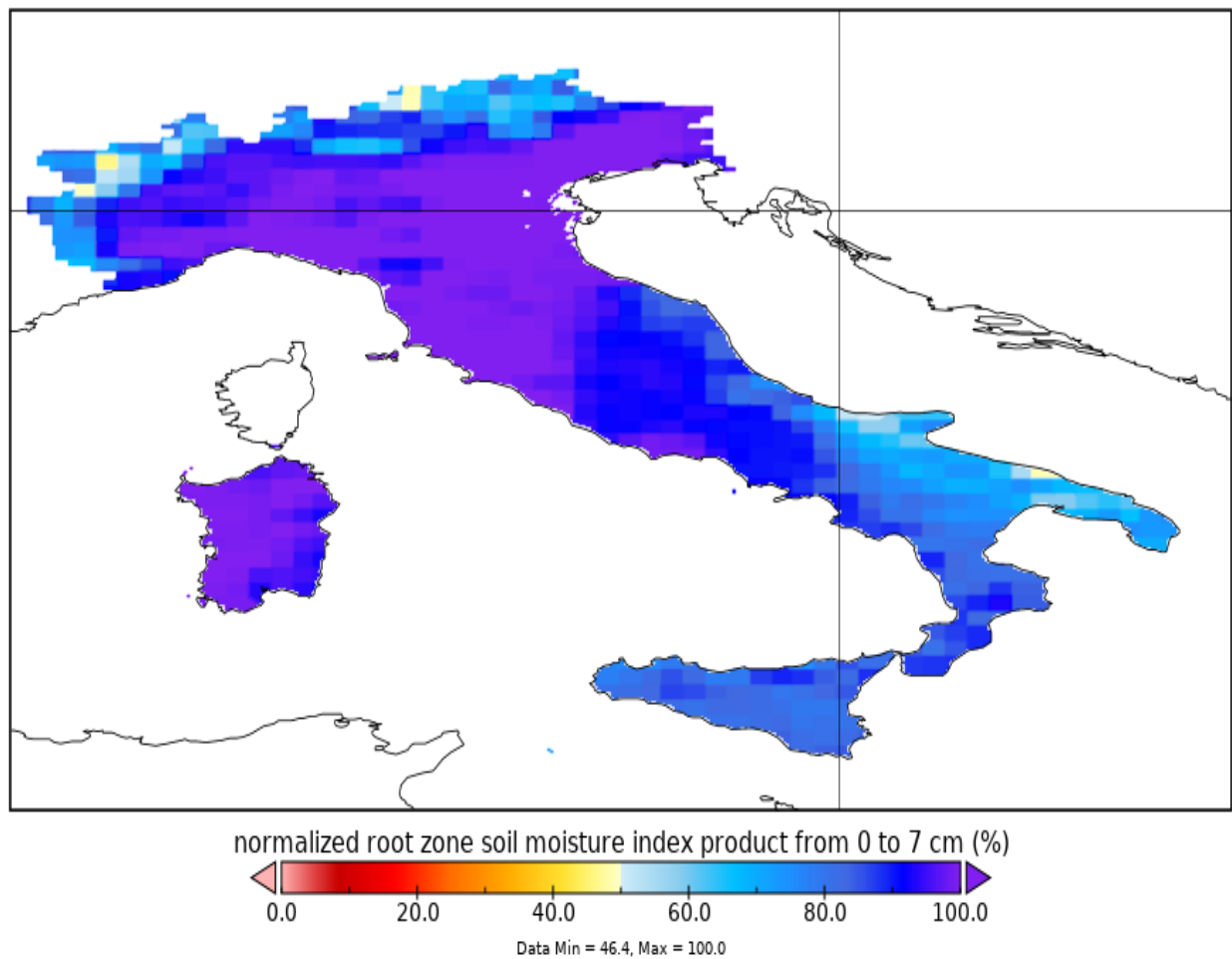Fig. 10: Example of ASCAT H14 root zone soil moisture 0 - 7 cm.

The product H14 is produced in a continuous way in order to ensure the time series consistency of the product (and also to provide values when there is no satellite data, from the model propagation). The SM-DAS-2 product is the first global product of consistent surface and root zone soil moisture available NRT for the NWP, climate and hydrological communities.

The **ASCAT MOD applications**, based on **H14** product, are develop in order to compute the time-series of SSM and SWI both for the **data record** (DR) and the **near-real-time** (NRT) datasets.

General use of ASCAT MOD application for computing data record dataset is reported below:

```
>> python3 HYDE_DynamicData_HSAF_ASCAT_MOD_DR.py -settingfile configuration_product.json
```

For computing the near-real-time datasets, the genearal command-line is as follows:

```
>> python3 HYDE_DynamicData_HSAF_ASCAT_MOD_NRT.py -settingfile configuration_product.
→json -time "%Y-%m-%d %H:%M"
```

The configuration files for all H-SAF products have to be filled in order to correctly set the algorithms.

2.2.2 MODIS

The MODIS Snow Cover products are generated using the MODIS calibrated radiance data products (MOD02HKM and MYD02HKM), the geolocation products (MOD03 and MYD03), and the cloud mask products (MOD35_L2 and MYD35_L2) as inputs. The MODIS snow algorithm output (MOD10_L2 and MYD10_L2) contains scientific data sets (SDS) of snow cover, quality assurance (QA) SDSs, latitude and longitude SDSs, local attributes and global attributes. The snow cover algorithm identifies snow-covered land; it also identifies snow-covered ice on inland water. Further information are available on MODIS Snow Cover WebSite.

General use of MODIS Snow Cover application is reported below:

```
>> python3 HYDE_DynamicData_MODIS_Snow.py -settingfile configuration_product.json
```

2.3 Numerical Weather Prediction Applications

2.3.1 Deterministic mode

The procedure of numerical weather prediction data consists in an application written in python3 available in HyDE package in "/hyde-master/apps/{nwp_name}" named "HYDE_DynamicData_NWP_{nwp_name}.py"; for configuring this procedure, a JSON file usually named "hyde_configuration_nwp_{nwp_name}.json" has to be filled in all of its part. For example, in case of wrf is used the name of the HyDE application will be updated using "wrf" to complete names of folders, procedures and configuration files.

General use of the nwp application is reported below:

```
>> python3 HYDE_DynamicData_NWP_{nwp_name}.py -settings_file hyde_configuration_nwp_{nwp_
→name}.json -time "%Y-%m-%d %H:%M"
```

For example, if the application is about wrf at 2019-05-22 13:00:

```
>> python3 HYDE_DynamicData_NWP_WRF.py -settings_file hyde_configuration_nwp_wrf.json -
→time "2019-05-22 13:00"
```

In the configuration file, users can change settings of variables and timing information. Time fields have to be changed according with nwp timestep resolution and forecast length.

```
"time": {
  "time_forecast_step": 48,
```

(continues on next page)

Fig. 11: Example of variable list for an nwp output file.

```
  "time_forecast_delta": 3600,
  "time_observed_step": 0,
  "time_observed_delta": 0
},
```

About variables and their attributes, users can update name, type and computational methods of each variable. Scale factor, units, missing value and fill value are requested to get and filter values.

```
"air_temperature_data": {
  "id": {
    "var_type": ["var3d", "istantaneous"],
    "var_name": ["T2"],
    "var_file": "wrf_data",
    "var_method_get": "get2DVar",
    "var_method_compute": "computeAirTemperature"
    },
  "attributes": {
    "ScaleFactor": 1,
    "Missing_value": -9999.0,
    "_FillValue": -9999.0
    }
}
```

Output format and features of the variable have to be set too. The users can specify, for instance, units, format and ancillary variables. The function to write data in netCDF4 format can be changed according with data expected by the other applications in forecasting chain.

```
"air_temperature_data":{
  "id": {
    "var_type": ["var3d", "istantaneous"],
    "var_name": "AirTemperature",
    "var_file": "wrf_product",
    "var_colormap": "air_temperature_colormap",
    "var_method_save": "write3DVar"
    },
  "attributes": {
```

```
    "long_name": "",
    "standard_name": "",
    "ancillary_variables": ["T2"],
    "units": "C",
    "Format": "f4",
    "description": "Temperature at 2m"
    }
}
```
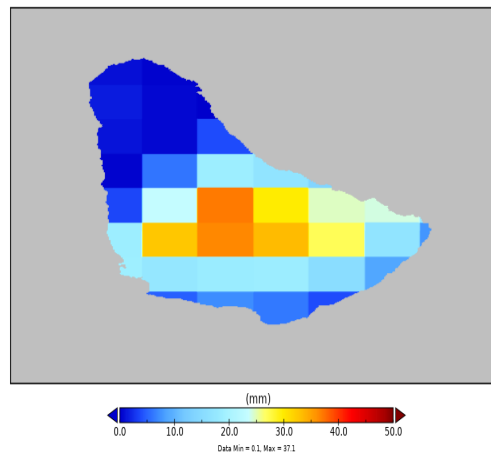

Fig. 12: Example of rain map for Barbados island.
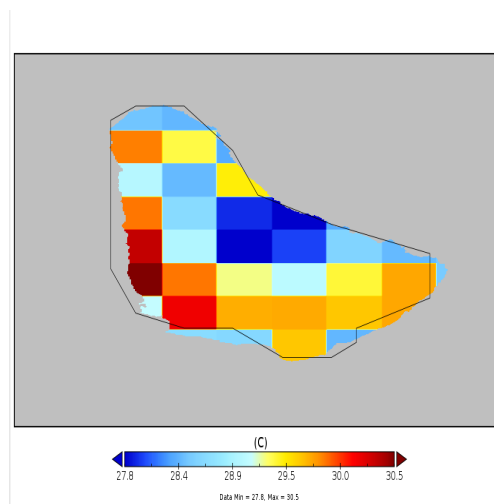
Fig. 13: Example of air temperature map for Barbados island.

2.3.2 Probabilistic mode - Rainfarm model

The procedure of Rainfarm model consists in an application written in python3 available in HyDE package in "/hyde-master/apps/rfarm" named "HYDE_Model_RFarm_{nwp_name}.py"; for configuring this procedure, a JSON file usually named "hyde_configuration_rfarm_{nwp_name}.json" has to be filled in all of its part. For example, in case of WRF numerical model is used the name of the HyDE application will be updated using "wrf" to complete names of folders, procedures and configuration files.

General use of the nwp application is reported below:

```
>> python3 HYDE_Model_RFarm_{nwp_name}.py -settings_file hyde_configuration_rfarm_{nwp_
→name}.json -time "%Y-%m-%d %H:%M"
```

For example, if the application is about wrf at 2019-05-22 13:00:

```
>> python3 HYDE_Model_RFarm_WRF.py -settings_file hyde_configuration_rfarm_wrf.json -
→time "2019-05-22 13:00"
```

latitude	latitude	Geo2D
longitude	longitude	Geo2D
Rain		Geo2D
Terrain	geometric height	Geo2D
time	time	1D

Fig. 14: Example of variable list for an nwp output file.

Obviously, according with NWP version and features, users have to set the reliable spatial scale of LAM Lo (cs_sf in the configuration file) and the reliable temporal scale of LAM T0 (ct_sf in the configuration file). The number of ensembles have to be set, too.

```
"parameters": {
    "ensemble": {"start": 1, "end": 30},
    "ratio_s": 6,
    "ratio_t": 1,
    "slope_s": null,
    "slope_t": null,
    "cs_sf": 2,
    "ct_sf": 2,
    "multi_core": false,
    "domain_extension": 0,
    "tmp": true
}
```

In the configuration file, users can change settings of variables and timing information. Time fields have to be changed according with nwp timestep resolution and forecast length.

```
"time": {
    "time_forecast_period": 48,
    "time_forecast_frequency": "H",
    "time_observed_period": 0,
    "time_observed_frequency": "H",
    "time_rounding": "H"
}
```
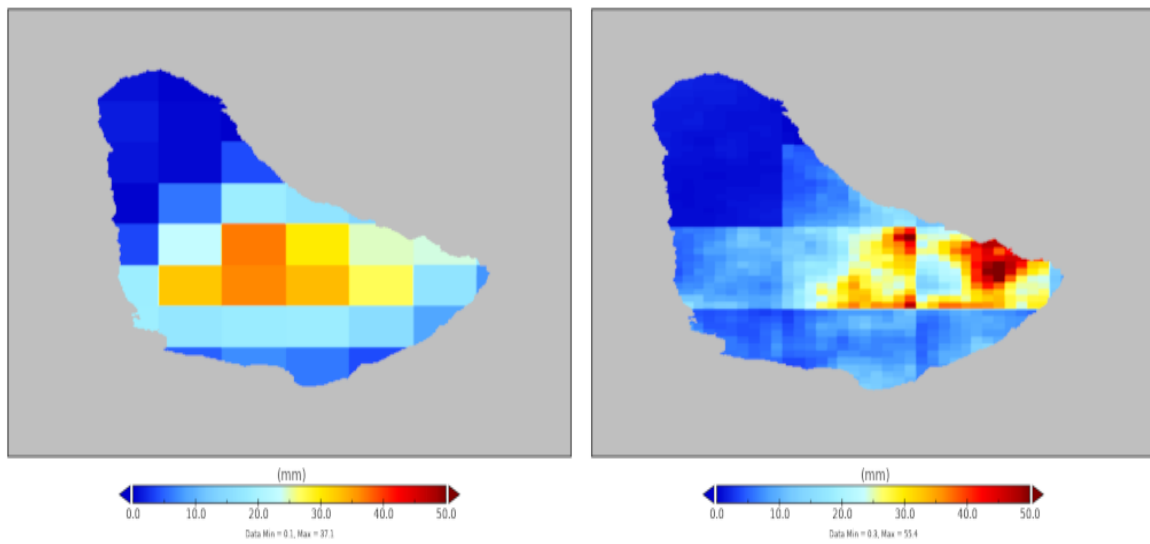
Fig. 15: Example of comparison between nwp map and disaggregated map for Barbados island.

Further details on the Rainfarm model are reported in AnnexB.

EXECUTABLES

In this section, **executables** for different data type are presented; each of them run procedures to prepare data using a **python3** or **bash** scripts. These scripts are the link between the user machine and the applications developt for HyDE package. For example, they are used **to import** environmental variables (such as **PATH** and **LD_LIBRARY_PATH**) needed to execute applications or **to set up** jobs using **cron** scheduler. All applications are stored in the bin folder in the root of the package; an example of bin tree is reported:

```
hyde-master
├── apps
├── **bin**
│   ├── downloader
│   │   ├── ground_network
│   │   │   ├── rs
│   │   │   └── ws
│   │   ├── nwp
│   │   │   ├── lami
│   │   │   └── wrf
│   │   └── satellite
│   │       ├── modis
│   │       └── hsaf
│   ├── runner
│   │   ├── ground_network
│   │   │   ├── rs
│   │   │   └── ws
│   │   └── nwp
│   │       ├── lami
│   │       └── wrf
│   ├── utils
│   └── ...
├── docs
├── src
│   ├── common
│   └── hyde
├── test
├── AUTHORS.rst
├── CHANGELOG.rst
├── LICENSE.rst
└── README.rst
```

As shows in the previous part, scripts are divided in downloaders, runners and generic utils. The **Downloaders** are used to get data from ftp, databases and other external sources. The **Runners** are used to execute applications to process data used by FloodPROOFS modelling system. The **Utils** are dedicated to solve a generic task to adapt data and procedures

in special case to guarantee, for example, an operational use.

3.1 Downloader

As previously said, the first step of procedures is about the downloading/collecting data from a database or a data repository; to perform this step, the downloaders used in HyDE package are usually written in Python3 or Bash programming languages. In Hyde package at the moment are available scripts to download/collect:

- ground network data;

- numerical weather prediction data;

- satellite data.

3.1.1 Python procedure

The procedures developt using Python3 are organized in two part:

- a **main file** in Python3 format;

- a **configuration file** in **JSON** format.

For example, the extraction of weather stations data is performed by an application available in HyDE in "/hyde-master/bin/downloader/ground_network/" named "hyde_downloader_{database_name}_ws.py"; for configuring this procedure, a JSON file usually named "hyde_downloader_{database}_ws.json" has to be filled in all of its part. The name of used database has to be specified by the users according with their settings. At the end, a output CSV file will be created with all data for each variable and for every timestep.

Commonly, the script is organized as follows:

```python
#!/usr/bin/python3 # declaration of interpreter

"""
Procedure description
"""

# Libraries
import logging
import numpy
import os
import time
from argparse import ArgumentParser
from json import load

# Script main
def main():

    # Info
    logging.info(' script start ... ')

    # Get algorithm arguments
    with open(get_args(), "r") as file_obj:
        file_data = load(file_obj)
```

(continues on next page)

```python
    ...

    # Call func_01
    result_1 = func_01(arguments_1)

    ...

    # Call func_02
    result_2 = func_02(result_1, arguments_2)

    if res2 is None:
        logging.warning(' res2 is None!')
    ...

    try:
        result_03 = func_03()
    except:
        logging.error('func_03 not implemented')
        raise NotImplementedError('check your code to add func_03')

    ...

    # Info
    logging.info(' ... script end')

# Method to set logging information
def set_log(logger_file='log.txt',
            logger_msg='%(asctime)s %(name)-12s %(levelname)-8s %(filename)s:[%(lineno)-
→6s - %(funcName)20s()] %(message)s'):

    # Remove old logging file
    if exists(logger_file):
        remove(logger_file)

    # Open logging basic configuration
    logging.basicConfig(level=logging.INFO,
                        format=logger_msg,
                        filename=logger_file,
                        filemode='w')

    # Set logger handle
    log_handle_1 = logging.FileHandler(logger_file, 'w')
    log_handle_2 = logging.StreamHandler()
    # Set logger level
    log_handle_1.setLevel(logging.DEBUG)
    log_handle_2.setLevel(logging.DEBUG)
    # Set logger formatter
    log_formatter = logging.Formatter(logger_msg)
    log_handle_1.setFormatter(log_formatter)
    log_handle_2.setFormatter(log_formatter)
    # Add handle to logging
    logging.getLogger('').addHandler(log_handle_1)
```

```python
    logging.getLogger('').addHandler(log_handle_2)


# Method to get script argument(s)
def get_args(settings_file_default='configuration.json'):
    parser_obj = ArgumentParser()
    parser_obj.add_argument('-settings_file', action="store", dest="settings_file")
    parser_values = parser_obj.parse_args()

    if oParserValue.settings_file:
        filename = parser_values.settings_file
    else:
        filename = settings_file_default

    return filename

def func_01(args):
    ...
    return res1

def func_02(data, args):
    ...
    return res2

# Call script from external library
if __name__ == "__main__":
    main()
```

3.1.2 Bash procedure

The procedures developt using **Bash** consist in a **script** with settings specified in the first part of the code. An example
of Bash script is reported below:

```bash
#!/bin/bash -e # declaration of interpreter

# Script information
script_name='HYDE DOWNLOADER - HSAF PRODUCT PRECIPITATION H03B'
script_version="2.0.2"
script_date='2019/10/07'

# Script argument(s)
data_folder_raw="/source/h03b/%YYYY/%MM/%DD/"
days=5
proxy="http://xxx.xxx.xxx.xxx:xxxx"

ftp_url="ftp.meteo.it"
ftp_usr="********"
ftp_pwd="********"
ftp_folder="/products/h03B/h03B_cur_mon_data/"

str_grib='fdk'
```

```
str_tmp_step1='fdk'
str_tmp_step2='fulldisk'
str_nc='europe'

list_var_drop='x,y'
var_rename_1='IRRATE_no_level,IRRATE_Data'
var_rename_2='IRRATE_surface,IRRATE_QualityIndex'

domain_bb='-15,30,30,60'

wgrib2_exec="/bin/wgrib2"
ncks_exec="//bin/ncks"
ncrename_exec="/bin/ncrename"
cdo_exec="/bin/cdo"

...
```

In general, the downloaders written in Bash are available in HyDE in "/hyde-master/bin/downloader/{application_type}/" named "hyde_downloader_{application_type}_{product_type}.sh". Usually, some external applications, such as lftp, rsync, wgrib2, cdo, nco are used to execute operations over variables to change resolutions. domain reference, projections and so on.

3.2 Runner

Once the static and dynamic datasets are collected, in HyDE package some scripts written in Bash programming are available to run applications (usually named runners). In HyDE package, they are available in "/hyde-master/bin/runner/" and are divided following as follows:

- ground network executables;

- numerical weather prediction executables;

- satellite executables.

The procedures develop using Bash consist in a **script** with settings specified in the first part of the code. An example of a script is reported below:

```
#!/bin/bash -e

#-------------------------------------------------------------------------------
↪-
# Script information
script_name='HYDE RUNNER - HSAF ASCAT OBS [METOP-SM] REALTIME - NRT'
script_version="1.0.0"
script_date='2019/08/29'

script_folder='/home/hyde-master/'

virtual_env_folder='/home/libs_python3/bin/'
virtual_env_name='libs_env_python3'
#-------------------------------------------------------------------------------
↪-
```

```
#-----------------------------------------------------------------------------------
↪-
# Get file information
script_file='/home/hyde-master/apps/hsaf/soil_moisture/HYDE_DynamicData_HSAF_ASCAT_OBS_
↪NRT.py'
setting_file='/home/hyde-master/apps/hsaf/soil_moisture/hyde_configuration_hsaf_ascat_
↪obs_nrt_realtime.json'

# Get information (-u to get gmt time)
time_now=$(date -u +"%Y%m%d%H00")
#-----------------------------------------------------------------------------------
↪-

#-----------------------------------------------------------------------------------
↪-
# Add path to pythonpath
export PYTHONPATH="${PYTHONPATH}:$script_folder"
# Add virtual env
export PATH=$virtual_env_folder:$PATH
source activate $virtual_env_name
#-----------------------------------------------------------------------------------
↪-

# -----------------------------------------------------------------------------------
↪-
# Info script start
echo "␣
↪======================================================================================"
echo " ==> "$script_name" (Version: "$script_version" Release_Date: "$script_date")"
echo " ==> START ..."
echo " ==> COMMAND LINE: " python $script_file -settingfile $setting_file -time $time_now

# Run python script (using setting and time)
python3 $script_file -settingsfile $setting_file -time $time_now

# Info script end
echo " ==> "$script_name" (Version: "$script_version" Release_Date: "$script_date")"
echo " ==> ... END"
echo " ==> Bye, Bye"
echo "␣
↪======================================================================================"
# -----------------------------------------------------------------------------------
↪-
```

Usually, as we can see in the previous snippet, the **runners** are **configured** using a **python virtual environment** created by the miniconda system. It is possible to install all libraries and applications using the virtual environments of FloodProofs modelling system The configuration of miniconda virtual environment is performed in the runners by the following code lines:

```
...
```

```
# Add path to pythonpath
export PYTHONPATH="${PYTHONPATH}:$script_folder"
# Add virtual env
export PATH=$virtual_env_folder:$PATH
source activate $virtual_env_name

...
```

3.3 Utils

The utilities executables are ancillary scripts that are used to:

- **change** the default results of an HyDE applications;

- **manage** the content of the modelling system folders (deleting, syncroning and so on);

- **monitor** the machine memory (RAM, process, swap and so on);

- **synchronize** the data between different machines;

- …

In HyDE package, they are available in "/hyde-master/bin/utils/". Usually, they are used to solve a **specific problem** or request in developing FloodProofs modelling system.

3.4 Scheduler and Environment configuration

3.4.1 Scheduler configuration - Crontab

The first step to configure the executables file (downloaders and runners) is to use the Crontab **(CRON TABle)** scheduler to automatically launch HyDE executables. The Crontab is a file which contains the schedule of cron entries to be run and at specified times. File location varies by operating systems. Moreover, cron job or cron schedule is a specific set of execution instructions specifying day, time and command to execute. Crontab can have multiple execution statements.

```
# HYDE - DATAPROCESSING - GroundNetwork - WeatherStations
25 * * * * . $HOME/.profile; $HOME/hyde-master/bin/runner/ground_network/hyde_runner_
→groundnetwork_ws.sh
# HYDE - DATAPROCESING - NWP  WRF
30 7,8,19,20 * * * . $HOME/.profile; $HOME/hyde-master/bin/runner/wrf/hyde_runner_nwp_
→wrf.sh
# HYDE - DATAPROCESING - RFARM WRF
35 7,8,19,20 * * * . $HOME/.profile; $HOME/hyde-master/bin/runner/rfarm/hyde_runner_
→rfarm_wrf.sh
# HYDE - UTILS - CLEANER - HOURLY/DAILY
0 0-23 * * * . $HOME/.profile; $HOME/hyde-master/bin/utils/hyde_utils_cleaner_folders.sh
```

For editing or updating Crontab file, the users have to use the following command in the terminal:

```
>> crontab -e
```

3.4.2 Environment configuration - profile

The second step, to configure the HyDE executables for performing data processing or models running, is to set the profile file in the machine environment. The **.profile** file in Linux comes under the System startup files. File like /etc/profile controls variables for profile of all users of the system whereas, .profile allows you to customize your own environment. The .profile file is present in the user home ($HOME) directory and lets you customize user's individual working environment. The .profile file contains user's individual profile that overrides the variables set in the /etc/profile file.

An example of .profile file is reported below:

```
# Export library for flood proofs modelling system
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HOME/library/zlib-1.2.11/lib/
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HOME/library/hdf5-1.8.17/lib/
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HOME/library/netcdf-4.1.2/lib/
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HOME/library/geos-3.5.0/lib/
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HOME/library/proj-4.9.2/lib/

# Export binaries for flood proofs modelling system
export PATH=$HOME/library/hdf5-1.8.17/bin:$PATH
export PATH=$HOME/library/zlib-1.2.11/bin:$PATH
export PATH=$HOME/library/netcdf-4.1.2/bin:$PATH
export PATH=$HOME/library/geos-3.5.0/bin:$PATH
export PATH=$HOME/library/proj-4.9.2/bin:$PATH
```

In this example, the user filled .profile with export commands for LD_LIBRARY_PATH and PATH environment variables. After this step, the executbales are ready to run using a particular or a different version of a library. Sometimes, libraries set by default in a linux machine are not enough to solve a specific problem.

FOUR

EXAMPLES

As previously said, the FloodProofs modelling system is 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 Albania, Barbados, Bolivia, Guyana and Lebanon. It is also used for water management purposed by Italian hydro-power companies (Compagnia Valdostana delle Acque, CVA, ERG Power Generation). It is useful for managing hydro-meteorological hazard alerts and warnings for Civil Protection purposes. The main priority is the defence of the citizens, their goods and, in general, the safeguard of all that can be damaged by the ground effects of intense precipitation events. Water management companies employ this forecasting chain mainly for preventing damages to the hydraulic structures in the case of extreme precipitation events.

4.1 Italy

description here

4.2 Barbados

The FloodProofs modelling system was implemented linking numerical weather predictions (CIMH WRF 4km), a stochastic rainfall procedure (RainFarm), real-time observations (rain gauges) and a fully distributed hydrological model (HMC). The flood forecasting chain is based on operational run of the HMC model.

In the actual version of the flood forecasting chain, the discharges are generated by the HMC model using the following sources of forcing data:

- CIMH WRF 4km, whenever a new run is available (deterministic forecast run);

- rain gauges observation (nowcasting run);

- rain gauges observation (initial conditions run)

In addition, to improve the forecasting of rainfall events, the probabilistic part was added to accomplish operational probabilistic discharge forecast:

- RainFARM based on CIMH WRF 4km, whenever a new run is available (probabilistic forecast).

The food forecasting system is installed at the CIMH server room. It takes advantage of the existing CDP which stores and makes available all hydro-meteorological data to the different simulation lines. The workflow is shown in the following figure. Automated weather station data for time intervals ranging from 1 – 10 minutes are collected and stored in the CDP database. Similarly, new WRF outputs 00Z and 12Z runs are ingested into the CDP.

These data are utilized in the flood forecasting chain when required. The combination of weather station data and WRF data are utilized to run FloodProofs operational chain consisting of the HMC model and all required components. To perform a probabilistic discharge forecast we feed the ensemble of high-resolution fields provided by RainFarm to the
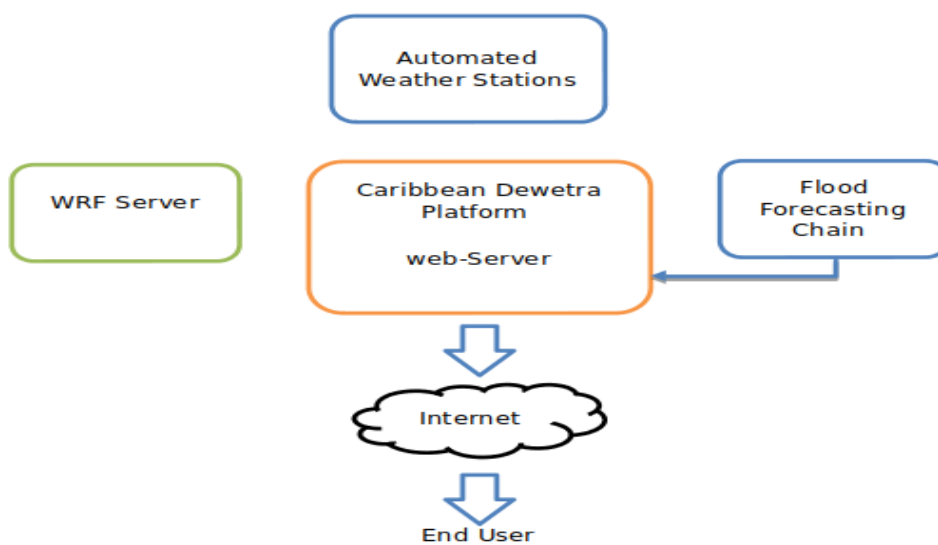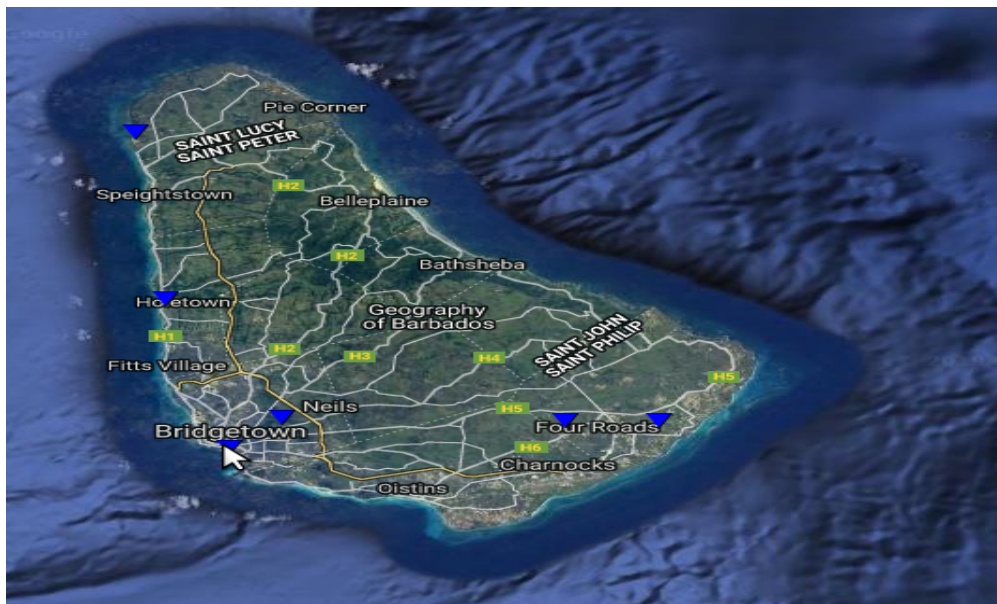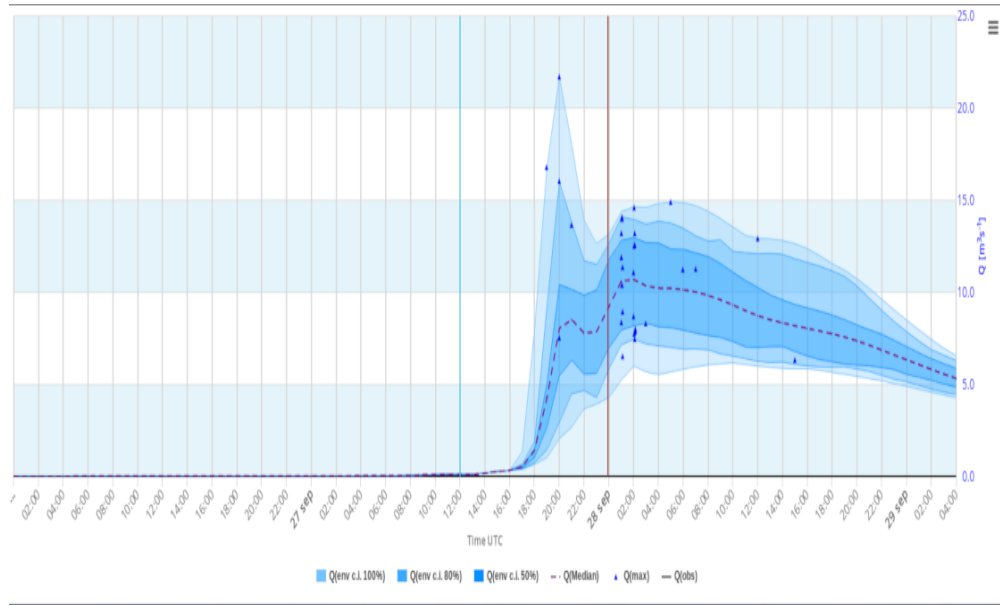
HMC model. The benefits of applying RainFarm to the estimate of the uncertainty in flood predictions over small and medium catchments, becomes crucial for basins with areas smaller than a few hundred square kilometers. The outputs of FloodProofs for a probabilistic chain are reported in the figure above.

4.3 Bolivia

The FloodProofs system modellling run operationally at SENAMHI, and results are available to all the institutions involved in flood management at local and national level. The main components are a limited area meteorological model, which provides future rainfall fields, a stochastic downscaling model for rainfall and HMC fully distributed hydrological model, which transform rainfall into run-off. Flood-PROOFS ingests data and manages the model workflow needed for the hydro-meteorological forecast: meteorological models, real-time meteorological stations data, real-time data of the operation of regulating hydraulic structures and satellite data are considered in this process. It provides a quantitative evaluation of discharge and peak flow and evaluates the probability to exceed critical thresholds in critical outlet sections of the catchments.

The system workflow is illustrated in figure above. Two different sources of forecast can feed the meteorological chain: a QPF by meteorological models (e.g. IFS, GFS, COSMO-I7, WRF) or the predictions issued by expert regional forecasters. The forecasters take into account different meteorological models, their knowledge of the territory and its climatic features to express QPFs in terms of accumulated rainfall on predefined areas and durations.

In both cases, QPFs are downscaled to a proper spatial and temporal scale using a stochastic downscaling model: RainFarm. It generates high-resolution (1 km and 30 min) precipitation ensemble forecast. The same approach, with some adaptation is applied to QPF given by regional forecasters. In both cases, it gives an ensemble of 50 downscaled precipitation forecast scenarios at the small resolution needed for hydrological applications. The third ingredient is the HMC hydrological model needed to simulate the streamflow caused by the predicted precipitation event. It is a continuous, distributed and physically based hydrological model able to reproduce the spatial-temporal evolution of soil moisture, energy fluxes, surface soil temperature, evapotranspiration and discharge. It was designed to find a balance between a detailed description of the physical processes and a robust and parsimonious parameterisation. Continuum was proved to be a model suitable for properly benefit of satellite information. Since the distinction between rain and snow is crucial for issuing a reliable flood warning in mountain environment, FloodProof implements a procedure named Snow Multidata Mapping and Modelling - S3M able to determine whether the precipitation is liquid or solid and models snow melt rate. Hence, S3M gives to the HMC the precipitation fields that take into account snow melting or snow accumulation. S3M uses, in a data assimilation framework, the near-real-time MODIS data and real-time snow
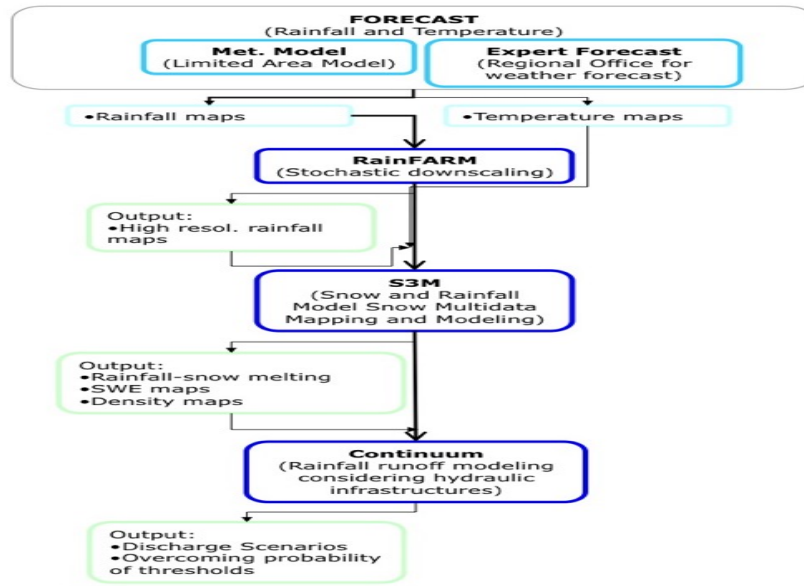
depth measurements. MODIS data are used for the estimation of the Snow Cover Area: the interpolation of the real-time snow depth is performed over pixels marked as "snow" using a multi-linear regression on geomorphologic (e.g. altitude, aspect, slope) and climatologic parameters. The main outputs produced by the computational system are both a probabilistic and deterministic discharge forecast in a defined number of outlet sections of the chosen catchments.
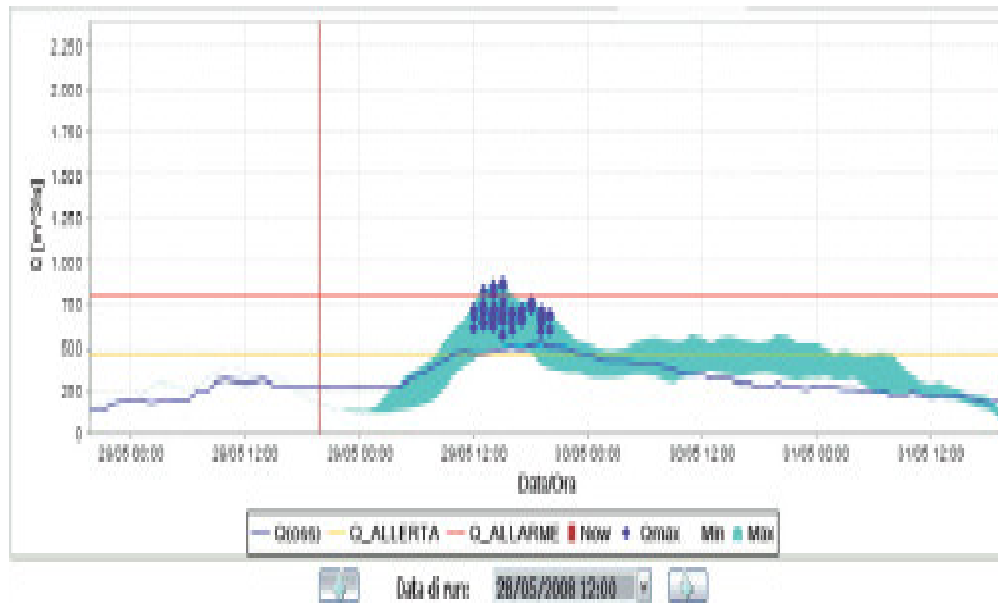
DEVELOPERS

- 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>

ANNEX A - HYDROLOGICAL MODEL CONTINUUM

Hydrological Model Continuum (HMC) is a distributed model based on a space-filling representation of the network, directly derived from a DEM. The DEM resolution coincides with the model resolution. The mass and energy balances are solved at cell scale referring to a scheme with two soil layers (subsurface and water table) and one vegetation layer (interception). The overland and channel flow are described by a linear and a non-linear tank schematization respectively.
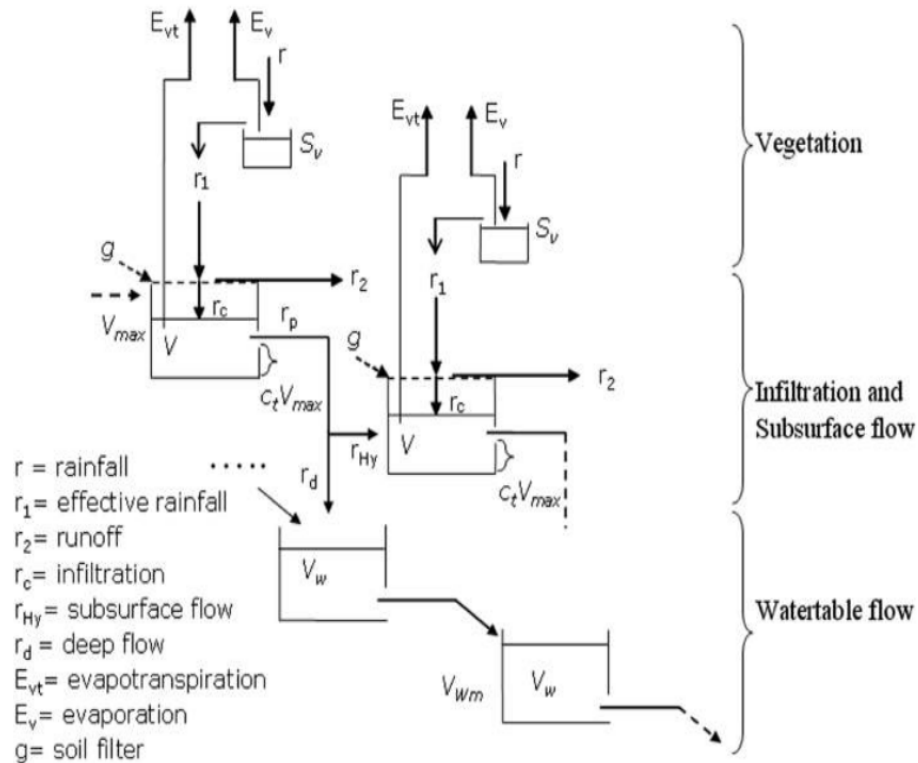
Fig. 1: Sketch of flux partition in the HMC model. Two consecutive pixels are illustrated.

The main characteristics of the presented model and its advantages respect most of the existing models can be summarized as follows:

1. A reduced number of territorial data is needed for the implementation (DEM and CN maps are the essential information needed).

2. There are a reduced number of calibration parameters reducing the problem of parameter identification

3. The capability of exploiting the possibilities offered by remote sensing (e.g. satellite data, better clarified in the following sections) smoothing the problem of equifinality by augmenting the model's constraints 4 The capability of modeling a large number of state variables, similarly to more complex models

Six model parameters need calibration on the basis of input-output time series: cf, ct, uh, uc, Rf, VWmax (see table below). The first two parameters cf, and ct mainly rule the generation of runoff and the movement of water in the different soil layers, while uh and uc control the surface water motion, VWmax represents the maximum storage capacity of the aquifer and Rf summarizes the effect of soil porosity as well as of the ratio between vertical and horizontal saturated soil conductivity.

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

Further details on the HMC are reported in library hmc documentation.

ANNEX B - RAINFARM

Rainfarm belongs to the family of metagaussian models and it is based on a nonlinear transformation of a linearly correlated process. This approach is closely related to the Turning Bands Method and has been used both for satellite-based rainfall measurement validation and for stochastic rainfall modelling.

The model is able to generate small-scale rainfall fields that take into account not only the total amount of precipitation predicted by the meteorological model but also its linear correlation structure and the position of the main rainfall patterns. Due to the straightforward link between the model parameters and the large-scale field, this model is suitable for operational downscaling procedures.
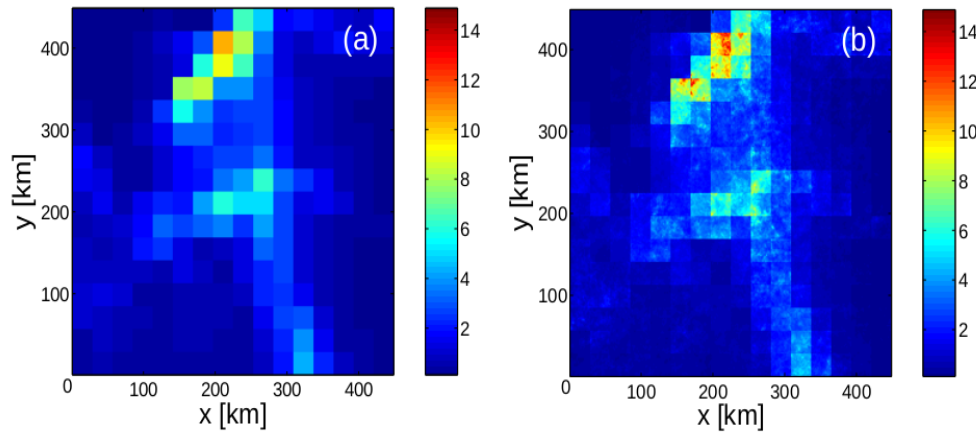

Fig. 1: Time average of the original, LAM field panel (a) and of one realization of the stochastic field generated by the Rainfarm panel (b). The values indicate the average precipitation in mm/h.

Rainfarm uses the spectral information of large-scale meteorological predictions and generates fine resolution precipitation fields by propagating this information to smaller scales. The basic idea is to reconstruct the Fourier spectrum of the small-scale precipitation field by preserving the LAM information at the scales where we are confident in the meteorological prediction. The rainfall field is seen as the superposition of a finite number of harmonics with amplitudes decreasing as spatial and temporal scales become smaller. For a given realization (the predicted field at hand) the harmonics at large scales are assumed to be well predicted by the meteorological model, and should be preserved. A major concern is to figure out which are the scales below which the forecasted fields are considered to be unreliable.

Table 1: Summary of the model Rainfarm parameters

Parameter	Description
L_0	Reliable spatial scale of LAM
T_0	Reliable temporal scale of LAM

As shown in the previous section, the Rainfarm model has four free parameters, named L_0, T_0, α and β. L_0 and T_0 represent the spatial and temporal resolutions at which the precipitation prediction is considered reliable and they should be fixed by the user depending on the numerical model considered. The spectral parameters α and β are estimated in real-time from the LAM power spectrum. The RainFARM works as follows. The aggregation of the LAM field on spatial and temporal scales L_0 and T_0, generates the starting field, called P. From this coarse-grained prediction RainFARM produces downscaled fields by following these steps:

1. Computation of the Fourier transform of the coarse-grained LAM-predicted field $P(X, Y, T)$. This procedure generates a spatial-temporal Fourier spectrum $\dot{P}(K_X, K_Y, \Omega)$ defined as follows:

$$\ddot{P}(K_X, K_Y, \Omega) = \sum_{X=0}^{D_X} \sum_{Y=0}^{D_Y} \sum_{T=0}^{D_T} P(X, Y, T) e^{-i(XK_x + YK_y + T\Omega)}$$

Where D_X, D_Y and D_T are the domain sizes in space and time respectively. Here, D_X, D_Y are the wave numbers in the X and Y directions and Ω is the angular frequency. Clearly, $K_X, K_Y <= \pi/L_0$ and $\Omega <= \pi/T_0$, where π/L_0 is the Nyquist wavenumber and π/T_0 is the Nyquist frequency of the field to be downscaled.

2. Estimate of the space-time power spectrum $|\dot{P}(K_X, K_Y, \Omega)^2|$ of the aggregated LAM field P as the square of the modulus of its Fourier transform $\hat{P}$.

3. Extrapolation of the power spectrum $|\hat{P}^2|$ to small scales. To do so, we assume the power spectrum of the rainfall field to have an approximate power-law behavior, consistent with the outcome of several analyses on the structure of precipitation fields. The procedure estimates the spatial and temporal logarithmic slopes (α and β respectively) of the LAM power spectrum, $|\hat{P}|$. For simplicity, we assume isotropy in the two spatial directions.

4. Generation of a Fourier spectrum $\hat{g}(k_x, k_y, \omega)$, defined as:

$$\ddot{g}(k_x, k_y, \omega) = |\hat{g}(k_x, k_y, \omega)| \exp(i\phi)$$

where $\theta(k_x, k_y, \omega)$ are random, uniformly distributed phases. The wavenumbers (K_x) and $k_y)$ and the frequency (ω) range from the scales corresponding to the downscaling domain size to those associated with the downscaling resolution. We use the functional form:

$$|\hat{g}(k_x, k_y, \omega)^2| = (k_x^2 + k_y^2)^{-\alpha/2} \omega^{-\beta}$$

By inverting the Fourier spectrum, , we obtain a Gaussian field defined on the whole range of scales between the domain size and the downscaling resolution, which we normalize to unit variance.

5. Generation of a synthetic precipitation field, $\hat{r}(x, y, t)$, by taking a nonlinear transformation of the Gaussian field q. Here we use the simple transformation:

which leads to a log-normal field $\hat{r}$.

6. We force the synthetic field to be equal to the original field P when aggregated on the scales L_0, T_0 by defining a new field:

Where $\hat{R}$ represents the field aggregated at the scale L_0, T_0. When aggregated on space and time scales larger or equal to L_0 and to T_0, the field behaves exactly as the original field P. The stochastic nature of the downscaled field r is associated with the choice of the set of random Fourier phases. By choosing different sets of random Fourier phases, one can generate a large number of stochastic fields which are all equal to P when aggregated on space and time scales larger than L_0 and T_0, and which are different, but with similar statistical properties, on smaller scales.

$$\widetilde{r}(x,y,t)=\exp[g(x,y,t)]$$

$$r(x,y,t)=\widetilde{r}(x,y,t)\frac{P(X,Y,T)}{\widetilde{R}(X,Y,T)}$$

INDICES AND TABLES

- genindex
- modindex
- search