SIROCCO : SYMPHONIE
Drifter 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

OFFLINE Lagrangian Particles

Drifters simulations : Lagrangian transport

With SYMPHONIE, you can perform Lagrangian particle transport in OFFLINE or ONLINE mode. The ONLINE mode means that you run the full dynamics (i.e. you evaluate the prognostic variables ssh, u, v, t and s at the internal time step). On the contrary, the OFFLINE mode consists in reading the averaged dynamic states at a given frequency (which can vary in time) and reproducing the prognostic variables at any time during the simulation. These OFFLINE simulations are much less time consuming than ONLINE simulations for dynamics. It is therefore very convenient for biogeochemical simulations or to perform a large number of pollutant dispersion scenarios over the same simulated period. It is also very convenient for the transport of particles such as Lagrangian particles.

Those particles are referred to as **drifters** in SYMPHONIE. Those drifters can be completely passive and so be transported neutrally-buoyant with the surrounding water (it is the default setting), you also can give them a settling velocity. Their trajectory can also be influenced by the local turbulence using randomly parametrized velocities. *It is also possible to give these drifters a behavior but you will have to code it yourself in the source code. For example, you can decide that the drifter is a floating object in the surface layer (and not the sea surface itself) and let it move only under the effect of horizontal currents. Or you can decide on a more complex situation like a larva that will migrate in the water column with the diurnal cycle between two depths...* A drifter is located geographically and its 3D position translated into decimal grid indexes. The evolution of the indexes of the drifter is then evaluated by using interpolated currents (*adjusted by some behavior if requested or self -coded*) at the drifter position.

For this simulation we are going to use the configuration environment : **nokoue_lt_Drifter** . The first thing to do is loading environment for SYMPHONIE : **source ~/.bashrc_symphonie**

The code has been already compiled for gaining time during the Training.

Offline mode settings

To set the **Offline Mode**, if you start from scratch with offline files provided, all the forcing terms are already switched off. So for a simple offline simulation without river runoff (useful for tracer sources for exemple) and atmospheric forcing (use in the case of geo-biochemical simulation), you don't need any forcing terms. And then the simulation needs only to set the notebooks dealing with : the **time**, the **grid**, and the **offline** files.

From the RDIR :

/pyramides/$USER/FORMATION_2026/SYMPHONIE_CODE/RDIR/nokoue_lt_Drifter

- **notebook_time.f** :

  Open the notebook_time.f :

  kate ../../../nokoue_lt_Drifter/NOTEBOOK/notebook_time.f

The **notebook_time.f** file is a notebook located in the **REGION part** : **nokoue_lt_Drifter/NOTEBOOK/** directory, it is given in a namelist fortran format. In the Offline Mode, you will have to deal with the Start and End time for the simulation. Those information are stored into an array variable : datesim (1:6,1:2). For the training course, we provide offline files enough to cover the period between 2016/10/03 to 2017/12/31, so let's set for first test the start and end date to :

datesim(1:6,1)= 2017 , 03 , 12 , 06 , 00 , 00 ! Start time (yyyy mm dd hh mm ss)

datesim(1:6,2)= 2017 , 03 , 17 , 06 , 00 , 00 ! End time (yyyy mm dd hh mm ss)

*Note : The initial time in **Offline Mode** will respect one rule : it has to be after the time of the second output files you have. For example if you have the first two files for the time : 2009-12-30 at 00:30:43 and 2009-12-30 at 01:30:40 then the start date must be after the second date. So 2009-12-30 at 01:30:41 is accepted by Symphonie. Of course for the end time, you can't go any farther than the time of the last output file you have.*

In the case of **Offline Mode and Lagrangian simulation**, because the dynamic is read and reconstructed from averaged output files, it is possible to increase the internal time step **dti_fw**. The stability of the code is not sensible anymore to the CFL criteria, however, the value of the internal time step will have an impact on the trajectories of the lagrangian particles. So the value of dti_fw can be set at a larger value than the ONLINE computation and also in correspondence with the frequency of your Offline files. For example, with a 1h averaged output offline files, dti_fw has to be set between the online value (depending on the original simulation) and 3600.s, over this value you will lose the benefits of the output frequency. We propose you to set : **dti_fw=180.** (around line 63). And we are set for the notebook_time.f.

- **notebook_offline.f** :

To set an OFFLINE mode, we need OFFLINE files! To create those files, you should have previously set an online simulation with the forcing terms you are interested to set for the circulation and adjust the notebook_offline.f in order to create those offline files. Let's take a look at this notebook with an editor kate or vi or… :

kate ../../../nokoue_lt_Drifter/NOTEBOOK/notebook_offline.f

This notebook is dealing with the offline files through the parameter **ioffline** in two types of use : **ioffline=1** means : creating the output offline files or **ioffline=2** means : reading a list of offline files.

Here in **Offline Mode**, you will set the parameter : **ioffline** to **2**. In this mode, Symphonie will look for a list of file information stored into the variables : **directory_offline** and the file

containing the list of offline files : **offlinefile(1)**. By default those variables are set to :

directory_offline = '../../../nokoue_lt_Drifter/OFFLINE/'

offlinefile(1)='../../../nokoue_lt_Drifter/OFFLINE/liste_offline_March.txt'

For the training those parameters are set correctly.

- **notebook_grid.f** :

We already deal with several pieces of information like the grid size which is defined by the **iglb** and **jglb** parameters line 6 and 7 and the number of vertical level **kmax** line 8 (set to 10 now). In the **Offline Mode**, it is mandatory to set those 3 variables at the size of the original simulation (the one you create the offline output files).

In **Offline Mode**, the variable **lonlatfile** (line 33) is needed to be set to a file that is the **output grid.nc** of the original run. SYMPHONIE would have made a file (by default **grid.nc**) and stored it in the **nokoue_lt_Drifter/OFFLINE/** directory. The grid is simply read from that file. So the lonlatfile must be equal to :

lonlatfile='../../../nokoue_lt_Drifter/OFFLINE/grid.nc'

At line 39 you have the "**mpi section**" where you can modify the number of the processors you want to use for your simulation. For example, we will use a 4 x 4 mpi description : **nbdom_imax=2**, **nbdom_jmax=2**

Drifters settings

You will see the standard procedure today : the transport of neutrally-buoyant drifters and also a case with settling velocity added to the particles. The drifter simulation is piloted by a notebook : **notebook_drifter**.

Edit this file :

kate ../../../nokoue_lt_Drifter/NOTEBOOK/notebook_drifter

Note, that this notebook is also not in a namelist format but is in an ascii format.

Line 1 is again a switch : set it to **1** to set some drifters.

Line 2 describes the way you will set your particles: with the present notebook or coded in the fortran routine. Here we are using the standard procedure so we will use : "**notebook**".

Line 3 sets the output sampling frequency in seconds for each drifter. set the value to **1800.** *Each drifter will have a file giving the location (in 3D) of the drifter in function of time.*

Line 4 gives the type of output file : 0 for 1 file per drifters, 1 for 1 file by mpi subdomain and 2 one file at every time step in netcdf format. Both formats have advantages and flaws. **Keep it set to 0.**

Line 5 you can set the order of the Runge-Kutta time stepping for the Lagrangian transport time scheme. **Let's keep it to 2.**

Line 6 you can set a ratio between the dynamic time step and the lagrangian time step. **Here set to 1 by default**.

Line 7 two numbers : the first describes if you want to set a vertical velocity or not (1 or 0) and the second sets the index for storing the vélocity (**keep the value at 6**).

Line 8 offers the possibility to add a vertical velocity to represent some chaotic behavior du to turbulent kinetic energy or random behaviour or not. Values 0, 1 or 2, **keep it to 0** .

Line 9 deals with the beaching/unbeaching process (set the value to 7 is used, but it will be not discussed here in the Training). **Keep the flag value -999**.

Then, starting at the line 11, you will have to set the number of drifters you want, by giving for each one 3 lines like in the example :

- The first line consists of 4 numbers and gives the start position of the drifter : The 4th number defines what kind of information : geographical position : lat lon depth if the last number is 0 and in index : i, j and k if the number is 1.
- The second is the vertical velocity of the particle due to its relative density. The value will be considered as a settling velocity (so downward)
- The third line is setting the start date of this drifter.

To set 10 drifters, you will have to give this information for each drifter, so you will have to write 30 lines.

For the test you can set the same drifter location : 6.430 2.435 -0.5 but with a delay of 2hr between each :

```
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 02 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 04 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 06 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 08 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
```

2017 03 12 10 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 12 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 14 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 16 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 18 00 00 ! year,month,day,hour,minute,second
6.430 2.435 -0.5 0 ! arg4=1: i j k, arg4=0: lat lon z of the drifter
0.
2017 03 12 20 00 00 ! year,month,day,hour,minute,second

If you want, you can choose a location in the channel also like : **6.393285 2.435682** .

What happens if the time I set for the start time for a drifter is before the beginning of the simulation setted in the notebook_time.f ? The code will simply let the drifter start with the beginning of the simulation.

In this first example, we will analyze the difference in the trajectory of each drifter with their starting date. Here, we are again running in **OFFLINE** mode because of the gain in computing time, but drifters are also working in online (direct) mode.

To set the offline mode, we recall the major point you should care :

- no tidal and ogcm forcing :

  *line 2 : **0** for **notebook_tide** and **iobc_ogcm=0** for **notebook_obcforcing_nemo.f***
- Check that you are effectively using the correct **lonlatfile** in the notebook_grid.f .
- Check that you set the good **start/end date** you want and the good **dti_fw** in the notebook_time.f.
- Then check that you are effectively using the read option **ioffline = 2** in the notebook_offline.f and pointing toward the correct "**liste_offline.txt**".
- Then set a run the period of time from 12th march 2017 to 20th march 2017 in the **notebokk_time.f**.

Now run the code : **./s 4**

The start is long due to the long list of offline files, to be faster it is a good habit to constraint the list of files to the necessary time period.

Visualization of trajectories

An example with gnuplot and individual drifter output files

How to see the output : the drifters positions are stored separately under the tmp directory in a format name : drifter00000XX . XX is the number of the drifter in the order you entered them in the notebook_drifter. Edit one of them :

kate tmp/drifter0000001

The information are given with the following format :

time i_index j_index k_index lon lat depth

You can use **gnuplot** to display one trajectory of your drifter with **gnuplot :**

plot "lake_outline.txt" u 1:2 w l

replot “tmp/drifter0000001” u 5:6 w l

or all trajectories

replot for [i=1:10] sprintf("tmp/drifter%07.0f", i) u 5:6 w l not

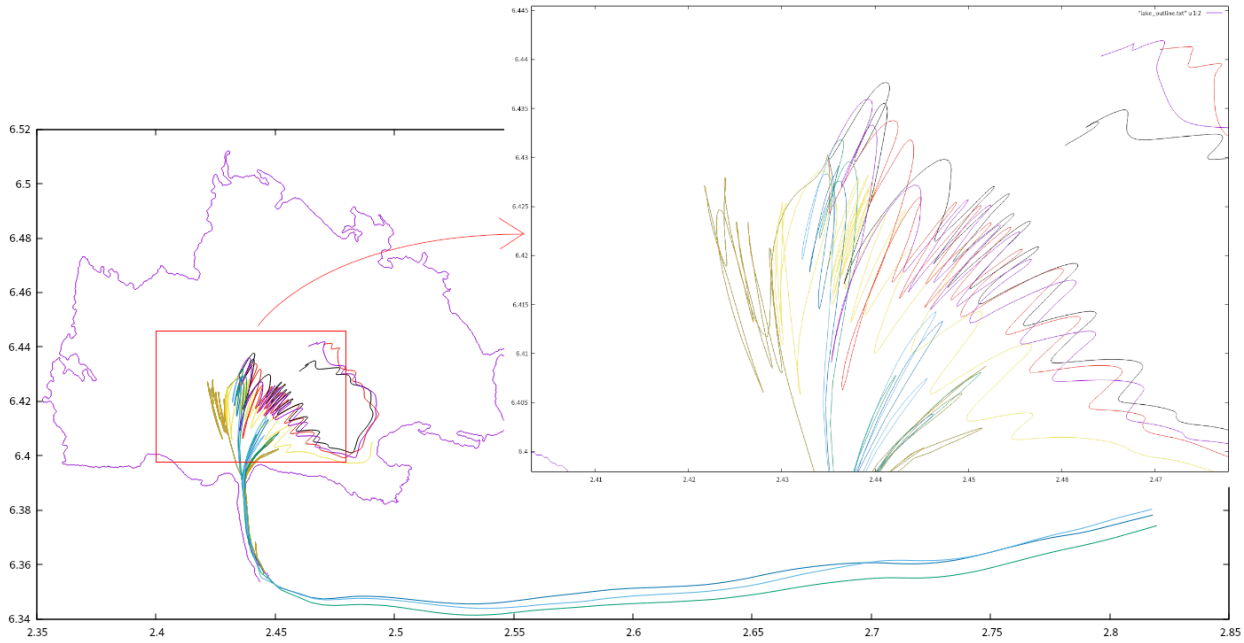


Figure 5 : Trajectories of 10 particles released at the same location but one every 2h.

Or in 3D :

splot for [i=1:10] sprintf("tmp/drifter%07.0f", i) u 5:6:7 w l not

Deposit scénario

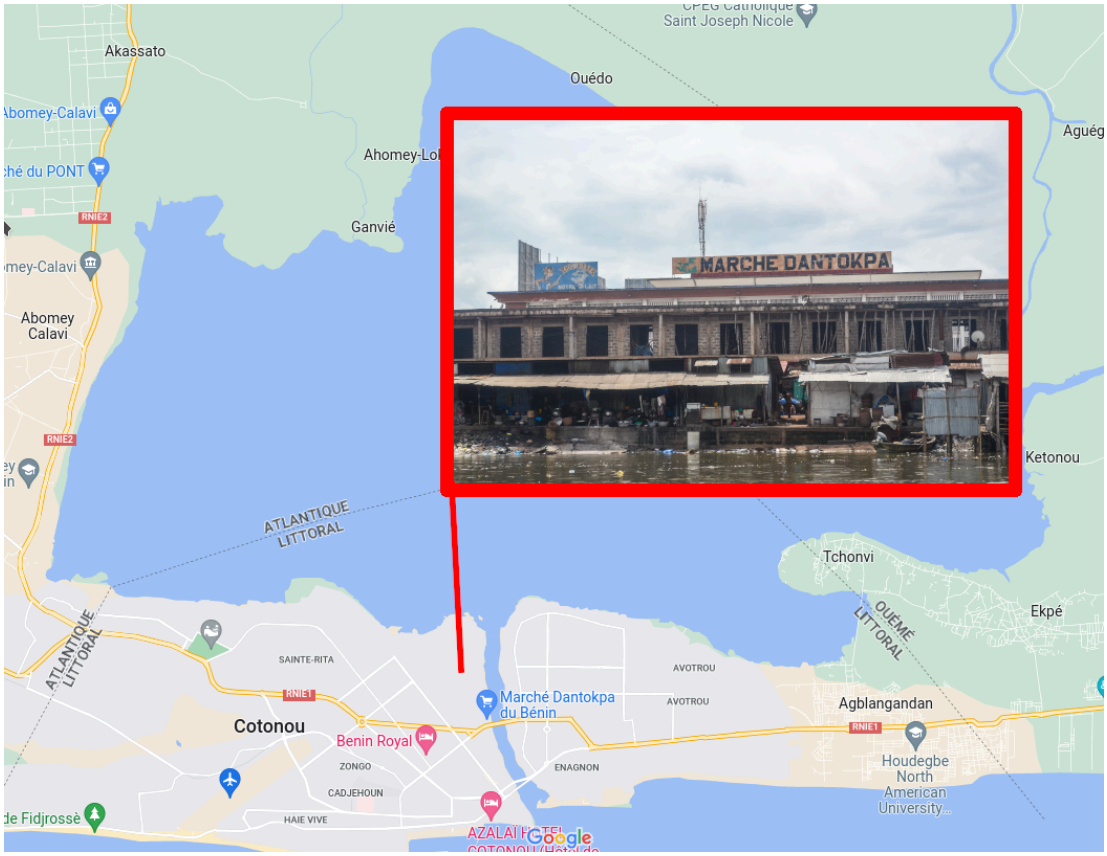


Figure 6 : Location of Dantokpa market on the banks of the channel in Cotonou.

For the next example, we want to make an estimate of the deposit zones of trash generated by the largest open-air market in West Africa : DANTOKPA. About this market here is few information from the wikipedia page :

- 15,342 stores
- One million daily shoppers in 2012
- The banks of the lagoon in Dantokpa are a vast field for spreading the by-products (waste, packaging, etc.) of commercial activity.

The photo taken from the banks of the canal clearly shows this pollution described in the last point. With the Lagrangian particles we would like to propose to estimate the impact zone of this waste, inside and outside of Lake Nokoue.

For this, we will use 1000 buoys randomly dispersed around the point: longitude = 2.4359 and latitude = 6.3825 with a maximum variation in longitude and latitude of 0.0012° and 0.002° respectively. The position of these particles is shown in the figure below.

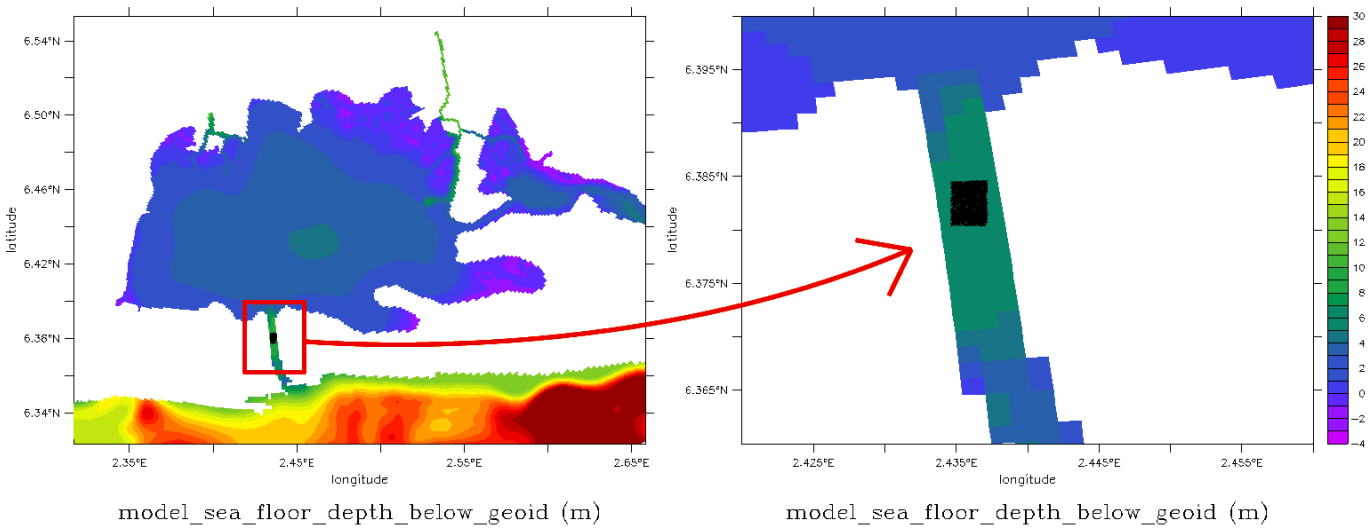


Figure 7 : Particles' initial positions in front of Dantokpa open-air Market.

Those positions are set in the **notebook_drifter_dantokpa** file stored into the **nokoue_lt_Drifter/NOTEBOOK** directory. Let's edit the local (**RDIR/nokoue_lt_Drifter**) **notebook_list.f** and set the name **notebook_drifter_dantokpa** instead of **notebook_drifter** for the parameter : **`nomfichier(16)`**.

In the **notebook_drifter_dantokpa** file, you may notice that we set the vertical velocity to - 2.5 mm/s, which is quite fast for debris. This velocity could be adjusted faster/slower.But for the training we want to get the particles settling fast enough to keep runs not too long.

The OFFLINE files stored in the nokoue_lt_Drifter/OFFLINE/ directory have two high-frequency (1h frequency) output periods. These two periods are respectively in March during the low water period, and in September-October 2017 during the high water period. For these two periods, you will of course find the neap and spring tide cycles. We have identified two interesting dates for each river flow regime.

- For march :

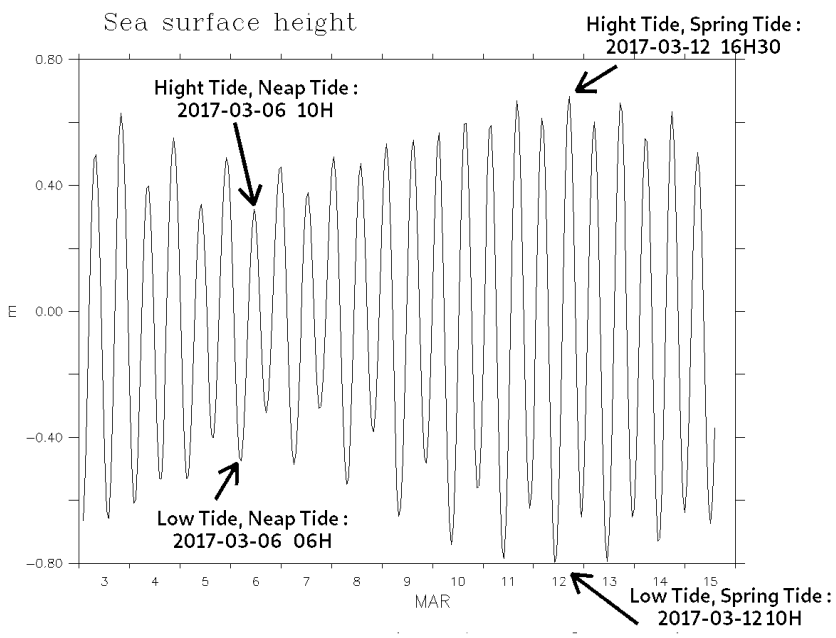


Figure 8 :

We identify :

- Neap tide the 2017/03/06 with high tide at 10h and low tide at 06h. Referenced as : **NT_HT_LW** and **NT_LT_LW**.

- Spring tide the 2017/03/12 with high tide at 16h30 and low tide at 10h. Referenced as : **ST_HT_LW** and **ST_LT_LW**.

- For autumn :

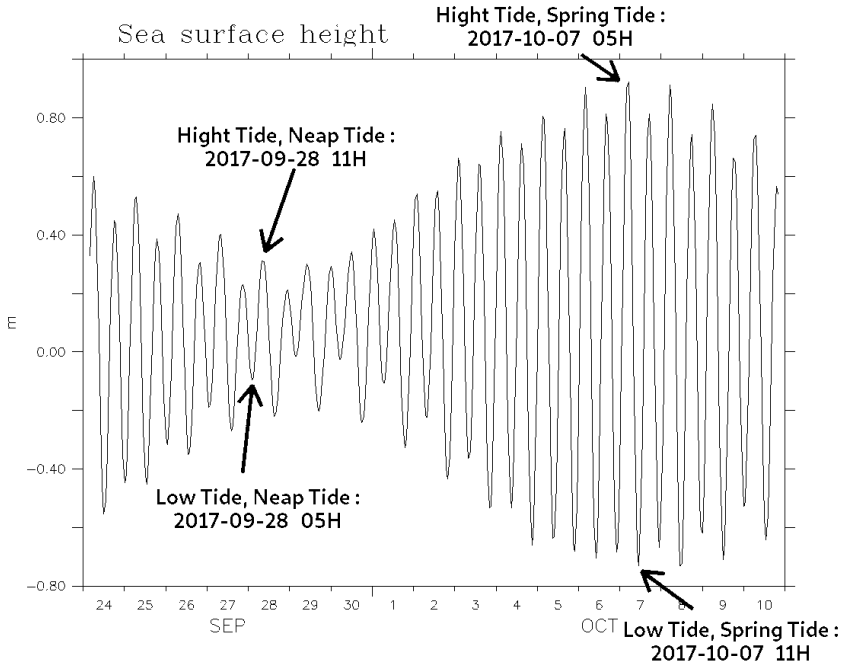


Figure 9 :

We identify :

- Neap tide the 2017/09/28 with high tide at 11h and low tide at 05h. Referenced as : **NT_HT_FL** and **NT_LT_FL**.

- Spring tide the 2017/10/07 with high tide at 05h and low tide at 11h. Referenced as : **ST_HT_FL** and **ST_LT_FL**.

So we will try to make some simulations for different situations to see how the particles will disperse depending on the time of the tide and also the moment of the river discharge season.

You can decide what period you want to run by simply fixing the start date and end date in the **notebook_time.f**. Edit the notebook and choose your first experiment period. Don't make the run too long, 10 days will do for a start. We set a couple of time periods already in the notebook, with the same name (for example : **NT_LT_LW** , it is probably the most interesting case, etc…) mentioned above so you can just pick one. You can simply comment or uncomment by putting or removing the character "!" at the desired lines.

In order to get some new output to deal with, in the **notebook_drifter_dantokpa**, we are going to change the parameter at line 4 and choose the option "2". This option will provide a **netcdf** format output files containing all drifters at a fixed time. You can set this time output with the value in seconds line 3. For the test every 1h will be way enough!

When ready run your simulation : **./s 4**

N.B. To run the Flood period case, on October, you will need to open the **notebook_offlines.f** and **change liste_offline_March.txt** to **liste_offline_Sept-Octo.txt** for the parameter **offlinefile(1) :**

kate ../../../nokoue_lt_Drifter/NOTEBOOK/notebook_offline.f

!offlinefile(1)='../../../nokoue_lt_Drifter/OFFLINE/RUN3D_Drifter/liste_offline_March.txt'

offlinefile(1)='../../../nokoue_lt_Drifter/OFFLINE/RUN3D_Drifter/liste_offline_Sept-Octo.txt'

After the run, you will find the output netcdf formatted files inside the **tmp** directory. To help visualize the drifters we create a python script to make several images : **drifter.make_plot_drifters.py** which is part of a python library : Spytools developed by SIROCCO. To run the script :

- set a conda environment :
  - source ~/.bashrc_conda
  - conda activate retrieve_data
- set the parameter file : ParamDrifter.ini

[Data_in]
expname=RUN1 -> Experiment name, bypassed if passed as argument, is used for naming the output images
dirin=./tmp -> Directory for input files
drifter_inifile=drifter_initial.nc -> file with initial position of drifters
pattern=drifter_201*.nc -> Pattern to create a list of input drifter files
dirgrd=./tmp -> Directory for the file grid
gridfile=grid.nc -> Grid file name
[Data_out]
dirout=./IMAGES/ -> Directory for input files
wanted_format=png -> Output image format
dpi_res=100 -> Output image resolution
if_animate=False -> Boolean for asking animated trajectories (it can take few minutes to proceed)
if_occurence=True -> Boolean for asking for occurrence map of the drifters
lonmin=2.22 -> Min Longitude for images
lonmax=2.845 -> Max Longitude for images
latmin=6.1 -> Min Latitude for images
latmax=6.55 -> Max Latitude for images
depmin=0 -> Min Depth for bathymetry background maps
depmax=40 -> Max Depth for bathymetry background maps
Modulo=1 -> Modulo for decimating the number of drifters to plot

- then run the command :

python -m drifter.make_plot_drifters ParamDrifter.ini Expname

where **Expname** is the name of your experiment, you can for example give **NT_LT_LW** if you have chosen this setting in the **notebook_time.f**. The submission script will run the python

script : **Make_Plot_Drifters.py** this script is using a parameter file named **ParamDrifter.ini**.

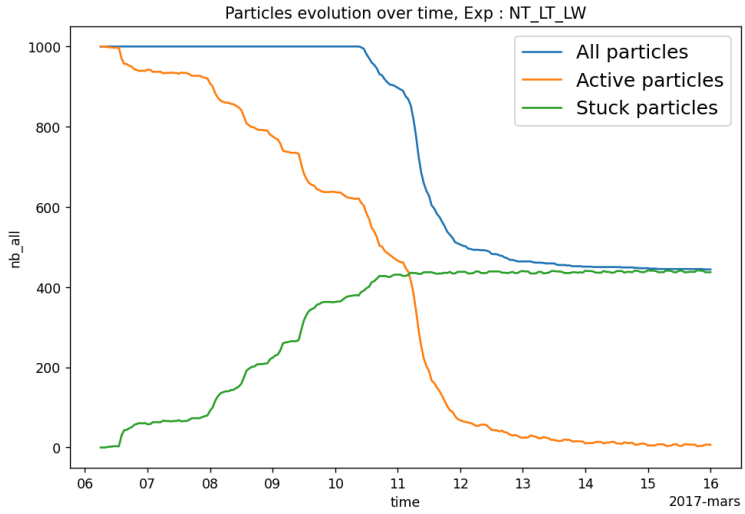
The script will send logs close to :

```
============================================================
=  Drifters analysis out of a list of symphonie files  =
============================================================

-------------------------------------
!!  Plot trajectories with color   !!
-------------------------------------
------------------------------------------------------------
-> There is xx Stuck Particles in this TEST simulation.
------------------------------------------------------------
---------------------------------------
!! Times series : particles in domain !!
---------------------------------------
-------------------------------------
!!  Making Animated trajectories   !!
!!  It would take few minutes      !!
-------------------------------------
-------------------------------------
!!  Plot Occurrences in grid cells  !!
-------------------------------------
mean  : xxxx
max   : xxxx

############################################################
############################################################
```

In the standard settings, you will get 5 images :

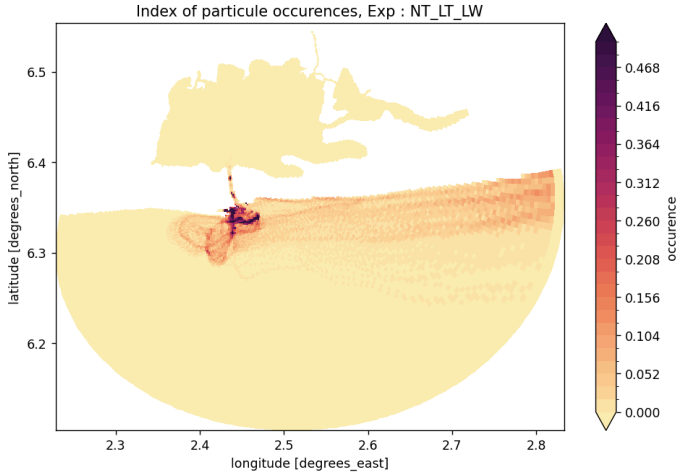
- Evolution_Particles_NT_LT_LW.png
- Some_trajectories_NT_LT_LW.png
- Some_trajectories_wtr_time_NT_LT_LW.png
- Stuck_Particles_NT_LT_LW.png (in the Log_Nokpython.out you will have also the number of stuck particles)
- Occurence_Particles_NT_LT_LW.png

  For example : Evolution_Patricles_NT_LT_LW.png



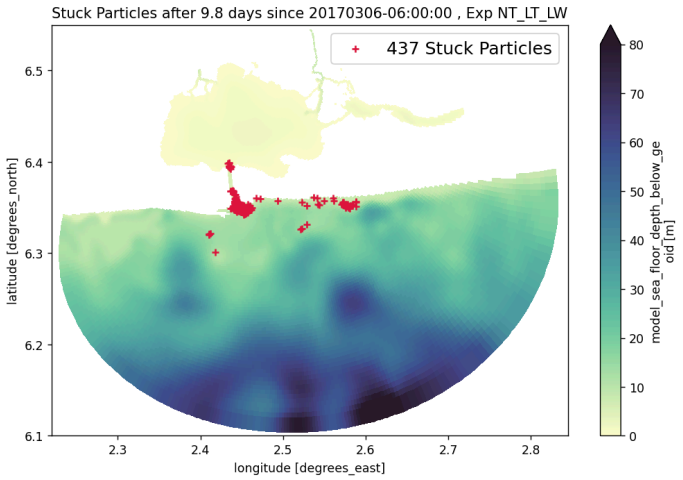
Give the time evolution of the number of active particles (in orange) inside the domain, and (in green) the number of particles stuck (which have not moved between two output time steps). in the domain.

Occurence_Particles_NT_LT_LW.png



This image gives a map of the number of occurrences of particles normalized by the max value. We count the number of particles passing through each grid cell.

Stuck particles locations : Stuck_Particles_NT_LT_LW.png



This image gives the locations and the number of stuck particles in the domain at the end of the simulation.

To play with the code, you could take the case of NT_LT_LW time, moving the initial time by one or two hours (earlier or later) and/or change the vertical velocity of the settling velocity in the **notebook_drifter_dantokpa**.

To help you to gain time, we have put the output netcdf files and the images and animation in the directory **RESULTS_DRIFTER/**. You will find the netcdf file in each directory : **RESULTS_DRIFTER/ST_HT_FL , RESULTS_DRIFTER/NT_LT_LW** etc….

To look at the animation, like : **animate_traj_NT_LT_LW.gif** you will need to download it on your local machine. Note that by default the animation is not produced by the python script..