CCAR-Modeling Documentation

Release 1.0

CCAR-Modeling Team

Mar 14, 2017

Contents

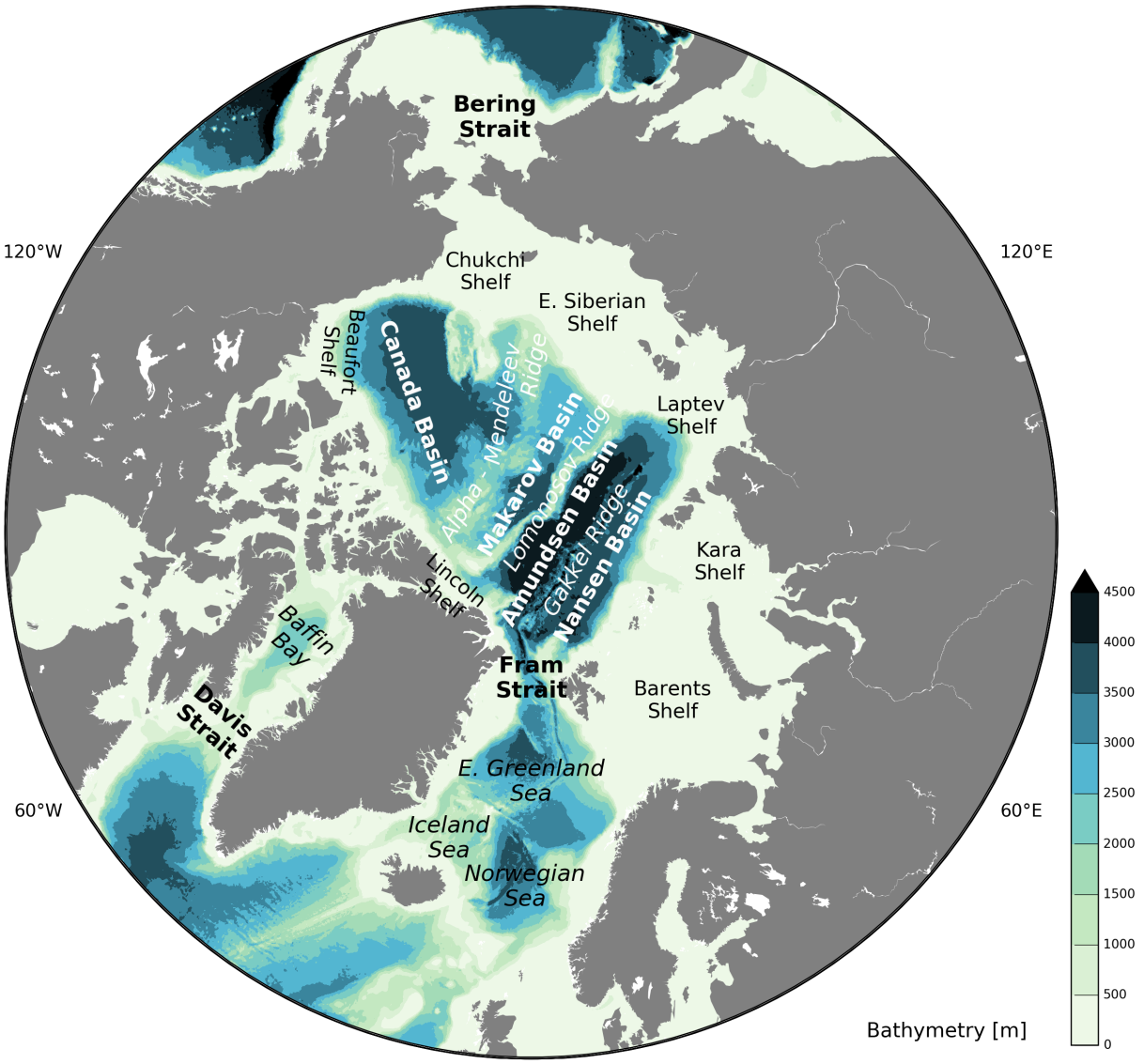
This is a collection of documentation about the CCAR Modeling Project. It is mostly focused on the UBC modeling component of the GEOTRACES project.

CHAPTER 1

About the Project

The Arctic Ocean as one of the major ocean basins in the high latitudes, is strongly affected by the climate change. The continuous warming trend since 1970s has resulted in significant sea-ice melt, affecting the hydrological cycle and potentially leading to the state of a "new Arctic". At present, due to the limited understanding of the climate-induced changes in high latitudes, Arctic-GEOTRACES Project, under the support of Climate Change and Atmospheric Research (CCAR), provides an opportunity to investigate the interaction of the physical and biogeochemical processes in the Arctic Ocean from the prospect of Trace Elements and Isotopes (TEIs).

The CCAR modeling team is developing a set of parameterizations under the architecture of NEMO 3.4 to simulate TEIs offline in the Arctic Ocean and incorporate the result with observations from the two back-to-back GEOTRACES cruises in the Canadian side of the Arctic and the Labrador Sea. The modeling efforts is aiming to produce refined estimates of the current state of the TEIs and examine its linkage with other factors in the climate system


Bering Strait
120°W
120°E
Chukchi Shelf
E. Siberian Shelf
Beaufort Shelf
Canada Basin
Mendeleev Ridge
Alpha
Laptev Shelf
Makarov Basin
Lomonosov Ridge
Amundsen Basin
Gakkel Ridge
Nansen Basin
Kara Shelf
Lincoln Shelf
Baffin Bay
Davis Strait
Fram Strait
Barents Shelf
E. Greenland Sea
60°W
60°E
Iceland Sea
Norwegian Sea
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Bathymetry [m]

CHAPTER 2

Contents

Quick Start Guide

Working on `ocean`

This section describes the steps to set up and test NEMO v3.4 on `ocean` .

Create a Workspace

NEMO v3.4 requires

- bash installed
- perl installed
- svn installed
- FORTRAN90 compiler installed
- netcdf installed

FORTRAN compiler on ocean cluster is G95.

The directory of netCDF library in `ocean` is:

```
cd /usr/lib
```

Create a space for NEMO v3.4 code and file I/O

```
mkdir -p /ocean/$NAME/GEOTRACES/
```

Access to NEMO v3.4 code

- Register in NEMO Homepage.

- Follow the steps on NEMO User Guide.
- Download NEMO v3.4. The package is "nemo_v3_4" or "dev_v3_4_STABLE_2012".
- One can also refers to a similar page on Salishsea-MEOPAR.
- Windows users can get the code by using TortoiseSVN for a backup.
- The tree structure of directory can be viewed on NEMO Quick Start Guide.

Change the permission of code directory

```
cd /ocean/$NAME
chmod -R a+x GEOTRACES
```

Compile the code

The arch file for NEMO

```
cd /ocean/$NAME/GEOTRACES/$CODEDIR/NEMOGCM/ARCH
vim arch-ocean.fcm
```

`arch-ocean.fcm`:

```
# generic gfortran compiler options for linux
# NCDF_INC    netcdf include file
# NCDF_LIB    netcdf library
# FC          Fortran compiler command
# FCFLAGS     Fortran compiler flags
# FFLAGS      Fortran 77 compiler flags
# LD          linker
# LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
# FPPFLAGS    pre-processing flags
# AR          assembler
# ARFLAGS     assembler flags
# MK          make
# USER_INC    additional include files for the compiler,  e.g. -I<include dir>
# USER_LIB    additional libraries to pass to the linker, e.g. -l<library>
# ============================================================================
%NCDF_INC             -I/usr/include
%NCDF_LIB             -L/usr/lib -lnetcdff
%FC                   gfortran
%FCFLAGS              -fdefault-real-8 -O3 -funroll-all-loops -fcray-pointer
%FFLAGS               %FCFLAGS
%LD                   gfortran
%LDFLAGS
%FPPFLAGS             -P -C -traditional
%AR                   ar
%ARFLAGS              -rs
%MK                   make
%USER_INC             %NCDF_INC
%USER_LIB             %NCDF_LIB
```

Configuring a testing case

```
cd /ocean/$NAME/GEOTRACES/$CODEDIR/NEMOGCM/CONFIG
./makenemo -m ocean -r ORCA2_OFF_PISCES -n case_name add_key "key_nosignedzero key_
↪netcdf4"
```

Run your case

Download forcing files from NEMO Homepage and place all the files in:

```
cd /ocean/$NAME/GEOTRACES/$CODEDIR/NEMOGCM/CONFIG/$case_name/EXP00
mv $forcing_file .
```

After changing NEMO's output in `ocean` and other options in different namelists. We can run the model

```
./opa &
```

The export information is saved in `ocean.output`.

MY_TRC Code

NEMO-code Repository

These notes describe the GEOTRACES Arctic project. This project is currently using the same NEMO code base as the Salish Sea MEOPAR *NEMO-code* repository

The *NEMO-code* repo is a Mercurial repository in which is maintained the merger of the trunk of the main NEMO **svn** repository and the changes made by the Salish Sea MEOPAR project team.

Note: The *NEMO-code* repository is a private repository for members of the Salish Sea MEOPAR project team and associated teams: GEOTRACES Arctic included. That is because it contains parts of the NEMO codebase. Although that codebase is openly licensed it's developers require registration to access the code.

If you have completed that registration and would like access to the *NEMO-code*, please contact Susan Allen, the Salish Sea MEOPAR project leader.

Getting the Code

Team members using SSH key authentication on Bitbucket may clone the *NEMO-code* repo with:

```
hg clone ssh://hg@bitbucket.org/salishsea/nemo-code NEMO-code
```

For password authentication use:

```
hg clone https://<you>@bitbucket.org/salishsea/nemo-code NEMO-code
```

where `<you>` is your Bitbucket user id.

Configurations

Making Gyre-Lobster with 2 Extra Tracers

To create a this new configuration based on, GYRE_LOBSTER use:

```
cd NEMO-code/NEMOGCM/CONFIG
./makenemo -r GYRE_LOBSTER -n YourTrcGyreLobster -m ocean add_key "key_netcdf4 key_
↪nosignedzero key_my_trc"
```

That will use the existing GYRE_LOBSTER configuration as a basis to build a new configuration called YourTrcGyreLobster with the ocean architecture definitions. The C Pre-Processor (CPP) keys key_netcdf4 and key_nosignedzero will be added to configurations. The key key_my_trc added the code in TOP_SRC/MY_TRC to the compile. The resulting configuration, including a compiled and link NEMO executable, is located in NEMO-code/NEMOGCM/CONFIG/YourTrcGyreLobster.

See **./makenemo -h** for details of options and sub-commands.

Running the Model

For now, we will run the model in the EXP00 directory. In future, once we have stabilized a few model configurations, we will move to running outside.

Inside EXP00 there are two namelist files: namelist_lobster and namelist_top, two output definition files: iodef.xml and xmlio_server.def and three links.

By including the key_my_trc flag we have added two tracers but have not defined them in the io file. We need to add four lines.

```
<field id="TR_7"     description="Northern Source"                                    unit=
↪"none" />
<field id="TR_8"     description="Southern Source"                                    unit=
↪"none" />
<field ref="TR_7"     />
<field ref="TR_8"     />
```

To get these lines and put them in the correct place, copy into your EXP00 the iodef.xml from the configuration MyTrcGyreLobster

```
cd YourGyreLobster/EXP00
cp ../../MyTrcGyreLobster/EXP00/iodef.xml .
```

We also need to add these tracers to namelist_top to initialize them

```
sn_tracer(7)    = 'TR_7'   , 'Southern Source            ',  'none      ' ,   .false.    ␣
↪ ,   .false.
sn_tracer(8)    = 'TR_8'   , 'Northern Source            ',  'none      ' ,   .false.    ␣
↪ ,   .false.
```

To get these lines and put them in the correct place, copy into your EXP00 the namelist_top from the configuration MyTrcGyreLobster

```
cp ../../MyTrcGyreLobster/EXP00/namelist_top .
```

In addition we need to modify two of the fortran codes. First we need a version of trcnam_trp.F90 that does not assume tracer damping has been set. Files that are changed from the base configuration go in your MY_SRC directory.

```
cd ../MY_SRC
cp ../../MyTrcGyreLobster/MY_SRC/trcnam_trp.F90 .
```

Second, the generic tracer source sink algorithm put the tracers into the Pacific... but our simulation is the Atlantic. We need a different trcsms_my_trc.F90. This is also the file you should edit to simulate your traces of choice.

```
cp ../../MyTrcGyreLobster/MY_SRC/trcsms_my_trc.F90 .
```

Now we need to remake the code. Go back upto CONFIG and run:

```
cd ../../
./makenemo -n YourTrcGyreLobster
```

Then we can run the code by going back into EXP00 and typing

```
cd YourTrcGyreLobster/EXP00
nice ./opa &
```

After a good little while, you will see

```
namelist read -->  F F nemo.x
↪                                        ionemo
filename : iodef.xml
Le parsing est termine !!!
trc_rst_wri_my_trc: No specific variables to write on unit           1  at time
↪  4318        4320
trc_rst_wri_my_trc: No specific variables to write on unit           1  at time
↪  4319        4320
trc_rst_wri_my_trc: No specific variables to write on unit           1  at time
↪  4320        4320
```

and then your job is done. Results from the tracers are in: `GYRE_5d_00010101_00011230_ptrc_T.nc`

you can look at this using a notebook, An example is at:

`/ocean/sallen/allen/research/MEOPAR/NEMO-code/NEMOGCM/CONFIG/MyTrcGyreLobster/EXP00/LookAtTracers.ipynb`

Making ORCA-Lim-PISCES with 2 Extra Tracers

This is almost identical to above except:

- In `namelist_top` these are traces sn_tracer(25) and sn_tracer(26) as PISCES has more tracers. Note, however, for some undetermined reason, they are still called TR_7 and TR_8.
- You don't need to copy a new version of trcnam_trp.F90
- and don't forget that you need to download all the forcing files for ORCA-LIM-PISCES

Running NEMO Offline using 5-day Files

NEMO is designed to use yearly, monthly, weekly or daily files but Paul Myers' group is producing five-day (fday) files.

We have re-written daymod.F90, dom_oce.F90 and fldread.F90 to handle fday, non-climatology files. Also see the namelist. All the files are commited in configuration CindyOff in MY_SRC and EXP00.

To run with fday files you need to: 1) make unspecified data files. NEMO uses these to look at how many variables are in each type of file. You can make these by linking to any specific time file. So, for example,

```
ln -s ANHA4-EXH001_gridT.nc ANHA4-EXH001_y2010m01d05_gridT.nc
```

You need a gridT, gridU, gridV, gridW and icemod file.

2) reorganize the order in the naming of the files: the date needs to come last AND 3) the dates need to be the beginning of the five days, not the end. So for example:

```
ln -s ANHA4-EXH001_gridT_y2010m01d01.nc ANHA4-EXH001_y2010m01d05_gridT.nc
```

Reading netCDF4 file in MY_TRC

This section describes the ways to read an user-defined nerCDF4 file during the time stepping of `MY_TRC` in NEMO 3.4.

Through NEMO's existing file channel

By replacing the variable information in `namelist/&namdta_dyn` to your own, the tracer model can read your files during the simulation. An example of it can be found in here. *(this section needs more explainations)*

By creating a new namelist

Note: The examples here and below are requiring a netCDF4 file in `EXP00` which contains **nav_lat**, **nav_lon**, **time_counter** and **var_name**. The netCDF4 file should have the same longtitude/latitude dimensions as the file `coordinate.nc` in your case. The time dimension should be "UNLIMITED" and **var_name** is the variable you want `MY_TRC` to read.

The arrangment of the dimensions (For Python users) should be (TIME, LAT, LON).

Create namelist_my_trc

In `$your_case/EXP00/`, create the file `namelist_my_trc`:

```
&namelist_section
!,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
↪,,,,,,,,,,,,,,,,,,,,,,,,,,
!      ! file name ! frequency (hours) ! variable ! time interp. !  clim  ! 'yearly'/␣
↪! weights  ! rotation !
!      !           !  (if <0  months)  !   name   !   (logical)  !  (T/F) ! 'monthly'␣
↪! filename ! pairing  !
sn_var ='file_name',        -12        ,'var_name',    .false.   , .true. ,  'yearly'␣
↪,     ''    ,     ''
cn_dir = './'
/
```

- **frequency** is the reading frequency of the variable in time dimension
- **clim** is the flag of file batching ...
- A section should be ended with `/`

Edit MY_TRC scripts

The structure and access information can be added in `trcini_my_trc.F90` and `trcnam_my_trc.F90`. When the variable is read, it can be used in `trcsms_my_trc.F90`.

In `trcini_my_trc.F90`, assign the structure of your file by `fld_fill(...)`.

```
IMPLICIT NONE
PRIVATE

PUBLIC   trc_ini_my_trc   ! called by trcini.F90 module
CONTAINS

SUBROUTINE trc_ini_my_trc
  IF(trc_sms_my_trc_alloc() /= 0) THEN
     CALL ctl_stop('STOP', 'trc_ini_my_trc: unable to allocate MY_TRC arrays')
  ! Assign structure
  CALL fld_fill(sf_var, (/sn_var/), cn_dir, 'trc_ini_my_trc', 'docs', 'namelist_
↪section')
END SUBROUTINE trc_ini_my_trc
```

In `trcnam_my_trc.F90`, read the variable through `ctl_opn(...)`. The name of the namelist `"namelist_my_trc"` should be consistent with the one created in the section above.

```
IMPLICIT NONE
PRIVATE

PUBLIC   trc_nam_my_trc   ! called by trcnam.F90 module

CONTAINS

SUBROUTINE trc_nam_my_trc
  INTEGER :: numnatl
  NAMELIST/namelist_section/ cn_dir, sn_var
  CALL ctl_opn(numnatl, 'namelist_my_trc', 'OLD', 'FORMATTED', 'SEQUENTIAL', 1,␣
↪numout, .FALSE.)
  REWIND(numnatl)
  READ  (numnatl, namelist_section)
END SUBROUTINE trc_nam_my_trc
```

`trcsms_my_trc.F90` call the two scripts above and allocates the array.

```
IMPLICIT NONE
PUBLIC

PUBLIC   trc_sms_my_trc       ! called by trcsms.F90 module
PUBLIC   trc_sms_my_trc_alloc ! called by trcini_my_trc.F90 module

CHARACTER(len=100), PUBLIC :: cn_dir = './'    ! Root directorY
TYPE(FLD_N) :: sn_var                          ! information about the file to be read
REAL(wp), ALLOCATABLE, DIMENSION(:,:) :: var   ! Array receives the value from netCDF
TYPE(FLD), ALLOCATABLE, DIMENSION(:) :: sf_var ! structure variable (PUBLIC for TAM)

CONTAINS

SUBROUTINE trc_sms_my_trc( kt )
  INTEGER, INTENT(in) :: kt ! ocean e-step index
  INTEGER :: i, j
  IF(nn_timing == 1) CALL timing_start('trc_sms_my_trc')
  !
  CALL fld_read (kt, 1, sf_var)
  var(:, :) = sf_var(1)%fnow(:, :, 1)
  ! More code ...
END SUBROUTINE trc_sms_my_trc
```

```
INTEGER FUNCTION trc_sms_my_trc_alloc()
  INTEGER :: ierror
  ALLOCATE(var(jpi,jpj), STAT=trc_sms_my_trc_alloc)
  ALLOCATE(sf_var(1), STAT=ierror)
  IF(ierror > 0) THEN
     CALL ctl_stop('trc_sms_my_trc_alloc: unable to allocate');
     RETURN
  ENDIF
  ALLOCATE(sf_var(1)%fnow(jpi, jpj, 1))
  IF(trc_sms_my_trc_alloc /= 0) THEN
     CALL ctl_warn('trc_sms_my_trc_alloc : failed to allocat')
END FUNCTION trc_sms_my_trc_alloc
```

For 4 dimension variables (time dimension has been subtrackted by keyword "frequency" in the namelist): var(:, :, :) = sf_var(1)%fnow(:, :, :).

Adding user-defined tracers

This section describes the way to build-up the tracer scheme in NEMO 3.4 based on the `MY_TRC` module.

Edit `MY_TRC` files

The following files are useful when adding tracers into the model.

Script	Location	Functionality
`par_my_trc.F90`	`TOP_SRC/MY_TRC`	Claim the number & indexing of tracers
`trcsms_my_trc.F90`	`TOP_SRC/MY_TRC`	Initialization & parameterization
`trcnxt.F90`	`TOP_SRC/MY_TRC`	Boundary conditions

An ideal way to edit these files is copying them to `MY_SRC` and without changing things in the original folder. Here some examples are provided, but the real editing depends on the type of tracer. The OPA Tracer Mannuel explained that `MY_TRC` is designed for "passive tracers".

The example below sets user defined tracer as `.TRUE.` and claimed two tracers with index `jpmyt1` and `jpmyt2`. So during the simulation, the first tracer can be indexed as: `trn(lon, lat, dep, jpmyt1)`.

`par_my_trc.F90`:

```
! Line 43
  !!----------------------------------------------------------------------
  !!   'key_my_trc'                      user defined tracers (MY_TRC)
  !!----------------------------------------------------------------------
  LOGICAL, PUBLIC, PARAMETER ::   lk_my_trc     = .TRUE.   !: PTS flag
! Line 48
! assign an index in trc arrays for each PTS prognostic variables
  INTEGER, PUBLIC, PARAMETER ::   jpmyt1 = jp_lm + 1 !: 1st MY_TRC tracer
  INTEGER, PUBLIC, PARAMETER ::   jpmyt2 = jp_lm + 2 !: 1st MY_TRC tracer
```

`trcsms_my_trc.F90` is an important file to work on. The example below demostrates the way of doing initialization, derivative and variable printing.

`trcsms_my_trc.F90`:

```
! Initialization
     IF( kt < 5 ) THEN
```

```
        WRITE(*, *) '~~~~ Initialization ~~~~'
        trn(:, :, :, jpmyt1) = your_ini_field(:, :, :)
     ENDIF
! Param time derivative
tra(:, :, 1, jpmyt1) = tra(:, :, 1, jpmyt1) + delta
! Print surface tracer values to check
WRITE(*, *) 'trn@Lena estuary', trn(600, 450, 1, jpmyt1)
```

`kt` is the timestep, `trn` is the storage of tracer value and `tra` is the "delta" term. *(this block needs more explainations)*

The example below is the open boundary condition for our group. In order to preserve tracer values at the boundary from the impact of advection from "zero grids", you can hold values at the boundary as their initial condition.

`trcnxt.F90`:

```
! Inside the scope of SUBROUTINE trc_nxt( kt )
! Set grids below 60N latitude to zero (only simulates Arctic)
      DO jn = 1, jptra
        DO jk = 1, jpk
           WHERE (gphit < 60)
              tra(:, :, jk, jn) = 0.
           END WHERE
        ENDDO
      ENDDO
```

Note: It is always necessary to read initial/boundary conditions and other physical fields when simulating the tracer, this part of information is in here.

Edit I/O options‘

`EXP00/iodef.xml` in general contains two parts: (1) the variable definition section and (2) the output section. User-defined tracers should be defined in (1), ideally the `ptrc_T` group. The information of output frequency is in (2), varies from 1-day to 10-year.

`iodef.xml`:

```
<!-- In ptrc_T scope -->
<group id="ptrc_T" axis_ref="deptht" grid_ref="grid_T">
      <!-- My Stupid Tracer -->
      <field id="T01" description="My tracer 01" unit="none" />
      <field id="T02" description="My tracer 02" unit="none" />
</group>
```

The output definition locates at the bottom of the file.

```
<!-- In the output definition scope -->
<file_definition >
     <group id="5d" output_freq="432000" output_level="10" enabled=".TRUE.">
       <file id="5d_ptrc_T" name="auto" description="pisces sms variables" >
          <field ref="T01" /> <!-- output every 5-day -->
          <field ref="T02" />
       </file>
     </group>
     <group id="1m" output_freq="-1"     output_level="10" enabled=".TRUE.">
       <file id="1m_ptrc_T" name="auto" description="pisces sms variables" >
```

```
        <field ref="T01" /> <!-- output every month -->
        <field ref="T02" />
      </file>
    </group>
</file_definition>
```

Edit `namelist_top`

`namelist_top`:

```
!                !    name   ! title of the field ! initial data ! initial data !
↪save   !
!                !           !                   !  units       ! from file    ! or
↪not !
!                !           !                   !              ! or not       !
↪   !
   sn_tracer(1)  = 'T01'     , 'My tracer 01'     ,  'none'      ,  .false.     ,  .
↪true.
   sn_tracer(2)  = 'T02'     , 'My tracer 02'     ,  'none'      ,  .false.     ,  .
↪true.
/
```

The original `namelist_top` contains many existing `sn_tracer(#)` fields, these are prepared for PISCES and can be deleted. `namelist_top` can set-up the restart tracer files which is useful for long-term simulations

Calculate emps with ANHA4 Output

This section describes the steps to calculate evaporation minus precipitation (emps) term in `MY_TRC` with ANHA4 forcing fields. This approache was tested with forcing files from ANHA4-EXH001 and EXH005.

The equation for emps is [Schmitt, 1989]:

$$\mathrm{emps}(:,:) = -\frac{\mathrm{iocesafl}(:,:) * \mathrm{soce}}{\mathrm{rday} * (\mathrm{isssalin}(:,:) + 1.0\mathrm{E}^{-16})}$$

- `iocesafl` is the salt flux at ocean surface, defined in icemod files.
- `isssalin` is Sea surface salinity, defined in icemod files.
- `soce` is 34.7, `rday` is 3600*24, they are constants.

In NEMO 3.4, the "p" part of emps indlues both precipitation and sea-ice melt but does **not** take runoff into account. In Kyle's research, emps was calculated for estimating the dilution of tracers (combined with runoff) and the parameterization of isotopic fractionation. By definition, net precipitation minus emps is the sea-ice melt water.

Use `namelist/&namdta_dyn`

A simple way to read `iocesafl` and `isssalin` in the model is using the file channels in `namelist/&namdta_dyn`. sn_emp can be used, and since ANHA4-EXH001/EXH005 does not have "key_eiv", here we also choose sn_eiw.

In `namelist`, add the following

```
!-----------------------------------------------------------------------
&namdta_dyn        !   offline dynamics read in files                ("key_offline")
!-----------------------------------------------------------------------
!            !  file name  ! frequency (hours) ! variable  ! time interp. !  clim  !
↪'yearly'/ ! weights  ! rotation !
!            !             !  (if <0  months)  !   name    !   (logical)  !  (T/F) !
↪'monthly' ! filename ! pairing  !
    sn_emp  = 'ANHA4-EXH001_icemod',    120    , 'iocesafl',    .true.    , .false., ␣
↪ 'fday'  , ''       , ''
    sn_eiw  = 'ANHA4-EXH001_icemod',    120    , 'isssalin',    .true.    , .false., ␣
↪ 'fday'  , ''       , ''
 /
```

Then copy `datdyn.F90` from `OFF_SRC` and on row 295, add the following:

```
emp(:, :) = -1.0*emp(:, :)*34.7/(3600.0*24*(aeiw(:, :)+1.0e-16))
emps(:,:) = emp(:,:)
```

Reference

- 18. (a) Schmitt, P. S. Bogden, and C. E. Dorman. Evaporation minus precipitation and density fluxes for the North Atlantic. J. Phys. Oceanogr., 19(9):1208–1221, 1989.

Modeled TEIs

Dissolved Barium

Biogeochemical Properties

Dissolved Barium is a type of bio-intermediate element which in general behaves like hard-part nutrients [Chan et al., 1977, Falkner et al., 1993].

Dissolved Barium was first posed as a tracer of Arctic river and halocline water by Falkner et al. [1994]. Then the on-going field measurements in Beaufort Sea, Chukchi Sea, Laptev Sea and Eurasian marginal seas found that North American rivers like Yukon and Mackenzie have dissolved Barium concentration significantly higher than major Eurasian rivers. Therefore, dissolved Barium is able to separate North American runoff from Eurasian runoff. [Guay, 1997, Guay and Falkner, 1997, 1998, Taylor et al., 2003]. Since the ??background?? surface dissolved Barium level in the Arctic Ocean is lower than both Eurasian and North American runoff, dissolved Barium also acts as a proxy of Arctic runoff water in general [Falkner et al., 1994].

Parameterization set-up

To come

Reference

- 12. (a) Chan, D. Drummond, J. M. Edmond, and B. Grant. On the Barium data from the Atlantic GEOSECS expedition. Deep Sea Res., 24(7):613?C649, 1977.
- 11. (a) Falkner, T. Bowers, J. Todd, B. Lewis, W. Landing, J. Edmond, et al. The behavior of Barium in anoxic marine waters. Geochim. Cosmochim. Acta, 57(3):537?C554, 1993.
- K. K. Falkner, R. W. MacDonald, E. C. Carmack, and T. Weingartner. The potential of Barium as a tracer of Arctic water masses. In O. M. Johannessen, R. D. Muench, and J. E. Overland, editors, The polar oceans

and their role in shaping the global environment, volume 85 of AGU Geophysics Monograph, pages 63?C76. American Geophysical Union, Washington, D.C., 1994.

- 3. (a) Guay and K. K. Falkner. Barium as a tracer of Arctic halocline and river waters. Deep Sea Res. Part II: Topical Studies in Oceanography, 44(8):1543?C1569, 1997.
- 3. (a) Guay and K. K. Falkner. A survey of dissolved Barium in the estuaries of major Arctic rivers and adjacent seas. Cont. Shelf Res., 18(8):859?C882, 1998.

Oxygen Isotope Ratio

To come

FAQs and Debugs

NEMO 3.4 Configuration Issues

namelist not terminated

Error message:

```
Fortran runtime error: namelist not terminated with / or &
```

Debug

Each section (divided by "&") should end with "/"

Building failed on limrhg.F90

You may see the message like:

```
make: *** No rule to make target `limrhg.o', needed by `limdyn_2.o'.  Stop.
```

When you are building a ORCA2_LIM, ORCA2_LIM3 or ORCA2_LIM_PISCES. This is because limrhg.F90 in LIM3 cannot link successfully to LIM2.

Debug

Link limrhg.F90 manually:

```
cd /ocean/$NAME/GEOTRACES/$CODEDIR/NEMOGCM/NEMO/LIM_SRC_2
rm -rf limrhg.F90 # delete the file
ln -s ../LIM_SRC_3/limrhg.F90 # link to limrhg.F90 in LIM_SRC_3
```

You can also directly copy and replace the limrhg.F90 script on NEMO/LIM_SRC2:

```
cd /ocean/$NAME/GEOTRACES/$CODEDIR/NEMOGCM/NEMO/LIM_SRC_2
cp -rf ../LIM_SRC_3/limrhg.F90 limrhg.F90 # copy in force
```

Fortran runtime error for line 214 on nemogcm.F90

You may see the message like:

```
fortran runtime error, line 214 on nemogcm.f90
```

When you are running all kinds of configurations contain ORCA2, LIM2 or LIM3 component. This is because namelists in ORCA2, LIM2, LIM3 reference configuration do not link successfully to your configuration.

The line 214 on nemogcm.f90 is:

```
READ(numnam, namctl)
```

Debug

Copy and replace the namelist, namelist_ice, namelist_ice_lim2, namelist_ice_lim3.

```
cd /ocean/$NAME/GEOTRACES/$CODEDIR/NEMOGCM/CONFIG/$case_name/EXP00/
cp -rf ../../ORCA2_LIM/EXP00/namelist namelist
cp -rf ../../ORCA2_LIM/EXP00/namelist_ice_lim2 namelist_ice
cp -rf ../../ORCA2_LIM/EXP00/namelist_ice_lim2 namelist_ice_lim2
cp -rf ../../ORCA2_LIM/EXP00/namelist_ice_lim3 namelist_ice_lim3
```