SIROCCO : SYMPHONIE
Basics Tutorial guide

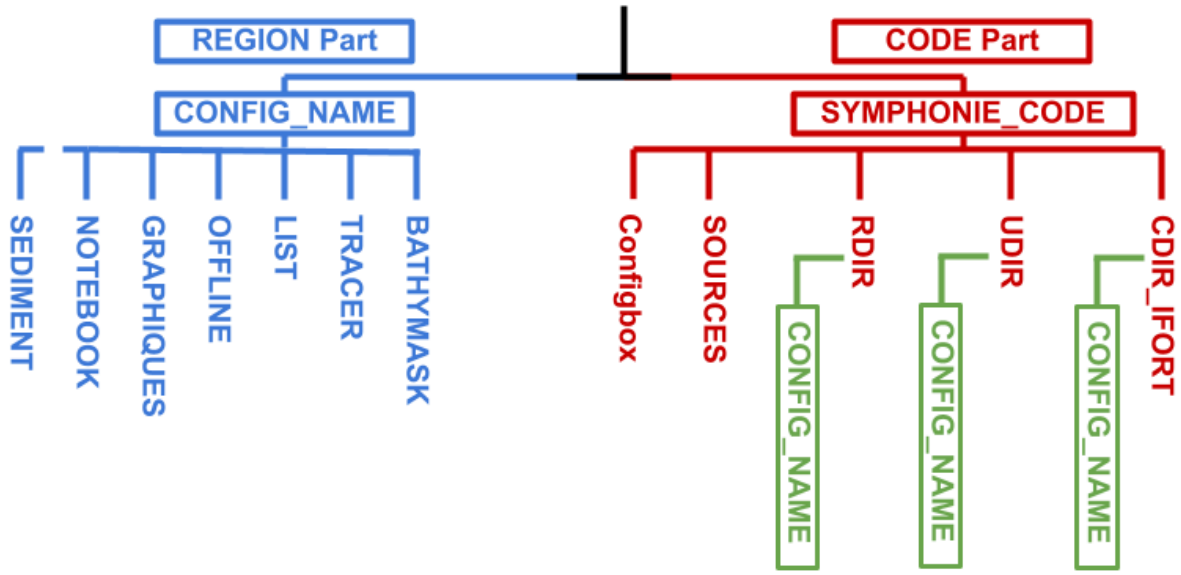
Toulouse, FRANCE, September 2026

A documentation about SYMPHONIE can be found here :
https://doc-symphonie-1de4ad.pages.in2p3.fr/
or
https://sirocco.pages.in2p3.fr/doc_symphonie

Basics Training Course

The SYMPHONIE model directories environment :

Let's start by taking a good view of the "environment" of the model. Inside the global directory (here **SYMPHONIE**), to be general, here is a global description of a typical SYMPHONIE environment tree as seen from the "root" ; ***SYMPHONIE*** in the present case : from the "root" directory (not exhaustively):



In a first approach, you can see that the code is divided into two sections : the **Region part** and the **Code part**. In the **Code part** you will notice in green the directories related to your configurations with here the "**CONFIG_NAME**" exemple (here "CONFIG_NAME=noukoue_lt").

The **Region part** is where you will give the material for your simulation, by material we mean : the geographical zone(bathymetry/grid/vertical coordinates), set the time period, the different sources of forcing (data, tidal constituent, atmospheric, rivers, ocean open-boundaries, etc...), the graphical output of your simulation and some less obvious parameters.

The **Code part** is the "engine" of the simulation, it is the place where you will find :

- The **SOURCES** directory : this is where you will find the source code of SYMPHONIE,

- The **UDIR/CONFIG_NAME** directory : here you will put the specific routines you may want to modify/adjust (or maybe even create) subroutines of the code. It is also where you will compile the fortran code to create the executable program.
- The **RDIR/CONFIG_NAME** directory : This is the directory where you will find that executable program that you compiled from the **UDIR** directory. This is also where you will launch your simulation and define the dashboard (**notebook_list.f**) that will control it.
- The **CDIR_GFORTRAN** directory : This is where the source code from **SOURCES** routines and the modified routines from **UDIR** are being compiled to create binaries needed for the executable program.

For the purpose of this course, we will provide you some already installed configuration environments. If you are interested in using some of our bash script tools to create such a configuration environment, you can take a look at the end of this document.

Compilation :

Before compiling and go further you need to load environment for **SYMPHONIE** :

source ~/.bashrc_symphonie

From now, we work on the **nokoue_lt** simulation. The compilation is done with the **makefile** file in the **UDIR/nokoue-lt** directory.

cd SYMPHONIE/SYMPHONIE_CODE/UDIR/nokoue_lt

In that directory you will find several files, in particular : ***makefile*** and others "***makefile.inc***...." This "makefile" deals with the file **makefile.inc** for user and system dependent information. This means we have to know :

- the compiler
- the name of the config (here **nokoue_lt**)
- the optimisation options for compiler
- and the location of netcdf and pnetcdf libraries and include files.

In the present directory you will find some ***makefile.inc*** examples. For the present case, you will have nothing to change in this ***makefile.inc***, but give a look to this file :

kate makefile.inc

If you want to give a try by yourself, you can try the following commands for searching the right informations :

- *mpif90 -show => to see the compiler*
- *nf-config --fflags => the NETINC for netcdf*
- *nf-config --flibs => the LIB for netcdf (If nf-config does not exist you can try nc-config instead)*

Now how to compile SYMPHONIE, just type the command :

make

You may encounter some difficulties at the end of the compilation, it is often due to linking problems with the libraries.

When the compilation is done, explore your environment (**RDIR/nokoue_lt**) and find the executable file (**symphonie.exe**). The binary (.o and .f90) files are stored into the **CDIR_GFORTRAN/nokoue_lt** directory and the executable program is created under the **RDIR/nokoue_lt** directory.

SYMPHONIE dashboard : notebook_list.f

First we are going to check the notebook that SYMPHONIE will open first : **notebook_list.f** located in the **RDIR/nokoue_lt** directory. Let's have a look at this file (use kate, vim or gedit) :

kate notebook_list.f

First, line 7 reminds you where the "notebook" files are located, it is the directory where we made the changes previously. Then we see that the notebooks are classified by themes (**Time**, **Grid**, **Forcings**, **I/O** etc ...). They are currently standardized names but you can change (***don't do it for the training course***) the names to better personalize these files according to your simulations. Anyhow, you should be careful, because the program is using the **notebook_list.f** file as a dashboard. And so everything inside THIS namelist file has to exist in THE directory selected in THIS namelist (line 7 here)!

*Advice: if you keep many versions of these notebook files, you must be sure not to get the wrong files. Check by editing notebook files from the run directory **RDIR/nokoue_lt** and secondly by having the **notebook_list.f** file open in a second window X. Then proceed in 2 steps: select in notebook_list.f, the directory path of the notebook files with the mouse and complete with the name of the notebook, also selected with the "mouse" in the **notebook_list.f**.*

Setting the time, grid, bathymetry and mask :

Setting notebooks

Now we need to adapt this new space to the specificities of our new application : the Nokoue grid domain. Normally, you should carefully check (and in some cases modify) the parameters of each file in the **NOTEBOOK** directory. As we don't have time to explore all the possibilities of the code, we will only concentrate on the description of some notebook files.

For a first test run configuration of the training, we only focus on few notebook files :

- **notebook_time.f**

- **notebook_grid.f**
- **notebook_bathy.f**

Then we are going to analyze and modify the few things in the notebooks listed before, go under the **nokoue_lt/NOTEBOOK** directory and edit the **notebook_grid.f**

cd SYMPHONIE/nokoue_lt/NOTEBOOK/

kate notebook_grid.f

(or remotely from RDIR/nokoue_lt/ : kate ../../../nokoue_lt/NOTEBOOK/notebook_grid.f)

This file is a namelist file. You will notice several information like the grid size which is defined by the iglb and jglb parameters line 6 and 7 (here set at 206 and 218 respectively) and the number of vertical level kmax, equal to 5 here line 8. We will make a 2D simulation purely forced by the tides, so :

put 1 to the value of kmax.

With Symphonie you have mainly two possibilities to generate a grid mesh : you can create it directly or you can read an “already-set-grid” set into a netcdf (or ascii) file (it must respect a few rules of course). The choice is made by setting the parameter **lonlatfile** line 32 (or so…). Set it equal 'nofile' will lead the code to create the grid using the parameters from line 6 to 20. In our case, the variable is set to a file (*path relative to the RDIR/nokoue_lt directory included*). The code will therefore create the grid by simply reading the different information of the grid stored in this file for **longitude** and **latitude** of the center of the cell. Symphonie will deduce all the other parameters needed.

At line 37 you have the “MPI SECTION” where you can modify the number of the processors you want to use for your simulation. For example, if we have 4 procs and so decide to set the number of procs to 4 by splitting the global domain in 2 subdomains in the i and j directions (line 38 and 39 nbdom_imax=2 nbdom_jmax=2). For our test run we first only use one processor.

So set nbdom_imax and nbdom_jmax at 1.

nbdom_imax=1

nbdom_jmax=1

Then save and close this file.

The next notebook you should check is notbook_bathy.f, edit the file :

kate notebook_bathy.f

This notebook, like the notebook_grid.f, is a namelist file. Here we tell the code what kind of bathymetry it will have to deal with and some other parameters related to the bathymetry. The variable texte250 (here line 9) contains the file (here in ascii format) that has to be taken into account to create the land-sea mask and the bathymetry of the model.

Let's take a quick look at this file. Edit it with kate :

kate ../BATHYMASK/bathycote_in_cotonou_SRTM_River_Channel_PortoNovo.ijh

You will first notice a series of lines "iglb" of "jglb" integers (0 or 1), these represent the land-sea mask of the domain. After the mask, for each grid point (i,j) you will get the depth of the grid cell counted positively downwards.

Then the next block deals with the Bounds and Smoothing options. Those options are turned on or off by setting the variable ioption to 1 or 0. When this option is turned on, the code will modify the read bathymetry depending on the following parameters (here from line 21 to 28). Note that most of the terrain-following coordinates needed a smoothed bathymetry to avoid truncation error when dealing with the pressure gradient term in the momentum equations. Then you have a series of parameters which set the wet-dry system of the code and a mangrove possibility. For the present case we will keep the values and we are dealing with a mangrove swamp.

mangrove_file_name='../../../Nokoue/BATHYMASK/Mangrove_nokoue_lt.dat'. This mangrove could be set in a linear friction term or a quadratic friction. Then you can also set the value of the dissipation coefficient. At the same time as the creation of the grid we will see how to create the files for the bathymetry and mangrove description.

Finally, edit the notebook_time.f :

kate notebook_time.f

In the present situation, we are going to make a test run and stop the code at the end of the initialisation process. But, because later we are going to use forcing data over a 40 day period from 1st October 2016, we set here the start time and the end time at 1st october 2016 and 10th november 2016. So the line 5 and 6 should be set at :

datesim(1:6,1)= 2016 , 10 , 01 , 00 , 00 , 00 ! Start time (yyyy mm dd hh mm ss)

datesim(1:6,2)= 2016 , 10 , 10 , 00 , 00 , 00 ! End time (yyyy mm dd hh mm ss)

To stop a run at the initialisation phase you need to set the variable **run_option** to **-1**. This variable can be found at line 58.

The rest of the notebook_time.f file deals with the parameter that will define the time step of the code. The code can compute its own ideal time step using the cfl parameters that will force the code to evaluate the "worst" scenario in terms of currents and wave propagation (parameter cfl_sshmax and cfl_umax). Then, the CFL criteria is reduced by a factor : cfl_reduce. This is a safe way to define the time step. The model uses a mode splitting technique to evaluate the fast external mode and the internal mode separately. The parameter iteration2d_max_now defines the number of the external time step performed between two internal time steps. But it is also possible to manually define the time step by setting the value of dti_fw. It is not recommended except for test cases.

Note : Do not forget to save your changes when you are closing the file!

Now we are almost ready to run our first test!!

First execution : nokoue_lt

We remind you that the executable program has been created by the compilation and has been put into the **SYMPHONIE_CODE/RDIR/nokoue_lt** directory. So go under that directory :

cd SYMPHONIE/SYMPHONIE_CODE/RDIR/nokoue_lt

To run the program (locally) we are using the **s** script. If you have set the number of procs (in the notebook_grid.f) at 1 just use the command :

./s 1

And if everything has been set properly you should see a lot of information printed on your screen then the date of the simulation. (In case the script does not work properly, check the execution permission and eventually change it by using : **chmod +x s**)

NOTE : The test run should be fast. At the end of the test run the code should wrote on the screen this message :

RUN stopped after initial state as requested in notebooktime

When the run is done, check the files which have been created in the **tmp** directory under **RDIR/nokoue_lt**. You will find several files, and we want to bring your attention to some of them: **messages**, **kount**, **interrupteur** and **grid.nc**

● **grid.nc** is a netcdf file containing the actual grid used by the model during the present simulation.

● ***interrupteur*** *(it means "switch" in french) is a very simple file : it contains three "0". Those zero control the code online. The first one simply stops the simulation, the second one controls an extra output graphic file and the last one is used to create an extra restart file. To use this interrupteur file, it is simple : you just have to change the value of the 0 you want to 1. For example, set the second 0 to 1 and save the file. The code will generate an extra output (extra because it was not planned by the notebook_graph) as soon as it can (basically at the next internal time step).*

● ***messages*** *is also interesting. It gives the major parameters that the simulation is currently using concerning the grid, the maximum depth, the vertical distribution of the vertical levels and the time steps used by the model : the external time step and the internal time step. At the end, you will find the correspondence of geographical position into the actual grid mesh. But we will come back to that part during the realistic scenario.*

● ***kount*** *(only if you have made at least one time step) is an incremented file during the simulation. The file gives you the actual time in the simulation, the number of time steps (internal mode), the number of days already done and the percentage of simulation already done. It is very interesting for a simulation running on a cluster. You can have information on the state of the simulation through that file.*

Visualization of the grid file

The grid.nc file is the grid that the code generates with the information we provide in the notebook files. You can visualize this file with **ncview** but it is better with **ferret** or **python script show_grid.py**.

- **ferret :**

  ferret

Then type the line:

go "../../configbox/FERRET_TOOLBOX/image_visu_bathy.jnl"

With that command line, you are launching a ferret script stored into the "configbox" directory.

You should see that picture :

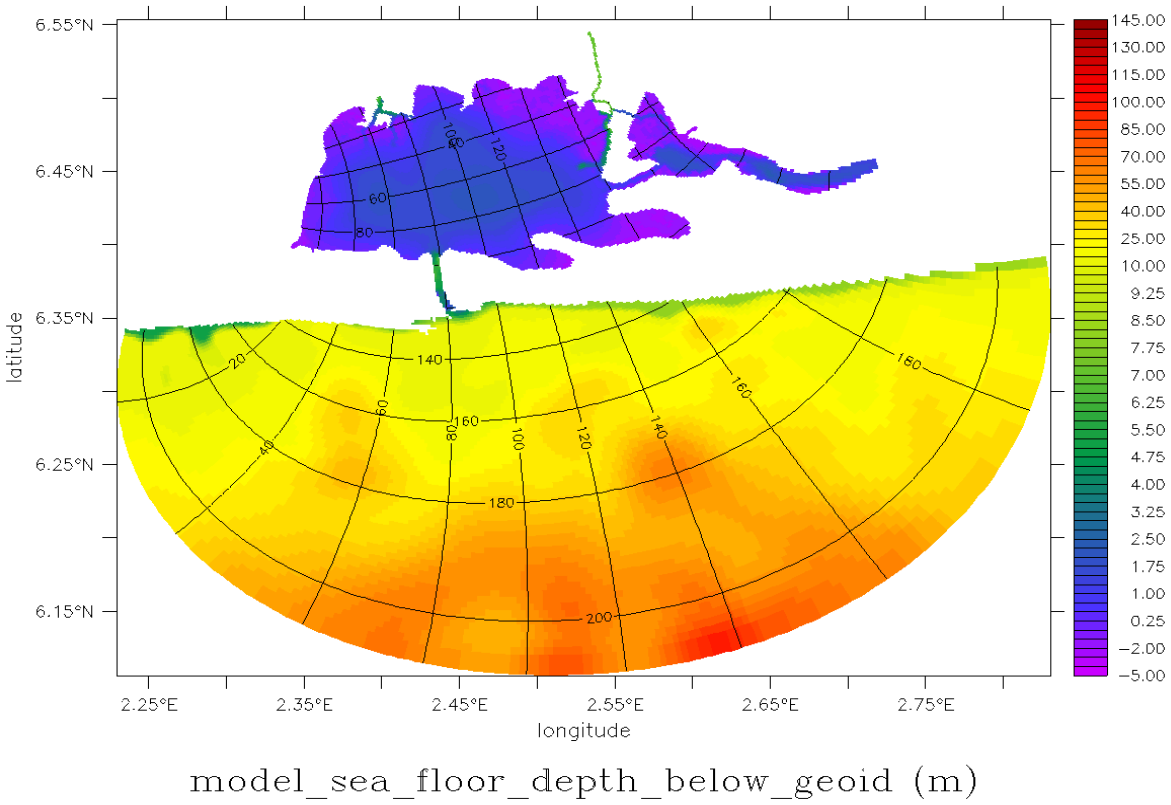


Figure 1 : Bathymetry (in m) and grid of the Nokoué_lt configuration.

- **ncview :**

You can use ***ncview*** for a quick visualization of the other grid parameters :

ncview tmp/grid.nc

- **python script :**

conda activate retrieve_data

python show_grid.py

2D run, Tidal simulation : The Choking effect in Lake Nokoué

At this stage we have access to a SYMPHONIE Environment (**Code** and **Region part**) and we have compiled the code and then learned to launch a quick run directly on the front node of the server . We also set a grid file and a bathymetry file, and finally the notebook_time.f file has been set to stop after the initial phase.

The location we chose for this training is Lake Nokoue located next to Cotonou in Benin. The seasonal dynamics of the water level in the lake is dominated by the tides and the flow of the rivers. In the south the lake is connected to the ocean by a narrow channel (~4km long and 300m wide). This channel has the effect of dampening the tidal signal from the ocean side to the

interior of the lake. This is a choking effect. We will study this effect with a simple 2D simulation.

We are now testing our grid by running a tidal simulation by setting the Tidal forcings with 9 waves M2,N2, S2, K2, K1, O2, P1, Q1 and M4.

From the directory **RDIR/nokoue_lt** :

For the Tidal test run, we are checking only four notebook files : **notebook_grid.f**, **notebook_time.f**, **notebook_tide** and finally **notebook_graph**.

notebook_grid.f

From the **RDIR/nokoue_lt** directory edit the **notebook_grid.f**, :

kate ../../../nokoue_lt/NOTEBOOK/notebook_grid.f

During the creation of the grid we set the number of subdomains to one, now we want to set the number of processors to perform a parallelized computation with the code. What are the parameters we need to adjust? Answer: nbdom_imax and nbdom_jmax!

Set them at : **nbdom_imax=2** and **nbdom_jmax=2**

Because we are making a barotropic tidal simulation, we have set the **kmax** parameter at 1 (1 vertical level). Line 8 : **kmax=1**

Do not forget to save your changes before closing the file.

notebook_time.f

kate ../../../nokoue_lt/NOTEBOOK/notebook_time.f

We already set the length of the simulation. We also ask the code to stop at the end of the initialisation phase by setting the parameter run_option to -1. Now we want to run the simulation so we are going to set this parameter to 0: **run_option=0** at line 58.

notebook_tide

IMPORTANT : unlike the two previous notebook files, this one is not into a namelist format, it is then very important to respect the format of it or the code may have trouble dealing with it.

From the **RDIR/nokoue_lt** directory edit the notebook_tide :

kate ../../../nokoue_lt/NOTEBOOK/notebook_tide

In this file you will notice several distinct parts. Line 2 is obvious ; you will have to set the **0** to **1** if you want to compute the tide in your simulation.

Line 5, is a choice of how the tide is imposed into the simulation. We will keep the parameter equals to 1. *Here you have possibilities of a combination of two potentials : astronomical and Self Loading Attraction and one open boundary condition.*

Line 8, 9 and 10, it is a part of an “online” analysis of the tidal signal you generate in your simulation. Set the first parameter (line 8) to **1**. then let the two others parameters at the default value.

Line 13, it is a choice of how the tide is imposed at the open boundary conditions (OBC). **If it is**

not done, set the value at 1. *(Standard (1) means you are forcing the SSH and the velocities. Clamped (0) means that you only impose the SSH at the boundaries.)*

In the next block, the lines 21, 23, 25 and 27 are giving the path to a different directory. Line 21 gives the path to the Global Tidal Atlas FES2014. From this path the code will find the good file for the different tidal waves we are going to use. Line 23 is useless here. Line 25 gives the directory where the code is going to store the harmonic reanalysis we ask at line 8.

Line 27 gives the path to the nodal parameters.

From line 29, there are several blocks of 10 lines, like :

```
1                          ! 1=the following harmonic is selected, 0 otherwise
M2.FES2014_BENIN.nc        ! ssh input forcing file
M2.FES2014_BENIN.nc        ! u,v input forcing file
none                       ! "detiding" file (if "none" detiding is made with forcing files)
M2.Sreanalysis.nc          ! Ouput reanalysis file
M2.FES2014_BENIN.nc        ! LSA file
none ! M2.sshbiascorrection.nc      ! SSH bias correction file if any (‘none’ otherwise, in directory c)
M2_nodal.txt               ! input nodal parameters file
2                          ! nu nomemclature
0.242334                   ! Amplitude d'equilibre (en mètres) (Apel p215)
```

Those blocks are controlling the major harmonic waves involved for tidal generation.

● The first line sets the use of the harmonic or not by setting the value 0 or 1.

In the test run here, we want to use 9 waves, so in each block of wave information, we set the first parameter to **1** (line 29 for the wave M2, line 40 for the wave N2, etc…).

notebook_graph

Finally, edit the **notebook_graph** (n.b. it is an ascii file format again) :

kate ../../../nokoue_lt/NOTEBOOK/notebook_graph

In this notebook you will be able to control the graphical output of the simulation. First the **DIRGRAPH** at line 3. This directory is the directory where the graphical outputs are going to be stored by the program.

By default this directory is : **../../../nokoue_lt/GRAPHIQUES/**

Line 6, you can set the periodicity (in days) of the output. For the purpose of our training we are going to set this parameter to **0.05** , beware to leave at least a space before the “!” character.

After line 8 you will find several pre-programmed output variables you can store into the outputs. You can take a look at the variable already set at 1 *(1 means that the field will be in the output and 0 that it will not)*. For the present case, we are only setting to 1 the **Sea Surface Height** and the **depth-averaged velocity**, being in a 2D simulation without any forcing terms for temperature

and salinity, any other output would be useless. So set to 1 only these two parameters (line 15 and 43) *(you could set the remaining parameters to 0 but it doesn't matter a lot)*.

To sum up, in the notebook files, you set the ***grid*** and ***parallelisation*** in **notebook_grid.f**, then you set the time and duration of the simulation in the **notebook_time.f**, you set the tidal forcing in the **notebook_tide** and finally the graphical output in **notebook_graph**. You are now ready to launch the first test run on 16 processors by using the command :

./s 4

On 16 processors, the run will take around 10 minutes to finish. Anyway for the purpose of example, you will notice several pieces of information printed on your screen. The code lists the major information like the harmonic waves it is dealing with. As we saw previously there are several files created under the tmp directory where you will find information about the run you performed. For example, the file messages where you will find the time steps and other information related to the grid.

When the run is done you can visualize the output using **ncview**. Use the command :

ncview ../../../nokoue_lt/GRAPHIQUES/2016*.nc

Ncview is not able to deal with non-regular grids as POCVIP or ferret do, anyhow it is a convenient tool to quickly check some output. But if you choose to visualize the ssh_w variable, then by clicking on a point on the ocean side and another on the lake side, you will be able to generate a time series of SSH calculated by the model. You will get something like that :

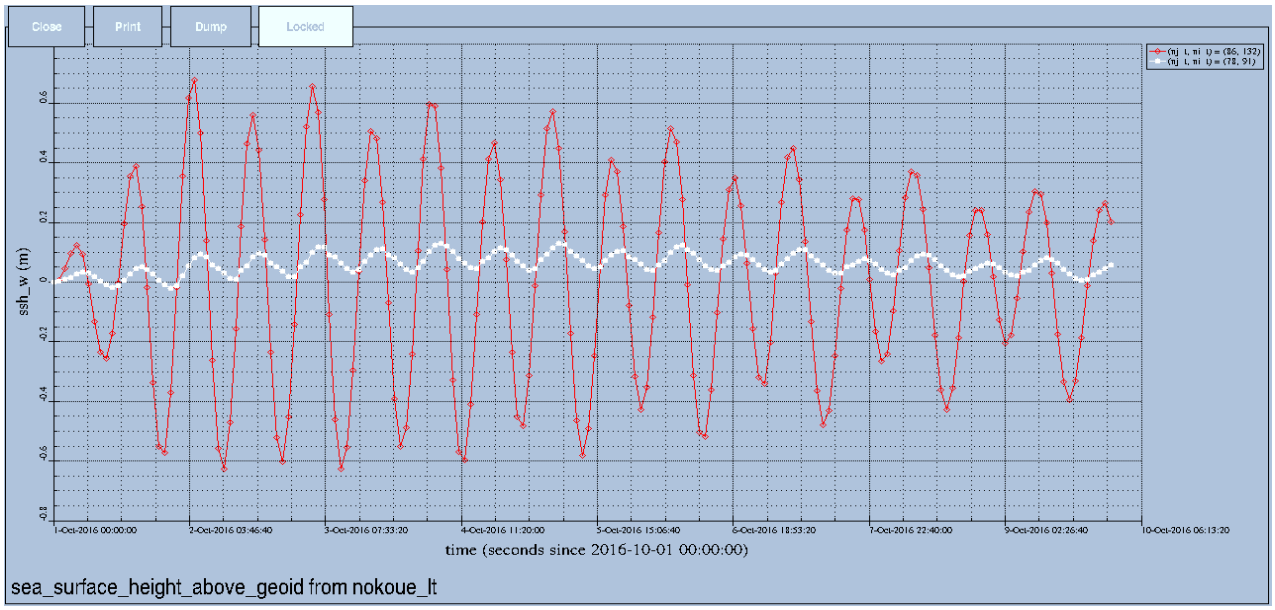


Figure 2 : Evolution of SSH (in m) on the ocean side (red) and in the middle of the lake (white)

The red line shows the time series of ssh on the ocean side and the white one gives the evolution on the lake side. We can notice that the red line clearly shows the cycle of neap and spring tides, but the average level is around 0. But in the lake, the white line, we can notice that the average level is not 0, the lake does not have time to empty completely during the ebb tide, the water it gains during the flood. In addition, the mean water level changes during the spring and neap tide cycle. The mean water level rises during spring tides and falls during neap tides.

*If we are missing time, you can check the graphical outputs under **nokoue_l/GRAPHIQUES/TIDE_2D/** directory, so the **ncview** command will be :*

ncview ../../../nokoue_lt/GRAPHIQUES/TIDE_2D/2016*.nc

Even more, if you want to make the reanalysis of the full set of waves (M2, N2, S2, K2, K1, O1, P1, Q1 and M4) you need to run the code for at least 12 months (simulated time...so more or less 1h30 on 16 processors). It could be even longer if you include more waves. So to gain some time, we propose to check directly the results of an harmonic Reanalysis we performed previously on our cluster.

The file we are going to look at is stored into the directory : ~/SYMPHONIE/FES2014/BENIN/

cd SYMPHONIE/FES2014/BENIN/

You can visualize it by using ncview:

ncview M2.Sreanalysis.nc

3D Run : Rivers and Restarted Simulation : the impact the Lake water level

The impact of the Choking effect on the mean surface level of the lake has been clearly shown under the action of the tides in a 2D simulation. Now we are going to explore a little in 3 dimensions the influence of the river discharge in the process. In addition, we will also see the "restart" procedure available in Symphonie, which allows it to resume a previous execution.

The previous run was a 3D simulation with 5 vertical levels (kmax value in notebook_grid.f) forced with tides and river discharges. We have run a 16 full days simulation from the date 2016/10/01 with a global river input of 500m3/s distributed over 6 river points (4 in the Ouémé in the north-east and 2 in the Sô at the north-west of the grid).

The goal of the simulation is to let you adjust the next phase of the river discharges : you can choose to let the river discharges get back to the low water situation, or you can decide to increase the flood.

First let's see how to ask Symphonie to restart from a previous run. The procedure is quite simple as long as you have the previous run done. You will have "restart_output" files in the "restart_output" directory (restart_output or restart_outbis, we are not gonna take time to explain now).

In a restart procedure, you have to :

- 1: copy the restart files in the **restart_input** directory of your **RDIR/nokoue_lt**

cp restart_output3D/* restart_input/.

- 2: Set the notebook_time.f into a restart configuration, edit the file :

  kate ../../../nokoue_lt/NOTEBOOK/notebook_time.f

  then line 12 set the "**initial**" parameter to **1** : **initial=1**

  You also need to change the end date (datesim(1:6,2) line 6) to run until the 6th of november 2016.

  datesim(1:6,2) = 2016, 11 , 01 , 00 , 00 , 00

When you run the simulation, the parameter "initial=1" will force Symphonie to read the previous state of the run store in the restart_input directory. It is important to know that the MPI subdomain description must be the same in the restarted simulation that it was in the first run.

kate ../../../nokoue_lt/NOTEBOOK/notebook_grid.f

The previous run was made on a 4x4 mpi description, so it corresponds to the previous 2D run too. So you will not have to adjust it in the **notebook_grid.f** file.

For running a 3D configuration with the same number of layers of the simulation we want to restart, we have to set the **kmax** value to 5. Edit the notebook_grid.f and set **kmax=5**.

Now let's take a look at the forcings, the tides and the rivers. The tides will be the same as the 2D run, so any changes are needed. For the rivers, we will have to look at the **notebook_rivers** to set them.

Form the **RDIR/nokoue_lt** directory edit the notebook_rivers :

kate ../../../nokoue_lt/NOTEBOOK/notebook_rivers

The notebook_rivers is not a namelist, it is an **ascii** format file that has to **respect the format**. As you can see in the file, the first line gives the number of rivers the model has to deal with. By default the number is 0. If you set the value to 2 (for the example) the code will create two rivers discharge and will read two blocks "rivers" from the file. For example, the line 3 to 20 are delimiting a block :

```
Oueme
1           ! river mouth location along i axis
183         ! river mouth location along j axis
0 ! length (grid nodes) of an additional water way
1 ! Flux Direction 1=increasing i, 2=increasing j, 3=decreasing i, 4=decreasing j RIVERDIR(:)
10          ! distance (grid indexes) from the mouth where upwind advection is applied
1000.       ! Default Value (if no river file) River discharge m3/s RIVERFLUX(:,1)
1. , 50. , 2000. ! In case of an additional water way, give H, cross DX, along DX of the first grid point
00.00       ! Incoming salinity                                  RIVER_S(:)
29.         ! Annual minimum value of the incoming temperature          RIVER_TMIN(:)
08          ! Month of the annual minimum value of the incoming temperature
31.         ! Annual maximim value of the incoming temperature          RIVER_TMAX(:)
0           ! Boundary condition scheme at river grid input cell        RIVER_OBCTYPE(:)
```

```
1            ! 1 if a data discharge file is available, 0 otherwise       REALRIVER(:)
24.          ! Discharge data file periodicity (hours)                    RIVERINFO(:)
../../../DATA/RIVERS/OUEME.dat
2016  01    01     12     00      00  ! Year Month Day Hour Minute Second of the first data
```

To explain this block we will refer line 3 to the first line of the block. So the first line gives the name of the river. The line 2 and 3 give the index position of the river mouth. This point has to be a land mask grid point next to an oceanic grid point! That's the rule, follow it (dura lex sed lex!). In line 5, you have to give the direction of the flux : if your river grid point is (1,183) and you want the river flows into the next point (2,183) that means that the flux is going in the "increasing i direction"; So the parameter line 5 must be set to 1. Line 6, the parameter is used for the numerical scheme dealing with the tracers (T and S) advection.

We start the data part : for a river, you have two choices ; the choice to put a constant river discharge or the choice for which you have discharge data. For the constant discharge the flow is fixed at line 7 (in m3/s). If you have a dataset you can use line 14 and set it to 1 (0 instead). Then you will have to give a river file (with the relative path) line 16, the periodicity of the dataset (in hours) line 15 and the date of the first input (first line of the file) line 17.

Finally regardless of the type of river discharge (constant or not), the discharge has a varying temperature over time. It is an annual cycle between the two temperature values given by the line 10 and 12. The minimum temperature is fixed by the value (in month) at line 11 and the cycle follows a simple sinusoidal variation.

For the training the discharge point has been set already, we will take some time if it is possible to explain how to proceed during the run.

So, we are going to set the **number of river input** to **6** (first line), and we are going to take a look at the values in the data discharge river file for the Ouémé : Oueme.dat .

kate ../../../DATA/RIVERS/OUEME.dat

This file consists in a simple series of data entries (one per line) with a fix frequency : 24h (the same given in the notebook,) and the first date will correspond to the one set in the notebook too.

Before running let's change a little the notebook_graph :

kate ../../../nokoue_lt/NOTEBOOK/notebook_graph

First, in order to complete the previous run we will set the output directory for the graphical output (**DIRGRAPH** line 3) to the name : **../../../nokoue_lt/GRAPHIQUES/RUN_3D/**

Then set the output variable to 1 for ssh (line 11), wet/dry areas (line 22), depth-averaged velocity (line 43), potential temperature (line 64), salinity (line 65) and 3D velocity (line 86).

Close and save the file.

Create the directory :

mkdir ../../../nokoue_lt/GRAPHIQUES/RUN_3D/

To summarize our 3D simulation, let's make a little checklist: we have

- Set the time and "initial" in the **notebook_time.f**
- Set kmax in the **notebook_grid.f**

- Set a complete tidal forcing : **notebook_tide**
- Set check the **notebook_rivers**
- Set the graphical output with **notebook_graph**

Then Run the code!

And now run the simulation with the submit command **:** **./s 4**

On our cluster with 4 processors the simulation is taking more or less 19 min for 10 simulated days. So approximately 11.5 hours for a year of simulation.

To finish, you can take a look at the results with ncview (for a quick view) :

ncview ../../../nokoue_lt/GRAPHIQUES/RUN_3D/2016*.nc

If we are missing time, some results are provided in :

../../../nokoue_lt/GRAPHIQUES/3D_RUN/

3D Run : Atmospheric and oceanic forcings

Forcing terms : OGCM initialisation

We continue to introduce gradually different forcings. Previously, we made a simple 2D tidal simulation and a 3D simulation with tidal and river discharge. Now the goal is to set a 3D grid and initialise it with the OGCM forcing from a wider simulation. For this new simulation we use the field generated from a global simulation with 1/12° horizontal resolution using the model NEMO. The forcing data consist in 3D fields of Temperature, Salinity, SSH and Currents (U and V) regridded on a regular grid provided from MyOCEAN Copernicus (http://marine.copernicus.eu/) and extracted only on an area of interest. The data are daily averaged values.

In this training course you will find those data under the directory :

~/FORMATION_2026/SYMPHONIE/DATA/OGCM

So go to the RDIR/nokoue_lt directory and edit the notebook_list.f:

cd ~/FORMATION_2026/SYMPHONIE/SYMPHONIE_CODE/RDIR/nokoue_lt
more notebook_list.f

In the “FORCING” block there is the line : **nomfichier(8) ='notebook_obcforcing_nemo.f'**
Edit this file :

kate ../../../nokoue_lt/NOTEBOOK/notebook_obcforcing_nemo.f

Like a lot of notebooks in SYMPHONIE, the first parameter is a switch : **iobc_ogcm=0** .
0 for not using the ogcm and 1 for using them. So set the parameter at 1 : **iobc_ogcm=1**.

The rest of the file does not need to be changed but, you should notice that the notebook gives the path and name of several obcfiles ; they are the files that the code will use to interpolate the initial state and to generate the OBC that will be needed during the simulation. Here you have to provide lists of files containing: the temperature, the salinity, the zonal and meridional currents, U and V respectively, the free surface (ssh) and the grid associated to the fields.

To help you to generate those lists of file, there is a script stored in the SYMPHONIE_CODE/configbox directory: **Do_list_ogcm** . Copy that file in the local RDIR directory (do not forget the "." at the end of the command line) :

cp ../../configbox/Do_list_ogcm .

To use this script is simple ; you have to give the relative path (relative to the RDIR/nokoue_lt directory) to the ogcm data directory (the complete path is also possible). Here the relative path is : **../../../DATA/OGCM** .
Then execute the script with this command :

./Do_list_ogcm ../../../DATA/OGCM

This should generate files beginning by : **list_** .
Then move those files from the current directory (RDIR/nokoue_lt) to the **~/FORMATION_2026/SYMPHONIE/nokoue_lt/LIST/** directory :

mv list_* ~/FORMATION_2026/SYMPHONIE/nokoue_lt/LIST/.
Or in relative path (better in our point of view):
mv list_* ../../../nokoue_lt/LIST/.

Now, it is the time to test the initialisation of your simulation with those data. To gain some time, you can ask the code to stop right after the initialisation step. This option is found in the notebook_time.f file. Edit this file :

gedit ../../../nokoue_lt/NOTEBOOK/notebook_time.f

Line 30, the is a ***run_option*** parameter : the default value is 0. But if you set the parameter to -1, the code will stop right at the end of the initialisation. Set the parameter to -1 : ***run_option=-1***

And the last setting is to effectively run the simulation on a 3D grid. Now we want to make a simulation using 10 vertical levels. So edit the notebook_grid.f file and set **kmax** to 10 (**kmax=10**) :

gedit ../../../nokoue_lt/NOTEBOOK/notebook_grid.f

NOTE: 10 levels is already enough in such a configuration, it is better than the 5 levels used in the previous test
Let's set another GRAPHIQUES directory output : open the notebook_graph .

kate ../../../nokoue_lt/NOTEBOOK/notebook_graph

First, we will set a new output directory for the graphical output (**DIRGRAPH** line 3) to the name : **../../../nokoue_lt/GRAPHIQUES/FULL_3D/**

Then set the output variable to 1 for ssh (line 11), wet/dry areas (line 22), depth-averaged velocity (line 43), potential temperature (line 64), salinity (line 65) and 3D velocity (line 86).

Close and save the file. And create the output directory :

mkdir ../../../nokoue_lt/GRAPHIQUES/FULL_3D/

You are now ready to generate your initial state. Launch the simulation with the ***s*** script:

./s 4

If everything is fine you should get a file : **20161001_000000_symphonie.nc** generated into the **GRAPHIQUES** directory : **../../../nokoue_lt/GRAPHIQUES/FULL_3D/** . You can visualize this file with ferret or ncview or any other software that deals with netcdf file. For example, with ferret, to visualize the temperature in the surface level:

Under ferret:

use "tmp/grid.nc"
use "../../../nokoue_lt/GRAPHIQUES/FULL_3D/20161001_000000_symphonie.nc"
shade/k=10 TEM,longitude_t[d=1],latitude_t[d=1]

Under ncview:
ncview ../../../nokoue_lt/GRAPHIQUES/FULL_3D/20161001_000000_symphonie.nc

To run the model a few days :
In notebook_time.f set the parameter run_option to 0 : ***run_option=0***
Launch the run :

./s 4

and after few minutes check the results :

ncview ../../../nokoue_lt/GRAPHIQUES/FULL_3D/201*_symphonie.nc

Forcing Terms : Atmospheric forcing

The atmospheric forcing is working a lot like the ogcm forcing we dealt with previously. For the training course, we propose to use some meteorological data produced by ECMWF (European Centre for Medium Range Forecasts). Those data are stored under the directory ***~/FORMATION_2026/SYMPHONIE/DATA/ECMWF***. Those data have a 3hr frequency and there is a file for each day. The file provides several air/sea fluxes like: the 10m wind speed (U10m and V10m), the Precipitation/Evaporation (TP), the Solar incident Radiation (SSR), the latent heat flux (SLHF), the sensible heat flux (SSHF), the 2m temperature and dewpoint and several other parameters on a grid resolution of 1/8°. Here we provide a local extraction of this dataset.

To use this data, we have to give the program the lists of files. Under the RDIR/nokoue_lt directory, check the notebook_list.f :

cd ~/FORMATION_2026/SYMPHONIE/SYMPHONIE_CODE/RDIR/nokoue_lt
cat notebook_list.f
then edit the meteo forcing related notebook : **notebook_airseaflux_ecmwf_s26.f**
gedit ../../../nokoue_lt/NOTEBOOK/notebook_airseaflux_ecmwf_s26.f

As for the ogcm, you will notice first a switch parameter: **iairsea=0** . You will have to set that parameter to….. 1 (yes… again!) in order to tell the simulation to take the meteorological forcing into account. So set : **iairsea=1** .
The remaining parameters have to be let to default values.
At the end of the file you will find a serie of *airseafile* which are related to several lists of files. As for the ogcm, you will have to create those files and store them in the directory given at the line 22.
Again, you will use a small script stored into the configbox directory: **Do_liste_ecmwf** . Copy this script locally :

cp ../../configbox/Do_liste_ecmwf .

To use the script it is like the one for the ogcm : you have to give the relative path (relative to the RDIR/nokoue_lt directory) to the ECMWF data directory (the complete path is also possible). Here the relative path is : **../../../DATA/ECMWF** :

Do_liste_ecmwf ../../../DATA/ECMWF

The script should have created the files: *liste_ecmwf_dp2m liste_ecmwf_p0m l iste_ecmwf_ssr liste_ecmwf_u10m liste_ecmwf_ir liste_ecmwf_rain liste_ecmwf_t2m liste_ecmwf_v10m* .
Move those files into the LIST directory of your simulation :

*~/FORMATION_2026/SYMPHONIE***nokoue_lt/LIST**

mv liste_ecmwf* ../../../nokoue_lt/LIST/.

So you have set the meteorological notebook for your simulation. It is time to test it!
There is nothing else to change as long as you did not change anything from the last test run you made with the “ogcm only run”. So go for it and run the code :

./s 4

Let the code runs for a couple of days. As for the ogcm, the code is interpolating the meteorological data on the grid simulation. The interpolating phase is more frequent for the meteorological forcing than the ogcm forcing. It is because the frequency of the data is higher, 1 per day for the ogcm and 8 per day for the atmospheric forcing.

Check some of the outputs after few days of simulation :

ncview ../../../nokoue_lt/GRAPHIQUES/FULL_3D/201*_symphonie.nc

If we are missing time you can check some outputs previously done :

*ncview ../../../nokoue_lt/GRAPHIQUES/3D_FULL/201*_symphonie.nc*

Creating a webcanal _

The purpose of that system is multiple in the case of a grid mesh too big to reproduce accurately the actual river :

- connecting two bassins to gather,
- creating a water pathway for the river discharge to get the influence of tides and on salinity, without reproducing the full rivers path.

Go to the directory : **~/FORMATION_2026/SYMPHONIE/SYMPHONIE_CODE/RDIR/nokoue_lt**

Those webcanal are declare in a file give as an argument in the **notebook_grid.f** file :

kate ../../../nokoue_lt/NOTEBOOK/notebook_grid.f

in your case line 65, uncomment this line :

```
webcanals_list='../../../noukoue_lt/LIST/webcanals_list'
```

Open this file and you will find a number of webcanal and a list of files to describe each webcanal.
here :

2

../../../nokoue_lt/BATHYMASK/webcanal/chanel1.txt
../../../nokoue_lt/BATHYMASK/webcanal/chanel2.txt

Here, 2 webcanal are declared (if 3 webcanal were given the last one will not be read at all)

Webcanal system for rivers :

Let's look at the first one :

kate ../../../nokoue_lt/BATHYMASK/webcanal/chanel1.txt

```
2   2    -99 -99 -99  ! point 1 icanal, jcanal, iocean, jocean, direction
12  2     4   94   1  ! point 2 icanal, jcanal, iocean, jocean, direction
.............................
i   j     h       dx    dy
.............................
2   2    0.20    200.   70.
3   2    0.20    200.   70.
```

```
4   2    0.20    200.   70.
5   2    0.20    200.   70.
6   2    0.20    200.   70.
7   2    0.20    200.   70.
8   2    0.20    200.   70.
9   2    0.20    200.   70.
10  2    0.20    200.   70.
11  2    0.20    200.   70.
12  2    0.20    200.   70.
```

The 2 first lines describe
-> the actual position of the webcanal : from point (4,127) to point(43,127) and each point is related to another point somewhere in the grid, except if the values are **flag with "-99"** value like in the first line. This point will be some kind of dead end for the canal. At this point, it is possible to feed some water discharge at the (4,127) location.
So the second line gives (8,135), the point linked to (43,127) and 1 give the direction of the flow to connect the two points. the direction value are the same as described for the river discharge point :
1, 2, 3 or 4 (1-> i index increasing, 2-> j index increasing, 3-> i index decreasing, 4-> j index decreasing)

Then the following line will describe the "physical" aspect of the webcanal point by point from (3,127) to (43,127).
As you can see we provide information like that : i j h dx dy (or i j h dx dy lon lat)

- i, j are obvious
- h is the local depth of the webcanal
- dx, dy the size of the "grid cell" like an unstructured approach.

If line 4 only contains i, j, h, dx, dy, only columns 1 to 5 will be read, and the longitude and latitude values will retain their initial values.

In case of i j h dx dy lon lat)

- This format implies that line 4 contains the characters ‘lon’ or ‘lat’, along with longitude and latitude values provided in columns 6 and 7.
- lon and lat give the position of each grid.

Depending on the values in the ‘dx’ field, the longitude and latitude assignments will have a broader or narrower scope.
In all cases, they are used to interpolate external fields (OGCM, METEO, etc.) onto the model grid, or in post-processing to generate plots—provided the plotting software can handle a lon,lat field that is coherent along the channel direction but discontinuous in the perpendicular direction (FERRET cannot handle this).

If the ‘dx’ field values are abnormal (e.g., negative, as in the following example), the longitude and latitude values are also used to calculate the resolution along the channel axis (‘dx’ field), while the width (‘dy’ field) is in any case determined by the values in the channel file.
If lon and lat are provided, the value of dx will be deduced using the distance between 2 consecutive points.

N.B. Channels **must be encoded along the Oi axis in ascending order**, but the flow direction at the connection point (from the channel to the sea) can follow **either the Oi or Oj axis**, and in **both ascending or descending directions**.
In the following example, the flow at **point 2** enters the sea in the **ascending Oi direction**.

To summarize, the river discharges are set upstream at the "entry" point of a webcanal and then flow through those canal until it reach the "outpout" of the wecanal which is connected to a grid point in the model, that will correspond to the "real" laguna.

Webcanal system for connecting bassins together :

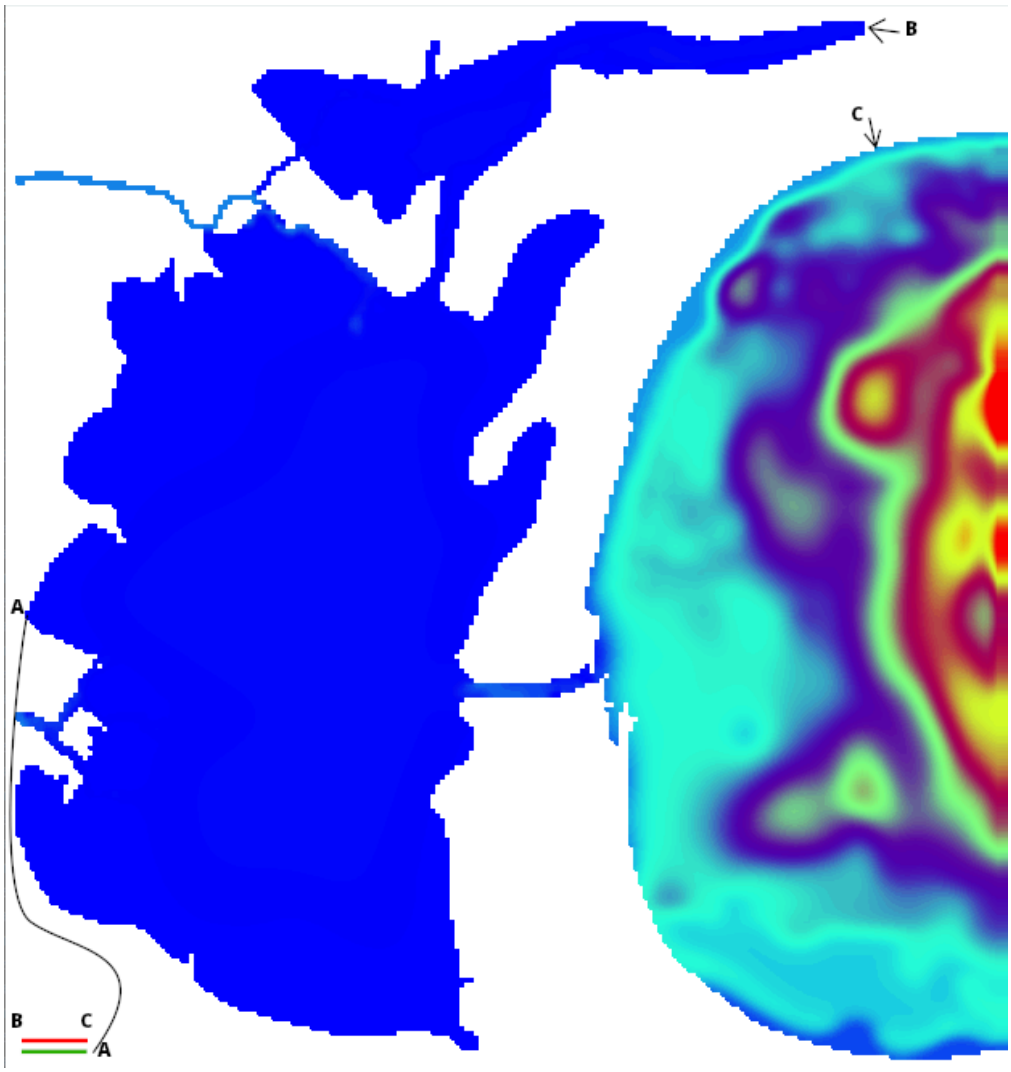
Let’s look at the second file :

kate ../../../nokoue_lt/BATHYMASK/webcanal/chanel2.txt

```
2   4    177 215   3  ! point 1 icanal, jcanal, iocean, jocean, direction
12  4    184 191   4  ! point 2 icanal, jcanal, iocean, jocean, direction
.............................
i   j     h       dx    dy
.............................
2   4    1.42    200.   100.
3   4    2.00    200.   100.
4   4    2.50    200.   100.
5   4    3.00    200.   100.
6   4    3.50    200.   100.
7   4    4.00    200.   100.
8   4    4.50    200.   100.
9   4    5.00    200.   100.
10  4    6.00    200.   100.
11  4    7.00    200.   100.
12  4    8.00    200.   100.
```

It works the same as the previous webcanal file but here both extremities of the channel are connected to “ocean” point : the first line, the point (2,4) is connected to point B and the second

point (12,4) is connected to point C. The flows direction are define by the direction the flow must flow out from the webcanal into the “ocean” points.



To test this configuration you need to :

- Open the **notebook_grid.f** :
  - uncomment the line 65 to set a series of webcanal
  - set the number of level to **kmax=5**

- Open the **notebook_time.f** and set the start date and end date to :
  datesim(1:6,1)= 2016 , 10 , 01 , 00 , 00 , 00 ! Start time (yyyy mm dd hh mm ss)
  datesim(1:6,2)= 2016 , 10 , 10 , 00 , 00 , 00 ! End time (yyyy mm dd hh mm ss)

- Open the **notebook_rivers** and set the number of river to 7 (first line)
- Set a new output directory for the GRAPHIQUES directory in the **notebook_graph** , for example :

../../../nokoue_lt/GRAPHIQUES/Webcanal/
don't forget to create this directory.

- Be sure the **ioffline** parameter is set to 0 in the **notebook_offline.f**
- Set the tidal forcing set the first parameter to 1 (line 2) in the **notebook_tide**
- The other forcing can be keep or set to "0"... it doesn't really matter.

Run the test : **./s 4**

Creating a Full Environment : "CODE Part" and "REGION Part"

To create those two parts of a new configuration, you can use two tools (one for each) :

- mkconfdir : for the CODE Part
- mkregiondir : for the REGION Part

Those two tools are to be used under the "***SYMPHONIE_CODE***" directory. If they are not directly located in this directory, you will find a copy of them under **SYMPHONIE_CODE/configbox/** directory (n.b. for most of the files used by SYMPHONIE, you will find a generic version of them in this **configbox**).

The tools are simple bash scripts that will take a **single argument** to run properly. This argument will be the name of your configuration. So as exemple, we will keep the name "CONFIG_NAME" as described in the environment tree given above at the beginning of this document.
So the command would be :

- ./mkconfdir CONFIG_NAME
  for "CODE part"
- ./mkregiondir CONFIG_NAME
  for "REGION Part"

What is the result of the 1st command ?
Warning: if the result of this operation is : *bash: ./mkconfdir: Permission denied*
this means that the command mkconfdir does not yet have the permission for execution. To give it that permission, do: ***chmod +x mkconfdir***
then try again : ***./mkconfdir CONFIG_NAME***

To see the result of this operation go to **CDIR_GFORTRAN**, **RDIR** and **UDIR** directories and examine their contents (with the command **ls** eventually). Several distinct "simulation" environments can be present in the general environment, enabling a rational multi-use of the same code.
Under the **UDIR** and the **RDIR**, the script will copy several generic environment files from the **configbox** and adapt them to the name you provide as argument.

The result of the 2nd command will be found in the "REGION part" (from the root directory of SYMPHONIE Environment, here ~/FORMATION_2026/SYMPHONIE). The script **mkregiondir** will create the full architecture shown in the "REGION Part" presented previously, and set particularly the notebook files in the NOTEBOOK directory.

N.B. It is not mandatory, but **strongly suggested**, to keep the same name for the argument of the two commands. The system is set that way, and it is far easier for anyone and beginners in particular.

Finally, a safe way to remove a "simulation" sub-level environment in order to avoid the proliferation of unused applications (*Clean your workspace once in a while, wishful thinking…*). Remove quickly and safely the old directories (without collateral damaging) with the **removeconfdir** command, for example :

./removeconfdir CONFIG_NAME