The G.F.D.L. Modular Ocean Model Users Guide

Version 1.0 - September 1991
G.F.D.L. Ocean Group Technical Report Number 2

Ronald C. Pacanowski
Keith W. Dixon
Anthony Rosati

Geophysical Fluid Dynamics Laboratory
Princeton, New Jersey
UNITED STATES DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration

TABLE OF CONTENTS

DISCLAIMER NOTICE

Mention of a commercial company or product does not constitute an endorsement by NOAA Environmental Research Laboratories.

Use for publicity or advertising purposes of information from this report concerning proprietary products or the tests of such products is not authorized.

The authors and NOAA Environmental Research Laboratories assume no responsibility for errors in, or incorrect use of the GFDL Modular Ocean Model. It is first and foremost an ocean modeling research tool used by researchers at NOAA's Geophysical Fluid Dynamics Laboratory. The model and this *Users Guide* have been made available as a service to the oceanographic community. No guarantees concerning the code, the documentation, nor their continued support are intended. It is left to the individual user to satisfy (him/her)self that a particular configuration is working correctly.

AVAILABILITY OF THIS DOCUMENT

Copies of *GFDL Ocean Group Technical Reports* can be obtained by writing the authors at...

Geophysical Fluid Dynamics Lab / NOAA

Princeton University, Box 308

Princeton, New Jersey 08542

Postscript files of this report may also be obtained via anonymous ftp across the internet from a GFDL server named `gfdl.gov` which has the IP address `140.208.1.9`

After making the anonymous ftp connection, the files containing this document can be found in the directory named.../`pub/GFDL_MOM/UsersGuide`

Updates to this *Users Guide* are planned, but no timetable has been set for those updates. Users of the GFDL Modular Ocean Model who have provided the authors with an e-mail address that can be reached via internet connections, will be notified as updates become available.

REFERENCING THE GFDL MODULAR OCEAN MODEL AND THIS DOCUMENT

Those who use the GFDL Modular Ocean Model in part of their research should reference the model in papers and other publications as as follows:

REFERENCE:

Pacanowski, R., K. Dixon and A. Rosati, "The GFDL Modular Ocean Model Users Guide version 1.0", *GFDL Ocean Group Technical Report #2*, (1991).

FONTS USED IN THIS DOCUMENT

This document was prepared using Frame Technology Corporation's copywrited FrameMaker workstation publishing software.

In the examples in this document, we have adopted the convention of representing user keyboard input in FrameMaker's **`bold Courier font`**, and computer responses in a `regular Courier font`.

Fortran variable names, UNIX commands and file names also appear in a `regular Courier font`

The names of GFDL Modular Ocean Model code options enabled through the use of ifdef directives are shown in a font FrameMaker calls **demibold Avant Garde.**

AVAILABILITY OF THE G.F.D.L. MODULAR OCEAN MODEL CODE

The Fortran code and related support files for the GFDL Modular Ocean Model can be obtained free of cost via anonymous `ftp` on the internet. Those wishing to obtain the model in this manner can find the files under the `/pub/GFDL_MOM` directory of the GFDL server named `gfdl.gov` (IP address `140.208.1.9`).

Example:

First, on your own machine, make a directory and a subdirectory and change to it:

`mkdir my_mom` (for the MOM files)
`cd my_mom`
`mkdir prep_data` (for the DATA files)
`cd prep_data`

Then, do the `ftp` and transfer the files:

`ftp -i 140.208.1.9`

`Name (140.208.1.9:xxx):` **`ftp`**

`Password:` **`anything`** (your last name would do nicely)

`cd pub/GFDL_MOM/PREP_DATA` (this directory contains programs for preparing data for use with the MOM code. It does not contain any datasets, and is not needed to run the model test case.)

`mget *` (transfer the DATA files)
`lcd ../` (change local directory)
`cd ../` (change remote directory)
`mget *` (transfer the MOM files)
`quit` (close the ftp connection)

On your local machine, change directories and list the contents of the directories to check that you have received the files:

NOTE: the MOM_NEWS file will change size as bug reports and other updates are added to it over time.

```
cd ../
ls -goF
total 1672
-r--r--r-- 1 19413   Apr 24 09:46  MOM_NEWS
dr-xr-xr-x 2 512     Dec 5 1990    prep_data/
-r--r--r-- 1 48290   Dec 14 1990   READ_ME
-r-xr--r-- 1 2109    Dec 14 1990   cray.run*
-r-xr--r-- 1 1524    Dec 14 1990   mergelib*
-r--r--r-- 1 555072  Dec 14 1990   mom_1.0
-r--r--r-- 1 361     Feb 2 1991    ocean.in
-r--r--r-- 1 221239  Dec 14 1990   printout
-r--r--r-- 1 3735    Dec 14 1990   splitlib.c
-r-xr--r-- 1 1188    Dec 14 1990   upgrade*
cd prep_data
ls -goF
total 157
-r--r--r-- 1 8670    Dec 14 1990   READ_ME
-r-xr--r-- 1 39      Dec 14 1990   export.batch*
-r-xr--r-- 1 1282    Dec 14 1990   export.data*
-r--r--r-- 1 14906   Dec 14 1990   ic.F
-r-xr--r-- 1 32      Dec 14 1990   ic.batch*
-r--r--r-- 1 211     Dec 14 1990   ic.i.salt
-r--r--r-- 1 211     Dec 14 1990   ic.i.temp
-r-xr--r-- 1 3418    Dec 14 1990   ic.run*
-r--r--r-- 1 6634    Dec 14 1990   prep_utl.F
-r--r--r-- 1 15068   Dec 14 1990   sbc.F
-r-xr--r-- 1 33      Dec 14 1990   sbc.batch*
-r--r--r-- 1 110     Dec 14 1990   sbc.i
-r-xr--r-- 1 2302    Dec 14 1990   sbc.run*
-r--r--r-- 1 18431   Dec 14 1990   topo.F
-r-xr--r-- 1 34      Dec 14 1990   opo.batch*
-r--r--r-- 1 194     Dec 14 1990   topo.i
-r-xr--r-- 1 2011    Dec 14 1990   topo.run*
-r-xr--r-- 1 39      Dec 14 1990   verify.batch*
-r-xr--r-- 1 1400    Dec 14 1990   verify.data*
```

PREFACE

The GFDL Modular Ocean Model is intended to be a flexible tool for exploring ocean and coupled air-sea applications over a wide range of space and time scales. This model code has been written as a collaborative effort by Ron Pacanowski, Keith Dixon and Anthony Rosati at the National Oceanographic and Atmospheric Administration's Geophysical Fluid Dynamics Laboratory in Princeton, New Jersey. It is the successor to the code written by Michael Cox, which he documented in the *GFDL Ocean Group Technical Report #1, (1984)*.

As was the case for the Cox model, the GFDL Modular Ocean Model is distributed freely to members of the oceanographic community. However, we can not certify that the code nor its documentation are free of errors. Also, we can not provide in-depth support for users of the model.

We recognize that many users will find this first version of the *GFDL Modular Ocean Model Users Guide* to be quite incomplete. We consider this *Users Guide* to be a dynamic document, and one that we will revise and expand. We also expect the model itself to continue to be improved and expanded. Our hope is that the model and documentation will grow together. Our goal is to have this *Users Guide* evolve to where it thoroughly describes the model and its options, as well as provides users with guidelines for its use.

For now, version 1 of the *Users Guide* is intended to serve as an introduction to the GFDL Modular Ocean Model version 1.0. Users unfamiliar with the earlier Cox model are referred to Cox's *GFDL Ocean Tech Report #1* and Kirk Bryan's 1969 *Journal of Computational Physics* article for a discussion of the numerics. As was the case for the Cox model and the model of Bert Semtner *(1974)* that preceded it, the GFDL Modular Ocean Model is a Fortran implementation of equations described by Bryan *(1969)*.

Comments and suggestions concerning the GFDL Modular Ocean Model are welcome. Anyone wishing to develop refinements and/or additions will be acknowledged for their efforts in subsequent releases of the code. Please contact one of us if you have something you feel might be worthwhile to add to the model or if you uncover an error in the existing code. We suggest that when developing code, the style and modularity evident in the model be followed. Its modular design is specifically intended to aid in ease of use, flexibility, and continued development without sacrificing clarity.

Ronald C. Pacanowski rcp@gfdl.gov
Keith W. Dixon kd@gfdl.gov
Anthony Rosati ar@gfdl.gov

September 1991

REFERENCES:

Bryan, K., *J. Computat. Phys., 4*, 347-376, (1969).

Cox, M. D., "A Primitive Equation, 3-Dimensional Model of the Ocean", *GFDL Ocean Group Technical Report #1*, (1984).

Semtner, A., *UCLA Dept. of Meteorology Tech. Report No. 9*, (1974)

ACKNOWLEDGEMENTS

Various aspects of the GFDL Modular Ocean Model can be associated with the efforts of many people over a number of years. Its lineage can be traced directly back in time to the codes of Michael Cox, Bert Semtner, and ultimately to the equations described by Kirk Bryan in1969.

Numerous others have added useful refinements to the GFDL Modular Ocean Model's predecessors, strengthening its heritage. We have endeavored to note contributions by citing names and references in this document and in the model code itself. No doubt some have been omitted through our own oversight or simply because the collective memory of GFDL ocean modeling can grow fuzzy over time. We apologize to those whose names have been omitted.

Of more recent importance has been the support we have received that afforded us the opportunity and freedom needed to devote time to writing the GFDL Modular Ocean Model. This continued support has come from Jerry Mahlman (Director of GFDL), Kirk Bryan, (GFDL Ocean Circulation Project Leader), Kikuro Miyakoda (GFDL Experimental Prediction Project Leader), and other GFDL colleagues.

We also wish to thank Christopher Kerr of Cray Research for his help in getting the model to autotask and multitask correctly.

This page was intentionally left blank

CHAPTER 1: Beginning to Work with the GFDL Modular Ocean Model

Once the GFDL Modular Ocean Model (MOM) files have been obtained, one naturally asks a question along the lines of... *"I've got all these files, so now what do I do with them?"*.

This chapter is intended to begin to answer the above question by giving the new user an overview of the contents of the files and suggestions regarding how to work with them.

NOTE: The following descriptions assume that the files have been arranged in UNIX directories as described earlier in this document, in the section entitled "AVAILABILITY OF THE G.F.D.L. MODULAR OCEAN MODEL CODE".

(1.1) BRIEF FILE DESCRIPTIONS

(a) mom_x ... the MOM source code in "mergelibed" form (see the entry for mergelib *(g)* below). The "x" suffix represents the version number. *i.e.:* mom_1.0)

(b) MOM_NEWS ... this file contains an up to date list of bugs and problems associated with the MOM code, along with recommended fixes, and other relevant information. This file should be consulted before a user constructs an ocean model for his/her own purposes. The bug fixes will not appear in the current mom_x code file. Bug fixes will of course be included in later released versions of the GFDL MOM code.

(c) printout ... output from a 3x4 degree idealized world ocean test case executed on a Cray YMP. Users should compare this file to results of the test case run on their own machines. For a description of the sample test case provided with the GFDL M.O.M. code, see section (1.7) at the end of this chapter

(d) ocean.in ... the input file containing "namelist" input needed to run the test case.

(e) cray.run ... a sample run deck for running the test case on a Cray YMP. While set up for the Cray YMP, this file should be consulted for running the test case on other computers as well

(f) READ_ME ... an ASCII text file containing much of the same information covered in this document.

(g) mergelib & splitlib.c (two utilities)

mergelib is a UNIX script that can be used to concatenate all code files with suffixes F, f, or h (abbreviated as *.[Ffh] files) into one big file.

```
USAGE:
mergelib [-h] [-d indir]
   [-o ofile] [file...]
   -h ; print this summary
   -d ; indir ; input directory,
     default = current directory
   -o ; ofile ; output file,
     default is standard output
   file... ; files to merge,
     default = *.[hfFc]
```

splitlib does the inverse of mergelib... breaking the mergelibed file back into its component files (*.[Ffh] files). splitlib is written in the C programming language and needs to be compiled.

```
USAGE:
splitlib [-h] [-d outdir]
   [file]
   -h ; print this summary
   -d outdir ; outdir is the
     directory that will contain
     the source files that
     splitlib will extract. The
      default is the current
     working directory.
   file ;if no file name is
     given standard input is
     assumed
```

(NOTE: The UNIX tar command can perform essentially the same functions as

`splitlib` and `mergelib`, but we provide these utilities, since there may be `tar` machine dependencies.)

(h) upgrade ... a sample script for source management on SUN workstations (for upgrading your changes to future versions of the MOM code)

(i) PREP_DATA ... is a directory which contains a collection of files (routines and rundecks) for interfacing Scripps 1 degree topography (Smith, Menard and Sharman, *[1966]*), Hellerman and Rosenstein *(1983)* wind stress, Oort *(198?)* air temperature, and Levitus *(1982)* temperature and salinity climatologies directly to the MOM code. Check the ***READ_ME*** file included in the ***PREP_DATA*** directory for details. Refer to Chapter 5 for more detailed descriptions.

REFERENCES:

Hellerman, S., and M. Rosenstein, *J. Phys. Oceanogr.*, *13*, 1093-1104, (1983).

Levitus, S.,*Climatological Atlas of the World Ocean: NOAA Prof. Paper 13,* US Govt. Printing Office, Washington DC, (1982)

Oort, A., *Global Atmospheric Circulation Statistics 1958-1973: NOAA Prof. Paper 14,* US Govt. Printing Office, Washington DC, (198?)

Smith, S., W. Menard, and G. Sharman, *World-wide ocean depths and continental elevations averaged for areas approximating one-degree squares of latitude and longitude*, *Ref. 65-80,* 14 pp., Scripps Instit. of Oceanogr., *(1966)*.

(1.2) REQUIREMENTS FOR RUNNING THE M.O.M. CODE

In addition to requiring a Fortran compiler, the C-language preprocessor (`cpp`) is required. Any computer that runs the C-language should have `cpp`, and many Fortran compilers have `cpp` functionality built into them.

The `cpp` is used stand alone or by compilers to allow sections of the GFDL Modular Ocean Model Fortran code to be selectively turned on or off for compilation via `ifdef` directives. Many compilers incorporate cpp into their processing, even if not explicitly stated. We use this feature of `cpp` to control which options and/or modules are used in any specific configuration of the MOM. code. UNIX manual pages for the `cpp` command can help users become familiar with `ifdefs`.

(1.3) EXTRACTING THE INDIVIDUAL FORTRAN SOURCE FILES

When beginning to work with the GFDL Modular Ocean Model, first create a new work directory and copy the original files into the work directory.

This is a good place to be reminded to always save your originals. They are needed when upgrading your changes to newer versions of the MOM code. We strongly recommend that users change the file permissions in the directory containing the original MOM files to "read only" (**`chmod 444 *`**).

Now, in the work directory, break `mom_x` into its component files by using the `splitlib` utility.

```
cc splitlib.c -o splitlib
   (compile    the    C-language
   program)
splitlib mom_x       (to run it)
```

Now there should be lots of files. They are classified by their filename suffix:

(a) *.F and *.f files: ....these are the Fortran source files (subroutines, main programs, and functions)

(b) *.h files:

....these are the "include" files (groups of parameter statements, common blocks, etc., that are inserted into a `*.F` file (or other files) via `#include` directives)

To merge the individual code files back into "mergelibed" form, do the following:

```
mergelib > onebigfile
```

Note that all the `*.[Ffh]` files are still left in the directory and there is a new one named `onebigfile`.

NOTE: be careful that there are no other non-MOM `.F`, `.f` or `.h` files in the directory, or they will be incorporated into the file `onebigfile`.

(1.4) GETTING ACQUAINTED WITH THE M.O.M. CODE

We recommend that after reading this documentation, you begin your exploration of the GFDL Modular Ocean Model by scanning the "`.h`" files. A convenient way to do this (under UNIX) is to use:

```
cat *.h > allh
```

This concatenates all the ".h" files together into the file named `allh`.

One can then print the `allh` file and browse through its contents. All variables contained in common blocks should be described in the comments (if we missed some, please let us know). By reading through the ".h" "Include" files, one can become acquainted with the variable names used in the model, and with some of the model organization.

Next, take a look at the model results from the test case stored in file `printout`.

Locate the section in the `printout` file which lists the model options available to the user. This list is produced by the subroutine found in file `docmnt.F`. One of the ways in which the GFDL MOM code may be customized for a particular application is by selecting various combinations of these options. The options are described in more detail in chapter 3 of this *Users Guide*.

(1.5) RECOMMENDATIONS FOR M.O.M. CODE MAINTENANCE

It is strongly recommended that source management be done on a UNIX based computer (workstation or mainframe). While we do source code management and version control using the UNIX Source Code Control System (SCCS), the method of source management is left to the individual. We've included our own `upgrade` utility based on SCCS, as an example of one way to do source management on a SUN workstation:

AN EXAMPLE USING THE "upgrade" UTILITY:

Suppose you have been working with a version of the GFDL Modular Ocean Model (let's call it `MOM.old`) and have added local changes to make `MOM.yours`. When the next version of MOM (`MOM.new`) is released, you would like to take advantage of its new features. How do you get `MOM.yours` upgraded from `MOM.old` to `MOM.new`?

If your modifications are mostly sections of code grouped together (as opposed to being strewn everywhere) you might simply "cut & paste" them into `MOM.new`. More extensive changes can be handled more or less in an automated way. An example using the `upgrade` script follows:

First, create a new working directory and copy four files into it: the mergelibed forms of `MOM.old, MOM.yours` and `MOM.new` along with the `upgrade` script itself.

Next, to run the script type **`upgrade`** and at the prompts input **`MOM.old`**,

MOM.yours and **MOM.new**. The script will produce a file named new_source which is MOM.yours upgraded to MOM.new.

The upgrade script will also likely point out places where possible problems and conflicts occurred by showing the line numbers in new_source.

After investigating and fixing all problems and conflicts (start from the highest line numbers (bottom) and work backwards in your favorite editor), you should then compare the new_source with MOM.new, as in:

diff MOM.new new_source > my.chgs

Inspect the file my.chgs to make sure that your modifications went into new_source properly.

NOTE: If code is altered in MOM.yours and the same code is altered in MOM.new but NOT in MOM.old, both the altered code from MOM.yours and MOM.new will appear in new_source.

Once one is satisfied that the upgrade has been completed and checked, the component *.[Ffh] files may be generated by using:

splitlib new_source

(1.6) NOTES ON THE M.O.M. TEST CASE:

The GFDL Modular Ocean Model code obtained from GFDL's anonymous ftp server is fully functional and ready to run a sample test case. We recommend that users attempt to run the test case on their machine of choice after they become familiar with the code.

To run the sample test case, one needs to select the set of ifdef code options listed in the sample cray.run file, and then compile the code. When successfully compiled, one should use the sample ocean.in file provided, and make a short test run that will produce output that should be compared with the sample printout file.

The test case uses an idealized configuration that permits the model resolution to be changed to fit on various computers without having to worry about land/sea geometry, model bathymetry, boundary conditions or initial conditions. All of the above automatically adapt themselves to changes in resolution. However, it should be noted that the intended purpose of the test case is to test the model code. The test case is not meant to be a scientifically meaningful configuration of the GFDL M.O.M. code.

The test case's land/sea geometry resembles the gross features of present day continental outlines. The model bathymetry, however, is unrealistic (there is a ridge in the Pacific, but most everywhere else there is a flat bottom). For the default sample provided, a 3x4

degree horizontal resolution was specified.

Initial conditions for the test case call for potential temperatures to vary with depth and latitude in a manner roughly consistent with observations. Salinity is initially constant everywhere.

Surface boundary conditions are held constant in time, in the test case. They vary with latitude, but not longitude. The surface momentum flux is given by the x and y components of the zonally averaged annual mean wind stresses (Hellerman and Rosenstein*, (1983)*. Surface salinities and temperatures are damped back towards zonally averaged, annual mean values calculated from Levitus *(1982).*

When comparing results with the sample printout file, we suggest that users first check that the land/sea geometries match. A quick look at the global surface area and volume statistics will reveal if there is good agreement. If this comparison reveals differences larger than those one might expect from roundoff errors, a check of the model bathymetry is called for in order to determine the location(s) where the sample Cray version differs from the user's run. The scheme use to determine land/sea boundaries can be sensitive to the precision or the way in which number are represented, causing variations in the model's continental outlines to arise on different machines.

REFERENCES:

Hellerman, S., and M. Rosenstein, *J. Phys. Oceanogr.*, *13*, 1093-1104, (1983).

Levitus, S.,*Climatological Atlas of the World Ocean: NOAA Prof. Paper 13,* US Govt. Printing Office, Washington DC, (1982)

This page was intentionally left blank

CHAPTER 2: Coding Design of the GFDL Modular Ocean Model

As mentioned in the ***Preface***, one of the primary goals in constructing the GFDL Modular Ocean Model (MOM) was to produce a model that was easily to configure and work with, while at the same time being flexible enough to be a research tool for various scientific problems. Increased flexibility and clarity have been achieved through modularity.

This chapter describes the ways in which the modular design and structure of the GFDL MOM code works toward this goal.

Since the GFDL MOM code is not intended to be a static product, we welcome user input as to how it may evolve to better achieve our goal of having a truly flexible and easy to use research tool.

(2.1) GENERAL CODING DESIGN

Users of previous versions of GFDL's ocean model codes will notice that there are many more subroutines in the MOM code. The modular design is intended to aid in the logical organization of the code with "hooks" readily available for adding new options to the model.

Deciding which modules to use is done at the pre-processing level. Modules do not interact with each other, but interact with the main code in only a few places through a short argument list and/or the include files. This approach to interfacing tends to localize code modifications, thereby keeping the code structure simple, understandable, and easily supplemented. Again, we strongly suggest that this approach, some of which is outlined below, be followed when users make their own code modifications.

The GFDL Modular Ocean Model code is written in standard ***Fortran-77***, except for the use of the common extension of `namelist`.

Variable organization: variables have been organized and grouped in terms of physical (or where appropriate logical) significance. This allows for better modularity. Look at the *.h files. Note that all variables are commented. To find out where variable yyy is used, use the UNIX command:

```
grep -i -n yyy *.[Ffh]
```

which searches all files with `.F, .f,` or `.h` suffixes for the string `yyy`. The use of `grep` is invaluable in tracing variables and options and its use is strongly encouraged. Note also that the include files may be nested (see the bottom of file `param.h` for example).

When modifying the code, one should consider whether it is appropriate to add new variables to existing common blocks and ".h" files, or if a new ".h" file and common block are called for to enhance modularity. When in doubt, try to minimize the interactions between various parts of the code.

No out-of-bounds references are found in the MOM code. This can simplify debugging new code by turning on a compiler's bounds checking option. If, on execution, an array subscript goes out of bounds, the compiler's bounds checker will let you know where and when this important design feature has been violated.

Statement functions are used to define physical operators. This allows one to write code that more closely resembles the equations. Another advantage is that complicated code can be made more understandable. At the same time, more operations per loop often allows a compiler to generate more efficient code. On computers like the Cray YMP, this can lead to significantly more parallelism. Look in file `tracer.F` and `clinic.F` to see how we have used them. We have also included an `ifdef` option named **`keepterms`**, that allows the use of arrays

(instead of statement functions) to retain terms in momentum and tracer equations. This can be useful when using the terms often in more than one place. The price for this is increased memory (but less computation).

Naming conventions have also been adopted, and appear as comments in the model. Conventions exist for naming numerical constants, grid variables, reciprocals, etc. (see file `pconst.h`). By adhering to these rules, one can more easily analyze unfamiliar sections of the code.

The ***slab variables*** have been defined as four dimensional variables, utilizing the concept of a memory slab window. Look at file `slabs.h` for details. This increases the generality and flexibility of the code and simplifies management of the data flow from disk through memory. In addition to longitude and depth indicators, the dimensions on the slabs include time level and j-row location "pointers". So instead of shuffling data into the "correct slab positions", the MOM code simply updates pointers to existing data

The MOM code has been written with the underlying assumption that ***memory size*** has been increasing for the types of computers that the code is typically run on. So, there are places where we trade-off extra memory usage in order to achieve clarity, modularity, and to minimize the probability of introducing errors when modifying the code in the future.

(2.2) SOME NEW FEATURES & NOTES FOR USERS OF THE COX CODE

As noted before, the GFDL Modular Ocean Model code is intended to be much more modular in design than the Cox *(1984)* model and other earlier codes. Increased modularity and an increased number of code options result in there being many more subroutines and include files in the MOM code. The simplified flow chart provided in this document (see section 2.3) should help users familiar with the Cox code to follow the flow of the MOM code. In this section, we point out some of the more substantial differences users will encounter when migrating from the Cox code to the GFDL MOM code.

Customizing via ifdef directives: Refer to the sample `printout` file and chapter 3 of this *Users Guide* for the complete list of currently allowable `ifdef` options available for customization of the model. Note that there is a **skipland** option for doing calculations only over ocean points (as opposed to everywhere), options for choosing various lateral and vertical mixing parameterizations (explicit or implicitly solved), plus a selection of Poisson solvers including a very accurate 9 point conjugate gradient scheme which allows for direct reconstruction of the rigid lid surface pressure.

Various MOM code options are controlled by `ifdef` options, rather than Cox's UPDOC program. It is recommended that users put `ifdef` directives around

personal code modules and that users try to keep modules from interacting, as in the base MOM code. Some UNIX tools for source management include: `diff`, `sdiff`, `diff3`, and SCCS. UNIX manual pages for the `cpp` command can help users become familiar with `ifdefs`.

No boundary conditions in the slabs: Note also that the slabs are defined in a way such that no boundary conditions have been added onto the slabs as in Cox's code. The vertical boundary conditions can be found in their own common block in file `cvbc.h` (common of vertical boundary conditions). By increasing the number of dimensions on the slabs to include time level and j-row location "pointers", the number of slab variable names and do loops that switch time and j-row positions have been reduced.

New and redefined variables: New variables have been added and some old ones redefined (see file `coord.h` for examples). There are also "source terms" for the momentum and tracer equations (variables `sourcu`, `sourcv` and `sourct`) which allow easy inclusion of sources and sinks (*i.e.*: biology, imposing baroclinic flows, etc.).

Consistency checks are carried out for `ifdef` options along with various other conditions by code found in file `checks.F`. Notice the "warning" and "error" messages. that are generated by `checks.F`. Warnings allow the model to procede, but error messages terminate execution.

Island calculations are done by line (rather than area) integrals and the island definition has been simplified by the automatic calculation of island perimeters (see file `iperim.F`) requiring only that one point within the island be specified. This allows for easy use of islands within complex geometries.

A time manager (see file `tmngr.F` and `ctmngr.h`) controls all time dependent decisions within the model. Time dependent information such as length of integration, time between energy diagnostics, etc., are entered through namelist in units of days (only variable `nmix`, the number of timesteps between mixing timesteps, is not in units of days). This information is used by the time manager to decide how logical variables in file `switch.h` are to be set each time step. These logical variables control time dependent data flow.

Included in `tmngr.F` are utilities for aiding in interpolating time dependent boundary conditions (*i.e.*: monthly wind stress, etc.) to the time step. However, these are not used in the test case.

Finding the nearest model grid point: Another utility is function `indp`, found in file `setgrid.F`. This function is used in many places and provides users with an easy way to map real world coordinates (latitudes, longitudes and depths) to the nearest model grid point. This allows all spatial information to be independent of changes in model resolution.

Minimize equivalence statements: In addition to outlawing out-of bounds references in order to improve the code's clarity, an attempt has been made to minimize equivalence statements. Equivalence statements should be added with caution.

Regional averages and term balances: The ability to compute volume weighted tracer averages, surface fluxes, and term balances for the momentum and tracer equations has been added to facilitate analysis and debugging. These computations can be done over arbitrary volumes in arbitrary locations. The volume sizes and locations can be specified using horizontal and vertical masks (see files `ocean.F` and `cregin.h` and option **readrmsk**).

Writing data for off-line analysis: The GFDL Modular Ocean Model code provides the user with a "`snapshot`" feature to write instantaneous data for offl-ine analysis. (see variable snaps in file switch.h and file `iounit.h` for details). Many users will want to modify or supplement this feature for their own local purposes.

Bottom drag in the MOM code is controlled by the linear drag coefficient variable `cdbot`. There is no bottom drag when `cdbot` = 0.0 (see files `scalar.h` and `blkdta.F`). As in the Cox model, lateral boundaries are no slip for momentum and no flux across boundaries for tracers.

Run-time documentation: The MOM code uses file `docmnt.F` to arrange data to be written out as a form of documentation summarizing the model characteristics that uniquely define a given model run. It also lists the requested `ifdef` options. The user will need to review this routine when configuring a model run or adding new options to the code. This feature can be beneficial to those setting up new model runs, comparing different model runs, and for analysis purposes.

REFERENCE:

Cox, M. D., "A Primitive Equation, 3-Dimensional Model of the Ocean", *GFDL Ocean Group Technical Report #1*, (1984).

(2.3) SIMPLIFIED CALLING TREE DIAGRAMS FOR THE GFDL M.O.M. CODE

These calling tree diagrams are intended to help illustrate how the coding design discussed previously has been implemented in the GFDL Modular Ocean Model code.

However, for clarity, not all calls are represented in the calling tree for the main ocean model program shown on the next page. The calls that have been omitted tend to appear often in the code (a sign of their modularity).

Specifically, calls which printout information using the code in file `matrix.F` have been omitted. Those involving the i/o routines `odam.F` and `getvar.F` have been omitted also. So too have references to function `indp`, a utility found in file `setgrd.F` that is used to find the nearest grid point to a specified location.

Calls which are conditional upon the `ifdef` options a user has defined appear within curly brackets {}.

The firstcalling tree represents the EQSTAT program found in file `denscoef.F`, which generates a `dncoef.h` file.

EQSTAT (main program for deriving coefficients of polynomial approximations for the equation of state)
{**KNUEKM**} or {**UNESCO**} (routines returning density as a function of T, S and P)
POTEM (returns potential temperature for in situ temperature, salinity and pressure)
{**KNUEKM**} or {**UNESCO**} (routines returning density as a function of T, S and P)
LSQSL2 (iteratively computes a least squares fit)

The stand alone `depths` program generates a `thick.h` file of model layer thicknesses.

DEPTHS

The stand alone `size` program determines memory and disk requirements for various configurations of the GFDL MOM code.

SIZE

The following flowchart is for the GFDL MOM code's main `ocean` program.

OCEAN (main program)
- **BLKDTA** (initialize variables)
- **GRIDS** (set up the grids: in file setgrd.F)
  - **MESH** (utility for setting up grids: in file setgrd.F)
    - **CMESH** (utility for setting up grids: in file setgrd.F)
- **DOCMNT** (documents the run)
- **OCN1ST** (set up initial conditions on the first timestep)
  - **TOPOG** (set up idealized world topography)
    - **SETKMP** (set the number of levels for the t,s grid)
- **RDREST** (read the restart file if it is not the first timestep: in file restio.F)
- {**REG1ST**} (set up for regional averages of tracers)
- {**IPERIM**} (calculate island perimeters)
- {**FINDEX**} (calculate indices for filtering)
- **CHECKS** (test for consistency of configuration)
- **TMNGR** (time manager called once per timestep)
  - **TMNSET** (function to set time dependent logical switches: in file tmngr.F)
- **STEP** (cycle latitude rows between disk & memory slab window)
  - {**DELSQ**} (calculate quantities for biharmonic mixing)
  - **SETVBC** (set vertical boundary conditions)
    - **BCEST** (get surface boundary conditions for test case)
  - {**PPMIX**} (Pacanowski/Philander richardson mixing)
    - **STATEC** (density calculation for vertical stability)
  - {**CNVMIX**} (set coefficients for constant vertical mixing)
    - {**STATEC**} (densities for vertical stability if implicitvmix: in file state.F)
  - {**TCMIX**} (Mellor-Yamada turbulence closure mixing)
    - **STATEC** (density calculation for vertical stability: in file state.F)
    - {**IMPLQ**} (for implicit vetical mixing)
  - {**ISOP0**} (Isopycnal mixing set up: in file isopyc.F)
    - **STATEC** (called 3 times for densities for isopycnal slopes: in file state.F)
  - {**CFL**} (for cfl monitoring)
  - **CLINIC** (calculate internal mode velocities)
    - **STATE** (density calculations for pressure gradients)
    - {**NLMIX**} (nonlinear Smagorinsky mixing)
    - {**INVTRI**} (for implicit vertical diffusion)
    - {**FILUV**} (filtering u & v at high latitudes)
      - {**FILTR**} (if fourfil option was selected)
      - {**FILFIR**} (if firfil option was selected)
  - **TRACER** (calculate tracers)
    - {**ISOP1**} (finish isopycnal mixing: in file isopyc.F)
    - {**INVTRI**} (for implicit vertical diffusion)
    - **STATEC** (density calculation for vertical stability: in file state.F)
    - {**FILT**} (filtering tracers at high latitudes)
      - {**FILTR**} (if fourfil option was selected)
      - {**FILFIR**} (if firfil option was selected)
    - **REGION** (periodically does basin averages of tracers)
  - **DIAG** (periodic energy diagnostic and term balance calculations)
- **VORT** (calculates vorticity)
  - {**FILZ**} (filtering vorticity at high latitudes)
    - {**FILTR**} (if fourfil option was selected
    - {**FILFIR**} (if firfil option was selected)
  - {**RELAX**}, {**HYPER**},{**CONGR5**},{**CONGR9**} (Poisson solvers for external mode)
    - **DIAG2** (periodic energy diagnostic & term balance printouts)
- **DOCMNT** (document the run)
- **RDREST** (write the restart file)

This page was intentionally left blank

CHAPTER 3: Code Options in the GFDL Modular Ocean Model

As discussed earlier, a very powerful feature of the GFDL Modular Ocean Model (MOM) is the ability to define `ifdef` options so as to configure the model in a manner suitable for the problem a user wishes to investigate. We also feel that the use of `cpp` (the C-language preprocesor) and `ifdef directives` helped us write a model in which numerous code options could co-exist and still leave the code readable and receptive for future development.

This chapter contains brief descriptions of the various MOM code customization options that a user may turn on or off through the use of `cpp` and `ifdef directives`. It is suggested that the user examine at the code itself and read the references given, in order to develop a fuller understanding of the options.

These descriptions introduce users of the GFDL MOM code to what it has to offer now. We will also briefly mention some tentative plans for future work. We invite users to sharie with us their ideas and code modules so that together the ocean modeling community can enhance this widely distributed research tool.

(3.1) OVERVIEW OF CODE OPTIONS

The options available in the MOM code are quite extensive and range from various subgrid mixing parameterizations to computing performance enhancements. The current implementation is such that if there are more than two options within a category of code options, there is no default value, so care should be taken to define the desired option.

Within the MOM code, subroutine `checks` (see the `checks.F` file) searches the selected options to determine if the user has asked for conflicting options.

For example, if a user in selecting an I/O scheme turned on both **diskless** (fully contained within memory) and **fio** (Fortran direct access), an obvious inconsistency exists and a message to this effect will be printed and execution of the model will halt.

Many Fortran compilers allow `ifdef` options to be enabled (defined) by using the "-D" option on the compile statement (as shown in the next chapter). This form of using a "-D" option may also be used with the `cpp command` (see file `cray.run`) to preprocess the code before sending separately it to the compiler.

(3.2) GRID OPTIONS

cyclic is an option for setting cyclic boundary conditons in a zonal direction. If enabled, anything exiting the eastern side of the MOM model grid comes back in through the western side. If not enabled, then solid walls are assumed on the eastern and western boundaries of the model.

symmetry is an option for setting a symmetric condition across the northern boundary of the model. The condition is : Tracers at row jmt on the "t" grid = tracers at row jmt-1 on the "t" grid. zonal velocity at row jmt-1 on the "u" grid = zonal velocity at row jmt-2 on the "u" grid. meridional velocity at row jmt-1 on the "u" grid is zero. meridional velocity at row jmt on the "u" grid is the negative of the meridional velocity at row jmt-2 on the "u" grid. If not enabled, solid walls exist at the northern and southern boundaries.

(3.3) EXTERNAL MODE OPTIONS

Basically, there are four Poisson solvers provided in the GFDL MOM code. All of them require the **rigidlid** option to also be enabled.

oldrelax duplicates the meothod used in Mike Cox's *(1984)* ocean model base code. If you use it, you should twiddle with variable `sor` (the overrelaxation constant) to minimize the number of scans for your geometry. Variable `crit` (which along with `sor` can be specified through the namelist) controls the accuracy. See file `relax.F` for more details.

congrad5pt is an option which specifies the use of a conjugate gradient technique. Variable `sor` is not needed here, but the convergence criterion `crit`, which when satisfied halts the iterative scheme, is used. This scheme is generally faster (in terms of cpu time) than **oldrelax**, but may be a bit less stable in the presence of steep topographic gradients. With **congrad5pt** defined, the surface pressure is calculated only as a basis for comparison with the **congrad9pt** option. See file `congr5.F` for more details.

congrad9pt is another conjugate gradient solver that uses a 9 point laplacian instead of the 5 point laplacians used by the other schemes. Its advantage is that it is the most accurate and thus allows the surface pressure to be calculated directly. Reducing `crit` when using **congrad9pt** will give better results (as shown by the closed line surface pressure integrals in the test case) than **congrad5pt** (this is because there is significant truncation due to the 5 point numerics). In fact, the accuracy of the **congrad9pt** scheme's streamfunction solution is limited only by the variable `crit` and computer precision.The 9 point scheme's down sides are that it may have stability problems for topographies with sharp gradients and it will take more cpu time. How much you ask? It depends greatly on your configuration. Try it and see. See file `congr9.F` for more details.

hypergrid is a variation of **oldrelax** solved using a checkerboard technique: first on the black squares, then on the red ones: Numerically it is very similar to **oldrelax** (`sor` and `crit` comments apply), however it does run somewhat faster than **oldrelax** on some types of computer architectures.

REFERENCES:

Cox, M. D., "A Primitive Equation, 3-Dimensional Model of the Ocean", *GFDL Ocean Group Technical Report #1*, (1984).

(3.4) LATERAL MIXING SCHEMES

Lateral mixing applies to both momentum and tracers. One (and only one) scheme must be enabled.

consthmix enables a linear second order mixing scheme with constant eddy coefficients: variable `am` is the eddy coefficient for momentum while variable `ah` is for heat and the other tracers. One usually attempts to choose these values to be just large enough to suppress small scale numerical noise. A typical value for variable `am` in a one degree resolution configuration of the MOM code might be 1.0×10^7 cm^2 sec^{-1}. Usually variable `ah` is chosen to be smaller so as not to diffuse away frontal features.

biharmonic selects a linear fourth order mixing with constant coefficients as in **consthmix**: `am` for momentum and `ah` for heat, salinity, and other tracers if present. The **biharmonic** scheme is more scale selective than the **consthmix** option (ie: more severly damps the small scale features). For a 1/3 degree resolution version of the MOM code, variable `am` should be roughly 10^{19} to 10^{20} cm^4 sec^{-1} (the minus sign is in the equations and thus does not appear in the eddy coefficients).

nlhmix is the non-linear horizontal subgrid-scale mixingoption, based upon Smagorinsky's non-linear viscocity scheme as implemented and described in Rosati and Miyakoda *(1988)*. In this formulation, the horizontal eddy diffusion coefficient is proportional to the horizontal grid length and to the local deformation field. The eddy coefficients computed by the scheme are sensitive to the spatial scales of motion. Accordingly, mixing coefficients are relatively small in the open ocean, where the scales of motion are comparatively large, and are relatively large in regions where the scales of motion are comparatively small.

REFERENCES:

Rosati, A., and K. Miyakoda, *J. Phys. Oceanogr., 18,* ??-??, (1988)..

(3.5) VERTICAL MIXING SCHEMES

Vertical mixing applies to both momentum and tracers. One (and only one) scheme must be enabled. The default is for explicit solution. Choosing the **implicitvmix** option will cause the vertical mixing equations to be solved implicitly. NOTE: in the GFDL MOM code, explicitfor the solution unstable density profiles lead to tracers being mixed via convective adjustment but NOT momenum. But both momentum and the tracers are vertically mixed when the **implicitvmix** option has been defined by the user.

implicitvmix solves the vertical diffusion term implicitly for all vertical mixing schemes. This allows for large mixing coefficients without the need to reduce the timestep. With this option enabled, convective adjustment is not done in file `tracer.F` and the unstable density profile is mixed using large mixing coefficient limits (see variables `vvclim` and `vdclim`). For the **constvmix** and **ppvmix** options, these limits are set in the `blkdta.F` file and for option **tcvmix** the coefficients are computed. Selecting the **implicitvmix** option also requires that varaible `aidif` (the implicit factor) be set (see files `cvmix.h`, and `blkdta.F` and the namelist inputs).

`0.5 < aidif < 1` ; always stable

`0 < aidif < 0.5` ; stable if ...

$$\kappa \times \frac{dt}{(dz)^2} < \frac{1}{(2 - 4 \times aidif)}$$

where κ is the maximum vertical mixing coefficient.

constvmix utilizes constant eddy coefficients (variables `fkpm` and `fkph` for momentum and heat, respectively) for linear second order mixing of momentum and tracers in the vertical. In the explicit case (when **implicitvmix** has not been defined), convective adjustment handles regions of gravitational instability for the tracers. If **implicitvmix** is defined, then the convective section is bypassed and the large mixing coefficient values (limits set in variables `vdclim` and `vvclim`) are used to set the vertical mixing coefficients to handle the instability for both the tracers and momentum.

ppvmix is a vertical mixing option based on the Pacanowski and Philander *(1981)* Richardson mixing scheme. This parameterization was designed primarily for equatorial models with vertical resolution of about 10 meters between levels in the upper ocean (we were most interested in the structure of the undercurrent). Previous versions of this code assumed the “t,s” grid was coincident with the “u,v” grid and gave good results. In the present case, we have relaxed this assumption. We tried this newer, more consistent version a few ways and got lots of numerical noise, at first (particularly on the equator off Brazil). The present configuration minimizes noise. If noise develops off steep shelf breaks it can sometimes be suppressed by turning on a bottom drag (see variable `cdbot`). The background mixing variable `bvdc` should be kept much less than 1.0 to

avoid diffusing the thermocline away on long time scales.

In regions of gravitational instability, **ppvmix** uses the vertical mixing limit variables `vdclim` and `vvclim` to set the vertical eddy coefficients. In the explicit case they must satisfy the "CFL" criterion and may be too small to remove the instability. However, the convective adjustment is operative and tries to remove the instability. Whether it does or not depends on the `ncon` variable (setable through namelist), which controlls the number of passes through the convective adjustment section of code in file `tracer.F`. If **implicitvmix** is used, `vdclim` and `vvclim` are set large and the convective adjustment section is bypassed. The **ppvmix** scheme also assumes the use of equal timesteps on the density and momentum equations. If a longer time step is desired, try using the **implicitvmix** option.

tcvmix invokes the Mellor-Yamada level 2.5 turbulence closure scheme. Here the mixing coefficients are a function of the turbulence length scale (l), the turbulent kinetic energy (q^2), the analytically derived stability factors (Sm & Sh), and the boundary conditions. There are two options within **tcvmix** to compute the length scale:

leq solves an additional equation for q^2 from which the length scale may be derived at each level.

lalg, obtains the length scale from an algebraic relationship. The **lalg** option may be sufficient for the boundary layer but not in cases where there would be multiple turbulent regimes. The horizontal diffusion coefficient of turbulent kinetic energy is set using the variable `aq` (see files ctcmix.h, blkdta.F and the namelist input). For this scheme to be effective the **implicitvmix** option should be enabled and also the vertical resolution should be sufficient.

REFERENCES:

Pacanowski, R. and S.G. Philander, *J. Phys. Oceaonogr., 11,* ??-??, (1981)
Mellor, G. and Yamata ?????

(3.6) HYBRID MIXING SCHEME OPTIONS

We are using the term “hybrid” here to describe schemes which apply to either momentum or tracers but not both. For all cases in which **implicitvmix** is not enabled, convective adjustment is the default. Recall that when solved explicitly convective adjustment mixes tracers vertically in regions of gravitational instability, but not momentum. The effectiveness of convective adjustment is controlled by variable `ncon` which is the number of passes through the convective section (use **grep -i -n ncon *.[Ffh]** to see its usage). If **impllicitvmix** is enabled, the convective adjustment section is bypassed. in file `tracer.F`. The assumption is that the vertical mixing schemes will handle it.

isopycmix is a scheme in which a mixing tensor is computed from the local isopycnal slope and the diffusion of tracers is then conducted along that direction. It therefore influences both the lateral and vertical mixing of tracers in the model. Its use is intended to partially mitigate a perceived shortcoming of z-coordinate primitive equation ocean models in parameterizing oceanic mixing due to mesoscale motions. Since such mixing is believed to take place along isopycnal surfaces, the **isopycmix** option seeks to orient the mixing of tracers along isopycnal surfaces rather than purely horizontal surfaces, as described by Redi *(1982)*.

As described by Cox *(1987)*, in addition to specifying the mixing coefficient for along isopycnal diffusion (variable `ahisop`), a non-zero background horizontal mixing coefficient (here we use variable `ah`) is often needed to supress gridpoint noise.

Additionally, a constraint is placed on the steepest isopycnal slope (see varable `slmxr`) that the scheme will consider when computing the components of the mixing tensor. A typical value of `slmxr` is 100.0, which translates into a slope of 1:100. Steeper slopes would then be considered to be 1:100 for the purpose of computing the mixing tensor.

A vertical mixing scheme must be specified for use with **isopycmix**. The zz component of the isopycnal mixing is added to the vertical diffusion coefficient produced by the vertical mixing scheme. Since **isopycmix** only affects tracers, one must also specify a lateral mixing scheme from the above list to be used for momentum. This additional lateral mixing scheme has no effect on tracer diffusion.

REFERENCES:

Cox, M., *Ocean Modelling, issue 74,* 1-5, (1987)

Redi, M., *J. Phys. Oceanogr., 12,* 1154-1158, (1982).

(3.7) OPTIMIZATION OPTIONS

skipland is an option that allows calculations in the GFDL Modular Ocean Model to be done over blocks of ocean while skipping land points. The trade off here is the time saved by not doing the calculation over land versus the time used in starting and stopping a lot of vectors. Whether or not cp time will be saved by using this option is largely a function of land mass geometry and grid resolution. A global ocean would be a good candidate for this option, but a idealized basin would not.

timing is an option that cp and wall clock times for running on a Cray YMP. Additionally, the time per grid point per time step is calculated.

multitasking is an option that was developed with the Cray YMP in mind. It allows the slabs to be divided into a series of tasks. Each task contains approximately the same number of slabs and there should ideally be one task per processor. This is all handled by simply specifying variable ntasks to be the number of processors (we've defaulted it to 8 in file `blkdta.F`).

The idea behind multitasking is that since all tasks are independent, all tasks can be worked on simultaneously. This reduces the total wall clock time for the job but NOT the total cpu time. Note: the `ntasks` variable may be set > 1 even when unitasking (running on one processor), but this is not recommended, since there is extra work being done at the interfaces between tasks. Also, on diagnostic, tracer averaging, and data saving time steps, the MOM code will revert to one task of jmt-2 slabs, so effectively multitasking is shut down for these time steps.

The **multitasking** option currently does not work with the **diskless** option (see below) because only two time levels are used for slabs on simulated disk (`ntlev`=2) to save memory and this is incompatible with `ntasks` > 1. This condition may, in principle, be removed by setting parameter `ntlev` = 3 when enabling the **diskless** option. We haven't explored it yet.

To date, we have confirmed that all combinations of unitasked, multitasked, and autotasked runs give the same answers down to the last hex digit on the Cray YMP. To really utilize this feature on the Cray YMP, the SSD (solid state disk) must be used for the slabs (unless large memory and the **diskless** option is used). We have not yet added the best i/o scheme for the SSD, so the wall clock times are higher than they could be.

(3.8) FILTERING OPTIONS

Filtering is used to combat the effect of the convergence of meridians (shrinking longitudinal grid spacing) near the poles. The problem is that the grid spacing severly limits the time step (especially when using large density time steps) due to the "CFL" criterion. Filtering relaxes this constraint by truncating high wave numbers. Users should be aware that the filtering options can be time consuming, and alter the model solution, and therefore should be used only as a last resort

Configurations of the GFDL Modular Ocean Model code that are prime candidates for filtering are coarse resolution global models used to examine long term climate. The timestep constraints imposed by including the Arctic Ocean, and the timescales of the problem being studied, often lead to the need for filtering at high latitudes.

fourfil is a fourier filter which acts on longitudinal strips of ocean points. The number of wave numbers to be allowed at each latitude is decided upon and wavenumbers above are truncated (without using any reasonable window). This option is provided as a backward compatability feature for the Cox (1984) code. Note: Both in the MOM code and in the Cox code, the amount of filtering depends on the number of ocean points within the longitudinal strip (determined by lmodel bathymetry). This actually leads to possibly different truncations at the same latitude!

firfil is a simple finite impluse response filter based on the familiar 1/4, 1/2, 1/4 weights (the response function is a cosine). It comes in two flavors: One to with symmetric boundary conditions (ie: for tracers) and one for asymmetric boundary conditons (ie: momentum).

The **firfil** option also works on longitudinal strips of ocean points, and may be applied an arbitrary number of times for an arbitrary number of latitudes. While the **firfil** option is far less cp time intensive alternative to **fourfil**, we really have not done any long running comparisons to comapre the two schemes

(3.9) I/O OPTIONS

diskless simulates disk usage using an array in memory. Since when using this option no "wall clock" time lost during disk transfers (because they are memory to memory transfers), this method gives the best efficiency (cp time / total time). The cost for achieving this cpu time efficiency is that the model may not be able to fit into available memory.

As a memory saving feature, **diskless** keeps only two time levels of prognostic variables (`ntlev`=2) instead of the usual three. See the `size.F` file to examine resource requirements. Users may wish to read the discussion of the **multitasking** option for related information.

crayio is an option tested on GFDL's Cray YMP. Currently, it uses the "`getwa/putwa`" routines. If the disk units are set up on "non SSD" type disk, then the efficiency (cp time / total time) will be bad. This shows the mismatch in speed between computationand disk use. SSD (solid state disk), however, is really the preferred way to go. It can be looked at as an extended memory and the efficiency is much better.

fio is the option for using standard Fortran direct access i/o. The comments from **crayio** apply here also.

(3.10) MISCELLANEOUS CODE OPTIONS

:keepterms is an option allows the use of arrays instead of statement functions for component terms in the MOM code's tracer and momentum equations. This may be desirable when using component terms over and over for some purpose. The trade off is the extra memory needed for the arrays versus the extra time needed to recalculate the statement functions. See the `size.F` file to look at resource requirements.

nohilats stands for no high latitudes. If the latitudinal domain of the model is limited to equatorial redions, for instance, then the metric nonlinear terms in the momentum equations may be dropped. Defining the **nohilats** option will save some cp time and allows for backward compatibility with the Cox (1984) code.

restorst stands for restore surface tracers to some prescribed values by a newtonian damping term. In the sample test case of the GFDL Modular Ocean Model code, temperatures and salinities are restored back toward zonally averaged annual mean conditions on a time scale of 50 days (ee file `setvbc.F`). Without much work, this option could easily be extended to damp certain regions while leaving others alone by simple making the damping time scale a function of latitude, for instance.

testcfl tests whether any velocity exceeds the local "CFL" criterion by a factor of `cflcrt`(see file `blkdta.F` for where variable `cflcrt` is set). This option will print out the places where the CFL criterion is most nearly met on diagnostic time steps. If the `cflcrt` factor is exceeded on any time step, the MOM code will stop its execution and print out the surrounding variables.

NOTE: the **testcfl** option is not meant to be left on all the time, since is uses much cp time. It is best used as an analysis aid. For example, it can be used infrequently on a model that is being run for a long time, or should a version of the model "blow up", it can be useful in pointing to where the problem arises.

This page was intentionally left blank

CHAPTER 4: Some Basics of the GFDL Modular Ocean Model

Any general circulation model contains an intimidating number of “things” to keep track of. For a new user, deciding which things to study in detail first so that other portions of the model can be understood can be important question. This chapter seeks to describe for the user several of the basic building blocks that one needs to know in order to develop working knowldege of the GFDL MOM code. We expect that more seasoned users will refer to these pages when seeking to refresh their memories.

The discussion of building blocks begins with the model grid itself. However, while describing the grid, we will also introduce some specifics concerning conventions used in the naming of variables.

(4.1)THE THREE DIMENSIONAL GRID

The structure of the model grid itself is a reasonable topic to tackle first when building up knowledge of the GFDL Modular Ocean Model code. Understanding the layout of the grid and some of the MOM code's naming conventions that relate to grid locations is crucial if one it to be able to mentally translate many parts of the code into visual images.

Figure 4.1.1 displays the horizontal grid structure used in the MOM code and Figure 4.1.2 displays the vertical grid.. Variables are arranged in the manner of the "B-grid" convention described by Arakawa and Lamb *(1977)*. Users may refer to the `coord.h` and `grdvar.h` files for lists and descriptions of many of the variables relating to the grid.

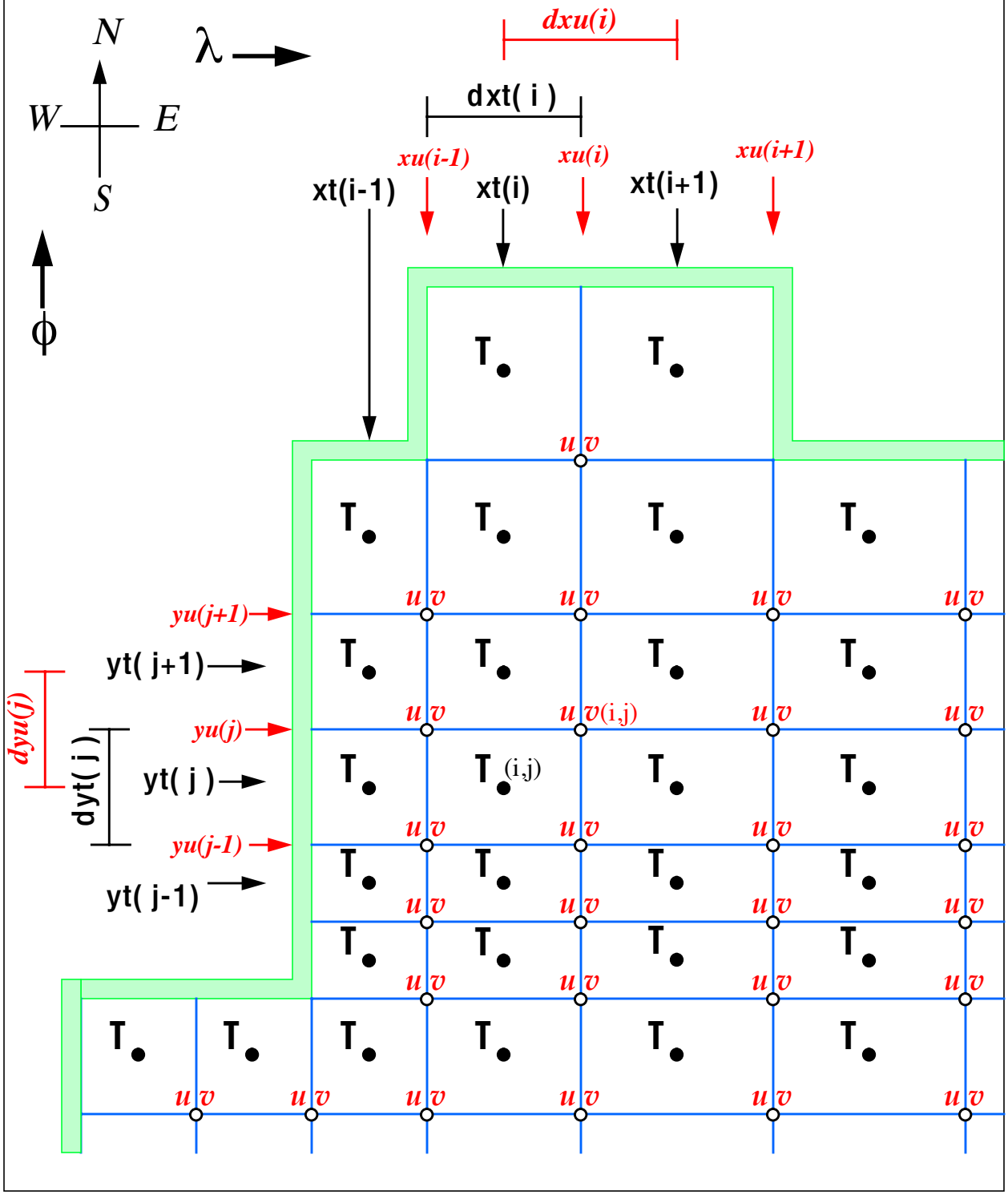
Parameters used to set the number of model grid points in the west-to-east, south-to-north, and top-to-bottom directions are `imt`, `jmt`, and km, respectively (see the `param.h` file). Thus, variables aligned along a specific latitude can be referred to as being in the same "j-row", with values of i increasing as one moves eastward. The first j-row (`j`=1) lies at the southern limit of the model domain, and values of `j` increase as one moves northward. In the vertical, values of `k` increase as one moves downward to greater depths.

Since the GFDL MOM code employs the B-grid, tracer quantities (potential temperature, salinity, and other tracers) are defined in the horizontal at different locations than the velocity component variables `u` and `v`. As shown in Figure 4.1.1, a "`u,v`" grid point with a given `i,j` designation is located to the northeast of the tracer grid location having the same `i,j` values. The two-dimensional volume transport streamfunction Ψ (variable `p` in the `emode.h` file) is also aligned on the horizontal tracer grid.

In the GFDL Modular Ocean Model, horizontal grid locations are carried in terms of degrees of latitude and longitude. Variable `xt(i)` represents the longitude of the "i-th" point on the tracer grid, and variable `xu(i)` represents the longitude of the "i-th" point on the `u,v` grid. Similarly, variable `yt(j)` represents the latitude of the "j-th" model row on the tracer grid, while the latitude of the "j-th"`u,v` j-row is stored in variable `yu(j)`.

In the horizontal, tracer grid points lie in the exact center of tracer grid box cells. The four points of a tracer grid cell lie at `u,v` grid point locations. The distance (in centimeters) across a tracer grid box cell in the north-south (meridional) direction at location `i,j` is stored in variable `dyt(j)`, while the distance in the east-west (zonal) direction at the Equator is stored in variable `dxt(i)`(see the `grdvar.h` file). As depicted in Figure 4.1.1, the GFDL Modular Ocean Model allows for variable grid spacing in the horizontal. Therefore, `u,v` grid points do not necessarily lie in the center of the four surrounding tracer gridpoints. Distances across `u,v` grid cells are given by variables `dyu` and `dxu`, and correspond to the distances (in centimeters) between tracer grid points in the north-south and east-west directions, respectively.

FIGURE 4.1.1 Skematic view of the GFDL Modular Ocean Model's horizontal grid arrangement.



Since the dxu and dxu variables are calculated as the distance (in centimeters) across a grid cell in the zonal direction at the Equator, one must multiply these variables by the cosine of the latitude of the j-row of interest in order to determine the true east-west distance between grid points at a given latitude. The cosines of the latitudes of tracer grid points (yt(j)) are stored in variable cst(j), and the cosines of the u,v grid point latitudes (yv(j)) of tracer grid points are stored in variable csu(j) (see file grdvar.h).

The following sentence attempts to tie together some of the elements presented thus far concerning the horizontal grid used in the GFDL MOM code:

The east-west distance, in centimeters, across the southern face of the tracer grid box cell centered at latitude yt(j) and longitude xt(i) is equal to the distance from longitude xu(i-1) to xu(i) at latitude yu(j), which can be computed as... csu(j-1) * dxu(i)

With the above preliminaries out of the way we can now turn to the question of how a user specifies the horizontal grid desired for a particular application in the MOM code. The process begins when program ocean calls subroutine grids. Subroutine grids, and supporting subroutines mesh and cmesh and function indp are found in file setgrid.F.

Upon entering subroutine grids, a call to subroutine mesh is made in order to determine tracer grid cell widths dxt and dyt. All other grid variables are calculated within the subroutine grids from these two variables.

As alluded to earlier, the GFDL MOM code provides a means by which variable grid spacing can be incorporated into the model. This is achieved in subroutine mesh. The nxres and nyres parameters are the number of subdomains of either constant or smoothly varying grid cell dimensions. For the simplest case of a grid that is uniform in terms of latitude and longitude grid spacing, parameters nxres and nyres are set to be equal to one in file param.h. This was done for the sample configuration of the MOM code.

Variables used to determine the grid spacing (be the spacing either uniform or variable) are set by the user in the blkdta.F file. These variables (namely, variables stlon, xmax, xmin, xwid and idir for longitudinal subdomains, and stlat, ymax, ymin, ywid and jdir for the latitudinal subdomains) are described in the comments found in the coord.h file.

For each of the longitudinal and latitudinal subdomains asked for (nxres and nyres) the user must specify the sum of the distances (in degrees) to be covered by tracer grid box cells within that subdomain. Subdomain widths are given by xwid(nx) and ywid(ny), where ny goes from 1 to nxres and ny from 1 to nyres. Variables xmax and xmin and ymax and ymin are used to specify the minimum and maximum tracer grid box cell widths within the given domain. Uniform spacing within a subdomain can be achieved by setting the minimum and maximum widths to be equal. Otherwise, smoothly varying grid cell widths will be computeda, and the idir and jdir variables are set to either +1 or -1 to signify whether the grid box cells are to

increase or decrease as one moves across the subdomain to higher i or j values. The calculations for the variable tracer grid cell widths are performed in `subroutine cmesh`.

The model grid point located in the far southwest of the entire model domain is a tracer grid point with an `i,j` location of (1,1). All model grid locations are computed using this position as a starting point.. This starting location for the model grid is given (in degrees) by variables `stlat` and `stlon`.

Variables `xt`, `yt`, `xt` and `xu` are each monotonically increasing in the MOM code. In cases of an ocean model with pseudo-realistic geometry, the **`cyclic`** option needs to be specified, and thus the sum of all the longitudinal subdomains must be greater than 360 degrees (as in the sample case).

Should a user have a special application in which it is desired to have latiitudal and/or longitudinal grid spacing that varies in a way not readily produced by the above described method, we suggest the following get-around. Add a new ifdef option to the code, such that when it is defined subroutine grids calls a new, user supplied routine, rather than subroutine mesh.. This new routine should return variables dxt and dyt. Once that is accomplished, all other grid variables will be computed, starting at the `stlat` and `stlon` tracer gridpoint location.

REFERENCES

Arakawa, A., and V. Lamb, “Computational Design of the basic dynamical processes of the UCLA general circulation model”, *Methods in Computational Physics, vol 17*, Academic Press, (1977)

INDEX

S

T

U

V

W