Quick start guide
running POP
On Snellius

Michael Kliphuis

1. Introduction

This informal document serves as a quick-start guide for running the Parallel Ocean Program (POP) model on the Dutch national supercomputer Snellius.

The Parallel Ocean Program (POP) is a three-dimensional ocean circulation model primarily designed for studying the ocean climate system. The model has been developed and supported by researchers at Los Alamos National Laboratory (LANL).

The version of the model used in this guide is pop2.1alpha_jan2005 and we focus on the low-resolution (1°) grid, which features a horizontal domain of 320 x 384 grid points (longitude x latitude) and 40 depth levels.

Section 2 describes how to install the model on Snellius.

In section 3 we show how to start the quasi-equilibrium hysteresis experiment that we did at IMAU in nov/dec 2024. This experiment begins from a statistical equilibrium solution of a present-day control simulation. A quasi-equilibrium approach is applied by introducing a slowly varying extra surface freshwater flux (F_H) in the North Atlantic, between latitudes 20°N and 50°N. This additional flux is compensated across the rest of the domain (excluding marginal seas). The surface freshwater flux forcing is then linearly increased at a rate of 3×10^{-4} Sv year^{-1}. In the experiment we ran it until model year 1500 where it reached a maximum of F_H = 0.45 Sv.

Section 4 provides instructions for checking the model output and section 5 for how to calculate the timeseries of the Atlantic Meridional Overturning Circulation (AMOC) at 26°N.

Finally in section 6 we show how to set up and start a new case.

2. Installation

If you are using the 'dijkbio' login on Snellius then you can skip this step.
The POP model is already installed there else ..

1. Go to: https://webspace.science.uu.nl/~kliph103/Projects/pop/download

   And click on pop.tar. This will download the tar file on your local machine.

2. Copy the pop.tar file to your login on Snellius

   Suppose your login is 'jansen'. Then use the secure copy command scp as follows:

   ```
   scp pop.tar  jansen@snellius.surf.nl:/home/jansen/
   ```

3. Log in on Snellius with your login (e.g. jansen) via:

   ```
   ssh –X jansen@snellius.surf.nl
   ```

4. After logging in create a directory models by typing:

   ```
   mkdir models
   ```

5. Move the pop.tar file to models

   ```
   mv pop.tar models/
   ```

6. untar the tar file

   ```
   tar xvf pop.tar
   ```

Now all the needed files are on Snellius.

In the next section we start a quasi-equilibrium hysteresis experiment.

3. Start the quasi-equilibrium hysteresis experiment

1. Log in on Snellius

2. Go to the so called 'scripts' directory where you will build and start the run

   ```
   cd ~/models/pop/scripts/gx1v6/pop.B2000.gx1v6.qe_hosing.001
   ```

   If you are using the dijkbio login you can also simply type:

   ```
   scr
   ```

   this is a shortcut (see lines with aliases in ~/.bashrc)

3. In order to do the quasi-equilibrium hysteresis experiment you need to make sure that the following files are in the current directory:

   ```
   -rwxr-x--- 1 dijkbio dijkbio 110930 Apr  6  2024 forcing_sfwf.F90
   -rwxr-x--- 1 dijkbio dijkbio  44863 Apr  6  2024 forcing_tools.F90
   -rwxr-x--- 1 dijkbio dijkbio  65849 Nov 21 01:30 state_mod.F90
   ```

   Fortran files that are put in the 'scripts' directory overrule the default source code located in: ~/models/pop/code/source

   The dates of the first two files should be April 6, 2024, and the date of the last one should be November 21, 2024. You can check the revision history in the header of each file to see what has been modified for this 'hosing experiment.'

   There are two more fortran files in the directory: domain_size.F90 and POP_DomainSizeMod.F90. Somehow this is default, please don't mind them.

4. Then build the executable ./pop by typing:

   ```
   ./case.build.sc
   ```

   Afterwards check that there is indeed an executable ./pop in the current directory.

5. Next if needed modify the file pop_in. The pop_in file is the primary namelist input file for configuring your POP simulation. The file contains parameters, options, and initial settings that control various aspects of the POP simulation, such as grid configuration, physical processes, numerical methods, and output settings.

For instance via parameters `stop_option` and `stop_count` you can control the length of the simulation and there are parameters like `bckgrnd_vdc1` and `bckgrnd_vdc2` that control the background vertical diffusivity for tracers temperature and salinity.

The appendix at the end of this document provides a description of all the parameters in the pop_in namelist.

6. If needed you can output different monthly and daily variables by modifying files: `movie_contents` resp. `tavg_contents`.

   File `transport_contents` contains the straits and gateways through which mass, heat and salt transports are calculated.

7. Open file `pointer.restart` and make sure that you start from the correct restart file

   At first start, this should be the restartfile of year 2050 of the present day control simulation i.e.

   `~/models/pop/inputdata/gx1v6/restart/r.x1_SAMOC_flux.20500101`

   After each model year a restart file will be generated and the pointer.restart file will automatically be updated. When you restart the model then the run will continue from this latest restart file.

8. Finally start the run with the job script `pop.slurm`

   Make sure that you reserve enough wallclock time for the job with line:

   `#SBATCH --time=120:00:00`

   With the setting above it is set to 120 hours which is 5 wallclock days and this is the maximum on Snellius. When the job finishes before the 120 hours it will simply stop
   and the amount of time spend on the job will be taken of your budget, not the 120 hours.

   You start the job by typing on the commandline:

   `sbatch pop.slurm`

   The job is then submitted to the queue, you can check if it is running by typing:

   `squeue`

   you will see something like:

```
JOBID     PARTITION   NAME    USER     ST   TIME  NODES NODELIST(REASON)
 8619141  genoa       LR_hyst dijkbio  R    0:05  4     tcn[607,625,633,645]
```

When it runs ST which stands for status has value R. When the job is waiting/pending in the queue it has value PD.

If you want to kill the job then type:

```
scancel 8619141
```

4. Check the output of the model

The output of the simulation will end up in the directories that were set by the pop_in namelist parameters:

```
restart_outfile
tavg_outfile
movie_outfile
```

For the simulation that we described in section 3 at the dijkbio login this is directory:

```
/projects/0/prjs1105/pop/gx1v6/pop.B2000.gx1v6.qe_hosing.001
```

On the dijkbio login you can quickly go to this directory by simply typing:

```
out
```

this is a shortcut (see lines with aliases in ~/.bashrc)

You can do a quick check on the output data with the application 'ncview'

Keep in mind that on Snellius, if you type:

```
module spider ncview
```

You can check the versions of the application ncview.

Doing this on Snellius will show you that in order to use the last version you need to type:

```
module load 2023 ncview/2.1.8-gompi-2023a
```

After this you can make use of ncview

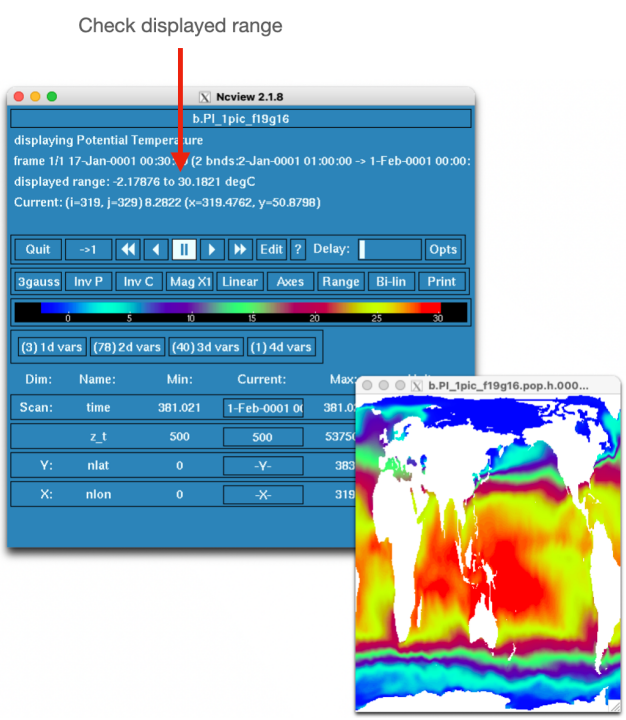
Check for instance the first outputfile of the simulation in section 3 by typing:

```
ncview t.x1_SAMOC_flux.000101.nc
```

Check for instance the 3d variables TEMP and SALT which stand for the temperature and salinity of the ocean.

When you select the TEMP variable you will see the temperature values at the surface of the ocean (SST) and they should be somewhere in a range between say -1.8° C and 42° C

When you select the SALT variable you will see the salinity values at the surface (SSS) and they should be in the range between say 2 g/kg and 40 g/kg



5. Calculate a timeseries of the AMOC

The installation contains the python script `AMOC_transport.py` that calculates a timeseries of the Atlantic Meridional Overturning Circulation (AMOC) at 26°N.

It is located in directory:

```
~/models/pop/code/tools/calc_amoc/gx1/pop.B2000.gx1v6.qe_hosing.001
```

and It calculates the AMOC timeseries for the simulation described in section 3.

On the dijkbio login you can quickly go to this directory by simply typing:

```
amoc
```

this is a shortcut (see lines with aliases in ~/.bashrc) which also activates a conda environment that contains the python libraries for matplotlib and netCDF4

Then do the actual calculation by typing:

```
python AMOC_transport.py
```

6. Set up and start a new case

Suppose you want to run a new simulation and you want to call it testrun

1. Log in on Snellius
2. Go to the base of the 'scripts' directory by typing:

   ```
   cd ~/models/pop/scripts
   ```

3. Simply copy the simulation from section 3 to testrun by typing:

   ```
   cp –r pop.B2000.gx1v6.qe_hosing.001 testrun
   ```

4. Go to the new directory and remove not needed files

   ```
   cd testrun
   rm slurm* tran.* diag.* build.log*
   ```

5. Modify all the appearances of string pop.B2000.gx1v6.qe_hosing.001 into string testrun.

   You can check this by typing:

   ```
   grep pop.B2000.gx1v6.qe_hosing.001 *
   ```

   This way you find out that you need to change

In file `case_build.sc`:

```
set case = pop.B2000.gx1v6.qe_hosing.001
```

into:

```
set case = testrun
```

Also in file `pop_in` change:

```
runid            = 'pop.B2000.gx1v6.qe_hosing.001'

restart_outfile  = '/projects/0/prjs1105/pop/gx1v6/pop.B2000.gx1v6.qe_hosing.001/restart/r'
tavg_outfile     = '/projects/0/prjs1105/pop/gx1v6/pop.B2000.gx1v6.qe_hosing.001/tavg/t'
movie_outfile    = '/projects/0/prjs1105/pop/gx1v6/pop.B2000.gx1v6.qe_hosing.001/movie/m'
var_viscosity_outfile = '/projects/0/prjs1105/pop/gx1v6/pop.B2000.gx1v6.qe_hosing.001/var_visc'
```

into:

```
runid            = 'testrun'

restart_outfile  ='/projects/0/prjs1105/pop/gx1v6/testrun/restart/r'
tavg_outfile     ='/projects/0/prjs1105/pop/gx1v6/testrun/tavg/t'
movie_outfile    ='/projects/0/prjs1105/pop/gx1v6/testrun/movie/m'
var_viscosity_outfile='/projects/0/prjs1105/pop/gx1v6/testrun/var_visc'
```

and finally in file `pop.slurm` change:

```
cd ~/models/pop/scripts/gx1v6/pop.B2000.gx1v6.qe_hosing.001
```

into:

```
cd ~/models/pop/scripts/gx1v6/testrun
```

6. From here continue from step 3. in section 3.

   Note that if you want to do a default POP run without the additional 'IMAU hosing' then remove the files `forcing_sfwf.F90` and `forcing_tools.F90` in the `~/models/pop/scripts/gx1v6/testrun` directory but do not remove the file `state_mod.F90`.

   Also remove the subnamelist `forcing_imau_nml` from the namelist file `pop_in`

APPENDIX

Description of the parameters in the pop_in namelist file (in red)

```
&domain_nml
 nprocs_clinic = 3840              < - number of cores to be used for the code, domain is divided over the cores
 nprocs_tropic = 3840              < - number of cores to be used for barotropic solver
 clinic_distribution_type = 'cartesian'    < - method for distributing blocks across processors
 tropic_distribution_type = 'cartesian'   <- method for distributing blocks across processors
 ew_boundary_type = 'cyclic'               < -type of boundary in the logical east-west direction for global domain
 ns_boundary_type = 'tripole'              < -type of boundary in the logical north-south direction for global domain
/

&context_nml
/

&io_nml
 num_iotasks = 1                   < - number of I/O processes for parallel binary I/O
 lredirect_stdout = .false.        < - flag to write stdout to log file
 log_filename = 'pop.out'          < - root filename (with path) of optional output log file
 luse_pointer_files = .true.       < - flag to turn on use of pointer files
 pointer_filename = 'pointer'
/

&time_manager_nml
 runid        = 'clim_closeditf'
 stop_option    = 'eom'            < - units of time for 'stop count', eom = end of month
 stop_count     = 1                < - how long in above units to run this segment (use yyyymmdd for date)
 time_mix_opt   = 'avgfit'         < - method to suppress leapfrog computational mode
 fit_freq       = 1                < - when using 'avgfit', the intervals per day into which full and half steps must fit
 time_mix_freq  = 17               < - requested frequency (in steps) for taking mixing steps
 dt_option      = 'steps_per_day' < - units for determining timestep (combined with dt count)
 dt_count       = 170              < - number of timesteps in above units to compute timestep
 impcor         = .true.           < - if .true., the Coriolis terms treated implicitly
 laccel         = .false.          < - if .true., tracer timesteps increase with depth
 accel_file     = 'unknown_accel_file'       < - file containing vertical profile of timestep acceleration factor
 dtuxcel        = 1.0              < - factor to multiply momentum timestep for different momentum and tracer tsteps
 allow_leapyear = .false.          < - use leap years in calendar
 iyear0         = 75               < - year (yyyy) at start of full run sequence
 imonth0        = 1                < - month at start of sequence
 iday0          = 1                < - day at start of sequence
 ihour0         = 0                < - etc.
 iminute0       = 0
 isecond0       = 0
 date_separator = ' '              <- character to separate yyyy mm dd in date (' ' means no separator)
/
```

```
&grid_nml
  horiz_grid_opt      = 'file'                < - read horizontal grid from a file OR create simple lat/lon grid
  horiz_grid_file    = ‘path_to_file/grid.3600x2400.fob.da'
  sfc_layer_opt       = 'varthick'            <- surface layer is variable thickness OR rigid lid OR old free
                                                 surface formulation
  vert_grid_opt       = 'file'                < - read vertical grid structure from file OR compute vertical grid internally
  vert_grid_file       = 'path_to_file/in_depths.42.dat'
  topography_opt     = 'file'
  topography_file     = 'path_to_file /kmt_noITF.big_endian' < - file containing depth level of each gridpoint
  partial_bottom_cells  = .true.                                  < - use partial bottom cells
  bottom_cell_file        = 'path_to_file/dzbc_pbc.p1_tripole.s2.0-og.20060315.no_caspian_or_black'
  region_mask_file       = 'unknown_region_mask'          < - file containing region number for each gridpoint
  topo_smooth           = .false.           < - if .true., smooth topography using 9-point averaging stencil
  flat_bottom             = .false.           < - if .true., flat bottom is used
  lremove_points          = .false.           < - if .true., remove isolated or disconnected ocean points
/

&init_ts_nml
  init_ts_option   = 'restart' < - start from restart OR read initial ocean conditions from a file OR
                                   create conditions from an input mean ocean profile OR create
                                   initial conditions based on 1992 Levitus mean ocean profile
                                   computed internally
  init_ts_file     = 'path_to_file/r.t0.1_42l_nccs01.00750101_fixedU' <- restart file OR file
                                                                         containing 3D potential
                                                                         temperature and salinity
                                                                         at grid points OR file
                                                                         containing depth pro-
                                                                         file of potential tempera-
                                                                         ture and salinity OR (ig-
                                                                         nored for ‘internal’ or
                                                                         when luse pointer files
                                                                         is enabled)
  init_ts_file_fmt = 'bin'            <- data format (binary or netCDF) for input init ts file (‘file’ and ‘restart’
                                         options only)
/

&diagnostics_nml
  diag_global_freq_opt   = 'nday'
  diag_global_freq      = 1          < - how often (in above units) to compute and print global diagnostics
  diag_cfl_freq_opt     = 'nday'
  diag_cfl_freq         = 1          < - how often (in above units) to compute and print CFL stability
                                       diagnostics
  diag_transp_freq_opt   = 'nday'
  diag_transp_freq      = 1          < - how often (in above units) to compute and print transport
                                       diagnostics
  diag_transport_file    = 'transport_file_141lines'
  diag_outfile          = 'diag'
  diag_transport_outfile = 'transp'
  diag_all_levels       = .false.     < - if true, tracer mean diagnostics at all vertical levels are output
  cfl_all_levels        = .false.
/
```

```
&restart_nml
  restart_freq_opt = 'nmonth'                < - units of time for 'restart freq'
  restart_freq     = 1                       < - number of units between output of restart files
  restart_outfile  = 'path_to_file/restart/r' < - root filename (with path prepended, if necessary)
                                                 for restart files ('runid' and suf- fixes will be added)
  restart_fmt      = 'bin'                   < - data format (binary or netCDF) for restart output files
  leven_odd_on     = .false.                 < - create alternating even/odd restart outputs
                                                 which over- write each other
  even_odd_freq    = 3840                    < - frequency (in steps) for even/odd output
  pressure_correction = .false.              < - if true, corrects surface pres- sure error due to (possible)
                                                 different timestep. use .false. for exact restart
/

&tavg_nml
  tavg_freq_opt  = 'nmonth'
  tavg_freq      = 1                         < - interval in above units for computation & output of time average history files
  tavg_start_opt = 'nstep'
  tavg_start     = 0                         < - time in above units after which to start accumulating time average
  tavg_infile    = ''                        < - restart file for partial tavg sums if starting from restart (ignored if luse pointer
                                                 files is enabled)
  tavg_fmt_in    = 'bin'                     <-  format for tavg restart file
  tavg_outfile   = 'path_to_file /tavg/t'
  tavg_fmt_out   = 'bin'                     < - format for tavg output files
  tavg_contents  = 'tavg_contents'           < - file name for input file containing names of fields
                                                  requested for tavg output
/

&history_nml
  history_freq_opt = 'never'                 < - this makes snapshot history files possible, we usually do not need
                                                 this, we want monthly mean history files so it's set to never
  history_freq     = 100000
  history_outfile  = 'unknown_history'
  history_fmt      = 'nc'
  history_contents = 'sample_history_contents'
/

&movie_nml
  movie_freq_opt  = 'nday'
  movie_freq      = 1                        < - number of units (movie_freq_opt) between output of movie files
  movie_outfile   = 'path_to_file'/movie/m'
  movie_fmt       = 'bin'
  movie_contents  = 'movie_contents'         < - file containing names of fields requested for movie output
/

&solvers
  solverChoice        = 'ChronGear'
  convergenceCriterion = 1.e-12 < - convergence   criterion: |δX/X| < convergenceCriterion
  maxIterations       = 1000        < - upper limit on number of iterations allowed
  convergenceCheckFreq = 25    < - check for convergence every convergenceCheckFreq  iterations
  preconditionerChoice = 'diagonal'
  preconditionerFile   = 'unknownPrecondFile'  < - file containing preconditioner coefficients for solver
```

```
&vertical_mix_nml
  vmix_choice = 'kpp'              < - method of computing vertical diffusion
  aidif      = 1.0                 < - time-centering parameter for implicit vertical mixing; use of the default
                                       value [1.0] is recommended
  bottom_drag = 1.0e-3             < - (dimensionless) coefficient used in quadratic bottom drag formula
  implicit_vertical_mix = .true.
  convection_type = 'diffusion'    < - convection treated by adjustment or by large mixing
                                       coefficients
  nconvad = 2                      < - number of passes through the convective adjustment algorithm
  convect_diff = 1000.0            < - tracer mixing coefficient to use with diffusion option
  convect_visc = 1000.0            < - momentum mixing coefficient to use with diffusion option
  bottom_heat_flx = 0.0            < - constant (geothermal) heat flux (W/m2) to apply to bottom layers
  bottom_heat_flx_depth = 100000.00 < - depth (cm) below which to apply bottom heat flux
/

&vmix_const_nml                    < - constant vertical mixing namelist
  const_vvc   = 0.25               < - vertical viscosity coefficient (momentum mixing) (cm2/s)
  const_vdc   = 0.25               < - vertical diffusivity coefficient (tracer mixing) (cm2/s)
/

&vmix_rich_nml                     < - Richardson-number vertical mixing namelist
  bckgrnd_vvc = 1.0                < - background vertical viscosity (cm2/s)
  bckgrnd_vdc = 0.1                < - background vertical diffusivity (cm2/s)
  rich_mix    = 50.0               < - coefficient for Richardson-number function
/

&vmix_kpp_nml
  bckgrnd_vdc1    = 0.55           < - base background       vertical  diffusivity (cm2 /s)
  bckgrnd_vdc2    = 0.303615       < - variation in background vertical diffusivity (cm2 /s)
  bckgrnd_vdc_dpth= 2500.0e2 < - depth (cm) at which background vertical diffusivity is vdc1
  bckgrnd_vdc_linv= 4.5e-5         < - inverse of the length scale (1/L in cm-1) over which diffusivity
                                       transition takes place
  Prandtl        = 10.0            < - (unitless) ratio of background vertical viscosity and diffusivity
  rich_mix       = 50.0            < - coefficient for Richardson-number function
  lrich          = .true.          < - use Richardson-number for interior mixing
  ldbl_diff      = .true.          < - add double-diffusive parameterization
  lshort_wave    = .true.          < - use penetrative shortwave forcing
  lcheckekmo     = .false.         < - check whether boundary layer exceeds Ekman or Monin-Obukhov
                                       limit
  num_v_smooth_Ri = 1              < - number of passes to smooth Richardson number
/

&advect_nml
  tadvect_ctype = 'centered'     < - centered differences OR 3rd-order up- winding
/

&hmix_nml
  hmix_momentum_choice = 'del4'        < - method for horizontal mixing of momentum (Laplacian,
                                           biharmonic or anisotropic)
  hmix_tracer_choice   = 'del4'        < - method for horizontal mixing of tracers (Laplacian, biharmonic
                                           or Gent-McWilliams)
/
```

```
&hmix_del2u_nml
  lauto_hmix        = .true.          < - computes mixing coefficient based on resolution
  lvariable_hmix    = .false.         < - scales mixing coeff by grid cell area
  am                = 1.e8            < - momentum mixing coefficient (cm2/s)
/

&hmix_del2t_nml
  lauto_hmix        = .true.          < - computes mixing coefficient based on resolution
  lvariable_hmix    = .false.         < - scales mixing coeff by grid cell area
  ah                = 1.e8            < - tracer mixing coefficient (cm2/s)
/

&hmix_del4u_nml
  lauto_hmix        = .false.          < - compute mixing coefficient based on resolution
  lvariable_hmix    = .true.           < - scale mixing coeff by grid cell area
  am                = -27.0e17         < - momentum mixing coeff (cm2/s)
/

&hmix_del4t_nml
  lauto_hmix        = .false.          < - compute mixing coefficient based on resolution
  lvariable_hmix    = .true.           < - scale mixing coeff by grid cell area
  ah                = -3.0e17          < -tracer mixing coefficient (cm2/s)
/

&hmix_gm_nml                           < - Gent-McWilliams horizontal mixing namelist
/

&hmix_aniso_nml                        < - anisotropic viscosity namelist
/

&state_nml                     <-  equation of state namelist
  state_choice = 'mwjf'        <-  McDougall et al. eos OR Jackett and McDougall eos
                                   OR polynomial fit to UNESCO eos OR linear eos
  state_file = 'internal'      < - compute polynomial coefficients inter- nally OR read from file filename
  state_range_opt = 'enforce'  < - ignore (ignore) when T,S outside valid polynomial range OR
                                   check (check) and report OR compute (enforce)  eos as if T,S
                                   were in valid range (but don't alter T,S)
  state_range_freq = 100000    < - frequency (steps) for checking T,S range
/

&baroclinic_nml
  reset_to_freezing = .true.   < - flag to prevent very cold water.
                                   if .true. and Tsurf(i,j) < Tfreezing, Tsurf(i,j) is reset to Tfreezing = -1.8°C
/

&ice_nml
  ice_freq_opt = 'never'       < - frequency units for computing ice formation
  ice_freq     = 100000        < - frequency in above units for com- puting ice formation
  kmxice   = 1                 < - compute ice formation above this vertical level
/
```

```
&pressure_grad_nml
  lpressure_avg = .true.          < - use pressure averaging to increase time step
  lbouss_correct = .false.        < - applies depth-dependent factor to correct for assumed constant density
/

&topostress_nml
  ltopostress = .false.           < - true if topographic stress enabled
  nsmooth_topo = 0                < - number of passes to smooth topography
/

&xdisplay_nml
  lxdisplay = .false.             < - if .true., enable x-display
  nstep_xdisplay = 1              < - frequency (in steps) for updating x-display
/

&forcing_ws_nml                   < - windstress forcing namelist
  ws_data_type    = 'monthly'     < - type or periodicity of wind stress forcing
  ws_data_inc     = 1.e20         < - increment (in hours) between forcing times if ws data type='n-hour'
  ws_interp_freq  = 'every-timestep'  < - how often to temporally interpolate wind stress data to
                                        current time
  ws_interp_type  = 'linear'      < - type of temporal interpolation for wind stress data
  ws_interp_inc   = 1.e20         < - increment (in hours) be- tween interpolation times if
                                      ws interp freq = 'n-hour'
  ws_filename     =
  '/work/e24/sar00059/sar00059/itamoc/scripts/prod_run3_0.5Sv/files_mat/forcing/ws.o_n_avg.mon' <-
                                  < - name of file containing wind stress, or root of filenames if
                                      ws data type='n-hour'
  ws_file_fmt     = 'bin'         < - format of wind stress file
  ws_data_renorm(1)  = 10.        < - renormalization constants for the components in the wind stress
                                      forcing file
  ws_data_renorm(2)  = 10.
/

&forcing_shf_nml                  < - surface heat flux forcing namelist
  shf_formulation    = 'normal-year'  < - surface heat flux formulation
  shf_data_type      = 'monthly'      < - type or periodicity of surface heat flux forcing
  shf_data_inc       = 1.e20          < - increment (in hours) between forcing times if shf data type='n-hour'
  shf_interp_freq    = 'every-timestep'  < - how often to temporally interpolate surface heat flux
                                           data to current time
  shf_interp_type    = 'linear'       < - type of temporal interpolation for surface heat flux data
  shf_interp_inc     = 1.e20          < - increment (in hours) be- tween interpolation times if
                                         shf interp freq = 'n-hour'
  shf_restore_tau    = 1.e20          < - restoring timescale (days) if type restoring
  shf_weak_restore   = 0.0            < - restoring flux for weak restor- ing in bulk-NCEP
  shf_strong_restore = 15.8           < - restoring flux for strong restoring in bulk-NCEP
  shf_filename       =
'/work/e24/sar00059/sar00059/itamoc/scripts/run_clim_closeditf/files_mat/forcing/shf.normal_year+Hurrell.monthly'
                                    < - name of file containing surface heat flux data, or root of
                                        filenames if shf data type='n-hour'
  shf_file_fmt       = 'bin'          < - format (binary or netCDF) of shf file
  shf_data_renorm(3) = 1.             < - renormalization constants for the components in the surface heat
                                        flux forcing file
```

```
  shf_data_renorm(4) = 1.
&forcing_sfwf_nml                    < - surface fresh water flux forcing namelist
  sfwf_formulation    = 'bulk-NCEP'  < - surface fresh water flux formulation. Bulk-NCEP means:
                                         calculate fluxes based on atmospheric state variables and
                                         radiation similar to a fully-coupled model (and using bulk
                                         flux formulations extracted from the NCAR flux coupler)
  sfwf_data_type      = 'monthly'    < - type or periodicity of surface fresh water flux forcing
  sfwf_data_inc       = 1.e20        < - increment (hours) between forcing times if sfwf data type='n-hour'
  sfwf_interp_freq    = 'every-timestep' < - how often to temporally interpolate surface fresh
                                         water flux data to current time
  sfwf_interp_type    = 'linear'     < - type of temporal interpolation for surface fresh water flux data
  sfwf_interp_inc     = 1.e20        < - increment (hours) between interpolation times if
                                         sfwf interp freq = 'n-hour'
  sfwf_restore_tau    = 1.e20        < - restoring timescale (days) if restoring
  sfwf_weak_restore   = 0.009        < - restoring flux for weak restoring in bulk-NCEP
  sfwf_strong_restore = 0.11         < - restoring flux for strong restoring in bulk-NCEP
  sfwf_filename       =
'/work/e24/sar00059/sar00059/itamoc/scripts/run_clim_closeditf/files_mat/forcing/sfwf.CORE+runoff.monthly
'                                    < - name of file containing surface fresh water flux data, or root of
                                         filenames if sfwf data type='n-hour'
  sfwf_file_fmt       = 'bin'        < - format (binary or netCDF) for sfwf file
  sfwf_data_renorm(1)    = 0.001     < - renormalization constants for components in
                                         sfwf forcing file
  sfwf_data_renorm(2)    = 1.
  ladjust_precip      = .true.       < - adjust precipitation to balance water budget
  lfw_as_salt_flx     = .true.       < - treat fresh water flux as virtual salt flux
                                         when using varthick sfc layer
  runoff            = .true.
/

&forcing_pt_interior_nml             < - interior potential temperature forcing namelist
  pt_interior_formulation = 'restoring' < - interior pt formulation
  pt_interior_data_type   = 'none'   < - type or periodicity of interior pt forcing
  pt_interior_data_inc    = 1.e20    < - increment (hours) between forcing times if data type 'n-hour'
  pt_interior_interp_freq = 'never'  < - how often to temporally interpolate interior pt data to
                                         current time
  pt_interior_interp_type = 'nearest' < - type of temporal interpolation for interior pt data
  pt_interior_interp_inc  = 1.e20    < - increment (hours) between interpolation times if
                                         interp freq      = 'n-hour'
  pt_interior_restore_tau = 1.e20    < - restoring timescale (days) if restoring
  pt_interior_filename    = 'unknown-pt_interior'      < - file containing interior pt data, or root of
                                                         filenames if data type='n-hour'
  pt_interior_file_fmt    = 'bin'    < - file format (binary or netCDF)
  pt_interior_data_renorm = 1.       < - renormalization constants for components in interior pt forcing file
  pt_interior_restore_max_level = 0  < - maximum level for interior pt restoring
  pt_interior_variable_restore  = .false. < - enable variable interior pt restoring
  pt_interior_restore_filename  = 'unknown-pt_interior_restore' < - name of file containing
                                                                  variable interior pt restoring data
  pt_interior_restore_file_fmt  = 'bin'  < - file format (binary or netCDF)
/
```

```
&forcing_s_interior_nml                  < - interior salinity restoring namelist
  s_interior_formulation = 'restoring'   < - forcing formulation
  s_interior_data_type   = 'none'        < - type or periodicity of interior salinity forcing
  s_interior_data_inc    = 1.e20         < - increment (hours) between forcing times if data type 'n-hour'
  s_interior_interp_freq = 'never'       < - how often to temporally interpolate interior S data to
                                             current time
  s_interior_interp_type = 'nearest'     < - type of temporal interpolation for interior S data
  s_interior_interp_inc  = 1.e20         < - increment (in hours) between interpolation times if
                                             interp freq 'n-hour'
  s_interior_restore_tau = 1.e20         < - restoring timescale (days) if restoring
  s_interior_filename    = 'unknown-s_interior'   < - name of file containing interior S data, or
                                                    root of filenames if data type 'n- hour'
  s_interior_file_fmt    = 'bin'         < - format (binary or netCDF) of s interior file
  s_interior_data_renorm = 1.            < - renormalization constants for components in interior S forcing file
  s_interior_restore_max_level = 0       < - maximum level for inte- rior S restoring
  s_interior_variable_restore  = .false. < - enable variable interior S restoring
  s_interior_restore_filename  = 'unknown-s_interior_restore'  < - name of file containing variable interior S
                                                                 restoring data
  s_interior_restore_file_fmt  = 'bin'
/

&forcing_ap_nml                          < - atmospheric pressure forcing namelist
  ap_data_type   = 'none'                < - type or periodicity of atmospheric forcing forcing
  ap_data_inc    = 1.e20                 < - increment (in hours) between forcing times if ap data type='n-hour'
  ap_interp_freq = 'never'
  ap_interp_type = 'nearest'
  ap_interp_inc  = 1.e20
  ap_filename    = 'unknown-ap'
  ap_file_fmt    = 'bin'
  ap_data_renorm = 1.
/
```

The subnamelist forcing_imau_nml below was added to enable additional surface freshwater flux (hosing).
If one wants to do a default POP run then remove this subnamelist from the pop_in namelist file.
The three files set by imau_filename, imau_filename_next and imau_filename_prev provide an additional surface freshwater flux forcing (hosing) in the form of a climatology (12 timesteps, one for each month) for the current year, the next year, and the previous year, respectively. Linear interpolation is used to compute the additional forcing for specific days. For days in January, data from December of the previous year is required,
while for days in December, data from January of the next year is needed. If the additional forcing is constant across all model years, then imau_filename, imau_filename_next, and imau_filename_prev will all have the same value. Interpolation will still be performed, even though it is not strictly necessary.

```
&forcing_imau_nml
 imau_data_type = 'monthly'
 imau_filename = '/home/dijkbio/models/pop/inputdata/gx1v6/sfwf_monthly/SFWF_gx1v6_imau.nc'
 imau_filename_next = '/home/dijkbio/models/pop/inputdata/gx1v6/sfwf_monthly/SFWF_gx1v6_imau.nc'
 imau_filename_prev = '/home/dijkbio/models/pop/inputdata/gx1v6/sfwf_monthly/SFWF_gx1v6_imau.nc'
/

&coupled_nml
  coupled_freq_opt = 'never'     <- unit of time for coupled freq
  coupled_freq     = 100000
/
```

```
&tidal_nml
/

&passive_tracers_on_nml
  dye_on = .false.
  iage_on = .false.
/


&dye_nml
  init_dye_option = 'restart'
  init_dye_init_file = 'same_as_TS'
  dye_region_file = 'blablabla'
  dye_region_file_fmt = 'bin'
  tracer_init_ext(1)%mod_varname  = 'DYE'
  tracer_init_ext(1)%filename     = 'unknown'
  tracer_init_ext(1)%default_val  = 0.0
  dye_tadvect_ctype = 'lw_lim'
/

&sw_absorption_nml
/

&float_nml
/
```

For more information about parameters that are not described here check the POP user guide at:
https://ncar.github.io/POP/doc/build/html/users_guide/index.html