Providing the ARCHER community with adjoint modelling tools for high-performance oceanographic and cryospheric computation

Gavin J. Pringle (EPCC), Daniel C. Jones (British Antarctic Survey), Sudipta Goswami (British Antarctic Survey), Sri Hari Krishna Narayanan (Argonne National Laboratory), and Daniel Goldberg (University of Edinburgh)

Version 1.1, 11th October, 2016

1 Introduction

The MITgcm (MIT General Circulation Model, http://mitgcm.org/) is a numerical model designed for study of the atmosphere, ocean, and climate (Marshall et al. 1997a,b). Its non-hydrostatic formulation enables it to simulate fluid phenomena over a wide range of scales; its adjoint capability enables it to be applied to sensitivity analysis and state estimation problems. By employing fluid isomorphisms, one hydrodynamical kernel can be used to simulate flow in both the atmosphere and ocean.

This report describes the work undertaken under the embedded CSE programme of the ARCHER UK National Supercomputing Service (www.archer.ac.uk), and was entitled "Providing the ARCHER community with adjoint modelling tools for high-performance oceanographic and cryospheric computation", where the main proposer was Dr Dan Jones (British Antarctic Survey, or BAS), with co-proposers Dr Dan Goldberg (University of Edinburgh, or UoE), Dr Paul Holland (BAS), and Dr David Ferreira (Reading University). The technical work, performed by Dr Sudipta Goswami (BAS) and Dr Gavin Pringle (UoE), was undertaken between mid-February and mid-December, 2015. A recent version of MITgcm, namely version c65i, has been ported to ARCHER for all ARCHER users.

1.1 MITgcm

The MITgcm is a flexible, open-source general circulation model that is widely used by the global scientific community (including many in the ARCHER community) to study the ocean, atmosphere, cryosphere, and climate. It has been run on numerous HPC platforms (e.g. ARCHER, HECToR, NASA Columbia, NASA Pleiades, NSF Yellowstone) in a wide variety of single and multi-core configurations (300-1000 core applications are common). Over 100 research articles published in 2013 involved MITgcm in some way. MITgcm is also highly modular, featuring a wide variety of "packages" that can be activated or de-activated depending on the experimental design. For instance, sea ice, biogeochemistry, and/or sub-grid scale mixing can be included if needed.

The method of installing MITgcm is different to most other software packages on ARCHER, a Cray XC30, in that users have access to the source code and build the specific executables depending on the target simulation. As such, MITgcm is not available via modules; however, the MITgcm source code now includes the required ARCHER build options file for use with the Cray compiler.

1.2 Technical Information

MITgcm comprises about 132,000 lines of code (mainly Fortran90). It also comprises five different sub-models (land, ocean, atmosphere, land ice and sea ice), each needing pools of parallel processes, of potentially different sizes.

Algorithmic differentiation tools

One of the most useful features of MITgcm is its adjoint modelling capability. The main code has been written in parallel with two algorithmic differentiation (AD) tools, namely TAF (http://www.fastopt.com/products/taf/taf.shtml) and OpenAD (http://www.mcs.anl.gov/OpenAD/), that take large, frequently-updated numerical code as input and produce "differentiated" code as output. The differentiated code can be used for sensitivity analysis, optimisation, and state estimation. It is worth noting that, as TAF and OpenAD produce compatible code, the adjoint model inherits many of the properties of the forward model, such as domain decomposition, multithreading, and choice of linear solvers (see below). Thus, gains from optimising forward model performance will directly improve adjoint model performance as well. TAF is a well-developed commercial package that produces very efficient code but is expensive to license, while OpenAD is a newer package that is less efficient but is open-source and free to use.

2 Porting MITgcm forward modelling to ARCHER

This part of the project looked to port MITgcm for forward models to ARCHER for the benefit of the ARCHER community. Three simulations were considered when porting and optimizing. These are named the Global Ocean model, the Regional Amundsen Sea model, and the Halfpipe Streamice model. (This latter model replaced the planned Pine Island Glacier model.)

A particular version of the MITgcm is referenced as its "checkpoint", and the MITgcm checkpoint employed was this work was c65i.

2.1 Porting to ARCHER

The porting process was much slower than expected for a number of reasons.

Firstly, the RAID tests on the file system can be up to 72 hours and, at that time, were run weekly. During testing, we found MITgcm would suffer slow down by at least a factor of two. It should be noted that these tests are now started on the 1st of each month. Further, the associated yet upgrade to the /work disk, held during the project, made a significant impact on performance.

Secondly, the given Global Ocean model employed a mix of two checkpoints and, as such, was found to be unsuitable as it was not able to employed for the adjoint modelling. As such, a revised Global Ocean model was created employing only c65i. For consistency, the Amundsen Sea model was also changed to employ the older c65i checkpoint.

Thirdly, as is reported below, the it was found that the MITgcm verification suite employment of our third target model, namely the Halfpipe Streamice model, had not been implemented correctly and would display a failed unit test report even if the code was ported correctly. Moreover, the suite was unable to employ the Intel compiler due to the configuration of ARCHER at that time.

2.2 Port Verification

We employed two methods of verification. Firstly, we employed MITgcm's own verification Suite, which ran a series of small unit tests automatically. Secondly, we devised a method whereby each of the three test models would print the value of a characteristic variable to the std out.

Verification Suite

Firstly, the suite runs in serial rather than in parallel. As such, the suite only ensures any port works correctly when run in serial, whereas the strength of MITgcm is its ability to harness parallel architectures. As such, the author recommends extending the verification suite to include parallel cases.

Secondly, when running the Verification Suite on the target nodes of ARCHER, the suite employs a single node and, despite employing a single core, is rightly charged for all 24 cores as all cores are removed from the users' pool; thus, our initial tests employed ARCHER's Serial Queue. However, ARCHER's Serial Queue employs different hardware/software than that employed by the parallel Compute Queues. As such, even if the suite runs successfully in the Serial Queue, we have not demonstrated that the code has been ported successfully to the target nodes where parallel simulations are run.

Finally, at the time of testing, it had yet to be determined which compiler would be employed. The example Global case employed the Cray compilers, whilst the Amundsen Sea case employed the Intel compilers, and the GNU compilers had not been investigated at this point. It was found that only the Cray compilers could be used in conjuncture with the Verification Suite and, as such, neither an Intel- nor GNU-based solution could be tested.

Given these three limitations, we generated a local solution to ensure all tested environments produced the required output. For all three cases, namely the Global case, the Amundsen Sea case, and the PETSc case, a single double precision number was printed to standard output. These figures were plotted and compared visually with figures gleamed from other previously verified ports

to other HPC systems. It was agreed that, if the results were correct 'by eye', then the port was successful.

It should be noted that one of the unit tests is based on the target Global Ocean model, however, this project's target Global Ocean model is more refined (1 degree) than that in the verification suite (3.5 degrees).

Model verification

For all three cases, namely the Global case, the Amundsen Sea case, and the PETSc case, a single double precision number was printed to standard output.

The verification method employed for all three models was to run the following command on the std output:

```
grep dynstat_sst_mean STDOUT.0000 | cut -c59-77
```

These figures were plotted and compared visually with figures gleamed from other previously verified ports to other HPC systems. It was agreed that, if the results were correct 'by eye', then the port was successful.

Measuring Performance.

It should be noted that all cases were tested for correctness as described above. When measuring performance, the simulation was run at least three times at different times of the day, and the minimum runtime was recorded for that configuration.

2.3 Determining the target Makefile

Once the models had been ported to ARCHER, we then determined the best Makefile for this project for all three models.

For this task we employed only the first two models, namely the given Global Ocean and Regional Amundsen Sea models. The given configurations employed the Intel and Cray Programming Environments, respectively. A third

configuration was generated for the Gnu programming environment, and was created based upon research into how Gnu had been used for MITgcm on other platforms.

A further three option files were created using aggressive compiler options for the three existing configurations for Intel, Cray and Gnu, giving six option files in total for investigation.

It is important to note that this options file, for the Cray Programming Environment on ARCHER, is now available by default when users download MITgcm from their website, and compile for ARCHER. This build options file is located in the source code under MITgcm/tools/build_options/ linux_ia64_cray_archer and is currently maintained by David Ferreira (University of Reading).

For the Global Ocean model, both GNU versions were extremely slow, with the 'optimised' gnu options file 20 times slower than either Cray or Intel versions. The given Intel version was fast, the optimised Intel version was slightly faster, the given Cray version was faster again, and the optimised Cray version was the fastest.

The Sea model was also tested with these 6 options file. The GNU version fails to run but, as the Global Ocean model ran so slowly with GNU, this failure was ignored. The given Intel options file is fast, whereas the 'optimised version' is slower. The given Cray options file is very slow, but the optimised Cray is comparable to the given Intel version.

As the 'aggressively' optimised Cray option files gave the best performance for both models, it was employed for the rest of this project.

2.4 Optimising the Simulations

Here we present a summary of the optimisation tests performed for this eCSE. Full testing data (e.g. additional figures, spreadsheets) are available from the authors.

Reducing the overlap

When running MITgcm, it is recommended that care is taken when setting the overlap regions before compilation.

For the Amundsen Sea model, the given overlap of 4 was correct; however, for both the Global Sea and the Halfpipe Streamice cases, the given overlap was set too large. For both these cases, the overlap was reduced from 4 to 3 and the results of both models were identical. The overlap value is set before compilation in SIZE.h via the constants OLx and OLy. Reducing the overlap by 1 for the Global Sea model gave a reduction of the memory footprint required to run, which will enable large simulations to be run at no extra cost; and a speed up of 10% of the execution of both models.

Optimum number of core for each simulation

We then determined the optimum number of cores for both the Global Sea and the Amundsen Sea models.

For both models, we ran on the smallest number of cores possible. This was used as the baseline to determine parallel efficiency. The number of cores were then doubled, and performance and parallel efficiency measured. The optimum number of cores is defined as being the largest number of cores which gives a parallel efficiency of 50% or greater.

It was found that the given Amundsen Sea model was already employing the optimum number of course, whilst the Global Sea model could double the number of cores and still retain at least 50% parallel efficiency. Using the optimum number of cores gave a speed-up factor of 2.5

Mixed-mode MITgcm

The next stage was to consider switching on mixed-mode, i.e. enabling OpenMP to run in conjuncture with MPI.

The global mesh employed by the model is distributed over MPI tasks by creating so-called sub-grids or “tiles”, which are a small sub-mesh of the global mesh. A number of these tiles are then assigned to each MPI task. The boundary mesh points in a tile are shared, i.e. ghost cells.

OpenMP is used to parallelise over tiles. Thus, if MPI task contains more than one tile, then OpenMP may be enabled. For both the given Global Sea and Amundsen Sea models, the number of tiles per MPI task is set to one.

It is surprising to note that the word OpenMP did not appear in the MITgcm user guide, thus users had some difficulty in employing mixed-mode. The MITgcm terminology is threaded or multithreaded.

To enable mixed-mode, the user must alter both the compilation environment and the runtime environment.

The compile parameters are set in SIZE.h and in the compiler options file.

For SIZE.h, the global domain is defined as Nx by Ny 2D grid, where

Nx = sNx*nSx*nPx,

Ny = sNy*nSy*nPy,

where

nPx*nPy is the total number of MPI tasks, in a nPx by nPy 2D grid

nSx*nSy is the total number of tiles per MPI task, in a nSx by nSy 2D grid

sNx*sNy is the dimension of each tile, in a sNx by sNy 2D grid.

The parameters sNx, sNy, nSx, nSy, nPLx and nPy are all set in model/inc/SIZE.h and must be set before compilation.

Before compiling, remember to switch on OpenMP compiler flag in the options file. For the Cray options file, use "-h omp". (For Intel and Gnu, use "-openmp" and "-fopenmp", respectively.)

The runtime parameters are set in eedata and in job's associated batch script.

The number of threads is set at the eedata file using nTx and nTy, where their product is equal to the number of OpenMP threads employed at runtime. The values of nTx and nTy depend on the values of sNx and sNy, respectively. For example, if a 4x1 grid of tiles is set (nSx=4 and nXy=1 in SIZE.h), then we may run with nTy=1 (only) and either nTx=1, 2 or 4 to run with 1, 2 or 4 OpenMP threads, respectively. By way of another example, say we have 2x2 tiles per MPI task, then we may run with 1 or 2 OpenMP threads with nTx=2 and Ty=1, or nTy=1 and nTy=2. For further details, please see Section 4.3.2.1 or the MITgcm User Guide.

Lastly, the batch script must be updated. To enable 4 OpenMP threads, say, set "export OMP_NUM_THREADS = 4" before the apron command, and then ensure the OpenMP aprun flag is set correctly, i.e. "-d 4".

2.4.1.1 Results

An extensive suite of tests was considered, wherein we looked at the performance of using 1x1, 1x2, 2x1, 2x2, 1x4 and 4x1 tiles per MPI task, for a large range of core counts, *without* enabling OpenMP. We also considered under-populating nodes, by using every 2nd or every 4th core. All these cases were run for a range of core counts.

Despite extensive testing, it was found the optimum manner to run both models was to employ the given settings, namely 1 tile per MPI tasks.

We then tested mixed-mode by enabling OpenMP for 1x2, 2x1, 2x2, 1x4 and 4x1 tiles per MPI task where, for example, we testing 1, 2 and 4 OpenMP threads for 1x4 tiles. It was expected that, given MITgcm is written in Fortran, that

parallelizing over 1x4 tiles would out-perform 4x1 tiles, given the benefits the former might give to cache performance; however, the opposite was found to be true: running with 4x1 tiles was faster than running with 1x4 tiles. This may be misleading, however, as both of these cases are highly inefficient.

When comparing the execution times for multiple tiles per MPI task, with and without OpenMP, and with under-populating nodes, it is important to note that under-populating nodes increases the execution times, and then enabling OpenMP increase the execution times even more. From this result, we can state that MITgcm is not memory-bound on ARCHER.

It is clear that, for these two models at least, running on ARCHER, users should *not* employ OpenMP, as the simulations run slower in all cases.

It is important to remember that the models were verified to produce the expected results for all cases benchmarked, i.e. with multiple tiles, with or without OpenMP enabled. In other words, introducing OpenMP did not change the numerical results. This means that, in the future when the number of cores per nodes increases and memory becomes a premium and codes must employ mixed-mode parallelism, it is likely that MITgcm can safely be run in such an environment.

Optimisation summary

For both the Global Ocean and Amundsen Sea models, a very large parameter sweep was made to determine the best domain decomposition configuration for each model, wherein we considered cubic, and both vertical and horizontal tiling for each MPI task's local domain. Furthermore, the use of mixed-mode was also investigated, where the code permits threaded parallelisation when each MPI task has multiple local domains. We considered 2 and 4 threads per MPI-task for the three possible local domains. It was found that the default cubic local domains performed the best. For all test cases, when multiple tiles were assigned to MPI tasks, it was found that the code slowed when doubling the number of cores and slowed further when then employing OpenMP. This indicates that the

simulations are not memory-bound and that, at present for the given cases, MPI alone should be employed.

It should be noted that all cases were tested for correctness, and benchmarking times were determined by running each case at least three times, and recording the fastest time.

Using the optimum number of cores and the improved options file, we gave the Global Sea model a speed-up of 2.5. It was found that the Amundsen Sea model was already employing the optimum number of cores, but a slight performance benefit was gained when employing changing to the Cray programming environment using our new options file.

The use of OpenMP threads was not straight-forward, given the name OpenMP does not appear in the current MITgcm User Documentation. OpenMP is referred to as threaded or multithreads execution. However, despite slowing down execution, the introduction of OpenMP does not alter the numerical results.

2.5 The Halfpipe Streamice model and the PETSc library

The ocean and ice-sheet codes involve the repeated solution of large-scale linear systems, accounting for between 25% and 80% of forward model computation.

PETSc is a leading open-source software package for the efficient parallel solution of large linear systems. Initially, an MITgcm interface had been written for the Halfpipe Streamice model to the PETSc library, and some investigation of preconditioner and solver suitability has been carried out, but had not been ported to ARCHER.

Porting to ARCHER

Given the results of this project's research, it was assumed that the best options file for the Halfpipe Streamice model is to employ one single tile per MPI task

and, as such, no OpenMP would be employed. It was found that employing more than one tile per MPI task reduces performance.

As was mentioned above, the Verification Suite instance of this model contained a bug, wherein the unit test would fail even if the port was successful. This was due to the number of iterations employed by the Verification Suite unit test was set to low.

During testing of the Verification Suite, it was found that when using the optimised Cray options file, using the -O3 flag gave extremely poor numerical results, whereas reducing the level of optimisation to -O2 was significantly better. Specifically, using -O3 with the given call to the PETSc library resulted in more iterations when compared to -O2. As such, the overall runtime was faster when using -O2. As such, it was agreed to alter the Cray options file for this model only, and replace -O3 with -O2.

Reducing the overlap

As previously stated, the given overlap for this Halfpipe Streamice model was set too large. Via testing, we reduced the overlap 4 to 3 and the results were found to be identical. The overlap value is set before compilation in SIZE.h via the constants OLx and OLy. Reducing the overlap by 1 gave a reduction of the memory footprint required to run, which will enable large simulations to be run at no extra cost, but also gave a speed up of 10%.

Testing various PETSc numerical routines

For each case under investigation, we ran the model three times and the fastest time recorded. As before, each case was checked for correctness by examining the value of dynstat_sst_mean.

Results

We tested a number of both preconditioners and matrix solvers from the PETSc library, namely Conjugate Gradient plus Block Jacobi, Conjugate Gradient plus GAMG solver, MUMPS, MICG plus Block Jacobi, BICG plus GAMG, GMRES plus

Block Jacobi, GMRES plus GAMG, and the numerical solver given as part of MITgcm itself.

For each of these eight cases, we ran on a range of core counts. To compute the parallel efficiency, we ran on the lowest number of cores possible and employed this as the base line. We then increased the number of cores until the parallel efficiency was at least 50%.

Through this extensive and time consuming processes, it was found the Conjugate Gradient plus Block Jacobi far out-performed all the other cases, with a speed up of a factor of 2.5, when compared to the given MITgcm solver when running on the optimum number of cores for the given test case.

The possibility to employ the PETSc library has now been incorporated into the MITgcm suite and a small test now forms part of the given Validation Suite. Our improved options file is employed, and users are directed to employ the Conjugate Gradient preconditioner with the Block Jacobi solver.

3 Adjoint modelling

One of the most useful features of MITgcm is its compatibility with algorithmic differentiation tools. These tools apply the chain rule to the source code, producing gradients of a specified function (sometimes called the “objective function” or “cost function”) with respect to specified variables. In particular, MITgcm includes customisable directives that control the behaviour of either TAF (a commercial tool produced by FastOpt, http://www.fastopt.com/) or OpenAD (an open source tool, available at http://www.mcs.anl.gov/OpenAD/). Both tools can produce tangent linear models (i.e. linearisations about a basic reference state) and adjoint models (the transpose of the tangent linear model).

Application: state estimation

Suppose that we have a set of oceanic observations, possibly of several different types (e.g. ARGO autonomous float profiles, ship-track data, satellite sea surface height data). The relatively sparse nature of observations means that we will naturally have significant spatial and temporal gaps in the observational record. This inevitable deficiency can be partially addressed using numerical ocean models with appropriate sets of initial conditions, boundary conditions, and parameters (e.g. vertical diffusivity, equation of state). So, how do we best combine ocean models and observations? One approach is to manually "tune" the numerical models until they agree with observations, but this is potentially time consuming and inexact. An alternative approach is to use adjoint models to bring the ocean model state into better agreement with the available observational suite. We can call the time-evolving model state that is consistent with available observations the ``optimal model trajectory", or more simply, the ``state estimate".

We can identify the optimal model trajectory by minimising a cost function *J* that measures the misfit between available observations and the model state. To put it concisely, we want to find a model input x that solves the following minimisation problem:

$$\min\left[J(\mathbf{x})\right] = \min\left(\sum_i \left[F_i^{model}(\mathbf{x}) - F_i^{data}\right]^2\right),$$

where the sum is over the squared differences between the model state and the available suite of data. In order to solve this problem, we need gradients, such as those that can be computed using adjoint methods.

The "Estimating the Circulation and Climate of the Ocean" (ECCO; http://www.ecco-group.org/) consortium was established as part of the World Ocean Circulation Experiment (WOCE) in order to create observationally-constrained, physically realistic estimates of the state and circulation of the ocean. ECCO maintains a large number of state estimate products (http://www.ecco-group.org/products.htm) that cover various spatial domains, horizontal and vertical resolutions, and timescales. The Southern Ocean State Estimate (http://sose.ucsd.edu/) is one example of a final product that has been used extensively since its release (Mazloff et al. 2010).

Application: sensitivities

We can best explain this application using an example. Suppose that we wish to estimate the effect of sea surface temperature on sea ice export. The traditional "forward" or "finite difference" approach involves perturbing the sea surface temperature at a certain place/time and by a particular amount and examining the impact on sea ice export. This approach can only reveal the impact of a single perturbation on the model state, so a complete characterisation of the sensitivity via this method requires an often impractically large number of perturbations.

An alternative approach involves calculating the full sensitivity field in the form of exact gradients. More generally, the gradient field *is* a sensitivity field (e.g. heat content, tracer concentration). The gradient approach allows us to calculate the sensitivity of one output to all inputs and is therefore much more computationally efficient than running a large suite of forward perturbation experiments.

3.1 Installing OpenAD

OpenAD is an open-source tool used for algorithmic differentiation (AD, also known as automatic differentiation) of numerical programs. AD generates differentiated code that can be used to construct an adjoint model. An adjoint model depends on the choice of cost function and control variables, among other things. As such, there is no such thing as "the" adjoint model. OpenAD is currently maintained by Argonne National Labs. The source code and documentation is available here:

http://www.mcs.anl.gov/OpenAD/

OpenAD requires the following utilities. Availability on ARCHER is listed in parentheses:

- Perl (version 5.10.0)
- Python (version 2.7.6; won't work with Python 3)
- gcc version (version 4.3.4)
- GNU Make (version 3.81)
- Subversion (optional; available on ARCHER)

Subversion is useful for checking out the latest source code from the repository. Since OpenAD uses GNU make, you should switch to the GNU module before installing (i.e. `module swap PrgEnv-cray PrgEnv-gnu`). OpenAD has been successfully tested on a login node of ARCHER:

```
uname -a
Linux eslogin001 3.0.74-0.6.8-default #1 SMP Wed May 15
07:26:33 UTC 2013 (5e244d7) x86_64 x86_64 x86_64
GNU/Linux
cat /proc/version
Linux version 3.0.74-0.6.8-default (geeko@buildhost) (gcc
version 4.3.4 [gcc-4_3-branch revision 152973]
(SUSE Linux) ) #1 SMP Wed May 15 07:26:33 UTC 2013
(5e244d7)
```

To install OpenAD on ARCHER, start in your *home* directory. Putting OpenAD in the *work* filesystem seems to create path-related issues. Download the OpenAD source code into the new directory “OpenAD":

```
svn co https://svn.mcs.anl.gov/repos/OpenAD/trunk/ OpenAD
```

You should receive a 'Checked out revision N' message. At this stage, you want to make sure that you are using the GNU compiler:

```
 module swap PrgEnv-cray PrgEnv-gnu
```

Next, enter the OpenAD directory and use one of the scripts there to set paths and environment variables. There are scripts for both the C shell family (.csh) and the Bourne shell family (.sh):

```
cd OpenAD
source ./setenv.sh
```

You can check that the environment variable OPENADROOT has been set using “echo $OPENADROOT”. Use the update utility to download all recent updates, just in case. Even if you have just downloaded OpenAD, this process can be rather slow.

```
openadUpdate
```

Set the environment variables again (just in case) and build OpenAD:

```
source ./setenv.sh
make
```

The build process is very slow. Using `make -j N` to speed this up produces errors almost immediately. Set environment variables again just in case:

```
source ./setenv.sh
```

If you run into errors, make sure that you have the gnu module loaded using the command module list.

MITgcm Packages

3.1.1.1 Compatibility

Not every MITgcm package is compatible with OpenAD. Below is an incomplete list of packages that are not yet compatible with OpenAD:

- kpp (might work, but can be troublesome)
- layers
- exf (the external forcing package)
- cal (the calendar package)

3.1.1.2 Adjoint Model Required Packages

OpenAD uses several different packages to construct an adjoint model. These packages must be listed in packages.conf (in the code directory) at compile time. If you want to apply non-default compile time parameters for one or more of these packages, then the associated header files must be present in the code directory at compile time. Below is a brief description of the packages used for algorithmic differentiation.

- autodiff : support for algorithmic differentiation, active file handling
- cost : the cost function is defined here. Simple test cost functions are included (e.g. heat transport across 26N)
- ctrl : control variables are defined here. The cost function is differentiated with respect to these variables.
- grdchk : gradient check package; used to verify that the adjoint gradients agree well with gradients calculated using finite differences.

Some additional files are needed in the code directory to handle the conflicts between what TAF normally wants and what OpenAD wants as input. Additional files needed:

- code_ad_diff.list
- code_ad_extra.list (empty, historical)
- dontCompile
- dontTransform
- findAndDiff.bash
- keepOriginal

These files are needed in the code directory now (as of July 2014), but they may be moved "under the hood" later.

Compiling

In the OpenAD directory, run:

```
source setenv.sh
```

in order to set environment variables. You can check the value of the environment variable in bash using:

```
echo $OPENADROOT
```

If the oad directories are not present, create them:

```
mkdir build_oad code_oad input_oad run_oad
```

Note that run_oad is optional. To compile, run

```
../../../tools/genmake2 -oad -mods=../code_oad/ -of=(path
to options file)
```

You do not typically need to specify a separate adjoint options file. **Do not run make depend**. This is not necessary and may confuse the compiler. Next, run

```
make adAll
```

Compiling the adjoint model may take longer than compiling its corresponding forward model. Successful compilation produces an executable; both the forward and adjoint model are contained within that executable. The following make commands may also be useful. In order to remove everything in the build directory except the makefile, use

```
make CLEAN
```

If the Makefile still exists and is correct, then the command

```
make makefile
```

re-generates the Makefile using the same command.

We have successfully compiled an adjoint model using OpenAD on ARCHER using the gfortran compiler (module PrgEnv-gnu/5.1.29) and a default build options file.

An error with cb2mFiles

In response to the following error:

```
error: make: *** No rule to make target
`CD_CODE_VARS_mod.h', needed by `mitgcmuv_ad'. Stop.
```

We removed CD_CODE_VARS from cb2mFiles.

We added the following lines to the end of SIZE.h:

```
integer nobcs
parameter ( nobcs = 4)
```

In CPP_OPTIONS.h, the following was added for the OpenAD test:

```
C ADDED THIS BIT FOR OPENAD TEST ----
```

```
C o Allow full 3D specification of vertical diffusivity
#define ALLOW_3D_DIFFKR
C ------------
```

In the following chapters, we discuss two OpenAD/MITgcm test cases that cover two of the most common adjoint modelling applications:

- Global ocean state estimation
- Tracer sensitivity experiment

Choice of adjoint experiments

In the original plan for this eCSE, we intended to use different ocean configruations than the ones presented here. However, we found that some packages used in the original setup are not yet compatible with OpenAD (e.g. cal, sea ice). As an alternative, we ported a state estimation example and a sensitivty analysis example. Although the adjoint cases are different from those originally planned, the new suite gives a broader set of possible applications (i.e. state estimation problems and sensitivity analyses).

4 Global ocean test case : state estimation

Here we describe our setup and usage of the global state estimate test case on ARCHER using MITgcm with OpenAD. This test case can be used as a basis for generating state estimates. It is based on the "tutorial_global_oce_optim" verification exercise, and some of the material in this section is adapted from the MITgcm user guide for use on ARCHER. The source code and input files for this test case can be downloaded here:

http://www.archer.ac.uk/documentation/software/mitgcm/ecse_global_oce_optim.tar.gz

4.1 Overview

State estimation is a process of generating a physically consistent, observationally contrained, time-evolving ocean state. Typically this is accomplished by iteratively minimising a measure of model-data misfit by adjusting the initial conditions, surface forcing (e.g. wind stress), and other parameters. In this example, only the surface heat flux will be adjusted. The physical model features global, realistic bathymetry on a 4°x4° latitude-longitude grid.

One measure of model-data misfit is the least-squares difference, weighted by the a-priori uncertainties in the observations:

$$J = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{T_i - T_i^{lev}}{\sigma_i}\right]^2,$$

where the temperares are, respectively, the model and observed temperatures at each grid point *i*, and σ is the a-priori uncertainty. The error is only a function of depth, ranging from 0.5 K at the surface to 0.05 K at the bottom, mostly due to the much lower variability in the deep interior ocean than at the surface. The adjoint model returns sensitivities of the cost function to the net surface heat flux, i.e. $\frac{\partial J}{\partial Q_{\{net\}}}$. Using a line-search algorithm (optim.x), the net surface heat flux

is adjusted to reduce the model-data misfit *J*. This procedure is repeated iteratively until the desired convergence is achieved.

4.2 Code configuration

Under the directory *ecse_global_oce_optim*, modifications to the MITgcm source code designed to work with OpenAD are found in the directory *code_oad*. Modification that can be used with the commercial tool TAF are included in the directory *code*_ad. Finally, modifications that can be used with the "forward" (non-adjoint) code are found in the directory *code*. Many of the files in the various *code** directories are header files (*.h) in which compile-time options are set for various MITgcm packages (e.g. autodiff, cost, ctrl, ecco). The 'cb2mFiles' file is required by OpenAD, but it may be moved "under the hood" in future releases of OpenAD. If a file is not included in code_oad, then the default source code (under MITgcm/pkg/ or MITgcm/src/) is used if possible.

- AUTODIFF_OPTIONS.h
- COST_OPTIONS.h
- CPP_OPTIONS.h
- CTRL_OPTIONS.h
- ECCO_CPPOPTIONS.h
- FFIELDS.h
- GMREDI_OPTIONS.h
- SIZE.h
- SIZE.h_single
- cb2mFiles
- code_ad_diff.list
- cost_hflux.F
- cost_temp.F
- cost_weights.F
- external_forcing_surf.F
- ini_forcing.F
- packages.conf

4.2.1 AUTODIFF_OPTIONS.h

- Compile-time options for the C Preprocessor (CPP) relevant for the autodiff package. Note that when the ECCO_CPPOPTIONS.h header file is defined/used, then this options file is esentially not used (see the "ifdef" control statement in the file – if "ECCO_CPPOPTIONS.h" is defined, then all of the relevant define/undef statements are skipped). The most straightforward option is likely to use the multi-package "ECCO" approach

here, since that is how the test case has been configured.

4.2.2 COST_OPTIONS.h

- As with AUTODIFF_OPTIONS, this file is effectively not used in this configuration (see the "ifdef" control statement in the file). This model configuration is designed to work with the mutli-package ECCO header file.

4.2.3 CPP_OPTIONS.h

- Various C Preprocessor (CPP) options for the source code. This is where many parameterisations and features can be turned on/off (e.g. shortwave heating – not defined, non-hydrostatic mode – not defined because this is a hydrostatic configuration).

4.2.4 CTRL_OPTIONS.h

- As with AUTODIFF_OPTIONS.h and COST_OPTIONS.h, this file is effectively not used in this configruation. However, it should be kept, should users decide not to use the mutli-package ECCO approach.

4.2.5 ECCO_CPPOPTIONS.h

- In this header file, compile-time options for several different packages (autodiff, cost, ctrl, ecco, cal, and exf).

- In the "cost function package" section of this header file, various cost function terms can be turned on/off.

- In the "control vector package" section, various controls can be turned on/off. The cost function will be differentiated with respect to each control variable.

4.2.6 FFIELDS.h

- Model forcing fields (this is a bit of a work-around, as OpenAD does not yet work with the external forcing [exf] package).

4.2.7 GMREDI_OPTIONS.h

- CPP options for the eddy flux parameterisation package (gmredi). Here various clipping/tapering options can be set.

4.2.8 SIZE.h

- This file controls MITgcm's domain decomposition, which is set at compile time. The model domain is split into tiles that can then be handled by different MPI processes.

4.2.9 SIZE.h_single

- In order to use this configuration in serial mode (single process), replace SIZE.h by this file (make sure you save SIZE.h elsewhere first).

4.2.10 cb2mFiles

- This file is required by OpenAD, but in future releases it may be moved "under the hood". The order of packages/parameters in this file **does** matter, so modify at your own risk!

4.2.11 code_ad_diff.list

- Instructions for the AD tool regarding which files to differentiate.

4.2.12 cost_hflux.F

- This is where the heat flux term of the cost function is defined/computed.

4.2.13 cost_temp.F

- This is where the temperature error term of the cost function is defined/computed.

4.2.14 cost_weights.F

- Sets weights used in the cost function (e.g. weighting cost function terms by a-priori errors) and set how sensitivities are normalised.

4.2.15 external_forcing_surf.F

- Determines the forcing terms based on external fields (e.g. relaxation terms).

4.2.16 ini_forcing.F

- Sets model initial forcing fields.

4.2.17 packages.conf

- Various MITgcm packages can be turned on/off here. Note that the "monitor" package has been turned off, as it can cause problems with OpenAD.

- **openad, autodiff, cost, ctrl** : required for algorithmic differentiation
- **grdchk** : performs a finite-difference check on the gradient to ensure accuracy. This should be used during testing
- **gfd** : a collection of geophysical fluid dynamics packages (required)

- **cd_code** : allows for a hybrid C-D model grid with C-grid model variables augmented with D-grid velocity variables
- **gmredi** : Gent-McWilliams/Redi sub-grid scale eddy paramerisation

Run-time options
Also under the directory *ecse_global_oce_optim*, run-time options are set in various files in the input_oad directory:

- Err_hflux.bin
  - Binary file containing the heat flux errors

- Err_levitus_15layer.bin
  - Binary file containing errors relative to Levitus temperature climatology

- cycsh
  - c-shell script for first optimisation step

- cysh
  - bash script for first optimisation step

- data
  - continuous equation parameters, elliptic solver parameters, time stepping parameters, gridding parameters, and input datasets

- data.autodiff
  - run-time options for the AUTODIFF package.

- data.cost
  - run-time options for the cost package. Includes multipliers (mult) for the "temp" and "hflux" terms of the cost function and "lastinterval", the interval over which the cost function is averaged – the cost function is evaluated at the end of the run

- data.ctrl
  - set control variable file names

- data.gmredi
  - eddy flux parameterisations for the Gent-McWilliams/Redi scheme

- data.grdchk
  - gradient check options (e.g. variable to perturb for accuracy test, location of perturbation)

- data.optim
  - offline optimisaiton parameters (e.g. "optimcycle" is the iteration number of the optimisation procedure)

- data.pkg
  - turn on/off packages at run time. Note that a package must be included

at run time (in code_ad/packages.conf) in order to be turned on

- eedata
  - multi-threading options (set to one thread per process at the moment)

- lev_t_an.bin
  - annual mean Levitus temperature file

- prepare_run
  - script that links other input files at run-time (linked from 'tutorial_global_oce_latlon')

4.3 Compiling and running the model

Compiling. This state estimation example can be compiled using the script included in the appendix of this report. Here we present a brief description of the steps required for compilation. This assumes that OpenAD has already been compiled in the user's home directory. Note that one should *not* need to use the same compiling environment for building both OpenAD and the MITgcm case. OpenAD typically works best when compiled in the Gnu environment, whereas MITgcm can be compiled in one's preferred environment. We follow the ARCHER-recommended procedure of compiling on the backed-up "home" filesystem and running jobs on the "work" filesystem.

Issue with C++ library links

We found that the OpenAD component "whirl2axif" had difficulty finding the C++ libraries at compile-time. Specifically, we recived error messages like this one from the whirl2axif component:

```
/usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.20' not found
```

We found it necessary to manually link the C++ libraries before compiling the MITgcm-OpenAD adjoint case. So before running "genmake", try running:

```
export LD_LIBRARY_PATH=/opt/gcc/4.9.2/snos/lib64:$LD_LIBRARY_PATH
```

This is normally not required on ARCHER, as library locations are handled by the "modules" approach. However, the OpenAD build process (or at least the underyling Open64 component) are hard-coded to use 'cc' for C compiling and 'g++' for C++ compiling. Testing by Iain Bethune (EPCC) indicates that this linking issue may only affect the compiling process when using the Intel compiler / programming environment – caution and further testing is advised.

Issue with active files

During production we ran into a declaration/definition error:

```
ftn -O3 -ftz   -c -fixed the_model_main_cb2m.f90
the_model_main_cb2m.f90(869): error #6535: This variable or component must be of a
derived or structure type   [XX_GENARR3D]
      foo=xx_genarr3d(:,:,:,:,:,ik)%d
------------^
the_model_main_cb2m.f90(869): error #6460: This is not a field name that is defined in the
encompassing structure.   [D]
      foo=xx_genarr3d(:,:,:,:,:,ik)%d
---------------------------------------^
the_model_main_cb2m.f90(869): error #6158: The structure-name is invalid or is missing.
[XX_GENARR3D]
      foo=xx_genarr3d(:,:,:,:,:,ik)%d
------------^
compilation aborted for the_model_main_cb2m.f90 (code 1)
make[1]: *** [the_model_main_cb2m.o] Error 1
```

This error is an issue that has been resolved by ongoing OpenAD development. In the early versions of the code, OpenAD did not have the ability to read/write "active files", i.e. the xx_* files that play a role in the differentiation process. The temporary, ad-hoc solution was to comment out a section of "the_model_main.F" as follows:

```
560:cph      do ik = 1, maxCtrlArr3D
561:cph        foo=xx_genarr3d(:,:,:,:,:,ik)%d
562:cph        write(fname,'(A,I2.2,A)') 'adxx_genarr3d_',ik,'.'
563:cph        call write_fld_xyz_rl(fname,suff,foo,myIter,1)
564:cph      enddo
599:cph-- ad hoc fix for OpenAD time stepping counter lagging one step
600:cph-- after final adjoint step
```

MITgcm after checkpoint c65i has the ability to write active variables (e.g. xx_genarr2D), so the commented-out section of code is now obsolete. This change has been implemented in more recent versions of MITgcm/OpenAD, so it should not be necessary to repeat this step (still, it is worth checking that your version does not have the above obsolete section of code.

Running. This state estimation example can be run using “input_oad/cysh” (or “input_oad/cycsh” for c-shell users) as a template. First, run the “zeroth” step of the state estimation procedure by running mitgcmuv_ad. The output should include “ecco_cost_MIT_CE_000.opt0000” and “ecco_ctrl_MIT_CE_000.opt0000”, which contain gradient/cost function information in a compressed format. Note that the optimisation procedure also requires the “optim.x” script, which must be compiled separately. The source code for “optim” is found in “MITgcm/optim” and “MITgcm/lsopt”.

Optimisation. Optimisation is carried through from the forward model optimisation (i.e. the new build options file), as adjoint-specific options for optimisation (e.g. changing the balance of storage vs recompuation) are limited in OpenAD.

5 Adjoint test case : tracer sensitivity

Here we describe our setup and usage of the tracer sensitivity test case on ARCHER using MITgcm with OpenAD. This test case can be used as a basis for calculating gradients in the context of sensitivity analysis. It is based on the "tutorial_tracer_adjsens" verification exercise, and some of the material in this section is borrowed from the MITgcm user guide for use on ARCHER (available via http://mitgcm.org/). The source code and input files for this specific test case can be found here:

http://www.archer.ac.uk/documentation/software/mitgcm/ecse_tracer_adjsens.tar.gz

5.1 Overview

In this numerical experiment, the cost function is the total amount of tracer outgassed from the ocean into the atmosphere at each timetep. The evolution of this passive tracer C is governed by:

$$\frac{\partial C}{\partial t} = -U \cdot \nabla C - \mu C + \Gamma(C) + S,$$

where U is a combination of the mean circulation and parameterised eddies (using the Gent-McWilliams scheme (Gent and Mcwilliams 1990)), $1/\mu$ is a relaxation timescale, Γ is a convective mixing function, and S is a source term. The outgassing timescale $1/\mu$ is set to 1 year in the surface ocean and μ=0 elsewhere. The cost function measures the total outgassing from the ocean to the atmosphere at each timestep:

$$J(t = T) = \int_{t=0}^{T} \int \mu C \, dA \, dz \, dt$$

where the spatial integral is carried out over the entire ocean surface volume (area A and thickness dz). Only the upper grid cell is included in this integral. In the terminology of the adjoint approach, the above cost function "accumulates" cost over the entire surface area A up to time T. Here we examine the sensitivity of the outgassing J to the spatial location of the source S in order to identify

regions where carbon dioxide may be sensitive to rapid re-emission back into the atmosphere after sequestration (Hill 2004).

5.2 Model configruation

The model has constant 4°x4° horizontal resolution on a latitude-longitude grid, realistic geometry, and realistic bathymetry. It features 20 vertical levels with vertical sizes from 50 m near the surface to 815 m at depth. It is driven by steady climatological (i.e. long-term mean) wind stress, and uses relaxation to prescribed salinity and potential temperature profiles with a relaxation timescale of roughly one month.

5.3 Code configuration

Under the directory *ecse_tracer_adjsens*, modifications to the MITgcm source code designed to work with OpenAD are found in the directory *code_oad*. Many of these are header files (*.h) in which compile-time options are set for various MITgcm packages (e.g. autodiff, cost, ctrl, ecco). The 'cb2mFiles' file is required by OpenAD, but it may be moved "under the hood" in future releases of OpenAD. If a file is not included in code_oad, then the default source code (under MITgcm/pkg/ or MITgcm/src/) is used if possible.

- AUTODIFF_OPTIONS.h
- COST_OPTIONS.h
- CPP_OPTIONS.h
- CTRL_OPTIONS.h
- CTRL_SIZE.h
- ECCO_CPPOPTIONS.h
- GAD_OPTIONS.h
- GMREDI_OPTIONS.h
- PTRACERS_SIZE.h
- SIZE.h
- SIZE.h_single
- cb2mFiles
- ctrl_map_ini_genarr.F
- packages.conf
- ptracers_forcing_surf.F

5.3.1 AUTODIFF_OPTIONS.h

- Compile-time options for the C Preprocessor (CPP) relevant for the autodiff package. Note that when the ECCO_CPPOPTIONS.h header file is defined/used, then this options file is esentially not used (see the "ifdef"

control statement in the file – if "ECCO_CPPOPTIONS.h" is defined, then all of the relevant define/undef statements are skipped). The most straightforward option is likely to use the multi-package "ECCO" approach here, since that is how the test case has been configured.

5.3.2 COST_OPTIONS.h

- As with AUTODIFF_OPTIONS, this file is effectively not used in this configuration (see the "ifdef" control statement in the file). This model configuration is designed to work with the mutli-package ECCO header file.

5.3.3 CPP_OPTIONS.h

- Various C Preprocessor (CPP) options for the source code. This is where many parameterisations and features can be turned on/off (e.g. shortwave heating – defined but not used here, non-hydrostatic mode – not defined because this is a hydrostatic configuration).

5.3.4 CTRL_OPTIONS.h

- As with AUTODIFF_OPTIONS.h and COST_OPTIONS.h, this file is effectively not used in this configruation. However, it should be kept, should users decide not to use the mutli-package ECCO approach.

5.3.5 CTRL_SIZE.h

- Defines the maximum number of generic control variables. This is defined at compile-time for memory allocation purposes.

5.3.6 ECCO_CPPOPTIONS.h

- In this header file, compile-time options for several different packages (autodiff, cost, ctrl, ecco, cal, and exf).

- In the "cost function package" section of this header file, various cost function terms can be turned on/off. At present, only the "tracer" term of the cost function is enabled:

```
#undef ALLOW_COST_TEST
#undef ALLOW_COST_TSQUARED
#define ALLOW_COST_TRACER
#undef ALLOW_COST_ATLANTIC_HEAT
#undef ALLOW_COST_ATLANTIC_HEAT_DOMASS
```

The definition of the "tracer" term is the default one that ships with MITgcm, found in MITgcm/pkg/cost/cost_tracer.F. As discussed in the overview, this term represents the total tracer outgassed over the last timestep. The contributions are summed up on each tile of the domain decomposition (indexed by bi and bj), and the total cost function is summed up in cost_final.F (also in pkg/cost). It is worth checking these files in the source code to make sure that the cost function is summed up as desired.

- In the “control vector package” section, various controls can be turned on/off. The cost function will be differentiated with respect to each control variable. For instance, “THETA0” is the intial potential temperature, and “SALT0” is the initial salinity. “GENARR3D” is a generic 3D control field that can be changed at run-time.

5.3.7 GAD_OPTIONS.h

- Options for the generic advection-diffusion (gad) package.

5.3.8 GMREDI_OPTIONS.h

- CPP options for the eddy flux parameterisation package (gmredi). Here various clipping/tapering options can be set.

5.3.9 PTRACERS_SIZE.h

- The maximum number of passive tracers is set here. For this experiment, it has been set as PTRACERS_num=1.

5.3.10 SIZE.h

- This file controls MITgcm’s domain decomposition, which is set at compile time. The model domain is split into tiles that can then be handled by different MPI processes.

5.3.11 SIZE.h_single

- In order to use this configuration in serial mode (single process), replace SIZE.h by this file (make sure you save SIZE.h elsewhere first).

5.3.12 cb2mFiles

- This file is required by OpenAD, but in future releases it may be moved “under the hood”. The order of packages/parameters in this file **does** matter, so modify at your own risk!

5.3.13 ctrl_map_ini_genarr.F

- This modification to the ctrl package source code allows OpenAD to use the generic control vector approach.

5.3.14 packages.conf

- Various MITgcm packages can be turned on/off here.

- **autodiff, cost, ctrl** : required for algorithmic differentiation
- **grdchk** : performs a finite-difference check on the gradient to ensure accuracy. This should be used during testing
- **gfd** : a collection of geophysical fluid dynamics packages (required)
- **cd_code** : allows for a hybrid C-D model grid with C-grid model variables augmented with D-grid velocity variables
- **gmredi** : Gent-McWilliams/Redi sub-grid scale eddy paramerisation
- **kpp** : K-Profile Parameterisation (vertical mixing scheme)
- **ptracers** : passive tracers package
- **sbo :** special diagnostics package (not presently used at run-time)
- **timeave** : time-averaging package (not used at run-time, sometimes

does not work well with the adjoint)
- **obcs** : allows for open boundary conditions (not used here)

5.3.15 ptracers_forcing_surf.F
- precomputes the surface forcing term (due to KPP non-local transport)

Run-time options
Also under the directory *ecse_tracer_adjsens*, run-time options are set in various files in the input_oad directory:

- data
  - Parameters for physics, elliptic solver, time stepping, gridding, and the location of the input data sets

- data.autodiff
  - Blank in this implementation

- data.cost
  - Here the various cost function terms can be switched on/off at run-time using various multipliers

- data.ctrl
  - Blank in this model setup

- data.gmredi
  - Parameters for Gent-McWilliams/Redi/Visbeck parameterisaiton

- data.grdchk
  - Gradient check options (set to test 'genarr3d')

- data.optim
  - Set optimcycle=0 here (no other parameters)

- data.pkg
  - Turn on/off various packages at run-time using logical flags

- data.ptracers
  - Select advection scheme and other passive tracer parameters

- eedata
  - Multi-threading parameters (should be set =1 here).

- ncheckLev.conf
  - required for OpenAD implementation

- prepare_run
  - script that links various input files to the current directory. This script is intended for use in the run directory.

As indicated in the input_oad directory, the model initial conditions, bathymetry, and restoring files are located in the input directory:

- SSS.bin : sea surface salinity (used for restoring) (psu)
- SST.bin : sea surface temperature (used for restoring) (°C)
- salt.bin : initial 3D salinity field (psu)
- theta.bin : initial 3D potential temperature field (°C)
- topog.bin : 2D bathymetry field (m)
- windx.bin : zonal wind (m/s)
- windy.bin : meridional wind (m/s)

5.4 Compiling and running the model

Compiling. This sensitivity experiment can be compiled using the script included in the appendix of this report. Here we present a brief description of the steps required for compilation. This assumes that OpenAD has already been compiled in the user's home directory. Note that one should *not* need to use the same compiling environment for building both OpenAD and the MITgcm case. OpenAD typically works best when compiled in the Gnu environment, whereas MITgcm can be compiled in one's preferred environment. We follow the ARCHER-recommended procedure of compiling on the backed-up "home" filesystem and running jobs on the "work" filesystem.

Issue with C++ library links

We found that the OpenAD component "whirl2axif" had difficulty finding the C++ libraries at compile-time. Specifically, we recived error messages like this one from the whirl2axif component:

```
/usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.20' not found
```

We found it necessary to manually link the C++ libraries before compiling the MITgcm-OpenAD adjoint case. So before running "genmake", try running:

```
export LD_LIBRARY_PATH=/opt/gcc/4.9.2/snos/lib64:$LD_LIBRARY_PATH
```

This is normally not required on ARCHER, as library locations are handled by the "modules" approach. However, the OpenAD build process (or at least the underyling Open64 component) are hard-coded to use 'cc' for C compiling and 'g++' for C++ compiling. Testing by Iain Bethune (EPCC) indicates that this linking issue may only affect the compiling process when using the Intel compiler / programming environment – caution and further testing is advised.

Issue with active files

During production we ran into a declaration/definition error:

```
ftn -O3 -ftz   -c -fixed the_model_main_cb2m.f90
the_model_main_cb2m.f90(869): error #6535: This variable or component must be of a
derived or structure type   [XX_GENARR3D]
      foo=xx_genarr3d(:,:,:,:,:,ik)%d
------------^
the_model_main_cb2m.f90(869): error #6460: This is not a field name that is defined in the
encompassing structure.   [D]
      foo=xx_genarr3d(:,:,:,:,:,ik)%d
-------------------------------------^
the_model_main_cb2m.f90(869): error #6158: The structure-name is invalid or is missing.
[XX_GENARR3D]
      foo=xx_genarr3d(:,:,:,:,:,ik)%d
------------^
compilation aborted for the_model_main_cb2m.f90 (code 1)
make[1]: *** [the_model_main_cb2m.o] Error 1
```

This error is an issue that has been resolved by ongoing OpenAD development. In the early versions of the code, OpenAD did not have the ability to read/write "active files", i.e. the xx_* files that play a role in the differentiation process. The temporary, ad-hoc solution was to comment out a section of "the_model_main.F" as follows:

```
560:cph      do ik = 1, maxCtrlArr3D
561:cph        foo=xx_genarr3d(:,:,:,:,:,ik)%d
562:cph        write(fname,'(A,I2.2,A)') 'adxx_genarr3d_',ik,'.'
563:cph        call write_fld_xyz_rl(fname,suff,foo,myIter,1)
564:cph      enddo
599:cph-- ad hoc fix for OpenAD time stepping counter lagging one step
600:cph-- after final adjoint step
```

MITgcm after checkpoint c65i has the ability to write active variables (e.g. xx_genarr2D), so the commented-out section of code is now obsolete. This change has been implemented in more recent versions of MITgcm/OpenAD, so it should not be necessary to repeat this step (still, it is worth checking that your version does not have the above obsolete section of code.

Running. This experiment can be run using the job submission script included in the appendix of this document.

5.5 Sample output

Here we include some sample output for comparison. The model reproduces large-scale oceanographic features after spin-up (e.g. subtropical gyres, western boundary currents, equatorial countercurrent, the West Pacific warm pool, the Antarctic Circumpolar Current). Sample surface variables are shown in the figure below, taken after 2 years of run-time.

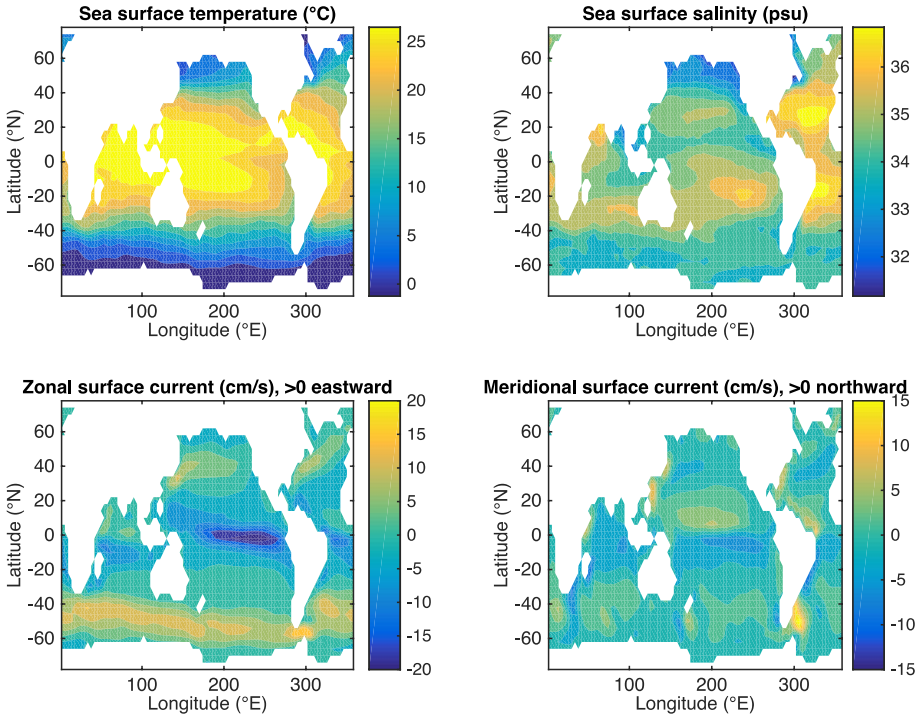


Figure X. Sea surface state after two years. The sea surface temperature (top left), salinity (top right), and current speed (bottom row) reproduce the large-scale features of the ocean state and circulaiton.

The gradient check (grdcheck) can be used to check the accuracy of the gradients computed via the adjoint method. The required accuracy depends on the application – in this application, the agreement is to 6-7 decimal places. Large discrepancies between the adjoint gradient and the finite difference gradient

may indicate non-linearities, i.e. parameter regimes where the linearisation of the model state used by the adjoint is a poor representation of the full non-linear model. This may happen, for instance, in regions with strong convection or turbulent flow.

Adjoint gradient	Finite-diff gradient
2.821722215264E+11	2.821722000000E+11
3.620541296674E+11	3.620541600000E+11
3.744790401795E+11	3.744790000000E+11
3.195323285410E+11	3.195323200000E+11
2.856230440485E+11	2.856231000000E+11

Performance relative to TAF. Performance tests indicate that the OpenAD-generated adjoint takes longer than the TAF-generated adjoint to complete the same calculation, although the difference is not as large as expected from earlier scaling tests. For the tracer sensitivty experiment, the wall clock time of the OpenAD adjoint was longer than the TAF adjoint by a factor of 2.5, and the user time was longer by a factor of about 1.4. Roughly 60% of run-time was spent in the "adjoint run" portion of the main loop.

Optimisation. Optimisation is carried through from the forward model optimisation (i.e. the new build options file), as adjoint-specific options for optimisation (e.g. changing the balance of storage vs recompuation) are limited in OpenAD.

6.0 New “divided adjoint” feature in OpenAD

Adjoint methods allow for the computation of gradients within a time that is only a very small multiple of the time needed to evaluate the underlying function itself. However, as soon as the process under consideration is nonlinear, the memory required to compute the adjoint information is in principle proportional to the operation count of the underlying function. Checkpointing strategies alleviate the high memory complexity using a small number of memory units (checkpoints) to store the system state at distinct times. Subsequently, the recomputation of information that is needed for the adjoint computation is performed using these checkpoints in an appropriate way. Several checkpointing techniques have been developed all of which seek an acceptable compromise between memory requirements and the length of the model run. The binomial checkpointing strategy implemented in OpenAD minimizes the number of time step recomputations and additionally minimizes the number of times a checkpoint is.

As part of this eCSE, we have used the binomial checkpointing approach as a basis to develop a two-level checkpointing approach that can also handle a failure of the computing system. This includes a foreseen suspension, where the application should suspend itself gracefully after completing the set number of forward or adjoint time steps. However, also an unforeseen failure is also covered, where the application is killed externally because of machine failure or expired time allocation. The system can handle a failure both in the forward sweep as well as the reverse sweep of the computation. The outer level of the approach uses regular checkpoints at a coarse granularity to support restarts. Within each coarse grained block, the binomial checkpointing approach is used (see Figure X).

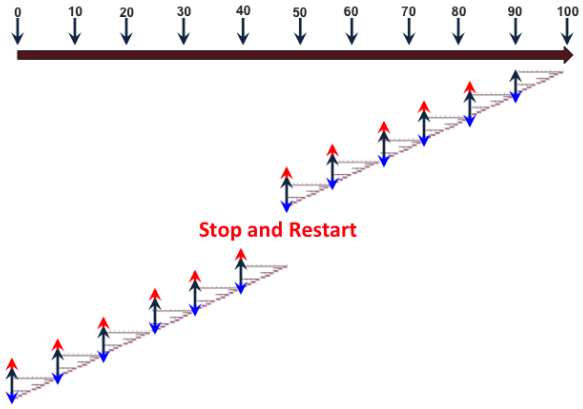


Figure X. For 100 time steps, the approach takes a state checkpoint every 10 forward timesteps. Then, in the reverse sweep, the state (and possibly adjoint) checkpoint is read, the adjoint is computed using binomial checkpointng, and the adjoint checkpoint is written.

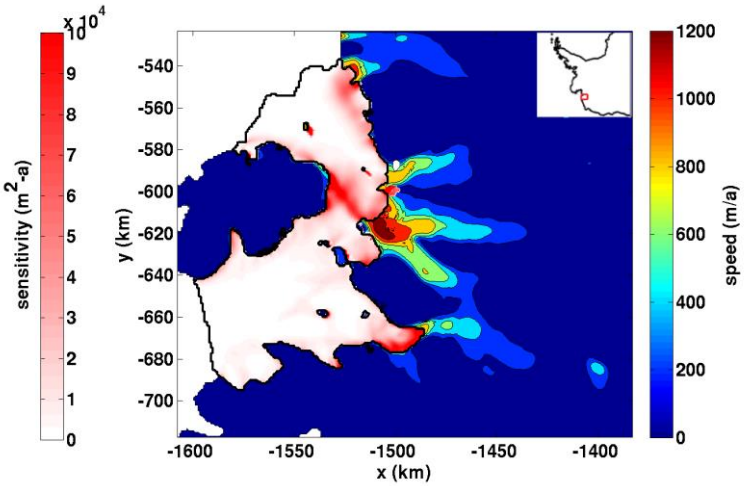
Using DIVA in an OpenAD/MITgcm run

Supposing that the original experiment was for nTimeSteps=16 with the experiment being tutorial_global_oce_biogeo from the MITgcm validation suite, we can employ the resilient checkpointing scheme in the following manner:

1. To the file ../code_oad/OPENAD_OPTIONS.h add the line '#define ALLOW_OPENAD_DIVA'
2. Change the value of nTimeSteps_l2 that is hardcoded in ./ini_parms.F to 4
3. Invode genmake2 using the command '$ ../../../tools/genmake2 -oad -mods ../code_oad -–ieee **–diva'**
4. Run 'make clean' (if already built because OPENAD_OPTIONS.h does not auto trigger)
5. Run 'make adAll'
6. Modify ../input_oad/data to make nTimeSteps = 4.
7. In the 'run' directory, run the commands '. ./prepare_run' and '../prepare_diva :'. This creates a file called ad_reg_state.000 which runs the forward as well as reverse sweep through the use of 2 runs each.

7.0 New OpenAD applications in glaciology

In our work it was discovered that the standard generated adjoint code for MITgcm's land ice component had very large memory requirements, even for moderate-sized problmes. We thus applied an optimized method to the adjoint generation, involving a special treatment of the fixed-point iteration required to solve the nonlinear stress balance (Goldberg et al. 2016). The method differs from a straightforward application of AD software, and leads to smaller memory requirements and in some cases shorter computation times of the adjoint. The optimization is done via implementation of reverse accumulation of fixed-point problems (Christianson 1994). For test problems, the optimized adjoint is shown to have far lower memory requirements, potentially enabling larger problem sizes on memory-limited machines. In the case of the land ice model, implementation of the algorithm allows further optimization by having the adjoint model solve a sequence of linear systems with identical (as opposed to varying) matrices, greatly improving performance. We have applied the method to both idealised and realistic model configurations. The methods introduced will be of value to other efforts applying AD tools to ice models, particularly ones which solve a hybrid shallow ice/shallow shelf approximation to the Stokes equations.



8 Appendix

Build options files

The MITgcm source code now includes an optimised build options file that has been produced by this eCSE project for use on ARCHER. This build options file is maintained by David Ferreira (University of Reading) and is automatically tested nightly with the most recent version of the MITgcm source code. The officially maintained build options file is designed for use with the cray compiler, and current version (as of the time of this report) is included below for completeness.

Cray build optilns file, distributed with MITgcm as the file
MITgcm/tools/build_options/linux_ia64_cray_archer

```
#!/bin/bash
#
# $Header: /u/gcmpack/MITgcm/tools/build_options/linux_ia64_cray_archer,v 1.4 2016/03/13
18:30:34 dfer Exp $
# $Name:  $

# To be used with the suite of cray compilers (PrgEnv-cray).
# To get netcdf, add:
# module load cray-hdf5-parallel/1.8.14
# module load cray-netcdf-hdf5parallel/4.3.3.1

CC='cc'
FC='ftn'
F90C='ftn'

DEFINES='-DWORDLENGTH=4 -D_BYTESWAPIO -DHAVE_LAPACK'
CPP='cpp -traditional -P'
EXTENDED_SRC_FLAG='-Mextend'
GET_FC_VERSION="-V"
CHECK_FOR_LAPACK=t

INCLUDES='-I/opt/cray/netcdf-hdf5parallel/4.3.3.1/cray/83/include                -
I/opt/cray/mpt/7.2.6/gni/mpich-cray/83/include'
LIBS='-L/opt/cray/netcdf-hdf5parallel/4.3.3.1/cray/83/lib   -L/opt/cray/mpt/7.2.6/gni/mpich-
cray/83/lib'

NOOPTFLAGS='-O0'
NOOPTFILES=''

if test "x$IEEE" = x ; then   #- with optimisation:
#   FOPTIM='-O2 -hfp3 -Oipa5' for less agressive optimization
#   Be aware not all experiments pass the restart test with optimization
   FOPTIM='-O3 -hfp3 -Oipa5'
else
```

```
   if test "x$DEVEL" = x ; then  #- no optimisation + IEEE :
     FOPTIM='-O0 -hfp0'
   else                  #- development/check options:
     FOPTIM='-O0 -hfp0'
     FOPTIM="$FOPTIM -g -Rbc -rm -hmsgs -hnegmsgs"
   fi
fi

#- For big executable, this could help:
#FFLAGS='-h pic -dynamic'
#CFLAGS='-h pic -dynamic'
```

In addition, this eCSE produced a build options file for use on ARCHER with the Intel compiler. This build options file has been tested, but it is *not* maintained.

Intel build options file - not distributed with the MITgcm source code; not maintained

```
#!/bin/bash
#
# Intel compiler with MPI

FC='ftn -O3 -ftz'
F90C='ftn -O3 -ftz'
F90FIXEDFORMAT='-fixed'
CC='cc -O3'
CPP='/usr/bin/cpp  -traditional-cpp -P'

MPI='true'
DEFINES='-DALLOW_USE_MPI -DALWAYS_USE_MPI -DWORDLENGTH=4'

LIBS='-L${CRAY_MPICH2_DIR}/lib -L${HDF5_DIR}/lib -L${NETCDF_DIR}/lib  -lnetcdf -lnetcdff  -
lhdf5 -lhdf5_hl'

INCLUDES='-I${CRAY_MPICH2_DIR}/include -I${HDF5_DIR}/include -I${NETCDF_DIR}/include -
I${HDF5_INCLUDE_OPTS}'

FFLAGS='-h byteswapio -assume byterecl -convert big_endian'
```

Finally, this eCSE produced a Gnu build options file. As with the Intel build options file, the Gnu build options file has been tested but it is *not* maintained.

Gnu build options file - not distributed with the MITgcm source code; not maintained

```
#!/bin/bash
#
# GNU compiler with MPI

FC='ftn -O3 -frecursive -funroll-loops'
F90C='ftn -O3 -frecursive -funroll-loops'
CC='cc -O3'
CPP='/usr/bin/cpp  -traditional-cpp -P'
```

```
MPI='true'
DEFINES='-DALLOW_USE_MPI -DALWAYS_USE_MPI -DWORDLENGTH=4 -D_BYTESWAPIO -DNML_TERMINATOR="/"'

FFLAGS='-m64 -fPIC -frecord-marker=4'
FOPTIM='-O3 -funroll-loops'

INCLUDES='-I${CRAY_MPICH2_DIR}/include -I${HDF5_DIR}/include -I${NETCDF_DIR}/include -I${HDF5_INCLUDE_OPTS}'

LIBS='-L${CRAY_MPICH2_DIR}/lib -L${HDF5_DIR}/lib -L${NETCDF_DIR}/lib -lnetcdf -lnetcdff -lhdf5 -lhdf5_hl'
```

Compilation and job submission scripts

Here we include the bash script used to compile adjoint model configurations of MITgcm with OpenAD. Users will need to modify the ROOTDIR variable in order to indicate the location of their MITgcm source code.

Script used to compile MITgcm on ARCHER (not adjoint case)

```
#!/bin/bash
#
# Compile MITgcm case on ARCHER
# - Designed for use with default modules as of 07/03/16
#

# Define MITgcm directory (select source code to use)
export HOMEDIR=~
export ROOTDIR=$HOMEDIR'/MITgcm_c65i'

# Select build options file
optfile=$ROOTDIR/tools/build_options/linux_ia64_cray_archer

# To enable NetCDF:
module load cray-hdf5-parallel
module load cray-netcdf-hdf5parallel

# Compile with mpi
../../../tools/genmake2 -ieee -mpi -mods=../code -of=$optfile
make depend
make
```

Script used to compile MITgcm adjoint using OpenAD on ARCHER

```
#!/bin/bash
#
# Compile MITgcm case on ARCHER using OpenAD
# - Designed for use with default modules as of 07/03/16
#

# Define MITgcm directory (select source code to use)
export HOMEDIR=~
export ROOTDIR=$HOMEDIR'/MITgcm_c65i'
```

```
# Select build options file
optfile=$ROOTDIR/tools/build_options/linux_ia64_cray_archer

# Manually link the c++ libraries
# -this step is needed, confirmed by ARCHER support (early 2016)
export LD_LIBRARY_PATH=/opt/gcc/4.9.2/snos/lib64:$LD_LIBRARY_PATH

# Set OpenAD environment variables (i.e. OPENADROOT)
workingDir=$(pwd)
cd ~/OpenAD/
source ./setenv.sh
cd $workingDir
echo "OPENADROOT is set to: "
echo $OPENADROOT

# To enable NetCDF:
module load cray-hdf5-parallel/1.8.13
module load cray-netcdf-hdf5parallel/4.3.2

# Use genmake2 to build make file
$ROOTDIR/tools/genmake2 -oad -mpi -mods=../code_oad -of=$optfile

# Use makefile to build adjoint using OpenAD
make adAll
```

Below we include a sample PBS submission script for a small (4 MPI process) run. The script was generated using ARCHER's "bolt" tool. It is executed at the command line using the *qsub* command.

PBS job submission script used for a small production run

```
#!/bin/bash --login
#
# Parallel script produced by bolt
#       Resource: ARCHER (Cray XC30 (24-core per node))
#    Batch system: PBSPro_select
#
# bolt is written by EPCC (http://www.epcc.ed.ac.uk)
#
#PBS -l select=1
#PBS -N [YOUR DESIRED JOB NAME]
#PBS -A [YOUR BUDGET CODE]
#PBS -l walltime=16:00:00

# Switch to current working directory
cd $PBS_O_WORKDIR

# Make sure any symbolic links are resolved to absolute path
export PBS_O_WORKDIR=$(readlink -f $PBS_O_WORKDIR)

# Set the number of threads to 1
# This prevents any system libraries from automatically
# using threading.
export OMP_NUM_THREADS=1

# Run the parallel program
aprun -n 4 -N 4 -d 1  ./mitgcmuv_ad
```

References

Christianson, B., 1994: Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, **3**, 311–326, doi:10.1080/10556789408805572.

Gent, P., and J. Mcwilliams, 1990: Isopycnal mixing in ocean circulation models. *J. Phys. Oceanogr*, **20**, 150–155.

Goldberg, D., S. H. K. Narayanan, L. Hascoet, and J. Utke, 2016: An optimized treatment for algorithmic differentiation of an important glaciological fixed-point problem. *Geosci. Model Dev. Discuss*, 1–24, doi:10.5194/gmd-2016-11-RC2.

Hill, C., 2004: Evaluating carbon sequestration efficiency in an ocean circulation model by adjoint sensitivity analysis. *J. Geophys. Res*, **109**, C11005, doi:10.1029/2002JC001598.

Marshall, J., A. Adcroft, C. Hill, and L. Perelman, 1997a: A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. *J. Geophys. Res*, **102**, 5753–5766.

Marshall, J., C. Hill, L. Perelman, and A. Adcroft, 1997b: Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *J. Geophys. Res*, **102**, 5733–5752.

Mazloff, M., P. Heimbach, and C. Wunsch, 2010: An Eddy-Permitting Southern Ocean State Estimate. *J. Phys. Oceanogr*, **40**, 880–899, doi:10.1175/2009JPO4236.1.