ECCO v4 development notes

Gaël Forget
Department of Earth, Atmospheric and Planetary Sciences
Massachusetts Institute of Technology

June 8, 2015

abstract

These notes pertain to the ECCO v4 state estimate, model setup, and associated codes (Forget et al., 2015). Section 1 points to the other elements of documentation that are available online, and associated download procedures. Section 2 provides guidance to ECCO v4 users interested in operating the ECCO v4 model set-up and/or reproducing the ECCO v4 solution. Section 3 documents the revamped and augmented estimation modules of MITgcm. Some of the included material is expected to move to to the MITgcm manual.

Contents

References

Forget, G., J.-M. Campin, P. Heimbach, C. N. Hill, R. M. Ponte, and C. Wunsch, 2015: Ecco version 4: an integrated framework for non-linear inverse modeling and global ocean state estimation. *Geoscientific Model Development Discussions*, **8 (5)**, 3653–3743, doi:10.5194/gmdd-8-3653-2015, URL http://www.geosci-model-dev-discuss.net/8/3653/2015/.

1 downloads

Here I document locations and directions to download the MITgcm, the ECCO v4 model setup, the ECCO v4 state estimate output, and related diagnostic matlab tools.

1.1 MITgcm

Pre-requisites are cvs, gcc, gfortran (or alternatives), and mpi (only for parallel runs). Then :

- The MITgcm web-page is mitgcm.org
- Install MITgcm using cvs as explained @ cvs
- Run MITgcm using testreport (for one experiment) as explained @ manual, howto

For example, my laptop setup, including mpi and netcdf, involved the following mac ports :

- cvs @1.11.23_1 (active)
- wget @1.14_5+ssl (active)
- gcc48 @4.8.2_0 (active)
- mpich-default @3.0.4_9+gcc48 (active)
- mpich-gcc48 @3.0.4_9+fortran (active)
- netcdf @4.3.0_2+dap+netcdf4 (active)
- netcdf-fortran @4.2_10+gcc48 (active)

Side note – overridding the default mac gcc and mpich with the above, further requires

- sudo port select –set gcc mp-gcc48
- sudo port select –set mpich mpich-gcc48-fortran

Side note – using mpi and netcdf within MITgcm requires two environment variables :

- export MPI_INC_DIR=/opt/local/include
- export NETCDF_ROOT=/opt/local

1.2 ECCO version 4 setup

Pre-requisites are MITgcm (see above) and mpi (except for small setup). User can then install the ECCO v4 setups, as explained @ README, using the setup_these_exps.csh shell script. This script downloads global_oce_cs32/ (small setup), global_oce_llc90/ (bigger setup) and global_oce_input_fields.tar.gz binary model inputs to global_oce_tmp_download/ (local subdirectory). User can then move these directories to MITgcm/verification/ to allow for automated execution by testreport using genmake2 (Fig.1).

Figure 1: MITgcm directory structure downloaded using cvs. The ECCO v4 directories indicated with "+" were downloaded separately using setup_these_exps.csh script and moved to MITgcm/verification/.

```
MITgcm/
├── model/ (core of MITgcm)
├── pkg/ (MITgcm modules)
├── verification/
│   ├── testreport (shell script)
│   ├── aim.5l_cs (mitgcm regression test)
│   ├── + global_oce_cs32/ (for laptops)
│   ├── + global_oce_llc90/ (for computers)
│   ├── + global_oce_input_fields/ (inputs)
│   ├── hs94.128x64x5 (mitgcm regression test)
│   └── ...
└── tools/
    ├── genmake2 (shell script)
    ├── build_options (wrt compilers)
    └── ...
```

1.3 solution

The release 1 solution directory linked to ecco-group.org contains :

- 20 year solution output : readme, fields, profiles, grid
- additional input files required to run the full 20 year solution (coming soon...).

1.4 analysis tools

Tools (e.g. matlab scripts) available to analyze the release1 solution, and others, include :

- download, set-up gcmfaces + MITprof using shell script or manually (see getting_started.m)
- download MITgcm/utils using cvs (basic functionalities only).

The so-called standard analysis.pdf is generated in matlab by means of diags_driver.m and diags_driver_tex.m in the following sequence :

```
diags_driver('release1/','release1/mat/',1992:2011);
diags_driver_tex('release1/mat/',{},'release1/tex/standardAnalysis');
```

assuming the conventional directory structure shown in Fig.2.

Figure 2: Directory structure as expected by gcmfaces and MITprof toolboxes. The toolboxes themselves can be relocated anywhere as long as their locations are included in the matlab path. Advanced analysis using diags_driver.m and diags_driver_tex.m will respectively generate the mat/ directory (for intermediate computational results) and the tex/ directory (for standard analysis). This diagnostic process relies on the depicted organization of GRID/ and solution/ for automation (user will otherwise be prompted to enter directory names) and depends on downloaded copies of fields to nctiles/ (local subdirectory).

```
./
|__gcmfaces/ (matlab toolbox)
|  |__sample_input/ (binary files)
|  |__@gcmfaces/ (matlab codes)
|  |__gcmfaces_calc/ (matlab codes)
|  |__...
|__MITprof/ (matlab toolbox)
|  |__profiles_samples/ (netcdf files)
|  |__profiles_process_main_v2/ (matlab codes)
|  |__profiles_stats/ (matlab codes)
|  |__...
|__GRID/ (binary output)
|__release 1 solution/
|  |__diags/ (binary output)
|  |__nctiles/ (netcdf output)
|  |__MITprof/ (netcdf output)
|  |__mat/ (created by gcmfaces)
|  |__tex/ (created by gcmfaces)
|__other solution/
|  |__diags/ (binary output)
|  |__...
|__...
```

2 MITgcm runs

Here I document a few procedures, commands and submission scripts that may be relevant to run the ECCO v4 MITgcm setup – either in short regression tests or for multi-decadal simulations such as the full 20 year state estimate. Downloading MITgcm and the ECCO v4 setups is a pre-requisite (section 1.2).

2.1 regression tests

MITgcm and ECCO v4 regression tests are run using testreport utility (see Fig.2; howto). Serial regression tests can always be executed simply with, e.g.

```
./testreport -t global_oce_cs32
```

```
./testreport -skipdir global_oce_llc90
```

If something goes wrong and/or interrupts the process it is often safer to clean up experiment directories (e.g., by executing ./testreport -clean -t global_oce_*) and start over. For example, the global_oce_llc90 experiments require 12 processors in forward (96 in adjoint), and may crash your laptop if you attempted to run them in serial mode.

Often in massive computing environments, however, mpi jobs can only be run within a queuing system. The, machine specific, submission script in Fig.3 provides an example. It contains 3 hard-coded switches : fwdORad = 1 (2 for adjoint); numExp = 1 (2 for llc90); excludeMpi = 0 (1 for serial). This script is located and submitted from MITgcm/verification. If compute nodes cannot access the remote adjoint compiler (TAF), then proceed in two steps :

1. compile outside of the queuing system using e.g.

   ```
   ./testreport -of ../tools/build_options/linux_amd64_ifort+mpi_ice_nas \
   -j 4 -MPI 96 -command 'mpiexec -np TR_NPROC ./mitgcmuv' \
   -t global_oce_llc90 -norun
   ```

2. Before submiting the Fig.3 script, add -q to the 'opt' variable to skip compilation.

Adjoint test require access to the TAF compiler. Then the call to testreport only needs to be altered by appending the '-ad' option and replacing 'mitgcmuv' with 'mitgcmuv_ad'. The ECCO v4 regression tests do not include the common, adjoint specific 'code_ad/' directory, which is generally un-necessary. Since testreport relies on the existence of 'code_ad/' for its adjoint option though, it is necessary to soft link 'code/' to 'code_ad/' in both global_oce_cs32/ and global_oce_llc90/ to run 'testreport -ad' on those experiments.

2.2 full ECCO v4 runs

Note to self ... [1]

There are three main differences between regression tests and full model runs (see howto) :

- compilation and run are done without testreport and with compiler optimization.

[1] Mention memory and disk space requirements, and additional input downloads.

- additional forcing, control vectors and/or observational inputs is necessary.
- additional memory and/or disk space is often necessary.

The typical compilation sequence is shown in Fig.4. The tamc.h_itXX and profiles.h_itXX headers allow for additional time steps, and additional in situ data input, respectively. Also note that compiling the adjoint requires a TAF license. Once that is done, user creates and enters a run directory, links everything into place (see Fig.5), and finally submits a job to the queueing system (see Fig.6).

A mechanism, analogous to testreport but for long runs, has been introduced recently (Forget et al., 2015) that is testreport_ecco.m run within Matlab, which requires the downloaded 'MITgcm/verification/global_oce_llc90/results_itXX/' to be in the Matlab path. By itself is compares cost functions and global mean time series to the reference state estimate values. These can be extended to meridional transports, which requires gcmfaces. The typical call sequence is indicated in the help of testreport_ecco.m and Fig.7.

Figure 3: Example script to run mpi testreport via a queueing system (machine dependent).

```
#PBS -S /bin/csh
#PBS -l select=1:ncpus=16:model=ivy+4:ncpus=20:model=ivy
#PBS -l walltime=02:00:00
#PBS -q devel
#PBS -m n

#environment variables and libraries
#-----------------------------------------------
limit stacksize unlimited
module purge
module load modules comp-intel/2013.1.117 mpi-sgi/mpt.2.10r6 netcdf/4.0
#
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${HOME}/lib
setenv MPI_IB_TIMEOUT 20
setenv MPI_IB_RAILS 2
setenv MPI_IB_FAILOVER 1
setenv  MPI_CONNECTIONS_THRESHOLD  2049

#local variables and commands
#-----------------------------------------
set fwdORad = 1
set numExp  = 1
set excludeMpi = 0
#
if ( ${numExp} == 1) then
  set nameExp = global_oce_cs32
  set NBproc  =  6
else
  set nameExp = global_oce_llc90
  set NBproc  = 96
endif
#
if ( ${excludeMpi} == 1 ) then
  set opt = '-of ../tools/build_options/linux_amd64_ifort -j 4'
else
  set opt = '-of ../tools/build_options/linux_amd64_ifort+mpi_ice_nas -j 4'
endif
#
if ( ${fwdORad} == 1 && ${excludeMpi} == 0 ) then
  ./testreport ${opt} -MPI \
  ${NBproc} -command 'mpiexec -np TR_NPROC ./mitgcmuv' -t ${nameExp}
else if ( ${fwdORad} == 2 && ${excludeMpi} == 0 ) then
  ./testreport ${opt} -MPI \
  ${NBproc} -command 'mpiexec -np TR_NPROC ./mitgcmuv_ad' -ad -t ${nameExp}
else if ( ${fwdORad} == 1 && ${excludeMpi} == 1 ) then
  ./testreport ${opt} -t ${nameExp}
else if ( ${fwdORad} == 2 && ${excludeMpi} == 1 ) then
  ./testreport ${opt} -ad -t ${nameExp}
endif

exit
```

Figure 4: Compilation directives, outside testreport, for intensive model runs. On a different machine (computer) another build option file such as linux_amd64_gfortran or linux_amd64_ifort11 should be used. To compile the adjoint, users need a TAF license and to replace 'make -j 4' with 'make adall -j 4'. Note : the '-mods=../code' specification can be omitted if the build directory contains the 'genmake_local' file).

```
cd verification/global_oce_llc90/build
../../../tools/genmake2 -optfile=\\
../../../tools/build_options/linux_amd64_ifort+mpi_ice_nas -mpi -mods=../code

make depend

\rm tamc.h profiles.h
cp ../code/tamc.h_itXX tamc.h
cp ../code/profiles.h_itXX profiles.h

make -j 4
```

Figure 5: Example script to setup the 20 year ECCO v4 state estimate. It is implied that user has filled directories /bla, /blaa, /blaaa and /blaaa with appropriate forcing, observational, control vector, and pickup files.

```
#!/bin/csh -f

set forcingDir  = ~/bla
set obsDir      = ~/blaa
set ctrlDir     = ~/blaaa
set pickDir     = ~/blaaaa

source ../input_itXX/prepare_run
cp ../build/mitgcmuv .
\rm pick*ta EIG*
ln -s ${forcingDir}/EIG* .
ln -s ${obsDir}/* .
ln -s ${ctrlDir}/xx* .
ln -s ${pickDir}/pick* .

exit
```

Figure 6: Example script to run the 20 year ECCO v4 state estimate on 96 processors (machine dependent).

```
PBS -S /bin/csh
#PBS -l select=1:ncpus=16:model=ivy+4:ncpus=20:model=ivy
#PBS -l walltime=12:00:00
#PBS -q long

#environment variables and libraries
#-----------------------------------------------
limit stacksize unlimited
module purge
module load modules comp-intel/2013.1.117 mpi-sgi/mpt.2.10r6 netcdf/4.0
#
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${HOME}/lib
setenv MPI_IB_TIMEOUT 20
setenv MPI_IB_RAILS 2
setenv MPI_IB_FAILOVER 1
setenv  MPI_CONNECTIONS_THRESHOLD  2049

#run MITgcm
#----------------
mpiexec -np 96 dplace -s1 ./mitgcmuv
exit
```

Figure 7: Calling sequence to be executed form within matlab to verify that their re-run of the 20 year ECO v4 state estimate is acceptably close to the released state estimate.

```
addpath ../results_itXX;%necessary .m and .mat files
mytest0=testreport_ecco([],'release1'); mytest0.info.interactive=0;%initialization
mytest=testreport_ecco(mytest0,'release1',[-1:4],'./',1);%compute the tests
testreport_ecco(mytest,'release1');%display the results
%testreport_write(mytest,'myRun');%save the results to a mat file
```

3 re-implemented ecco and ctrl packages

State estimation consists in minimizing a least squares distance, $\mathrm{J}(\mathfrak{u})$, that is defined as

$$
\begin{aligned}
\mathrm{J}(\mathfrak{u}) &= \sum_i \alpha_i \times (\mathrm{d}_i^T\, \mathbf{R_i}^{-1}\, \mathrm{d}_i) + \sum_j \beta_j \times (\mathfrak{u}_j^T \mathfrak{u}_j) && (1) \\
\mathrm{d}_i &= \mathcal{P}(\mathrm{m}_i - \mathrm{o}_i) && (2) \\
\mathrm{m}_i &= \mathcal{SDM}(\mathfrak{v}) && (3) \\
\mathfrak{v} &= \mathcal{Q}(\mathfrak{u}) && (4) \\
\mathfrak{u} &= \mathcal{R}(\mathfrak{u}') && (5)
\end{aligned}
$$

where d_i denotes a set of model-data differences, α_i the corresponding multiplier, $\mathbf{R_i}^{-1}$ the corresponding weights, $\mathfrak{u}_j$ a set of non-dimensional controls, β_j the corresponding multiplier, and additional symbols appearing in Eqs. 2-5 are defined below. The implementation of Eqs.1-5 and the adjoint interface within the MITgcm is charted in Fig. 8. A general presentation of Eqs.1-5 and Fig. 8 can readily be found in Forget et al. (2015). The focus here is on the underlying recent code development in the 'pkg/ecco' and 'pkg/ctrl' packages of MITgcm. These features are now tested daily via global_oce_cs32/ (adjoint experiment) that will also serve here for illustration in this document.

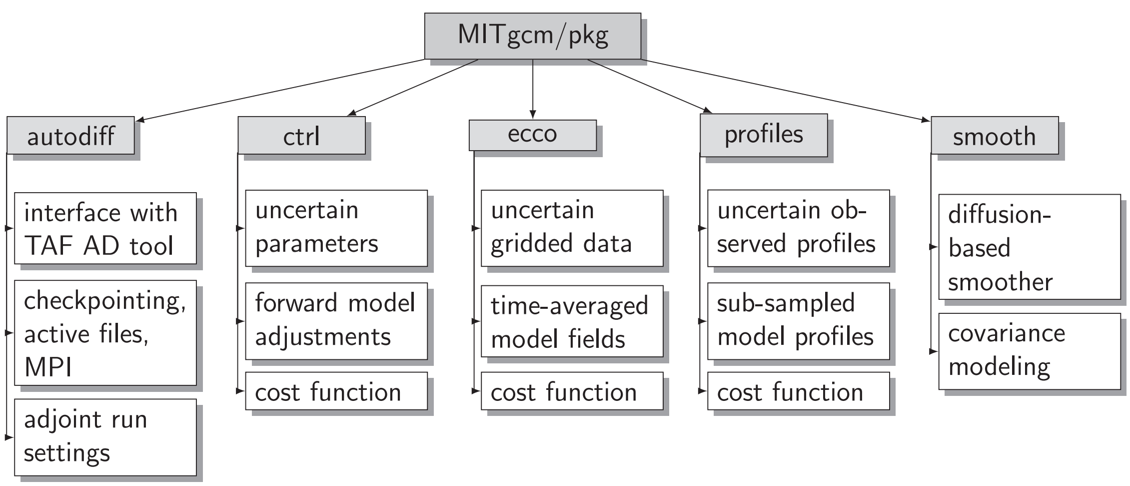

Figure 8: Chart of the organization and roles of MITgcm estimation modules. Additional details are reported in the MITgcm manual, Forget et al. (2015), and section 3.

3.1 pkg/ecco run-time parameters

Note to self ... [2]

Model counterparts (m_i) to observational data (o_i) derive from control parameters ($\mathfrak{v}$) through the model dynamics ($\mathcal{M}$), diagnostic computations ($\mathcal{D}$), and averaging (or subsampling in 'pkg/profiles') in space and time ($\mathcal{S}$). The physical variable in m_i is specified at run time via the first characters in 'gencost_barfile' (to match the observed variable in o_i) as illustrated in this data.ecco and that data.ecco. The list of implemented variables as of the MITgcm checkpoint c65l consists of 'eta', 'sst', 'sss', 'bp', 'tauZon', 'tauMer', 'theta', 'salt' (list obtained by: grep gencost_barfile pkg/ecco/cost_gencost_customize.F). In the case of three dimensional variables (e.g. 'theta' or 'salt') the 'gencost_is3d' run-time option must be set to .TRUE. (it .FALSE. by default). The file name for the observational fields (o_i) and the model-data uncertainty field ($\sqrt{\mathbf{R_i}}$) are specified at run time via 'gencost_datafile' and 'gencost_errfile' respectively. The cost function multiplier (α_i) further needs to be specified by 'mult_gencost' (it is 0. by default).

Both $\mathcal{D}$ and $\mathcal{S}$ in Eq.3 are mainly carried out as the forward model steps through time, respectively by ecco_phys.F and cost_averagesgeneric.F, and m_i is written to file periodically. m_i and o_i normally are time series of daily or monthly averages, as specified at run time via 'gencost_avgperiod'. However dense time series of model time steps can also be employed for testing purposes as illustrated in this data.ecco. Furthermore climatologies of m_i can be formed from its time series by cost_genread.F to allow for comparison with observational o_i climatologies. This part of the m_i processing is carried out after the full time series has been written to file. It is activated via the 'gencost_preproc' option as illustrated in this data.ecco.

Model-data misfits are computed (Eq. 2) upon completion of the forward model simulation by cost_generic.F that relies on ecco_toolbox.F for elementary operations and on cost_genread.F for re-reading m_i from file. Plain model-data misfits ($\mathrm{m}_i - \mathrm{o}_i$) can be penalized directly (i.e. used in Eq. 1 in place of d_i). More generally though misfits to be penalized (d_i in Eq. 1) derive from $\mathrm{m}_i - \mathrm{o}_i$ through a generic post-processor ($\mathcal{P}$ in Eq. 2). They can thus be smoothed in space at run time via 'gencost_posproc' for example (see this data.ecco). The overall sequence of operations for one cost function term is charted in Fig.9.

[2] Mention summary in stdout and cost function printouts

Algorithm 1 Generic cost function algorithm.

```
function COST_GENERIC(...)                    ▷ Argument list defines the cost function
    call ecco_zero                            ▷ Initialize local array to 0
    call ecco_cprsrl                          ▷ Copy mask to local array
    for irec = 1, nrecloop do                 ▷ Loop over time steps, days or months
        call cost_gencal                      ▷ Get file names, pointers
        Begin cost_genread                    ▷ Read, process model field
            if no preproc then
                call ecco_readbar             ▷ Read one record
            else if preproc=clim then
                call ecco_readbar within loop ▷ Average records
            end if
        End cost_genread
        call mdsreadfield                     ▷ Read observational field
        call ecco_diffmsk                     ▷ Compute masked model-data misfit
        if posproc=smooth then
            call smooth_hetero2d              ▷ Smooth masked misfit
        end if
        call ecco_addcost                     ▷ Add to cost function
    end for
end function
```

Figure 9: Chart of the generic cost function routine in pkg/ecco.

3.2 pkg/ctrl run-time parameters

Note to self ... [3]

The control problem is non-dimensional, as reflected by the omission of weights in control penalties ($\mathfrak{u}_j^T \mathfrak{u}_j$, Eq.1). Non-dimensional controls are scaled to physical units through multiplication by their respective uncertainty fields, as part of the generic pre-processor Q (Eq.4) that can also include the spatial correlation model and/or a mapping in time such as the cyclic repetition of mean seasonal controls for example. Pre-conditioner $\mathcal{R}$ (Eq.5) does not appear in the estimation problem itself (Eq.1), as it only serves to push an optimization process preferentially towards certain directions of the control space.

Key pkg/ctrl generic routines :

- ctrl_map_ini_gen.F computes dimensional control vector adjustments (Eq.4).
- ctrl_map_ini_gentim2d.F computes dimensional control vector adjustments (Eq.4).
- ctrl_map_gentim2d.F maps time varying controls to active model variables.
- ctrl_map_ini_genarr.F maps time invariant controls to active model variables.
- ctrl_cost_gen.F computes cost function penalties for all generic controls (in Eq.1).

3.3 MITgcm compiling options

Note to self ... [4]

Much of the legacy code that has been distributed as part of 'pkg/ecco' and 'pkg/ctrl' in the past is now deprecated – it is superseeded by the generic cost functions and controls codes presented above. Most of the deprecated codes had not been tested or maintained for many years, and consist of variations of the same operations duplicated many times. Another issue was the lack of organization amongst the deprecated codes (unlike in Fig.8). The consensus was that there was no point in keeping them around much longer.

For the time being the deprecated codes still exist but they are not compiled anymore unless the 'ECCO_CTRL_DEPRECATED' compile option is added in e.g. 'ECCO_CPPOPTIONS.h' (see below for details). To further facilitate the transition from old to new setup, the ctrlUseGen run-time parameter was added that switches between the old and new (generic) treatment of control vectors (assuming that 'ECCO_CTRL_DEPRECATED' was defined at compile time). As a side note: there is one non-generic feature that ISN'T deprecated since it has not been re-implemented in generic fashion, which is the control of open boundary conditions.

The deprecation of the legacy codes leads to a vast reduction in the volume of estimation codes (30% of the code treated by automatic differentiation, which includes the entire physical model, was removed in the process), a vast addition of capabilities (new or pre-existing functionalities are now available for any gridded data set), and a greatly improved flexibility (virtually all options can now be switched on/off at run time). Furthermore, the ecco, ctrl and autodiff packages were made independent of each other, and to follow general principles of MITgcm packages. Thus they can now be switched on/off at run time, independently (by virtue of useECCO, useCTRL, useAUTODIFF).

[3] Mention this data.ctrl ... that data.ctrl ... this CTRL_OPTIONS.h ... eccodevel email

[4] Mention optim and packing

Compiling options are typically found in the 'code/' directory of any given setup of MITgcm (when customized) or in the corresponding MITgcm package (when using defaults). The most obvious difference between the new setup and an old setup is that CPP_OPTIONS.h now disregards ECCO_CPPOPTIONS.h and uses the following instead :

- AUTODIFF_OPTIONS.h contains the few compile directives of pkg/autodiff. The maximum numbers of time steps are set in tamc.h
- ECCO_OPTIONS.h contains compile directives of pkg/ecco. Very few remain necessary, since all generic cost function settings can now be chosen at run time. The maximum numbers of cost terms are set in ecco.h
- CTRL_OPTIONS.h contains compile directives of pkg/ctrl. Very few remain necessary, since all generic control settings can now be chosen at run time. The maximum numbers of controls are set in CTRL_SIZE.h
- along with MOM_COMMON_OPTIONS.h, GMREDI_OPTIONS.h, GGL90_OPTIONS.h, PROFILES_OPTIONS.h, EXF_OPTIONS.h, SEAICE_OPTIONS.h, DIAG_OPTIONS.h