

# The MOM4 Tangent-Linear and Adjoint Model Documentation

Geoffrey Gebbie<sup>1</sup>, Benny Cheng<sup>2</sup>, Tony Lee<sup>2</sup>,  
Matthew Harrison<sup>3</sup>, Shaoqing Zhang<sup>3</sup>, Eli Tziperman<sup>1</sup>,  
Ralf Giering<sup>4</sup>, Tony Rosati<sup>3</sup>, Ichiro Fukumori<sup>2</sup>, and Andrew Wittenberg<sup>3</sup>

(1) Harvard University, (2) Jet Propulsion Laboratory,

(3) Geophysical Fluid Dynamics Laboratory, (4) FastOpt, Inc.

February 23, 2007

## 1 Overview

The tangent-linear and adjoint of the MOM4 ocean model is documented here. A “differentiated” version of the numerical code has been produced, including both the tangent-linear (forward mode) and adjoint (reverse mode). Furthermore, the TAF (Transformations of Algorithms in Fortran) tool of FastOpt is used to semi-automatically generate the differentiated code such that this code can be recomputed for any changes in the forward model (Giering and Kaminski 1998). In computer science jargon, this project could also be called the MOM4 “autodifferentiation” or “algorithmic differentiation” (AD) project (following Griewank (2000)).

The ECCO-GODAE (Estimating the Circulation and Climate of the Ocean/Global Ocean

Data Assimilation Experiment) Group is a collaboration between GFDL, Harvard, JPL, and MIT, along with Ralf Giering at FastOpt, Germany.

## 2 MOM4

The Modular Ocean Model is a representation of the hydrostatic primitive equations of the ocean. The origins trace back to Kirk Bryan, Mike Cox, and Bert Semtner between the 1960's–1980's (see Pacanowski (1996) for more details). It is supported and developed by investigators at the Geophysical Fluid Dynamics Laboratory (GFDL) in Princeton, New Jersey. MOM4 is the latest version of MOM, with its first public release in January, 2004 (Griffies et al. 2003).

For adjoint development, MOM4 version 'mom4p0d,' released in April, 2005, is being used as our “frozen” source code. All tests are conducted in the hybrid, coupled model configuration, originally designed to study the El Niño-Southern Oscillation (ENSO) phenomenon. In this configuration, the MOM4 ocean model is coupled to a statistical atmosphere (to be turned off in early development). The tripolar grid is used, and resolution is enhanced in the tropics ( $1/2^\circ$  latitudinal resolution in tropics, gradually expanding to  $5^\circ$  in the extratropics). Some specifics of the configuration follow:

- Gridpoints:  $180 \times 96 \times 25 = 4.3 \times 10^6$
- Resolution:  $2^\circ$  longitude by  $(1/2 - 5)^\circ$  latitude
- Timestep: 4 *hr* (ocean), 1 *day* (atmosphere)
- Memory (1 Process): 547 MB

### 3 Development strategy

The development strategy is summarized by two main principles: 1) modify the forward code as little as possible in order to make it differentiable, and 2) build the adjoint code in a modular way. The first principle is adopted in order to simplify the use of the adjoint model by those familiar with the forward model. The second principle is a general principle of all programming, especially in large computer codes. To proceed in a modular way, small sections of code are sent to TAF, the correctness of the differentiated code is checked, and then the process is reiterated.

#### 3.1 Detailed development guide

The step-by-step process of adding a new module to differentiated by TAF is documented here. A simple example, adding diffusion to an advection model, will be used to illustrate some key points.

Development steps: =====

1. The new module is added to the forward model.

This is a difficult step because it involves a detailed understanding of the forward model.

The new forward model must be at least semi-physical so that the forward integration is stable.

Example: Start with a 2-D advection-only forward model and add horizontal diffusion.

```
cvs -d :pserver:developername@eddy.eps.harvard.edu:/home/ecco-godae/cvsroot login
```

```
cvs -d :pserver:developername@eddy.eps.harvard.edu:/home/ecco-godae/cvsroot check-out -r 'advect_only mom4_ad
```

(Unfortunately the make script and run script may be out-of-sync in this release).

We want to differentiate the module `ocean_horz_diffuse_mod` .

Changes to forward model: add calls to `horz_diffuse_init` and `horz_diffuse_end` in `ocean_model.F90`, as well as a call to `horz_diffuse` in `ocean_tracer.F90`.

A consideration: The new module must only make calls or refer to other modules which are differentiated. TAF must see a collection of codes that are self-referential and closed. Usually, this involves commenting out calls to non-differentiated modules in the first pass.

Secondary considerations: All the active variables in `ocean_horz_diffuse_mod` must be initialized in the subroutine `ocean_model_mod::ocean_model_init`. In case we call the forward model twice (or the forward model and then the adjoint), we want to make sure that all required and active variables are properly re-initialized for each call. Most passive modules and passive variables are initialized in `ocean_nondiff.F90`, which is called by the driver. This is so that TAF does not have to differentiate these modules.

Make sure that the call for `ocean_horz_diffuse_init` is in `ocean_model_init` and that there is a complementary "end" routine, e.g., `ocean_horz_diffuse_end` in `ocean_model_end`.

2. Test step #1 by performing the "FWD" test.

The "FWD" test runs the forward model twice in one program to make sure that the code is deterministic. Non-deterministic code is non-differentiable. The main test here is to see if everything is re-initialized properly before each run.

```
cd /mom4_ad/scripts
./mk_fwd
./run_fwd
```

The test is passed if the difference between the two cost functions is identically zero.

### 3. OPTIONAL BUT RECOMMENDED. Make tangent linear code with TAF.

#### A. Invoke TAF.

The Makefile for TAF is `/mom4_ad/exec/taf_env`. The new module must be added to the list of source files that TAF will differentiate. To invoke TAF, (TAF 1.8.18 seems to be broken, so the flag `-version 1.8.17` should be used with TAF)

```
cd /mom4_ad/exec
```

```
make -f taf_env tlm
```

TAF will now produce tangent-linear code with extension `*_tl.f90` in the `/mom4_ad/exec/` directory.

Potential problems: the code will refer to other procedures that TAF can not see. These must either be commented out or a directive must be given to TAF.

B. Preprocess the TAF-generated files. In some releases, the file `/mom4_ad/scripts/tlm_preprocess.sh` will have some information on this step. Copy the TAF-generated files to `/mom4_ad/exec/tlm` which is the directory for hand-modifications. (An exception is the FMS and MPP modules. TAF produces too many errors in the differentiated code even though these are primarily passive modules! Therefore the forward FMS/MPP modules are copied to the `/mom4_ad/exec/tlm` with the new `*_tl.f90` extension.) Hand-modifications from the previously-differentiated modules will have to be repeated, or these files do not have to be overwritten. Many hand-modifications become clear during compile-time.

### 4. OPTIONAL BUT RECOMMENDED. Run the "TLM" test to check the correctness of tangent-linear model.

A. Move the source code name of the newly-differentiated module in `/mom4_ad/scripts/mk_tlm`. E.g., in the source list, `ocean_horz_diffuse.f90` should be moved into the list of tangent-linear

files with the new name `ocean_horz_diffuse_tl.f90`.

B. Compile and make hand-changes.

```
cd /mom4_ad/scripts
```

```
./mk_tlm
```

C. Do the "TLM" test – a tangent-linear gradient check.

```
./run_tlm
```

At this point, it is basically a bug-hunt through the new module in order to get the code correct. The "totalview" debugger has proved invaluable in this process.

5. Make adjoint code with TAF.

A. Invoke TAF.

The Makefile for TAF is `/mom4_ad/exec/taf_env`. The new module must be added to the list of source files that TAF will differentiate. To invoke TAF, (TAF 1.8.18 seems to be broken, so the flag `-version 1.8.17` should be used with TAF)

```
cd /mom4_ad/exec
```

```
make -f taf_env adm
```

TAF will now produce adjoint code with extension `*_ad.f90` in the `/mom4_ad/exec/` directory.

Potential problems: the code will refer to other procedures that TAF can not see. These must either be commented out or a directive must be given to TAF regarding a "black-box" routine. Directives for black-box routines go into `/mom4_ad/exec/flowdir.f90`.

B. Preprocess the TAF-generated files. In some releases, the file `/mom4_ad/scripts/adm_preprocess.sh` will have some information on this step. Copy the TAF-generated files to `/mom4_ad/exec/adm` which is the directory for hand-modifications. (An exception is the FMS and MPP mod-

ules. TAF produces too many errors in the differentiated code even though these are primarily passive modules! Therefore the forward FMS/MPP modules are copied to the /mom4\_ad/exec/tlm with the new \*\_ad.f90 extension.) Hand-modifications from the previously-differentiated modules will have to be repeated, or these files do not have to be overwritten. Many hand-modifications become clear during compile-time because they are simple syntax errors. At this point, the "TLM" code is also helpful; most of the ADM errors have an equivalent error in the "TLM" code.

6. Run the "ADM" test to check the correctness of adjoint model.

A. Move the source code name of the newly-differentiated module in /mom4\_ad/scripts/mk\_adm.

E.g., in the source list, ocean\_horz\_diffuse.f90 should be moved into the list of adjoint files with the new name ocean\_horz\_diffuse\_ad.f90.

B. Compile and make hand-changes.

```
cd /mom4_ad/scripts
```

```
./mk_adm
```

C. Do the "ADM" test – an adjoint gradient check.

```
./run_adm
```

At this point, it is basically a bug-hunt through the new module in order to get the code correct.

=====

In the example, checkout the release 'advect\_diff' to see a differentiated horizontal advection-diffusion model.

## 4 MOM4\_ad CVS repository

For ease of collaboration, a CVS repository has been created for the tangent-linear and adjoint branch. Please email 'gebbie@eps.harvard.edu' for a cvs username. To access the files, use the commands:

```
cvs -d :pserver:cvsname@eddy.eps.harvard.edu:/home/ecco-godae/cvsroot login
```

```
cvs -d :pserver:cvsname@eddy.eps.harvard.edu:/home/ecco-godae/cvsroot checkout mom4_ad
```

The adjoint-modified MPP routines are included in the GFDL CVS tree. Future plans include adding the full adjoint CVS tree to GFDL, either as a branch to MOM4, or as a parallel development tree.

## 5 Functional form of the forward model

The concept of code differentiation is most easily conceived when given a function, such as  $b = f(a)$ . Differentiated code calculates the gradient of the output,  $b$ , with respect to the input,  $a$ . How does this relate to a large GCM? For clarity, the GCM driver should call the ocean model with clearly-defined input and output arguments. This is helpful for programs which must call the ocean model more than once, and, in fact, TAF requires that the driver make a call to an external procedure which defines the influence of the input upon output. (This external procedure is the so-called “toplevel” differentiated routine. The driver routine calls the toplevel routine, but does not need to be differentiated. Therefore,

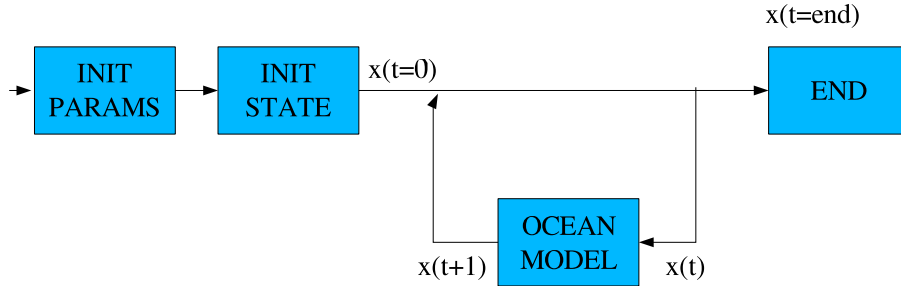


Figure 1: Schematic of the MOM4 main trunk code flow, starting from left to right. The driver calls an initialization procedure for each module. The initial ocean state,  $\mathbf{x}(t = 0)$ , is then sent through the timestepping loop. Upon exit, a few shutdown routines are executed.

the toplevel routine is the highest level in the code which must be shown to TAF.) All of the above reasons dictate that the ocean code must be written in “functional” or “black-box” form.

Figure 1 shows a schematic of the MOM4 main trunk code flow. The MOM4 driver sequentially calls the initialization routines for all the required parameters and the ocean state. Then, the code comes to the main timestepping loop. And, finally finishes with some shutdown routines. It is not written in functional form for differentiation.

Figure 2 shows a schematic of MOM4 as it has been written in functional form for AD development. The initialization of the state and the timestepping loops are all encapsulated inside the toplevel routine (i.e., the black box). Initialization of non-state variables occurs in the driver. This reorganization of MOM4 required distinguishing between differentiable

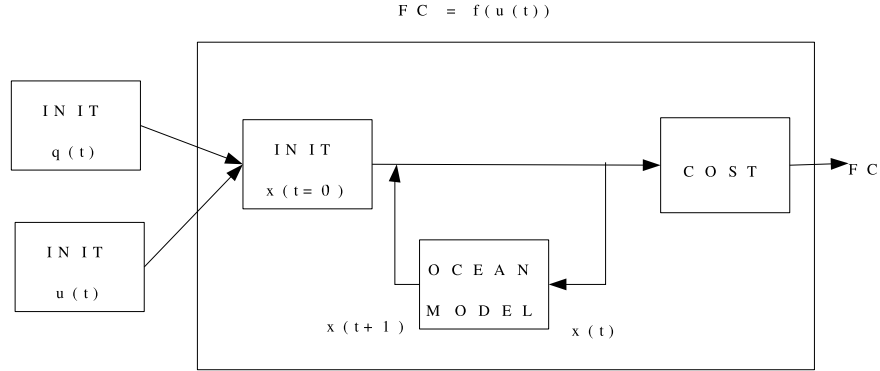


Figure 2: MOM4 flow diagram for the AD development code. Model parameters (“passive variables”),  $\mathbf{q}(t)$ , and the input variables (“control variables”),  $\mathbf{u}(t)$ , must be supplied to the ocean model in functional form. The larger black box represents the toplevel differentiable routine called from the MOM4 driver. This black box performs state vector initialization, timestepping, and computation of the output (“cost function (FC)”). In this form, the model can be easily given to TAF, multiple tests on the correctness of differentiated code can be performed, and the AD code can be developed in an incremental way.

and nondifferentiable routines, and when any new module is to be differentiated, this reorganization of the code must be reassessed. Also, notice that two new types of routines must be written, the initialization of the input (or “control”) variables  $\mathbf{u}(t)$  in the driver, and the evaluation of the output or “cost function (FC).” Omitted from the schematic but still important are the shutdown routines. MOM4 does not fully shutdown, but if one wishes to run the ocean model multiple times in one program, all shutdown routines of the differentiable portion of the code must be written. These shutdown routines have been incrementally added to the code over the last year.

## 6 Testing the differentiated code

After adding a new module to be differentiated, four numerical tests are performed.

1. Forward test for deterministic behavior.
2. Tangent-linear gradient check.
3. Adjoint test gradient check.
4. Inner-product test.

The forward test checks to see if the code is deterministic. Given a particular input (or so-called “independent” variable), the output (or “dependent” variable) is checked. In order to perform a strict check, the ocean model is run twice in the same program. In this way, we check if the initialization and shutdown of the code is complete (no corrupted or lost memory locations). For a parallel code this test is also important because the order of operations may affect whether the code is deterministic. In the case that the code is not deterministic, it is no longer differentiable.

The tangent-linear gradient check runs the forward model in a baseline state, does a second, perturbed forward model run, and a third, tangent-linear model run initialized with the perturbation from run #2. The tangent-linear model should predict the difference between runs #1 and #2.

The adjoint gradient check proceeds in the same way as the tangent-linear test, except that the adjoint-calculated gradient information is compared against two forward model runs.

Shaoqing Zhang is working on the inner-product test which tests a combination of both the tangent-linear and adjoint code without any dependence upon perturbation size.

## 7 Fortran 90 Adjoint Issues

In order to improve and implement coupled modeling and data assimilation, Fortran 90 constructs are heavily used in the code. Some Fortran 90 language constructs which pose problems for AD are: 1) derived-types (structures), 2) data encapsulation (private and public modules), 3) hardware-specific constructs (e.g., Cray pointers), and 4) pointer-valued arrays of active variables.

### 7.1 Example 1

Fortran 90 derived-types add considerable complication to the code because class definitions must be extended. Consider the assignment operator “=” written as a procedure.

```
subroutine assign(a,b)

type(derived_type), intent(in) :: a

type(derived_type), intent(out):: b

b = a

end subroutine assign
```

Clearly, assignments are very common in any numerical code. To differentiate this routine and find the adjoint function, we can follow the rules laid out by Marotzke et al. (1999).

Writing the equation in matrix form:

$$\begin{pmatrix} a_{out} \\ b_{out} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_{in} \\ b_{in} \end{pmatrix}. \quad (1)$$

The adjoint operator is defined as the transpose:

$$\begin{pmatrix} a_{out}^* \\ b_{out}^* \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} a_{in}^* \\ b_{in}^* \end{pmatrix}. \quad (2)$$

The adjoint matrix operates on the starred variables, known as the adjoint variables. Using the “\_ad” suffix for the adjoint variables, we can deduce the adjoint subroutine.

```
subroutine assign_ad(a_ad,b_ad)

type(derived_type_ad), intent(inout) :: a_ad

type(derived_type_ad), intent(inout) :: b_ad

a_ad = a_ad + b_ad

b_ad = 0

end subroutine assign_ad
```

If the variables were type “**real**,” there would be no extra complication with the adjoint subroutine. But with derived-type variables, there is some extra work to do. TAF automatically creates an adjoint derived type (“**derived\_type\_ad**”) for the adjoint variables. The operators “+” and “=” have not been defined for such a derived type, and therefore must be appended to the class definition. (Even if the adjoint variables were of the forward type “**derived\_type**” and the operator “=” already existed, the operator “+” would still need to be added.) In Fortran 90, the ensuing adjoint code is longer, more complicated, and

more error-prone than the adjoint of FORTRAN77 code. (Also, notice that other details of the adjoint code are not easily determined from the matrix structure, e.g., the `intent` declarations.)

## 7.2 Example 2

Calls to multiple functions seem like a standard Fortran 77 feature. For example,

```
subroutine horz_diffuse
```

```
L314 :  fx(:, :, k) = diff_cet(:, :, k)*FDX_PT(tracr(:, :, 1:2), k)*FMX(...)
```

There are two functions embedded in this line. TAF will transform this line into multiple lines with one operation each. To do this, TAF has to save auxiliary variables. This is where Fortran 90 allocated arrays can become a problem. If the variables `tracr` or `diff_cet` are allocatable arrays, then their complementary help arrays will be allocatable, too. TAF does not know how to properly allocate and initialize these arrays.

## 7.3 Example 3

TAF directives also can correct mistakes that TAF makes. Ralf has recently introduced the `' $!TAF PASSIVE '` directive which allows the developer to declare a variable passive. The “Grid” variable needs this designation. TAF directives are also necessary with complicated loop structures. An example is `horz_advect_tracer_2nd_order`. TAF does not recognize statements which are on the trailing side of “do” loops, but the TAF “sequential” directives fix the issue. TAF directives are also important to produce AD code which is computationally efficient.

## 7.4 Other TAF issues

Many of the problems with Fortran 90 and TAF were because of public and private modules. The “-v2” TAF flag has fixed most of these problems.

Cray pointers are used in the FMS routines and TAF can not easily use them. In addition, hexadecimal variables can not be parsed by TAF. Hand-modification is the only fix.

TAF can not handle pointer-valued function with active arguments. Therefore, these routines are handwritten. TAF should resolve this problem in the future.

## 8 Advection-diffusion test

The MOM4 adjoint has been derived for advection and diffusion of temperature and salinity. The forward, tangent-linear, and adjoint gradient checks have passed in multiple configurations. The tests have been performed with 1-timestep and multiple timestep runs. Two different cost functions have been tried: a point sample of temperature at final time, and a global sum of temperature at final time. Inner-product tests have not yet been completed.

Besides providing a strict numerical test, an advection-diffusion model and its adjoint can be understood physically. In the 1-D case of a passive tracer with homogeneous advection and diffusion, we can write:

$$\frac{\partial C}{\partial t} + U \frac{\partial C}{\partial x} - \kappa \frac{\partial^2 C}{\partial x^2} = 0, \quad (3)$$

where  $U$  is the constant advection velocity and  $\kappa$  is the diffusion coefficient. For the cost function,

$$J = \int^x f(x) C(x, t_f) dx, \quad (4)$$

it is possible to solve for the adjoint equation by the calculus of variations (see Bennett et al. 2002). Introducing  $\lambda(x, t)$  as the adjoint solution (or Lagrange multiplier), the resulting adjoint equation is:

$$-\frac{\partial \lambda}{\partial t} - U \frac{\partial \lambda}{\partial x} - \kappa \frac{\partial^2 \lambda}{\partial x^2} = f(x) \delta(t - t_f). \quad (5)$$

The physics for the evolution of the adjoint solution is nearly identical to the evolution of  $C(x, t)$ . There are a few changes to mention: 1) only the final condition for  $\lambda(x, t)$  is known, so the adjoint equation is written with time reversed, 2) advection operates upstream, and 3) there is a forcing term at final time due to the form of the cost function. Diffusion is self-adjoint and acts the same on  $\lambda(x, t)$  as  $C(x, t)$ . The main point of this paragraph is that a time-series of  $\lambda(x, t)$  can be interpreted in terms of diffusion and reverse advection – something that can be visually checked.

To perform this visual check, we use a velocity field that is spun-up from a 20-year coupled run (see Figure 3 and Figure 4). Laplacian diffusion is set to the large value of  $5 \cdot 10^4 m^2/s$ . A Gaussian perturbation to the initial temperature field is centered at the equator and  $140^\circ W$ . The evolution of this perturbation can be tracked by two forward model runs or one tangent-linear model run (Figure 5). The perturbation spreads due to diffusion, and the center of concentration of the perturbation moves downstream. The South Equatorial Current forms two branches, one north of the equator and one south of the equator, and the perturbation begins to split by this advection.

To perform the visual check of the adjoint dynamics, a particular cost function is chosen. The form of  $f(x)$  is identical to the perturbation in the forward run. In this way, the adjoint model will be “initialized” with the identical Gaussian at final time. Then, the adjoint

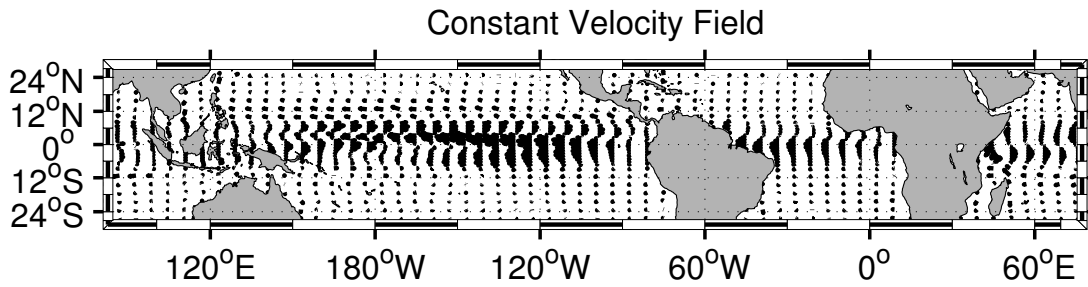


Figure 3: A global view of the velocity field for the advection-diffusion test. The velocity field is held constant for the duration of the experiment. This field results from the 20-year spinup of the coupled model.

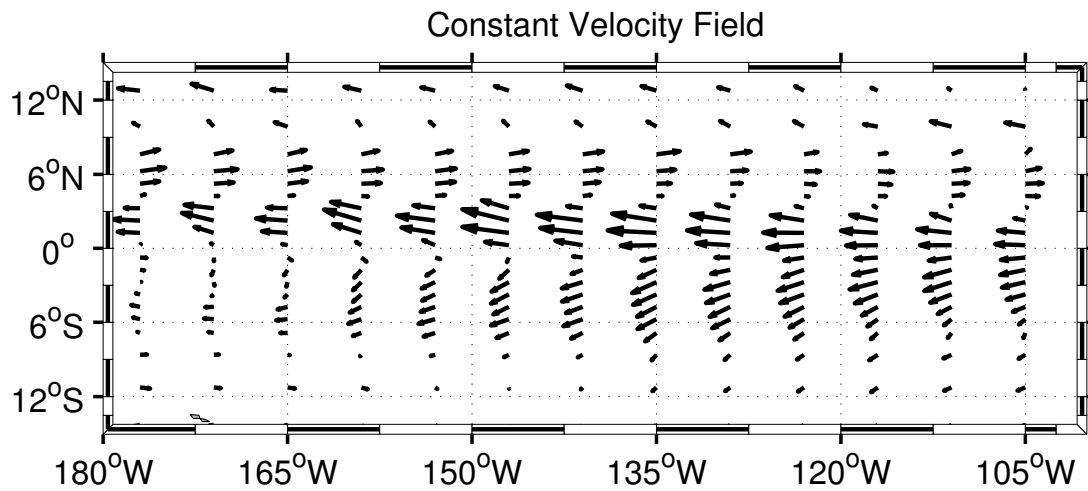


Figure 4: Same as Figure 3, except focused on the central Pacific Ocean, site of the advection-diffusion perturbation experiment.

dynamics will advect and diffuse the Gaussian. The Gaussian represents the area that influences the cost function. Figure 6 shows how the influence (or sensitivity) propagates backwards in time. The area of influence expands backwards in time due to the self-adjoint nature of diffusion. Advection, on the other hand, operates in an opposite sense. The area of influence is upstream from the site where the cost function is evaluated. The plots show that the center-of-influence moves upstream, or eastward, in the South Equatorial Current. This visually confirms that the adjoint dynamics are operating in the correct way.

## 9 Adjoint of the momentum equation

Tables 1-5 show the current status of the project. The adjoint of the baroclinic and barotropic momentum equations is completed.

## 10 Next steps

1. A first test of the complete adjoint model will be an ENSO sensitivity study following the results of Galanti and Tziperman (2003).
2. Seasonal-to-interannual forecasts and ocean reanalyses with the adjoint model may be explored.

## References

Bennett, A. F., 2002: *Inverse Modeling of the Ocean and Atmosphere*, p. 234, Cambridge University Press.

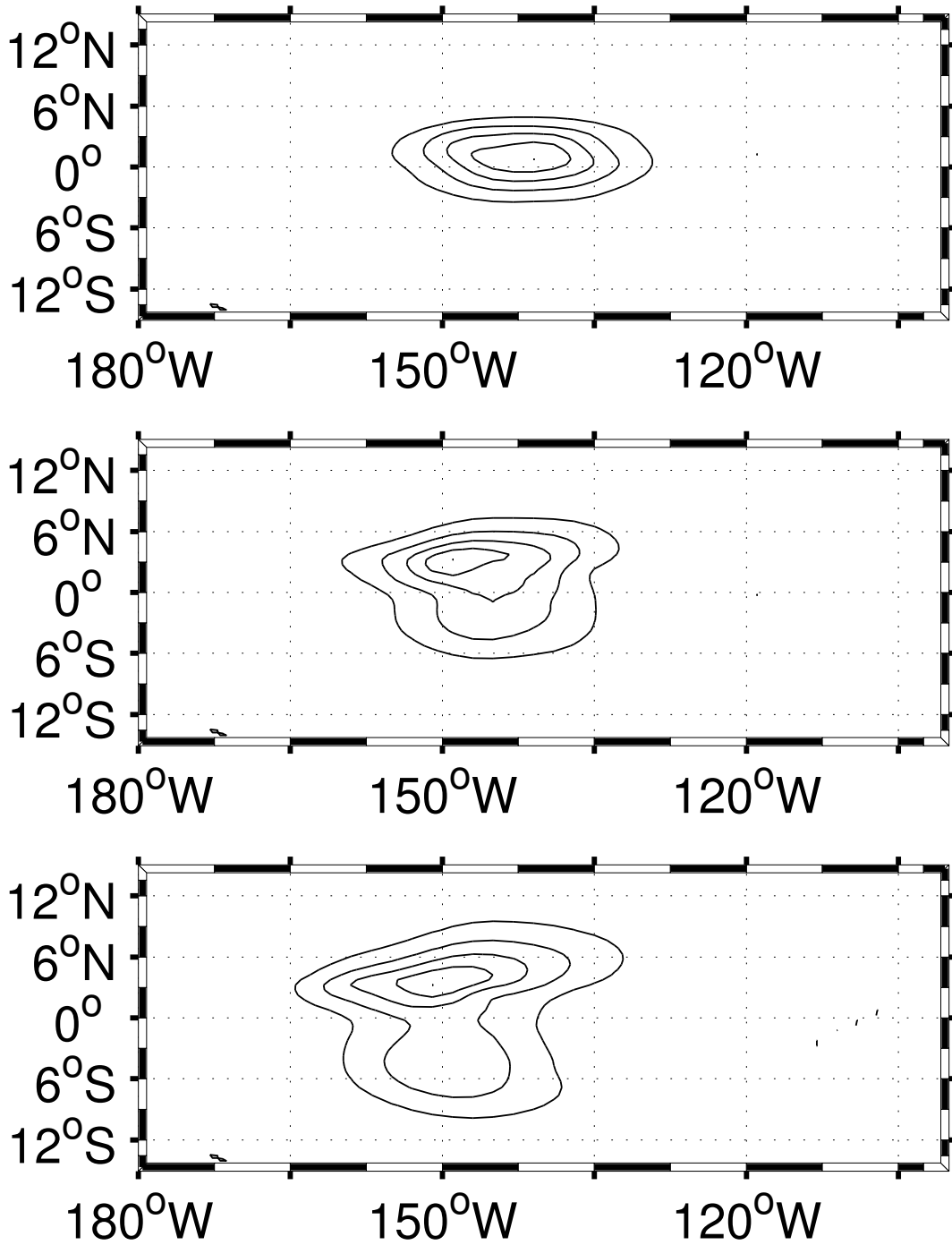


Figure 5: Snapshots of the temperature perturbation field at initial time (*top panel*), 15 days later (*middle panel*), and 30 days later (*Bottom panel*). This was calculated by taking the difference of two forward model runs. One tangent-linear run could also be used to calculate the same field – such a run gave the same results within machine precision and is not shown here.

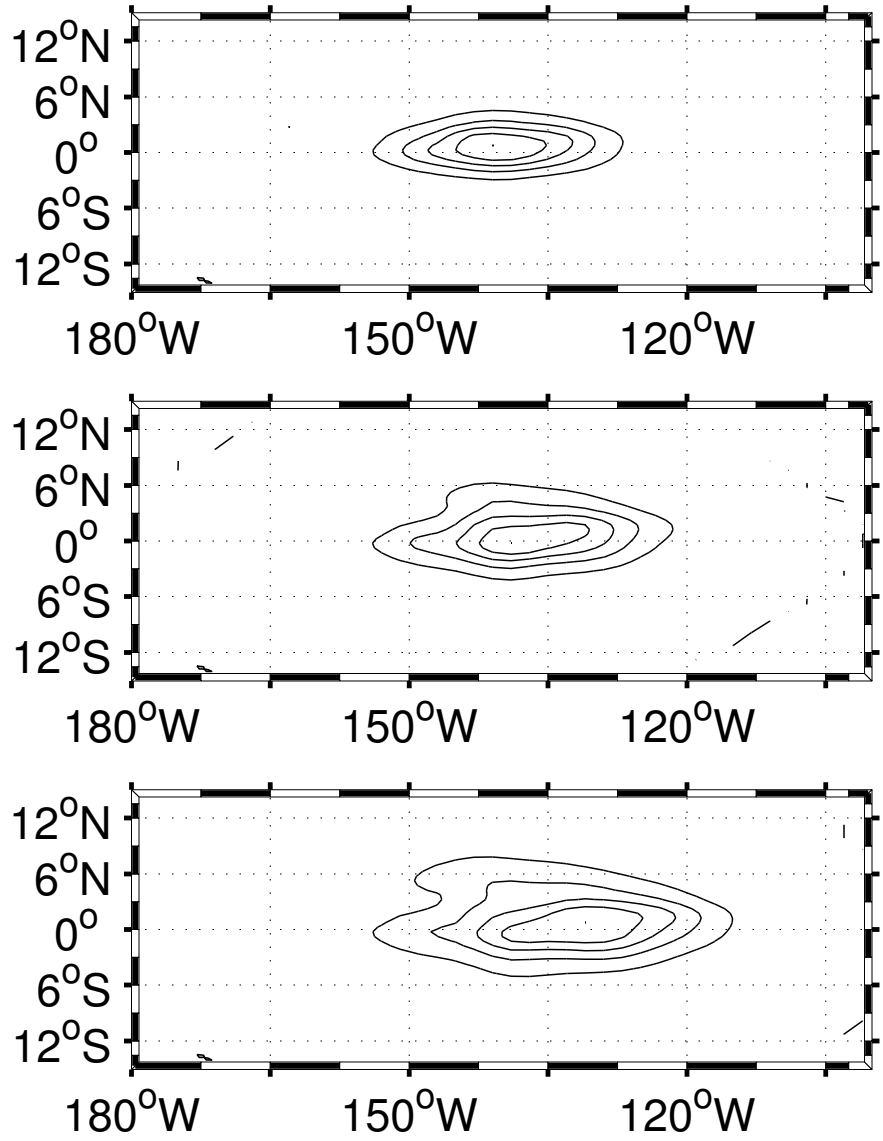


Figure 6: Sensitivity of the temperature field to the cost function at final time (*top panel*), 15 days earlier (*middle panel*), and 30 days earlier (*bottom panel*). These results were calculated by the adjoint model. The sensitivity fields plotted here are sometimes called the “adjoint state” because of the direct correspondence to the forward state. Note that the plots are arranged in reverse order relative to Figure 5.

- Galanti, E., and E. Tziperman, 2003: A mid-latitude ENSO teleconnection mechanism via baroclinically unstable Long Rossby Waves, *J. Phys. Oceanogr.*, **33**, 1877–1888.
- Giering, R., and T. Kaminski, 1998: Recipes for adjoint code construction, *ACM Trans. Math. Software*, **24** (4), 437–474.
- Griewank, A., 2000: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, p. 369, Society for Industrial and Applied Mathematics, Philadelphia.
- Griffies, S. M., M. J. Harrison, R. C. Pacanowski and A. Rosati, 2003: *A technical guide to MOM 4*, NOAA/ Geophysical Fluid Dynamics Laboratory, available online at [www.gfdl.noaa.gov](http://www.gfdl.noaa.gov), Princeton, NJ, USA 08542.
- Marotzke, J., R. Giering, K. Q. Zhang, D. Stammer, C. Hill and T. Lee, 1999: Construction of the adjoint MIT ocean general circulation model and application to Atlantic heat transport sensitivity, *J. Geophys. Res.*, **104**, 529–547.
- Pacanowski, R. C., 1996: The GFDL modular ocean model documentation, user’s and reference manual, *Tech. Rep. The GFDL Ocean Group Tech. Rep. 3.2*, p. 329, pp. 329, Geophys. Fluid Dyn., Princeton, New Jersey.

Accomplishment	Date	Primary Author
develop, deliver mom4p0b hybrid model datasets, test cases	4/2004	A. Wittenberg
TLM for Advection/Diffusion	2004	M. Harrison, S. Zhang
TLM of Neutral Physics/KPP	2004	S. Zhang, M. Harrison
Test Suite for Linearity/Gradient Check	2004	S. Zhang
ADM for Advection	10/2004	G. Gebbie
develop, deliver updated mom4p0d hybrid model	3/2005	A. Wittenberg
ADM for Parallel Communication	3/2005	B. Cheng, T. Lee
TLM/ADM Updated to MOM4p0d	3/2005	G. Gebbie
Combined Test Suite for TLM/ADM	4/2005	G. Gebbie
Updated ADM of Communication	5/2005	B. Cheng, T. Lee
ADM for Advection/Diffusion	6/2005	G. Gebbie
Parallel ADM	8/2005	B. Cheng, G. Gebbie, T. Lee
CVS Repository for MOM4-AD Branch	8/2005	G. Gebbie
Vectorized Gradient Check	8/2005	S. Zhang, G. Gebbie
New Cost/Control Package	9/2005	G. Gebbie
One-Month Gradient Check	9/2005	B. Cheng, T. Lee
ADM for External Mode Phys.	10/2005	G. Gebbie
ADM For Surface Forcing	12/2005	B. Cheng
Updated to TAF 1.8.34	4/2006	G. Gebbie
Updated to TAF 1.8.81	12/2006	G. Gebbie
ADM for Friction	1/2007	G. Gebbie
ADM for Surface Smoother	1/2007	G. Gebbie
ADM for Coriolis Force	2/2007	G. Gebbie
ADM for Baroclinic Pressure	2/2007	G. Gebbie
ADM for Full Internal Mode	2/2007	G. Gebbie

Table 1: Major accomplishments of the project.

Goal	Date	Primary Author
Optimization of Performance	2007	R. Giering, G. Gebbie
Inner-product Test	2007	S. Zhang
ENSO Sensitivity Study	2007	G. Gebbie
ADM for statistical atm.	2007	G. Gebbie, A. Wittenberg
Minimization/Optim.	2007	T. Lee
State Estimation	2007	T. Lee, S. Zhang

Table 2: Future project avenues.

Differentiated Modules	Active/Passive
cost	active
control	active
toplevel	active
ocean_freesurf	active
ocean_model	active
ocean_tracer	active
ocean_horz_diffuse	active
ocean_operators	active
ocean_thickness	active
ocean_tracer_advect	active
ocean_types	active
ocean_vert_mix	active
ocean_vert_mix_coeff	active
ocean_util	active
<b>ocean_density</b>	active
<b>ocean_sbc</b>	active
<b>ocean_advection_velocity</b>	active
<b>ocean_velocity</b>	active
<b>ocean_convect</b>	active
<b>ocean_shortwave_pen</b>	passive
ocean_workspace	passive

Table 3: Status of MOM4 modules. There are approximately 80 modules in MOM4, of which only a small proportion will need be actively differentiated, and many others which will only need to be passively parsed by TAF.)

Differentiated Modules	Active/Passive
mpp_domains	active
mpp_domains_misc	active
mpp_domains_reduce	active
constants	passive
fms	passive
fms_io	passive
memutils	passive
mpp	passive
mpp_comm	passive
mpp_data	passive
mpp_datatype	passive
mpp_domains_comm	passive
mpp_domains_util	passive
mpp_io	passive
mpp_io_connect	passive
mpp_io_read	passive
mpp_io_write	passive
mpp_io_util	passive
mpp_parameter	passive
mpp_util	passive
platform	passive
time_manager	passive

Table 4: Status of Flexible Modeling System (FMS), message passing (MPP), and shared modules.

Module	Issue	Date	Solution
ocean_model	pointer timestepping not handled	2004	forward code rewritten
various	private variables not available	2004	TAF 'v2' created
ocean_tracer	pointer-valued functions not handled		hand-written fix
fms_mod	blank module generated	4/4/2005	TAF fix
mpp_domains_mod	blank module generated	4/4/2005	TAF fix
fms_mod	blank module generated	4/4/2005	TAF fix
ocean_tracer_advect	adjoint of tracer improperly declared		TAF directive created
ocean_horz_diffuse	adjoint of tracer improperly declared	9/2005	TAF directive
ocean_model	waterflux not declared	10/2005	forward driver rewritten
ocean_model	adjoint double-initializes routine	10/2005	fwd code rewritten
ocean_model	adjoint of tracer nullified		temp. hand fix
ocean_tracer_advect	required advection variable missing	9/2005	TAF fix
ocean_horz_diffuse	improper use of allocatable arrays	9/2005	TAF + hand fix
time_manager	Extra TAF-generated routines	5/11/2005	TAF fix
ocean_advection_velocity	"	5/11/2005	TAF fix
ocean_model	Grid wrongly made active	6/23/2005	TAF fix
time_manager	help arrays wrongly declared		temp. hand fix
ocean_density	Density wrongly made active	9/2005	TAF directive
various	Modules not shutdown		additions to fwd model
mpp_domains_comm	Non-standard Cray ptr.	5/2005	hand fix
ocean_freesurf	Unneeded TLM/ADM procedures	10/2005	TAF directive
various	active "Grid" var. in comps.	10/2005	TAF directive
various mpp	Cray ptrs not parsed	10/2005	hand fix
various	TAF issues	12/2006	updated to TAF 1.8.81

Table 5: Summary of some issues solved in the MOM4 tangent-linear and adjoint development.