Itzï Documentation

Release 20.5

Laurent Courty

May 27, 2020

Contents:

Welcome to the documentation of Itzï, a software that allow dynamic simulation of floods. Please see the Itzï's home page.

CHAPTER 1

Installation

1.1 Availability

The python package for Itzï is on pypi. You can browse and download the source code on bitbucket.

1.2 Installation on GNU/Linux

Itzï depends on GRASS GIS 7.8 or above and NumPy. GRASS should therefore be installed in order to use Itzï. NumPy is normally installed along GRASS. All other dependencies are installed by pip.

To install Itzï, you'll need to have the Python installation software *pip* installed. On Ubuntu, the package is called *python-pip* and is installed as follow:

```
sudo apt-get install python-pip
```

1.2.1 Installation for a single user

This is useful when you do not have root access on the computer.

To download and install the last version of Itzï using pip:

```
pip install itzi --user
```

If Itzï is already installed and you want to update it to the last version:

```
pip install itzi --user --upgrade
```

If you prefer to download and install Itzï manually, you can do it that way:

```
tar -xvf itzi-20.5.tar.gz
cd itzi-20.5
python setup.py install --user
```

Note: For a reason not related to Itzï, pip does not always place the Itzï executable in an accessible place. If calling *itzi* returns a *command not found* error, you need to add the installation directory (usually *~/.local/bin*) to your PATH.

1.2.2 Installation for all users

This requires root access. The steps are the same as above, with the addition of the use of sudo:

```
sudo pip install itzi
```

1.3 Installation on Windows

Itzï can be run on Windows 10 using the Windows Subsystem for Linux (WSL). For that, you'll need at least Windows 10 64bits Creators Update.

To install WSL, follow the steps given by Microsoft.

You can then install the prerequisites:

```
sudo apt-get update
sudo apt-get install grass-dev grass-core python-pip
```

Once everything is installed, the installation steps are the same as GNU/Linux.

1.4 Verification of the installation

To check if everything went fine:

```
itzi version
itzi run -h
```

CHAPTER 2

Command line usage

2.1 Run a simulation

```
usage: itzi run [-h] [-o] [-p] [-v | -q] config_file [config_file ...]
```

2.1.1 Positional Arguments

config_file an Itzï configuration file (if several given, run in batch mode)

2.1.2 Named Arguments

-o overwrite files if exist

Default: False

-p activate profiler

Default: False

-v increase verbosity

-q decrease verbosity

2.2 Get the version number

```
usage: itzi version [-h]
```

CHAPTER 3

Tutorial

This tutorial shows how to run a basic Itzï simulation using freely available dataset.

It assumes that GRASS 7 and Itzï are properly installed on your machine and that you possess a basic knowledge of GRASS.

3.1 Get ready

Here we will use the GRASS North Carolina dataset. Please download the GRASS 7 version and extract it in your *grassdata* directory.

Then start GRASS in the PERMANENT mapset.

3.2 Surface modelling

3.2.1 Adjust the region

Fit the lidar elevation raster map and set a resolution of 5m:

```
$ g.region -p raster=elev_lid792_1m@PERMANENT res=5 save=lidar_5m
```

3.2.2 Resample the DEM

Please note that this step is not strictly necessary. The Itzï simulation will be carried out in any case on the defined computational region extent and resolution. However the bilinear interpolation smooth the surface, which prevent high slope values that could occur if using the GRASS default nearest-neighbour sampling.

```
$ r.resamp.interp input=elev_lid792_1m@PERMANENT output=elev_lid792_5m
```

3.2.3 Create a raster mask

Generate a drainage direction map and then create a watershed raster using the outlet point coordinates:

```
$ r.watershed elevation=elev_lid792_5m drainage=elev_lid792_5m_drainage
$ r.water.outlet input=elev_lid792_5m_drainage output=watershed coordinates=638888,
↪220011
```

Create a raster mask to prevent calculation outside of the watershed:

```
$ r.mask rast=watershed
```

3.2.4 Create boundary condition maps

Create a vector map with the watershed outlet point:

```
$ echo '638888|220011' > watershed_out.txt
$ v.in.ascii input=watershed_out.txt output=watershed_out
```

Using this vector map, create two raster maps for the boundary conditions. The first with a value corresponding to the type of condition, here 4 corresponds to a fixed water depth inside the domain. The second being the value of the depth wanted, here 0.

```
$ v.to.rast input=watershed_out type=point output=bctype use=val value=4
$ v.to.rast input=watershed_out type=point output=bcvalue use=val value=0
```

3.2.5 Create rainfall and friction maps

Create maps of uniform rainfall and friction coefficient:

```
$ r.mapcalc exp='rain=100'
$ r.mapcalc exp='n=0.05'
```

3.2.6 Create a parameters file

Create a new parameter file and fill it with the ID of the created maps. It should look like the following:

```
[time]
duration = 02:00:00
record_step = 00:05:00

[input]
dem = elev_lid792_5m@PERMANENT
friction = n@PERMANENT
rain = rain@PERMANENT
bctype = bctype@PERMANENT
bcval = bcvalue@PERMANENT

[output]
prefix = nc_itzi_tutorial
values = h, wse, v, vdir, boundaries
```

(continues on next page)

(continued from previous page)

```
[statistics]
stats_file = nc_itzi_tutorial.csv
```

3.2.7 Run the simulation

Run the simulation:

```
$ itzi run <parameter_file_name>
```

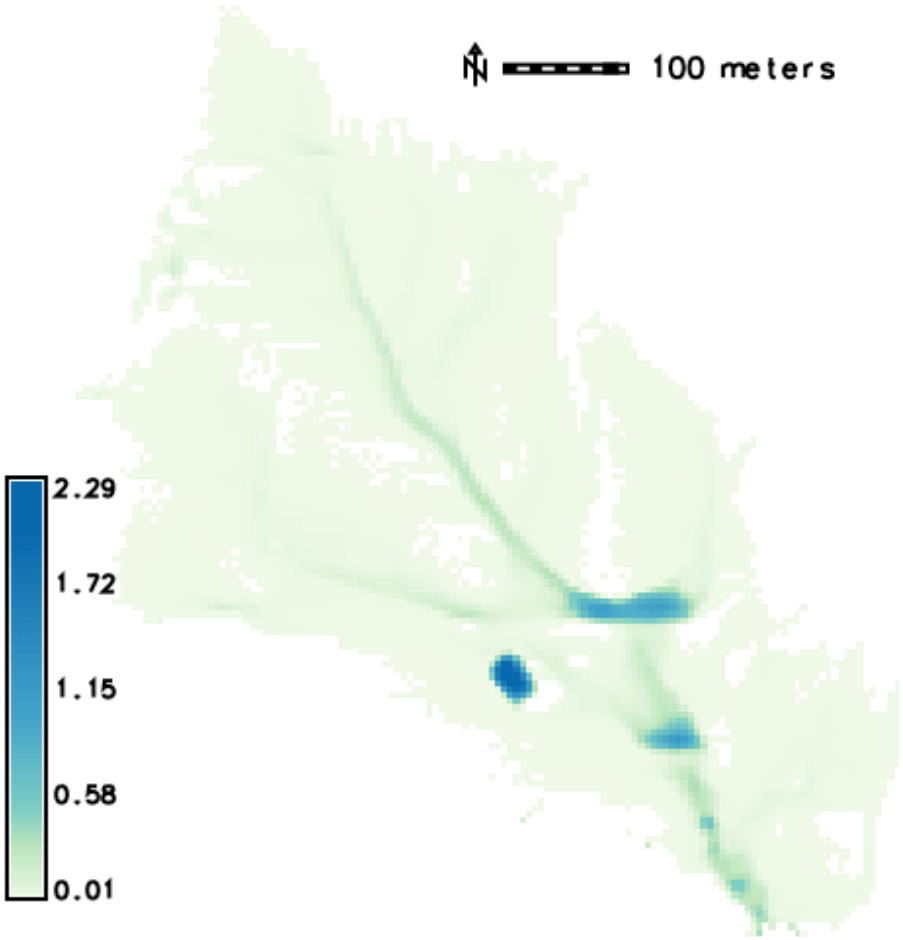
At the end of the simulation, Itzï should have generated five Space-Time Raster Dataset (STRDS) in the form:

```
<prefix>_<variable>
```

The maps contained in those STDRS are following this naming convention:

```
<prefix>_<variable>_<order_number>
```

Here is the example of the map *nc_itzi_tutorial_h_0020*:

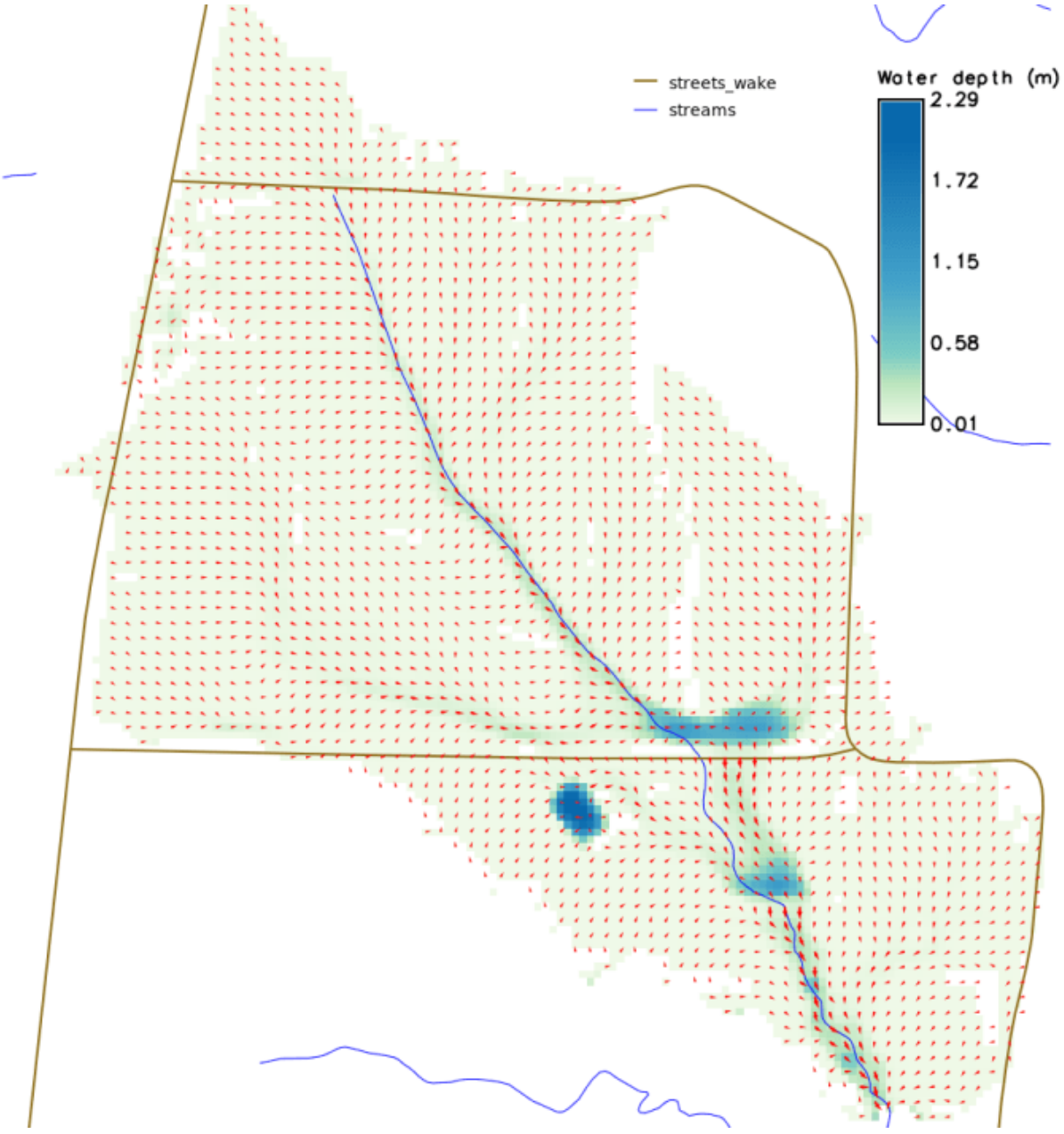


All the results can be processed using the GRASS tools for raster maps and / or space-time dataset. For instance, it is easy to generate an animation of the results using *g.gui.animation*.

3.3 Culvert modelling

New in version 17.7.

As you can notice in the image above, the flow accumulates at some points. One of this accumulation is due to a road that act like a dike and weir. It is better seen when displaying the streets and flow arrows:



One option to solve this problem is to model a culvert using the coupled modelling capacity of Itzï. Itzï is able to run the SWMM drainage model alongside the surface model, and model the interactions between the two.

In the present case, the first step is to decide where the culvert will be located, and annotate:

- The coordinates of the the input and output node,
- the altitude at those points.

Unfortunately, two issues limit the use for the modelling of culvert:

- SWMM needs to have a connected *outfall* node in the network model
- The Itzï coupling code is designed for manhole, not culvert entrance.

We can circumvent those limitations by first, adding an outlet at a higher elevation, linked to the rest of the network by a dummy pipe, and second, set the coupling surface to a large surface (here we'll set it equal to the cell surface).

3.3.1 SWMM configuration file

The description of the drainage network is done in a classic SWMM configuration file. More information could be found in the SWMM user's manual.

Note: *START_DATE* and *START_TIME* are not taken into account during a coupled simulation. The drainage model always starts and stops at the same time than the surface model.

```
[TITLE]
'Wake county culvert'

[OPTIONS]
FLOW_UNITS           CMS
INFILTRATION         HORTON
FLOW_ROUTING         DYNWAVE
START_DATE           01/01/0001
START_TIME           00:00:00
REPORT_START_DATE    01/01/0001
REPORT_START_TIME    00:00:00
END_DATE             01/01/0001
END_TIME             2:00:00
SWEEP_START          01/01
SWEEP_END            12/31
DRY_DAYS             0
REPORT_STEP          00:05:00
WET_STEP             00:00:05
DRY_STEP             01:00:00
ROUTING_STEP         2
ALLOW_PONDING        YES
INERTIAL_DAMPING     NONE
VARIABLE_STEP        .5
LENGTHENING_STEP     0
MIN_SURFAREA         25
NORMAL_FLOW_LIMITED  FROUDE
SKIP_STEADY_STATE    NO
FORCE_MAIN_EQUATION  D-W
LINK_OFFSETS         DEPTH
MIN_SLOPE            0

[JUNCTIONS]
;;          Invert    Max.    Init.   Surcharge  Ponded
;;Name      Elev.     Depth   Depth   Depth      Area
;;--------- --------- ------- ------- ---------- ------
J0          112       0.0     0       0          0
J1          111.4     0.0     0       0          0

[OUTFALLS]
```

(continues on next page)

(continued from previous page)

```
;;          Invert    Outfall  Stage/Table    Tide
;;Name      Elev.     Type     Time Series    Gate
;;--------- --------- -------- -------------- ----
O2          1000      FREE                    NO

[COORDINATES]
;;Node       X-Coord  Y-Coord
;;---------- -------  -------
J0           638752   220262
J1           638769   220233

[CONDUITS]
;;        Inlet  Outlet          Manning Inlet  Outlet
;;Name    Node   Node    Length N       Offset Offset
;;------ ------ ------- ------ ------- ------ ------
C0        J0     J1      34     0.017   0      0
C1        J1     O2      100    0.017   0      0

[XSECTIONS]
;;Link     Shape      Geom1 Geom2 Geom3 Geom4 Barrels
;;-------- ---------- ----- ----- ----- ----- -------
C0         CIRCULAR   1.5   0     0     0     2
C1         CIRCULAR   0.1   0     0     0     1
```

Here, *J0* and *J1* are the input and output nodes of the culvert, and *C0* is the culvert itself. The latter is made of two pipes of 1.5m of diameter. The outfall *O2* and the link *C1* are added to comply with the SWMM rule needing them.

3.3.2 Update the Itzï's parameter file

The parameter file of created in the precedent tutorial could be used and adapted by the addition of the *[drainage]* section, like so:

```
[time]
duration = 00:50:00
record_step = 00:05:00

[input]
dem = elev_lid792_5m@PERMANENT
friction = n@PERMANENT
rain = rain@PERMANENT
bctype = bctype@PERMANENT
bcval = bcvalue@PERMANENT

[output]
prefix = nc_itzi_tutorial_drainage
values = h, v, vdir

[statistics]
stats_file = nc_itzi_tutorial_drainage.csv

[drainage]
swmm_inp = tutorial_drainage.inp
output = nc_itzi_tutorial_drainage

[options]
```

(continues on next page)

(continued from previous page)

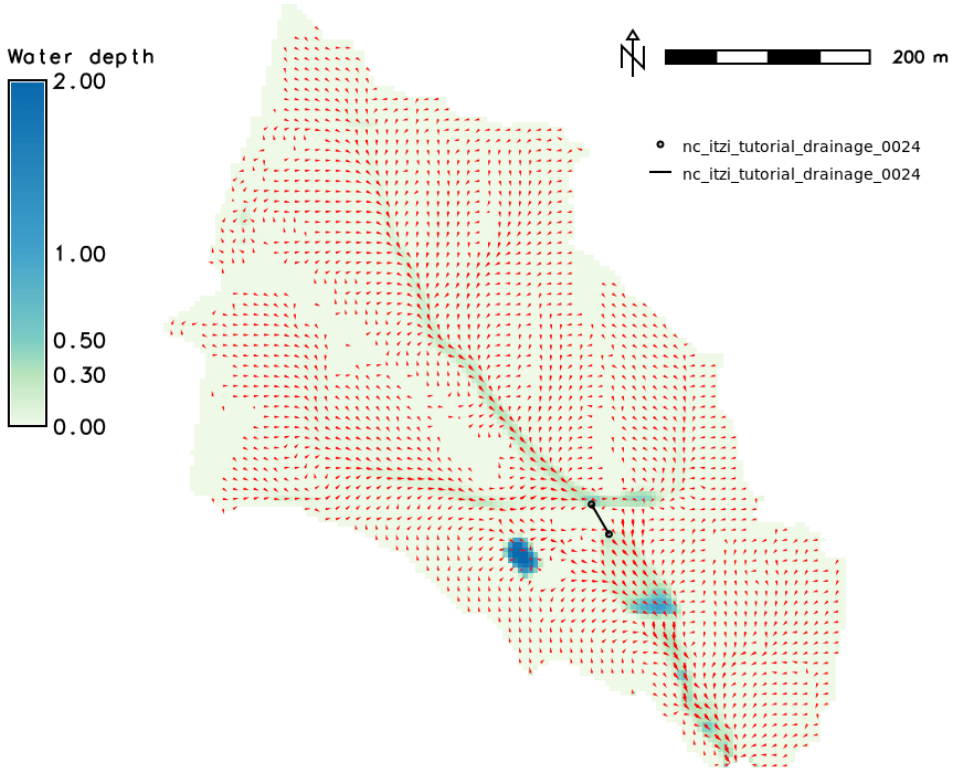
```
cfl = 0.7
theta = 0.9
dtmax = .5
```

Where *swmm_inp* is the path to the SWMM configuration file and *output* is the name of the Space-Time Vector Dataset where the drainage data will be written.

3.3.3 Running the simulation

The simulation is ran the same way as the previous tutorial. Itzï will call SWMM that will in turn loads its own configuration file automatically.

The resulting water depth map is shown here:



The area upstream the road is noticeably less flooded, with a maximum water depth coming down from 1.03m without culvert to 0.45m with culvert. You can use the temporal tools of GRASS to query the evolution in time of the drainage network values. For example, to get the evolution of the flow leaving the upstream node *J0* of the culvert:

```
t.vect.db.select input=nc_itzi_tutorial_drainage@itzi_results columns=outflow where=
↪"node_id=='J0'"

start_time|end_time|outflow
0||0
300||0.0425260290503502
600||1.63466286659241
900||4.20853137969971
1200||4.59034490585327
1500||4.64469814300537
```

(continues on next page)

(continued from previous page)

```
1800||4.6541862487793
2100||4.6692533493042
2400||4.65738391876221
2700||4.66986560821533
3000||4.66973972320557
```

CHAPTER 4

Configuration file

The parameters of a simulation are given through a configuration file in a format similar to Microsoft Windows INI files. An example is given in the tutorial above. The file is separated in sections described below.

4.1 [time]

Simulation duration could be given by a combination of start time, end time and duration. If only the duration is given, the results will be written as a relative time STRDS. In case start time is given, the simulation will use an absolute temporal type.

Keyword	Description	Format
start_time	Starting time	yyyy-mm-dd HH:MM
end_time	Ending time	yyyy-mm-dd HH:MM
duration	Simulation duration	HH:MM:SS
record_step	Time-step at which results are written to the disk	HH:MM:SS

Valid combinations:

- *start_time* and *end_time*
- *start_time* and *duration*
- *duration* only

4.2 [input]

Itzï does not support Lat-Long coordinates. A projected location should be used. The inputs maps could be given either as STRDS or single maps. First, the module try to load a STRDS of the given name. If unsuccessful, it will load the given map, and stop with an error if the name does not correspond to either a map or a STRDS.

The following inputs are mandatory:

- Digital elevation model in meters
- Friction, expressed as Manning's *n*

Keyword	Description	Format
dem	Elevation in meters	map or strds
friction	Manning's *n* (friction coefficient)	map or strds
start_h	Starting water depth in meters	map name
rain	Rainfall in mm/h	map or strds
inflow	Point inflow in m/s (ex: for 20 m3/s on a 10x10 cell, velocity is 0.2 m/s)	map or strds
bctype	Boundary conditions type	map or strds
bcval	Boundary conditions values	map or strds
infiltration	Fixed infiltration rate in mm/h	map or strds
effective_porosity	Effective porosity in mm/mm	map or strds
capillary_pressure	Wetting front capillary pressure head in mm	map or strds
hydraulic_conductivity	Soil hydraulic conductivity in mm/h	map or strds
losses	User-defined losses in mm/h (*new in16.9, renamed in 17.7*)	map or strds

Deprecated since version 17.7: *drainage_capacity* is renamed to *losses*

Deprecated since version 20.5: *effective_pororosity* is renamed to *effective_porosity*

Warning: If the selected input are located in another GRASS mapset than the current one (or the one specified in the [grass] section), you must define the full map ID (map@mapset) and add those mapsets to the GRASS search path with *g.mapsets*.

Boundary conditions type are defined by an integer as follow:

- 0 or 1: Closed boundary (default)
- 2: Open boundary: velocity at the boundary is equal to the velocity inside the domain
- 3: Not implemented yet
- 4: User-defined water depth inside the domain

The "open" and "closed" boundary conditions are applied only at the border of the GRASS computational region.

Note: *infiltration* and any of the Green-Ampt parameters are mutually exclusives. Likewise, if any of the Green-Ampt parameter is given, all the others should be given as well.

4.3 [output]

Keyword	Description	Format
prefix	Prefix of output STRDS	string
values	Values to be saved. Each one will be a STRDS	comma separated list

The possible values to be exported are the following:

Keyword	Description	Format
h	Water depth	meters
wse	Water surface elevation (depth + elevation)	meters
v	Overland flow speed (velocity's magnitude)	m/s
vdir	Velocity's direction. CCW from East	degrees
qx	Volumetric flow, x direction. Positive if going East	m^3/s
qy	Volumetric flow, y direction. Positive if going South	m^3/s
boundaries	Flow coming in (positive) or going out (negative) the domain due to boundary conditions. Average since the last record	m/s
infiltration	Infiltration rate. Average since the last record	mm/h
rainfall	Rainfall rate. Average since the last record	mm/h
inflow	Average user flow since the last record	m/s
losses	Average losses since the last record (*new in 17.1, renamed in 17.7*)	m/s
drainage_stats	Average exchange flow between surface and drainage model since the last record (*new in 17.7*)	m/s
verror	Total created volume due to numerical error since the last record (*new in 17.1*)	m^3

New in version 17.1: *drainage_cap* and *verror* are added.

Changed in version 17.7: *drainage_cap* is renamed to *losses*

Additionally to output a map at each *record_step*, *h* and *v* also produce a map of maximum values.

Note: Water depth maps, apart from map of maximum values, do not display values under the *hmin* threshold (See below). When the exported map is totally empty, it is deleted at the end of the simulation.

4.4 [statistics]

Keyword	Description	Format
stats_file	Statistics file	CSV table

4.4.1 Statistics file

Changed in version 17.1: Mass balance calculation now takes into account the volume from losses. Created volume calculation is changed.

The statistic file is presented as a CSV file and updated at each *record_step*. The values exported are shown in the table below.

Water entering the domain is represented by a positive value. Water that leaves the domain is negative. Volumes are in m^3.

Keyword	Description
sim_time	Elapsed simulation time
avg_timestep	Average time-step duration since last record
#timesteps	Number of time-steps since the last record
boundary_vol	Water volume that passed the domain boundaries since last record
rain_vol	Rain volume that entered the domain since last record
inf_vol	Water volume that left the domain due to infiltration since last record
inflow_vol	Water volume that entered or left the domain due to user inflow since last record
losses_vol	Water volume that entered or left the domain due to losses since last record
drain_net_vol	Water volume that entered or left the surface domain since last record due to exchanges with the drainage network
domain_vol	Total water volume in the domain at this time-step
created_vol	Water volume created due to numerical errors since last record record
%error	Percentage of the domain volume variation due to numerical error. Corresponds to *created_vol* / (*domain_vol* - *old_domain_vol*) * 100

Changed in version 17.7: *drain_cap_vol* is renamed to *losses_vol*

New in version 17.7: *drain_net_vol* is added.

4.5 [options]

Keyword	Description	Format	Default value
hmin	Water depth threshold in metres	positive float	0.005
cfl	Coefficient applied to calculate time-step	positive float	0.7
theta	Inertia weighting coefficient	float between 0 and 1	0.9
vrouting	Routing velocity in m/s	positive float	0.1
dtmax	Maximum surface flow time-step in seconds.	positive float	5.0
dtinf	Time-step of infiltration and losses, in s	positive float	60.0

When water depth is under *hmin*, the flow is routed at the fixed velocity defined by *vrouting*.

4.6 [drainage]

New in version 17.7.

This section is needed only if carrying out a simulation that couples drainage and surface flow.

Warning: This functionality is still new and in need of testing. It may be buggy. Use with care.

Keyword	Description	Default value
swmm_inp	Path to the EPA SWMM configuration file (.inp)	
output	Name of the output Space Time Vector Dataset where are written the results of the drainage network simulation	
orifice_coeff	Orifice coefficient for calculating the flow exchange	0.167
free_weir_coeff	Free weir coefficient for calculating the flow exchange	0.54
submerged_weir_coeff	Submerged weir coefficient for flow exchange calculation	0.056

New in version 17.11: *orifice_coeff*, *free_weir_coeff* and *submerged_weir_coeff* are added.

The output maps are organised in two layers. The nodes are stored in layer 1, the links in layer 2.

The values stored for the nodes are described below. All are instantaneous.

Column	Description
cat	DB key
node_id	Name of the node
type	Node type (junction, storage, outlet etc.)
linkage_type	Equation used for the drainage/surface linkage
linkage_flow	Flow moving from the drainage to the surface
inflow	Flow entering the node (m^3/s)
outflow	Flow exiting the node (m^3/s)
latFlow	SWMM lateral flow (m^3/s)
head	Hydraulic head in metre
crownElev	Elevation of the highest crown of the connected conduits
crestElev	Elevation of the top of the node in metres
invertElev	Elevation of the bottom of the node in metres
initDepth	Water depth in the node at the start of the simulation
fullDepth	*crownElev* - *invertElev* (m)
surDepth	Depth above *crownElev* before overflow begins
pondedArea	Area above the node where ponding occurs (m^2)
degree	Number of pipes connected to the node
newVolume	Water volume in the node
fullVolume	Volume in the node when *head* - *invertElev* = *crestElev*

The values stored for the links are as follows:

Column	Description
cat	DB key
link_id	Name of the link
type	Link type (conduit, pump etc.)
flow	Volumetric flow (m^3/s)
depth	Water depth in the conduit (m)
velocity	Average flow velocity (m/s)
volume	Water volume stored in the conduit (m^3)
offset1	Height above inlet node invert elevation (m)
offset2	Height above outlet node invert elevation (m)
yFull	Average water depth when the pipe is full (m)
froude	Average Froude number

4.7 [grass]

New in version 16.9.

Setting those parameters allows to run simulation outside the GRASS shell. This is especially useful for batch processing involving different locations and mapsets. If Itzï is run from within the GRASS shell, this section is not necessary.

Keyword	Description	Format
grass_bin	Path to the grass binary	string
grassdata	Full path to the GIS DataBase	string
location	Name of the location	string
mapset	Name of the mapset	string
region	Name of region setting	string
mask	Name of the raster map to be used as a mask	string

New in version 17.11: *region* and *mask* are added.

With GNU/Linux, *grass_bin* could be simply *grass*.

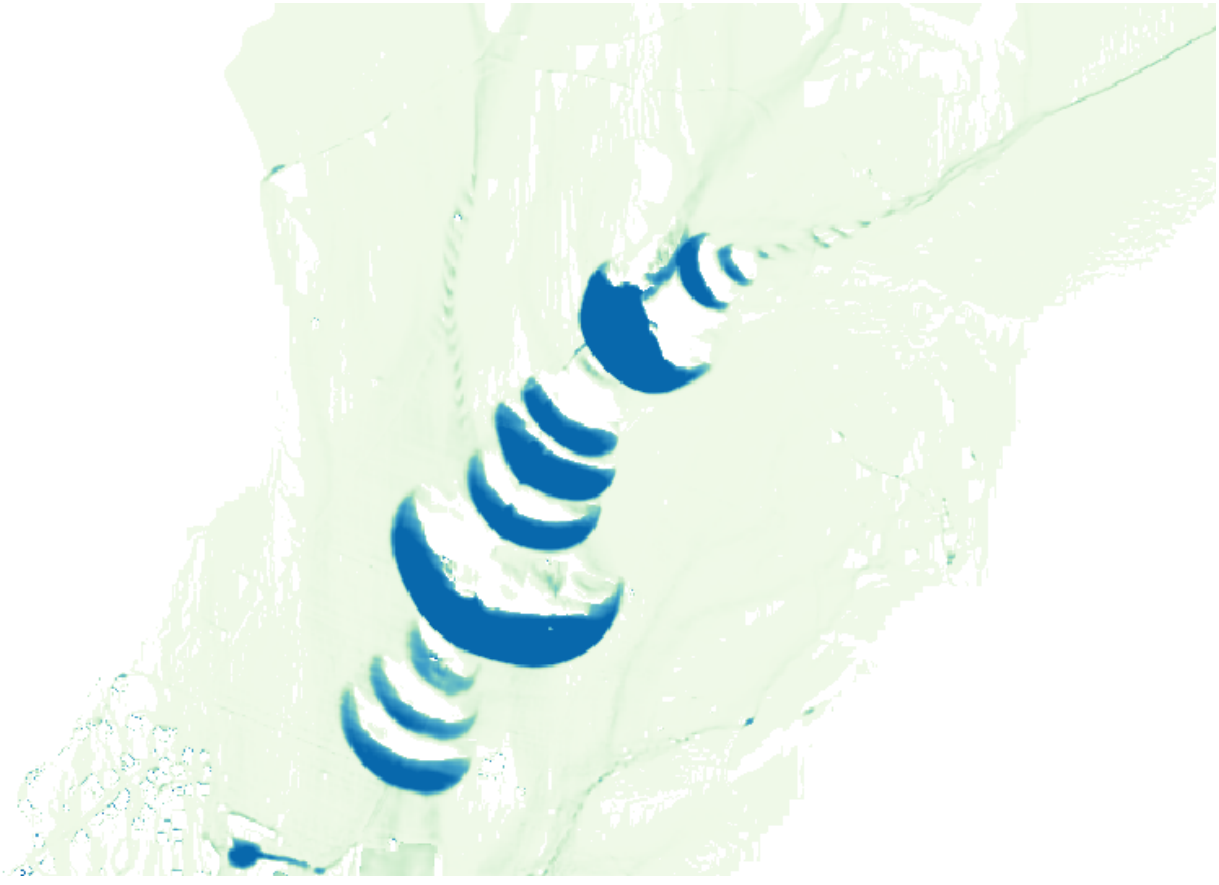
The *region* and *mask* parameters are optionals and are applied only during the simulation. After the simulation, those parameters are returned to the previous *region* and *mask* setting.

CHAPTER 5

Frequently Asked Questions

5.1 Controlling numerical instabilities

In some cases, runaway instabilities could occur, creating wave-like surface flow:



There are two ways to control them. The first one and the more effective is by reducing the time-step, which could be achieved by changing two options:

- *cfl* that applies to every calculated time-step
- *dtmax* that defines a maximum value for the time-step

The second one is by reducing the *theta* option. Please note however that a value below 0.7 could be counter-productive.

5.2 Performances and computer resources usage

Itzï is parallelized using OpenMP. By default, it will try to use all available hardware threads on the machine. The number of threads used can be changed by setting the environment variable OMP_NUM_THREADS.

Given the type of numerical scheme, using a computer with more cores and faster RAM will likely decrease the computation time. No parallel efficiency test has been performed so far, though. For an example of expected performance, a 24h simulation of urban floods with direct rainfall on a 5m DEM of 3.5 millions cells takes around 3 hours with an Intel Core i7-4790 (4 cores, 8 threads).

5.2.1 How to decrease computation time

The factors that influence the computation time are:

- The duration of the simulated event.
- The number of cells in the domain.
- The number of wet cells in the domain. Direct rainfall is more demanding.
- The cell size. A smaller cell size decreases the time-step.
- The maximum water depth in the domain. The higher the water, the smaller the time-step.
- The amount and frequency of result maps. Disk operations being slow and not yet parallelized (as of version 17.1), writing more maps to the disk will slow the simulation down.

As we can see, they are two main categories of factors. Those that increase the raw computation load (more cells), and those that lower the simulation time-step. For the same study area, increasing the cell size is the more efficient way to make a simulation faster, because it influence both the number of cells and the time-step.

5.2.2 Memory usage

On average, *Itzï* 17.1 uses around 250 MB of RAM for each million cells in the domain.

CHAPTER 6

Programer's manual

Itzï is written principaly in Python. The computationally intensive parts of the code and some C bindings are written in Cython. Itzï includes the SWMM source code, which is written in C. As of version 20.5, itzi only supports Python 3.

We do our best to keep Itzï PEP8-compliant. Please use the pycodestyle utility to check your code for compliance. Sometimes it is difficult to keep the line length under 72 characters. The line length could be extended to 90 characters in those cases.

6.1 Source code management

The source code is managed by git and hosted on Bitbucket. The best way to contribute is to fork the main repository, make your modifications and then create a pull request on Bitbucket. The repository have two branches:

- *master* than contain the current released verion.
- *dev* where the main development takes place.

The code should be tested in *dev* before being merged to master for release.

6.2 Development environment

We recommend to create a virtual environment to work on the source code.

```
$ python3 -m venv itzi_dev
```

Then you can activate the virtual env and install the dev version of Itzï.

```
$ source itzi_dev/bin/activate
$ pip install numpy
$ cd itzi
$ pip install -e .
```

Now, every change you make to the Python code will be directly reflected when running *itzi* from the command line. To leave the virtual env:

```
$ deactivate
```

6.3 Cython code

After modifying the Cython code, you should first compile it to C, then compile the C code.

```
$ cython -3 itzi/swmm/swmm_c.pyx itzi/flow.pyx
$ rm -rf build/
$ pip install -e .
```

6.4 Testing

Testing is done through pytest. Running the tests require the following additional requirements:

- pytest
- pytest-cov
- pytest-xdist
- pandas
- requests

pytest-xdist allows to run each test in a separate process. To do so, run the following command:

```
$ pytest --forked -v
```

To estimate the test coverage:

```
$ pytest --cov=itzi --forked -v
```

6.5 Packaging

The process for packaging and sending to pypi is done via a bitbucket pipeline, defined in the bitbucket-pipelines.yml file.