

# **SEVERN-SWOT** Documentation

generated from <https://github.com/JMMP-Group/SEVERN-SWOT/wiki>: 22.12.2022

# Contents

01. Compile NEMO and XIOS	4
02. Build domain configuration file	5
Make coordinates from parent	5
Make bathymetry file	5
Make the domain configuration file	6
03. Run unforced	7
04. Make tidal boundary conditions	8
1) Install PyNEMO on ARCHER 2:	8
2) Create a mask file	8
3) Prepare PyNEMO files	8
4) Run PyNEMO	9
5) Make sure any output is where it is supposed to end up	9
05. Run tide only	10
06. Add met forcing.	11
Add ERA5 met forcing	11
0. Download ERA5 forcing data	11
1. Extract ERA5 data	11
2. Extract the Land-Sea Mask	11
3. Create WEIGHTS for the atmospheric forcing	11
07. Run tides with met forcing	12
08. Make full boundaries	13
09. Run full met forcing and full boundaries	14
10. Create and run with initial conditions	15
Initial condition input file	15
Run	15
11. Create and run with river forcing	16
River input file	16
Run	16
12. Setting up wetting and drying	17
Download and switch to the WAD branch of the repository:	17
Create WAD compatible bathymetry:	17
Re-compile modified NEMO tools:	17
Create tidal boundaries	17
Run the model	17
My modifications to the code for the domain creation	17
My modifications to the code for the initial conditions in MY_SRC	18
Notes about setting boundaries	19
Notes on CMEMS data for open boundaries	20
Notes on generating wiki pdf	21
Obtain wikidoc code	21
Install library	21
Edit Home.md	21
Add a HTML header to each page	22
Compile pdf	22
Notes on processor decomposition	23
Processor decomposition	23
Generating a submission script (not currently used here)	23
Notes setting up PyNEMO on ARCHER2	24

Welcome to the SEVERN-SWOT wiki!

Scripts to replicate the work on ARCHER2 are in the SCRIPTS directory

- Examples of basic git commands:

```
#get master branch
git clone https://github.com/JMMP-Group/SEVERN-SWOT
#switch to another branch and check it out into your working directory
cd SEVERN-SWOT
git checkout WAD
#fetch and merge any commits from the tracking remote branch
git pull https://github.com/JMMP-Group/SEVERN-SWOT.git

#push changes to the master rep online
git add BUILD_CFG/OPEN_BOUNDARIES/NCML/CMEMS.ncml
git add BUILD_CFG/OPEN_BOUNDARIES/namelist_FES14.bdy
git add RUN_DIRECTORIES/EXP_barotropicTide/namelist_cfg
git config --global user.email "user@email.ac.uk"
git config --global user.name "user"
git commit -m "ARCHER2 updates"
git push
```

## 01. Compile NEMO and XIOS

To build the NEMO, XIOS and TOOLS executables the following scripts should be followed in order (edit first `make_paths.sh` as necessary). The scripts are in `SCRIPTS` directory.

```
cd SCRIPTS
. ./make_paths.sh
. ./make_directories.sh
. ./make_xios.sh
. ./make_nemo.sh
```

These are executed before the domain file is made so some of the tooling and directory structures can be reused.

## 02. Build domain configuration file

To create a new configuration a domain configuration file must be created that defines the grid. This only needs to be done once for a new configuration. This includes bathymetry, depth and horizontal grid discretisation.

Edit `make_paths.sh` as necessary (if not done previously), then execute

```
cd SCRIPTS
. ./make_paths.sh
. ./make_directories.sh
. ./make_tools.sh
```

### Make coordinates from parent

Start with a coordinates file from a parent NEMO simulation. In this example we use the AMM15 coordinate file (attached within the [Zenodo Version history](#)). This parent coordinates file needs to be saved in the DOWNLOADS folder and named `coordinates_AMM15.nc`. This file stores the horizontal grid information.

For NOC users only:

The file is available locally (E.g. `/projectsa/NEMO/nibrun/ARCHER_DATA/INPUTS/AMM15/coordinates.nc`). For example from livjobs\* to get the file on ARCHER2 (edit according to your ARCHER2 aliasing syntax):

```
scp /projectsa/NEMO/nibrun/ARCHER_DATA/INPUTS/AMM15/coordinates.nc $USER@login.archer2.ac.uk:/work/n01/n01/$USER/SEVERN-SWOT/DOWNLOADS/coordinates_AMM15.nc
```

The AGRIF NESTING tools are used to generate child grid horizontal coordinates from the parent grid. To control this one needs to figure out the indices of the child domain start and end, and also the resolution scale factor. This information is edited in the `BUILD_CFG/DOMAIN/namelist.input` file. For the Severn domain (`BUILD_CFG/DOMAIN/namelist.input_SEVERN`) to get a horizontal resolution of 500m from a parent with resolution of 1.5km increase resolution by factor of 3. To achieve this the namelist file is as follows:

```
&nesting
  imin = 694
  imax = 807
  jmin = 400
  jmax = 490
  rho = 3
  rhot = 3
```

Then run in `SCRIPTS`:

```
. ./make_coordinates_from_parent_Severn.sh
```

The output is a `coordinates.nc` file in `BUILD_CFG/DOMAIN`.

### Make bathymetry file

This is mostly covered in the generic `./make_bathymetry_from_gebco.sh` but this step also requires some pre-processing of a bathymetry file, downloaded from [GEBCO](#), which doesn't really automate. The region downloaded should be copied to 'DOWNLOADS' for pre processing. E.g.

```
scp gebco_2020_n51.8_s50.1_w-6.9_e-2.0.nc $USER@login.archer2.ac.uk:/work/n01/n01/$USER/SEVERN-SWOT/DOWNLOADS/.
```

Then using convenient relabeling

```
export BATHYFILE=gebco_2020_n51.8_s50.1_w-6.9_e-2.0.nc
```

Process the bathymetry in DOWNLOADS. Modify the 'elevation' variable such that water depths are positive; land points are zero

```
module load nco/5.0.5
module load libfabric/1.11.0.4.71
## Flatten the land
ncap2 -s 'where(elevation > 0) elevation=0' $BATHYFILE tmp.nc
## Make negative depths positive
ncflint --fix_rec_crd -w -1.0,0.0 tmp.nc tmp.nc gebco_in.nc
rm tmp.nc
```

Then block out areas of water that you don't want to simulate in python. 1st Make a copy for editing:

```
cp gebco_in.nc fixed_bathy.nc
```

2nd load python modules:

```
module load cray-python
module load netcdf4/1.5.7
```

3rd execute in python:

```
python
import netCDF4
import numpy as np
dset = netCDF4.Dataset('gebco_in.nc','r')
dout = netCDF4.Dataset('fixed_bathy.nc','a')

dout.variables['elevation'][0:99,:]= 0
dout.variables['elevation'][0:200,650:]= 0

dset.close()
dout.close()
exit()
```

If happy relabel and copy to `BUILD_CFG/DOMAIN` location (as required in script: `make_bathymetry_from_gebco.sh`):

```
cp fixed_bathy.nc $WDIR/BUILD_CFG/DOMAIN/gebco_in.nc
```

Then in `SCRIPTS` run:

```
. ./make_bathymetry_from_gebco.sh
```

The output is a `bathy_meter.nc` file in `BUILD_CFG/DOMAIN`. Some post-processing might be needed, typically one might want a minimum depth of 10m:

```
# load nco modules (if not done already)
module load nco/5.0.5

# Remove weirdness with negative bathymetry and make minimum bathymetry
#equal to 10 m (resolve any possible wet-drying problems)
ncap2 -s 'where(Bathymetry < 0) Bathymetry=0' bathy_meter.nc tmp1.nc
ncap2 -s 'where(Bathymetry < 10 && Bathymetry > 0) Bathymetry=10' tmp1.nc -O bathy_meter.nc
rm tmp1.nc

# Copy it if you want for safe keeping
cp bathy_meter.nc bathy_meter_10m.nc
```

Or fix bathymetry to deal with instabilities (e.g. opening some straights that have only 2 grid points), using an `nco` command like:

```
ncap2 -s 'Bathymetry(0,0)=0' bathy_meter.nc bathy_meter.nc -O
ncap2 -s 'Bathymetry(105:116,108:117)=0' bathy_meter.nc bathy_meter.nc -O
```

*NOTE: There is a creek/estuary at i=113, j=114 (river Taw) that looks too narrow to let the water easily flow in and out. This leads to a hydrological jump at time step =261, (dt=20s) starting at 1st Jan 2013. Water in the estuary has a sea level of -5m. At the mouth it is -2m. So there is an unphysical steep SSH gradient at the mouth. This seems to result in a blow up. The previous line of code closes the estuary. This might need to be revisited and follow a different approach (smoothing bathymetry)*

\_NOTE: problem with IC.

```
ncap2 -s 'Bathymetry(225:255,321:341)=0' bathy_meter.nc bathy_meter.nc -O
ncap2 -s 'Bathymetry(249:263,0:20)=0' bathy_meter.nc bathy_meter.nc -O
ncap2 -s 'Bathymetry(105:115,105:115)=0' bathy_meter.nc bathy_meter.nc -O
```

## Make the domain configuration file

Finally generate the `domain_cfg.nc` file using the `coordinates.nc`, `bathy_meter.nc`. This is done in the `tools` directory using `make_domain_cfg.sh` script in the `SCRIPTS` directory. The tools used are the `DOMAINcfg` and `REBUILD_NEMO` tools. The `DOMAINcfg` tool is submitted as a slurm job.

If making changes to the configuration, these are controlled by the `namelist_cfg` file. In this configuration, it is `$DOMAIN/"$REPO"_s-sig_DOMAINcfg_namelist_cfg`, as per `$WDIR/SCRIPTS/make_domain_cfg.sh`. Ensure this has the appropriate parameters and number of lat,lon,depth levels etc set.

For example the lateral size of the domain can be extracted from `bathy_meter.nc` using `ncdump -h bathy_meter.nc`, then `namelist` can be edited accordingly e.g.: for the SEVERN-SWOT domain

```
vi namelist_cfg
...
jpidta    =      342  ! 1st lateral dimension ( >= jpi )
jpldta    =      273  ! 2nd  "              "   ( >= jpj )
jpkdta    =       31  ! number of levels      ( >= jpk )
...
```

When ready, run:

```
cd SCRIPTS
. ./make_domain_cfg.sh

creating $DOMAIN/domain_cfg.nc
```

### 03. Run unforced

In this experiment an unforced ocean is initialised from rest. It is initialised with constant density. Any velocities that are generated are the result of model errors. These can be either i) genuine code bugs (hopefully zero), or ii) numerical errors. This can be a useful check that things are working as expected.

The model can be initialised with constant density either by prescribing it as an initial condition, or by compiling it into the code as an analytic function. Here we do the latter.

```
export CONFIG=NEMOconstTS # NEMO exec with hardwired constant T,S
```

Then compile NEMO (this was already done in Step 1 so need not be repeated). It will select and compile with a prescribed stratification given in `$NEMO/cfgs/$CONFIG/MY_SRC/usrdef_istate.F90`

First set paths if not already done `./make_paths.sh`

Go to the experiment folder `cd /RUN_DIRECTORIES/EXP_unforced_constTS`

Verify that the boundary condition in `namelist_cfg` is false

```
vi namelist_cfg
ln_bdy=.false.
```

Run the experiment from `SCRIPTS` folder

```
./run_unforced_constTS.sh
```

NB the `run_unforced_constTS.sh` script assumes you can submit jobs to the `n01-ACCORD ARCHER2` account. Edit `submit.slurm` accordingly.

Be efficient with HPC resource: see [notes on processor decomposition](#)

## 04. Make tidal boundary conditions

In this page we describe how to run a tide-only NEMO run. We make the open ocean boundary conditions using the PyNEMO tool. Tidal boundary files contains tidal harmonics, that are used by NEMO to generate tidal elevation on-the-fly. In this workflow we use the [FES2014](#) tidal products.

Even for a test simulation with frozen boundaries, you need a `coordinates.bdy.nc` file (generated here) to tell NEMO where the frozen boundaries are.

Workflow:

### 1) Install PyNEMO on ARCHER 2:

instructions [here](#)

### 2) Create a mask file

The mask variable takes values (-1 mask, 1 wet/active domain, 0 land). Need to only mask a single point around the edge since the rimwidth is considered to be part of the active domain.

PyNEMO looks for the interface between -1 and 1 to generate boundary forcings. Get a template from `domain_cfg.nc` and then modify as desired around the boundary.

Maked the entire boundary "land=0", except for the wet bits along the southern boundary which are "mask=-1":

```
cd /work/n01/n01/$USER/SEVERN-SWOT/BUILD_CFG/DOMAIN/
module load nco
rm -f bdy_mask.nc tmp[12].nc
ncks -v top_level domain_cfg_$REPO.nc tmp1.nc
ncrename -h -v top_level,mask tmp1.nc tmp2.nc
ncwa -a t tmp2.nc bdy_mask.nc
rm -f tmp[12].nc
```

Then in python edit the westward boundary value only

```
conda activate pynemo
```

python Then

```
import netCDF4
import numpy as np
dset = netCDF4.Dataset('bdy_mask.nc','a')
[ny,nx] = np.shape(dset.variables['mask'][:])
for i in range(ny):
    if dset.variables['mask'][i,1] == 1:
        dset.variables['mask'][i,0] = -1
    else:
        dset.variables['mask'][i,0] = 0

#dset.variables['mask'][248::,0:20] = 0 # Mask out rogue 'lake'.
dset.close()
exit()
```

### 3) Prepare PyNEMO files

The `namelist*.bdy` drives PyNEMO and has input files which point to the data sources `sn_src_hgr`, `sn_src_zgr`, `sn_src_msk`. These source files are the parent files for the (non-harmonic) time varying boundaries. Even if only tides are being generated these need to be specified and real because their presence is checked before running the tide bit of code.

In this case study, we use the Atlantic - European North West Shelf - Ocean Physics Analysis and Forecast data at 1.5 km resolution for the open boundaries. Variables needed and downloadable from [CMEMS AMM15](#) are currents, salinity, temperature and sea surface height. How to build the required coordinates and mask file is described [here](#).

For NOC users, in `/work/n01/n01/micdom/CMEMS_AMM15` there are `CMEMS_subdomain_coordinates.nc` and `CMEMS_subdomain_mask.nc`, copy or link them:

```
cd /work/n01/n01/$USER/SEVERN-SWOT/BUILD_CFG/OPEN_BOUNDARIES
ln -s /work/n01/n01/micdom/CMEMS_AMM15/CMEMS_subdomain_coordinates.nc CMEMS_subdomain_coordinates.nc
ln -s /work/n01/n01/micdom/CMEMS_AMM15/CMEMS_subdomain_mask.nc CMEMS_subdomain_mask.nc
ln -s ../DOMAIN/domain_cfg_SEVERN-SWOT.nc domain_cfg.nc
ln -s ../DOMAIN/bathy_meter.nc bathy_meter.nc
```

Modify `namelist_FES2014.bdy` so that it points to the correct files:

```
sn_src_hgr = './CMEMS_subdomain_coordinates.nc' ! parent /grid/
sn_src_zgr = './inputs_src_zgr.ncml' ! parent
sn_dst_hgr = './domain_cfg.nc' ! child
sn_dst_zgr = './inputs_dst.ncml' ! rename output variables
sn_src_msk = './CMEMS_subdomain_mask.nc' ! NOT NEEDED FOR TIDES ! parent
sn_bathy = './bathy_meter.nc' ! ! child
```

```
!-----
! I/O
!-----
sn_src_dir = './CMEMS.ncml' ! src_files/'
```

The `ncml` files remap some variable names in the source and destination files (edit/check the paths are correct). For NOC Users, the `CMEMS.ncml` points to data located in `/work/n01/n01/micdom/CMEMS_AMM15`, access to the folder is needed even if for a tide-only run those files are not used and the date and time specified in the `namelist` are dummy values.

To generate tidal boundaries select `ln_tide = .true.` and uncomment the constituents to be used:

```
!-----
! unstructured open boundaries
!-----
ln_coords_file = .true. ! =T : produce bdy coordinates files
cn_coords_file = 'coordinates.bdy.nc' ! name of bdy coordinates files
! (if ln_coords_file=.TRUE.)
ln_mask_file = .true. ! =T : read mask from file
cn_mask_file = '../DOMAIN/bdy_mask.nc' ! name of mask file
! (if ln_mask_file=.TRUE.)
ln_dyn2d = .false. ! boundary conditions for
! barotropic fields
ln_dyn3d = .false. ! boundary conditions for
! baroclinic velocities
ln_tra = .false. ! boundary conditions for T and S
ln_ice = .false. ! ice boundary condition
nn_rimwidth = 9 ! width of the relaxation zone

!-----

!-----
! unstructured open boundaries tidal parameters
!-----
ln_tide = .true. ! =T : produce bdy tidal conditions
sn_tide_model = 'FES2014' ! Name of tidal model (FES2014|TPX07p2)
cname(1) = 'M2' ! constituent name
cname(2) = 'S2'
cname(3) = 'N2'
cname(4) = 'O1'
cname(5) = 'K1'
cname(6) = 'K2'
cname(7) = 'L2'
cname(8) = 'NU2'
cname(9) = 'M4'
cname(10) = 'MS4'
cname(11) = 'Q1'
cname(12) = 'P1'
cname(13) = 'S1'
cname(14) = 'ZN2'
cname(15) = 'MU2'
ln_trans = .true. ! interpolate transport rather than velocities
```

For NOC users FES2014 data are stored in `/work/n01/n01/shared/jelt/FES2014/` or are obtainable from [AVISO](#).

```
-----
! Time information
!-----
nn_year_000   = 2019      ! year start
nn_year_end   = 2019      ! year end
nn_month_000  = 08        ! month start (default = 1 is years>1)
nn_month_end  = 08        ! month end (default = 12 is years>1)
sn_dst_calendar = 'gregorian' ! output calendar format
nn_base_year  = 1970      ! base year for time counter
! TPXO File locations
  sn_tide_grid = ''
  sn_tide_h    = ''
  sn_tide_u    = ''
! location Of FES data
  sn_tide_fes  = '/work/n01/n01/shared/jelt/FES2014/' ! SYMBOLIC LINK
```

#### 4) Run PyNEMO

```
conda activate pynemo
mkdir OUTPUT
pynemo -s namelist_FES14.bdy
```

This creates files in `OUTPUT`.

If the number of constituents to be created is more than two, is advisable to run PyNEMO as a job, using a slurm script like this `submitpynemo.slurm` :

```
#!/bin/bash
#SBATCH --job-name=pynemo
#SBATCH --time=00:10:00
#SBATCH --nodes=1
#SBATCH --ntasks=3
#SBATCH --account=n01-ACCORD
#SBATCH --partition=serial
#SBATCH --qos=serial

source ~/.bashrc
conda activate pynemo
cd /work/n01/n01/$USER/SEVERN-SWOT/BUILD_CFG/OPEN_BOUNDARIES/
pynemo -s namelist_FES14.bdy
~
```

to run it: `sbatch submitpynemo.slurm`

#### 5) Make sure any output is where it is supposed to end up

Rename and move files (names are not quite what will be expected `*grid_[UVT].nc`)

```
for var in M2 S2 O1 K1 K2 N2 L2 NU2 M4 MS4 Q1 P1 S1 2N2 MU2 #edit the the constituents
do
  echo $var
  mv 'OUTPUT/SEVERN_FES14_bdytide_FES2014_'$var'_grd_U.nc' 'OUTPUT/SEVERN_FES14_bdytide_'$var'_grid_U.nc'
  mv 'OUTPUT/SEVERN_FES14_bdytide_FES2014_'$var'_grd_V.nc' 'OUTPUT/SEVERN_FES14_bdytide_'$var'_grid_V.nc'
  mv 'OUTPUT/SEVERN_FES14_bdytide_FES2014_'$var'_grd_Z.nc' 'OUTPUT/SEVERN_FES14_bdytide_'$var'_grid_T.nc'
done

mkdir /work/n01/n01/$USER/SEVERN-SWOT/INPUTS/TIDES/
cp OUTPUT/SEVERN_FES14_bdy* /work/n01/n01/$USER/SEVERN-SWOT/INPUTS/TIDES/.
cp OUTPUT/coordinates.bdy.nc /work/n01/n01/$USER/SEVERN-SWOT/INPUTS/OBC/.
```

## 05. Run tide only

- Go in the `RUN_DIRECTORIES/EXP_barotropicTide` and modify start date & duration of the run and tidal constituents in `namelist_cfg`

```
nn_it000    = 1 ! first time step
nn_itend    = 8640 ! last time step
nn_date0    = 20000101
```

```
cname(1)='M2'
cname(2)='S2'
cname(3)= 'N2'
cname(4)= 'O1'
cname(5)= 'K1'
cname(6)= 'K2'
cname(7)= 'L2'
cname(8)= 'NU2'
cname(9)= 'M4'
cname(10)= 'MS4'
cname(11)= 'Q1'
cname(12)= 'P1'
```

- Go in the `SCRIPTS` folder
- First set paths if not already done `./make_paths.sh`
- then run `./run_barotropicTide.sh`

If running with wetting & drying (read [here](#) first):

- Go in the `RUN_DIRECTORIES/EXP_barotropicTide_WAD` and modify the `namelist_cfg` as above
- To run go in `SCRIPTS`

```
./run_barotropicTide_WAD.sh
```

## 06. Add met forcing.

### Add ERA5 met forcing

Prepare data for fully forced run.

#### 0. Download ERA5 forcing data

Check the [ECMWF](#) pages for the latest guidance on obtaining the ERA5 reanalysis dataset. Download data as recommended.

#### 1. Extract ERA5 data

The ERA5 data for the domain is post-processed using this script:

[https://github.com/NOC-MSM/DEV\\_nibrun/blob/master/FORCING\\_DATA/OFFICIAL\\_Generate\\_NEMO\\_Forcing\\_NEWERA.py](https://github.com/NOC-MSM/DEV_nibrun/blob/master/FORCING_DATA/OFFICIAL_Generate_NEMO_Forcing_NEWERA.py)

which is also available in `SCRIPTS/OFFICIAL_GENERATE_NEMO_Forcing_NEWERA.py`, change the data path, date range of the data and also the lat and lon coordinates of the region you want to extract. **IMPORTANT:** Make sure the extracted region is larger than your model domain size.

For NOC users, ERA5 data are stored locally on the livjobs server, in general you would need to update `path_EXTRACT` and `path_FORCING`, but `path_ERA5` should remain the same (if running on a livjobs server). However, in this example a copy is already checked out in the repo so no edits are necessary. If you do not have access to the NOC Liverpool local ERA5 archive, you will need to get these data independently.

To use the script, load anaconda:

```
module load anaconda
```

Then, there are two options:

(1) create and activate a very simple python environment with the modules needed in the script:

```
conda create -p /work/$USER/envs/swot_env python=3.9 numpy pip
conda activate /work/$USER/envs/swot_env
pip install netCDF4
pip install cython
pip install netcdftime
```

(2) alternatively, it is possible to use the PyNEMO environment, but after installing the netcdftime module

```
conda activate pynemo3
pip install netcdftime
```

If the python environment has been created previously then it only needs to be activated

```
conda activate /work/$USER/envs/swot_env OR conda activate pynemo3
```

When you have the environment ready, run the script:

```
python SCRIPTS/OFFICIAL_Generate_NEMO_Forcing_NEWERA.py
```

Depending on the size of the region and the data range of the data, this will take a while.

The files generated are:

```
ERA5_MSDMLWRF_*.nc
ERA5_MSDWSWRF_*.nc
ERA5_MTFR_*.nc
ERA5_MSR_*.nc
ERA5_MSL_*.nc
ERA5_SPH_*.nc
ERA5_T2M_*.nc
ERA5_U10_*.nc
ERA5_V10_*.nc
```

#### 2. Extract the Land-Sea Mask

You might also want to extract the land sea mask for a better representation of the coastline in NEMO. The ERA5 land/sea mask can be downloaded from the ERA5 website. For NOC users, one land/sea mask is stored under: `/projectsa/NEMO/Forcing/ERA5/era5_atmos_landseamask.nc`. The new format includes the ratio land/sea in each box as it comes from ERA5 atmospheric model - so you need to choose where to cut as not that many points have exactly 1 here (land only) and many have 0.99 etc; each user might want to think what to exactly use (and modify the threshold in `create_LSM.py`).

First, use the `ncks` command to cut out the region you want (should be same as that stated in `OFFICIAL_Generate_NEMO_Forcing_NEWERA.py`):

```
cd SEVERN-SWOT/BUILD_CFG/SURFACE_FORCING
ncks -d latitude,45.,55. -d longitude,350.,0. /projectsa/NEMO/Forcing/ERA5/era5_atmos_landseamask.nc ./my_era5_LSM.nc
```

Next, use the `create_LSM.py` script to create a mask file (`ERA5_LSM.nc`):

```
python create_LSM.py
```

Visually inspect the LSM file.

#### 3. Create WEIGHTS for the atmospheric forcing

Once the ERA5 data has been downloaded, generate weights for the atmospheric forcing on ARCHER2.

For NOC users, copy all netcdfs locally generated in `/work/$USER/SEVERN-SWOT/BUILD_CFG/SURFACE_FORCING` (livjobs) to ARCHER2

```
scp *nc $USER@login.archer2.ac.uk:/work/n01/n01/$USER/SEVERN-SWOT/BUILD_CFG/SURFACE_FORCING
```

This step requires that you have compiled successfully the NEMO-TOOLS in previous step

Update the namelists `namelist_reshape_bilin_atmos` and `namelist_reshape_bicubic_atmos` in `BUILD_CFG/SURFACE_FORCING` to reflect the year of your ERA5 data, paths, names, etc.

```
$grid inputs
input_file = './ERA5_MSL_2019.nc'
```

Run the script `SCRIPTS/create_weight_ERA5.sh` that will generate the weight files for atmospheric forcing.

The files generated are:

```
remap_data_grid_atmos.nc
data_nemo_bilin_atmos.nc
weights_era5_SEVERN_bilin.nc
data_nemo_bicubic_atmos.nc
weights_era5_SEVERN_bicubic.nc
```

Other examples of this process can be found at: [SANH Wiki page](#).

## 07. Run tides with met forcing

Run the tidal model with full ERA5 forcing (sea level pressure, 10m winds, fluxes, etc.).

- We assume here tidal boundaries and met-forcing have been generated
- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5` and modify start date & duration of the run and tidal constituents in `namelist_cfg`
- Change the path to met-forcing in `namelist_cfg`:

The `namelist_cfg` has the following flags on `.true.:`

```
ln_blk      = .true.      ! Bulk formulation              (T => fill namebc_blk )
ln_traqsr   = .true.      ! Light penetration in the ocean (T => fill namtra_qsr)
```

- To run go in `SCRIPTS`

```
. ./run_Tide_ERA5.sh
```

If running with wetting & drying:

- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5_WAD` and modify the `namelist_cfg` as above
- To run go in `SCRIPTS`

```
. ./run_Tide_ERA5_WAD.sh
```

## 08. Make full boundaries

- Follow the same instructions for the creation of the [tidal boundaries](#).
- In this case study, we use the Atlantic - European North West Shelf - Ocean Physics Analysis and Forecast data at 1.5 km resolution for the open boundaries. Variables needed and downloadable from [CMEMS AMM15](#) are currents, salinity, temperature and sea surface height. How to build the required coordinates and mask file is described [here](#). For NOC users, some data is available in `/work/n01/n01/micdom/CMEMS_AMM15`.
- Check that the parent files are meaningful: no dummy files, correct domain and time period (you need a full month).
- Change the PyNEMO namelist as follows:

```
!-----
! unstructured open boundaries
!-----
ln_coords_file = .true.          ! =T : produce bdy coordinates files
cn_coords_file = 'coordinates.bdy.nc' ! name of bdy coordinates files
                                   ! (if ln_coords_file=.TRUE.)
ln_mask_file   = .true.          ! =T : read mask from file
cn_mask_file   = '../DOMAIN/bdy_mask.nc' ! name of mask file
                                   ! (if ln_mask_file=.TRUE.)
ln_dyn2d       = .true.          ! boundary conditions for
                                   ! barotropic fields
ln_dyn3d       = .true.          ! boundary conditions for
                                   ! baroclinic velocities
ln_tra         = .true.          ! boundary conditions for T and S
ln_ice         = .false.         ! ice boundary condition
nn_rimwidth    = 9               ! width of the relaxation zone

!-----
! Time information
!-----
nn_year_000    = 2019           ! year start
nn_year_end    = 2019           ! year end
nn_month_000   = 08             ! month start (default = 1 is years>1)
nn_month_end   = 08             ! month end (default = 12 is years>1)
sn_dst_calendar = 'gregorian'    ! output calendar format
nn_base_year   = 1970           ! base year for time counter
```

- Run PyNEMO

It is advisable to run PyNEMO as a job, using a slurm script like this `submitpynemo.slurm`, check the correct namelist is used :

```
#!/bin/bash
#SBATCH --job-name=pynemo
#SBATCH --time=00:10:00
#SBATCH --nodes=1
#SBATCH --ntasks=3
#SBATCH --account=n01-ACCORD
#SBATCH --partition=serial
#SBATCH --qos=serial

source ~/.bashrc
conda activate pynemo
cd /work/n01/n01/$USER/SEVERN-SWOT/BUILD_CFG/OPEN_BOUNDARIES/
pynemo -s namelist_AMM15Severn.bdy
```

To run it: `sbatch submitpynemo.slurm`

- After producing the bdy file, copy them in the INPUTS directory:

```
cp OUTPUT/*bdy[TUV]* /work/n01/n01/$USER/SEVERN-SWOT/INPUTS/OBC/.
```

## 09. Run full met forcing and full boundaries

Run the model with full ERA5 forcing (sea level pressure, 10m winds, fluxes, etc.) and full boundaries

- We assume here the BDY boundaries and met-forcing have been generated
- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5_BDY` and modify start date & duration of the run in the `namelist_cfg`
- These are the flags activated for the bdy boundaries:

```
!-----
&nambody      ! unstructured open boundaries      (default: OFF)
!-----
ln_bdy        = .true.      ! Use unstructured open boundaries
nb_bdy        = 1          ! number of open boundary sets
!
ln_coords_file = .true.      ! =T : read bdy coordinates from file
cn_coords_file = 'coordinates.bdy.nc' ! bdy coordinates files
ln_mask_file   = .false.     ! =T : read mask from file
cn_mask_file   = ''          ! name of mask file (if ln_mask_file=.TRUE.)
cn_dyn2d       = 'flather'   !
nn_dyn2d_dta   = 3           ! = 0, bdy data are equal to the initial state
! = 1, bdy data are read in 'bdydata .nc' files
! = 2, use tidal harmonic forcing data from files
! = 3, use external data AND tidal harmonic forcing
cn_dyn3d       = 'specified' ! 'zerograd','none' !
nn_dyn3d_dta   = 1           ! = 0, bdy data are equal to the initial state
! = 1, bdy data are read in 'bdydata .nc' files
cn_tra         = 'frs'
nn_tra_dta     = 1           ! = 0, bdy data are equal to the initial state
! = 1, bdy data are read in 'bdydata .nc' files

ln_tra_dmp     = .false.     ! open boudaries conditions for tracers
ln_dyn3d_dmp   = .false.     ! open boundary condition for baroclinic velocities
rn_time_dmp    = 8.          ! Damping time scale in days
rn_time_dmp_out = 64.        ! Outflow damping time scale
nn_rimwidth    = 9           ! width of the relaxation zone
ln_vol         = .false.     ! total volume correction (see nn_volctl parameter)
nn_volctl      = 1           ! = 0, the total water flux across open boundaries is zero
/

!-----
!&nambody_ssh
!-----
ln_ssh_bdy     = .false.
rn_ssh_shift   = 0.
/

!-----
&nambody_dta   ! open boundaries - external data
!-----
ln_zinterp     = .false.     ! T if a vertical interpolation is required. Variables gdep[tuv] and e3[tuv] must exist in the file
! = 1, automatically defined to T if the number of vertical levels in bdy_dta /= jpk
ln_full_vel    = .true.      ! T if [uv]3d are "full" velocities and not only its baroclinic components
cn_dir         = './OBC/'
!
!-----
! file name      frequency (hours) variable time interp. clim yearly/ weights filename rotation land/sea mask
! (if <0 months) name (logical) (T/F) monthly pairing filename
bn_ssh          = 'SEVERN_bdyT' , 24 , 'sosshieg' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
bn_u2d          = 'SEVERN_bdyU' , 24 , 'vobtcrtx' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
bn_v2d          = 'SEVERN_bdyV' , 24 , 'vobtcrtx' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
bn_u3d          = 'SEVERN_bdyU' , 24 , 'vozocrtx' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
bn_v3d          = 'SEVERN_bdyV' , 24 , 'vomecrty' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
bn_tem          = 'SEVERN_bdyT' , 24 , 'votemper' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
bn_sal          = 'SEVERN_bdyT' , 24 , 'vosaline' , .true. , .false. , 'monthly' , ' ' , ' ' , ' ' , ' '
```

- To run go in `SCRIPTS`

```
./run_Tide_ERA5_BDY.sh
```

If running with wetting & drying:

- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5_River_IC_BDY_WAD` and modify the `namelist_cfg` as above
- To run go in `SCRIPTS`

```
./run_Tide_ERA5_River_IC_BDY_WAD.sh
```

- NOTE: [James Knowledge on setting BDY](#)

## 10. Create and run with initial conditions

### Initial condition input file

Create initial conditions with ad-hoc Matlab scripts

#### Run

- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5_BDY` and modify start date & duration of the run in the `namelist_cfg`
- Copy the initial conditions file in the directory
- Modify `namelist_cfg`

I'm using initial condition built from 1 daily CMEMS file. My file contains only 1 temporal record. NEMO expects a file with this name: `SEVERN_SWOT_IC_CMEMSAMM15_y2019m08d01.nc`

```
!-----
!namtsd      !      Temperature & Salinity Data  (init/dmp)      (default: OFF)
!-----
!      !      file name      ! frequency (hours) ! variable ! time interp.! clim ! 'yearly'/ ! weights ! rotation ! land/sea mask !
!      !      (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' ! filename ! pairing ! filename !
sn_tem  = 'SEVERN_SWOT_IC_CMEMSAMM15',      24      , 'votemper' , .false. , .false. , 'daily' , '' , '' , ''
sn_sal  = 'SEVERN_SWOT_IC_CMEMSAMM15',      24      , 'vosaline' , .false. , .false. , 'daily' , '' , '' , ''
!
cn_dir   = './'      ! root directory for the location of the runoff files
ln_tsd_init = .true.  ! Initialisation of ocean T & S with T & S input data (T) or not (F)
ln_tsd_dmp = .false.  ! damping of ocean T & S toward T & S input data (T) or not (F)
ln_tsd_interp = .false.      ! Interpolation of T & S in the verticalinput data (T) or not (F)
/
```

- in `SCRIPTS`:

```
. ./run_Tide_ERA5_BDY.sh
```

If running with wetting & drying:

- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5_River_IC_BDY_WAD` and modify the `namelist_cfg` as above
- Copy the initial conditions file in the directory
- To run go in `SCRIPTS`

```
. ./run_Tide_ERA5_River_IC_BDY_WAD.sh
```

## 11. Create and run with river forcing

### River input file

Follow the instructions to create the river forcing using the JRA-55 dataset [here](#)

Note: I have manually hacked the position of the river mouth in the river runoff output file, because it was in the wrong position (it was in the middle of the domain).

- Copy the river file in the directory `/work/n01/n01/$USER/SEVERN-SWOT/BUILD_CFG/SURFACE_FORCING/`

### Run

If running with wetting & drying:

- Go in the `RUN_DIRECTORIES/EXP_Tide_ERA5_River_IC_BDY_WAD`
- Modify `namelist_cfg`

```
ln_rnf      = .true.      ! runoffs                      (T => fill namsbc_rnf)
```

and

```
!-----
&namsbc_rnf      ! runoffs                                (ln_rnf =T)
!-----
ln_rnf_mouth     = .false.  ! specific treatment at rivers mouths
rn_hrnf          = 15.e0    ! depth over which enhanced vertical mixing is used (ln_rnf_mouth=T)
rn_avt_rnf       = 1.e-3    ! value of the additional vertical mixing coef. [m2/s] (ln_rnf_mouth=T)
ln_rnf_depth     = .false.  ! read in depth information for runoff
ln_rnf_tem       = .false.  ! read in temperature information for runoff
ln_rnf_sal       = .false.  ! read in salinity information for runoff
rn_rfact         = 1.e0     ! multiplicative factor for runoff
cn_dir           = '/work/n01/n01/micdom/SEVERN-SWOT/BUILD_CFG/SURFACE_FORCING/' ! root directory for the flux data location

!-----
! file name      ! frequency (hours) ! variable ! time interp. ! clim ! 'yearly' / ! weights filename ! rotation ! land/sea mask !
! (if <0 months) ! name ! (logical) ! (T/F) ! 'monthly' ! pairing ! filename !
sn_rnf           = 'runoff'      ,      24      , 'rorunoff' , .true. , .false. , 'yearly' , '' , ''
sn_cnf           = 'runoff'      ,      0       , 'socoeffr0', .false. , .true. , 'yearly' , '' , ''
sn_s_rnf         = 'runoff'      ,      24      , 'rosaline' , .false. , .true. , 'yearly' , '' , ''
sn_t_rnf         = 'runoff'      ,      24      , 'rotemper' , .false. , .true. , 'yearly' , '' , ''
sn_dep_rnf       = 'runoff'      ,      0       , 'rodepth'  , .false. , .true. , 'yearly' , '' , ''

!-----
```

- To run go in `SCRIPTS`

```
. ./run_Tide_ERA5_River_IC_BDY_WAD.sh
```

## 12. Setting up wetting and drying

### Download and switch to the WAD branch of the repository:

```
git clone https://github.com/JMMP-Group/SEVERN-SWOT
cd SEVERN-SWOT
git checkout WAD
```

Follow the steps from [01. Compile NEMO and XIOS](#) to [02. Build domain configuration file](#) in this Wiki to create coordinates and bathymetry files.

### Create WAD compatible bathymetry:

If you wish to use the GEBCO bathymetry previously created just for testing purposes, the land values need to be changed to -9999.99

```
In $DOMAIN: ncap2 -s 'where(Bathymetry == 0) Bathymetry=-9999.99' bathy_meter.nc tmp1.nc
```

```
mv tmp1.nc bathy_meter.nc
```

The GEBCO bathymetry would require further processing, because the model is unstable). However, for wetting & drying and to get a better representation of intertidal areas we recommend building the bathymetry using the [FMODnet](#) dataset.

### Re-compile modified NEMO tools:

From the repository ([https://github.com/JMMP-Group/CO\\_AMM15](https://github.com/JMMP-Group/CO_AMM15)) I have copied the folder BUILD\_CFG/domwad with the modified .f90 tools routine. It is saved in the \$DOMAIN folder (NOTE: I have modified the routines because initially they were not compiling, my changes are included in the WAD branch and listed at the bottom of this page for clarity). I have modified the make\_tools.sh to add a copy of the modified routines before compiling the tools, in SCRIPTS run:

```
./make_tools_wad.sh
```

Then move into \$DOMAIN, I have modified the namelist\_cfg to include the changes specified here: [https://github.com/endaodea/DOMAINCFG\\_BASED\\_ON\\_36\\_NEMO/wiki](https://github.com/endaodea/DOMAINCFG_BASED_ON_36_NEMO/wiki), check the changes are there:

```
rn_sbot_min = 1.0e-5      ! minimum depth of s-bottom surface (>0) (m)
rn_sbot_max = 10000.0     ! maximum depth of s-bottom surface (= ocean depth) (>0) (m)
rn_hc       = 51.0        ! critical depth for transition to stretched coordinates
rn_rmax      = 0.1         ! maximum cut-off r-value allowed (0<r_max<1)

!-----
&namdom      ! space and time domain (bathymetry, mesh, timestep)
!-----
ldbletanh=.false.,
nn_bathy=1,
nn_msh=1,
ppa0=999999.0,
ppa1=999999.0,
ppa2=999999.0,
ppacr=9.0,
ppacr2=999999.0,
ppdzmin=6.0,
ppe1_deg=999999.0,
ppe1_m=999999.0,
ppe2_deg=999999.0,
ppe2_m=999999.0,
ppglam0=999999.0,
ppgphi0=999999.0,
pphmax=5720.0,
ppkth=23.563,
ppkth2=999999.0,
ppsurr=999999.0,
rn_bathy=100.0,
rn_rdt=60.0,
rn_wd_ref_depth=20.0,
```

In SCRIPTS run:

```
./make_domain_cfg.sh
```

Two new variables are added: rn\_wd\_ref\_depth and ht\_wd. The former is set in the namelist file in \$DOMAIN.

### Create tidal boundaries

Follow the steps in [04. Make tidal boundary conditions](#) in this Wiki

### Run the model

In SCRIPTS run:

```
./run_barotropicTide_WAD.sh
```

In the namelist I'm using the following parameters:

```
!-----
&namwad      ! Wetting and drying (default F)
!-----
ln_wd_il     = .false.    ! T/F activation of iterative limiter
ln_wd_d1     = .true.     ! T/F activation of directional limiter
ln_wd_d1_bc  = .true.     ! T/F Directional limiter Baroclinic option
ln_wd_d1_rmp = .true.     ! T/F Turn on directional limiter ramp
rn_wdmin0    = 0.3        ! depth at which Wad starts
rn_wdmin1    = 0.2        ! Minimum wet depth on dried cells
rn_wdmin2    = 0.0001     ! Tolerance of min wet depth on dried cells
rn_wdld      = 2.5        ! Land elevation below which Wad is allowed
nn_wdit      = 20         ! Max iterations for Wad limiter
rn_wd_sbcdp  = 5.0        ! Depth at which to taper sbc fluxes
rn_wd_sbcdp  = 0.999     ! Fraction of SBC fluxes at taper depth (Must be <1)
```

### My modifications to the code for the domain creation

These modifications are already included in the WAD branch

In domain.F90 I commented this 2 lines that were giving problems:

```
IF( ln_sco .OR. ln_mes ) THEN      ! s-coordinate: store grid stiffness ratio (Not required anyway)
CALL dom_stiff( z2d )`
CALL iom_rstput( 0, 0, inum, 'stiffness', z2d )      ! Max. grid stiffness ratio
! CALL dom_stiff_3D( z3d )
CALL iom_rstput( 0, 0, inum, 'stiff3D', z3d )      ! 3D stiffness ratio
! CALL saw_tooth( z2d )
CALL iom_rstput( 0, 0, inum, 'saw_tooth', z2d )      ! Saw_tooth diagnostic
ENDIF
```

I was not managing to compile with the given domzgr.F90, so what I did is adding in mine the rimwidth modification bit...

```
! CEOD
! Add on some reference level to make negative bathy positive to create
! sensible e3t_0's
!CEOD MAKE points on the perimeter assuming a boundary is here on shallower than
!10m so that tide can be specified here.
if(l==1) then !CEOD BDY CASE
DO jj = 1, jpi
DO ji = 1, jpi
!CEODIF( mig(ji) <= 2 .or. mjg(jj) <= 2 .or. mig(ji) >= jpiglo-2 .or. mjg(jj) >= jpiglo-2) then
IF( mig(ji) <= 9 .or. mjg(jj) <= 9 .or. mjg(jj) >= jpiglo-9) then
IF( bathy(ji,jj) > 0.0) then
```

```

        bathy(ji,jj) = max(10.0,bathy(ji,jj)) !
    ENDIF
ENDIF
END DO
END DO
endif !CEOD END BDY CAE
bathy(:, :) = bathy(:, :) + rn_wd_ref_depth ! Arbitrary reference level, This needs
! to be accounted for in istate in the actual model and bdys etc.
!CEODTEMP      where (bathy (:,:) .lt. 0.5) bathy(:, :) = 0.0 !added clearance for min depth values 0.5 never going to need more than 50cm
where (bathy (:,:) .lt. 0.5) bathy(:, :) = 0.0 !added clearance for min depth values 0.5 never going to need more than 50cm
!CEOD FOR EXPLICIT10M      where (bathy (:,:) .lt. 0.0) bathy(:, :) = 0.0 !added clearance for min depth values 0.5 never going to need more than 50cm
!First step excludes crazy points
!      where (bathy (:,:) .lt. 20.0 .and. bathy(:, :) > 0.0) bathy(:, :) = 20.0 !added clearance for min depth values 0.5 never going to need more than 50cm
!2nd step set to 10 any points btwee 0,-10

```

#### My modifications to the code for the initial conditions in MY\_SRC

These modifications are already included in the WAD branch.

The comment *!to be accounted for in istate in the actual model and bdys etc.* made me thinking a piece of code for the initial conditions was missing and I've copied it `usrdef_istate.F90` from Jason's code in MY\_SRC ([https://github.com/NOC-MSM/GCOMS1k/blob/master/MY\\_SRC\\_v4.0.4/usrdef\\_istate.F90](https://github.com/NOC-MSM/GCOMS1k/blob/master/MY_SRC_v4.0.4/usrdef_istate.F90))

## Notes about setting boundaries

You can use `ln_tra_dmp` and/or `ln_dyn3d_dmp` which are adjusted using `rn_time_dmp*` .

You probably won't need `ln_tra_dmp` as you're using `cn_tra ='frs'`.

If you use `cn_tra = 'specified'` then you may need `ln_tra_dmp`.

As you're using `cn_dyn3d = 'specified'` you may get issues on the boundary if the flows aren't completely balanced – which is why I use `cn_dyn3d = 'zerograd'` if I run into problems (but you could always try using `ln_dyn3d_dmp`).

With 'specified' you're clamping all rim width points equal to the values in the BDY files. If, when you run, the model crashes because you're getting fill values from the BDY files, you can turn `ln_zinterp = .true..` This will activate the BDY on-the-fly vertical interpolation and will remove `fill_values` from the arrays

## Notes on CMEMS data for open boundaries

In this case study, we use the Atlantic - European North West Shelf - Ocean Physics Analysis and Forecast data at 1.5 km resolution for the open boundaries. Variables needed and downloadable from [CMEMS AMM15](#) are currents, salinity, temperature and sea surface height.

Using cdo we post-process the CMEMS raw data, for example:

```
module load cdo

bdays=27
days=25

fileSSHgh='metoffice_foaml_amm15_NWS_SSH_b202111'$bdays'_hi202111'$days'.nc'
fileSSH='metoffice_foaml_amm15_NWS_SSH_b202111'$bdays'_dm202111'$days'.nc'

cdo timmean $fileSSHgh $fileSSH

fileTEM='metoffice_foaml_amm15_NWS_TEM_b202111'$bdays'_dm202111'$days'.nc'
fileSAL='metoffice_foaml_amm15_NWS_SAL_b202111'$bdays'_dm202111'$days'.nc'
fileOUT='CMEMS_2021_11_'$days'_download.nc'

cdo merge $fileTEM $fileSAL $fileSSH $fileOUT

fileCUR='metoffice_foaml_amm15_NWS_CUR_b202111'$bdays'_dm202111'$days'.nc'
fileOUTCUR='CMEMS_2021_11_'$days'_download_UV.nc'

cp $fileCUR $fileOUTCUR
```

Then using python we generate the coordinates and mask files:

```
module load nco/5.0.5
ncks -v lon,lat,depth,time CMEMS_2021_11_25_download.nc tmp1.nc
ncrename -d time,t -d lat,y -d lon,x tmp1.nc
ncap2 -O -s 'glamt[t,y,x]=lon' tmp1.nc tmp1.nc
ncap2 -O -s 'glamu[t,y,x]=lon' tmp1.nc tmp1.nc
ncap2 -O -s 'glamv[t,y,x]=lon' tmp1.nc tmp1.nc
ncap2 -O -s 'gphit[t,y,x]=lat' tmp1.nc tmp1.nc
ncap2 -O -s 'gphiu[t,y,x]=lat' tmp1.nc tmp1.nc
ncap2 -O -s 'gphiv[t,y,x]=lat' tmp1.nc tmp1.nc
mv tmp1.nc CMEMS_subdomain_coordinates.nc

#create masks that corresponds to the new field
ncks -v lon,lat,depth,time,so CMEMS_2021_11_25_download.nc tmp2.nc
ncks -A -v uo,vo CMEMS_2021_11_25_download_UV.nc tmp2.nc
ncatted -a _FillValue,d,, tmp2.nc
ncap2 -O -s 'where(so>0.) so=1' tmp2.nc tmp2.nc
ncap2 -O -s 'where(so<=0.) so=0' tmp2.nc tmp2.nc
ncap2 -O -s 'where(uo<=-10.) uo=1' tmp2.nc tmp2.nc
ncap2 -O -s 'where(uo<=-10.) uo=0' tmp2.nc tmp2.nc
ncap2 -O -s 'where(vo<=-10.) vo=1' tmp2.nc tmp2.nc
ncap2 -O -s 'where(vo<=-10.) vo=0' tmp2.nc tmp2.nc
ncrename -d time,t -d lat,y -d lon,x tmp2.nc
ncrename -v so,tmask tmp2.nc
ncrename -v uo,umask tmp2.nc
ncrename -v vo,vmask tmp2.nc
mv tmp2.nc CMEMS_subdomain_mask.nc
```

# Notes on generating wiki pdf

## Obtain wikidoc code

git clone <https://github.com/jpolton/wikidoc>

A handy utility to convert GitHub wiki docs into a PDF. Requires "invisible" HTML tags edits to wiki files. Updated for python3.

## Install library

I had to install [wkhtmltopdf](#)

## Edit Home.md

Edit Home.md to include HTML header information (as in the [wikidoc README](#))

In particular edit:

- line 2: which include the pdf filename
- line 46: which include the pdf cover title.

E.g.

```
<!-- WIKIDOC CONFIG
--filename SEVERN-SWOT_Apr22.pdf
--page-size A4
--margin-top 2cm
--margin-left 2cm
--margin-bottom 2cm
--margin-right 2cm
--footer-font-name Verdana
--footer-font-size 6
--footer-spacing 10
--footer-right [page]
WIKIDOC CONFIG -->

<!-- WIKIDOC HTMLHEAD
<html>
<head>
<STYLE type='text/css'>
    html { font-family: Verdana, Geneva, sans-serif; font-size: 13px; }
    .covertitle { padding-top: 40%; text-align: right; font-size: 40px; }
    .generated { font-size: 12px; }

    table { border-collapse: collapse; margin: 1em auto 1em auto; width: 90%; border: 1px solid #ccc; }
    tr td:first-child { border-right: 1px solid #ccc; width: 2.5cm !important; }
    table tr:first-child { background-color: #ddd; }
    table td { font-family: Verdana, Geneva, sans-serif; font-size: 8pt; vertical-align: top; padding: 5px; }

    h1 { page-break-before: always; font-size: 26.6px; }
    h2 { margin-top: 3ex; font-size: 20px; }
    h3 { margin-top: 3ex; }

    .breakbefore { page-break-before: always; }
    .wiki_only { display: none; }
</STYLE>
</head>
<body>
WIKIDOC HTMLHEAD -->

<!-- WIKIDOC HTMLFOOT
</body>
</html>
WIKIDOC HTMLFOOT -->

<!-- the following WIKIDOC comments are optional -->

<!-- WIKIDOC COVER
<div class='covertitle'><b>SEVERN-SWOT</b> Documentation<br><span class='generated'>generated from https://github.com/JMMP-Group/SEVERN-SWOT/wiki: 22.12.2022</span>
WIKIDOC COVER -->

<!-- WIKIDOC TOCXSL
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:outline="http://wkhtmltopdf.org/outline"
    xmlns="http://www.w3.org/1999/xhtml">
<xsl:output doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    indent="yes" />
<xsl:template match="outline:outline">
<html>
<head>
<title>Contents</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style>
    h1 {
        text-align: left;
        font-size: 26.6px;
        font-family: verdana;
    }
    div {border-bottom: 1px dashed rgb(200,200,200);}
    span {float: right;}
    li {list-style: none;}
    ul {
        font-size: 13px;
        font-family: verdana;
    }
    ul ul {font-size: 85%; }
    ul {padding-left: 0em; }
    ul ul {padding-left: 1em;}
    a {text-decoration:none; color: black;}
    ul.toplevel li { margin-bottom: 1em; }
    ul.sublevels li { margin-bottom: 0em; }
    ul li:last-child { margin-bottom: 1em; }
</style>
</head>
<body>
<h1>Contents</h1>
<ul class="toplevel"><xsl:apply-templates select="outline:item/outline:item"/></ul>
</body>
</html>
</xsl:template>
<xsl:template match="outline:item">
<li>
<xsl:if test="( @title!='') and ( @title!='Contents') ">
<div>
<a>
<xsl:if test="@link">
<xsl:attribute name="href"><xsl:value-of select="@link"/></xsl:attribute>
</xsl:if>
<xsl:if test="@backLink">
<xsl:attribute name="name"><xsl:value-of select="@backLink"/></xsl:attribute>
</xsl:if>
<xsl:value-of select="@title" />

```

```

        </a>
        <span> <xsl:value-of select="@page" /> </span>
      </div>
    </xsl:if>
    <ul class="sublevels">
      <xsl:comment>added to prevent self-closing tags in QtXmlPatterns</xsl:comment>
      <xsl:apply-templates select="outline:item"/>
    </ul>
  </li>
</xsl:template>
</xsl:stylesheet>
WIKIDOC TOCXSL -->

<!-- end of wikidoc config section -->

```

### Add a HTML header to each page

Edit each wiki page, adding

```

<!-- WIKIDOC PDFONLY
<h1>Notes on generating wiki pdf</h1>
WIKIDOC PDFONLY -->

```

to the top of each page so that the file name is included as the page header

### Compile pdf

In a python environment execute the following. (I used python=3.8 but think it was an otherwise vanilla conda environment. E.g. `conda create --name wikidoc_env python=3.8`. Then activate `wikidoc_env`). Then from within the `wikidoc` repo directory:

```
python wikidoc.py <path-to-wkhtmltopdf> <path-to-wiki-repo>
```

E.g.

```
python wikidoc.py /usr/local/bin/wkhtmltopdf ~/GitHub/SEVERN-SWOT.wiki
```

The PDF is generated in the repo directory.

# Notes on processor decomposition

## Processor decomposition

First time through don't specify the processor decomposition. In `namelist_cfg`:

```
!-----  
$nammpp      !    Massively Parallel Processing              ("key_mpp_mpi")  
!-----  
...          ! if T: the largest number of cores tested is defined by max(mppsize, jpnj*jpnj)  
  ln_nnogather = .true. ! activate code to avoid mpi_allgather use at the northfold  
  jpnj         = -1     ! jpnj  number of processors following j (set automatically if < 1)  
  jpnj         = -1     ! jpnj  number of processors following j (set automatically if < 1)
```

After running inspect ocean.output to see the recommended processor decomposition and update accordingly. For example (as in the repository):

```
!-----  
$nammpp      !    Massively Parallel Processing              ("key_mpp_mpi")  
!-----  
...          ! if T: the largest number of cores tested is defined by max(mppsize, jpnj*jpnj)  
  ln_nnogather = .true. ! activate code to avoid mpi_allgather use at the northfold  
  jpnj         = 38     ! jpnj  number of processors following j (set automatically if < 1)  
  jpnj         = 55     ! jpnj  number of processors following j (set automatically if < 1)
```

## Generating a submission script (not currently used here)

See <https://docs.archer2.ac.uk/research-software/nemo/nemo> for guidance on running NEMO on ARCHER2.

For example: 1 xios (S)erver, (s)pacing of 16, 1 server-per-node (m), 2 ocean Cores, with a spacing of 2 (c)

```
mkslurm -S 1 -s 16 -m 1 -C 2 -c 2 -t 00:10:00 -a n01-ACCORD -j nemo_test
```

Probably the thing to play with on this small job is the ocean core spacing 'c' and xios server spacing 's'.

## Notes setting up PyNEMO on ARCHER2

Install miniconda

```
cd /work/n01/n01/$USER
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod u+x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

Install in /work/n01/n01/\$USER/miniconda3 when prompted.

Allow it to modify PYTHONPATH in .bashrc

Close and open shell (do it!)

---

Install PyNEMO (to be skipped - IMMERSE branch is not updated, go to the next section and install the master branch)

```
cd /work/n01/n01/$USER
git clone https://github.com/NOC-MSM/PyNEMO.git
```

```
cd PyNEMO
git checkout IMMERSE
conda env create -f pynemo_37.yml
conda activate pynemo3
```

```
# Build and install pynemo
#export JAVA_HOME=/usr/lib64/jvm/java # Already set
```

```
python setup.py build
export PYTHONPATH=/work/n01/n01/$USER/miniconda3/envs/pynemo3/lib/python3.7/site-packages:$PYTHONPATH
python setup.py install
```

Verify it works: pynemo -h

---

Install PyNEMO (master branch)

```
cd /work/n01/n01/$USER
git clone https://github.com/NOC-MSM/PyNEMO.git
```

```
cd PyNEMO
git checkout master
conda env create -f pynemo_39.yml
conda activate pynemo
```

```
python setup.py build
export PYTHONPATH=/work/n01/n01/$USER/miniconda3/envs/pynemo/lib/python3.9/site-packages:$PYTHONPATH
python setup.py install
```

Verify it works: pynemo -h