ECCO v4 Documentation

Release 0.1.0

Gael Forget

Apr 19, 2024

CONTENTS:

Learn to download, analyze, or reproduce *ECCO version 4* solutions [FCH+15]. In *Download*, *Analyze*, and *Reproduce* we focus on ECCO V4r2 as an example, and link to other solutions.

CHAPTER

ONE

INTRODUCTION

ECCO version 4 is an ocean state estimate [FCH+15] and solution of the MIT general circulation model (MITgcm). ECCO V4 release 2 (*V4r2*) covers the period from 1992 to 2011 [FCH+16].

Download for an installation guide and links to data.

Analyze for analyses tools in Julia, Python, Matlab.

Reproduce for simple instructions to reproduce the model simulation.

Note: Reproducing the model simulation (in the cloud or locally) enables you to generate additional output if needed, or prepare a new numerical experiment with the model for example.

CHAPTER

TWO

DOWNLOAD

This section provides directions to download the *ECCO V4r2* model output for analysis (Section 2.1), or download the underlying model setup to reproduce *ECCO V4r2* (Section 2.2).

2.1 Model Solution

ECCO V4r2 was the first update to the original ECCO V4 solution [FCH+15] with :

1. additional corrections listed in [FCH+16]
2. additional model-data misfit and model budget output
3. easier to rerun than ECCO version 4 release 1

The reference output for *ECCO V4r2* is permanently archived in this Dataverse , which provides citable identifiers for the various datasets (see README.pdf). For direct downloads we recommend Dataverse.jl in Julia with code provided below.

Folders that start with `nctiles_` contain data on the native LLC90 grid in the nctiles format [FCH+15]. This format is easily read in *Julia*, *Python*, and *Matlab/Octave* (see Section 3.1).

Note: Alternatively interpolated fields, on a $1/2 \times 1/2^{\circ}$ grid in standard NetCDF, is available via ecco-group.org.

Folders that start with `input_` directories contain binary and netcdf input files for *MITgcm* (Section 4.1). The `profiles/` folder contains the *MITprof* collections of collocated in situ and state estimate profiles [FCH+15].

2.2 Model Setup

Donwload MITgcm source code and ECCO V4r2 setup from GitHub.

```
git clone https://github.com/MITgcm/MITgcm
git clone https://github.com/gaelforget/ECCOv4
mkdir MITgcm/mysetups
mv ECCOv4 MITgcm/mysetups/.
```

To run *ECCO V4r2* requires surface forcing input (96G of 6-hourly fields), initial condition, grid, etc. input (610M), and observational input (25G) from this Dataverse.

```
cd("MITgcm/mysetups/ECCOv4")
include("input/download_files.jl")
import Main.baseline2_files: get_list, get_files

list1=get_list()
[get_files(list1,nam1,pwd()) for nam1 in list1.name]
```

The *Recommended Directory Organization* is shown below. While organizing the downloaded directories differently is certainly possible, the Section 4.1 instructions to *Compile, Link, And Run* the model and *Verify Results Accuracy* are based on this organization.

Recommended Directory Organization

```
MITgcm/
  model/   (MITgcm core)
  pkg/     (MITgcm modules)
  tools/
    genmake2        (shell script)
    build_options   (compiler options)
  mysetups/         (user created)
    ECCOv4/
      build/                (build directory)
      code/                 (compile-time settings)
      input/                (run-time settings)
      test/                 (reference results)
      forcing_baseline2/    (user installed)
      inputs_baseline2/     (user installed)
```

Note: Some subdirectories are omitted in this depiction.

CHAPTER

THREE

ANALYZE

This section is focused on tools for analyzing outputs (Section 3.1), and additional resources (Section 3.4).

3.1 Julia Toolbox

To be continued ...

3.2 Python Toolbox

To be continued ...

3.3 Matlab Toolbox

The *gcmfaces* toolbox [FCH+15] can be used to analyze model output that has either been downloaded (Section 2.1) or reproduced (Section 4.1) by users. From the command line, you can install either the *Matlab* version by executing:

```
git clone https://github.com/gaelforget/gcmfaces
```

or the *Octave* version by executing:

```
git clone -b octave https://github.com/gaelforget/gcmfaces
```

The *gcmfaces* toolbox can be used, e.g., to reproduce the *standard analysis* (i.e., the plots in [FCH+16]) from released, nctiles model output (Section 2.1) or from plain, binary model output (Section 4.1). For more information, please consult the gcmfaces user guide.

3.4 Other Resources

- A series of three presentations given during the May 2016 *ECCO* meeting at *MIT* provides an overview of *ECCO v4* data sets, capabilities, and applications (Overview; Processes; Tracers).
- Various Python tools are available to analyse model output (see, e.g., this tutorial).
- Any *netcdf* enabled software such as Panoply (available for *MS-Windows*, *Linux*, or *macOS*) can be used to plot the interpolated output (`interp_*` directories).

- The stand-alone eccov4_lonlat.m program can be used to extract the lat-lon sector, which spans the 69S to 56N latitude range, of native grid fields [FCH+15].
- *ECCO v4* estimates can be plotted via the *NASA* Sea Level Change Portal tools (interpolated output) or downloaded from the Harvard Dataverse APIs (native grid input and output).

CHAPTER

FOUR

REPRODUCE

This section first explains how the *MITgcm* can be used to re-run the *ECCO v4 r2* solution over the 1992–2011 period (Section 4.1). Other state estimate solutions (Section 4.2), short regression tests (Section 4.3), and optimization tests (Section 4.4) are discussed afterwards.

Required Computational Environment

Running the model on a linux cluster requires *gcc* and *gfortran* (or alternative compilers), *mpi* libraries (for parallel computation), and *netcdf* libraries (e.g., for the *profiles* package) as explained in the MITgcm documentations. In *ECCO v4 r2*, the 20-year model run typically takes between 6 to 12 hours when using 96 cores and modern on-premise clusters.

Users who may lack on-premise computational resources or IT support can use the included cloud computing recipe to leverage *Amazon Web Services*'s `cfncluster` technology. This recipe sets up a complete computational environment in the *AWS* cloud (hardware, software, model, and inputs). When this recipe was tested in January 2017, the 20-year *ECCO v4 r2* model run took under 36h using 96 vCPUs and *AWS spot instances* for a cost of about 40$.

4.1 The Release 2 Solution

This section assumes that *MITgcm*, the *ECCO v4* setup, and model inputs have been installed according to the *Recommended Directory Organization* (see Section 2.2). Users can then *Compile, Link, And Run* the model to reproduce *ECCO v4 r2*, and *Verify Results Accuracy* once the model run has completed.

Compile, Link, And Run

```
#1) compile model
cd MITgcm/mysetups/ECCOv4/build
../../../tools/genmake2 -mods=../code -optfile \
     ../../../tools/build_options/linux_amd64_gfortran -mpi
make depend
make -j 4
cd ..

#2) link files into run directory
mkdir run
cd run
ln -s ../build/mitgcmuv .
ln -s ../input/* .
```

(continues on next page)

(continued from previous page)

```
ln -s ../inputs_baseline2/* .
ln -s ../forcing_baseline2 .

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

Note: On most clusters, users would call `mpiexec` (or `mpirun`) via a queuing system rather than directly from the command line. The cloud computing recipe provides an example.

Other compiler options, besides `linux_amd64_gfortran`, are provided by the *MITgcm* development team in `MITgcm/tools/build_options/` for cases when *gfortran* is not available. The number of cores is 96 by default as seen in *Compile, Link, And Run*. It can be reduced to, e.g., 24 simply by copying `code/SIZE.h_24cores` over `code/SIZE.h` before compiling the model and then running *MITgcm* with `-np 24` rather than `-np 96` in *Compile, Link, And Run*. It can alternatively be increased to, e.g., 192 cores to speed up the model run or reduce memory requirements. In this case one needs to use `code/SIZE.h_192cores` at compile-time and `input/data.exch2_192cores` at run-time.

Verify Results Accuracy

In Julia, `testreport_ecco.jl` provides means to evaluate the accuracy of solution re-runs [FCH+15]. To use it, open julia and first proceed as follows:

```
using Distributed

@everywhere begin
 include("test/testreport_ecco.jl")
 using SharedArrays
end

report=eccotest.compute("run")
```

And then compare the reference result :

```
using CSV, DataFrames
ref_file="test/testreport_baseline2.csv"
ref=CSV.read(ref_file,DataFrame)

eccotest.compare(report,ref)
```

In Matlab, `testreport_ecco.m` provides means to evaluate the accuracy of solution re-runs [FCH+15]. To use it, open Matlab or Octave and proceed as follows:

```
cd MITgcm/mysetups/ECCOv4;
p = genpath('gcmfaces/'); addpath(p); %this can be commented out if needed
addpath test; %This adds necessary .m and .mat files to path
mytest=testreport_ecco('run/'); %This compute tests and display results
```

When using an up-to-date copy of *MITgcm* and a standard computational environment, the expected level of accuracy is reached when all reported values are below -3 [FCH+15]. For example:

```
------------------------------------------------------------
      &   jT &   jS &      ... &  (reference is)
```

(continues on next page)

(continued from previous page)

```
run/   & (-3) & (-3) &      ...  &  baseline2
-------------------------------------------------------------
```

Accuracy tests can be carried out for, e.g., meridional transports using the *gcmfaces* toolbox (see Section 3.1), but the most basic ones simply rely on the *MITgcm* standard output file (`STDOUT.0000`).

4.2 Other Known Solutions

ECCO version 4 release 3: extended solution that covers 1992 to 2015 and was produced by *O. Wang* at JPL; to reproduce this solution follow O. Wang's directions or those provided in ECCOv4r3_mods.md.

ECCO version 4 baseline 1: older solution that most closely matches the original, *ECCO version 4 release 1*, solution of [FCH+15]; to reproduce this solution follow directions provided in ECCOv4r1_mods.md.

Users who may hold a TAF license can also:

1. compile the adjoint by replacing `make -j 4` with `make adall -j 4` in *Compile, Link, And Run*
2. activate the adjoint by setting `useAUTODIFF=.TRUE.,` in `input/data.pkg`
3. run the adjoint by replacing `mitgcmuv` with `mitgcmuv_ad` in *Compile, Link, And Run*.

4.3 Short Forward Tests

To ensure continued compatibility with the up to date *MITgcm*, the *ECCO v4* model setup is tested on a daily basis using the `MITgcm/verification/testreport` command line utility that compares re-runs with reference results over a few time steps (see below and the MITgcm howto for additional explanations). These tests use dedicated versions of the *ECCO v4* model setup which are available under MITgcm_contrib/verification_other/.

global_oce_llc90/ (595M) uses the same LLC90 grid as the production *ECCO v4* setup does. Users are advised against running even forward LLC90 tests with fewer than 12 cores (96 for adjoint tests) to avoid potential memory overloads. global_oce_cs32/ (614M) uses the much coarser resolution CS32 grid and can thus be used on any modern laptop. Instructions for their installation are provided in this README and that README, respectively. Once installed, the smaller setup can be executed on one core, for instance, by typing:

```
cd MITgcm/verification/
./testreport -t global_oce_cs32
```

The test outcome will be reported to screen as shown in *Sample Test Output*. Daily results of these tests, which currently run on the *glacier* cluster, are reported on this site. To test global_oce_llc90/ using 24 processors and *gfortran* the corresponding command typically is:

```
cd MITgcm/verification/
./testreport -of ../tools/build_options/linux_amd64_gfortran \
-j 4 -MPI 24 -command 'mpiexec -np TR_NPROC ./mitgcmuv' \
-t global_oce_llc90
```

Sample Test Output

Below is an abbreviated example of testreport output to screen.

```
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
```

Note: The degree of agreement (16 digits in *Sample Test Output*) may vary from computer to computer, and `testreport` may even indicate *FAIL*, but this does not mean that users won't be able to reproduce 20-year solutions with acceptable accuracy in Section 4.1.

4.4 Other Short Tests

Running the adjoint tests associated with Section 4.3 requires: (1) holding a TAF license; (2) soft linking `code/` to `code_ad/` in global_oce_cs32/ and global_oce_llc90/. Users that hold a TAF license can then further proceed with the iterative optimization test case in global_oce_cs32/input_OI/. For this demo, the ocean model is replaced with a simple diffusion equation.

The pre-requisites are:

1. run the adjoint benchmark in global_oce_cs32/ via testreport (see section *2.3*).
2. Go to `MITgcm/lsopt/` and compile (see MITgcm manual).
3. Go to `MITgcm/optim/`, replace *natl_box_adjoint* with *global_oce_cs32* in the Makefile, and compile as explained in MITgcm manual to generate the `optim.x` executable. If this process failed, please contact mitgcm-support@mit.edu
4. go to `global_oce_cs32/input_OI/` and type `source ./prepare_run`

To match the reference results from `input_OI/README`, users should proceed as follows

1. `./mitgcmuv_ad > output.txt`
2. `./optim.x > op.txt`
3. increment *optimcycle* by 1 in `data.optim`
4. go back to step #1 to run the next iteration
5. type `grep fc costfunction00*` to display results

BIBLIOGRAPHY

[FCH+15] G. Forget, J.-M. Campin, P. Heimbach, C. N. Hill, R. M Ponte, and C. Wunsch. ECCO version 4: an integrated framework for non-linear inverse modeling and global ocean state estimation. *Geoscientific Model Development*, 8(10):3071–3104, 2015. URL: http://www.geosci-model-dev.net/8/3071/2015/, doi:10.5194/gmd-8-3071-2015.

[FCH+16] G. Forget, J.-M. Campin, P. Heimbach, C. N. Hill, R. M Ponte, and C. Wunsch. ECCO version 4: second release. 2016. URL: http://hdl.handle.net/1721.1/102062.