

Using ECCO v4

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

October 23, 2015

abstract

These notes pertain to the ECCO v4 state estimate, model setup, and associated codes (Forget et al., 2015). Section 1 summarizes download procedures and links to additional documentation¹. Section 2 explains how ECCO v4 solutions, or corresponding short regression tests, can be re-run.

Contents

1	Downloading ECCO v4	2
1.1	Released ECCO v4 Solution	2
1.2	Diagnostic Tools	2
1.3	ECCO v4 setup	2
2	Running ECCO v4	3
2.1	Baseline ECCO v4 solutions	3
2.2	Short Regression Tests	4
2.3	Iterative Optimization Test Case	6

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*, **8** (10), 3071–3104, doi:10.5194/gmd-8-3071-2015, URL <http://www.geosci-model-dev.net/8/3071/2015/>.

¹Throughout this document links are indicated by blue colored font.

1 Downloading ECCO v4

This section first provides direction to download the ECCO v4-release 1 state estimate output (section 1.1) and associated matlab analysis tools (section 1.2). It then explains download procedures for the ECCO v4 model setup and MITgcm (section 1.3). Sections 1.2 and 1.3 rely on the MITgcm cvs server (see section 1.3). As an alternative, frozen versions are available [here](#).

1.1 Released ECCO v4 Solution

The model output for the ECCO v4-release 1 state estimate is available via [this opendap server](#) and [this ftp server](#) from [ecco-group.org](#). The servers provide the [grid files](#) and [monthly output fields](#) in ‘nctiles’ format, as well as [collocated in situ and state estimate profiles](#) in ‘MITprof’ format. The ‘nctiles’ and ‘MITprof’ format are described in Forget et al. (2015). The files can be downloaded at the command line, e.g. within a linux environment, by typing

```
wget --recursive ftp://mit.ecco-group.org/ecco_for_las/version_4/release1/nctiles_grid
wget --recursive ftp://mit.ecco-group.org/ecco_for_las/version_4/release1/nctiles
wget --recursive ftp://mit.ecco-group.org/ecco_for_las/version_4/release1/MITprof_release1
```

1.2 Diagnostic Tools

To analyze model output from section 1.1 or section 2.1, two sets of Matlab tools are available:

- The [gcmfaces+MITprof](#) framework (see Forget et al., 2015) normally gets installed using [this shell script](#) (via [cvs](#)). Alternatively, frozen versions are available [here](#).
- Basic MITgcm [tools](#) can also be downloaded via [cvs](#).

For example, one may readily regenerate [this set of diagnostics](#) for the ECCO v4-release 1 state estimate using ‘diags_driver.m’ as explained in [the gcmfaces.pdf documentation](#).

1.3 ECCO v4 setup

First, install the MITgcm using cvs as explained [here](#). Second, install the ECCO v4 model setup on the LLC90 and CS32 grids (see Forget et al., 2015) also via [cvs](#) according to:

```
cd MITgcm/verification
cvs co -P -d global_oce_llc90 MITgcm_contrib/gael/verification/global_oce_llc90
cvs co -P -d global_oce_cs32 MITgcm_contrib/gael/verification/global_oce_cs32
```

Alternatively, frozen versions are available [here](#). [global_oce_cs32/](#) (614Mo) is a small setup used only for testing, whereas [global_oce_llc90/](#) (595Mo) is the production setup that typically runs on 96 processors. Running the section 2.1 solutions furthermore requires downloading the three-hourly forcing fields (127Go) from:

```
cd MITgcm/verification
wget --recursive ftp://ecco2.jpl.nasa.gov/data1/ecco/version4r1/StandAloneForward/era_xx/
```

Running the section 2.2 regression tests instead requires ‘global_oce_input_fields/’ (1.6Go):

```

35 cd MITgcm/verification
36 wget http://mitgcm.org/~gforget/global_oce_input_fields.tar.gz
37 gunzip global_oce_input_fields.tar.gz
38 tar xf global_oce_input_fields.tar
39 \rm -f global_oce_input_fields.tar

```

40 2 Running ECCO v4

41 This section explains how the ECCO v4 setup is used to rerun 20-year state estimates (section
 42 2.1) or short regression tests (section 2.2). As a pre-requisite, one must have downloaded the
 43 MITgcm as well as the ECCO v4 model setup and inputs (section 1.3). Based upon the section
 44 1.3 directions, the various downloaded directories should be organized as illustrated in Fig.1
 45 within the MITgcm directory. Running MITgcm furthermore requires the following software:
 46 gcc, gfortran (or alternatives), mpi (only for parallel runs) and netcdf (only if ‘pkg/profiles’ is
 47 used). Additional information can be found in [the MITgcm howto](#) and in [the MITgcm manual](#).

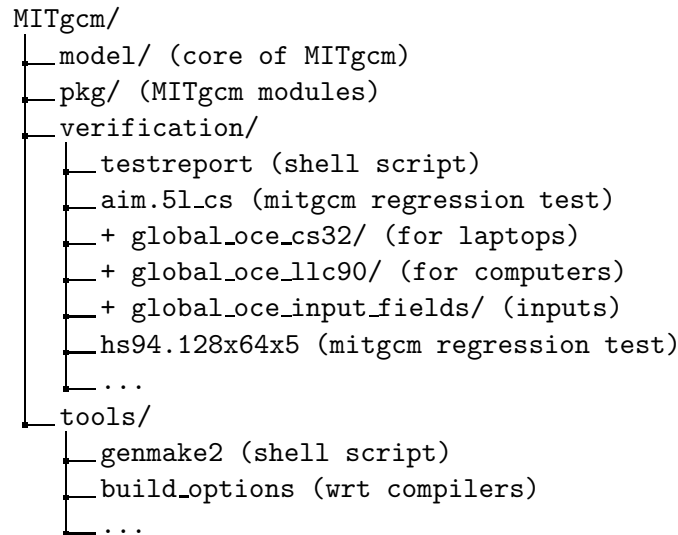


Figure 1: MITgcm directory structure including the ECCO v4 directories (indicated with “+”) downloaded according to the section 1.3 directions.

48 2.1 Baseline ECCO v4 solutions

49 The ‘baseline1’ and ‘baseline2’ versions of the 1992-2011 ECCO v4 state estimate are minor up-
 50 dates of ‘release1’ (Forget et al., 2015) that are easiest for outside users to reproduce. ‘baseline1’
 51 most closely matches the ‘release1’ output (section 1.1). ‘baseline2’ further benefits from: 1)
 52 inclusion of geothermal heating at the sea floor; 2) increased adjoint stability due to additional
 53 bottom viscosity. The small impact of these additional features is documented by Forget et al.
 54 (2015). The standard analysis for ‘baseline1’ and ‘baseline2’ are furthermore available [here](#)².

²coming soon...

To re-run ‘baseline2’ one proceeds according to Fig. 2. A 20-year ECCO v4 model run typically takes between 6 to 12 hours on 96 cores (depending on the computing environment). To verify the re-run results one proceeds according to Fig. 3. The expected level of accuracy for 20-year re-runs (with an up to date MITgcm; on any given computer) is reached when the displayed values are all < -4 (see Forget et al., 2015, for details).

To re-run ‘baseline1’ instead of ‘baseline2’ one needs a couple modifications to the setup:

- (a) define ‘ALLOW_KAPGM_CONTROL_OLD’ and ‘ALLOW_KAPREDI_CONTROL_OLD’ in ‘global_oce_llc90/code/GMREDI_OPTIONS.h’ before compiling the model;
- (b) copy ‘global_oce_llc90/input.ecco_itXX/data’ over ‘global_oce_llc90/input.ecco_v4/data’ before running the model. Users who may want to reproduce ‘release1’ even more precisely than ‘baseline1’ does should contact ecco-support@mit.edu to obtain additional model inputs.

```
#1) compile the model
cd MITgcm/verification/global_oce_llc90/build
../../tools/genmake2 -mods=../code -optfile \
    ../../tools/build_options/linux_amd64_gfortran -mpi
make depend
make -j 4

#2) link files into run directory
cd ../run
ln -s ../build/mitgcmuv .
ln -s ../input.ecco_v4/* .
ln -s ../input_fields/* .
ln -s ../era_xx/* .

#3) run model
mpiexec -np 96 ./mitgcmuv
```

Figure 2: Procedure to re-run the ECCO v4 state estimate (‘baseline2’ version). Pre-requisites: (1) installation of gcc, gfortran (or alternatives), and mpi (only for parallel runs); (2) installation of the MITgcm and ECCO v4 setup installation according to section 1.3.

The number of cores (96 by default and in Fig. 2) can be reduced to, e.g., 24 by copying ‘global_oce_llc90/code/SIZE.h_24cores’ over ‘global_oce_llc90/code/SIZE.h’ before compiling the model and then running it with ‘mpiexec -np 24 ./mitgcmuv’. Different compiler options (alternatives to ‘linux_amd64_gfortran’ in Fig. 2) are available in ‘MITgcm/tools/build_options’. Additionally the adjoint model can be compiled and run as explained in the MITgcm manual by users holding a TAF license.

2.2 Short Regression Tests

To ensure continued compatibility with the up to date MITgcm, the ECCO v4 model setup is also tested on a daily basis using the ‘testreport’ command line utility (indicated in Fig.1) that compares re-runs with reference results over a few time steps. The reader is referred to

Figure 3: Top: instructions to verify (using ‘testreport_ecco.m’ within Matlab) that a re-run of the baseline ECCO v4 solution (currently ‘baseline2’) is acceptably close to the reference result. Bottom: example output from testreport_ecco.m where the re-run agrees up to 6 digits with the reference result.

```
cd MITgcm/verification/global_oce_llc90
matlab -nodesktop -nodisplay

addpath results_itXX;%necessary .m and .mat files
mytest=testreport_ecco('run/');%compute the tests and display result
```

```
-----
& jT & jS & jTs & ... & (reference is)
run/ & (-6) & (-6) & (-6) & ... & baseline2
-----
```

76 ‘testreport -help’ and [the MITgcm howto](#) for details. The short regression test of the smaller
77 ECCO v4 setup ([global_oce_cs32/](#)) is thus executed by typing:

78 `./testreport -t global_oce_cs32`

79 If everything proceeds as expected then the regression test results are reported to screen as
80 shown in Fig. 4. On other machines the degree of agreement (16 digits in the Fig. 4 example)
81 may vary and ‘testreport’ may indicate ‘FAIL’. Despite the dramatic character of such message,
82 it may not prevent reproducing 20-year solutions (see section 2.1) with acceptable accuracy. An-
83 other side note: for short regression test purposes, one can copy ‘global_oce_llc90/code/tamc.h_short’
84 and ‘global_oce_llc90/code/PROFILES_SIZE.h_short’ over ‘global_oce_llc90/code/tamc.h’ and
85 ‘global_oce_llc90/code/PROFILES_SIZE.h’ to somewhat reduce the memory footprint and disk
86 storage.

```
default 10 ----T----- ----S-----
G D M      c          m s          m s
e p a R g m m e . m m e .
n n k u 2 i a a d i a a d
2 d e n d n x n . n x n .

Y Y Y Y>14<16 16 16 16 16 16 16 16 pass global_oce_cs32
```

Figure 4: Abbreviated output of testreport to screen.

87 It should be stressed that the bigger ECCO v4 setup ([global_oce_llc90/](#)) requires at least 12
88 cores in forward mode (96 in adjoint mode) and therefore should not be run using the above
89 command (or on a laptop). Instead the [global_oce_llc90/](#) regression tests use mpi:

```

90 ./testreport -of ../tools/build_options/linux_amd64_gfortran \
91 -j 4 -MPI 24 -command 'mpiexec -np TR_NPROC ./mitgcmuv' \
92 -t global_oce_llc90

```

with 24 processors and gfortran (these settings may differ on another machine). To prevent users from running these experiments in serial mode (e.g. via a plain ‘./testreport’ command) the results are in ‘global_oce_llc90/results/hidden/’. To activate the experiments the ‘output*’ files contained in this directory must be linked to ‘global_oce_llc90/results/’.

Adjoint regression tests are also performed daily, and are available to users holding a [TAF](#) license. The ECCO v4 daily regression test results (forward & adjoint, [cs32](#) & [llc90](#), using gfortran and 24 processors) from a machine called ‘glacier’ are reported [on this site](#).

100 2.3 Iterative Optimization Test Case

101 The [global_oce_cs32/input_OI](#) directory implements an iterative optimization test case. It boils down to optimal interpolation solved by a variational method using the MITgcm adjoint (the ocean model being replaced with a simple diffusion equation here). The pre-requisites are:

- 104 1. run the adjoint benchmark in [global_oce_cs32/](#) via testreport (see section [2.2](#)).
- 105 2. Go to MITgcm/lsopt and compile (see section 3.18 of [manual](#)).
- 106 3. Go to MITgcm/optim, replace ‘natl_box_adjoint’ with ‘global_oce_cs32’ in [this Makefile](#),
107 and compile as explained in section 3.18 of [manual](#). An executable named ‘optim.x’ should
108 get created in MITgcm/optim. If otherwise, please contact ecco-support@mit.edu
- 109 4. go to MITgcm/verification/global_oce_cs32/input_OI and type ‘source ./prepare_run’

110 Then the iterative optimization itself proceeds as follows

- 111 1. ./mitgcmuv_ad > output.txt
- 112 2. ./optim.x > op.txt
- 113 3. increment optimcycle by 1 in data.optim
- 114 4. go back to step #1, to run the next iteration
- 115 5. type ‘grep fc costfunction000*’ to display results (Fig. [5](#)).

costfunction0000: fc =	4118.1987222194211	0.00000000
costfunction0001: fc =	1523.9310891186672	0.00000000
costfunction0002: fc =	1053.3611790049420	0.00000000
costfunction0003: fc =	790.10479375339185	0.00000000

Figure 5: Results of iterative optimization after 3 iterations carried out as explained in section [2.3](#).