Brief Report

SCHISM/WWM Tips and Tricks

Christelle Auguste [1]

[1] University of Tasmania; christelle.auguste@utas.edu.au

Contents

1. Introduction

This guide could be an appendix to the SCHISM manual available at: http://ccrm.vims.edu/schismweb/SCHISM_v5.8-Manual.pdf. It was written to help beginners better deal with SCHISM v5.8 and the different steps to run simulations. It aims to try to gather as much information and links as possible on how to produce the input files in one place for 2D barotropic model and only for hindcast data studies. Of course this will evolve with the development of SCHISM and the tools that surround it, but it can be a good start to dive into the SCHISM world. In this document the term "the manual" refers to the SCHISM Manual v5.8. This guide is primarily aimed at implementing SCHISM on a Linux platform, although, as you will see, our chosen grid generator is Windows-based.

2. Download and Compilation

2.1. SCHISM v5.8

The installation and compilation of SCHISM can be scary for beginners not used to Linux. Stay calm!

2.1.1. Download

The section 1.6 of the manual is very clear to download SCHISM modeling system.

2.1.2. Compilation

Following on the 1.7.1 of the manual for the compilation with Makefile (based in src folder):
this file is going to call the Make.defs.local (where you have updated the MPI compiler name, path names for netcdf library, and name of the .EXE) and your make.defs.local is going to call the file include_modules (where you turn on/off the modules).
For the Make.defs.local the parts you need to edit are:

```
FCS = /mpif90 or /gfortran or /ifort
FCP = /mpif90
CDFLIBS = -L/pathtoyourlibrary -lnetcdf -lnetcdff
CDFMOD = -I/pathtoyourmodules/include   # modules for netcdf

Example:
CDFLIBS = -L/usr/lib/ -lnetcdf -lnetcdff
CDFMOD = -I/usr/include -I/usr/local/include -I/usr/local/include/malloc
```

First you will certainly try without modules:

```
$ ~/schism$ cd src
$ ~/schism/src$ make clean
$ ~/schism/src$ make psc
```

And you will obtain by example "schism.exe". Copy it in a folder of your future run with the mandatory files. To run it:

```
$ ~/simu_test$ ./schism
```

2.2. Modules

Modules are described in Chapter 7 of the manual. To include these modules, you need to recompile SCHISM to have another executable. You have to turn on/off module(s) in "include_modules" (based in /schism/mk directory) then proceed as the first time. For example to turn on the Wave module:

```
# Wind wave model WWM
```

```
USE_WWM = yes
EXEC := $(EXEC)_WWM
```

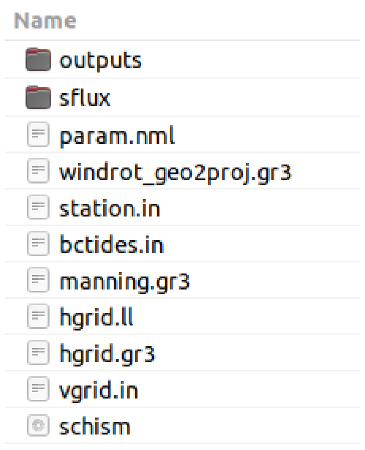
You will get the same beginning for the name of the executable but with the name of the modules at the end.
i.e.: "schism_WWM.exe"
If more than one module is turned on, they will all be tacked on to the end of schism_XX_XX.

3. Pre-Processing: Input files

This part provides additional information about the chapter 4 and 5 of the SCHISM manual (4: SCHISM I/O and 5: Using SCHISM).
Interesting links for the inputs files required by SCHISM:
http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/main.html
http://opencoasts.lnec.pt/index_en.php#eventos(Module T3)
http://ccrm.vims.edu/yinglong/SVN_large_files/NOAA-SCHISM-bootcamp-slides-Feb2021/
The mandatory input files are:

1. Horizontal grid : hgrid.gr3
2. Vertical grid: vgrid.in
3. Parameter input : param.nml
4. Boundary conditions input : bctides.in
5. Bottom drag input : drag.gr3/manning.gr3/rough.gr3

For the bottom drag input, drag.gr3 uses drag coefficient (dimensionless) and manning.gr3 uses the Manning's number n. Rough.gr3 uses C_D bottom roughness in meters with specific equations from the logarithmic boundary layer profile.
Example of directory for your SCHISM run is shown in Figure 1.



```
Name
outputs
sflux
param.nml
windrot_geo2proj.gr3
station.in
bctides.in
manning.gr3
hgrid.ll
hgrid.gr3
vgrid.in
schism
```

Figure 1. Example of directory for your run with mandatory and optional files for atmospheric forcing (sflux folder) and stations outputs

3.1. Mandatory file hgrid.gr3

The horizontal grid can be created with different tools, a selection of which are listed in Table 1:
The specific aspect of gridgen using SMS is described in Chapter 5 of the manual, here we focus on BlueKenue. Interesting information for grid generation can also be found in Module T5 of the OPENCoast tutorials:
http://opencoasts.lnec.pt/index_en.php#eventos

Table 1. Grid generators

Name	Node placement	Triangulation	Optimization	Boundaries	Availability
xmgredit	Y	Y	Y	Y	Free
SMS	Y	Y	Y	Y	Commercial
Gmsh	Y	Y	Y	Y	Free
OceanMesh2D	Y	Y	Y	Y	Free
JIGSAW	Y	Y	Y	Y	Free
ADmesh	Y	Y	Y	Y	Free
BlueKenue	Y	Y	Y	N	Free
triangle	Y	Y	N	N	Free
nicegrid	N	N	Y	N	Free

3.1.1. Mesh creation with Blue Kenue

First step is to download and install Qgis(Open source; Linux/Windows) and Blue Kenue (Open source; Windows-based application). For Blue Kenue:
https://nrc.canada.ca/en/research-development/products-services/software-applications/blue-kenuetm-software-tool-hydraulic-modellers
Then in Qgis you can import the shoreline for the area, adjust it to your domain and export it in a shapefile format. There are several sources to download shoreline in your areas of interest (Openstreetmap, NOAA with the Global Self-Consistent Hierarchical, High-resolution Geography Database, local navy charts etc...). Shapefile with coordinates in lon/lat can be saved in UTM using the option "save vector layer as CRS=..." (ex: 32755 for Tasmania).
In BlueKenue (BK), import the shapefile, create the domain with shoreline + additional lines, upload bathymetry (Digital Elevation Model) and generate your mesh following these tutorials (9 steps) on youtube:
https://www.youtube.com/watch?v=hFUycwskJUo&t=2s

Precision : once the coastline is appended in the domain, you may need to delete part of coastline to have only one line (no overlap between layers) at the same location (otherwise creation of error during the extraction).
Once your mesh is created you have to export it in SMS format(.2dm). Then you can convert it to .gr3 using the perl script "2dm2gr3_m2m.pl" (link : *schism/src/Utility/SMS*) with the following commands:

```
$ ./2dm2gr3_m2m.pl [version of SMS 0 1 2] [yourfile.2dm] [hgrid.gr3]
```

For the version of SMS from BK: 0 is working. For a quick check open the .gr3 file in xmgredit5 (you have to compile it before, see the file "install_notes_new" in *schism/src/Utility/ACE*), you will have an error if it's not correct. You can also check if there is the number of lines in the file corresponds to the number of elements directly by opening the file.
If your bathymetry is negative, transform it with xmgredit5:

```
./xmgredit5 hgrid.gr3
```

Then: Edit/Edit over regions/Evaluate and then in the dialogue box type depth=depth*(-1).
Beware of CFL number condition, being an implicit model SCHISM require CFL>0.4 (see manual part 5.1.1 & 5.1.2). The general rules are well explained in the section 5.1 of the manual.

3.1.2. Option optimization with nicegrid

You can optimize your mesh with the tool nicegrid2 (https://adcirc.org/home/related-software/adcirc-utility-programs/,download then compile), example is shown in Figure 2.

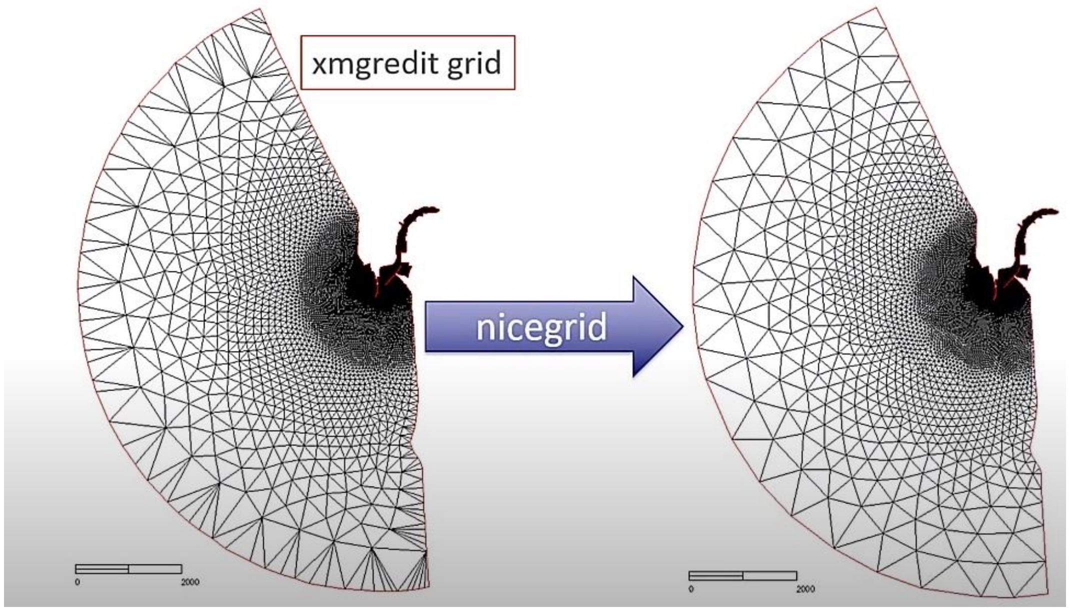


Figure 2. Example of optimization of mesh with nicegrid (from Module T5 of OPENCoast Tutorials)

3.1.3. Define boundaries

Boundaries can be defined easily with xmgredit5. You have to define the boundaries (anti-clockwise) see this GIF for all the steps:
http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/hgrid.html
Save file as bnb file. And check at the interface open bnd/land that the nodes are well defined.

3.1.4. Completion of hgrid.gr3, hgrid.ll as required for SCHISM

You can create the lat/lon version of your grid from your UTM domain with another perl script "proj_wrap.pl " (Location: *schism/src/Utility/Pre − Processing*):

```
$ ./proj_wrap.pl [epsg: "code␣zone␣of␣your␣area"]  [1: proj to ll; 2: ll to proj]
[1: input format is .gr3; 2: input xyz] [file.gr3] [file.ll]
[for xyz input, separator (1: space; 2: comma)]
[for xyz input, line number (0: without; 1: with)]
#example:
$ ./proj_wrap.pl epsg:32755 1 1 hgrid.gr3 hgrid.ll 1 1
```

The file hgrid.ll (coordinates in lat/lon) is mandatory for the creation of sflux files for atmospheric forcings.
Then merge hgrid.gr3 and hgrid.ll with the boundaries file created with xmgredit5:

```
$ cat bnb >> hgrid.gr3
```

You're ready to check grid issues by running the pre-processor flag (ipre=1).

3.2. Mandatory file vgrid.in

The section 4.2.2 in the manual explained the different cases. Figure 3 shows an example of a SZ model.
For 2D barotropic model you can use the "pure S" model:

```
2 !ivcor (SZ=2, LSC^2=1)
2 1 1.e6 ! Nz (# of Surface−levels), kz (# of Z−levels); h_s (transition depth between
Z levels
1  −1.e6  ! coordinate of the last Z−level must match −h_s
```

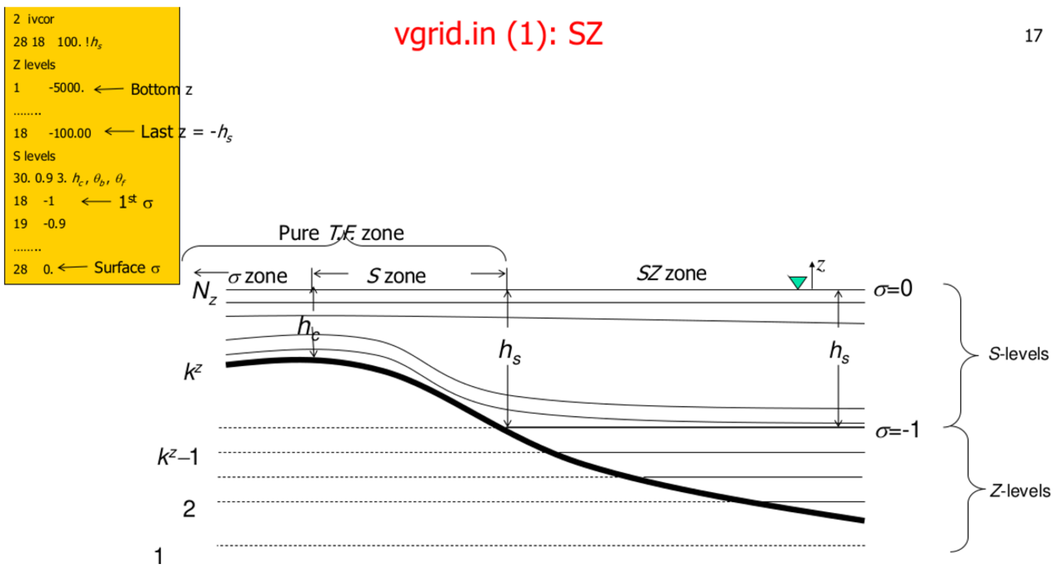


Figure 3. "vgrid.in" diagram

```
S levels
40. 0. 1.e−4   ! constants used in S−transformation: h_c, theta_b, theta_f
   1     −1. !first S−level ( first sigma coordinate must be −1)
   2     0.  !last S−level ( last sigma coordinate must be 0)
```

3.3. Mandatory file param.nml

This file is well described in section 4.2.4 of the manual, and there are examples of this file in *schism/sample_inputs* and in the test suite "schism_verification_tests".

3.4. Mandatory file bctides.in

All the options supported by SCHISM are described in the section 4.2.3 of the manual. A good description of the file is available with this link:
http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/bc.html

3.4.1. Matlab

You have a matlab code "one_click_bctides_schism_fes2014" in folder *schism/src/Utility/Tides* to create bctides with FES2014 data. First you have to download the FES2014 data, e.g. from https://www.aviso.altimetry.fr/en/data/products/auxiliary-products/global-tide-fes/.

3.4.2. Python

For TPXO there is the possibility to use pychsim developed by Jaime Cazalda:
https://github.com/schism-dev/pyschism
After installation, the cmd line to have information:

```
$ pyschism  bctides −h
usage: pyschism bctides [−h] [−−output−file OUTPUT_FILE] [−−include−velocity]
 [−−tidal−database {hamtide,tpxo}] [−−Z0 Z0] [−−hgrid−crs HGRID_CRS]
 [−−overwrite] [−−log−level {critical,fatal,error,warn,warning,
   info,debug,notset,detail,partitioning}] hgrid start_date run_days

positional arguments:
  hgrid
  start_date
  run_days

optional arguments:
  -h, --help            show this help message and exit
  --output-file OUTPUT_FILE
  --include-velocity
  --tidal-database {hamtide,tpxo}, --tidal-db {hamtide,tpxo}
  --Z0 Z0
  --hgrid-crs HGRID_CRS
  --overwrite           Allow overwrite of output file.
  --log-level {critical,fatal,error,warn,warning,info,debug,notset,
  detail,partitioning}
```

After downloading the data from TPXO (i.e. from https://www.tpxo.net/: h_tpxo9.v1.nc and u_tpxo9.v1.nc), copy the netcdf TPXO files to $Home/.local/tpxo/*.nc and then you can create the "bctides.in" file:

```
$ pyschism bctides [your file hgrid.ll] YYYY-MM-DDT00:00:00
[number of days for simulation] --tidal-database=[your source]
--include-velocity --output-file=bctides.in\\
Example:
$ pyschism bctides hgrid.ll 2018-03-01T00:00:00 7
--tidal-database=tpxo --include-velocity --output-file=bctides.in
```

This will put into "bctides.in" for each constituent requested:

- angular frequency and tidal species (not varying with time and space)
- nodal factor and earth equilibrium argument (varying with time)
- amplitude and phases of constituents at the boundaries (varying with space)

3.5. Mandatory file "bed roughness".gr3

Information at:
http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/other_gr3.html http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/rough.html

3.5.1. Manning.gr3

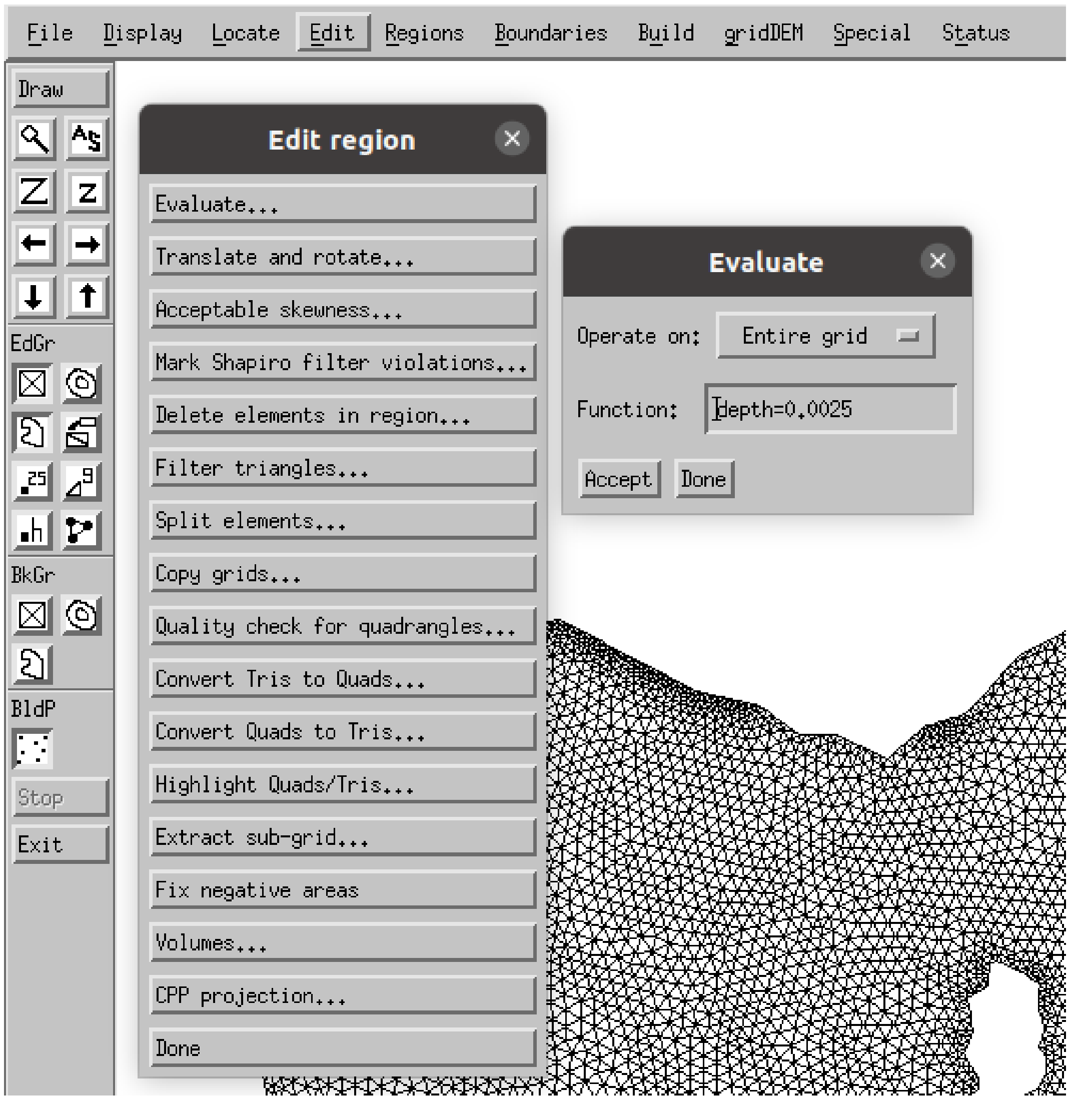
For a constant value of the Manning number([n]=$s/m^{1/3}$) follow these steps (Figure 4):

- open hgrid in xmgredit5
- Edit grid/region
- Evaluate
- depth=value of Manning's number n
- save as manning.gr3

You can also use BlueKenue to create a spatially varying Manning.gr3 (Credit to Huy Quang Tran). See the steps in the pdf under /src/Utility/Pre-Processing/step_by_step_bottom_friction_BK.pdf.

3.5.2. Rough.gr3

Bottom roughness in meters are read from rough.gr3. [from Joseph: we've removed negative roughnes in newer versions like v5.9] In this case, negative or 0 depths in rough.gr3 indicate time-independent C_D, not roughness. With this option, two additional parameters are needed in param.nml: dzb_{min} in meters and dzb_{decay} nondimensional. Information can be found in the section 4.2.4.2 of the manual and in param.nml for the calculation of C_D for rough.gr3:



Operate on: Entire grid

Function: Depth=0.0025

Figure 4. Steps for the creation of manning.gr3

```
Cd is calculated using :
 the log law, when dzb>=dzb_min;
when dzb<dzb_min, Cd=Cdmax*exp[dzb_decay*(1-dzb/dzb_min)],
where Cdmax=Cd(dzb=dzb_min),
and dzb_decay (<=0) is a decay const specified below.
We recommend dzb_decay=0.
```

3.6. Optional files sflux.nc for atmospheric forcing

The section 4.3.8 of the manual described how to generate your own sflux files with matlab, and gave the conventions for the nc files.
Others useful links:
http://ccrm.vims.edu/w/index.php/Atmospheric_forcing
http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/sflux.html

Here is underlined the method to create sflux_air only with python from GFS data.
GFS data for SCHISM is available at https://rda.ucar.edu/datasets/ds084.1/ (need to register for access), you need to have the start of your downloaded data file = start_time of your simulation.
Grab the following fields:

- wind speed (u,v) (10m above MSL)
- air pressure (MSL)
- surface air T (2m above MSL)
- specific humidity (2m above MSL)

Merge by time each variables with the command cdo (download from https://sourceforge.net/projects/cdo/):

```
cdo mergetime gfs.0p25.*.f000.grib2.yourname.nc output.nc
```

Then merge all variables:

```
cdo merge [].nc [].nc [].nc sflux.nc
```

Then change the time units to "days since start_date" with cdo (before it was hours since 1880):

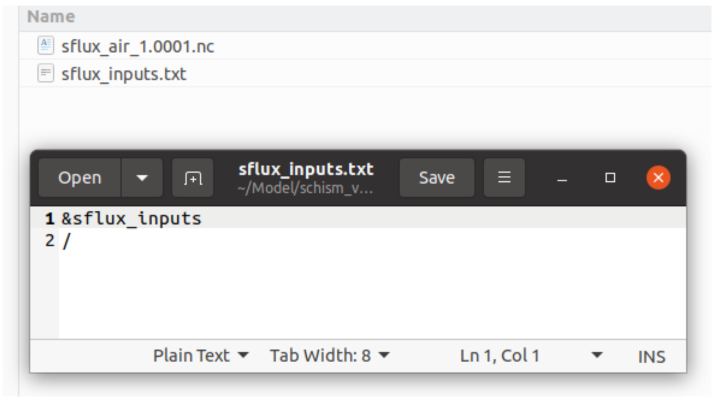
```
cdo -setreftime,2018-03-16,00:00:00 sflux.nc sflux_new.nc
cdo -settunits,day sflux_new.nc sflux_air.nc
OR in one command
cdo -setreftime,'2018-03-16','00:00:00',day sflux.nc sflux_air.nc
```

Now you need to put all these variables in a new NetCDF compatible for SCHISM (sflux_air_1.0001.nc), using the script in Appendix A.1. Put the file created in a folder "sflux" with a file "sflux_inputs.txt" (In this file advanced users can specify parameters like renaming variable names, resetting max time window etc. Most users won't bother about these), and this folder in the folder of your run (Figure 5).
Then create the file windrot_geo2proj.gr3 : it's hgrid.gr3 with rotation value instead of depth . It rotates the wind in case they do not align with coordinates axes. If no rotation is needed put 0: you can use xmgredit5 for that by changing all values of 'depth' to 0 and saving it as windrotgeoproj.gr3.
If you use this option don't forget to put nws=2 in param.nml.

3.7. Optional files elevation non tidal (2D): elev.th.nc

Here is described the way to deal with global ocean (Mercator) data. First download the data, available through CMEMS Data Access Portal (https://marine.copernicus.eu/, need to register first):

- 'zos' (sea surface height above geoid)



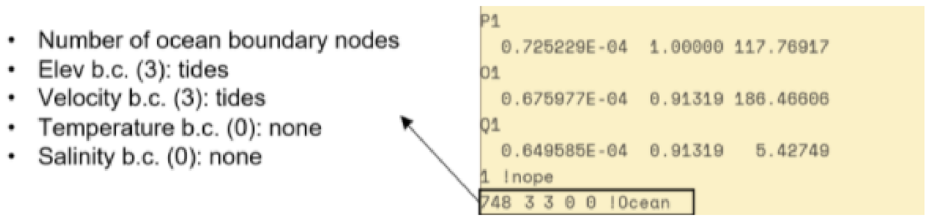
```
&sflux_inputs
/
```

Figure 5. Example of contents for sflux folder, also showing the contents the file sflux_inputs.txt

Then change the reftime to start_time and units to seconds.

```
cdo -setreftime,YYYY-MM-DD,HH:MM:SS,second yourfile.nc elev.nc\
```

Run the python script "ElevUV_bnb.py" (see Appendix A.2) to produce the "elev.th.nc" file for elevation. Then in bctides.in change the options for elevation: here 5 instead of 3, so the last line of Figure 6 will become 748 5 3 0 0. Reminder all options for bctides are explained in table 4.1 of the manual.
It's also possible to include non-tidal velocities by downloading eastward and northward velocities



```
P1
  0.725229E-04   1.00000 117.76917
O1
  0.675977E-04   0.91319 186.46606
Q1
  0.649585E-04   0.91319   5.42749
1 !nope
748 3 3 0 0 !Ocean
```

Figure 6. Part of bctides.in to modify (from http://ccrm.vims.edu/yinglong/feiye/Workshop_20190701/TEMP/Doc/bc.html)

and creating "uv3D.th.nc". (Script in progress).

3.8. Optional file stations.in

This file is to extract data (elev, air, pressure, windx, windy, T,S, u,v,w) at specific stations (model locations). The description can be found in section 4.3.7.1 of the manual.
Example:

```
1 1 1 1 1 1 1 1 1   !on (1)|off(0) !flags for elev, air pressure,
windx, windy, T, S, u, v, w
4                   !# of stations
1 6.5833 54.0000 0 !Format: station #,x,y,z;
if ics=2, x,y are degr in lon/lat. #ics is defined in param.nml
2 7.1583 55.195 0
3 6.35   54.1667 0
4 8.4514 54.7942 0
```

4. Coupling with WWM (Wave Wind Model)

There are three mandatory files for the WWM:

- wwminput.nml (equivalent of param.nml)
- wwmbnd.gr3 (file containing the description of the boundary condition)
- hgrid_wwm.gr3 (copy of hgrid.gr3)

4.1. Mandatory file wwminput.nml

Example can be found in *schism/sample_inputs* for wwminput.nml. You have to update this file to match param.nml (i.e.: dtstep, start time of simulation), define your parameters, your option for wave forcing at the boundaries (IBOUNDFORMAT) and the start date of your wave forcing files. The manual under github (https://github.com/schism-dev/schism/tree/master/src/WWMIII/Manual) is obsolete for some sections, especially for the forcing part at the boundary (IBOUNDFORMAT). Two options are described here:

- IBOUNDFORMAT=3 is for bulk parameters, requires 5 nc files (hs, t02, dir, spr, fp) and a parametric spectra is imposed along the open boundary considering a JONSWAP spectrum or other.
- IBOUNDFORMAT=6 is for wave spectra from WWIII or any equivalent model able to produce the variables requested. One netcdf file with the following variables:
  - efth (directional variance spectral density)
  - dpt (depth)
  - wnd (wind speed)
  - wndir (wind direction)
  - cur (current speed)
  - curdir (current direction)
  - frequency 1 (lower frequency)
  - frequency 2 (higher frequency)

4.1.1. Bulk parameters

For IBOUNDFORMAT=3: write the name of the 5 nc files in a file called 'bndfiles.dat', the order is very important, to confirm someone? I have not yet managed to get this option to work in v5.8.:

1. dir
2. fp
3. hs
4. spr
5. t02

Copy these files in your folder for the run. Create wwwbnd.gr3 with python script (see Appendix A.3), and update param.nml to add the WWM global outputs: example in folder Test_WWM_Analytical (test suite : schism_verification_tests) and to ensure msc2 and mdc2 are equal to values in wwminput.nml. Important items to check:

- INITSTYLE= 1 (parametric Jonswap)
- LBCWA=T (Parametric Wave Spectra, T/F=TRUE/FALSE)
- LBCSP=F (Specify (non-parametric) wave spectra, specified in 'FILEWAVE' below), T/F=TRUE/FALSE

4.1.2. Wave spectra

With IBOUNDFORMAT=6, you have to specify for the item FILEWAVE the name of the netcdf with all the spectral data.
Important items to check:

- INITSTYLE= 2 (read from global nc file you define in FILEWAVE)
- LBCWA= F (Parametric Wave Spectra)
- LBCSP= T (Specify (non-parametric) wave spectra, specified in 'FILEWAVE' below)

4.2. Mandatory file wwmbnd.gr3

For wwmbnd.gr3, it's a .gr3 style, you have to change the depth variable as follows:

1. put 0 for every grid point (easily done with xmgredit5)
2. put 1 for land boundary
3. put -1 for island boundary
4. put 2 for open boundary with Dirichlet conditions
5. put 3 for open boundary with Neumann conditions (i.e. zero gradient)

To create this file you can find a fortran script under github, a matlab script in the archives of schism_users lists, and a python script in Appendix A.3. There may be some duplicated nodes that are between land and open boundaries, you can manually work out to remove and keep the correct one.

4.3. Data for bulk parameters

4.3.1. Ifremer data

Ftp links must be open in a dedicated software (like filezilla for linux).
ftp://ftp.ifremer.fr/
Alternatively, an option is to download data (hs, fp , spr, t02, dir) with a bash script "get_WW3.sh" (Credit to Ivica Janekovic):

```
# input argument is yyyy mm
YEAR=$1
MNTH=$2
vars=( "fp" "dir" "hs" "t02" "spr" )
for i in "${vars[@]}"
do
echo "getting $i for ${YEAR} ${MNTH}"
wget -q --progress=dot "ftp://ftp.ifremer.fr/ifremer/ww3/HINDCAST/GLOBAL/${YEA\
R}_ECMWF/$i/WW3-GLOB-30M_${YEAR}${MNTH}_$i.nc" .
echo "--------------------------------------------------------------------------\
--------------------------------"
done
```

cmd line:

```
./get_WW3.sh 2018 03
```

(may need to do chmod +x get_WW3.sh)
Then subset to your domain with:
ncks -dlatitude, min, max, stride -dlongitude, min, max, stride in.nc out.nc
Do this for each single variable nc file containing the entire period (not necessary to have the same start date as in param.nml).
In this example all variables are by month, we have 2 files for Hs (March and April)

```
cdo mergetime inputs.nc output.nc
Ex: cdo mergetime hs_march.nc hs_april.nc hs_0304.nc
```

You have to do this for each variable and you will obtain 5 nc files which you have to put in the directory of your run (Figure 7).

4.3.2. CAWCR data

The Centre for Australian Weather and Climate Research (CAWCR) is a partnership between the Bureau of Meteorology and CSIRO. These wave fields are generated using the WWIII model and the CFSR surface winds.
Link : http://data-cbr.csiro.au/thredds/catalog/catch_all/CMAR_CAWCR-Wave_archive/CAWCR_Wave_Hindcast_aggregate/gridded/catalog.html.
Input data: NCEP CFSR surface winds and sea ice, http://cfs.ncep.noaa.gov/cfsr/.
Wave model used: WaveWatch III, http://polar.ncep.noaa.gov/waves/wavewatch/wavewatch.shtml.

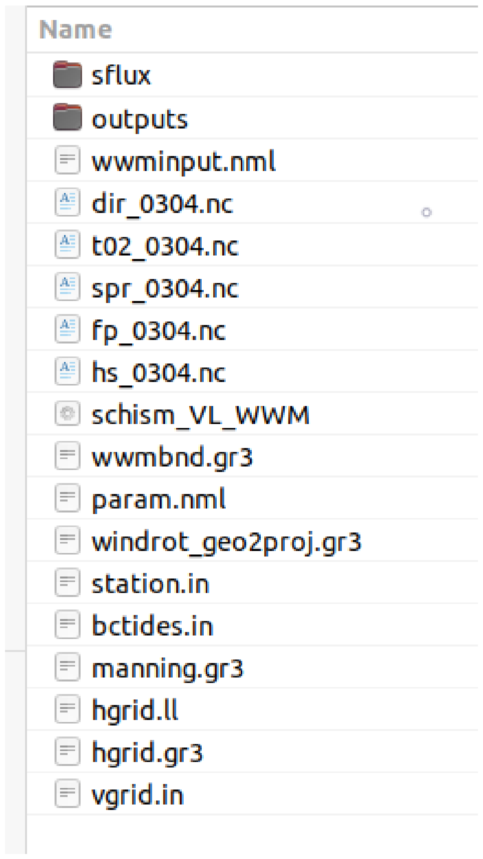
With Netcdfsubset you can subset to your domain and download nc files per month, for example "ww3.aus_4m.YEARMONTH.nc"
You can download variables by variables or all variables in one file. If you download all variables together, to have each variable in a separate file (e.g. 1 file for hs, 1 file for dir etc.) you need to use the following commands:

```
$ ncks –O –v hs gridded_ww3.aus_4m.201803.nc hs_03.nc #save only hs in new file
$ ncrcat hs_03.nc hs_04.nc hs_0304.nc #merge months
```

4.4. Data for wave spectra

The variables to download are:

- efth (directional variance spectral density)
- dpt (depth)
- wnd (wind speed)



```
Name
sflux
outputs
wwminput.nml
dir_0304.nc
t02_0304.nc
spr_0304.nc
fp_0304.nc
hs_0304.nc
schism_VL_WWM
wwmbnd.gr3
param.nml
windrot_geo2proj.gr3
station.in
bctides.in
manning.gr3
hgrid.ll
hgrid.gr3
vgrid.in
```

Figure 7. Configuration of SCHISM for coupling with waves and with atmospheric forcings

- wndir (wind direction)
- cur (current speed)
- curdir (current direction)
- frequency 1 (lower frequency)
- frequency 2 (higher frequency)

4.4.1. Ifremer data

Ftp links (ftp.ifremer.fr) must be open in a dedicated software (like filezilla for linux) or you can use the python script "ncget.py" in Appendix A.4.
Example of path:
$/ifremer/ww3/HINDCAST/GLOBMULTI\_ECMWF\_02/GLOB-30M/2018/SPEC\_SE$
Time period for:

- ERA5: 1993-2020
- ECMWF: 2018-2019

4.4.2. CAWCR data

Spectra data can be found at :
http://data-cbr.csiro.au/thredds/catalog/catch_all/CMAR_CAWCR-Wave_archive/CAWCR_Wave_Hindcast_aggregate/spec/catalog.html
Time period: 1979-Jan to now

5. Post-processing

The easiest way to visualize SCHISM nc4 outputs is decribed in section 5.3 of the manual: VisIT (you have to install it separately). Useful post-processing tools can be found in the *src/Utility* directory and are described in the section 5.4 of the manual. Here some commands are detailed for the combination of nc4.

1. to work with VisIT tool
   Use the combine_ouptut11.f90 (need to compile it first, follow instruction in the header of the file). Then in the outputs directory of your run:

   ```
   outputs$ ./combine_output11 -b [start number of output file]
   -e [end number of output file]
   outputs$ ./combine_output11 -b 1 -e 5 #example
   ```

   This means that files schout_0000_1.nc, schout_0000_2.nc etc will become schout_1.nc etc see Figure 8. For more information on partitioning (4-digit zeros) see the manual section 4.4.2. Then you can launch VisIT to visualize the results (Figure 8):

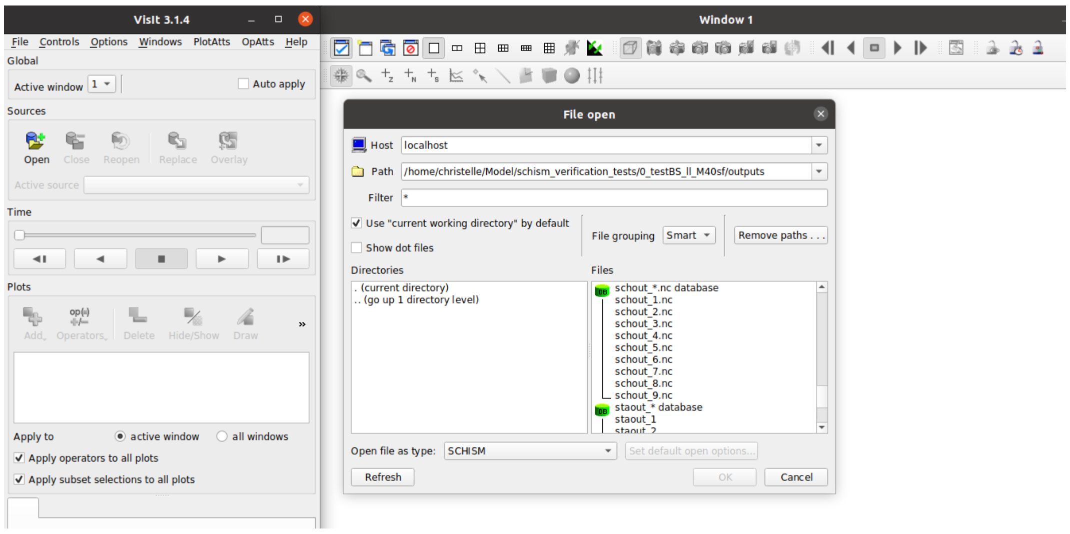
   ```
   .../visit/bin$ ./visit
   ```

2. to work with python, you can use the following command to deal with one file

   ```
   outputs$ ncrcat schout_0000_[123456789].nc schout_combine.nc
   ```

6. Full example for 2D model barotropic with tide forcing

-Install and compile SCHISM to get schism.exe
-Install tools for pre-processing : xmgredit5, Qgis, Blue Kenue
-Create your mesh in Blue Kenue, export it in SMS format, convert it to .gr3 with perl script (Figure 9)
-Create your vertical grid vgrid.in with example given in the manual
-Create bctides.in with pyschism (Figure 10)
-Create your bottom drag file : manning.gr3


VisIt 3.1.4
File Controls Options Windows PlotAtts OpAtts Help
Global
Active window 1
Auto apply
Sources
Open Close Reopen Replace Overlay
Active source
Time
Plots
Add Operators Delete Hide/Show Draw
Apply to
active window
all windows
Apply operators to all plots
Apply subset selections to all plots
Window 1
File open

Host: localhost
Path: /home/christelle/Model/schism_verification_tests/0_testBS_ll_M40sf/outputs
Filter: *
[✓] Use "current working directory" by default
[ ] Show dot files
File grouping: Smart
Remove paths . . .

Directories:
```
. (current directory)
.. (go up 1 directory level)
```

Files:
```
schout_*.nc database
  schout_1.nc
  schout_2.nc
  schout_3.nc
  schout_4.nc
  schout_5.nc
  schout_6.nc
  schout_7.nc
  schout_8.nc
  schout_9.nc
staout_* database
  staout_1
  staout_2
```

Open file as type: SCHISM
Set default open options...
Refresh OK Cancel

Figure 8. Screenshot for opening a file in VisIT

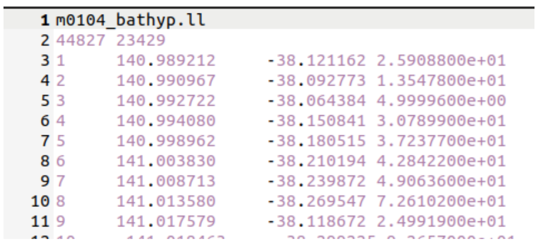
-Create your file station.in if needed
-Create your param.nml from template
-Put all of these files in a directory for your run
-Create a folder inside called outputs (see Figure 11)
-You're ready to test your mesh with ipre=1 in param.nml (you can do this step as soon as you have your mesh ready)
-It's successful? So you can now run your simulation with ipre=0 -And you can see the files in outputs following a successful run in Figure 12. Here the full example for 2D model barotropic with tide forcing was created to develop a tidal model for Banks Strait in Bass Strait, Tasmania. And we aimed to obtain elevation and depth-averaged velocities in all domain and at specific stations (= location of ADCP data).
Global output is done every "nspool" steps, and a new output stack is opened every "ihfskip" steps (these parameters are defined in param.nml), you obtained schout_0000_1.nc etc. The files staout_1 etc are the outputs for the stations you defined in stations.in. Mirror.out, fatal.error, nonfatal_0000 are main diagnostic output files.

7. Error messages when running simulations

7.1. sflux files

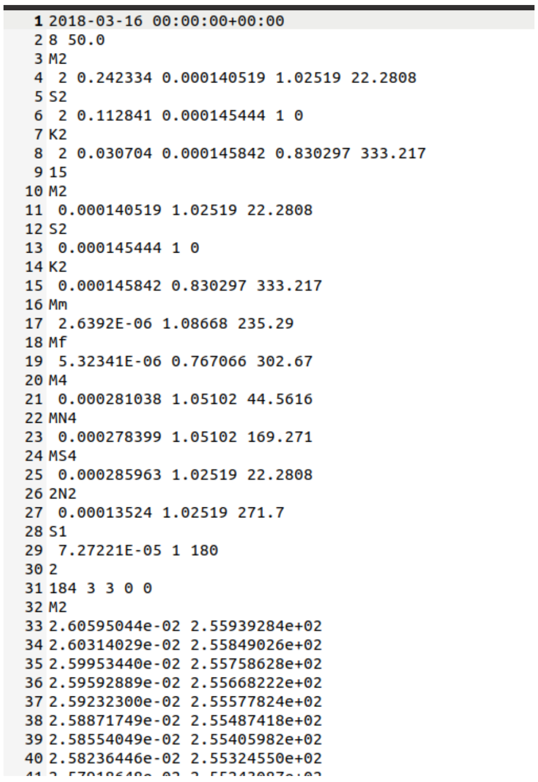
```
0: ABORT:  Could not find suitable element in input grid for output node
# 1 x_out, y_out =    2.6708680168154237
-0.47438771635516203
in_elem_for_out_node, ae_min =          0    1.0000000000000001E+025
```

This imply x1,x2,x3 are opposite, i.e. clockwise instead of anticlockwise in your input grid sflux.



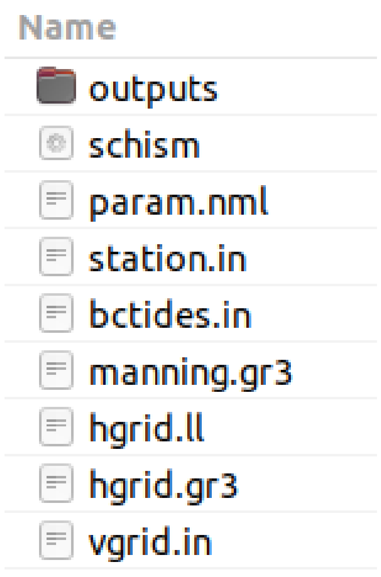
```
m0104_bathyp.ll
44827 23429
1       140.989212      -38.121162 2.5908800e+01
2       140.990967      -38.092773 1.3547800e+01
3       140.992722      -38.064384 4.9999600e+00
4       140.994080      -38.150841 3.0789900e+01
5       140.998962      -38.180515 3.7237700e+01
6       141.003830      -38.210194 4.2842200e+01
7       141.008713      -38.239872 4.9063600e+01
8       141.013580      -38.269547 7.2610200e+01
9       141.017579      -38.118672 2.4991900e+01
```

Figure 9. Start of hgrid.gr3 file



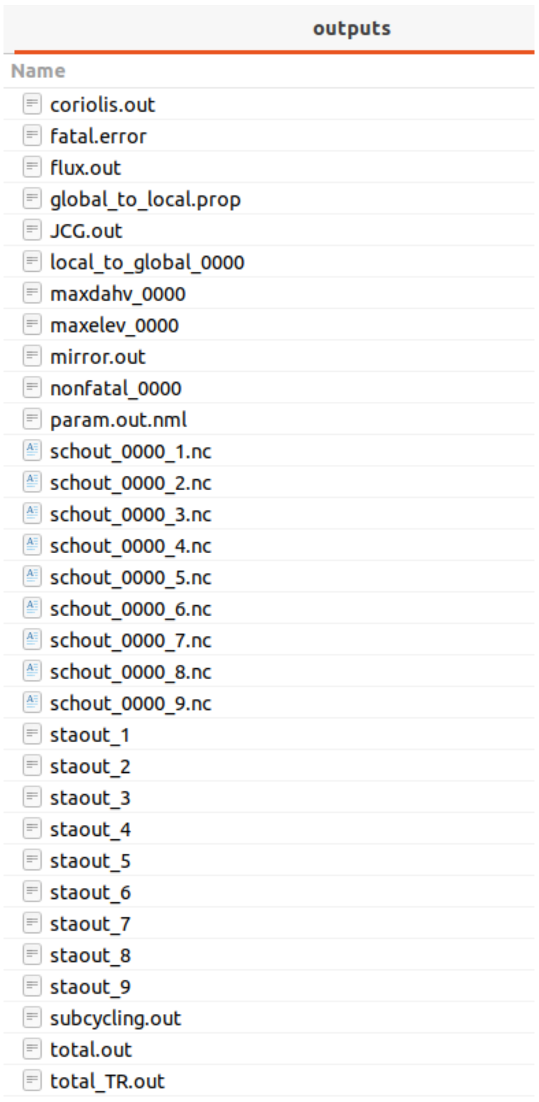
```
2018-03-16 00:00:00+00:00
8 50.0
M2
 2 0.242334 0.000140519 1.02519 22.2808
S2
 2 0.112841 0.000145444 1 0
K2
 2 0.030704 0.000145842 0.830297 333.217
15
M2
 0.000140519 1.02519 22.2808
S2
 0.000145444 1 0
K2
 0.000145842 0.830297 333.217
Mm
 2.6392E-06 1.08668 235.29
Mf
 5.32341E-06 0.767066 302.67
M4
 0.000281038 1.05102 44.5616
MN4
 0.000278399 1.05102 169.271
MS4
 0.000285963 1.02519 22.2808
2N2
 0.00013524 1.02519 271.7
S1
 7.27221E-05 1 180
2
184 3 3 0 0
M2
2.60595044e-02 2.55939284e+02
2.60314029e-02 2.55849026e+02
2.59953440e-02 2.55758628e+02
2.59592889e-02 2.55668222e+02
2.59232300e-02 2.55577824e+02
2.58871749e-02 2.55487418e+02
2.58554049e-02 2.55405982e+02
2.58236446e-02 2.55324550e+02
```

Figure 10. Start of bctides.in file



```
Name
outputs
schism
param.nml
station.in
bctides.in
manning.gr3
hgrid.ll
hgrid.gr3
vgrid.in
```

Figure 11. Configuration for 2D barotropic with only tidal forcing



```
outputs
Name
coriolis.out
fatal.error
flux.out
global_to_local.prop
JCG.out
local_to_global_0000
maxdahv_0000
maxelev_0000
mirror.out
nonfatal_0000
param.out.nml
schout_0000_1.nc
schout_0000_2.nc
schout_0000_3.nc
schout_0000_4.nc
schout_0000_5.nc
schout_0000_6.nc
schout_0000_7.nc
schout_0000_8.nc
schout_0000_9.nc
staout_1
staout_2
staout_3
staout_4
staout_5
staout_6
staout_7
staout_8
staout_9
subcycling.out
total.out
total_TR.out
```

Figure 12. Contents of outputs folder for 2D barotropic run with only tidal forcing

Acknowledgments: Thanks to Xavier Bertin, Joseph Zhang, Vanessa Hernaman, Jerome Lefevre, Ivica Janekovic, Jaime Calzada, Huy Quang Tran and Roger Proctor.

Appendix A

These scripts are in Github: https://github.com/Krys1202/Guide_SCHISM/tree/main/Python_scripts

Appendix A.1 Python script to create sflux air 1.0001.nc

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 21 14:41:09 2021
@author: Christelle Auguste inspired by "gfs.py" from pyschism and matlab scripts in src/Utility
"""
from netCDF4 import Dataset,num2date
from datetime import datetime
import numpy as np

#define your domain
lon_points = np.arange(lon value,lon value,0.25)
lat_points = np.arange(lat value,lat value,0.25)
lon , lat = np.meshgrid(lon_points, lat_points)

##create new netcdf
filename = 'sflux_air_1.0001.nc'
#nc = Dataset(filename, 'w', format='NETCDF4')
nc = Dataset(filename, 'w', format='NETCDF3_CLASSIC')
nc.setncatts({"Conventions": "CF-1.0"})
nc.createDimension('time', None)
nc.createDimension('ny_grid', len(lat_points))
nc.createDimension('nx_grid', len(lon_points))

lon_var = nc.createVariable('lon', np.float32, dimensions=('ny_grid','nx_grid'),)
lon_var[:]= lon
lon_var.units = 'degrees_east'
#lon_var.axis = 'X' # Optional
lon_var.standard_name = 'longitude'
lon_var.long_name = 'Longitude'

lat_var = nc.createVariable('lat', np.float32, ('ny_grid','nx_grid'),)
lat_var[:] = lat
lat_var.units = 'degrees_north'
#at_var.axis = 'Y' # Optional
lat_var.standard_name = 'latitude'
lat_var.long_name = 'Latitude'


time_units='days since 20XX-XX-XX'
time_var = nc.createVariable('time', np.float32, ('time',))
time_var.units = time_units
```

```
#time_var.axis = 'T'  # Optional
time_var.standard_name = 'time'
time_var.long_name = 'Time'
time_var.base_date=(20XX, XX, XX, 0)

prmsl_var = nc.createVariable('prmsl', datatype=np.float32,
                              dimensions=('time', 'ny_grid', 'nx_grid'),
                              zlib=True)
prmsl_var.units = 'Pa'
prmsl_var.standard_name = 'Pressure reduced to MSL'
prmsl_var.long_name = 'air_pressure_at_sea_level'
prmsl_var.missing_value = -9999

uwind_var = nc.createVariable('uwind', datatype=np.float32,
                              dimensions=('time', 'ny_grid', 'nx_grid'),
                              zlib=True)
uwind_var.units = 'm/s'
uwind_var.standard_name = 'eastward_wind'
uwind_var.long_name = 'Surface Eastward Air Velocity (10m AGL)'

vwind_var = nc.createVariable('vwind', datatype=np.float32,
                              dimensions=('time', 'ny_grid', 'nx_grid'),
                              zlib=True)
vwind_var.units = 'm/s'
vwind_var.standard_name = 'northward_wind'
vwind_var.long_name = 'Surface Northward Air Velocity (10m AGL)'

stmp_var = nc.createVariable('stmp', datatype=np.float32,
                              dimensions=('time', 'ny_grid', 'nx_grid'),
                              zlib=True)
stmp_var.units = 'K'
stmp_var.standard_name = 'air_temperature'
stmp_var.long_name = 'Surface Air Temperature (2m AGL)'

spfh_var = nc.createVariable('spfh', datatype=np.float32,
                              dimensions=('time', 'ny_grid', 'nx_grid'),
                              zlib=True)
spfh_var.units = '1'
spfh_var.standard_name = 'specific_humidity'
spfh_var.long_name = 'Surface Specific Humidity (2m AGL)'

#Fill the new nc file with GFS 0.25deg data (combined in one nc file)
d  = Dataset('sflux_air.nc')
tvar = 'initial_time0_hours'
t = d[tvar][:]
t_unit = d.variables[tvar].units
print(t_unit)
tvals = num2date(t,units = t_unit)

str_t = [i.strftime('%Y%m%d %H%M') for i in tvals]
```

```
#to display dates as string
datetime_t = [datetime.strptime(i,"%Y%m%d %H%M") for i in str_t]
# print(str_t)
# print(datetime_t)
time_var[:]= t

#check if you need to rotate or flip the variables

pvar = 'PRMSL_P0_L101_GLL0'
prmsl=d[pvar][:][:][:]
print(prmsl.shape)
prmsl_var[:,:,:] = prmsl


uvar = 'UGRD_P0_L103_GLL0'
uwind = d[uvar][:][:][:]
print(uwind.shape)
uwind_var[:,:,:] = uwind

vvar = 'VGRD_P0_L103_GLL0'
vwind = d[vvar][:][:][:]
print(vwind.shape)
vwind_var[:,:,:] = vwind

sphvar = 'SPFH_P0_L103_GLL0'
sph = d[sphvar][:][:][:]
print(sph.shape)
spfh_var[:,:,:] = sph

stmpvar = 'TMP_P0_L103_GLL0'
stmp = d[stmpvar][:][:][:]
print(stmp.shape)
stmp_var[:,:,:] = stmp
d.close()
nc.close()
```

Appendix A.2 Python script to create elev2D.th.nc

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue May  4 11:42:35 2021

@author: christelle
"""
from netCDF4 import Dataset,num2date
from datetime import datetime
import numpy as np
import xarray as xr
#import pandas as pd

# get lon/lat of open boundaries nodes from hgrid.ll
f=open('hgrid_principal.ll','r')
data=f.readlines()

num_nodes_and_elements = data[1]
num_nodes = num_nodes_and_elements.split(" ")[1]
num_elements= num_nodes_and_elements.split(" ")[0]

nodes_start = 2
node_lines = data[nodes_start:nodes_start+int(num_nodes)]

elements_start = nodes_start + int(num_nodes)
element_lines = data[elements_start:elements_start + int(num_elements)]

metadata1_start = elements_start + int(num_elements)
metadata1_line = data[metadata1_start:metadata1_start + 3]
metadata_obnb=data[metadata1_start+1]
numnodes_obnb=metadata_obnb.split(" ")[0] #to have total number of nodes in open bnd
metadata_obnb1=data[metadata1_start+2]
numnodes_obnb1=metadata_obnb1.split(" ")[0]

obnodes1_start =metadata1_start+ 3
obnnode1_lines=data[obnodes1_start:obnodes1_start+int(numnodes_obnb1)]

metadata2_start = obnodes1_start + int(numnodes_obnb1)
metadata_obnb2=data[metadata2_start]
numnodes_obnb2=metadata_obnb2.split(" ")[0]

#only one open bnd in this case
# obnodes2_start =metadata2_start+ 1
# obnnode2_lines=data[obnodes2_start:obnodes2_start+int(numnodes_obnb2)]


openbnd_lines=obnnode1_lines#+obnnode2_lines #all the lines for all nodes in open bnd

#get the coordinates corresponding to the open bnd nodes
```

```
bnd_coord=[]
nodeLon=[]
nodeLat=[]
nodeLont=[]
nodeLatt=[]
for i in range(len(openbnd_lines)):
        node_index = int(openbnd_lines[i])
        node= node_lines[node_index-1]
        #node_num=node[0:5]
        node_lont=node.split(" ")[5]
        node_latt=node.split(" ")[10]
        #append lon, lat
        nodeLont.append(node_lont)
        nodeLatt.append(node_latt)

bnd_coord= np.column_stack([nodeLont, nodeLatt])

nOpenBnbNodes=int(numnodes_obnb)

##create new netcdf elev2D.th.nc
filename = 'elev2D_RES.th.nc'
nc_elev = Dataset(filename, 'w', format='NETCDF4')
#nc_elev = Dataset(filename, 'w', format='NETCDF3_CLASSIC')
nc_elev.setncatts({"Conventions": "CF-1.0"})
nc_elev.createDimension('time', None)
nc_elev.createDimension('nComponents', 1)
nc_elev.createDimension('nLevels', 1)
nc_elev.createDimension('nOpenBndNodes', len(bnd_coord)) #nOpenBnbNodes
nc_elev.createDimension('one', 1)

one=1

time_var = nc_elev.createVariable('time', np.double, ('time',))
time_var.units = 's'
time_var.long_name = 'simulation time in seconds'


timestep_var = nc_elev.createVariable('time_step', np.float32, 'one')
timestep_var.units = 's'
timestep_var.long_name = 'time step in seconds'
timestep_var[:]= 86400


elev_var = nc_elev.createVariable('time_series',np.float32,('time','nOpenBndNodes',
                              'nLevels', 'nComponents'),)
elev_var.long_name = 'elev_nontidal'
elev_var.units='m'
```

```
###############Fill the new nc files with Mercator data ( ex 1month  01/07/2018)
d  = Dataset('merc-all-201012-1bis.nc') #modified from original version with
#cdo to have time in seconds
tvar = 'time'
t = d[tvar][:]
t_unit = d.variables[tvar].units
#print(t_unit)
tvals = num2date(t,units = t_unit)
str_t = [i.strftime("%Y%m%d %H%M") for i in tvals] # to display dates as string
datetime_t = [datetime.strptime(i,"%Y%m%d %H%M") for i in str_t]
time_var[:]= t


######### need to find the values for lon/lat of openbnb =bnd_coord

nc=xr.open_dataset('merc-all-201012-1bis.nc')
subset=nc.sel(longitude=bnd_coord[:,0], latitude=bnd_coord[:,1], method='nearest')
SSH=nc.zos
# #########elev2D
elev_test=subset.zos.values
elev_test2=np.zeros([len(t),len(bnd_coord)])

for j in range(len(t)):
    for i in range(len(bnd_coord)):
      elev_test2[j,i] = elev_test[j,i,i]

#put nan values=0
elev_nonan=np.nan_to_num(elev_test2)
SE_RES=np.mean(elev_nonan)
SSH=elev_nonan-SE_RES

elev_var[:,:,:,:] = SSH



d.close()
nc_elev.close()
```

Appendix A.3 Python script to create wwmbnd.gr3

```
#Create wwmbnd.gr3 based on SELFE open bnd segments
import csv
import numpy as nu

#     Inputs:
#     hgrid.gr3,
#     gen_wwmbnd.in (see sample below)

#     Output: wwmbnd.gr3

#     python create_wwmbnd_gr3.py

#     Sample gen_wwmbnd.in
#4 !ncond, i.e. number of conditions
#1 2 2 !cond1 #seg flag ob
#2 2 1 !cond2 #seg flag land
#3 22 -1 !cond3 #seg flag island
#4 99 0 !cond4 #dummy flag other nodes

#     Input files, info about flags for different boundaries
#     conditions ordered as they appear in hgrid.gr3
#     i.e. open, land, island
#     condition #, # boundaries, flag for open boundary nodes
#     condition #, # boundaries, flag for land boundary nodes
#     condition #, # boundaries, flag for island boundary nodes
#     condition #, # boundaries, flag for other nodes (set inside code as initial condition)

f = open('gen_wwmbnd.in')
bnd_in = f.readlines()
f.close()

ncond = int(bnd_in[0][0])
print('number of node condition flags is',ncond,' these are:')
for i in range(ncond):
    ii=i+1
    x = bnd_in[ii].split(' ')
    #print(x)
    print(f'condition {x[0]}, # boundaries {x[1]}, flag {x[2]}')

#read in details from hgrid
with open('hgrid.gr3') as f:
    fr = csv.reader(f,delimiter=' ')
    next(fr)
    nenp = next(fr)
    #initialise arrays (ne = number of elements; np = number of nodes)
    #xnd = lond, ynd = lat, ibnd = node number, nm = 3 nodes numbers per element
    ne = int(nenp[0])
    np = int(nenp[1].strip())
    xnd =nu.zeros([np])
```

```
ynd =nu.zeros([np])
ibnd=nu.zeros([np], dtype='i')
nm=nu.zeros([ne,3], dtype='i')

print(ne,np,nm.shape)

#read in xnd,ynd for each node
for i in range(np):
    rec=next(fr)
    #print(rec)
    k=0
    for j in range(len(rec)):
        if rec[j] !='':
            if k == 1: xnd[i] = float(rec[j])
            if k == 2: ynd[i] = float(rec[j])
            k=k+1
print(xnd[0],ynd[0])

#read in node numbers for each element
for i in range(ne):
    rec = next(fr)
    if i==0: print(rec)
    for j in range(2,5):
        m = int(rec[j])
        nm[i,j-2] = m
        if i==0: print(rec,j,m,nm[i,j-2])

print(nm[0,:],nm.dtype)


rec = next(fr)
nope = int(rec[0])
print(nope)

#check expected number of open boundaries
x = bnd_in[1].split(' ')
nope2 = int(x[1])
if nope!=nope2:
    print('nope/=nope2')
    exit()

#ok, set up flag
ifl_ob = int(x[2])

neta = next(fr)

for k in range(nope):
    rec=next(fr)
    nond = int(rec[0])
    print(nond)
    for i in range(nond):
        rec = next(fr)
        iond = int(rec[0])
        if i == 0: print(rec,iond)
        if iond>np or iond<=0:
            print('iond>np')
            exit()
        ibnd[iond-1]=ifl_ob

#now land/island boundaries
#total number of land/island boundaries
rec = next(fr)
nlandb = int(rec[0])
#total number of land boundary nodes
rec = next(fr)
nlandn = int(rec[0])
print('number of land/island boundaries',nlandb)
print('number of land nodes',nlandn)

#check expected number of land boundaries
x2 = bnd_in[2].split(' ')
x3 = bnd_in[3].split(' ')
print(x2)
print(x3)
nlb = int(x2[1]) #number of land boundaries
nib = int(x3[1]) #number of island boundaries
nlib = nlb+nib
if nlandb!=nlib:
    print('nlandb/=nlb',nlandb,nlib)
    exit()

ifl_l = int(x2[2])
ifl_i = int(x3[2])
print('flags for land and island',ifl_l,ifl_i)

for k in range(nlandb):
    rec = next(fr)
    nond = int(rec[0])
    print(nond)
    for i in range(nond):
        rec = next(fr)
        iond = int(rec[0])
        if i == 0: print(rec,iond)
        if iond>np or iond<=0:
            print('iond>np')
            exit()
        #check if land or island boundary
        if k <= nlb-1:
            ibnd[iond-1]=ifl_l
        else:
                ibnd[iond-1]=ifl_i

#
#     Output
#
#     Output file
with open('wwmbnd.gr3','w') as w:
    ww = csv.writer(w,delimiter=' ')
    ww.writerow('')
    ww.writerow([ne,np])
    for i in range(np):
        ww.writerow([i+1,xnd[i],ynd[i],float(ibnd[i])])
    for i in range(ne):
        ww.writerow([i+1,3,nm[i,0],nm[i,1],nm[i,2]])

# check duplicated nodes, and check for the first node of open
# bnd
```

Appendix A.4 Python script to download Ifremer spectral data (ncftpget.py)

```
import subprocess
year = 2018
lats = [54.0]
lons = [2.5]
model = 'GLOB-30M'
fils = f'WW3-{model}-'
filf = '_spec.nc'

k=0
for lo in lons:
    lon = lo*10
    lon = str(int(lon)).zfill(4)
    print(lon)
    for la in lats:
        ifail=0
        lat = la*10
        lat = str(int(lat)).zfill(3)
        print(lat)
        fil = f'{fils}E{lon}S{lat}_{year}{filf}'
        print(fil)
        #ftp://ftp.ifremer.fr/ifremer/ww3/HINDCAST/GLOBMULTI_ECMWF_02/GLOB-30M/
        #2018/SPEC_SE/WW3-GLOB-30M-E0000S680_2018_spec.nc
        cmd =
        f'ncftpget ftp://ftp.ifremer.fr/ifremer/ww3/HINDCAST/GLOBMULTI_ECMWF_02/{model}/{year}/SPEC_
        print(cmd)
        ifail = subprocess.call(cmd,shell=True)
        if ifail != 0:
            print(f'{fil} not found')
        #if k==0:exit()
```