Tutorial 1: Hello flox

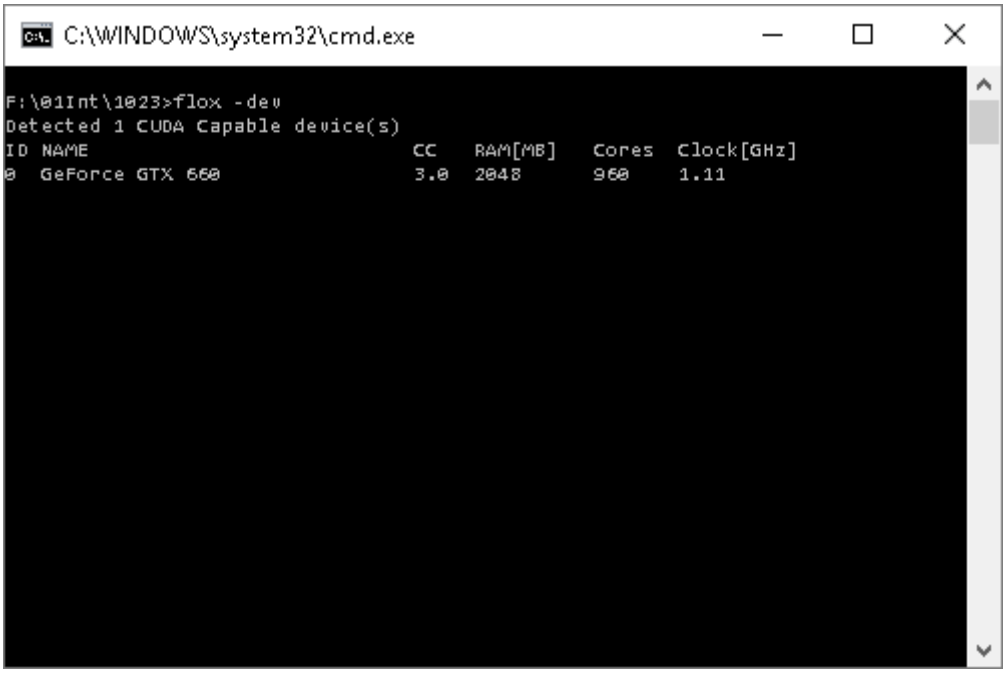
This is your very first tutorial for flox. It shows the basics on how to successfully run a simulation with flox. This tutorial can be run with the demo version and the full version. You can get your demo here.

Step 0: System Check

After the installation of either the demo version or the full version, we first check if everything works as intended and if we have a compatible GPU available. Open a command line prompt (e.g. run 'cmd' on Microsoft Windows systems). Change to the folder where you installed flox and run it with the parameter -dev.

command line	flox -dev

The output should look something like this:



Running this command gives you a list of all CUDA enabled devices on your current machine and some of their specifications. If this program call crashes, your system is probably not 64bit compatible, you have an old version of drivers, or flox is missing a DLL. Missing DLLs happen most commonly if you move the flox.exe away from the installation directory. If you have NVIDIA hardware installed but the "old driver" message pops up, consider installing newer drivers from their site.

If the list of all devices is empty, or the column 'CC' (compute capability) shows a number below 3.0 you won't be able to run flox on your GPU. If you have one or more GPUs showing up with CC 3.0 or higher, you should be ready to go.

Hint: If you have multiple GPUs you can select the one to be used in your simulation, by giving its number (column 'ID' in the list) in the infile. If you don't specify which GPU to use, the system will try to determine the fastest out of the available GPUs.

Step 1: Setting up your simulation

The input to a flox simulation is always a text file ("infile"), which is given to the program as a parameter:

command line	flox *infile.txt*

Using this file you can specify a variety of model parameters to define the characteristics of your model. See the reference for all parameters.

For this very first tutorial we will write this infile all by ourselves, using only the bare minimum of commands necessary to define a running simulation. Open the text editor of your choice (notepad is fine) and create a new text file!

First off, we need to define the surface (grid, or mesh) on which we want to run our simulation. Mesh generation is not covered in this tutorial, as it is a complex topic in itself and can be done with standard GIS operations. For this tutorial we therefore will provide you with a mesh:. You can find it here. Download and unzip it anywhere on your harddrive or on a network drive. Remember the path and filename however. In our example we will save the mesh (the .tif file) as

c:\flox\grids\hello1m.tif

So the first line in our infile should tell flox where the mesh is. The keyword for this is 'mesh':

infile

```
mesh c:\flox\grids\hello1m.tif
```

(Adjust the path to YOUR mesh location.)

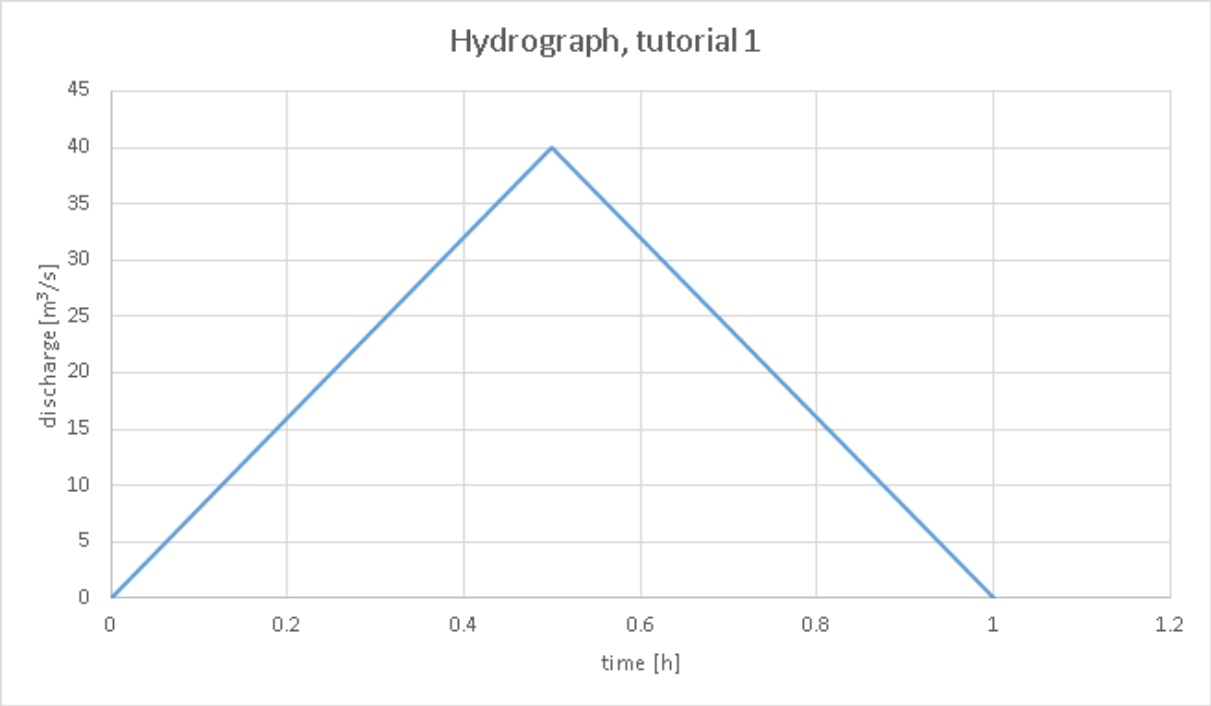
So far so good. Next up we set a water source, as without a source there will be no water to simulate. The program is for hydraulic simulating after all!

We do this with the following lines:

infile

```
source 227 73
0 0
0.5 40
1 0
end_source
```

The keyword 'source' is followed by the coordinates where we want the water to be added in x and y. These coordinates (x = 227/ y = 73) are absolute values in the coordinate system the grid is defined in. The next 3 lines are value pairs of time ***t*** (in hours) and discharge ***Q*** (in m^3/s). They form a hydrograph of 3 points:



The last line ‘end_source’ is a keyword to tell flox, that the hydrograph has ended. The hydrograph can have hundreds of points or just one (which means constant inflow after time ***t*** is reached), but you always need to terminate the list with ‘end_source’

Next on the list: flox does not know when it is done. So we need to tell it how long we want our simulation to run. The keyword for this is ‘runtime’. We want it to run as long as our hydrograph provides water (1h). This is the line to do that:

infile	`runtime 1.0`

Note: the runtime is specified in hours! Alternatively, you can use ‘runtime_s’ to specify the runtime in seconds.

Lastly we have to worry about the output. flox generates grids and images in the resolution of the input grid (which is 1m in this case). In order not to confuse the results of multiple simulations and keep everything orderly we suggest creating a separate folder for this tutorial. We call our folder

c:\flox\tutorial_1

You are free to choose a different name or place (network drives are fine too, but beware of their performance during real-life simulations).
To tell flox where you want it to save the output date of the simulation use the following line:

infile	`path c:\flox\tutorial_1`

You have to replace c:\flox\tutorial_1 with your actual path of course. Make sure the path exists and that you have read and write access.

That’s it. Here is our complete file:

infile	`mesh c:\flox\grids\hello1m.tif`

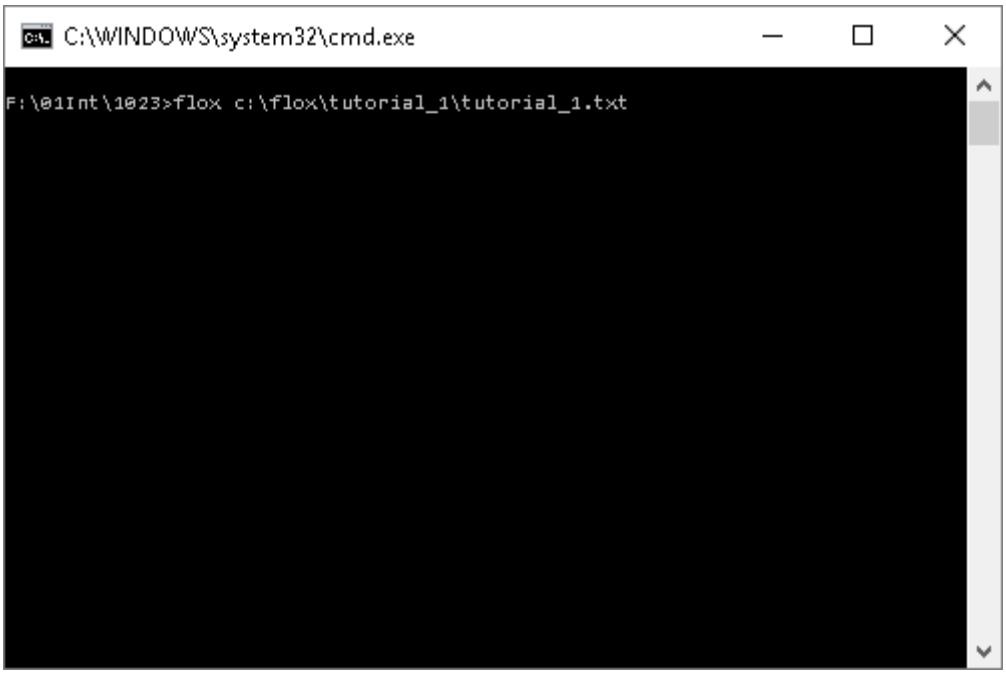
```
source 227 73
0 0
0.5 40
1 0
end_source

runtime 1.0

path c:\flox\tutorial_1
```

Now save your text file. In our case we call it

c:\flox\tutorial_1\tutorial_1.txt

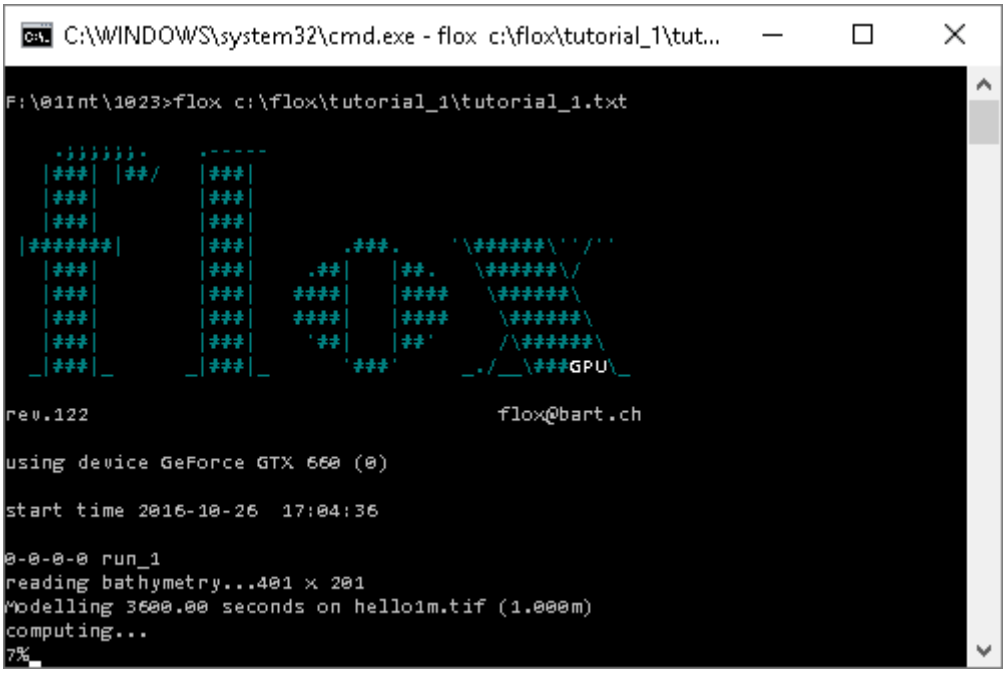
Step 2: Run flox with your infile

If you successfully passed step 0 (System Check) of this tutorial and created your infile, you are good to go. Now, just call flox with your infile as parameter. You can do this either by switching to the directory in which you installed flox, and calling just flox.exe or you can start it anywhere on your filesystem if you call it with its full path and filename (eg. f:\01Int\1023\flox.exe). Of course you can also add the install path to your system PATH variable for more convenience. After the call to the program, you add your infile (path and name) at the end. Beware that if you have blank spaces in your infile path you should enclose it with quotes. In our case the command looks like this:



Step 3: Look at it go!

If you did everything correctly you can see the flox splash, displaying which device is used for the computation, your start time the dimensions and cellsize of the bathymetry (mesh) and the time to simulate.



Once finished, you can see some statistics on how long the simulation took to compute (152 seconds on our GTX 660), how much time each timestep took on average and other stuff.

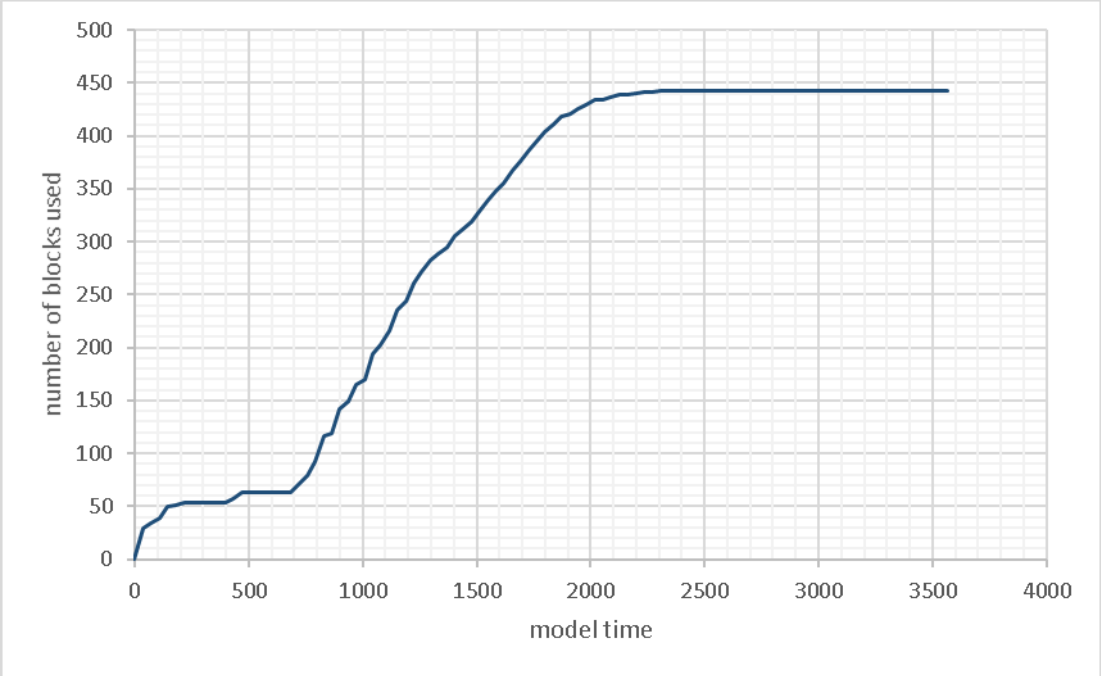
Step 4: Looking at the output

Once your simulation completed you will want to check out what we actually simulated. If you look at the folder you specified in the 'path' parameter, you will find a bunch of files, and 3 subfolders. You can customize the amount of output you desire for your simulation, but without specifying anything in the infile you get the following:

Textfiles	
error_log.txt	In case your model fails, here you can look up the errors
in.txt	A copy of the infile you used. This is mainly for archiving purposes, as infiles get changed around and adapted, this gives you the file actually used to achieve the results in this folder.
progress_log.txt	A log of the steps (in percent of the modelling percentage, we call them 'milestones'). Provides simple metrics, such as timestamp, run time, model time, timestep, number of blocks used etc.
summary.txt	A summary of the simulation, containing more or less the information displayed in the console at the end and some more environment variables.
time_log.txt	A simple log of the model time and the corresponding timestep. Using the default settings, you get about one timestep per second of modelling time.
PNG files (Imagery)	
hillshade.png	As the name suggest, it's a hillshade of your computation mesh. We used this as a background to make a superposition with the progress images further down.
system.png	This is probably the most important thing to look at during the simulation. It gives you an overview of your calculation domain using a hillshade background and displays all the objects you added, such as sources, observers, culverts and so on. Like the hillshade it is generated right after the start of the simulation, so it

	is a great way to check if you did not forget anything and cancel the simulation if you have to. You can see that in our simulation the bathymetry is a flat surface with the word flox engraved as holes. The blue cross in the "o" marks the cell we specified as the location of our source. It's always advisable to check the system.png to see how flox interpreted your infile. If you make mistakes in your infile (eg. typos in coordinates) this is the best place to see what happened.
GeoTIFFs (result grids)	
d_max.tif	This is the maximum water depth reached during the simulation.
v_max.tif	Maximum velocity (scalar) reached during the simulation.
i_max.tif	This is the maximum intensity. Intensity in this context is defined as flow depth times velocity (d * max(v,1.0)) This is a typical value used in risk assessment in Switzerland.

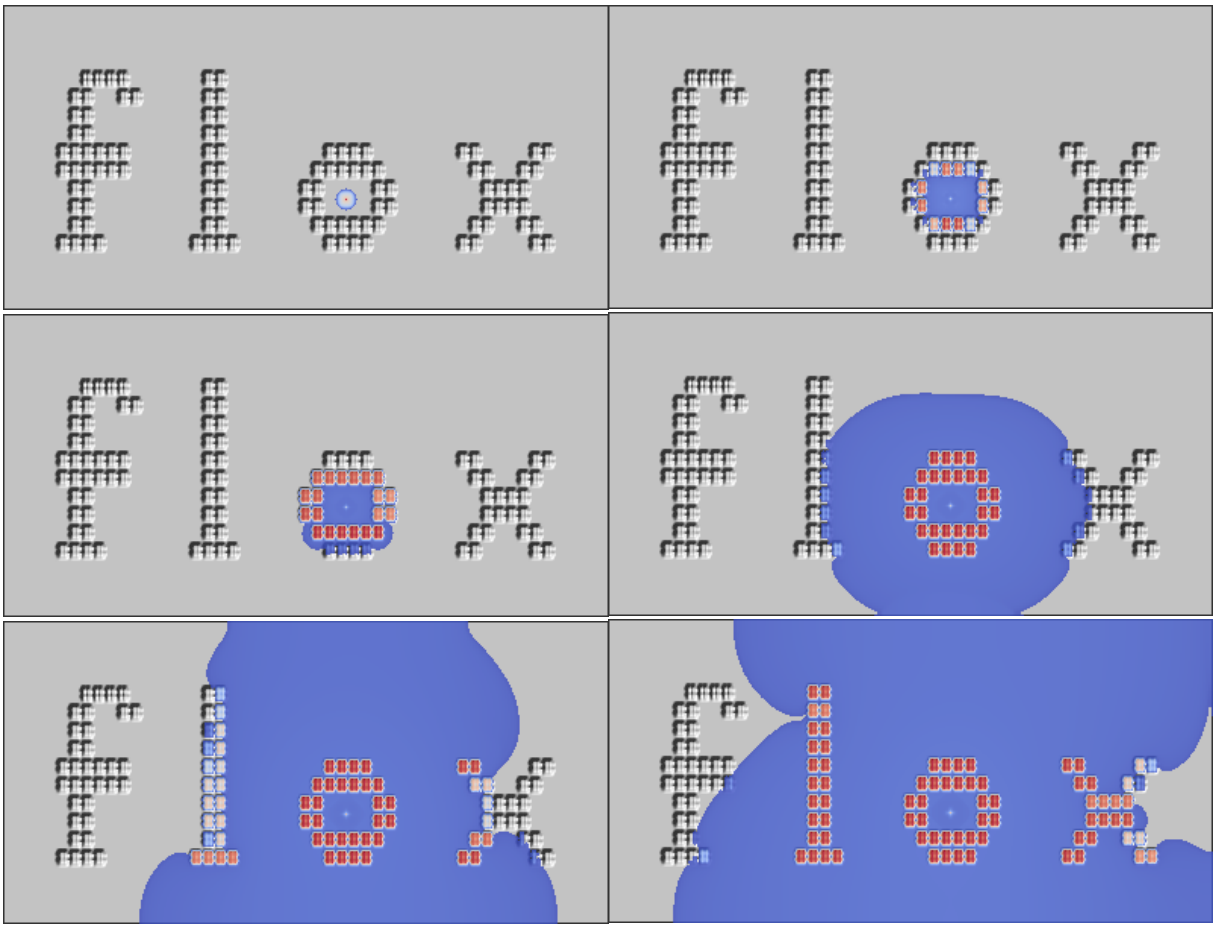
Here you see a simple graph of the number of blocks (a memory unit in flox) used during the 3600 seconds of simulation. At one point the domain is completely wet and no more blocks are needed. This kind of info is easily acquired from the progress log.



Now you wonder: What exactly happened during the simulation? flox does not have a GUI mostly because of performance constraints, it is therefore somewhat of a black box during the simulation. flox does provide you with information though, although not directly to the console. It is writing data for every percent of progress it makes (we call those milestones).

Besides writing to the progress log, flox also exports two images for every milestone. You can find those in the folder called 'progress'. The 2 images are called 'x.png' and 'CFL_x.png' where x is the number of the milestone. They show you the flow depth in full resolution and the CFL information per block (therefore in reduced resolution). CFL information can be critical to your model performance, as the CFL-criterion ultimately limits the timestep of your simulation. Red areas mean that the areas require a small timestep, blue areas mean a larger timestep.

Here you see what the simulation looks like at the 1% / 10% / 20% / 30% /40% / 50% mark. For these images we put the progress images on top of the hillshade:



That's it. You finished your first tutorial with flox. Feel free to play around with the input file a bit, try customizing your simulation with other parameters as you find them in the reference. On the tutorial page you can see a video oft he simulation. How to make those videos will be covered in subsequent tutorials.

bl 27. 10. 2016