HOW-TO manual for CEN Earth System Assimilation Model (CESAM)

Iuliia Polkova*

with input from

Guokun Lyu, Silke Schubert, Frank Lunkeit and Armin Köhl

*Institute of Oceanography, Universität Hamburg

This manual contains information about Plasim and MITgcm components of the adjoint model CESAM as well as a few examples for forward and adjoint model simulations.

*Corresponding author: iuliia.polkova@uni-hamburg.de Date: 30.09.2023

Content

1. TAF Licence

The Universität Hamburg, Remote Sensing and Data Assimilation (RSDA) group purchased a group license for TAF software (Transformation of Algorithms in Fortran) from the FastOpt company. Thus, if you are affiliated with this research group, you do not need to acquire the license independently but to activate the TAF software with a license key. An ssh key pair is generated by the `staf` script. The private key is copied to the `.ssh` directory and the public key is sent to FastOpt automatically. After that, you should get confirmation from FastOpt per email. The script `staf` can be obtained from the RSDA group at the university or from the FastOpt.

Model webpage:

www.cen.uni-hamburg.de/en/research/cen-models/cesam.html

Contact persons to get the model source code:

Prof. Detlef Stammer, detlef.stammer@uni-hamburg.de

Dr. Armin Köhl, armin.koehl@uni-hamburg.de

Contact persons regarding the license:

Dr. Armin Köhl, armin.koehl@uni-hamburg.de and

Dr. Ralf Giering, ralf.giering@fastopt.com, www.fastopt.com/products/taf/taf.shtml

2. Computer cluster "marin"

Model runs are carried out on the CEN computer cluster called **marin** (https://www.cen.uni-hamburg.de/facilities/cen-it/compute.html).

Login to marine is possible with the DKRZ user account:

```
ssh -X userid@marinN.cen.uni-hamburg.de
```

Choose number ***N*** from 02, 03, or 04.

Run model experiments in a working directory of the research group, where you are affiliated, for instance: `cd /scratch/ifmrs/userid`

2.1. (Re)-Compile the model on the new cluster

Load modules

```
> module load intel/19.0.4
> module load openmpi/2.0.0-static-intel19
```

To get the linking sequence, run in the terminal
`/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/bin/nf-config --flibs`

To get the path to the include file, run in the terminal
`/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/bin/nf-config --inludedir`

Modify `ifmli+ifort+mpi` and `ifmli+ifort+serial` which are located in the subdirectory of the ocean model `MITgcm_c62y/tools/build_options`

E.g., the new `ifmli+ifort+serial` will look smth like, where nothing except the paths was modified (in bold):

```
#!/bin/bash
#

FC='ifort'
F90C='ifort'
CC='icc'
LINK='ifort -no-ipo -mcmodel=medium -shared-intel'
DEFINES='-DWORDLENGTH=4'
CPP='cpp -traditional -P'
F90FIXEDFORMAT='-fixed -Tf'

NOOPTFLAGS='-O0 -g -m64 -fPIC'
```

```
NOOPTFILES=''

INCLUDEDIRS=''
INCLUDES=''
LIBS=''

FFLAGS='-132 -r8 -i4 -W0 -WB -convert big_endian -assume byterecl -fPIC
-mcmodel=medium -shared-intel'
FOPTIM='-O2 -align -ip'

CFLAGS='-O2 -ip -fPIC'


   INCLUDEDIRS="/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/include/"
   INCLUDES="-I/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/include/"
#      INCLUDES:= -I$(shell /sw/buster-x64/netcdf_fortran-4.5.2-gccsys/bin/nf-config
--includedir)
#      LIBS:= $(shell /sw/buster-x64/netcdf_fortran-4.5.2-gccsys/bin/nf-config
--flibs)
   LIBS="-L/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/lib -Wl,-rpath
-Wl,/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/lib -lnetcdff
-L/sw/buster-x64/netcdf_c-4.7.4-gccsys/lib -Wl,-rpath
-Wl,/sw/buster-x64/netcdf_c-4.7.4-gccsys/lib -lnetcdf -lnetcdf -ldl -lm"
```

And the new `ifmli+ifort+mpi` will look smth like, where nothing except the paths was modified (in bold):

```
#!/bin/bash
#

FC='mpif90'
F90C='mpif90'
CC='mpicc'
LINK='mpif90 -no-ipo -mcmodel=medium -shared-intel'
DEFINES='-DALLOW_USE_MPI -DALWAYS_USE_MPI -DWORDLENGTH=4'
CPP='cpp  -traditional -P '
F90FIXEDFORMAT='-fixed -Tf'

NOOPTFLAGS='-O0 -g -m64 -fPIC'
NOOPTFILES=''

INCLUDEDIRS=''
INCLUDES=''
LIBS=''

FFLAGS='-132 -r8 -i4 -W0 -WB -convert big_endian -assume byterecl -fPIC
-mcmodel=medium -shared-intel'
FOPTIM='-O2 -align -ip'

CFLAGS='-O2 -ip -fPIC'

#      INCLUDES:= -I$(shell /sw/buster-x64/netcdf_fortran-4.5.2-gccsys/bin/nf-config
--includedir)
#      LIBS:= $(shell /sw/buster-x64/netcdf_fortran-4.5.2-gccsys/bin/nf-config
--flibs)
```

```
   INCLUDEDIRS="/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/include/"
   INCLUDES="-I/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/include/"
   LIBS="-L/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/lib -Wl,-rpath
-Wl,/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/lib -lnetcdff
-L/sw/buster-x64/netcdf_c-4.7.4-gccsys/lib -Wl,-rpath
-Wl,/sw/buster-x64/netcdf_c-4.7.4-gccsys/lib -lnetcdf -lnetcdf -ldl -lm"

       MPI_INC_DIR="/sw/buster-x64/mpi/openmpi-4.1.4-static-intel22/include"
#      MPI_INC_DIR="/sw/buster-x64/mpi/mpich-4.0.3-static-intel22/include"

INCLUDES="$INCLUDES -I$MPI_INC_DIR"
INCLUDEDIRS="$INCLUDEDIRS $MPI_INC_DIR"
MPIINCLUDEDIR="$MPI_INC_DIR"
MPI_HEADER_FILES='mpif.h mpiof.h'
```

The LIBS part is appended manually with `-Wl` option in bold:
`LIBS="-L/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/lib` **`-Wl,-rpath -Wl,/sw/buster-x64/netcdf_fortran-4.5.2-gccsys/lib -lnetcdff`** `-L/sw/buster-x64/netcdf_c-4.7.4-gccsys/lib` **`-Wl,-rpath -Wl,/sw/buster-x64/netcdf_c-4.7.4-gccsys/lib`** `-lnetcdf -lnetcdf -ldl -lm"`

Then copy `mpif.h` form `/sw/buster-x64/mpi/openmpi-4.1.4-static-intel22/include/mpif.h` to the directory where you have your main `Makefile`. For me it is `/scratch/ifmrs/uXXXXXX/CESAM_4cpu_template` and change the content of the `mpif.h` by copying the content of the `/sw/buster-x64/mpi/openmpi-4.1.4-static-intel22/include/mpif*` files in it. Normally, one should not do that, but this work-around is implemented because the TAF compiler cannot recognise `mpif.h` and without this `make adm` will keep giving errors about not found `mpif*` files.

After this you should be able to compile the model as usual with `make function`. Do `make scratch` before if you are recompiling the model to clean up previously generated data.

Not used, just FYI, another way to figure out the paths for `ifmli+ifort+mpi` and `ifmli+ifort+serial` is to run `module load netcdf` and `nf-config --all`

```
This netCDF-Fortran 4.5.2 has been built with the following features:

  --cc          -> gcc
  --cflags      ->
-I/sw/buster-x64/io/netcdf-c-4.7.4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/include
-I/sw/buster-x64/io/netcdf-c-4.7.4-for
```

```
tran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/include -I/sw/buster-x64/io/libaec-1.0.4-gccsys/include
-I/sw/buster-x64/io/hdf5-1.10.6-gccsys/include

  --fc         -> gfortran
  --fflags     ->
-I/sw/buster-x64/io/netcdf-c-4.7.4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/include
  --flibs      ->
-L/sw/buster-x64/io/netcdf-c-4.7.4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/lib -lnetcdff
-L/sw/buster-x64/io/netcdf-c-4.7.          4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/lib
-Wl,-rpath,/sw/buster-x64/io/netcdf-c-4.7.4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/lib
-L/sw          /buster-x64/io/libaec-1.0.4-gccsys/lib
-Wl,-rpath,/sw/buster-x64/io/libaec-1.0.4-gccsys/lib
-L/sw/buster-x64/io/hdf5-1.10.6-gccsys/lib -Wl,
-rpath,/sw/buster-x64/io/hdf5-1.10.6-gccsys/lib -lnetcdf -ldl -lm -lnetcdf -lhdf5_hl -lhdf5
-lz -lcurl
  --has-f90   ->
  --has-f03   -> yes

  --has-nc2   -> yes
  --has-nc4   -> yes

  --prefix     -> /sw/buster-x64/io/netcdf-c-4.7.4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys
  --includedir->
/sw/buster-x64/io/netcdf-c-4.7.4-fortran-4.5.2-cxx4-4.3.1-cxx-4.2-gccsys/include
  --version   -> netCDF-Fortran 4.5.2
```

3. CESAM Model

3.1. Resolution

The ocean model contains 90 × 44 longitude–latitude grid cells equivalent to a 4° horizontal resolution, with 15 vertical layers. The atmosphere is resolved on a spectral T21 horizontal resolution (64 x 32 grid cells or 5.61° at equator) and split into 10 equidistant vertical sigma levels. The land surface is represented by the original Earth orography discretized on the T21 Gaussian grid (Stammer et al 2018).

The **ocean time step** is set to **8 hours** (`TClock=28800` seconds, **3 time steps per day**) and the **time step** in the **atmosphere** is **48 minutes** (`TClock=2880` seconds, **30 time steps per day**). The ocean model and atmosphere model exchange sea surface temperature, heat and momentum fluxes every **8 hours**.

3.2. Model components

The atmospheric model PLASIM is described here:
https://www.mi.uni-hamburg.de/en/arbeitsgruppen/theoretische-meteorologie/modelle/sources/psusersguide.pdf

The ocean model MITgcm is described here https://mitgcm.readthedocs.io and
http://mitgcm.org/public/r2_manual/latest/online_documents/manual.pdf

The model is available in two configurations:

- **Maximal:** with moisture parameterizations in Plasim and
- **Minimal:** without moisture parameterizations in Plasim.

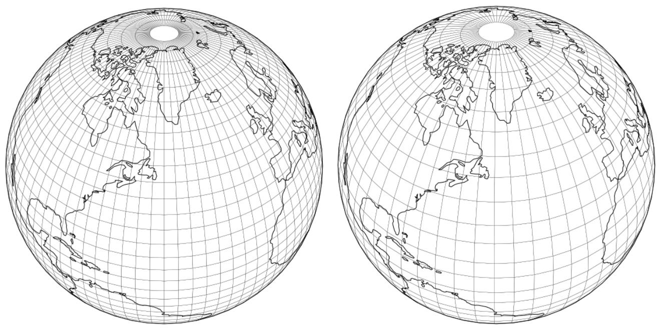


Figure: MITgcm grid (left), Plasim grid (right).

3.3. Set up the model

```
>> tar -xvf CESAM_4cpu.tar

>> module load intel/22.2.1

>> module load openmpi/4.1.0-static-intel22

>> module load cdo

>> cd CESAM_4cpu
```

- On thunder, the libraries used were `intel/17.0.4` and `openmpi/2.0.0-static-intel17`

Directory structure

```
drwxr-sr-x 3 userid ifmrs 4096 Feb 6 10:41 src_2dmap_2dweight
-rw-r--r-- 1 userid ifmrs 210 Feb 6 10:41 Makefile.bak
drwxr-sr-x 3 userid ifmrs 4096 Feb 6 10:41 drivers_parallel
drwxr-sr-x 13 userid ifmrs 4096 Feb 6 10:41 MITgcm_c62y
-rw-r--r-- 1 userid ifmrs 117 Feb 6 10:42 test_radius.m
-rwxr--r-- 1 userid ifmrs 13498 Feb 6 10:42 staf
-rw-r--r-- 1 userid ifmrs 3053 Feb 6 10:42 PACKAGES_CONFIG.h
drwxr-sr-x 2 userid ifmrs 4096 Feb 6 10:42 omit_ad
drwxr-sr-x 2 userid ifmrs 4096 Feb 6 10:42 numrec
-rw-r--r-- 1 userid ifmrs 24825 Feb 6 10:42 mpif.h
-rw-r--r-- 1 userid ifmrs 18808 Feb 6 10:42 Makefile.old
-rw-r--r-- 1 userid ifmrs 18455 Feb 6 10:42 Makefile_mpi
-rw-r--r-- 1 userid ifmrs 18261 Feb 6 10:42 Makefile_ifort
-rw-r--r-- 1 userid ifmrs 3013 Feb 6 10:42 genmake.log
-rw-r--r-- 1 userid ifmrs 4770 Feb 6 10:42 filter_parallel.f
-rw-r--r-- 1 userid ifmrs 4770 Feb 6 10:42 filter_parallel25.f
-rw-r--r-- 1 userid ifmrs 4778 Feb 6 10:42 filter.f_25
```

```
-rw-r--r-- 1 userid ifmrs 4770 Feb 6 10:42 filter.f
-rw-r--r-- 1 userid ifmrs 3448 Feb 6 10:42 depend_tl.inc
-rw-r--r-- 1 userid ifmrs 2920 Feb 6 10:42 depend_cpl.inc
-rw-r--r-- 1 userid ifmrs 3915 Feb 6 10:42 depend_ad.inc
-rw-r--r-- 1 userid ifmrs 390 Feb 6 10:42 AD_CONFIG.h
drwxr-sr-x 2 userid ifmrs 4096 Feb 6 10:42 bin
-rw-r--r-- 1 userid ifmrs 18805 Feb 6 10:46 Makefile
drwxr-sr-x 6 userid ifmrs 8192 Feb 6 11:30 run
drwxr-sr-x 3 userid ifmrs 4096 Feb 6 11:31 bld_mpi
```

Source code

Directories `src` usually contain a source code of a model, which represents governing equations of motion:

- The source code of the oceanic model MITgcm is in `MITgcm_c62y/model/src`. Directory `verification` contains model configurations e.g., `adplethora/code_t21p_forward_op_2datm`.
- The atmospheric source code for PLASIM is in `src_2dmap_2dweight`.
- `omit_ad`: contains adjoint codes (that are sometimes modified manually!).
- `numrec`: minimization algorithms (LBFGS algorithm or DFP algorithm).
- `mpif.h`: used when generating adjoint codes. Note: Don't touch it, unless you change your compiler!

Executables

- `Makefile`: makefile for compiling all executables, you will often modify something in this file. In other models it is generated automatically, but in CESAM one modifies things manually in this file, e.g., providing the directories of the source code and the length of the state vector.
- `bin`: contains all executables: `function`, `function_genobs`, `tstadm`, `opti`. They have to be copied manually to the directory, where one runs the simulations.
- `bld_mpi`: all generated codes. If you want to modify smth in the model code, copy the program codes to `MITgcm/verification/adplethora/code_.../`, modify the code there. Then after compiling the model, you can check if your modifications are implemented in the built code in `bld_mpi`.
- `drivers_parallel`: all main programs for the executables. Note: For tailored 4D-VAR runs, it is needed to change a subroutine `prgdfpmin_ad.f90`.

Experiment directory

- `run`: you will submit/run model simulations here. The directory contains restarts, namelist, data, etc.

License

- `staf`: commercial software for generating adjoint code. Note: One needs to get a license (`./staf -setup`) by contacting http://www.fastopt.com/contact.shtml
- Generated when compiling: `depend_tl.inc`, `depend_cpl.inc`, `depend_ad.inc`.
- If you have further questions about `staf` contact http://www.fastopt.com/contact.shtml

3.4. Create executables

This step automatically copies all necessary source files from `src*` to `bld_mpi` and modifies them according to the selected parameter configuration specified in `code_t21p_forward_op_2datm` and `src_2dmap_2dweight`. The original source codes in the `src*` directories are not changed**. The program modules are then compiled and linked using the `make` command. The executable is then automatically copied to the models `bin` directory** after building. Run `make` commands where `Makefile` is located. **What needs to be done manually is to copy an executable from `bin` to the `run` directory,** where you run model experiments.

```
>> make scratch # run this if you need to clean up the directory.
If you run things for the first time. There is nothing to clean up.

>> make function_genobs # run this if you need to generate "pseudo
observations". This is usually needed only when you run twin
experiments with the adjoint. They are generated for the last
atmospheric model time step of the run. Number of atmospheric model
time steps for creation of pseudo observations is set in
plasim_namelist

>> make function # this is the executable for the forward runs and
the cost function calculation. After running make function new
Makefile has been generated in
/scratch/cen/ifmrs/userid/CESAM_4cpu/bld_mpi
```

3.5. Example of experiment's configurations

The directory `MITgcm_c62y/verification/adplethora` contains coupled and uncoupled model configurations.

```
ls /CESAM_4cpu/MITgcm_c62y/verification/adplethora

code_t21p_forward_atm_only
code_t21p_forward
test_code
```

3.6. Running the model with MPI

Run the model with the appropriate `mpirun` provided with your particular implementation of MPI. CESAM runs can be launched as follows:

```
mpirun -np 4 function > simulation_log.txt &
```

If the simulation experiences an error it will not be displayed in `simulation_log.txt` and also on the screen if you have `&` at the end of the command.

4. General guide to customize your experiments

A complete list of MITgcm model namelist runtime parameters set in the file `data`, which needs to be located in the `run` directory, where you submit model experiments.

Model parameters are defined and declared in the file `PARAMS.h` and their default values are generally set in the routine `set_defaults.F`, otherwise when initialized in the routine `ini_parms.F`. These files are in the `MITgcm_c62y/model/src/` and `bld_mpi.`

The "execution environment" namelist parameters is in file `eedata`, which must reside in the current `run` directory as well.

For the full list of MITgcm parameters in the namelist `data` visit the mitgcm website.

4.1. Start and duration of model simulation

The most common parameters to change are start time and duration of the run:

- The beginning of a simulation is set by specifying a start time (in seconds) through the real variable `startTime` or by specifying an initial iteration number through the integer variable `nIter0`. If these variables are set to non-zero values, the model will look for a "`pickup`" file (by default, `pickup.0000nIter0`) to restart the integration.
- The end of a simulation is set through the real variable `endTime` (in seconds). Alternatively, one can specify the number of time steps to execute through the integer variable `nTimeSteps`.
- Iterations are referenced to `deltaTClock`, i.e., each iteration is `deltaTClock` seconds of model time. Here, it is `deltaTClock = 28800.,` which corresponds to 8 hours.

4.2. Initial conditions for model simulation

- Ocean restarts are named `pickup.00000nIter` and `pickup_cd.00000nIter`. Numbers in the name of the file stand for the number of timesteps `pickup_cd.(Xend/deltaX).data`. Existence of the pickup file in the `run` directory forces the model to start from the pickup file even if `nIter0=0`. If the model does not find these files, it will start from the default initial conditions, which represent Levitus temperature and salinity climatology fields: `templev_90x44x15.bin` and `saltlev_90x44x15.bin`.

- Atmosphere restarts are named `plasim_restart`. If the model does not find this file, it will start from the default initial condition `surface.txt` that contains default climatological data such as surface geopotential, land-sea mask, surface roughness, background albedo, glacier mask, bucket size, soil temperature, climatological annual cycle of the surface temperature, climatological annual cycle of the soil wetness.

4.3. Writing output of model simulation

By default, MITgcm writes output (snapshots, diagnostics, and pickups) separately for individual tiles (four tiles for CESAM ocean) of the Earth, leaving it to the user to merge these into global files. There is an option however to have the model do this automatically. According to the `mitgcm` manual, this could be done as follows:

- Setting `globalFiles` to `.TRUE.` should always work in a single process setup (including multi-threaded processes), but for MPI runs this will depend on the platform – it requires simultaneous write access to a common file (permissible in typical Lustre setups, but not on all file systems). Alternatively, one can set `useSingleCpuIO` to `.TRUE.` to generate global files. Note: However, this did not work on thunder and marin clusters.
- Thus, alternatively, Silke Schubert wrote a script to glue the MITgcm outputs a posteriori.

Frequency/Amount of Output:

The frequency (in seconds), with which output is written to disk, needs to be specified in `data` too. `dumpFreq` controls the frequency with which the instantaneous state of the model is written. `monitorFreq` controls the frequency with which monitor output is dumped to the standard output files. The **frequency of output** is referenced to `taveFreq`. For instance, `taveFreq = 2592000.,` which corresponds to 30-day averages. Other parameters are `dumpFreq` to specify interval of the model state/snapshot data and `dumpInitAndLast` to write out the initial and last iteration model state on/off flag.

4.4. Post-processing of model output

Some post-processing and visualization scripts are available from MITgcm developers MITgcm/utils.

Ocean data needs to be glued

You will need these two files to glue the data:

- `gluemncbig.x` # the one which actually glues data

- `glueexbig.job` # the script which executes `gluemncbig.x` for large amounts of data. This script is written by Silke Schubert, silke.schubert@uni-hamburg.de

Modify the path to the data in `glueexbig.job` and execute as: `csh glueexbig.job`

Atmospheric data needs to be transformed from binary to NetCDF

You will need these two files to transfer data to NetCDF:

- `burn7` # the one which actually glues data
- **`all_pl.nl`** # the script which executes `gluemncbig.x` for large amounts of data. This script is provided by Frank Lunkeit, frank.lunkeit@uni-hamburg.de

Execute it as: `./burn7 < all_pl.nl > test.out  -d ./puma_output ./puma_out.nc`

5. Forward simulation

5.1. Objective

Get an idea about the model's mean state behavior (For instance, what the AMOC strength is, what kind of atmospheric modes the model captures, etc).

What one can do with this type of experiments

- Long-term free running experiments are often used as control/benchmark simulations to evaluate a model's undisturbed mean state and variability. Next, we can do a long term simulation to evaluate the model's climatology and compare how this low resolution model performs as compared to observations.
- This setup can be used to run retrospective climate predictions also known as hindcasts.

5.2. Instructions to run the experiment

1. Create new `run` directory for the new experiment
2. After compiling the model, copy `function` executable from `bin` to the `run` directory.
3. Here is the list of files in the `run` directory that you will need to edit in order to customize your experiment:

    - Namelist files:
        - `data` # configuration file for the ocean
        - `puma_namelist` # configuration file for the atmosphere
        - `data.cal` # calendar file

Model calendar - 360 day calendar		

`In data.cal:`
 	
`TheCalendar = `**`'model'`**
	`In data:`
	
	`nTimeSteps = 3 steps/day * 360 days = `**`1080`**
	
	`pChckFreq = chckFreq = monitorFreq = 86400 sec/day * 360 days = `**`31104000`**
		`In puma_namelist:`
		
		`N_DAYS_PER_YEAR = 360`

Gregorian calendar - 365 day calendar		

In data.cal:
 	
TheCalendar = **'gregorian'**
	In data:
	
	nTimeSteps = 3*365 = **1095**
	
	pChckFreq = chckFreq = monitorFreq = 86400 * 365 = **31536000**
		In puma_namelist:
		
		N_DAYS_PER_YEAR = 365

To set the length of the simulation, it is sufficient to specify the length of the simulation only in the `data` file. That is done in `nTimeSteps`. For instance, the ocean time step is set to 8 hours (`TClock=28800` seconds, 3 time steps per day). `nTimeSteps=10800` means that the run duration will be 3600 days or 10 years.

Note, in Guokun's model configuration: The maximum timestep is `nIter0=12960` (He set it to a maximum 12 years). I need my simulation to be longer than 12 years, so I needed to modify the `checkpoint 2` in the `tamc.h` file `(MITgcm_c62y/verification/adplethora/code_t21p_forward_…/tamc.h` with the originally set value to `nchklev_2=4321`. Increase this parameter `nchlev_2` and modify `nIter0` and `nTimeSteps` such that `nTimeSteps < nchlev_1*nchlev_2`.)

- **Default restarts:**
  - Binary data in `*bin` or `*data` files: If the run does not start from the wished `pickup`, it will start from the default initial conditions that is contained in temperature and salinity data files: `templev_90x44x15.bin` and `saltplev_90x44x15.bin`, for the ocean model, and `surface.txt`, for the atmospheric model.
- **Target restarts:**
  - `pickup.00000nIter` and `pickup_cd.00000nIter` files are restart files for the ocean: `pickup_cd.(Xend/deltaX).data`. Existence of the file in the `run` directory forces the model to start from the pickup file even if `nIter0=0`.
  - `plasim_restart`: If you submit a follow-up run, do: `cp plasim_status plasim_restart`. Otherwise, it starts from `surface.txt`

To enable atmospheric output, in `puma_namelist` change `NOUTPUT= 0` to `NOUTPUT= 1`

The CESAM model is designed for optimization experiments. So one has to take care that the cost function calculation in `puma_namelist` is switched off `ncost=0` in free runs.

Submit the run as **`mpirun -np 4 function > function.out &`**.

If you want to run the simulation year by year, Guokun (run.sh and burn.sh), Silke and Yulia (submit_year.sh), wrote various shell scripts to do that which contain the following steps:

- Edit `data` file and link required initial conditions.
- Submit the run using `mpirun -np 4 function > function.out`.
- Prepare restarts for another year or simulation.
- Postprocess.

Running the model year by year has an advantage that the atmospheric model saves restarts for each year, otherwise if you run the simulation in a large chunk, the atmospheric model will only have one restart at the end of the run.

Note: even when switching off the calculation of the cost function in `puma_namelist`, the cost function is still calculated at the end of the run. For calculating the cost, it needs 12 values in `temp_init*` and `salt_init* data`. Otherwise, the model gives an error and the job is terminated. The output of the ocean and the atmosphere are written down anyway. But the atmospheric restart is not written. Thus, if one needs to make a follow-up run, it will not start from the end of this simulation, because the atmospheric restart is not written down, but from a climatological restart `surface.txt` instead. As a work-round, one can add 12 values in those files, e.g., in `matlab`:

```
wrslice('temp_initial_f64.data',[90 44 15 12],1,'real*8')
wrslice('salt_initial_f64.data',[90 44 15 12],1,'real*8')
```

The model will not complain and will write down the atmospheric restarts. It would have been better though not to calculate the cost function if it is not really needed. Ion did some modification for this not to happen (He modified `code_t21p_forward`, the results are in `test_code)`. However, I am using the version of Guokun with the regularization scheme that does not include Ion's modifications.

5.3. Result of the exercise

Initial conditions for the follow-up run and the output data

- Ocean restarts `pickup.00000nIter` and `pickup_cd.00000nIter` files
- Atmosphere restarts `plasim_status` are written automatically at the end of each run. It is given a different name so as not to overwrite the old `plasim_restart` file accidentally. For continuing the run you need to copy `plasim_status` to `plasim_restart`
- `U., V., W., T., S., Eta.00000nIter` # files with the instantaneous state of the model

- **Ocean output** will be in `mnc`. However, the global domain is split into 4 tiles! See in FAQ how to glue data.
- **Atmos output** will be in the file `puma_output`. In order to change it to NetCDF format execute `./burn7 < all_pl.nl > test.out  -d ./puma_output ./puma_out.nc` This will store data in `puma_out.nc`. Check plasim user guide. All available variable codes are at the end of these manuals. Alternatively, to see the full list of Plasim variables, you can execute `./burn7 -c`. Put the variable code that you want to be in the NetCDF file in `all_pl.nl`.

Usually, 10 model years are run in 50 minutes. Because we write output data it could take a bit longer. 10 minutes for 1 year is reasonable.

```
cdo sinfo tave.0000000000.t001.nc
Warning (cdfScanVarAttr): NetCDF: Variable not found - >XC<
Warning (cdfScanVarAttr): NetCDF: Variable not found - >YC<
Warning (cdfScanVarAttr): NetCDF: Variable not found - >RC<
Warning (cdfScanVarAttr): NetCDF: Variable not found - >XU<
Warning (cdfScanVarAttr): NetCDF: Variable not found - >YU<
Warning (cdfScanVarAttr): NetCDF: Variable not found - >XV<
Warning (cdfScanVarAttr): NetCDF: Variable not found - >YV<
Warning (cdfInqContents): Coordinates variable iter can't be assigned!
Warning (cdfInqContents): Coordinates variable iter can't be assigned!
Warning (cdfInqContents): Coordinates variable iter can't be assigned!
Warning (cdfInqContents): Coordinates variable iter can't be assigned!
Warning (cdfInqContents): Coordinates variable iter can't be assigned!
Warning (cdfInqContents): Coordinates variable iter can't be assigned!
   File format : NetCDF
       -1 : Institut Source   T Steptype Levels Num      Points Num Dtype : Parameter ID
       1 : unknown  unknown  v instant     1    1         990    1  F64   : -1
       2 : unknown  unknown  v instant     1    1         990    1  F64   : -2
       3 : unknown  unknown  v instant     15    2        990    1  F64   : -3
       4 : unknown  unknown  v instant     15    2        990    1  F64   : -4
       5 : unknown  unknown  v instant     15    2        1001    2  F64   : -5
       6 : unknown  unknown  v instant     15    2        1080    3  F64   : -6
       7 : unknown  unknown  v instant     15    3        990    1  F64   : -7
       8 : unknown  unknown  v instant     15    2        1001    2  F64   : -8
       9 : unknown  unknown  v instant     15    2        1080    3  F64   : -9
       10 : unknown  unknown  v instant    15    3        990    1  F64   : -10
       11 : unknown  unknown  v instant    15    2        1001    2  F64   : -11
       12 : unknown  unknown  v instant    15    2        1080    3  F64   : -12
       13 : unknown  unknown  v instant    15    3        990    1  F64   : -13
       14 : unknown  unknown  v instant    15    2        990    1  F64   : -14
       15 : unknown  unknown  v instant    15    2        1001    2  F64   : -15
       16 : unknown  unknown  v instant    15    2        1080    3  F64   : -16
       17 : unknown  unknown  v instant    15    2        990    1  F64   : -17
       18 : unknown  unknown  v instant    15    2        990    1  F64   : -18
       19 : unknown  unknown  v instant    15    2        990    1  F64   : -19
       20 : unknown  unknown  v instant    1    1         990    1  F64   : -20
       21 : unknown  unknown  v instant    1    1         990    1  F64   : -21
       22 : unknown  unknown  v instant    15    2        990    1  F64   : -22
       23 : unknown  unknown  v instant    1    1         990    1  F64   : -23
       24 : unknown  unknown  v instant    1    1         990    1  F64   : -24
       25 : unknown  unknown  v instant    1    1         990    1  F64   : -25
       26 : unknown  unknown  v instant    1    1         990    1  F64   : -26
   Grid coordinates :
       1 : lonlat                    : points=990 (90x11)
```

```
                             X : 2 to 358 by 4 degrees_east  circular
                             Y : -86 to -46 by 4 degrees_north
       2 : lonlat                   : points=1001 (91x11)
                             Xp1 : 0 to 360 by 4 degrees_east
                             Y : -86 to -46 by 4 degrees_north
       3 : lonlat                   : points=1080 (90x12)
                             X : 2 to 358 by 4 degrees_east  circular
                             Yp1 : -88 to -44 by 4 degrees_north
   Vertical coordinates :
       1 : surface                  : levels=1
       2 : generic                  : levels=15
                             Z : -25 to -4855 meters
       3 : generic                  : levels=15
                             Zl : 0 to -4510 meters
   Time coordinate :  11 steps
       RefTime =  1990-01-01 00:00:00  Units = seconds  Calendar = standard
  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss
  1990-01-31 00:00:00  1990-03-02 00:00:00  1990-04-01 00:00:00  1990-05-01 00:00:00
  1990-05-31 00:00:00  1990-06-30 00:00:00  1990-07-30 00:00:00  1990-08-29 00:00:00
  1990-09-28 00:00:00  1990-10-28 00:00:00  1990-11-27 00:00:00
cdo    sinfo: Processed 26 variables over 11 timesteps [0.07s 42MB].
```

6. Atmospheric nudging

6.1. Objective

Get familiarized with the nudging routines of CESAM. Nudge atmospheric component toward ERA-Interim fields during one year.

What one can do with this type of experiments

- Nudging runs are used in climate predictions to assimilate available reanalysis into the prediction system. Usually atmospheric and oceanic reanalyses are assimilated simultaneously into the coupled model. In this experiment, we will learn how to assimilate data from the ERA5 reanalysis.

- The nudging strength in the ocean for climate predictions is about 10 to 30 days, whereas in the atmosphere it depends on the variable and varies from 6 hours for vorticity, 24 hours for temperature and surface pressure to 48 hours for divergence (see Polkova et al 2019, https://doi.org/10.1029/2018MS001439).

- Here we use 1-day nudging in the atmosphere for humidity, divergence and vorticity in one simulation and for the same parameters plus temperature in the second simulation. The biases are compared in the final figure.

6.2. Nudging options

- To **switch on/off** atmospheric nudging go to `/scratch/cen/ifmrs/userid/CESAM_4cpu/src_2dmap_2dweight/Makefile` and uncomment the second line:

```
L278:

plasim.f90: plasim.F90
$(CPP) $(CPPOPTS)    $< > $@                              # nudging is off
#$(CPP) $(CPPOPTS) -DNUDGING   $< > $@                     # nudging is on
```

- All nudging fields should be in one file `observations_nudg.srv`. **For 365-day calendar, it contains the following number of records: 24*365 + 2 time steps.** See next subsection on how to prepare the data.
- A list of **fields** that are **nudged** are in the table below. If you need nudging only at some model levels, specify 10 values (one for each level) separated by comma.

From `puma_namelist`:

```
 tnudg = 10*0.0        temperature 10 levels 0-flag for no nudging
 dnudg = 10*1.0        divergence 10 levels 1-day nudging
 qnudg = 10*1.0        moisture 10 levels 1-day nudging
 znudg = 10*1.0        vorticity 10 levels 1-day nudging
 pnudg = 0.            surface pressure 0-flag for no nudging
```

6.3. Instructions to run the experiment

1. Make sure that the nudging option is switched on by checking if the flag `-DNUDGING` in `src_2dmap_2dweight/Makefile` for `plasim.f90` is active (see table above).

2. `make scratch`

3. `make function`

4. Create `run` directory for the new experiment. Copy necessary files (restarts, namelist, etc.) from the default run directory in this new `run` directory.

5. Copy `function` executable from `bin` to the `run` directory. Rename it to `function_relax`

6. Copy the last pickup files from the control run here.
    - And modify `nIter0` e.g., as follows: `nIter0 = 324000`, if the name of the pickup file is `pickup*.0000324000.001.001.data`
    - Normally nudging is started from the spin-up or historical run. Thus, just link the initial conditions to the restarts of that run, e.g., as `ln /control_run/output/plasim_status_0000324000 plasim_restart`

Model calendar - 360 day calendar		

In data.cal:
 	
TheCalendar = **'model'**
	In data:
	
	nTimeSteps = 3 steps/day * 360 days = **1080**
	
	pChckFreq = chckFreq = monitorFreq = 86400 sec/day * 360 days = **31104000**
		In puma_namelist:
		
		N_DAYS_PER_YEAR = 360

Gregorian calendar - 365 day calendar		

In data.cal:	In data:	In puma_namelist:

TheCalendar = **'gregorian'**	nTimeSteps = 3*365 = **1095**
	
	pChckFreq = chckFreq = monitorFreq = 86400 * 365 = **31536000**
		N_DAYS_PER_YEAR = 365

7. For a test, use a prepared nudging `observation_nudg.srv` The next section explains how to prepare custom nudging fields. Use `cdo` to check what is in `srv` as `cdo -sinfo observations_nudg.srv` The file with nudged fields needs to contain the data plus one extra timestep. Otherwise without it, the model will finish without a pickup file. Note: It is important that the model time matches the time in the nudging files!

```
 File format : SERVICE  BIGENDIAN
       -1 : Institut Source   T Steptype Levels Num       Points Num Dtype : Parameter
ID
       1 : unknown  unknown  v instant      10   1          2048   1  F64  : 138
       2 : unknown  unknown  v instant      10   1          2048   1  F64  : 155
       3 : unknown  unknown  v instant      10   1          2048   1  F64  : 130
       4 : unknown  unknown  v instant      10   1          2048   1  F64  : 133
       5 : unknown  unknown  v instant      1    2   2048    1   F64   : 134
   Grid coordinates :
       1 : generic                  : points=2048  (64x32)
   Vertical coordinates :
       1 : generic                  : levels=10
                           lev : 1 to 10 by 1
       2 : surface                  : levels=1
   Time coordinate :  unlimited steps
  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss
  0001-01-01 00:00:00  0001-01-01 00:06:00  0001-01-01 00:12:00  0001-01-01 00:18:00
  0001-01-02 00:00:00  0001-01-02 00:06:00  0001-01-02 00:12:00  0001-01-02 00:18:00
 ................................................................................
   ................................................................................
   .....
  0001-12-30 00:00:00  0001-12-30 00:06:00  0001-12-30 00:12:00  0001-12-30 00:18:00
  0001-12-31 00:00:00  0001-12-31 00:06:00  0001-12-31 00:12:00  0001-12-31 00:18:00
  0002-01-01 00:00:00
cdo    sinfo: Processed 5 variables over 1461 timesteps [0.30s 44MB].
```

5 variables are `var138` - vorticity, `var155` - divergence, `var130` - air temperature, `var133` – specific humidity and `var134` - surface pressure. **1461 timesteps** are for 4 time steps per day over 365 days.

8. The model is designed for the optimization experiments. Thus, one has to take care that the cost function calculation in `puma_namelist` is switched off `NCOST=0`.

9. Nudging at all vertical levels for divergence, vorticity and humidity (10*1.0) and no nudging for pressure and temperature in this experiment. Multiplication by 1 means 1-day nudging.

From `puma_namelist`:

```
tnudg = 10*0.0        temperature 10 levels 0-flag for no nudging
dnudg = 10*1.0        divergence 10 levels 1-day nudging
qnudg = 10*1.0        moisture 10 levels 1-day nudging
znudg = 10*1.0        vorticity 10 levels 1-day nudging
pnudg = 0.            surface pressure 0-flag for no nudging
```

10. Edit `data` file: For one year nudging run `nTimeSteps = 1080`

11. BUG: Use 12 value data sets for `temp_` and `salt_initial_f64.data`. They are needed for the cost (even though it is switched off, somehow it is not switched off completely), otherwise the `plasim_status` will not be written down
```
ln -s OBS/gecco3/temp_gecco3.data temp_initial_f64.data
ln -s OBS/gecco3/salt_gecco3.data salt_initial_f64.data
```

12. Submit the nudging run using
```
mpirun -np 4 function_relax > function_relax.out &
```

6.4. Result of the exercise

It is the one year assimilation run, where ERA5 was nudged into CESAM. Plot time series of the original ERA5 and the nudged run to see how close the nudged run is to the original data.

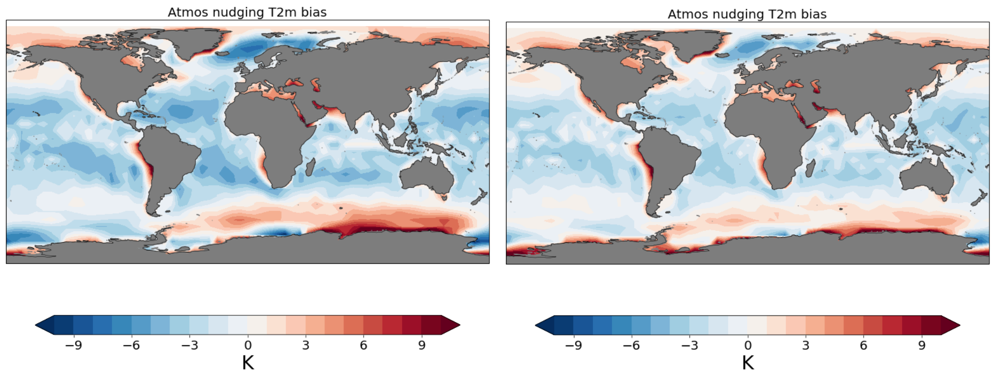


Figure: Annual bias of the 2-meter temperature from the two nudging runs: (left) without and (right) with temperature nudging. The bias is calculated with respect to the ERA5 data.

7. Ocean nudging

7.1. Objective

Get familiarized with the nudging routines of CESAM. Nudge oceanic component toward GECCO3 fields during one year.

What one can do with this type of experiments

- Nudging runs are used in climate predictions to assimilate available reanalysis into the model. Usually atmospheric and oceanic reanalyses are assimilated simultaneously into the coupled model. In this experiment, we will learn how to assimilate data only from the ocean GECCO3 reanalyses.

- The nudging strength in the ocean for climate predictions is about 10-30 days (see Polkova et al 2019, https://doi.org/10.1029/2018MS001439).

7.2. Nudging options

The ocean nudging package is called `rbcs`. For the parameters, check `rbcs_readparms.F`. To enable ocean nudging add the package name to `packages.conf` and get the file `data.rbcs` in the `run` directory, enable nudging in `data.pkg` with `useRBCS =.true.`

- The `rbcs` package is described on https://mitgcm.readthedocs.io/en/latest/phys_pkgs/rbcs.html?highlight=relax#introduction
- http://svante.mit.edu/~jscott/html/phys_pkgs/rbcs.html#introduction
- http://mailman.mitgcm.org/pipermail/mitgcm-support/2007-May/004862.html

7.3. Instructions to run the experiment

1. The version of the MITgcm that is used in CESAM is defined in the Makefile as `P_CODE  = verification/adplethora/code_t21p_forward`

2. Go to this directory and check if the `rbcs` is in the `packages.conf` as follows `cat MITgcm_c62y/verification/adplethora/code_t21p_forward/packages.conf`

```
#
#  $Header:
/u/gcmpack/MITgcm/verification/my_run/code/packages.conf, Exp
$
#  $Name: checkpoint62x $
#

autodiff
ctrl
grdchk
ecco
cal

oceanic
cd_code
zonal_filt

mnc
diagnostics
timeave
```

3. In my case, `rbcs` was not in the list, so I needed to add the `rbcs` package in this list of already included packages in the file `packages.conf`

4. In the `run` directory, enable nudging in `data.pkg` with `useRBCS =.TRUE.,`

5. Recompile the model:
   `make scratch`
   `make function`
   `cp bin/function run/function_relax`

6. In the run directory in the namelist file that is called `data.rbcs`, if you do not have this namelist file, write it as follows: in the `PARM01` section add:

```
 &RBCS_PARM01
# useRBCtemp   # RBCS for T
# useRBCsalt   # RBCS for S

useRBCtemp = .TRUE.,
useRBCsalt = .TRUE.,
#  relaxTFile   # File containing theta climatology used for relaxation
#  relaxSFile   # File containing salt climatology used for relaxation
#  relaxMaskFile   # File containing 3-D mask for (1=temperature, 2=salinity)

relaxTFile = 'temp_gecco3.data',
relaxSFile = 'salt_gecco3.data',
relaxMaskFile =
'Temp_gecco3_mask.data','salt_gecco3_mask.data',  # T and S masks
for nudging.

#  tauRelaxT  # Relaxation to climatology time scale (s)
#  tauRelaxS  # Relaxation to climatology time scale (s)

tauRelaxT = 2628000.0,
tauRelaxS =2628000.0,

#   rbcsForcingPeriod   #  Time  interval  between  forcing  fields  (s)
31536000/12=2628000
#   rbcsForcingCycle     #   Repeat  cycle  of  forcing  fields  (s),  e.g.
86400*365=31536000

rbcsForcingPeriod = 2628000.,
rbcsForcingCycle = 31536000.,
&
```

7. Repeat steps 6-12 from the experiment described in 6.3.

8. Coupled nudging

8.1. Objective

See 6.1 and 7.1.

8.2. Nudging options

See 6.2 and 7.2.

I do a long-nudging run from 1980 to 2018, which starts from the historical restarts:

```
nIter0=   322920,
bathyFile       = 'bathy_90x44x15_mod_f64.bin',
hydrogThetaFile = 'templev_90x44x15_mod_f64.bin',
hydrogSaltFile  = 'saltlev_90x44x15_mod_f64.bin',
```

```
pickup.0000324000.data                                             ->
../../CESAM_4cpu_control/hist_run4_1850_200yrs_forcing1850/pickup.000
0324000.data
pickup.0000324000.meta                                             ->
../../CESAM_4cpu_control/hist_run4_1850_200yrs_forcing1850/pickup.000
0324000.meta
pickup_cd.0000324000.data                                          ->
../../CESAM_4cpu_control/hist_run4_1850_200yrs_forcing1850/pickup_cd.
0000324000.data
pickup_cd.0000324000.meta                                          ->
../../CESAM_4cpu_control/hist_run4_1850_200yrs_forcing1850/pickup_cd.
0000324000.meta
```

```
plasim_restart                                                     ->
../../CESAM_4cpu_control/hist_run4_1850_200yrs_forcing1850_r1/plasim_
restart.130
```

8.3. Instruction to run the experiment

1. See 6.3 and 7.3

2. Check if `data` and `puma_namelist` use the same configuration as the historical, from which it is started. It means that these files should differ in different experiments only by runtime parameters and the pickup. If more things differ, then the settings are not the same.

Prepare `data-begin-nudging` that contains the ocean namelist file for the first year of your simulation. Pay attention to check namelist and restarts if you need to resubmit the broken run, such that these files correspond to the year of the simulation from which you really want to start the run. Get these files ready so that if the run breaks, you can quickly restart it again, e.g.:

- `cp data-begin-nudging data`
- `cp ../../CESAM_4cpu_control/hist_run4_1850_200yrs_forcing1850_r1/plasim_restart.130 plasim_restart`

3. The initial pickup for the nudging comes from the historical simulation. Due to model biases historical mean climate is not the same as that of nudged fields. It means that nudging will force the model to come down from a warmer initial state to the state in the nudged fields. It happens within a month. This adjustment may contaminate the 1st year. I tried to replace T and S of the pickups with the observed T and S climatology. However, doing that negatively affects AMOC, resulting in the overturning cell split. The cell recovers within 1 year. Thus, neither of the ways seems optimal, and one needs to treat the 1st year of nudging with caution.

9. PREPARE NUDGING FIELDS

Table overview of the names of the data files:

Ocean Nudging:	Atmosphere Nudging Fields

Produced with `griddata_temp_salt_1d.m`
 	
Binary data
`temp_gecco3.data`
`salt_gecco3.data`
`temp_gecco3_mask.data`
`salt_gecco3_mask.data`
	Binary data `observation_nudg.srv`

Ocean Optimization Fields:	Atmosphere Optimization Fields

Binary data
`temp_initial_f64.data`
`salt_initial_f64.data`
	Binary data `observation.srv`

9.1. Ocean nudging fields in the model format

1. Download the data that you want to nudge CESAM to, e.g., GECCO3 data are available from ICDC, see https://icdc.cen.uni-hamburg.de/en/gecco3.html

2. Get the script `griddata_temp_salt_1d.m` and `griddata_temp_salt_1d_mask.m`

3. Get ready the CESAM grid, namely X, Y, Z and Zl coordinates and `maskCtrlC.data`.

4. Run `griddata_temp_salt_1d.m` and `griddata_temp_salt_1d_mask.m`

   To use the MITgcm functions, e.g., on reading the grid etc, add: `addpath('CESAM_4cpu/MITgcm_c62y/utils/matlab')` to `griddata_temp_salt_1d.m`

5. The data will be written into files with `*.data` ending.

9.2. Atmospheric nudging fields in the model format

Ocean Nudging/Optimization Fields	Atmosphere Nudging Fields
Binary data produced with `griddata_temp_salt_1d.m`	Binary data `observation_nudg.srv`
	Atmosphere Optimization Fields
	Binary data `observation.srv`

STEP 1a: Download atmospheric reanalysis data

Variables that are usually nudged in the atmospheric model are:

- Temperature (Guokun did not nudge this one)
- Vorticity
- Divergence
- Specific humidity
- Surface pressure (Guokun did not nudge this one)

Scripts to download the ERA-Interim reanalysis:

- `run_temp.sh` (from Guokun)
- get_data_ecmwf_temp.py (from ecmwf):

```
Example of get_data_ecmwf_temp.py

#!/usr/bin/python
from ecmwfapi import ECMWFDataServer

server = ECMWFDataServer()

server.retrieve({
      'dataset' : "interim",
      'date'     : "YYYYMM01/to/YYYYMMDE",
      'time'     : "0/6/12/18",
      'step'     : "0",
      'levtype' : "ml",
      'type'     : "an",
      'grid'     : "48",
      'class'    : "ei",
      'param'    : "130.128",
      'levelist' :
"1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19/20/21/22/23/2
4/25/26/27/28/29/30/31/32/33/34/35/36/37/38/39/40/41/42/43/44/
45/46/47/48/49/50/51/52/53/54/55/56/57/58/59/60",
      'target'   : "YYYYMM_tempN48.grib"
      })
```

STEP 1b: Get the data from the pool

ICDC (https://icdc.cen.uni-hamburg.de/) or MPI-M have a repository of the downloaded data. For instance, ERA5 are available from `/pool/data/ERA5`. All necessary variables for nudging are available from this directory.

STEP 2: Interpolate downloaded data on the model's horizontal grid using cdo

Scripts to run on netcdf files:

- `p_uv.sh` # velocity transformed from divergence and vorticity
- `p_data.sh`

Note: When interpolating vorticity and divergence from higher to coarser resolution, the resulting fields show local high values. As a work-around, Guokun transforms vorticity and divergence into horizontal velocity fields (u and v), interpolates u and v and then transforms them back to divergence and vorticity.

STEP 3: Interpolate data on the model's vertical grid using matlab

`/scratch/cen/ifmrs/userid/data/era_interim/matlab_2001_uv_intp`

Scripts:

- `do_interp_p.m`

Output : `era_avg_01.nc`

- Land mask in CESAM is not the same as in ERA-Interim. To fit the land mask use `hinterp_10layer.m`

Output: `era_hintp_01.nc` containing variables u and v

- Script `p_uv.sh` transforms horizontal velocity fields (u and v) back into divergence and vorticity.

Output: `era_hintp_04_dv_gp.nc` containing variables sd and svo

STEP 4: Write data in the model's format to `observations_nudg.srv` using Fortran

Here, important to know is that the model time should match the time of nudging files:

- model running time
- nudging fields time that are explained in section PREPARE NUDGING FIELDS and
- `observations.F90` in `src_2dmap_2dweight`:
  ```
  call ntomin(kstep+1,nmin,nhour,nday,nmonth,nyear)
  call mmdd2yday(kyday,nyear,nmonth,nday)
  nrec=(kyday-1)*4+((nhour)*60+nmin)/360 +1
  itr2=(mod(real(nhour),6.)+real(nmin)/60.)/6.
  itr1=1-itr2  ! for nrec record
  ```
- subroutine `mmdd2yday(kyday,kyear,kmon,kday)`

Scripts:

- `prep_data_trunk_all_yearly_360d.m` (modify to the correct number of time steps)
- `copy_srv_day365.f90` read nc, write binary srv

10. PREPARE FIELDS FOR OPTIMIZATION

10.1. Oceanic data fields

Use `prep_initial.m` to generate `temp_initial_f64.data` and `salt_initial_f64.data.`

10.2. Atmospheric data fields

Prepare fields for the control parameters for data assimilation/cost function optimization.

During optimization, the model will prepare the model state and compare it with observations in `observations.srv`. The data in this file are prepared with

- `prep_obs.m` and
- `copy_srv_day360_45.F90`.
- Weighting fields for observations prepared with `prep_weight.m`

11. Run tstadm algorithm

For a gradient check, run tstadm.

11.1. Instructions to run the experiment

1. After `make tstadm`, check if the model requires a lot of recomputations in `the_main_loop`. If yes, the model takes too long to run, which is not efficient:

   ```
   grep -i recomputation bld_mpi/taf*log | grep -i the_main_loop
   ```

   Currently, there is one warning: `TAF WARNING TAF RECOMPUTATION WARNING CALL_STMT the_main_loop.f:6247 in the_main_loop`.

2. In case restarted after an error do `rm fc.b* g.b* x.b*`
3. `./writexb 6063712 0.` # Here `6063712` represents the total size of the control vector NX. And `0.` means that the initial perturbation is set to 0.
4. `limit stacksize unlimited` # To avoid segmentation fault error
5. `mpirun -np 4 tstadm_relax > tstadm_relax.out &`

12. Optimization for **pseudo observations (TWIN experiments)**

12.1a. Some background for the experiment

Optimization algorithms estimate parameters by minimizing a cost function which measures a quadratic difference of the model simulations to a given set of observations. In practice, parameters are perturbed one by one to evaluate the cost function and to approximate the gradients of the cost function with respect to those parameters (Lyu et al , 2018).

In general terms the cost function can be expressed as

$$J(x) = 1/2(x_B - x)^T B^{-1} (x_B - x) + 1/2 \sum_{t=0}^{\Delta t} (y(t) - H_t^0 x(t))^T E^{-1} (y(t) - H_t^0 x(t))$$

The cost function expression expression differs; depending which method one uses to solve the minimization problem and for which purpose the cost function is used. **The cost function is referred to as the dependent variable. It is a function of input variables, which are referred to as independent variables or control variables. All relevant routines to the treatment of the cost function are located in `pkg/cost`, the routines relevant for the treatment of the control vector are in `pkg/ctrl`.**

From the MITgcm manuals, the routines tree is as below. However, in CESAM some of the parts might have been modified and need to be double-checked:

For the cost function:

```
the_model_main
|
|-- initialise_fixed
|       |
|       |-- packages_readparms
|               |
|               |-- cost_readparms from data.cost
|               o
|
|-- the_main_loop
...     |
        |-- initialise_varia
        |       |
        |       |-- packages_init_variables
        |       |
        |       |-- cost_init
```

```
       |      o
       |-- do iloop = 1,nTimeSteps
       |      |-- forward_step
       |      |-- cost_tile
       |      |      |
       |      |      |-- cost_tracer
       | end do
       |
       |-- cost_final (accumulates the total cost function fc from each contribution and sums
over all tiles (bi,bj))
       o
```

For the control vector::

```
The_model_main
|
|-- initialise_fixed
|      |
|      |-- packages_readparms
|             |
|             |-- ctrl_init                        - initialise control
|             o package
|
|-- ctrl_unpack                                           - unpack control vector
|
|-- adthe_main_loop                                       - forward/adjoint run
|      |
|      |-- initialise_variables
|      |      |
|      |      |-- packages_init_variables
|      |             |
|      |             |-- ctrl_map_ini                     - link init. state and
|      |             o                                    parameters to control
|      |                                                  variables
|      |-- ctrl_map_forcing                        - link forcing fields to
|      …                                                  control variables
|
|-- ctrl_pack                                      - pack control vect
```

The cost function calculations are enabled by adding `cost` and `ctrl ` in `packages.conf`. All cost-specific options are set and control variables are enabled in `ECCO_CPPOPTIONS.h ` In order to save memory, the control variable arrays are read from file and added to the initial fields during the model initialization phase. The adjoint fields which represent the gradient of the cost function w.r.t. the control variables are written to file at the end of the adjoint integration. The files with fields and vectors of the control variables and gradient are generated and initialized in `ctrl_unpack`.

Differentiation w.r.t. the controls starts with adding a perturbation onto the input variable:

- `ctrl_map_init` (initial value sensitivity): temperature and salinity are initialized in `ini_fields.` In `ctrl_map_init` perturbations are added to the control variable. Files generated in the run directory and named as `xx_tr` contain perturbations.
- `ctrl_map_forcing` (forcing sensitivity). This does not work in CESAM because the Plasim model does not see the MITgcm optimized fluxes. Somebody somewhen removed something… Yulia put the option of optimization of fluxes back.
- `ctrl_map_params` (parameter sensitivity).

The control (state) variable x represents a set of parameters of the climate model that we are trying to improve. A state vector at initial time is $x(t)$. $x_B$is a background estimate which comes from the model forecast and is used as another version of observation with filled data gaps, B is the background error covariance matrix, $y(t)$ is the set of observations, H_t^0 is a forward model which predicts observations and E is the observational error covariance matrix.

The control variables are a subset of the model input (initial conditions, boundary conditions, model parameters). The control variable comes from the model and its observational counterpart is y. **In data assimilation, if we try to assimilate e.g., observed temperature fields into the climate model, the control vector and the observational state vector would represent the same variable.** In parameter optimization, the control vector is usually a set of model parameters, which we optimize to find the best fit to the observed temperature field, thus in parameter estimation the control and the observed variables are not the same.

CESAM model is equipped with the adjoint which solves the minimization problem by computing the gradient of the cost function with respect to the control variable.

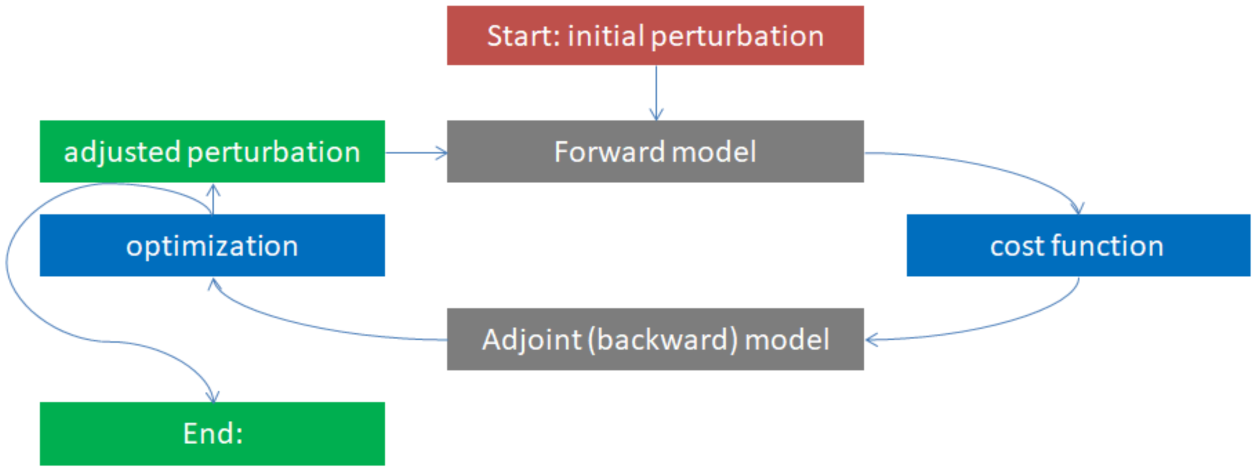
12.1b. Objective

The procedure described below is usually carried out to find model parameters that best agree with observations. However, in the current experiment the observations are perfectly known as well as the parameters. So the purpose is just to test things and, most importantly, test the 1-year assimilation window.

Task	Test 1-year assimilation window. Due to the relaxation procedure implemented by Guokun Lyu, it is possible to run the optimization for 12 months. At the end of this experiment, - perturbation should be optimized - cost function should be reduced - model errors reduced However, this simulation is idealistic and we possess the perfect knowledge about the model parameters and the observations. So here we are just interested in the minimization of the cost function.
Control vector	**6 parameters from the atmosphere:** `tfrc`, (radiative friction) `th2oc`, `acllwr` (longwave radiation) `tswr1`,`tswr2`,`tswr3` (shortwave radiation) **2 parameters from ocean:** initial temperature and salinity
Observations	ocean temperature and salinity (monthly climatology) atmosphere: longwave radiation and shortwave radiation at the surface and the top (176,177,178,179, see `cost-function-definition-atm.dat`)
Experiments	**Step 1:** Forward run to generate observations for nudging of the atmosphere. **Step 2:** Forward run to generate observations for the data assimilation in the ocean and the atmosphere. **Step 3:** Optimization run with relaxation scheme and the assimilation window of 12 months.

Below is a scheme of simulations described in this section. In STEP 1, we will perform the forward model simulation. Usually data assimilation assumes assimilation of some observations into the model with the purpose of, for instance, performing climate predictions. Climate predictions usually begin at some point in time from the observed state of the climate system assimilated into the climate model. In the first experiment, we will not use any actual observations but generate so-called pseudo observations for the ocean and atmospheric mode components. End optimization if criteria is fulfilled. The criteria is that optimization cannot further reduce the cost function (then End:).

Scheme of the experiment 2:



12.2. Instructions to run the experiment

STEP 1: Run the forward model and generate synthetic-observations for nudging

At the end of this simulation, initial conditions ("pseudo observations") for the subsequent experiment will be generated. "Pseudo observations" for temperature and salinity will be in `tbar.0000000000000.data` and `sbar.0000000000000.data`, respectively. They should be renamed to `temp_initial_f64.bin` and `salt_initial_f64.bin` after the run is over. The empty file `observations.srv`, which was created manually before submitting the job, will be overwritten as a result of this simulation, it contains geophysical fields at every timestep of the model. The file `observations.srv` should be renamed to `observations_nudg.srv`, which will be used in the subsequent simulation in STEP 2, where these pseudo observations will be assimilated into the model.

1) `make function_genobs`
   `cp ./bin/function_genobs ./opti_run`
2) `cd opti_run`
3) `vim puma_namelist`, `NCOST = 1` # Write out observations for every timestep, `NCOST=30` (1 day atm). It has to correspond to `nTimeSteps=3` in `data` (1 day ocean);
4) `vim data`, `nTimeSteps = 1083` # Run the model for 361 days. Timestep is 8 hours in this configuration;
5) `cp cost-function-definition-atm.dat_pseudo cost-function-definition-atm.dat`. # This file tells the model which atmospheric variables to write out: 155-divergence, 138-vorticity, 130-air temperature, 133-specific humidity, 134-surface pressure. See Table in Appendix;
6) `vim observations.srv` # Save the empty file. One needs to create an empty file manually;
7) `./writexb 8 0` # Set initial perturbations for eight control variables to 0;
8) `mpirun -np 4 function_genobs>function_genobs.out` # To submit a job module `openmpi` should be loaded. It uses 4 processors.
9) After the CESAM model finishes running, `observations.srv` will be generated. You can check the file with `cdo -sinfo observations.srv`

```
File format : SERVICE  BIGENDIAN
     -1 : Institut Source   T Steptype Levels Num   Points Num Dtype : Parameter ID
      1 : unknown  unknown  v instant   10   1       2048   1  F64  : 138
      2 : unknown  unknown  v instant   10   1       2048   1  F64  : 155
      3 : unknown  unknown  v instant   10   1       2048   1  F64  : 130
      4 : unknown  unknown  v instant   10   1       2048   1  F64  : 133
      5 : unknown  unknown  v instant         1   2        2048   1  F64  : 134
  Grid coordinates :
     1 : generic                : points=2048 (64x32)
   Vertical coordinates :
     1 : generic                : levels=10
                          lev : 1 to 10 by 1
     2 : surface                : levels=1
   Time coordinate :  unlimited steps
  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss  YYYY-MM-DD hh:mm:ss
  2461-01-01 00:48:00  2461-01-01 01:36:00  2461-01-01 02:24:00  2461-01-01 03:12:00
  2461-01-01 04:00:00  2461-01-01 04:48:00  2461-01-01 05:36:00  2461-01-01 06:24:00
  2461-01-01 07:12:00  2461-01-01 08:00:00  2461-01-01 08:48:00  2461-01-01 09:36:00
  2461-01-01 10:24:00  2461-01-01 11:12:00  2461-01-01 12:00:00  2461-01-01 12:48:00
  2461-01-01 13:36:00  2461-01-01 14:24:00  2461-01-01 15:12:00  2461-01-01 16:00:00
  2461-01-01 16:48:00  2461-01-01 17:36:00  2461-01-01 18:24:00  2461-01-01 19:12:00
  2461-01-01 20:00:00  2461-01-01 20:48:00  2461-01-01 21:36:00  2461-01-01 22:24:00
  2461-01-01 23:12:00  2461-01-02 00:00:00  2461-01-02 00:48:00  2461-01-02 01:36:00
  2461-01-02 02:24:00  2461-01-02 03:12:00  2461-01-02 04:00:00  2461-01-02 04:48:00
  2461-01-02 05:36:00  2461-01-02 06:24:00  2461-01-02 07:12:00  2461-01-02 08:00:00
  2461-01-02 08:48:00  2461-01-02 09:36:00  2461-01-02 10:24:00  2461-01-02 11:12:00
  2461-01-02 12:00:00  2461-01-02 12:48:00  2461-01-02 13:36:00  2461-01-02 14:24:00
  2461-01-02 15:12:00  2461-01-02 16:00:00  2461-01-02 16:48:00  2461-01-02 17:36:00
  2461-01-02 18:24:00  2461-01-02 19:12:00  2461-01-02 20:00:00  2461-01-02 20:48:00
  2461-01-02 21:36:00  2461-01-02 22:24:00  2461-01-02 23:12:00  2461-01-03 00:00:00
   ................................................................................
   ................................................................................
   ................................................................................
   ..
  2461-12-30 02:24:00  2461-12-30 03:12:00  2461-12-30 04:00:00  2461-12-30 04:48:00
  2461-12-30 05:36:00  2461-12-30 06:24:00  2461-12-30 07:12:00  2461-12-30 08:00:00
  2461-12-30 08:48:00  2461-12-30 09:36:00  2461-12-30 10:24:00  2461-12-30 11:12:00
  2461-12-30 12:00:00  2461-12-30 12:48:00  2461-12-30 13:36:00  2461-12-30 14:24:00
  2461-12-30 15:12:00  2461-12-30 16:00:00  2461-12-30 16:48:00  2461-12-30 17:36:00
  2461-12-30 18:24:00  2461-12-30 19:12:00  2461-12-30 20:00:00  2461-12-30 20:48:00
  2461-12-30 21:36:00  2461-12-30 22:24:00  2461-12-30 23:12:00  2462-01-01 00:00:00
  2462-01-01 00:48:00  2462-01-01 01:36:00  2462-01-01 02:24:00  2462-01-01 03:12:00
  2462-01-01 04:00:00  2462-01-01 04:48:00  2462-01-01 05:36:00  2462-01-01 06:24:00
  2462-01-01 07:12:00  2462-01-01 08:00:00  2462-01-01 08:48:00  2462-01-01 09:36:00
  2462-01-01 10:24:00  2462-01-01 11:12:00  2462-01-01 12:00:00  2462-01-01 12:48:00
  2462-01-01 13:36:00  2462-01-01 14:24:00  2462-01-01 15:12:00  2462-01-01 16:00:00
  2462-01-01 16:48:00  2462-01-01 17:36:00  2462-01-01 18:24:00  2462-01-01 19:12:00
  2462-01-01 20:00:00  2462-01-01 20:48:00  2462-01-01 21:36:00  2462-01-01 22:24:00
  2462-01-01 23:12:00  2462-01-02 00:00:00
cdo    sinfo: Processed 5 variables over 10830 timesteps [3.24s 89MB].
```

10) `mv observations.srv observations_nudg.srv` # This data file contains atmospheric fields for nudging for 5 variables.

STEP 2: Run the forward model to generate synthetic-observations for assimilation

1) `vim src_2dmap_2dweight/Makefile`, search for

```
plasim_genobs.f90: plasim.F90
      $(CPP) $(CPPOPTS) -DGENOBS  $< > $@
```

And add "`-DNUDGING`" after "`-DGENOBS`"

2) `make function`
3) `make function_genobs`
   `cp ./bin/function opti_run/function_relax`
   `cp ./bin/function_genobs opti_run/function_genobs_relax`
4) `cd opti_run`
5) `vim puma_namelist`, `NCOST  = 10800` # Write out obs as annual mean;
6) `vim data`, `nTimeSteps = 1080` # Run the model for 360 days;
7) `cp cost-function-definition-atm.dat_01 cost-function-definition-atm.dat` # This file tell the model which atm variables are written-out and used in the computing of the cost function, 131-u wind, 130-air temperature, 133-specific humidity, 134-surface pressure, 176-179 radiation flux at the top and at the surface, 278-net top radiation flux, 164-total cloud cover;
8) `vim observations.srv` # Save the empty file;
9) `mpirun -np 4 function_genobs_relax>function_genobs_relax.out`
10) start matlab (`matlab -nodesktop`), execute `prep_initial.m`, generate synthetic ocean temperature and salinity initial conditions used in the following simulation.

```
-rw-r--r-- 1 userid ifmrs      5702400 Mar 20 19:30 temp_initial_f64.data
-rw-r--r-- 1 userid ifmrs      5702400 Mar 20 19:30 salt_initial_f64.data
```

STEP 3: Run the optimization algorithm

Latest at this state you should generate the ssh key pair to enable access to the TAF server (see section TAF Licence). Test the access by running `./staf -test`.

During the optimization step, the data files will be used and updated with the following names:

- `fc.*` for total (ocean+atmos) cost function,
- `g.*` for gradient of cost function per parameter and iteration and
- `x.*` for parameter perturbation per parameter and iteration.
    - In `x.b`, there are 14 2-D fields (32x64). The atmospheric parameter increment is stored in `data_atm(latitude,longitude,parameter,iter)`.
    - The increments for oceanic parameters such as initial temperature and salinity, `kapgm` and `kapredi` are stored in `data5k(latitude,longitude,depth)`. The `kapgm` is the parameter to represent the effect of geostrophic eddies and `kapredi` is for the Redi scheme that diffuses tracers along isopycnals.
    - `wtheta`, `wsalt`, `wkapgm`, `wkapredi` are weighting factors in `1/sigma`^{2}

Then the initial parameter perturbations are read from the file `x.b`. Otherwise, parameter perturbation is set by `initmod` (it sets all perturbation norms to `-10%`). At the end of the optimization, the new defined perturbation for the parameters are written in `x.b` and copied to `x.b_00n`.

1) `make adm` # Run this command in `CESAM_4cpu_optiera5`. This step gives an error (TAF ERROR func cannot save file status : _cont_33,_ptr_33,_ptr_35,_unit_33). However, the generation of the adjoint code is not affected. At this step, code is sent to TAF and adjoint code is generated in bld dir.
2) `cp ./omit_ad/* ./bld_mpi`
3) `gedit ./bld_mpi/func_ad.f`
    a) add in line 65 `include 'mpif.h'` after `implicit none` and
    b) change `mpi_comm_rank( 91,nrank,iers )` to `mpi_comm_rank(MPI_COMM_WORLD,nrank,iers )`
4) `make tstadm` # executable to verify ADM
    a) `cp ./bin/tstadm opti_run/tstadm_relax`
5) `make opti`
6) `cp ./bin/opti ./opti_run/opti`
7) `cd opti_run`
    a) `./writexb 8 0.1` # Set initial perturbations
    b) `./opti>opti_test.out&` # it took about 7 hours to finish
8) `figure_xb.m` can be used to read `x.b`, `g.b`, `fc.b`.

12.3. Result of the exercise

At the end of this simulation, 15 files for each of the following are generated:

- `fc.b` for total cost function,
- `x.b` for parameter perturbation per parameter and iteration and
- `g.b` for gradient of cost function per parameter and iteration.

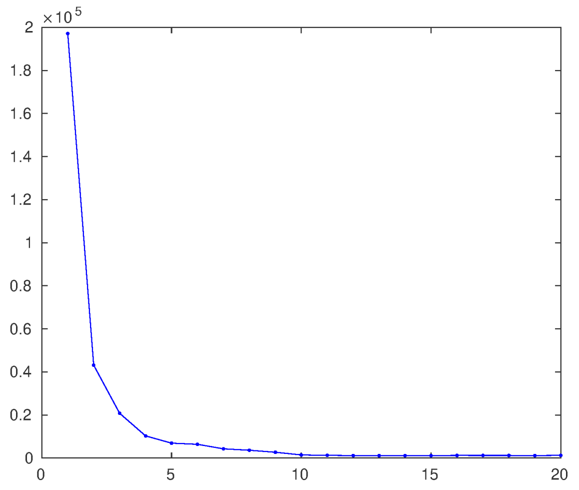


Figure: Cost function of the first optimization experiment. On the X-axis number of iterations. On the Y-axis the cost function value.

13. Optimization for **ERA5-reanalysis**

13.1. Objective

Test ERA5 data assimilation for the year 1980 with the 1-year assimilation window.

Task	Test 1-year assimilation window. Due to a relaxation procedure implemented by Guokun Lyu, it is possible to run the optimization for 12 months. At the end of this experiment, - perturbation should be optimized. - cost function should be reduced. - model errors reduced.
Control vector	**14 parameters from the atmosphere:** ● `th2ocg, acllwrg` (longwave radiation) ● `tswr1g, tswr2g, tswr3g` (shortwave radiation) ● `tpofmtg` (mean transmissivity in layer) ● `albgmaxg` (max. albedo for glaciers) ● `rcritg` (critical relative humidity for cloud formation) ● `rcritg` (critical relative humidity for cloud formation) ● `rcritg` (critical relative humidity for cloud formation) ● `rcritg` (critical relative humidity for cloud formation) ● **`alpha1g`** ● **`beta1g`** ● **`gamma1g`** (is set to give a potential temperature in the bulk formula) This set of parameters (14 out 36; see `#elif (NX-NX_OCEAN)==36` in `./src_2dmap_2dweight/mo_mapping.F90`) is chosen for the optimization due to their large sensitivity to ocean T&S, top solar radiation, top longwave radiation, surface solar radiation and surface longwave radiation). **4 parameters from ocean:** ● initial temperature ● initial salinity ● mixing terms
Observations	**For nudging:** ● **Atmosphere:** ERA5 year 1980 **For optimization:** ● **ocean:** temperature and salinity (WOA monthly climatology) ● **atmosphere:** longwave radiation and shortwave radiation at the surface and the top (176,177,178,179, see

	`cost-function-definition-atm.dat`)
All experiments	**Step 1:** Nudging the coupled model toward ERA5. **Step 2:** Optimization run with the assimilation window of 12 months. The assimilation run is as long as the `observations.srv` data is! (360 days, `1080` ocean time steps in specified `data` configuration file)

13.2. Instructions to run the experiment

STEP 1a: Preparation.

1. Create a directory named e.g., `CESAM_4cpu_opti`. Copy in it the source code and run files as in the steps done for the previous experiments.

2. Make sure that the nudging option is switched on by checking if the flag `-DNUDGING` in `src_2dmap_2dweight/Makefile` for `plasim.f90` is active (see Table above).

3. Create `opti_era` directory for the new experiment. Copy necessary files (restarts, namelist, etc.) from the default run directory in `opti_era`.

4. Copy `function` executable from `bin` to the `opti_era` directory. Rename `function` to `function_relax`.

5. Copy the last pickup files from the control run here. Edit `data` file:
    - For one year run `nTimeSteps = 1080`.
    - e.g., `nIter0 = 327624`, if the name of the pickup file is `pickup*.0000327624.001.001.data`

   It makes sense to pick initial conditions carefully: If the model initial state is too far from the observations, all what the optimization is doing then is to bring the model to the observed climatology. Therefore, it makes sense to start with a good initial state, e.g, from the obs climatology itself. See FAQ for how to read/write the pickup file.

6. Data are prepared following sections 7 and 8. Copy or link the data to the `run` directory, e.g., `era_day365.srv_1980` to `observations_nudg.srv`. (`ln -sf` is better, avoiding duplicate the data!)

STEP 1b: Nudge the coupled model toward ERA5. Or omit STEP 1b and start from the spin-up and go directly to STEP 2.

We perform the nudging run, in which CESAM is pushed toward the ERA5 state. The nudging run begins from the 300-year control run. STEP 1 is the second spinup phase after the 300-year control run. Here, we want to bring CESAM closer to the ERA5 state.

7. The model is designed for the optimization experiments, thus make sure that the cost function calculation in `puma_namelist` is switched off `NCOST=0.`

8. Nudging at all vertical levels for divergence, vorticity and humidity (10*1.0) and no nudging for pressure and temperature in this experiment. Multiplication by 1 means 1-day nudging:

```
tnudg = 10*0.0 # air temperature
dnudg = 10*1.0 # divergence
qnudg = 10*1.0 # specific humidity
znudg = 10*1.0 # vorticity
pnudg = 0.     # surface pressure
```

9. Run function using `mpirun -np 4 function_relax > function_relax.out &`

STEP 2: Run the optimization algorithm

For ocean data assimilation experiments generate ocean temperature and salinity initial conditions from the ocean profile data. For now, use the prepared ones `temp_initial_f64.data` and `salt_initial_f64.data`, they represent ocean "observations" for temperature and salinity containing climatology from WOA-13.

`cost-function-definition-atm.dat` # This file tells the model which atmospheric variables are written-out and used in the computing of the cost function, e.g., 131-u wind, 130-air temperature, 133-specific humidity, 134-surface pressure, 176-179 radiation flux at the top and at the surface, 278-net top radiation flux, 164-total cloud cover etc.

`observations.srv` # This data file contains ERA5 data for 1980; it is used for assimilation/optimization (10-day mean values, `cdo timselmean,10 -daymean observations_nudg.srv observations.srv`).

`observations_nudg.srv` # This data file contains ERA5 data for 1980; it is used by the nudging scheme (hourly values).

`weight.txt` # Initial estimate of the observational error. If the model has large errors more weight will be given to observations.

`mask.b` # Tells the opti program whether a grid cell is land. For atmosphere variables, all mask values equal 1. For ocean parameters, land grid cells are set to 0. See `/data/cen/qfs10/uXXXXXX/opti_150y/prep_mask.m` The order of `mask.b` should be the same as `x.b`. mask.b indicates whether a point in x.b is land or ocean. Land points will be excluded from optimization since it is 0 or NaN.

During the optimization step, the initial data files will be used and updated in the files with the following names:

- `fc.*` for total (ocean+atmos) cost function,
- `x.*` for parameter perturbation per parameter and iteration and
- `g.*` for gradient of the cost function per parameter and iteration.

Then the initial parameter perturbation is read from the file `x.b`. If you do not have an initial perturbation, it should be set by `initmod` (it sets all perturbation norms to -10%). At the end of the optimization run, the correction terms for the optimized parameters are written in `x.b` and copied to `x.b_00n`.

Recipe to run the experiment:

1. `make adm` # This step gives an error. However, the generation of the adjoint code is not affected. At this step, code is sent to TAF and adjoint code is generated in `bld_mpi` directory. An executable is not generated yet. With `make tstadm` an executable can be created.

2. `cp ./omit_s15_2dmap/* ./bld_mpi`

3. `gedit ./bld_mpi/func_ad.f`
   - add in line 65 `include 'mpif.h'` after **`implicit none`** and
   - change in line 406 `mpi_comm_rank( 91,nrank,iers )` to `mpi_comm_rank( MPI_COMM_WORLD,nrank,iers )`

```
5. make tstadm
```
# executable to verify ADM will be created in `bld_mpi`

```
6. mkdir opti_run
```
# where you will run the experiment

```
cp ./bin/tstadm opti_run/tstadm_relax
```

```
7. make opti
```

```
8. cp ./bin/opti ./opti_run/opti
```

9. `cd opti_run`
    - In case restarted after an error do `rm fc.b* g.b* x.b*`
    - Set perturbation and initiate optimization
      `./writexb 266272 0.` # 266272 the size of the control in Guokun's experiments. Set initial perturbations in x.b
    - Prepare file `weight.txt` with observational errors using `prep_weight.m`
    - To avoid stack overflow error-message: `limit stacksize unlimited`
    - Run the experiment `./opti > opti_test.out &` # it took about 7 hours to finish
10. Check the correction term with `figure_xb.m` It can be used to read `x.b, g.b, fc.b.`

STEP 3: Data assimilation product

1. Copy `x.b_00[last-1]` to `x.b`
2. Enable output in `puma_namelist: NOUTPUT=1`
3. Set `tnudg, dnudg, znudg, pnudg` and `qnudg=0`. Or run with nudging on, if it was used for regularization during the assimilation.
4. Run `function_relax: mpirun -np 4 function_relax > function_relax.out &`

13.3. Result of the exercise

At the end of this simulation, files for each of the following are generated:
- `fc.b` for total cost function,
- `x.b` for parameter perturbation per parameter and iteration.

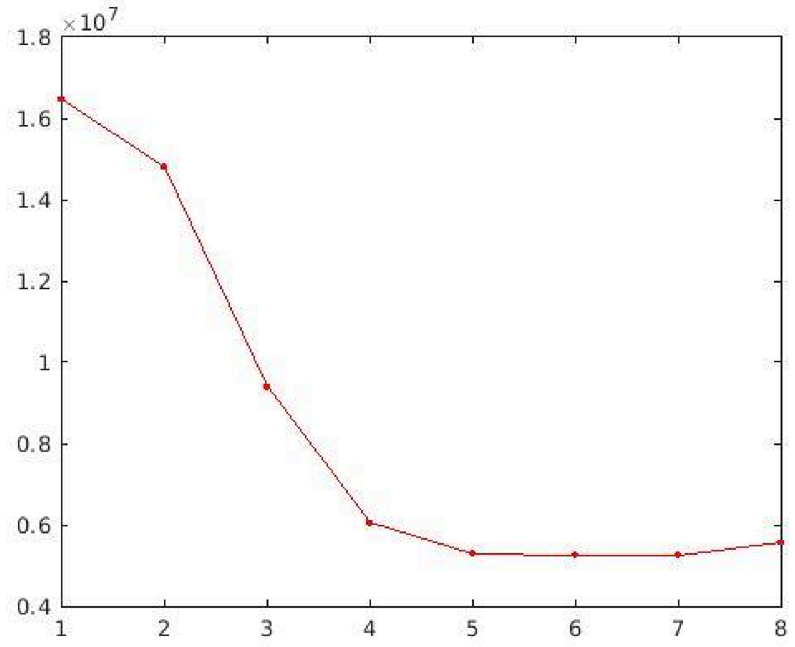


Figure: Cost function of the first optimization experiment. On the X-axis number of iterations. On the Y-axis the cost function value. x.b_006 (iteration 7 on the plot) represents the minimum of the cost function. The cost function is reduced by 68 %.

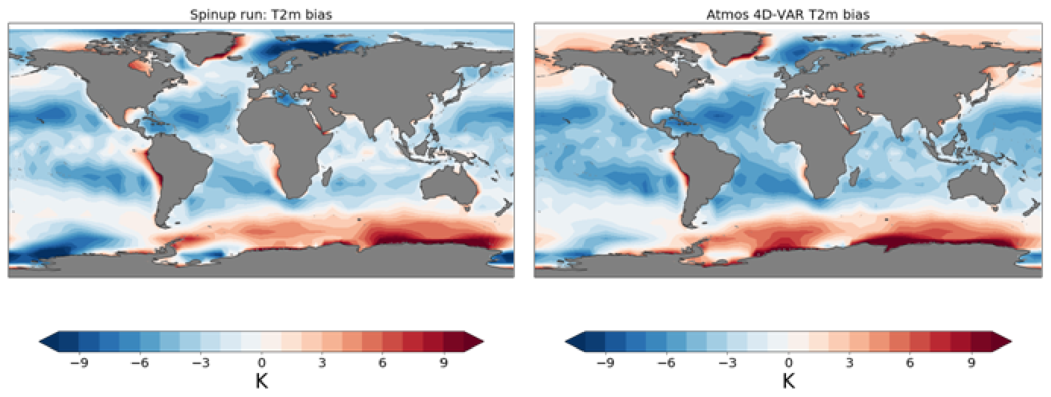


Figure: Bias of the 2-meter temperature for 1980 from the spinup run: (left) and (right) optimization run with the estimated perturbation which gives the best fit to observations and prior information. The bias is calculated with respect to the ERA5 data.

14. Assimilation of EN4 profiles and ERA5

14.1. Objective

Assimilate EN4 and ERA5 data for Earth System Reanalysis.

14.2. Instructions to run the experiment

STEP 1: Preparation.

1. Create a directory named e.g., `CESAM_4cpu_opti`. Copy in it the source code and run files as in the steps done for the previous experiments.

2. Make sure that the nudging option is switched on by checking if the flag `-DNUDGING` in `src_2dmap_2dweight/Makefile` for `plasim.f90` is active.

3. Create `run_opti_coupled` directory for the new experiment. Copy necessary files (restarts, namelist, etc.) from the default run directory in `opti_coupled`.

4. Copy `function` executable from `bin` to the `run_opti_coupled` directory. Rename `function` to `function_relax`.

5. Copy the last pickup files from the historical simulation to `run_opti_coupled`. Edit `data` file:
    - For one year run `nTimeSteps = 1080`.
      e.g., `nIter0 = 1`, if the name of the pickup file is `pickup*.0000000001.001.001.data`
    - `cp plasim_restart.130 plasim_restart`

   Note: It makes sense to pick initial conditions carefully: If the model initial state is too far from the observations, all what the optimization is doing then is to bring the model to the observed climatology. Therefore, it makes sense to start from a good initial state that is close to observations. But if one just uses climatology from observations as a starting point, it might not necessarily be a state that is consistent with the model. For instance, the model needs a year to get AMOC of the right shape. See FAQ for how to read/write the pickup file.

6. Optimization and nudging data are prepared following sections 8 and 9. Copy or link the data to `run_opti_coupled`, e.g., `era_day365.srv_1980` to `observations_nudg.srv`. (`ln -sf` is better, avoiding duplicating the data!)

STEP 2: Run the optimization algorithm

For ocean data assimilation experiments generate ocean temperature and salinity initial conditions from the ocean profile data. For now, use the prepared ones `temp_initial_f64.data` and `salt_initial_f64.data`, they represent ocean "observations" for temperature and salinity containing climatology from WOA-13.

`cost-function-definition-atm.dat` # This file tells the model which atmospheric variables are written-out and used in the computing of the cost function, e.g., 131-u wind, 130-air temperature, 133-specific humidity, 134-surface pressure, 176-179 radiation flux at the top and at the surface, 278-net top radiation flux, 164-total cloud cover etc.

`observations.srv` # This data file contains ERA5 data for 1980; it is used for assimilation/optimization (10-day mean values, `cdo timselmean,10 -daymean … observations.srv`). These data are not the same as `observations_nudg.srv`

`observations_nudg.srv` # This data file contains ERA5 data for 1980; it is used by the nudging scheme (hourly values).

`weight.txt` # Initial estimate of the observational error. If the model has large errors more weight will be given to observations.

`mask.b` # Tells the opti program whether or not a grid-point is land. For atmosphere variables, all mask values equal 1. For ocean parameters, land =0. See `/data/cen/qfs10/uXXXXXX/opti_150y/prep_mask.m` The order of `mask.b` should be the same as `x.b`. `mask.b` indicates whether a point in `x.b` is land or ocean. Land points will be excluded from optimization since it is 0 or NaN.

During the optimization step, the initial data files will be used and updated in the files with the following names:

- `fc.*` for total (ocean+atmos) cost function,
- `x.*` for parameter perturbation per parameter and iteration and
- `g.*` for gradient of the cost function per parameter and iteration.

Then the initial parameter perturbation is read from the file `x.b`. If you do not have an initial perturbation, it should be set by `initmod` (it sets all perturbation norms to -10%). At the end of the optimization run, the correction terms for the optimized parameters are written in `x.b` and copied to `x.b_00n`.

Recipe to run the experiment:

4. `make adm` # This step gives an error. However, the generation of the adjoint code is not affected. At this step, code is sent to TAF and adjoint code is generated in `bld_mpi` directory. An executable is not generated yet. With `make tstadm` an executable can be created.

5. `cp ./omit_s15_2dmap/* ./bld_mpi`

6. `gedit ./bld_mpi/func_ad.f`
    - add in line 65 `include 'mpif.h'` after **`implicit none`** and
    - change in line 433 `mpi_comm_rank( 91,nrank,iers )` to `mpi_comm_rank( MPI_COMM_WORLD,nrank,iers )`

5. `make tstadm` # executable to verify ADM will be created in `bld_mpi`

6. `mkdir opti_run` # where you will run the experiment
`cp ./bin/tstadm opti_run/tstadm_relax`

7. `make opti`

8. `cp ./bin/opti ./opti_run/opti`

9. `cd opti_run`
    - In case restarted after an error do `rm fc.b* g.b* x.b*`

    - Set perturbation and initiate optimization
For 1 year: `./writexb 6063712 0.` # Here `6063712` represents the total size of the control vector NX. And `0.` means that the initial perturbation is set to 0.
For 39 years: `./writexb 222659872 0.` # Here `222659872` represents the total size of the control vector NX. And `0.` means that the initial perturbation is set to 0.

    - Prepare file `weight.txt` with observational errors using `prep_weight.m`

    - To avoid stack overflow error-message: `limit stacksize unlimited`

    - Run the experiment `./opti > opti_test.out &` # it took about 7 hours to finish

10. Check the correction term with `figure_xb.m` It can be used to read `x.b, g.b, fc.b.`

STEP 3: Data assimilation product

5. Copy `x.b_00[last-1]` to `x.b`

6. `pickup` files, `plasim_restart` and `data` file are the same as for optimization

7. `puma_namelist` is not the same as for optimization. Enable output in `puma_namelist: NOUTPUT=1` and set `tnudg, dnudg, znudg, pnudg and qnudg=0`.

8. Run `function_relax` as usually
```
mpirun -np 4 function_relax > function_relax.out &
```

STEP 4: Cycle through optimization 5-10 times to get a good starting point for 1980 - the start of the optimization.

Cycle 1:

From the historical run with replaced T and S in the pickup. This replacement leads to weird AMOC. Therefore a spin-up in the form of this cycling is needed.

The plasim_restart comes from the historical plasim_restart.130

Cycle 2:

From cycle 1 last iteration take pickup and plasim_status

Cycle 3:

From cycle 2 last iteration take pickup and plasim_status

Cycle N:

From cycle N-1 last iteration take pickup and plasim_status

15. Historical simulation

15.1. Objective

Obtain a historical simulation, which serves as a reference for the runs with initialization (climate predictions).

What one can do with this type of experiments

- Long-term historical simulations are often used for climate-change studies and as a benchmark to evaluate added-value from initialized climate predictions.
- This setup will be used to run climate predictions. From the assimilation run, we will start ensembles of decadal climate predictions.

15.2. CMIP6 external forcing for historical run

Forcing datasets: /pool/data/ECHAM6/input/r0008/

- CO2 emissions and aerosols
- Solar radiation
- Ozone

Variable name in puma_namelist	Meaning	Source file
CO2 = 360.0000	CO2 concentration [ppmv]	**Mistral:** /pool/data/ECHAM6/input/r0008/greenhouse_ssp245.nc **Thunder:** /scratch/cen/ifmrs/uXXXXXX/OBS/external_forcing/cmip6
GSOL0 = 1365.0000	Solar constant [W/m2]	**Mistral:** /pool/data/ECHAM6/input/r0008/solar_irradiance/swflux_14band_*nc
NO3 = 2	Switch for ozone (0 = off, 1 = idealized distribution, 2 = externally prescribed)	**Mistral:** /pool/data/ECHAM6/input/r0008/T63/ozone/T63_ozone_ssp245_*.nc

	If NO3 is set to 2, the ozone distribution is read from surface.txt.	

Appendix

Useful commands

- To check whether the model is running `top -u u24*****`
- If you are running "`opti`", you can also use `grep 'fc =' opti_relax.out`

Table 1: “make” Options

```
make install   # create required directories
make function_genobs     # generate pseudo observations
make eraobs    # generate observations from ERA-40 data
make plasim.x  # generate plasim executable
make plasim.x.f90 : generate monolithic plasim source
make cpl.obj   # generate monolithic plasim_all.o
make function  # executable to evaluate function
make runfunction  : evaluate function
make function3 # executable to evaluate function 3 times
make runfunction3 : evaluate function 3 times
make tlm       # generate tangent-linear code (TLM)
make tsttlm    # executable to verify TLM
make runtsttlm # verify TLM
make adm       # generate adjoint code (ADM)
make tstadm    # executable to verify ADM
make runtstadm # verify ADM
make clean     # clean up leaving TAF-generated files and
executables untouched
make scratch   # clean up as much as possible
```

Table 2: Plasim output

To see all the variables of Plasim: `./burn7 -c`

Code	Variable	Name, units
Atmosphere		

130	ta	air temperature [K]
131	ua	U-velocity [m/s]
132	va	V-velocity [m/s]
133	hus	specific humidity [kg/kg]
134	ps	surface pressure
135	wap	Vertical_air_velocity [Pa s-1]
138	zeta	vorticity [1/s]
139	ts	surface_temperature K
142	prl	lwe_of_large_scale_precipitatio m s-1
143	prc	convective_precipitation_rate m s-1
146	hfss	surface_sensible_heat_flux W m-2
147	hfls	surface_latent_heat_flux W m-2
155	d	divergence [1/s]
162	cl	cloud_area_fraction_in_layer 1
164	clt	total cloud cover [frac]
167	tas	air_temperature_2m K
176	rss	surface solar radiation [W/m^2]
177	rls	surface thermal radiation [W/m^2]
178	rst	top solar radiation [W/m^2]
179	rlut	top thermal radiation [W/m^2]
278	flpr	net top radiation flux
182	evap	lwe_of_water_evaporation m s-1
204	ssru	surface_solar_radiation_upward W m-2
205	stru	surface_thermal_radiation_upwar W m-2

Table 3: MITgcm output

(also see http://mitgcm.org/public/r2_manual/latest/online_documents/node269.html and http://mitgcm.org/sealion/code_reference/vdb/code/1407.htm)

Variable	Units	Levels	Description
ETAtave	m	1	Surface Height Anomaly
Eta2tave	m^2		Square of Surface Height Anomaly
Ttave	C	Nr	potential temperature
Stave	psu	Nr	salinity
uVeltave	m/s	Nr	U-Velocity
vVeltave	m/s	Nr	V-Velocity
wVeltave	m/s	Nr	Vertical Velocity
UTtave	degC m/s		Zonal Transport of Potenial Temperature
VTtave	degC m/s		Meridional Transport of Potential Temperature
WTtave	degC m/s		Vertical Transport of Potential Temperature
UStave	psu m/s		Zonal Transport of Salinity
VStave	psu m/s		Meridional Transport of Salinity
WStave	psu m/s		Vertical Transport of Salinity
TTtave	degC^2		Squared Potential

			Temperature
UUtave	m^2/s^2		Zonal Transport of Zonal Momentum
VVtave	m^2/s^2		Zonal Transport of Zonal Momentum
UVtave	m^2/s^2		Product of meridional and zonal velocity
Tdiftave			vertical diffusion flux of temperature
PhHytave	m^2/s^2		Hydrostatic Pressure Pot.(p/rho) Anomaly
PHLtave	m^2/s^2		Bottom Pressure Pot.(p/rho) Anomaly
PHL2tave	m^4/s^4		Square of Hyd. Pressure Pot.(p/rho) Anomaly
Convtave	none [0-1]		Convective Adjustment Index
uFluxtave	N/m^2		surface zonal momentum flux, positive -> increase u
vFluxtave	N/m^2		surface meridional momentum flux, positive -> increase v
tFluxtave	W/m^2		net surface heat flux (>0 for increase in theta)
sFluxtave	g/m^2/s		total salt flux (match salt-content variations), >0 increases salt

Note: "tave" stands for time average.

`prep_mask.m` generates `mask.b`

```
nx=6063712 ;
mask=zeros([1 nx]);
mask(1:14*32*64)=1; %14params,ny,nx of Plasim


T=nc_varget(['../CESAM_4cpu_nudging/run_nudging/1980/DATAO/' ...
            'tave.0000324000.glob.nc'],'Ttave');


Tmean=squeeze(mean(T,1));
Tmean(Tmean==0)=nan;
Tmean=permute(Tmean,[3 2 1]);

nrec=14 * 32 * 64
%14 atmospheric parameters,ny,nx of Plasim
for t=1:4
      for k=1:15
      for j=1:size(Tmean,2)
            for i=1:size(Tmean,1)
            nrec=nrec+1;
            if(~isnan(Tmean(i,j,k)))
                  mask(nrec)=1;
            else
                  mask(nrec)=0;
            end
            end
      end
      end
end
nrec=(14 * 32 * 64) + (44 * 90 * 15 * 4)
%14 atmospheric parameters,ny,nx of Plasim + 4 variables,nx,ny,nz of MITgcm
for it=1:366
      for ivar=1:4
            for j=1:size(Tmean,2)
            for i=1:size(Tmean,1)
            nrec=nrec+1;
            if(~isnan(squeeze(Tmean(i,j,1))))
                  mask(nrec)=1;
            else
                  mask(nrec)=0;
            end
            end
            end
      end
end
%14 atmospheric parameters,ny,nx of Plasim + 4 variables,nx,ny,nz of MITgcm
 + 4 fluxes,nx,ny,nt of MITgcm
```

```
fid=fopen('./mask.b','w','ieee-be');
fwrite(fid,nx*8,'int32');
fwrite(fid,mask(1:end),'float64');
fwrite(fid,nx*8,'int32');
fclose(fid);
clear hr1;
```

FAQ:

- How to reset the date?

To reset the time of the ocean output make changes in the file `data.cal`:

```
#
# ******************
# Calendar Parameters
# ******************
 &CAL_NML
 TheCalendar='gregorian',
# TheCalendar='model',
 startDate_1=16800101,
 startDate_2=000000,
 &
```

- How do I know that the model uses 365 days calendar?

To set the calendar modify in `data` and `puma_namelist` the following parameters, respectively:

For the Gregorian calendar:
`N_DAYS_PER_YEAR = 365`
`TheCalendar='gregorian'`

For the 360_day calendar:
`N_DAYS_PER_YEAR = 360`
`TheCalendar='model'.`

- ## Why are there more time steps in the output than calculated?

  The write-out frequency in the ocean model is every 30 days. CESAM works best with the 360-day calendar.

- ## How to calculate number of time steps for the data namelist file?

  To make sure to include leap years for the time steps calculation in the case of the gregorian calendar you can use `matlab` command, e.g., `(datenum(1979,12,31)-datenum(1680,1,1)+1)*3`.

- ## Is there available space for running the model?

  ```
  /usr/lpp/mmfs/bin/mmlsquota -j ifmrs gpfs04 --block-size T
  ```

- ## Do I always create a new run directory for new experiments?

  Yes, make a new run directory for a new experiment.

  Moreover, make a copy of the source code of `src_2dmap_2dweight` and `MITgcm_c62y/verification/adplethora/code_t21p_forward`. This is needed because you might need to change some of the source codes.

  The model version is specified in `P_CODE` and `P_PLASIMDIR` in the `Makefile`.

  One can also make a new directory in the `run` directory (e.g., `expt`) for each experiment and store the corresponding ocean and atmosphere outputs and pickup files in there.

- ## If I want to run some control simulations from a cold or warm start, what is the typical setup?

  For the cold start:

  `nIter0=0` in `data`.
  `surface.txt` is the start file for the atmosphere and for the ocean:
  `hydrogThetaFile = 'templev_90x44x15.bin',`

```
hydrogSaltFile  = 'saltlev_90x44x15.bin',
```

For the warm start:

You need initial conditions for the ocean and the atmosphere (e.g., ocean: `pickup.0000432000.data` and atmosphere: `plasim_restart`). You need to change `nIter0=432000` to define the start step of the ocean model.

- ## Where to specify boundary conditions such as external forcing?

CO2 and solar const. can be set via namelist (e.g., by updating the namelist in the run_script every year). O3 is (default) only given by an idealized distribution. However, this distribution can also be modified by namelist parameters or you may provide an external field. Aerosols are, so far, not included in Plasim.

In `puma_namelist`

```
CO2   =   360.0000
GSOL0 =  1365.0000
NO3   = 2
```
#Switch for ozone (0 = off,1 = idealized distribution, 2 = externally prescribed). If NO3 is setto 2, the ozone distribution is read from `surface.txt`.

- ## Where is model output generated? Is it in the netcdf format?

The results will be generated in the directory `run`. One can make a new directory (expt) for each experiment and put corresponding results in there. Ocean model output is in the netcdf format, in `mnc_test_0001-0004`. You need to set `NOUTPUT=1` to enable atmosphere model output, the atmospheric output is stored in `puma_output`. The atmospheric output can be converted to netcdf format with `burn_opti.sh`. You can modify `all_pl.nl` or `all_sigma.nl` to define which atmospheric states to be written out.

Content of `all_pl.nl:`

```
HTYPE  = Grig
VTYPE  = P
CODE =
182,135,131,132,130,134,133,138,155,178,179,146,147,176,177,20
4,205,142,143,164,162,129
hPa = 10,50,100,150,200,250,300,400,500,600,700,850,1000
NETCDF = 1
MEAN   = 1
```

Content of `burn_opti.sh`

```
./burn7 < all_pl.nl > test.out  -d ./puma_output ./puma_out.nc
```

To see all the variables of Plasim: `./burn7 -c`

- ## How to get the global ocean output?

The output is split in 4 regions and stored in 4 directories because 4 nodes are used for running the model. One needs to specify in the namelist that the output should be stored by one node. As a work around one can merge the regions using: [`gluemncbig.x`](gluemncbig.x) (the one which actually glues data) and **`glueexbig.job`** (the script by Silke Schubert, which executes gluemncbig.x for large amounts of data). Execute the script as `csh glueexbig.job.`

Alternatively, for matlab users:

```
addpath('/opt/cen/sw/sw/common/matlab/toolbox/mexcdf/netcdf_to
olbox/')
addpath('/opt/cen/sw/sw/common/matlab/toolbox/mexcdf/mexnc/')
addpath('/opt/cen/sw/sw/common/matlab/toolbox/mexcdf/snctools/
')

lat=zeros([1 44]);
yr=1;
filename=dir(['./mnc_all/tave*.nc']);

for ti=1:length(filename) % 4 output files
     lat1=nc_varget(['./mnc_all/',filename(ti).name],'Y');
     lat(11*(ti-1)+1:11*ti)=lat1;
     lon=nc_varget(['./mnc_all/',filename(ti).name],'X');
     depth=nc_varget(['./mnc_all/',filename(ti).name],'Z');
```

```
    temp1=nc_varget(['./mnc_all/',filename(ti).name],'Ttave');
     tempa((yr-1)*120+1:yr*120,:,11*(ti-1)+1:11*ti,:)=temp1(25
     :end,:,:,:);
end
tempa(tempa==0)=nan;
```

For non frequent matlab users, `nc_varget` is not part of the standard matlab package installation. You need to add a path to this package, which is */opt/cen/sw/sw/common/matlab/toolbox/mexcdf/* or */opt/cen/sw/sw/common/matlab/toolbox/mexcdf/netcdf_toolbox/.* You can do this by setting path and store the '`pathdef.m`' file manually from matlabroot or userpath in an arbitrary directory. To use the copied '`pathdef.m`' file at initialization, create a MATLAB file titled '`startup.m`' and place it in the '`$USERPATH`' directory.

- Are there any routines for post processing?

    - The data can be converted to netcdf format with `burn_opti.sh`
    - See also gluing ocean output.

- To run the model in the background mode put "&" at the end of the command

```
mpirun -np 4 function > function.out &
```

- How to switch on/off atmospheric nudging?

In `/scratch/cen/ifmrs/userid/CESAM_4cpu/src_2dmap_2dweight/Makefile`:

```
plasim.f90: plasim.F90
$(CPP) $(CPPOPTS)   $< > $@                           # nuding is off
#$(CPP) $(CPPOPTS) -DNUDGING   $< > $@          # nudging is on
```

- How apart from changing the `namelist` to change model settings to allow for running a long simulation, e.g. 300 years?

To allow for running a long simulation, e.g. 300 years, change parameters in two model routines:

1. `MITgcm_c62y/pkg/ecco/ecco_cost.h` to allow a 300 years forward run (365*301)

```
PARAMETER ( maxNumDays = 109865 )
```

2. Check your Makefile, which mitgcm source code your setup uses, e.g., if `code_t21p_forward_op_2datm` then update `MITgcm_c62y/verification/adplethora/code_t21p_forward_op_2datm/tamc.h`

```
      parameter( nchklev_1   =   3 )
      integer    nchklev_2
      parameter( nchklev_2   =  109580  )
```

After these changes are done you need to recompile the model:

```
> make scratch
> make function
```

When is the optimization run considered to be successful?

The optimization run is successful if the cost function gets reduced by more than 50 %.

How to resubmit the optimization run?

Copy `x.b_00[last]` to `x.b` and resubmit with `./opti>opti_test.out&`

Where to specify ocean data for the assimilation?

The ocean fields for the assimilation are specified in the `data.ecco` namelist.

Pickup files have name ckptA instead of numbers

Instead of changing the pickup file names you can also set in data file pickupSuff = ckptA or set nIter0 to whatever it says in the meta file for actual iteration: e.g. pickup.ckptA.meta.

The output "frequency" of pickup files with pchkFreq is set to value of nIter*deltaTClock to get one for a certain nIter. The "rolling" pickups (ckptA and ckptB) are always overwritten by the next one.

- MODULE not found

This means that `taf` didn't find this module. But you see that the subroutine is supplied in `bld_mpi`. It is because `mo_mapping_plothora.f` appears after `plasim_get_atmdat.f.` Change the order of these routines in the dependencies file `code_ad_diff.list`.

- How to read/write pickup files?

Use the MITgcm modules such as `rdslice` and/or `rdmds` for reading and `wrslice` for writing. The scripts are in the `MITgcm/utils`, try `help` command in `matlab` to see the usage options or check out the manuals on the MITgcm website.

Here is an example of usage for `rdslice:`

```
salt=rdslice('pickup.0000000000.data',[90,44,15],rec,'real*8');
```

- Cost function contribution detects missing value

There can be several reasons:

- In prgdfpmin.f90, you set wfac a weight factor for the control parameters. Make sure, they are defined properly real(kind=8):: wfac(1:18). Otherwise they will be zero if not defined and perturbations will be scaled by zero then.
- In plasim_get_atmdat.F you have scaling of the perturbations by the 1/STD. There might be smth wrong with scaling.
- The model might not read scaling from the prior error files correctly. It may also contribute to scaling perturbations by zero.

- make[1]: mpif90: Command not found

  If the node on marin is unavailable you may log-in to other node, however your configuration preferences might not work on a new node. Thus, you need to load modules to avoid the mentioned error:

```
module load intel/17.0.4
module load openmpi/2.0.0-static-intel17
module load cdo
```

- ## Abort_Message

Observation.srv files are corrupt. Resave and resubmit.

- ## Compilation error related to filter_ocean

In the old CESAM_4cpu_optiflx/bld_mpi/autodiff_inadmode_unset_ad.o has an older date than autodiff_inadmode_unset_ad.f that means it potentially was created from a different .f-file than we see now and that the one we see was modified after compilation. That's why it worked before but not now, because now the modification seems to happen before compilation.

Remove autodiff_inadmode_unset_ad.f and type `make autodiff_inadmode_unset_ad.f` to see how it is created (modified) and if it is without filter_ocean. `Then make autodiff_inadmode_unset_ad.o` This will avoid using manually modified filter_ocean.F introduced by Guokun.

Alternatively: The module filter_ocean.F is created by Guokun and is in dir "/scratch/cen/ifmrs/u241231/CESAM_ERA_parallel/MITgcm_c62y/verification/adplethora/code_t21p_forward_op_2datm_266272_150y". You should always put it in the mitgcm source code dir so that you can get a reasonable Makefile. OR JUST COPY IT TO the build dir bld*. Guokun only calls it in autodiff_inadmode_set_ad.f to remove some large adjoint sensitivity, especially when using a very large assimilation window. To add manually: 1) copy it to bld, 2) include it in autodiff_inadmode_set_ad.f (or overwrite it with the old ones that you generated). And 3) edit Makefile, include filter_ocean.f to the dependence tree.

- ## Attempt to access non-existent record

The data for optimization were stored in real*4 but was expected to be read in real*8 format.

- ## Empty g.b

The g.b file contains all NaNs. To find out which contribution to the cost function provides NaNs look at the file costfunction0000. Here Argo contributions are NaNs. It means either profiles or error files contain NaNs.

```
>> cat costfunction0000

fc     =  NaN 0.000000000000000D+00
f_temp =  0.180774306153022D+07 0.368608000000000D+06
f_salt =  0.795444794433715D+06 0.371628000000000D+06
```

```
f_argot = NaN 0.534160000000000D+05
f_argos = NaN 0.241710000000000D+05
f_sst = 0.272871941486412D+72 0.469700000000000D+04
```

Check if errors contain NaNs: sum(isnan(Levsalt(:))). The result is no NaNs in the error or profile files.

The cost function contribution for SST is very high. The T_err.bin and S_err.bin data should have 4*real format, while data profiles real*8.

- ## Opti run with observations_nudg.srv which contains data for years 1980-2018 is stalled

Either the file is corrupt (a) or it is too big (b).

a. Try to read in the merged file with copy_srv_day365.f90 and write it out to a new file to see whether it is the problem with cdo mergetime.

b. If it is the problem of the size, you should change the subroutine `nudgini` in `observations.F90` to allow reading in file every year. Armin suggests the size of data can go as high as 1TB.

- ## Submitting and post-processing routines:

a. `run.sh` and `burn.sh` scripts. The former one is for submitting multiple runs. The later `burn.sh` is provided by Guokun Lyu and writes the Plasim output in the NetCDF-format.

- ## How to modify the setting for the long opti run?

a. Adjust the setting of the control vector in Makefile, and mo_mapping and drivers.
b. Adjust reading of the observations.

Check in `calmod.f90, subroutine mmdd2yday(kyday,kyear,kmon,kday)`

the statement :

```
     if(n_days_per_year == 360 ) then
          kyday=(kmon-1)*30+kday
     endif
```

should be:

```
     if(n_days_per_year == 360 ) then
          kyday=(kmon-1)*30+kday+(kyear-1980)*360
! here assume you start from 1980
     endif
```

- The long optimization run stops at the end of the forward simulations with an unknown error…

  The run stops already after running function_relax. In STDOUT the HadISST records were read twice as less as the number of timesteps leading to the precision problem of the data - f32 instead of f64. After storing the HadISST data with the f64 precision, the issue was solved. Also all the other data that are specified in data.ecco were stored in f64 precision except of T_err and S_err.

- The long optimization run stops before integrating fields in the forward simulations with an unknown error…

  If the model starts in the very beginning, the error is likely related to the initial conditions or data assimilation fields. If you rely on the external data sets, check if the permission to the data changed.

- How do I know if the values of the cost function are reasonable?

```
sst=rdslice('HadISST198001-201812_af.bin',[90 44 100],1,'real*4');
sst(sst==0)=nan;
sst(sst<=-1.8)=nan; % lower boundary of SST
sst(sst>=40)=nan;   % upper boundary of SST
mask=zeros(size(sst));
mask(~isnan(sst))=1;
sum(sum(sum(mask(:,:,1:13))))
```

  The total number is 26071.

  Further, you can read the first layer in tbar.* and Hadsst, and compute the cost directly wsst*(tbar-Hadsst).^2

- Disc quota exceeded

Check quota by `df -h /scratch/cen/ifmrs`

References

Plasim manuals https://www.mi.uni-hamburg.de/en/arbeitsgruppen/theoretische-meteorologie/modelle/plasim.html%C2%A0

MITgcm manuals https://mitgcm.readthedocs.io/en/latest/index.html and https://github.com/MITgcm

Adcroft A., Hill C., Campin J.-M., Marshall J., and Heimbach P. (2004). Overview of the formulation and numerics of the MITgcm. *Proceedings of the ECMWF Seminar Series on Numerical Methods, Recent Developments in Numerical Methods for Atmosphere and Ocean Modelling*, 139–149, https://www.ecmwf.int/sites/default/files/elibrary/2004/7642-overview-formulation-and-numerics-mit-gcm.pdf

Blessing S., Kaminski T., Lunkeit F., Matei I., Giering R., Köhl A. and others. (2014). Testing variational estimation of process parameters and initial conditions of an Earth System Model. *Tellus A: Dynamic Meteorology and Oceanography, 66(1)*, 22606, https://doi.org/10.3402/tellusa.v66.22606

Giering R. and Kaminski T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS), 24(4)*, 437-474, https://doi.org/10.1145/293686.293695

Lyu, G., Köhl, A., Matei, I., & Stammer, D. (2018). Adjoint‐based climate model tuning: Application to the planet simulator. *Journal of Advances in Modeling Earth Systems*, *10*(1), 207-222, https://doi.org/10.1002/2017MS001194

Lyu, G. (2017) Calibrating an Earth System Model using the adjoint method. PhD Thesis.

Matei I., Köhl A. and Stammer D. (2017). CESAM and its skill in simulating the Earth climate. *MiKlip Status seminar 1-3 March 2017.* Berlin, Harnack-Haus.

Polkova, I., Brune, S., Kadow, C., Romanova, V., Gollan, G., Baehr, J., ... & Stammer, D. (2019). Initialization and ensemble generation for decadal climate predictions: A comparison of different methods. *Journal of Advances in Modeling Earth Systems*, *11*(1), 149-172, https://doi.org/10.1029/2018MS001439

Stammer, D., Köhl, A., Vlasenko, A., Matei, I., Lunkeit, F., & Schubert, S. (2018). A pilot climate sensitivity study using the CEN coupled adjoint model (CESAM). *Journal of Climate*, *31*(5), 2031-2056, https://journals.ametsoc.org/doi/full/10.1175/JCLI-D-17-0183.1