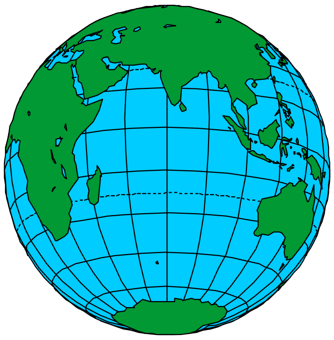
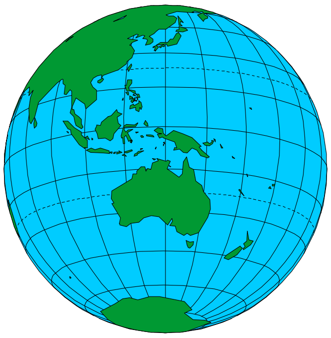
MOM 2

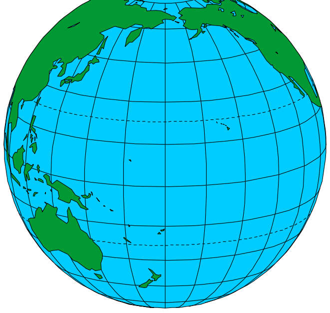
Version 2.0 (*Beta*)



Documentation
User's Guide
and
Reference Manual



Edited by Ronald C. Pacanowski



Release date: Nov 7,1996

GFDL Ocean Technical Report 3.2

0.1 Introduction

A bit of history. The GFDL Modular Ocean Model (acronym MOM 2) is a three dimensional primitive equation ocean model based on the pioneering work of Kirk Bryan (1969). Early Fortran implementations of Bryan's ideas were carried out chiefly by Mike Cox in Washington, D.C. during the late 1960's on a IBM 70301/stretch and then a CDC 6600 computer. Cox continued those developments by constructing a global model at GFDL on a UNIVAC 1108. Bert Semtner[1] converted that model to execute on Princeton University's IBM 360/91 in 1970 and both codes were in use through 1973 with Semtner's version surviving for use on an interim IBM 360/195 in 1974. While at GFDL and UCLA, Semtner (1974) rewrote the model to take advantage of the instruction stack on the IBM 360/195 and also with future vector architectures in mind. The first vector machine to arrive at GFDL in 1975 was a four pipeline Texas Instrument ASC (acronym for Advanced Scientific Computer) and the model of Semtner (1974) was used as the starting point for further conversion efforts by Cox and Pacanowski. After the ASC, Cox abandoned the ASC version of the model in favor of Semtner's latest version and optimized it for the CDC Cyber 205 which required very long vector lengths for efficiency. Actually, the Cyber 205 experience involved two Cyber 205's and a Cyber 170 front end delivered to GFDL in stages between 1982 and 1983. It was the resulting "Cyberized" version of the model that was distributed as the Cox (1984) ocean model code. Over the lifetime of the Cyber system, Cox installed other features such as variable horizontal resolution, multiple tracers, and isopycnal mixing until his untimely death in 1989.

In anticipation of a Cray YMP with 8 processors, 32 Mwords of central memory (eventually upgraded to 64 Mwords), and 256 Mwords of Solid State Disk at GFDL in 1990, the model was rewritten again. This time by Pacanowski, Dixon, and Rosati (1991) using ideas of modular programming to allow for more options and increased model flexibility. This development work, which became known as MOM 1 (Modular Ocean Model), could not have happened without reliance on workstations and the acceptance of UNIX[2]. With the realization of the importance of workstations for productivity within GFDL, SUN workstations were replaced by suite of SGI 4D/25, INDIGO, and INDIGO2's totaling 115 within the early 1990's. With the aid of these faster workstations, further design work was carried out primarily by Pacanowski and Rosati but with numerous contributions from others both inside and outside of GFDL. This led to the incarnation known as MOM 2 Version 1 (1995).

Early in 1996, a Cray C90 was installed at GFDL with 16 processors, 256 Mwords of central memory, 1 Gword of solid state disk, and 370 Gbytes of rotating disk. Later that year, the system was replaced by a Cray T90 having 20 processors, 512 Mword central memory and a 2 Gword solid state disk. The Cray T90 is scheduled for upgrades to 24 and then 30 processors by 1998. Additionally, a Cray T3E with 40 processors and 640 Mwords of memory is due in 1997. To take advantage of this environment and in anticipation that in the near future parallelization will be needed to keep overall system efficiency high, some attention has been given to multitasking. These and other changes described below are included in the latest version known as MOM 2 version 2. Throughout these developments, the intent has been to construct a flexible research tool useful for ocean and coupled air-sea modeling applications over a wide range of space and time scales.

Documentation. There has been no serious attempt at ocean model documentation at GFDL since the technical report of Cox (1984). Since the release of MOM 1 (1990) , there have been

[1] He was stationed at GFDL in the early 1970's as a commissioned officer in the NOAA CORPS.

[2] In the later half of the 1980's, SUN 3/50 workstations were introduced which ushered in a new era of model development. Before this, code development was done without the aid of editors or utilities like UNIX *grep*.

many requests for updated documentation. This manual was written to satisfy that need by Ron Pacanowski and can be referenced as "MOM 2 Version 2, Documentation, User's Guide and Reference Manual", GFDL Ocean Technical Report #3.2, Geophysical Fluid Dynamics Laboratory/NOAA, Princeton, N.J. 08542, edited by Ronald C. Pacanowski 1996. Other contributers to this documentation have been acknowledged within and their work is gratefully appreciated.

Although the design, components, and options of MOM 2 have been documented, in one sense this documentation is incomplete because of the very nature of the actively developing research tool which it describes. Nevertheless, the salient features are documented and as new features are added, there is a strong commitment at GFDL to keep this manual up to date. If certain sections appear "thin", this should be viewed as an opportunity to point out the need for clarification. Contributions are welcomed and contributers will be acknowledged as authors of their sections.

Model development. The focus of MOM 2 development work at GFDL is to maximize scientific productivity within the computational environment at GFDL. However, the model is sufficiently general to be of use elsewhere. Therefore, the MOM 2 code and this documentation are being made available (free) to the oceanographic and climate community. For system requirements, refer to Section 19.1.

As ideas are developed and progress is made by various groups within the modeling community, it is hoped that the modular structure of MOM 2 will be capable of incorporating their developments for use by the entire community. The vision is that if this can be done, everyone will benefit. In support of this effort, collaborations with those who share this vision are welcomed[3]. The hope is that in time, MOM 2 will become a repository for parameterizations and the documentation will grow into a storehouse of facts, examples, explanations, and modeling information useful to all researchers.

In addition to contributions to this manual, many people have contributed Fortran code to this effort. For developers of future parameterizations, old options can be used as prototypes for adding new ones. This is particularly made straightforward since UNIX preprocessor "ifdefs" surround all options which makes easy work of identifying where to put new code. Within the model, each section of code is identified with an author and e-mail address. The intent is to identify code authors for the purpose of answering questions and resolving problems if a "bug" is found. Questions or suspected problems should be directed accordingly. New parameterizations to be submitted should be thoroughly tested, well documented, and cleanly coded with adherence to the style in MOM 2. A clearly written documentation explaining the details along with appropriate guidance for usage is also required. It is highly recommended that this documentation be in a LaTeX[4] form.

Although the design of MOM 2 is intended to allow a wide range of parameterizations to co-exist as options, it should be noted that not all ideas having scientific merit are suitable candidates to be included. The deciding factor is whether or not the implementation would compromise the design of MOM 2 or in some way add impediments to future model development. Constructs that compromise the design can sometimes be recast into a more appropriate form. Those that bypass rather than incorporate the underlying data structure in MOM 2 and substitute their own are best thought of as a separate effort.

The current version of MOM 2 with its three dimensional array structure allows for lots of generality and is much closer to what was envisioned as the goal by developers of MOM.

[3] For this effort to succeed, collaborators are strongly encouraged to adhere to the underlying design principles and coding style which is used consistently throughout the model.

[4] An easy to use document preparation system by Leslie Lamport published by Addison-Wesley 1986 which is widely available and acts as a front end for TEX by Donald Knuth.

While MOM 2 was being developed, adding new options sometimes lead to problems which were alleviated by tweaking the underlying design. Consideration was given to how details of a parameterization affected overall coding structure and what the resulting changes in structure implied for details of parameterizations. Iterating over this bottom up / top down approach led to a design which required successively fewer modifications. The fundamental design now appears robust enough that future design changes are likely to be relatively small perturbations. However, it would be naive to expect that with time, changes and improvements would not be sprinkled throughout. On the other hand, new parameterizations are likely to involve substantial amounts of code, but if existing options are used as guidelines, this code should limited to a few contiguous code sections. In any case, the strategy for upgrading outlined in Section 19.15 will make elevating local modifications to future releases of the model relatively easy. This method of upgrading is straightforward and has been in use for some time by researchers within GFDL who like to keep abreast of the latest development version.

Differences between MOM 1 and MOM 2. There are major architectural differences between MOM 1 and MOM 2. As a result, there is no simple utility which will provide a meaningful upgrade path from MOM 1 to MOM 2. Migrating from MOM 1 to MOM 2 is a matter of biting the bullet and should only be attempted at the beginning of an experiment. One of the first differences to notice is a change in naming variables. To remove lack of uniformity and to provide guidance in choosing variable names for future parameterizations, a naming convention has been adopted as described in Section 5.1. Not only variable names but details of subscripts and numerics within this documentation consistently match what is found in the model code. Therefore, understanding this documentation will allow the researcher to take a big step towards gaining a working knowledge of MOM 2.

Apart from renaming of variables, the next thing to notice is that a latitude “j” index has been added to expose all indices of arrays in MOM 2. Although the organization of the code bears similarity to MOM 1, this added “j” index results in fewer variable names being required and triply nested “do loops” replacing the doubly nested loop structure in MOM 1. It also allows the slab architecture of MOM 1 to be extended to a more general memory window structure which permits solving equations on one or more latitude rows at a time. This has implications for parallelization and simplifies incorporating parameterizations (such as fourth order accurate schemes, flux corrected transport schemes, etc.) which require referencing data from more than one grid point away. For such parameterizations, the memory window is simply opened up to contain four latitude rows as opposed to the usual three. In the limit when enough central memory is available, the memory window can be opened all the way to contain all latitude rows, in which case all data is entirely within central memory, there is not disk, and therefore no movement of data between central memory and disk. Also, in contrast to a partially opened memory window, there are no redundant computations necessary. The main point is that all arrays and equations look the same regardless of the size of the memory window and whether one, a few, or all latitude rows are being solved at once. The details are given in Chapter 1.

The memory window also allows flexibility in parallelization by multitasking as described in Section 3.5. When executing on multiple processors, MOM 2 can make use of fine grained parallelism (“autotasking”) or the coarse grained parallelism (“microtasking”). Each method has its advantages and disadvantages. Fine grained parallelism makes efficient use of available memory and offers a robust coding environment which is easy to use thereby keeping the researchers efforts focused on science as opposed to debugging. It suffers from relatively low parallel efficiency[5] which limits its use to multitasking with a small number of processors. However, the highest parallel efficiency may not be important when multitasking on systems

[5] The efficiency is limited by how smart the parallelizing compiler is.

with tens of processors and when the number of jobs in the system exceeds the number of processors. Higher parallel efficiency, which is necessary when executing in a dedicated system, can be achieved through coarse grained parallelism. The down side is that this approach uses significantly more memory than fine grained parallelism and is more prone to introducing errors. The researcher who is intent on developing new parameterizations may find the tendency for the focus to shift from science to debugging.

Other features which are new to MOM 2 include the idea of a module which can be exercised alone or as part of a fully configured model and this is discussed in Chapter 6. An integrated DATABASE is described in Chapter 4 along with run_scripts in Section 19.6 which will automatically prepare this data for any of the configurations and arbitrary resolutions of MOM 2. Chapter 10 details a generalized surface boundary condition interface which handles all surface boundary conditions as if they come from a hierarchy of atmospheric models. This includes simple datasets which are fixed in time through complicated atmospheric GCM's. Mismatches in geometry and resolution between atmospheric GCM's and MOM 2 are automatically taken care of. Elliptic equation solvers for the external mode have been re-worked to be more accurate and give speedier convergence as discussed in Section 11.13. The vertical velocity fields have been reformulated to prevent numerical separation in the presence of sharp topographic gradients as described in Section 11.5. The grid is constructed by a module which allows for a MOM 1 type construction with grid points always in the center of tracer cells on non-uniform grids <u>or</u> a new way with grid points always in the center of velocity cells on a non-uniform grids. Both are second order accurate if the stretching function is analytic and are described in Chapter 7. All diagnostics have been re-written to be more modular, old ones have been improved, many new ones added (such as reconstructing the surface pressure from the stream function, calculating particle trajectories, time averaged fields, xbt's etc.), and all are described in Chapter 18. The prognostic surface pressure and implicit free surface methods of Dukowicz and Smith (1993,1994) and the isopycnal thickness diffusion of Gent and McWilliams (1990) are part of MOM 2. There are also more options for configuring MOM 2 as described in Chapter 15 and many other little features and code improvements too numerous to summarize here but covered in this manual.

<u>Newest features</u>. Some of the differences between MOM 2 version 1 and version 2 are as follows: All diagnostics have been given an interface to generate NetCDF formatted output as described in Chapter 18. The NetCDF format allows easy access to results without writing intermediate analysis code. A good way to visualize results is to use Ferret which is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL (email: ferret@pmel.noaa.gov URL: http://www.pmel.noaa.gov/ferret/home.html). An option for coarse grained parallelism (microtasking) has been added which makes more efficient use of multiple processors than the fine grained (autotasking) approach as discussed in Chapter 3. This has relevance on platforms such as the CRAY T90 when using tens of processors. An improved isopycnal mixing formulation based on a functional approach employing a new approximation to neutral directions has been added (paper in progress by Griffies, Gnanadesikan, Pacanowski, and Larichev). Consistent with the new isopycnal mixing formulation, improvements have also been made to the numerics in the Gent McWilliams parameterization. A fourth order advection scheme for tracers, the FCT scheme of Gerdes, Köberle and Willebrand (1991), a third order advection scheme for tracers (by Holland) very similar to the Quick scheme of Leonard (1979), the pressure gradient averaging technique of Brown and Campana (1978), the Neptune effect of Holloway (1992), a general grid rotation (by Eby), and open boundaries (by Biastoch) of Stevens (1990) have been added. The discretization of vertical mixing of Pacanowski/Philander (1981) has been changed to yield more accurate and stable solutions as indicated in Section 15.14.2. There has also been

some restructuring of the memory window logic to allow for a more robust implementation of parallelism and fourth order schemes.

Since the arrival of a CRAY T90 and Unicos 9 operating system in August 1996, there is no longer a Fortran 77 compiler "cf77". It has been replaced by a Fortran 90 compiler "f90" which for the most part is compatible with "cf77". However, the Fortran 90 compiler necessitated changes in run scripts and namelists have been made Fortran 90 compliant. These and other related changes are summarized in Section 19.2.

Work in progress. The coarse grained parallelism approach used on the CRAY T90 is being extended for distributed architectures with particular focus on the CRAY T3E as discussed in Section 3.5.3. Delivery of the CRAY T3E at GFDL is not expected until early next year and parallelization efforts can not be completed until after the CRAY T3E arrives. Anticipation is that one version of MOM 2 will execute efficiently at GFDL on both the CRAY T90 and CRAY T3E with machine specific code being kept to a minimum. Also, a nonlocal "K profile parameterization" (KPP) of Large et al. (1994) is being added (Rosati) and a version of Killworth's explicit free surface method (Schmidt). An additional feature under serious consideration for inclusion in MOM 2 at GFDL is a "partial step" method which will allow the bottom levels to have variable thickness as in Adcroft et al. (1996).

World Wide Web access. Because of its size, only the introduction and table of contents of this manual are available on the World Wide Web[6] (*htpp://www.gfdl.gov*). The manual in its entirety may be obtained by anonymous ftp from GFDL using:

ftp ftp.gfdl.gov	use *ftp* as your login name and your *e-mail* address as the password
cd pub/GFDL_MOM2	Change to the pub/GFDL_MOM2 directory
get manual2.2.ps.Z	Copy the manual to your directory
quit	disconnect from the ftp
uncompress manual2.2.ps.Z	Expand to manual2.2.ps

File *manual2.2.ps.Z* is about 1MB and the uncompressed postscript version *manual2.2.ps* is a about 3MB. The printed manual is about 330 pages in length and can be printed with the UNIX *lpr manual2.2.ps* command. However, this is not recommended because it can tie up a printer for a long time. The recommended way of printing is to use a postscript previewer such as *ghostview* to mark off about 50 pages at a time and print it in sections. If this cannot be done, then the manual should be printed in off hours to minimize the impact. Graphics and Figures are in color but black and white copies should be fine. If a color version of a figure is desired, mark it using a postscript previewer such as *ghostview* and save to a file. The file can then be sent to a color printer.

Special acknowledgments. To a large part, MOM 2 owes its existence to both Kirk Bryan and Jerry Mahlman (the director of GFDL) for creating an environment in which this work could take place. Their generosity is gratefully appreciated. Also appreciated is the time and efforts of countless of researchers who have tested beta versions, pointed out problems, and continue to suggest improvements along with offering their parameterizations.

Disclaimer: As with any research tool of this magnitude and complexity, bugs are inevitable and some have undoubtedly survived the testing phase. Researchers are encouraged to bring

[6]The design and maintenance of the MOM web page are due to the ongoing efforts of Keith Dixon (kd@gfdl.gov)

them to our attention. Anyone may use MOM 2 freely but the authors assume no responsibility for problems or incorrect usage. It is stressed that the researcher accepts full responsibility for verifying that their particular configuration is working correctly.

U.S. Department of Commerce (DOC) Software License for MOM 2

1. Scope of License Subject to all the terms and conditions of this license, DOC grants USER the royalty-free, nonexclusive, non transferable, and worldwide rights to reproduce, modify, and distribute MOM 2, herein referred to as the Product.

2. Conditions and Limitations of Use Warranties. Neither the U.S. Government, nor any agency or employee thereof, makes any warranties, expressed or implied, with respect to the Product provided under this License, including but not limited to the implied warranties or merchantability and fitness for any particular purpose. Liability. In no event shall the U.S. Government, nor any agency or employee thereof, be liable for any direct, indirect, or consequential damages flowing from the use of the Product provided under this License. Non-Assignment. Neither this License nor any rights granted hereunder are transferable or assignable without the explicit prior written consent of DOC. Names and Logos. USER shall not substitute its name or logo for the name or logo of DOC, or any of its agencies, in identification of the Product. Export of technology. USER shall comply with all U.S. laws and regulations restricting the export of the Product to other countries. Governing Law. This License shall be governed by the laws of United States as interpreted and applied by the Federal courts in the District of Columbia.

3. Term of License This License shall remain in effect as long as USER uses the Product in accordance with Paragraphs 1 and 2.

Contents

List of Figures

Chapter 1

Design Philosophy

1.1 Objective

The GFDL Modular Ocean Model "MOM 2" was designed with one purpose in mind: to maximize scientific productivity within the research environment at GFDL. As indicated in the introduction, the computational environment at GFDL has undergone change with each new computer procurement. To keep pace, efforts have focused on developing one model capable of taking advantage of scalar, vector and multiple processors within this increasingly varied computational environment. At the same time, consideration has been given to organizing the model to allow a large number of options, diagnostics, and physics parameterizations to co-exist in a way that is understandable, extendable, and easily accessible to scientists. In one sense, the design was strongly influenced by having CRAY platforms as the computational workhorses at GFDL since 1990. However, the focus remains on factors that influence overall scientific productivity and the design continues to be motivated by a search for a better way to do science from the trenches of scientific programming. The generality and flexibility within MOM 2 will make it well suited for use by the general oceanographic community[1].

Given the awarding of a computer contract in 1995 for a CRAY C90 at GFDL, which was followed by a CRAY T90 in 1996 and a CRAY T3E expected in early 1997, most scientific work over the next five years will be done on vector machines with a few tens of processors rather than MPPs with thousands of processors. The intent is for MOM 2 to take advantage of this environment without sacrificing scientific productivity to ideas that serve the needs of computational science at the expense of physical science.The following factors are considered to be important in maximizing overall scientific productivity.

1.1.1 Speed

In the past, speed was often thought of as being the equivalent of scientific productivity. In an operational setting where a model is rarely changed, it is justifiable to expend enormous effort to minimize wall clock time. In a research environment, it has become increasingly apparent that other considerations are important. This is particularly noticeable when changes introduced to take advantage of speed make implementation of science thereafter more difficult[2]. What is needed are changes which increase speed[3] but don't reduce clarity. Ultimately speed should be

[1]Although optimized for the environment at GFDL, MOM 2 is intended to execute reasonably well on a variety of computers. However, optimizing for the idiosyncrasies in computer environments outside GFDL is left to the researcher.

[2]As in the Cyber 205 experience.

[3]It is reassuring that the ideas influencing the design of MOM 2 have not significantly altered speed when compared to MOM 1. Early comparisons were carried out using the standard test case resolution of 4° by 3° and

the business of compilers and better algorithms, not physical scientists playing games to beat compilers.

Two philosophies of model building can be summarized by first stating the intent and then asking a question.

1. The aim is to do as much science as possible with this model. Now, how can it be made to execute as fast as possible?

2. The aim is to make this model execute as fast as possible. Now, what science can be done with it?

MOM 2 is the result of focusing on the first.

1.1.2 Flexibility

To be a useful research tool, MOM 2 needs to be easily configurable in many different ways. Also of importance is access to a large number of parameterizations for intercomparisons within the framework of one model. Using preprocessor "ifdefs" gives this flexibility. However, indiscriminant use of preprocessor "ifdefs" can lead to a tangled mess of limited usefullness.

1.1.3 Modularity

MOM 2 is continually being infused with new ideas and the resulting growth can present problems. For example, anyone who repeatedly changes or adds to a large model will appreciate that after time, the model can become unmanageable. At some point, inter-connectivity between sections of code increases to a point where making changes in one place inadvertently breaks something seemingly unrelated. Further limitations become apparent when previously added code acts as a road block to new development. To a large extent, modularity has been used as the key organizational approach to solve this problem and its use is explained in Chapter 6. The other part of the solution involves resisting temptation to make changes in a quick and dirty way for short term gains which inevitably turn into long term hindrances.

1.1.4 Documentation

A good documentation aids in understanding the big picture as well as the little details which are necessary if a model is to be used and extended by many researchers. To this end, details of numerics right down to the subscripts within this documentation consistantly match what is found in the model code. This level of detail plus a straightforward coding style bolsters the scientific accessibility of MOM 2. The manual should be considered a living document which actively reflects the current status of MOM 2 as well as serving as a repository for details inappropriate for published papers and guidelines for usage of parameterizations. Therefore, understanding this documentation will allow researchers to take a big step towards gaining a working knowledge of MOM 2.

1.1.5 Coding efficiency.

Inevitably, the size of a research model increases with time. However, economy of code is always desirable. Voluminous coding to support issues which are not central to science accumulates

15 levels. Changes in the external mode of MOM 1 were necessary to assure the same accuracy as in MOM 2 and there were no diagnostics enabled. MOM 2 ranged from 3% slower to 6% faster (depending on size of the memory window) than MOM 1. Minimum memory configuration in MOM 2 was 1% greater than in MOM 1.

and, if not restrained, starts to account for the bulk of model code. This practice is to be discouraged, although there is fine line to be drawn and the answers are not always unambiguous. The question to be asked is: How much code is this idea worth and can it be justified with respect to the prevailing level of scientific approximations being made? Some areas within MOM 2 have become overly large and complex but with questionable[4] gain. As time permits, simplifications will follow.

1.1.6 Ability to upgrade.

It is vitaly important for researchers to be able to incorporate code changes (which may be of interest personally but not appropriate for general dissemination) into newer versions of a model. It is in this way that researchers are able to take advantage of new parameterizations while retaining local personal changes. Also of importance is the ability to incorporate "bug" fixes. In the past, both of these operations have presented significant difficulties. These difficulties have been largely eliminated by the method described in Sections 19.15 and 19.16.

[4]Cases in point are the I/O manager, and time manager modules.

Chapter 2

Ocean Primitive Equations

2.1 Continuous equations

MOM 2 is a finite difference version of the primitive equations governing ocean circulation. As described by Bryan (1969), the equations consist of the Navier-Stokes equations subject to the Boussinesq[1], hydrostatic, and rigid lid approximations along with a nonlinear equation of state which couples two active tracers, temperature and salinity, to the fluid velocity. Additionally, MOM 2 has an option to relax the rigid lid approximation and solve the free surface equation.

The Boussinesq approximation is justified on the basis of the relatively small variations in density within the ocean. The mean ocean density profile $\rho_o(z)$ typically varies no more than 2% from its depth averaged value $\rho_o = 1.035\ gm/cm^3$ (Gill 1982). The Boussinesq approximation consists of replacing $\rho_o(z)$ by its vertically averaged[2] value ρ_o. In order to account for density variations affecting buoyancy, the Boussinesq approximation retains the full prognostic density $\rho = \rho(\lambda, \phi, z, t)$ when multiplying the constant gravitational acceleration. For scaling consistency, variations in density must also be neglected in viscous and diffusive terms (Turner 1973). In these terms ρ is replaced by $\rho = \rho_o$, Equivalently, the vertical scale for variations in the vertical velocity is much less than the vertical scale for variations in $\rho_o(z)$ and fluctuating density changes due to local pressure variations are negligible. The latter implies that the fluid can be treated as incompressible which excludes sound and shock waves.

In addition to the Boussinesq approximation, Bryan (1969) imposed the the hydrostatic approximation which implies that vertical pressure gradients are due only to density. When horizontal scales are much greater than vertical scales, the hydrostatic approximation is justified and, in fact, is identical to the long-wave approximation for continuously stratified fluids. According to Gill (1982), the ocean can be thought of as being composed of thin sheets of fluid in the sense that the horizontal extent is very much larger than the vertical extent[3]. It should therefore come as no surprise that most of the energy associated with motion lies in components with horizontal scales much larger than vertical scales.

Bryan (1969) also made the rigid lid approximation to filter out external gravity waves. The speed of these waves places a severe limitation on economically solving the equations numerically. As noted above, surface displacements are relatively small. Their affect on the solution is represented as a pressure against the rigid lid at the ocean surface.

---

[1] First introduced by Boussinesq in 1903.

[2] In MOM 1 and the Cox versions of the model, ρ_o was set to 1.0 gm/cm^3 (an error of 3.5% relative to the accepted value of 1.035 gm/cm^3) to eliminate a few multiplies in the momentum equations for reasons of computational speed. MOM 2 uses $\rho_o = 1.035\ gm/cm^3$.

[3] Note that this is not valid if the purpose is to accurately model convection where horizontal and vertical scales may be comparable.

Consistent with the above approximations, Bryan (1969) also made the thin shell approximation because the depth of the ocean is much less than the earth's radius which is assumed to be a constant. The Coriolis component and viscous terms involving vertical velocity in the horizontal momentum equations are ignored on the basis of scale analysis and an eddy viscosity hypothesis is invoked. This hypothesis implies that the affect of sub-grid scale motion on larger scale motions can be accounted for in terms of eddy mixing coefficients. Much of the physics since Bryan (1969) revolves around paramaterizing mixing within the ocean.

The continuous equations formulated in spherical coordinates (ϕ is latitude increasing northward with zero defined at the equator, λ is longitude increasing eastward with zero defined at an arbitrary longitude, and z is positive upwards with zero defined at the ocean surface) are:

$$u_t + \mathcal{L}(u) - \frac{uv\tan\phi}{a} - fv = -\frac{1}{\rho_\circ a\cdot\cos\phi}p_\lambda + (\kappa_m u_z)_z + F^u \tag{2.1}$$

$$v_t + \mathcal{L}(v) + \frac{u^2\tan\phi}{a} + fu = -\frac{1}{\rho_\circ a}p_\phi + (\kappa_m v_z)_z + F^v \tag{2.2}$$

$$T_t + \mathcal{L}(T) = (\kappa_h\cdot T_z)_z + \nabla\cdot(A_h\nabla T) \tag{2.3}$$

$$S_t + \mathcal{L}(S) = (\kappa_h\cdot S_z)_z + \nabla\cdot(A_h\nabla S) \tag{2.4}$$

$$w_z = -\frac{1}{a\cdot\cos\phi}\cdot(u_\lambda + (\cos\phi\cdot v)_\phi) \tag{2.5}$$

$$p_z = -\rho\cdot g \tag{2.6}$$

$$\rho = \rho(T,S,p) \tag{2.7}$$

where horizontal friction, advection, and horizontal diffusion are given by

$$F^u = \nabla\cdot(A_m\nabla u) + A_m\{\frac{(1-\tan^2\phi)\cdot u}{a^2} - \frac{2\sin\phi\cdot v_\lambda}{a^2\cos^2\phi}\} \tag{2.8}$$

$$F^v = \nabla\cdot(A_m\nabla v) + A_m\{\frac{(1-\tan^2\phi)\cdot v}{a^2} + \frac{2\sin\phi\cdot u_\lambda}{a^2\cos^2\phi}\} \tag{2.9}$$

$$\mathcal{L}(\alpha) = \frac{1}{a\cdot\cos\phi}\cdot(u\cdot\alpha)_\lambda + \frac{1}{a\cdot\cos\phi}\cdot(\cos\phi\cdot v\cdot\alpha)_\phi + (w\cdot\alpha)_z \tag{2.10}$$

$$\nabla^2\alpha = \frac{1}{a^2\cos^2\phi}\alpha_{\lambda\lambda} + \frac{1}{a^2\cos\phi}(\cos\phi\cdot\alpha_\phi)_\phi \tag{2.11}$$

$$f = 2\Omega\sin\phi \tag{2.12}$$

In the above equations, T and S are potential temperature[4] and salinity, (u, v,w) are the zonal, meridional and vertical velocities, p is the pressure, ρ is the potential density, g is the mean gravity (980.6 cm/sec^2), a is the mean radius of the earth (6370 x 10^5 cm), κ_m and κ_h are

[4]Potential temperature is used because local stability is dependent on potential density gradients. Also, in an adiabatic ocean, both salinity and potential temperature are materially conserved active tracers.

vertical eddy viscosity and diffusivity coefficients (cm^2/sec), A_m and A_h are horizontal eddy viscosity and diffusivity coefficients (cm^2/sec), and $\Omega = \frac{\pi}{43082.0} sec^{-1}$ is taken as the earth's rotation[5] rate. If coefficient A_m is not spatially constant[6], the additional viscous terms of Wajsowicz (1993) are accounted for.

These equations are solved within MOM 2 by dividing the ocean volume into a three dimensional lattice of rectangularly shaped cells of arbitrary size, discretizing the equations within each cell, and solving all cells by finite difference techniques which are discussed in subsequent chapters. The horizontal velocity (u, v) can be divided into two parts: a depth independent or external mode velocity $(\overline{u}, \overline{v})$ representing the barotropic flow and a depth dependent internal mode velocity $(\hat{u}, \hat{v})$ representing the baroclinic flow

$$u = \overline{u} + \hat{u} \tag{2.13}$$
$$v = \overline{v} + \hat{v} \tag{2.14}$$

A rigid lid at the ocean surface implies that the barotropic mode is non-divergent which allows the external mode velocity to be expressed in terms of a stream function ψ by

$$\overline{u} = -\frac{1}{Ha}\psi_\phi \tag{2.15}$$
$$\overline{v} = \frac{1}{Ha \cdot \cos\phi}\psi_\lambda \tag{2.16}$$

where H is the depth from the ocean surface to the bottom. Lateral boundary conditions are no-slip with insulating walls for heat and salt (i.e. no-flux on side walls). At the ocean surface, boundary conditions are supplied for heat, salt, and momentum. At the ocean bottom, there is an insulating condition for heat and salt. Bottom boundary conditions on horizontal velocity may be given as free-slip or a linear bottom drag. Since the bottom is a material surface at $z = -H(\lambda, \phi)$, the bottom boundary condition on vertical velocity (See Gill (1982), Chapter 4) is

$$w = -\frac{u}{cos\phi}H_\lambda - vH_\phi \tag{2.17}$$

If bottom flow exists, it is required to be tangent to the bottom slope. Another way to generate Equation (2.17) is to integrate Equation (2.5) from the surface to the ocean bottom using Equations (2.13),(2.14),(2.15),(2.16) and the condition that $w = 0$ at the surface $z = 0$. The finite difference equivalent of this second method will be used to generate vertical velocities in the interior as well as at the bottom.

Initial conditions typically consist of specifying a density structure through potential temperature and salinity with the ocean at rest. The finite difference equivalent of the continuous equations will be developed in Chapter 11.

2.2 Kinetic energy budget

This section discusses a kinetic energy budget for the ocean primitive equations in their continuous form. For this purpose, the work of Holland (1975) is followed quite closely. Bryan and

[5]The 43082.0 sec is arrived at assuming approximately $86400 * (1 - \frac{1}{366}) = 86164$ seconds in one siderial day. The 366 is used to account for a 1 day co-rotation.

[6]The viscosity coefficient is constant with option *consthmix* but varies with option *smagnlmix*. Options are discussed in Chapter 15.

Lewis (1979), F. Bryan (1986), Treguier (1992), and Goddard (1995) provide further discussions and examples. A diagnostic from MOM 2 provides the corresponding domain averaged budget for the finite difference solutions. The budget for available potential energy (APE), which is also of interest for energy studies, is not provided by the generic MOM 2 options. The previously mentioned references should be consulted for discussions of APE.

2.2.1 Total budget

The horizontal momentum equations are a relevant place to begin a derivation of the kinetic energy equation

$$\frac{\partial u}{\partial t} = -\vec{u}\cdot\nabla u - wu_z + fv + \frac{uv\tan\phi}{a} - \frac{p_\lambda}{a\rho_o\cos\phi} + (\kappa_m u_z)_z + F^u \tag{2.18}$$

$$\frac{\partial v}{\partial t} = -\vec{u}\cdot\nabla v - wv_z - fu - \frac{u^2\tan\phi}{a} - \frac{p_\phi}{a\rho_o} + (\kappa_m v_z)_z + F^v, \tag{2.19}$$

where $\vec{u} = (u, v)$ is the horizontal current, and the frictional terms $F^{\vec{u}} = (F^u, F^v)$ were defined in Equations (2.8) and (2.9). For the following, it will be useful to note the gradient

$$\nabla p = (\frac{\widehat{\lambda}}{a\cos\phi})\,p_\lambda + (\frac{\widehat{\phi}}{a})\,p_\phi + \widehat{z}\,p_z, \tag{2.20}$$

and continuity equation which is the vector form of Equation (2.5)

$$\nabla\cdot\vec{u} + w_z = \frac{1}{a\cdot\cos\phi}[u_\lambda + (v\cos\phi)_\phi] + w_z = 0. \tag{2.21}$$

Also note that for the scalings relevant to the primitive equations, the kinetic energy per unit volume is dominated by the contribution from horizontal currents

$$e \equiv \frac{\rho_o}{2}(u^2 + v^2). \tag{2.22}$$

Taking the scalar product of $\rho_o(u, v)$ with the horizontal momentum equations (2.18) and (2.19) yields

$$e_t = -(\vec{u}\cdot\nabla e + we_z + \vec{u}\cdot\nabla p) + \rho_o\,\vec{u}\cdot F^{\vec{u}} + \rho_o\,u\,(\kappa_m u_z)_z + \rho_o\,v\,(\kappa_m v_z)_z. \tag{2.23}$$

The time tendency of the kinetic energy is, therefore, determined by the combined effects of the advection of kinetic energy, $-(\vec{u}\cdot\nabla e + we_z)$, the work done by pressure $-\vec{u}\cdot\nabla p$, and the change in energy due to frictional forces

$$\begin{aligned} F &\equiv \rho_o\,\vec{u}\cdot F^{\vec{u}} + \rho_o\,u\,(\kappa_m u_z)_z + \rho_o\,v\,(\kappa_m v_z)_z \\ &= \rho_o\,\vec{u}\cdot F^{\vec{u}} + \rho_o\,(\kappa_m\vec{u}\cdot\vec{u}_z)_z - \rho_o\,\kappa_m\,\vec{u}_z\cdot\vec{u}_z. \end{aligned} \tag{2.24}$$

Using the continuity equation (2.21) and hydrostatic relation $p_z = -g\rho$, the local kinetic energy budget becomes

$$\begin{aligned} e_t &= -\nabla\cdot(e\vec{u}) - (ew)_z - \vec{u}\cdot\nabla p + F \\ &= -\nabla\cdot(e\vec{u}) - (ew)_z - \nabla\cdot(p\vec{u}) - pw_z + F \\ &= -\nabla\cdot[(e+p)\,\vec{u}\,] - [(e+p)\,w\,]_z + wp_z + F \\ &= -\nabla\cdot[(e+p)\,\vec{u}\,] - [(e+p)\,w\,]_z - \rho g w + F. \end{aligned} \tag{2.25}$$

It is of interest to integrate the previous local budget over some fixed volume to determine a finite domain budget

$$\begin{aligned}\frac{\partial}{\partial t}\int d\vec{x}\; e = & - \int (e+p)\,\vec{u}_3\cdot\hat{n} dS - \int d\vec{x}\;\rho g w + \rho_o \int d\vec{x}\;\vec{u}\cdot F^{\vec{u}} \\ & + \rho_o \int d\vec{x}\;(\kappa_m \vec{u}\cdot\vec{u}_z)_z \; - \rho_o \int d\vec{x}\;\kappa_m \vec{u}_z\cdot\vec{u}_z, \end{aligned} \tag{2.26}$$

where $\vec{u}_3 = (u,v,w)$, S is the boundary of the domain, $\hat{n}$ is the outward normal on the boundary, $d\vec{x} = a^2\; d(\sin\phi)\; d\lambda\; dz = a^2\;\cos\phi\; d\phi\; d\lambda\; dz$ is the volume measure, and dS is the measure on the particular boundary surface. An interesting domain is one whose upper boundary (z_{up}) is the upper surface of the ocean[7], since that is where energy is input (neglecting interior geothermal sources), and whose lower boundary is the bottom topography $z_{bottom} = -H(\lambda,\phi)$. For this domain, define the volume average $<> \equiv V^{-1}\int d\vec{x}$, where the domain volume is given by

$$V = a^2 \int H(\lambda,\phi)\, d(\sin\phi)\, d\lambda. \tag{2.27}$$

Volume averaging the kinetic energy equation yields

$$\begin{aligned} E_t = & - V^{-1}\int (e+p)\,\vec{u}_3\cdot\hat{n} dS - <\rho g w> + \rho_o <\vec{u}\cdot F^{\vec{u}}> - \rho_o <\kappa_m \vec{u}_z\cdot\vec{u}_z> \\ & + \rho_o V^{-1}\int dS\;\vec{u}\cdot\vec{\tau}_{wind} - \rho_o V^{-1}\int dS\;\vec{u}\cdot\vec{\tau}_{bottom}, \end{aligned} \tag{2.28}$$

where $E \equiv < e >$. This budget employed the upper and lower boundary conditions on the horizontal currents

$$\rho_o\kappa_m(u_z,v_z)_{z=z_{up}} \equiv (\tau^\lambda,\tau^\phi)_{wind} \tag{2.29}$$

$$\rho_o\kappa_m(u_z,v_z)_{z=-H} \equiv (\tau^\lambda,\tau^\phi)_{bottom}, \tag{2.30}$$

with $\vec{\tau}_{wind}$ the wind stress, $\vec{\tau}_{bottom}$ the bottom stress, and the currents dotted into each of these stresses are taken as the surface and bottom currents, respectively.

Each term in the previous budget admits an interpretation. First, the terms

$$A + G \equiv -V^{-1}\int e\;\vec{u}_3\cdot\hat{n} dS - V^{-1}\int p\;\vec{u}_3\cdot\hat{n} dS \tag{2.31}$$

represent, respectively, the advective redistribution of kinetic energy through the boundary and the performance of mechanical work by pressure along the boundary. Both of these terms vanish for domains whose only open boundary is the ocean surface, since the normal velocity $\vec{u}_3\cdot\hat{n}$ vanishes on the boundary of such a domain. The winds perform work on the system as represented by

$$W \equiv \rho_o V^{-1}\int dS\;\vec{u}\cdot\vec{\tau}_{wind}. \tag{2.32}$$

Buoyancy effects perform work as represented by the term

$$B \equiv - <\rho g w> . \tag{2.33}$$

Finally, there is the viscous dissipation of energy in the interior, walls, and bottom

$$D \equiv \rho_o <\vec{u}\cdot F^{\vec{u}}> - \rho_o <\kappa_m \vec{u}_z\cdot\vec{u}_z> - \rho_o V^{-1}\int dS\;\vec{u}\cdot\vec{\tau}_{bottom} \tag{2.34}$$

The budget for this volume therefore takes the form

$$E_t = A + G + W + B + D, \tag{2.35}$$

where, again, $A = G = 0$ for a closed domain such as the World Ocean.

[7] $z_{up} = 0$ for rigid lid.

2.2.2 External and internal mode budgets

As discussed by Bryan (1969), the primitive equations are conveniently separated into two general modes of flow: the *external* or *barotropic* mode, which represents the depth averaged flow; and the *internal* or *baroclinic* mode, which is the depth dependent flow. It is useful to consider the energetics of these two modes. For this purpose, introduce the depth averaging operator

$$\overline{(\ )} \equiv \frac{1}{H}\int_{-H}^{0}(\ )dz. \tag{2.36}$$

Denote deviations from the vertical average by $\widehat{(\ )}$. The horizontal velocity components are split into the two terms

$$(u,v) = (\overline{u},\overline{v}) + (\widehat{u},\widehat{v}). \tag{2.37}$$

The external mode's kinetic energy is given by $\overline{e} = (\rho_o/2)(\overline{u}\ \overline{u} + \overline{v}\ \overline{v})$; note that $\overline{e} \neq (\rho_o/2)\overline{\vec{u}\cdot\vec{u}}$. The budget for $\overline{e}$ is derived by taking the scalar product of $\rho_o\overline{\vec{u}}$ with the depth averaged momentum equations (2.18) and (2.19); an operation which yields

$$\begin{aligned}\overline{e}_t = & - \rho_o(\overline{u}\,\overline{\vec{u}\cdot\nabla u} + \overline{v}\,\overline{\vec{u}\cdot\nabla v} + \overline{u}\,\overline{wu_z} + \overline{v}\,\overline{wv_z}) + a^{-1}\rho_o\tan\phi(\overline{u}\,\overline{uv} - \overline{v}\,\overline{uu}) \\ & - \overline{\vec{u}}\cdot\overline{\nabla p} + \rho_o\overline{\vec{u}}\cdot\overline{(\kappa_m\vec{u}_z)_z} + \rho_o\overline{\vec{u}}\cdot\overline{F^{\vec{u}}}.\end{aligned} \tag{2.38}$$

The surface and bottom boundary conditions (2.29) and (2.30) bring this expression to the form

$$\begin{aligned}\overline{e}_t = & - \rho_o(\overline{u}\,\overline{\vec{u}\cdot\nabla u} + \overline{v}\,\overline{\vec{u}\cdot\nabla v} + \overline{u}\,\overline{wu_z} + \overline{v}\,\overline{wv_z}) + a^{-1}\rho_o\tan\phi(\overline{u}\,\overline{uv} - \overline{v}\,\overline{uu}) \\ & - \overline{\vec{u}}\cdot\overline{\nabla p} + \overline{\vec{u}}\cdot\vec{\tau}_{wind} - \overline{\vec{u}}\cdot\vec{\tau}_{bottom} + \rho_o\overline{\vec{u}}\cdot\overline{F^{\vec{u}}}.\end{aligned} \tag{2.39}$$

Next, consider the volume average $<>$ as defined in the previous section. This averaging defines the volume averaged external mode kinetic energy

$$< \overline{e} > \equiv \overline{E} \equiv V^{-1}\int d\vec{x}\,\overline{e} = \frac{a^2\int d(\sin\phi)\,d\lambda\ H\,\overline{e}}{a^2\int d(\sin\phi)\,d\lambda\ H}. \tag{2.40}$$

Taking this volume average on equation (2.39) yields

$$\begin{aligned}\overline{E}_t = & - \rho_o < \overline{u}\,\overline{\vec{u}\cdot\nabla u} + \overline{v}\,\overline{\vec{u}\cdot\nabla v} + \overline{u}\,\overline{wu_z} + \overline{v}\,\overline{wv_z} > - < \overline{\vec{u}}\cdot\overline{\nabla p} > \\ & + a\rho_o V^{-1}\int \sin\phi\, d\phi\, d\lambda\ H\,(\overline{u}\,\overline{uv} - \overline{v}\,\overline{uu}) + \rho_o < \overline{\vec{u}}\cdot\overline{F^{\vec{u}}} > \\ & + V^{-1}\int d(\sin\phi)\,d\lambda\ H\,\overline{\vec{u}}\cdot\tau_{wind} - V^{-1}\int d(\sin\phi)\,d\lambda\ H\,\overline{\vec{u}}\cdot\tau_{bottom}.\end{aligned} \tag{2.41}$$

These terms represent, respectively, the work done per unit volume on the external mode from certain nonlinear terms

$$N_e \equiv < \overline{u}\,\overline{\vec{u}\cdot\nabla u} + \overline{v}\,\overline{\vec{u}\cdot\nabla v} + \overline{u}\,\overline{wu_z} + \overline{v}\,\overline{wv_z} >, \tag{2.42}$$

pressure work

$$B_e \equiv - < \overline{\vec{u}}\cdot\overline{\nabla p} >, \tag{2.43}$$

a nonlinear term associated with the spherical metric

$$M_e \equiv a\rho_o V^{-1}\int \sin\phi\, d\phi\, d\lambda\ H\,(\overline{u}\,\overline{uv} - \overline{v}\,\overline{uu}), \tag{2.44}$$

viscous dissipation plus bottom topography

$$D_e \equiv \rho_o < \overline{\vec{u}} \cdot \overline{F^{\vec{u}}} > -V^{-1} \int d(\sin\phi)\, d\lambda\, H\, \overline{\vec{u}} \cdot \tau_{bottom}, \tag{2.45}$$

and wind forcing

$$W_e \equiv V^{-1} \int d(\sin\phi)\, d\lambda\, H\, \overline{\vec{u}} \cdot \tau_{wind}. \tag{2.46}$$

The external mode kinetic energy equation thus takes the form

$$\overline{E}_t = N_e + B_e + M_e + D_e + W_e. \tag{2.47}$$

For a flat bottom, rigid lid ocean, which is a common idealized model domain, pressure forces can do no work on the external mode. To prove this property, it is useful to start with the identity

$$\begin{aligned} \nabla \overline{p} &= \nabla \left(H^{-1} \int_{-H}^{0} dz\, p \right) \\ &= -\overline{p}\, \nabla \ln H + p_{bottom} \nabla \ln H + \overline{\nabla p} \\ &= \overline{\nabla p} + (p_{bottom} - \overline{p})\, \nabla \ln H, \end{aligned} \tag{2.48}$$

where p_{bottom} is the pressure at $z = -H$. The second term vanishes for a flat bottom domain ($\nabla H = 0$) and so the pressure work B_e defined in equation (2.43) becomes

$$B_e = - < \overline{\vec{u}} \cdot \overline{\nabla p} > = - < \overline{\vec{u}} \cdot \nabla \overline{p} > . \tag{2.49}$$

The continuity equation $\nabla \cdot \vec{u} + w_z = 0$ implies $\nabla \cdot \overline{\vec{u}} = w(z = -H) - w(z = z_{up})$, where $w(z = z_{up}) = 0$ for a rigid lid and $w(z = -H) = 0$ for a flat bottom. Therefore, $\overline{\vec{u}} \cdot \nabla \overline{p} = \nabla \cdot (\overline{\vec{u}}\, \overline{p})$, which vanishes when averaged over a closed domain on which the normal velocity vanishes.

The total kinetic energy E discussed in the previous section is given by the sum of the external and internal mode energies $\overline{E} + \widehat{E}$ since $< \overline{u} \cdot \widehat{u} > = < \overline{v} \cdot \widehat{v} > = 0$. Therefore, the domain averaged internal mode kinetic energy $\widehat{E}$ is simply the total energy E minus the external mode energy $\overline{E}$. Equivalently, $\overline{E_t} = (\rho_o/2) < \overline{\vec{u}} \cdot \vec{u}_t >$ and $\widehat{E_t} = (\rho_o/2) < \widehat{\vec{u}} \cdot \vec{u}_t >$, which is how MOM 2 computes the kinetic energy budgets.

Section 2.2 contributed by
Stephen M. Griffies
smg@gfdl.gov

Chapter 3

Dataflow

3.1 Memory management

Productivity is related to total throughput of a computer system over time. The throughput is limited by how well a mix of jobs fits into available storage (memory and disk) and how fast the mix executes. At GFDL, the job mix is a combination of production models, analysis, development and interactive work. It is time dependent and is determined by guidelines intended to optimize total throughput. Any model ties up a certain amount of memory resource. Although seemingly unimportant in low resolution studies, it is crucial in high resolution ones where large domains can easily exceed the computational storage capacity of the system. How much storage is enough? To resolve most eddy structures adequately would require a resolution of about $1/12^{\circ}$ which would take about 930MW for one time level of one variable assuming 100 vertical levels[1]. Even low resolution models that execute while wastefully using memory, limit the number of jobs in the mix and therefore overall throughput. Minimizing model memory requirements need not negatively impact factors affecting scientific productivity indicated in Chapter 1. In fact, there are unexpected gains to be realized.

To integrate the equations detailed in Chapter 2, a volume of ocean is divided into a large number of rectangularly shaped cells within which equations are solved by finite difference techniques. Storage for each variable must be allocated for each cell. If storage were to be allocated entirely within memory, the maximum attainable resolution would be severely limited[2]. This restriction can be greatly relaxed by allocating total storage on a secondary device such as disk and allocating only enough memory to integrate equations for one slice of the ocean's volume at a time. Successively reading, integrating, and writing slices back to disk[3] allows equations to be solved for the entire volume of ocean with significantly less memory in comparison to total storage requirements.

[1]At GFDL, the eight processor CRAY YMP had 32MW of central memory which had been upgraded to 64MW within the last year of its lifetime. Solid state disk space was 256MW.

[2]The assumption is that memory is a precious resource which is to be conserved. Historically, this has been true and the expectation is that it will continue to be in the future.

[3]The viability of this depends on disk access speed. Solid State Disk on the CRAY YMP is fast enough to allow this to work well. Slower disk access can also work if the reads from disk are buffered by the work involved in updating the slice. In practice this is not difficult to implement as long as the slices are to be accessed in a predetermined way.

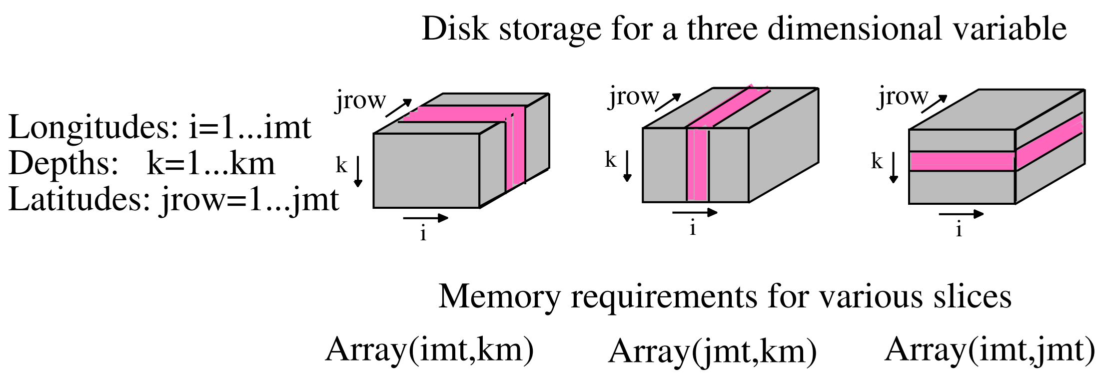


Figure 3.1: Various ways to slice a three dimensional volume of data on disk and the corresponding dimensions of the slice in memory.

There are various ways to slice through a volume of data on disk. As an example, refer to Figure 3.1 and consider the disk storage needed to contain a three dimensional block of cells arranged such that there are $i = 1 \cdots imt$ longitudes, $jrow = 1 \cdots jmt$ latitudes, and $k = 1 \cdots km$ depth levels. As indicated, slicing the volume in various ways and reading the data into memory implies that in memory, the slice can be dimensioned as Array(*imt,km*), Array(*jmt,km*), or Array(*imt,jmt*). Perhaps the most intuitive way of dimensioning arrays is Array(*imt,jmt*). However, in general, the number of model latitudes is comparable to the number of longitudes but the number of depth levels is typically 1/5 to 1/10 the number of longitudes. This eliminates dimensioning slices as Array(*imt,jmt*) because it is too wasteful of memory[4]. Memory requirements for dimensioning slices as Array(*jmt,km*) are comparable to those for dimensioning as Array(*imt,km*). However, dimensioning as Array(*jmt,km*) is less favorable based on other considerations: chiefly, the desire to reference data sections along constant circles of latitude and the ease and speed of performing zonal integrals.

The reason that slices are dimensioned as Array(*imt,km*) instead of as Array(*km,imt*) is largely historical and based on speed issues: the inner dimension is the vector dimension and longer vectors execute faster than shorter ones[5] on vector computers. Essentially, these ideas led to dimensioning of arrays as Array(*imt,km*) slices along constant circles of latitude.

Looking toward the future, indications are that size of cache (fast memory) will have significant impact on speed. Smaller sized arrays are more likely to fit within available cache than larger sized arrays resulting in speed improvements. In fact, when multitasking, fitting arrays into available cache is the reason for observed super linear speed ups as the number of processors are increased.

3.2 Dataflow between Disk and Memory

The idea of slicing volumes of data along lines of constant latitude was outlined in Section 3.1. Refer to Figure 3.2 and note that there are two disks: one for holding latitude rows

[4]Sectioning arrays this way severely limits the size of high resolution designs. It also leads to fewer jobs in the job mix with correspondingly lower total throughput

[5]Vector startup time can be significant for short vectors. In general, two dimensional variables will not collapse to one long vector on the CRAY YMP because operations along the first dimension typically do not include boundary cells $i = 1$ and $i = imt$. In the k dimension, limits are sometimes a function of i and j.

(slices) at time level $\tau - 1$ and another for holding latitude rows (slices) at time level τ. Each disk contains *jmt* latitudes stacked from southernmost (row 1 at the bottom) to northernmost (row *jmt* at the top) and each latitude row corresponds to a slice through the volume of all three dimensional[6] prognostic variables along a line of constant latitude. Assuming one processor, enough processor memory is assigned to hold three latitudes worth of prognostic variables at two time levels plus a work area. The size of the work area varies according to which options are enabled. In general, space is required to hold fluxes of quantities defined at cell faces. Depending on options, space for diffusive coefficients defined at cell faces may also be needed. Solving the equations by second order finite difference techniques necessitates accessing nearest neighbors in space and time which requires this amount of memory.

The process starts by reading data from the first three adjacent latitudes slices on the $\tau - 1$ disk into the processor's memory followed by data from the three corresponding latitude rows on the τ disk. Using boundary rows $j = 1$ and $j = 3$, equations (physics and dynamics) are solved for cells in the central latitude row $j = 2$ and written back to latitude row $jrow = 2$ on the $\tau - 1$ disk. Note that $tau - 1$ data in row $jrow = 2$ is destroyed in the process. Subsequent reading, calculating, and writing, overwrites latitude rows $jrow = 3$ through *jmt-1* on the $\tau - 1$ disk with $\tau + 1$ data. A more detailed description of this will be given in the next section.

3.3 The Memory Window

The memory window in MOM 2 is a generalization of the slab approach used in MOM 1 and prior version of the model where three latitudes rows were kept in memory. This generalized approach in MOM 2 is capable of simulating the older method but allows for greater flexibility. Some of the advantages of this memory window are:

- Higher order finite difference schemes and parameterizations which require access to more than three latitude rows can be implemented in a straight forward manner.

- There is a reduction in the number of names required for variables. For example, in MOM 1 and previous incarnations, tracers required three names: one for the row being computed, one for the row to the north and another for the row to the south. In MOM 2, there is only need for one name: rows to the north and south are accessed by meridional indices *j-1* and *j+1*.

- Essentially all prognostic variables are subscripted by three spatial dimensions as if infinite central memory were available. However, the actual memory needed is controlled by the size of the memory window. Equations and coding looks the same, regardless of how large or small the memory window is.

- Increases in speed can be realized when opening up the memory window even on a single processor. This is the case when lots of diagnostic options are enabled. The reason is less redundant computations are needed.

- Substantial amounts of memory and disk space can be saved when multitasking with fine grained parallelism as compared to the coarse grained parallelism used in MOM 1. However, it should be noted that the memory window can duplicate the coarse grained approach to multitasking used in MOM 1. Also, when the memory window is fully opened, there is no dataflow between memory and disks because disks are not needed. All data is retained within the memory window.

[6]Referring to spatial dimensions.

3.3.1 A description.

Refer to Figure 3.3a which schematically shows the arrangement of all three dimensional prognostic data on a $\tau - 1$ and τ disk. Each disk contains two components of velocity and two tracers but others may be added. In general only two time levels are required on disk because $tau + 1$ data can usually be written over the $tau - 1$ disk area[7].

Consider a longitudinal slice through all data on both disks indicated by the colored section in Figure 3.3a. When this data is read into memory, it is stored in the memory window shown in Figure 3.3b. For purposes of an example, it is opened wide enough to hold six latitudes. The size of the memory window is arbitrary and is controlled by setting parameter *jmw* in file *size.h* (*jmw=6* for this example). The minimum size of the memory window is *jmw=3* (which would mimic the slab architecture of MOM 1 and earlier versions of the model) and the maximum size is *jmw=jmt*. Each prognostic variable within the window is dimensioned by indices[8] *i,k,j* to denote longitude, depth, and latitude along with a fourth index to denote prognostic component (e.g. for velocity, an index of 1 would reference the zonal component of velocity and 2 would reference the meridional component. For tracers, a 1 would reference temperature and a 2 would reference salinity) and a fifth index to denote time level (e.g. either $\tau - 1$, τ, or $\tau + 1$.).

A work area is also shown in the memory window. The size of this area varies depending on which options are enabled. In general, space is required to hold diffusive and advective fluxes of prognostic quantities defined on the faces of each cell. These fluxes are also dimensioned by indices *i,k,j* but without a time index since they are recalculated for each prognostic variable to conserve memory. Additionally, some options require diffusive coefficients with spatial dependence. In this case, three dimensional arrays are used for diffusive coefficients which are also defined on cell faces.

Within the memory window, equations are solved for latitudes marked with a red color. The latitudes marked with a blue color are used as boundary cells. Figure 3.3c is a simplified schematic of the detailed window shown in Figure 3.3b. Note that the first row in the memory window is *j=1* and the last row is *j=jmw* but computations typically go from *j=jsmw=2* through *j=jemw=jmw-1*. Prognostic quantities (temperature, salinity, horizontal velocity components) within the memory window are dimensioned in the meridional direction by jmw although not all quantities are dimensioned this way. For instance, if a quantity 'q' involves meridional averages or differences of temperature and it were dimensioned as $q(imt, km, jmw)$, then 'q(i,k,jmw)' would referencer temperature at index $j + 1$ which is out of bounds because the meridional dimension of temperature is jmw. Such quantities are dimensioned as 'q(imt,km,1:jemw)' and can only be computed within the range $1 : jemw$. When computing these quantities, the idea is to do so over their full dimension taking into account that it may be less than jmw. In many of the following figures, the memory window will be represented in its simplist form given schematically in Figure 3.3d.

3.3.2 How it works.

A formal description of dataflow through a memory window of arbitrary size will now be given followed by a specific example of a minimum memory window size of $jmw = 3$. The case of a memory window opened to $jmw = 5$ is given as an example of fine grained multitasking in Section 3.5. To simplify figures, a memory window schematic of the form shown in Figure 3.3d is used.

[7]This is not case when multitasking with option *coarse_grained_parallelism*. In this case, a third disk is needed.
[8]Reasons for this ordering are given in Section 3.1.

Formal description.

Even though this description pertains to an arbitrary sized memory window, it amy be helpful to look at Figure 3.4. In all that follows, the global index $jrow$ refers to the latitude row on disk and the local index j refers to the latitude row within the memory window. Let there be jmt latitudes arranged monotonically from south to north with $jrow = 1$ representing the southernmost latitude and $jrow = jmt$ the northernmost one. Assume an arbitrary memory window of size jmw with $3 \le jmw \le jmt$. The first usage of the memory window loads latitude rows $jrow = 1$ through $jrow = jmw$ into memory window rows $j = 1$ through $j = jmw$. The number of rows where prognostic quantities are computed within a memory window is given by

$$ncrows \le jmw - 2 \tag{3.1}$$

The starting row for these computations is always $j = 2$. Typically for second order accurate numerical schemes, $ncrows = jmw - 2$ although $ncrows$ may be less for higher order schemes or those needing special treatment[9]. If latitude rows $jrow = 1$ and $jrow = jmt$ are used only as boundary rows, then the number of memory windows $num\_windows$ needed to update prognostic variables is given by

$$num\_windows = int((jmt-2)/ncrows) + (jmt-3)/(ncrows \cdot (int((jmt-2)/ncrows))) \tag{3.2}$$

where the $int()$ function represents the integer part of a quantity and the first term is the number of full memory windows needed to update prognostic variables on latitudes $jrow = 2$ through $jrow = jmt - 1$. To account for the last few rows, there may be an extra memory window which is only partially full and this is given by the second term in the calculation of $num\_windows$.

If $num\_windows > 1$, then preparation is made for computing a second group of latitudes by copying data from the northernmost $jmw - ncrows$ rows in the memory window into the southernmost $jmw - ncrows$ rows in the memory window. The ordering of the copy is important else data copied in one operation will be wiped out by the next copy. A general prescription for copying data southward (equivalent to moving the memory window northward) is given by

- For variables dimensioned as array(imt,km,jmw), copy all elements of i and k as follows:
  array(i,k,1) = array(i,k,jemw)
  array(i,k,2) = array(i,k,jmw)
- For variables dimensioned as array(imt,km,1:jemw), copy all elements of i and k as follows:
  array(i,k,1) = array(i,k,jemw)
- For variables dimensioned as array(imt,km,jsmw:jmw), copy all elements of i and k as follows:
  array(i,k,jsmw) = array(i,k,jmw)
- For variables dimensioned as array(imt,km,jsmw:jemw), no copy is necessary.

Latitudes within the memory window with local index j are related to latitudes on disk with global index $jrow$ by an offset $joff$ which is calculated as

$$joff = (mw - 1) * ncrows \quad for \quad mw = 1 \; to \; num\_windows \tag{3.3}$$

[9]For example, option *pressure_gradient_average* requires tracers to be solved on more rows than velocities within the memory window although this is easily done.

After the copy operation is completed, data is read from the next set of latitudes (jrows) on disk which are given by

$$js + joff \leq jrow \leq je + joff \tag{3.4}$$

into the memory window starting at row js and ending at row je where

$$\begin{aligned} js &= jmw - ncrows + 1 & (3.5)\\ je &= min(jmw, jmt - joff) & (3.6) \end{aligned}$$

The function 'min' takes the minimum of jmw and $jmt - joff$ to account for a potentially partially full last memory window. This prevents index out of bounds.
After each load of the memory window, some intermediate computations (Refer to Figure 11.1 for a flowchart of these steps) are needed to support solving of the prognostic equations. All such intermediate quantities are computed over the entire range of their dimensions within the memory window before solving the prognostic computations. Prognostic equations are solved starting at row $jscalc$ and ending at row $jecalc$ in the memory window. These rows are given by

$$\begin{aligned} jscalc &= 2 & (3.7)\\ jecalc &= min(jsmw + ncrows - 1, jmt - 1 - joff) & (3.8) \end{aligned}$$

where again the function 'min' limits $jecalc$ to memory window rows corresponding to latitude rows less than or equal to $jmt - 1$.

Specific example.

Refer to Figure 3.4 which gives a schematic of the disk and memory configuration for a minimum memory window size of $jmw = 3$. This is the minimum sized memory window appropriate for one processor and is represented by a schematic of the form given in Figure 3.3d. As indicated in Figure 3.4, two time levels of three dimensional prognostic data reside on disk in latitude rows from the southernmost (jrow = 1) to the northernmost (jrow = jmt). Initially, the first three latitude rows for two time levels are read into the memory window, the central row is updated to $\tau + 1$ and written to the second row on the $\tau - 1$ disk. The memory window is moved northward one row by copying the top two rows (τ and $\tau - 1$) into the bottom two rows and reading latitude row $jrow = 4$ from the τ and $\tau - 1$ disks into row $j = 3$ in the memory window. The ordering of the copies is important otherwise the second copy will destroy results from the first copy[10]. The process is repeated, calculating central row $j = 2$ and writing it to the $\tau - 1$ disk. The circled number represents the offset *joff* relating the latitude row ($jrow$) on disk to the local row (j) in the memory window . The value of *joff* is an indication of how far the memory window has been moved northward. Refer to Section 5.2 for a description of how *joff* relates to variables.

To actually see how the memory window operates for various values of *jmw*, disable all diagnostics in script *run_mom*, enable option *trace_indices*, set the integration time to a few time steps, and execute *run_mom* as described in Section 15.19. Alternatively, a memory

[10]This is reminiscent of the Cox implementation which, however, lacked generality due to the absence of dimensioning with a j index.

window simulator can be executed using script *run_mwsim*. Although this simulator is set up to show what happens in coarse grained parallelism, removing this option in the run script and changing the size of jmw and jmt in file *mwsim.F* will be instructive.

3.3.3 Dataflow in higher order schemes

In Section 3.3.2, dataflow between memory and disk was described for a memory window of minimum size *jmw=3*. In higher order schemes the minimum size increases.

For instance, option *biharmonic* is a fourth order horizontal mixing scheme. Fourth order schemes require two additional rows so the minimum size of the memory window is *jmw=5*. Using five rows, the essential point is to calculate second order fluxes defined on the three central rows. Meridionally differencing these second order fluxes yields one fourth order mixing term defined at the central computed row. However, since calculations proceed from south to north and the southern most latitude is land, the second order flux is assumed to be zero at *jrow=1*. If these northward fluxes are saved and moved as the memory window moves northward, there is no need to have a memory window with $jmw = 5$ rows. The minimum size of the memory window is reduced to *jmw=4* and the number of calculated rows within the memory window is $ncrows = jmw - 3 = 1$. This is typical for all other fourth order schemes in MOM_2 such as the fourth order advection of tracers. All such schemes require option *fourth_order_memory_window* which is automatically enabled in file *size.h* when any of the existing fourth order schemes are enabled. This is described further in Section 15.19.3.

Refer to Figure 3.5. Using results from the formal treatment in the previous section, dataflow for fourth order schemes is similar to Figure 3.4 with only one row being computed ($ncrows = jmw - 3 = 1$) and one additional latitude row in the north. In order to move this memory window northward, data must be copied from the three northernmost rows ($jmw - ncrows = 3$) instead of from two northernmost rows as for second order schemes. Figure 3.6 indicates what happens as this memory window is opened further to *jmw=5* rows. Again, dataflow is similar to Figure 3.7 except that two central rows are calculated ($ncrows = jmw - 3 = 2$).

Calculations always proceed up to latitude row $jrow = jmt - 1$ even with higher order schemes!. There are no out of bounds references because meridional indexing is limited to a maximum at latitude $jrow = jmt$ and a maximum corresponding memory window row given by $j = min(j + joff, jmt) - joff$. To accommodate higher order schemes when a fully open memory window $jmw = jmt$ is used, meridional fluxes are set to zero at latitude $jrow = jmt$ which allows calculations to proceed through latitude row $jrow = jmt - 1$.

3.4 Data layout on disk

All three dimensional prognostic data resides on disk. The ordering of the data is the same for each latitude row: The zonal component of velocity (internal mode only) is first, followed by the meridional component of velocity (internal mode only) and then each tracer $n = 1, nt$. Therefore there are $2 + nt$ data fields for each latitude row and each data field is written as if it were dimensioned as *data(imt,km)*.

3.5 Parallelization

Parallelization is achieved through multitasking which is spreading one job across multiple processors. When is it necessary to multitask? When the number of processors exceeds the number of jobs in the system, processors will stand idle and overall system efficiency will degrade unless multitasking is used. In an operational environment, a forecast may be required every

4 hours and if the model takes 8 hours on one processor then it makes sense to multitask. Other reasons include when long running experiments take too long to complete or when it is impossible to exhaust a monthly computer time allocation using one processor. Multitasking has not been used in any significant way on the Cray C90 with 16 processors at GFDL. The reason is that there are about 30 batch jobs in the system at all times and the efficiency averaged over 24 hours is about 93%. It is anticipated that with an upgrade to 30 processors, overall system efficiency will drop significantly unless multitasking is used. Basically, there are two approaches to multitasking in MOM 2:

- The *fine grained* approach where parallelism is defined at the level of each *nested do loop*. Here, all processors work simultaneously on each *nested do loop* for each group of latitudes. This implies that there are many parallel regions.
- The *coarse grained* approach where parallelism is defined at the level of latitude rows. For instance, all work associated with solving the equations for one latitude row is assigned to a single processor. All work associated with solving the equations for another latitude row is assigned to a second processor ... and so forth. Then, all processors work simultaneously and independently. This implies that there is only one parallel region.

Both of these approaches and their relative merits and deficiencies are discussed in the following sections.

3.5.1 Fine grained parallelism (autotasking)

This is the simplest form of multitasking. Simplest means fewest things to do and no way to get into trouble. It is equivalent to 'autotasking' on CRAY PVP's (parallel vector processors) like the C90.

How to do it.

There are no options needed to enable fine grained parallelism within MOM 2. All that is required is to set the desired number of processors and open the memory window up as described below. Be sure to set the compiler options needed for multitasking. On a CRAY C90, compiling should be done with the parallel compiler option '-Zp' and setting the desired number of processors with an environment variable[11] The essential thing to keep in mind is that for each nested do loop, each computed row[12] in the memory window is assigned to a separate processor and all processors work simultaneously.

About the only consideration is to insure that the total number of latitude rows which are to be solved $(jmt-2)$ divided by the number of requested processors $(num\_processors)$ is an integral number of rows per processor[13].

$$rows\_per\_processor = (jmt-2)/num\_processors \tag{3.9}$$

Otherwise, a static imbalance will occur where processors will stand idle. Note that the standard test case with $jmt = 61$ will typically not meet this condition for any reasonable number of

[11] Note that the parameter *num_processors* in file *size.h* does not have to be set since it is only used for option *coparse_grained_parallelism*. Only the environment variable NCPUS (which sets the number of processors on the CRAY C90) needs to be set for fine grained parallelism.

[12] A computed row is one where prognostic quantities are updated to time level $\tau+1$.

[13] Realize that just because a specific number of processors are requested is no guarantee that they will all be used when executing in a multi-programming environment. Some may be busy with other jobs. However, for this discussion the assumption will be that the system is dedicated to one single job.

processors. However, setting up the domain in module *grids* to yield $jmt = 66$ latitude rows will meet this condition for 2,4,8,16, and 32 processors. Once the above condition has been met, the memory window size is given by

$$jmw = rows\_per\_processor + 2 \tag{3.10}$$

where the '2' is for the number of buffer rows.

How it works.

Consider Figure 3.7 which portrays the situation when the memory window is opened to $jmw = 5$ rows which is appropriate for fine grained parallelism using $m = 3$ processors[14]. There is much similarity between this and the case of $jmw = 3$ given above in Section 3.3.2 except that three central rows ($j = 2, 3, 4$) are now calculated within each memory window and there are correspondingly fewer northward moves. The memory window moves three rows at a time but still only requires copying[15] data from the top two rows to the bottom two rows as in the case when $jmw = 3$. Note however, that a condition of static imbalance exists when the northernmost latitude rows are being solved. In this example, the above condition for an integral number of rows per processor is not met.

Apart from the condition of static imbalance, there is a possibility of dynamic imbalance which again leads to idle processors. This happens when latitude rows have unequal amounts of work. For instance, some latitudes may have convection and others may not. Or, some latitude rows may be filtered while others may not. In general, there is no easy solution here.

In the case where the memory window is opened all the way to $jmw = jmt$, the appropriate number of processors for fine grained parallelism is $num\_processors = jmt - 2$ and disk space is <u>not</u> used. Data for all three dimensional prognostic variables is retained in the memory window and it is <u>not</u> necessary to move the memory window northward. After all rows have been updated to $\tau + 1$ values, there is <u>no</u> need to copy data into proper locations for the next time step since time level pointers are recalculated to point correctly. This situation is indicated schematically in Fig 3.8.

Bear in mind that as the memory window is opened from minimum to maximum size, details in the equations do not change. All that is required to open the memory window is to specify parameter 'jmw' in file *size.h*. The equations know nothing about the size of the memory window except for the range limits in the meridional *do loops* which surround all calculations.

3.5.2 Coarse grained parallelism (microtasking)

In the early 1990's when MOM 1 was being developed, it was decided that multitasking was a feature to take advantage of on the eight processor CRAY YMP at GFDL. At the time, there were two ways to go about this: either the fine grained approach described in Section 3.5.1 or the coarse grained approach described here.

[14] This can also be used with one processor although it takes more memory. It is surprising at first that the memory required to run the test case with $jmw = jmt = 61$ is only about 2.5 times that required when using option *ramdrive* and $jmw = 3$!

[15] Instead of copying data, indices could be used to point to the proper location within the memory window. The down side is that equations become unreadable when *mod* functions are used to construct indices like j+1. In addition, if the value of these mod functions is stored in subscripted variables, extra computation is incurred for indirect addressing in addition to degrading readability even more. Assigning their value to un-subscripted variables represents a loss of generality when trying to implement higher order schemes. On the CRAY YMP, the copy operation is very efficient.

It was decided to follow the *coarse grained* approach pioneered by Bert Semtner on CRAY XMPs in the 1980's. After substantial effort, a multitasking option was installed in MOM 1. Surprisingly, it turned out that no one at GFDL used it. Looking back, the reason was clear. The benefit from multitasking is reduced wall clock time which meant faster model turn around which sounded like a good thing. However, when a researcher submitted a production model at the end of the day, that researcher didn't care if it sat on the computer for 4 wall clock hours or 8 wall clock hours. What the researcher did care about was that the results were back in the morning and that the model execution time was a minimum. Instead, it was noticed that model execution time, measured by cp time, increased and the model required more memory which in many cases forced models into categories which allowed fewer job submissions and therefore fewer results! Additionally, researchers felt no incentive to multitask because their accounts were being charged by cp time rather than wall clock time. Actually, there was one class of job which would have benefited from multitasking: the high resolution model because results were typically not back by morning. Ironically, these jobs could not be multitasked because their memory requirement exceeded what was available on the system.

How to do it.

For coarse grained parallelism, option *coarse_grained_parallelism* must be enabled. When enabled, the memory window size is automatically set to the minimum size appropriate for enabled options. On shared memory systems (e.g. CRAY T90), the intent is to use this option with option *crayio* or *fio* which keeps all latitude rows on disk (preferably solid state disk). Use of option *ramdrive* is also possible but only as a stepping stone to allow simulations of distributed systems on shared systems.

The number of processors *num_processors* is specified through namelist where it picks up the value of the environment variable NCPUS from the run script. When using option *coarse_grained_parallelism* and *ramdrive* to simulate distributed systems, the setting of *num_processors* is changed to equal the value of parameter *nprocessors* which is required for dimensioning purposes on distributed systems. Note that *num_processors* is the maximum number of processors to be used. On time steps where conflicts arise between diagnostics and coarse grained parallelism, the number of processors used is automatically reduced to one to avoid these problems. In principle, most of the problems involve diagnostic I/O from within the parallel region. Note that the standard test case with $jmt = 61$ will typically lead to a static load imbalance for any reasonable (power of two) number of processors. However, setting up the domain in module *grids* to yield $jmt = 66$ latitude rows will remove the imbalance condition for 2,4,8,16, and 32 processors. Be sure to set the compiler options needed for multitasking. For instance, on a CRAY C90, compiling with the parallel compiler option '-Zp' is also necessary.

How it works.

Dataflow for coarse grained parallelism is similar to what was described in Sections 3.2 and 3.3.2 although there are differences. To actually see a working example, execute script *run_mwsim*. This script indicates how data is cycled between a memory window and disk along with showing which latitude rows are updated by which processor for the case of $jmt = 8$ rows and various numbers of processors. Refer to Figure 3.9 which demonstrates the case for two processors. In comparison to the fine grained approach illustrated in Figure 3.2, note that there are now three disks with the new disk allotted for latitude rows at time level $\tau + 1$. Using only two disks will not work. Each disk still contains *jmt* latitudes but they are now divided into two logical sections. As indicated in Section 3.5.3, this approach is easily extendible to distributed memory systems.

In general, if there were "num_processors" processors, disk space would be divided into "num_processors" logical sections. Each processor would have one memory window of size *jmw=3* and work on latitude rows within one of the logical sections on all three disks. Processor "n" would be assigned the task of reading latitude rows on disks $\tau - 1$ and τ between starting row "jstask(n)" and ending row "jetask(n)" which are given by

$$\begin{aligned} jstask(n) &= max(int((n-1) * float(\frac{jmt-2}{num\_processors}) - jextra + 1.0001), \\ &\quad 1 - jextra) \qquad (3.11) \\ jetask(n) &= min(int(n * float(\frac{jmt-2}{num\_processors}) + 1 + jextra + 1.0001), \\ &\quad jmt + jextra) \qquad (3.12) \end{aligned}$$

where $jextra$ is the number of extra buffer rows (normally this is set to zero except when option *fourth_order_window* is enabled) and the number of calculated rows per task[16] is given by

$$num\_rcpt(n) = jetask(n) - jstask(n) + 1 - (jmw - ncrows) \qquad (3.13)$$

where $(jmw - ncrows)$ is the number of buffer rows.

For the case of two processors, the process starts by reading the first three adjacent latitudes slices from the $\tau - 1$ disk into the memory window of processor #1, followed by three corresponding latitude rows from the τ disk. After the equations are solved for the central memory window row j=2, updated values are written to latitude row jrow=2 on the $\tau + 1$ disk. Subsequently, the memory window is moved northward by copying τ and $\tau - 1$ data from memory window row j=2 into memory window row j=1, followed by copying both time levels from memory window row j=3 into memory window row j=2, and reading latitude row jrow=4 from the *tau* and $\tau - 1$ disks into memory window row j=3. After all equations are solved for the central row in the memory window j=2, updated values are written to row jrow=4 on the $\tau + 1$ disk. This process continues until latitude rows between $jstask(1) + 1$ and $jetask(1) - 1$ on the $\tau + 1$ disk have been updated in a sequential manner.

As indicated in Figure 3.9, the second processor *simultaneously* does the same operations described above using latitude rows $jstask(2)$ through $jetask(2)$. Note that there is some overlap between rows accessed by processor #1 and processor #2. Since the entire pool of disk space is accessible by any processor, this presents no problem. However, in a distributed system where memory local to each processor is being used instead of disk, communication between processors is required for latitude rows adjacent to the memory boundaries. Refer to Section 3.5.3 for details.

Some consequences.

These are the consequences when multitasking with option *coarse_grained_parallelism*.

- Memory requirements increase as m times the size of the memory window where m is the number of processors. This leads to a memory explosion which effectively limits large models to unitasking.
- Three disks are required. For instance, if one processor were to write its $\tau + 1$ data to a row on the $\tau - 1$ disk before another processor read data from that row on the $\tau - 1$ disk, the second processor would get $\tau + 1$ instead of $\tau - 1$ data.

[16]The rows on which prognostic variables are updated to time level $\tau + 1$ and written to disk $\tau + 1$.

- Memory windows are independent of each other and cannot access each other's memory. Therefore, in the limit where the number of processors approach the number of latitude rows, there is a maximum amount of redundant calculations involving meridional gradients and averages on each latitude row. Additionally, each latitude must be read three times per time step from the τ and $\tau - 1$ disks. Calculations involving density, adding external mode to internal mode velocities, and generating land/sea masks must also be done three times for each latitude row.

- Latitude rows are not guaranteed to be processed sequentially from $jrow = 2 \cdots jmt$ which means that results may be written in random order to diagnostic and analysis files.

- Each variable within the model needs to be defined as either shared or private. Mistakes in defining variables are easy to make and difficult to find. This situation limits flexibility and makes model development difficult. The unfortunate fact is that code that works correctly when unitasked is not guaranteed to work correctly when multitasked. The impact of this will be grossly under appreciated until experienced.

- Parallelism can reach well past 95% but parallel efficiency figures are degraded when:

  1. The number of latitude rows divided by the number of processors is not an integer. This is a static imbalance condition which can be accounted for at model design time.
  2. There is a variable amount of work per latitude row. This is a dynamic imbalance condition which may be dependent on space and time. This condition is difficult to remedy and is brought about by filtering, convection, and some diagnostics.

3.5.3 Parallelism on Distributed Systems

Option *distributed_memory* enables MOM 2 to execute on distributed memory systems with initial focus on the CRAY T3E. Refer to Section 3.5.2 which describes the preliminaries. Distributed systems such as the CRAY T3E have significant amounts of memory associated with each processor. Access to disk is relatively slow, so instead of using disk space to retain all three dimensional prognostic data, the disk space is mapped to memory using option *ramdrive*. Option *ramdrive* normally simulates two time levels of latitude data on disk by defining a huge array within memory. However, for multitasking, three distinct time levels for data storage are required. When option *coarse_grained_parallelism* is also enabled, the two time levels are extended to three. The ramdrive is usually configured as one huge pool of shared memory. However, when options *coarse_grained_parallelism*, *ramdrive*, and *distributed_memory* are enabled, the memory pool is distributed among processors (this part is still under development so details can be expected to change). The dimensioning of this distributed memory requires a parameter *nprocessors* which is specified in file *size.h* and sets the value of *num_processors* (which ordinarily would take its value from namelist). As described in Section 3.5.2, processor #1 will contain latitude rows from "jstask(1)" to "jetask(1)", processor #2 will have latitude rows "jstask(2)" to "jetask(2)" and so forth. Details will be hidden within file *odam.F* and the rest of the model will be untouched except for an area below the latitude loop in subroutine *mom* which is where communication between processors will take place.

Recall from Section 3.5.2 that there will be some overlapping of latitude rows adjacent to the ramdrive memory boundary on each processor. After all latitude rows have been updated to $\tau + 1$ data, latitude rows adjacent to the boundary of each memory section will have to be updated. This means that processors which share boundary latitude rows must read each other's memory to update these rows. Once these boundary rows have been updated, everything is in place for the next time step to begin.

Refer to Figure 3.12 for an example of how latitude rows are mapped into the ramdrive area of processor #1 and processor #2 of a "num_processor" processor system. As indicated, five latitudes are contained in each ramdrive area but only three of those latitude rows are updated with $\tau + 1$ data. The others are needed as buffer rows. The five latitude rows are arbitrary. If the total number of rows *jmt* were larger, then each processor would have more rows. The important point is that each processor should contain the same number of computed rows[17]. Within each processor, the memory window updates computed rows to time level $\tau + 1$ one row at a time until all computed rows are updated as indicated by the color red. After all processors have updated their computed rows, data from some of these rows must be copied into corresponding rows on other processors to prepare for the next time step. Communication (reading and/or writing the memory of other processors) is used to update these rows.

Figure 3.13 indicates a similar situation for a fourth order memory window. This time, two buffer rows are required on each side of a computed row and this requires additional communication. Note that the minimum size of the memory window is $jmw = 5$ rows as contrasted to $jmw = 4$ when not multitasking. The extra row is needed for boundary computations.

For all processors from $n = 2, num\_processors$ the following prescribes the communication for a second order memory window:

- copy all data from latitude row "jstask(n)+1" on processor "n" to latitude row "jetask(n-1)" on processor "n-1".
- copy all data from latitude row "jetask(n-1)-1" on processor "n-1" to latitude row "jstask(n)" on processor "n".

When MOM 2 is configured with options which require a fourth order memory window then the following communication is required:

- copy all data from latitude row "jstask(n)+3" on processor "n" to latitude row "jetask(n-1)" on processor "n-1".
- copy all data from latitude row "jstask(n)+2" on processor "n" to latitude row "jetask(n-1)-1" on processor "n-1".
- copy all data from latitude row "jetask(n-1)-3" on processor "n-1" to latitude row "jstask(n)" on processor "n".
- copy all data from latitude row "jetask(n-1)-2" on processor "n-1" to latitude row "jstask(n)+1" on processor "n".

For the above communication to work on a fourth order memory window, there must be at least two computed rows per processor. The CRAY T3E processors can be divided up more or less arbitrarily so that a particular job is not constrained to a number of processors that is a power of two.

3.5.4 Comparing Coarse and Fine Grained Parallelism

In the past, it turned out that the best way to maximize overall computational efficiency and throughput on multiple processors at GFDL was to multi-program (having one or more jobs

[17] A computed row is one where prognostic data is updated to time level $\tau + 1$.

per processor) rather than multitask (using more than one processor per job). This works as long as individual processor speed is fast and there are enough jobs to keep all processors busy. The situation dramatically changes when processors are slow or there are not enough jobs to keep all processors busy. If processors are slow, jobs don't turn around in reasonable time and if enough jobs are not available, processors stand idle. In either case, the system is under-utilized and multitasking increases overall throughput.

With eight processor on a CRAY YMP at GFDL, there were enough jobs to keep all processors busy. With sixteen processors on the CRAY C90, this was still the case and with twenty processors on the CRAY T90 this is still the case. However, at some point it is envisioned that this will no longer be the case and jobs will have to be multitasked.

This section describes advantages and disadvantages of the two approaches to parallelism available in MOM 2. Note that the coarse grained approach is essentially the method used in MOM 1 and comments apply there as well. Consideration is given to memory and disk requirements, redundant calculations, array structure, and parallel efficiency.

Memory requirements

There is a substantial difference in the memory requirements between coarse and fine grained approach to parallelism. One way to see this is to refer to Figure 3.10a, and restrict m memory windows (for use with m processors and coarse grained parallelism) to access m adjacent latitudes. Then there are $2(m-1)$ boundary rows that are redundant (marked with an X). Removing these redundant rows and compressing what remains leads to one memory window of size $m+2$ rows (for use with m processors and fine grained parallelism). Each component of the memory windows in Figure 3.10a is labeled. Computed rows are of memory size C with a work area of size WC. The boundary rows are of memory size B with work area of size WB. β expresses the size ratio of work arrays to prognostic arrays for calculated rows and α expresses the size ratio of work arrays for buffer rows to computed rows. Let *M1* represent the memory needed for m memory windows used with coarse grained parallelism and *M2* represent the memory needed for the corresponding memory window equivalent using fine grained parallelism. An expression for the ratio of memory sizes is given in Figure 3.10b. The table gives the ratio $\frac{M1}{M2}$ as a function of number of processors m and β for a typical value of $\alpha = \frac{1}{3}$. Note that the memory reduction is a strong function of m and a weak function of β. What is not shown is that the memory reduction is also a weak function of α for values that normally would be encountered. The bottom line is that coarse grained parallelism takes <u>twice</u> as much memory on $m = 10$ processors as fine grained parallelism and the same amount of memory for one processor.

Disk requirements

As indicated in Figure 3.10c, $\tau+1$ data can be written to the $\tau-1$ disk without conflict when using fine grained parallelism. Therefore only two disks are required. However, as indicated in Figure 3.9, coarse grained parallelism required three disks. Therefore, coarse grained parallelism requires 50% more disk space than fine grained parallelism. Additionally, since calculations for rows within the memory window proceed sequentially, there is no problem with components of diagnostic files being written in random order in fine grained parallelism.

Redundant calculations

Referring to Figure 3.9, since memory windows are independent of each other, there must be redundant calculations involving latitude rows on either side of the boundary line separating the logical sections on the disks. All meridional gradients and averages need to be computed redundantly for these latitude rows. Also, each latitude must be read three times per time step from the τ and $\tau - 1$ disks. Calculations involving density, adding external mode to internal mode velocities, and generating land/sea masks must also be done three times for these latitude rows. As the number of processors increases, so do the number of logical sections on the disk and in the limit of one processor per latitude, redundancy takes place on all latitude rows.

This results in coarse grained parallelism with lots of processors taking 15% more in cp time than running one processor. This redundancy is not needed in the fine grained approach to parallelism. However, there is still other redundancy in the fine grained approach which can be eliminated by opening up the window all the way to $jmw = jmt$. In this case, land/sea masks are calculated only once, and there is no copying or reading of data. This will speed up the calculations by 8% but the cost will be large amounts of memory. The bottom line is that as the number of processors increases, redundant calculations increase when using coarse grained parallelism but decrease[18] to zero when using fine grained parallelism with a fully opened memory window.

Private/Shared variables

When using fine grained parallelism, there is no issue as to whether variables are private or shared. Essentially, the compiler resolves this issue at the level of each do loop. However, when using coarse grained parallelism, all variables must be typed as being private or shared. This is of no grave consequence when a model is unchanging but is of major concern in a research model where development is actively ongoing.

Automatic parallelization tools such as those from Applied Parallel Research are currently being used for automatically typing variables. Although this approach requires some 'intelligent intervention', results have been very encouraging. Overall, these tools seem to be a rational solution for allowing MOM 2 to be executed on PVP systems like the CRAY T90 as well as distributed systems like the CRAY T3E. Remaining details are currently being worked out. Bingo.

Parallel efficiency

Refer to Figure 3.11 which schematically shows the difference between code structures. Let the coarse grained approach be viewed as the MOM 1 code structure (although MOM 2 can simulate this) and the fine grained approach be viewed as the MOM 2 code structure.

In the coarse grained approach (MOM 1 coding), the parallel region shown in red is the latitude loop which wraps around all of the dynamics and physics. This work can be reduced to a succession of doubly nested *do loops* as indicated where the scalar region is marked in blue and the vector region is marked in green on the a PVP (Parallel Vector Processor) like the CRAY YMP. Each processor gets all the work for one latitude row and all processors work simultaneously on separate latitudes. Since the parallel loop contains the totality of doubly nested *do loops*, there is *no* shortage of work for any processor. To insure a static load balance, $jmt - 2$ should be an integral multiple of the number of processors otherwise some will stand

[18]This is done to conserve memory. However, it is an interesting counter intuitive point that sometimes it is better to do redundant calculations rather than save intermediate results in temporary work arrays. Depending on the speed of the load and store operations compared with the multiply and add operations, if redundant computation contains little work it may be faster to compute redundantly. In practice, this depends on the computer and compiler optimizations. This can be verified by executing script *run_timer* which exercises the timing utilities in module *timer.F* by solving a tracer equation in various ways. It is one measure of a mature compiler when the speed differences implied by solving the equations in various ways is relatively small. On the CRAY YMP the differences are about 10% whereas on the SGI workstation, it can reach 100%

idle. Parallel efficiency can reach well into the upper 90% region although it is sobering to realize that even with a parallel efficiency of 99.5% only about 49 out of 64 processors are used! This is a result of Amdahl's law. To see results for other combinations execute the command 'amlaw' on any CRAY system.

In the fine grained approach (MOM 2 coding), a group of latitudes is solved within an expanded memory window. An outer loop controls the number of these latitude groups[19] and wraps around all of the dynamics and physics. Again, this work can be viewed as a succession of triply nested *do loops*. However, each triplet is now itself a parallel region with the scalar and vector region on the inside. In contrast to coarse grained parallelism (MOM 1 code), all processors work in parallel on each triplet. When one triplet is finished, all processors synchronize and move on to the next triplet and so forth until all loops within the group of latitudes are finished. Then the next group of latitudes is solved in a similar manner until all rows are solved.

Using ATExpert on GFDL's CRAY C90 with fortran 77, MOM 2 with 1° horizontal resolution and 15 vertical levels (TEST CASE 0) was estimated to have 80% parallel efficiency. This means that on an 8 processor system, only about 3 processors will be saturated. Equivalently, about 3 such jobs would be required to saturate the system. On a 32 processor system, about 4.4 processors would be occupied which implies about 8 such jobs are needed to saturate the system. This is clearly not desirable in a dedicated environment or one with hundreds of processors. However, it may be tolerable in a multiprogramming environment where the number of jobs significantly exceeds the number of processors. The important consideration is that the job mix should not dry up to the point of leaving fewer jobs than processors.

In general, the fine grained approach has lower parallel efficiency than the coarse grained approach. The reason is basically due to not enough work within the many parallel regions and the fact that the range on the parallel loop index is not the same for all loops. Increasing the model resolution can somewhat address the first problem but not the second. The result is that processors occasionally stand idle. As compilers get smarter, and are better able to "fuse" parallel regions, the efficiency of the fine grained approach can be expected to increase. However, it is difficult to imagine that efficiencies will approach the coarse grained parallelism approach.

To illustrate the problem of too little work, consider a memory window of size $jmw = 5$. Calculations would normally be thought of as involving a latitude loop index ranging from $j = jsmw, jemw$ (where jsmw=2 and jemw=4) and so the appropriate number of processors would be *m=3*. However, the latitude loop index for calculating meridional fluxes would range from $j = jsmw - 1, jemw$ because meridional fluxes are needed on the northern and southern faces of all cells on latitude rows 2 through 4. This means that all processors will first busy themselves by calculating meridional fluxes for the northern faces of cells on rows 1 through 3. When done, one processor calculates the flux for the northern face of cells on row 4 and the other two processors stand idle because they do not advance to the next triplet until all are synchronized at the bottom of the loop. Since all parallel loops do not have the same range of indices, there must necessarily be idle processors *regardless of the amount of work inside the loops.*

[19] Or equivalently the number of times the memory window is moved northward to solve for all latitudes.

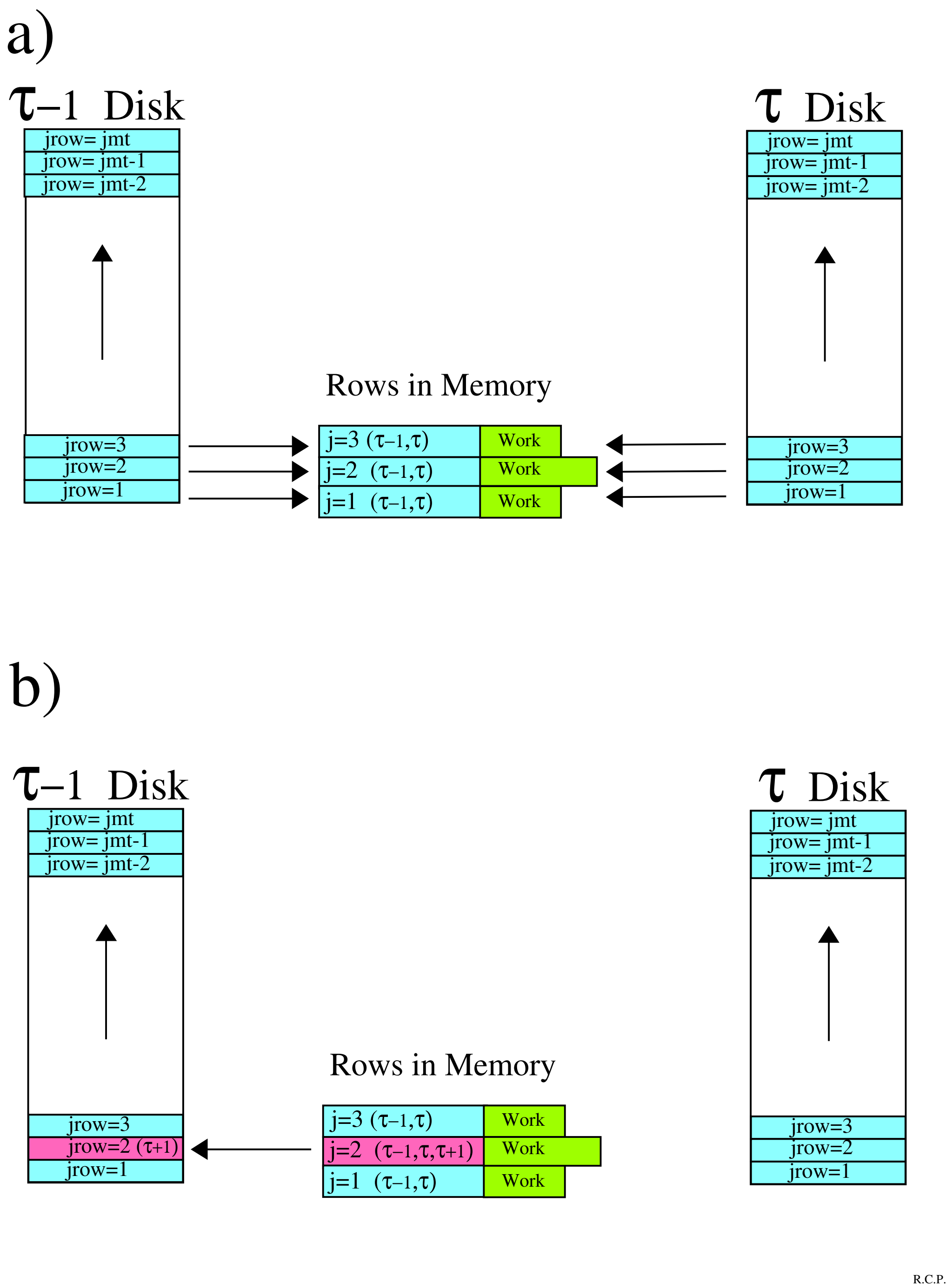


Figure 3.2: a) Loading the memory from disk. b) updating central row and writing results back to disk.

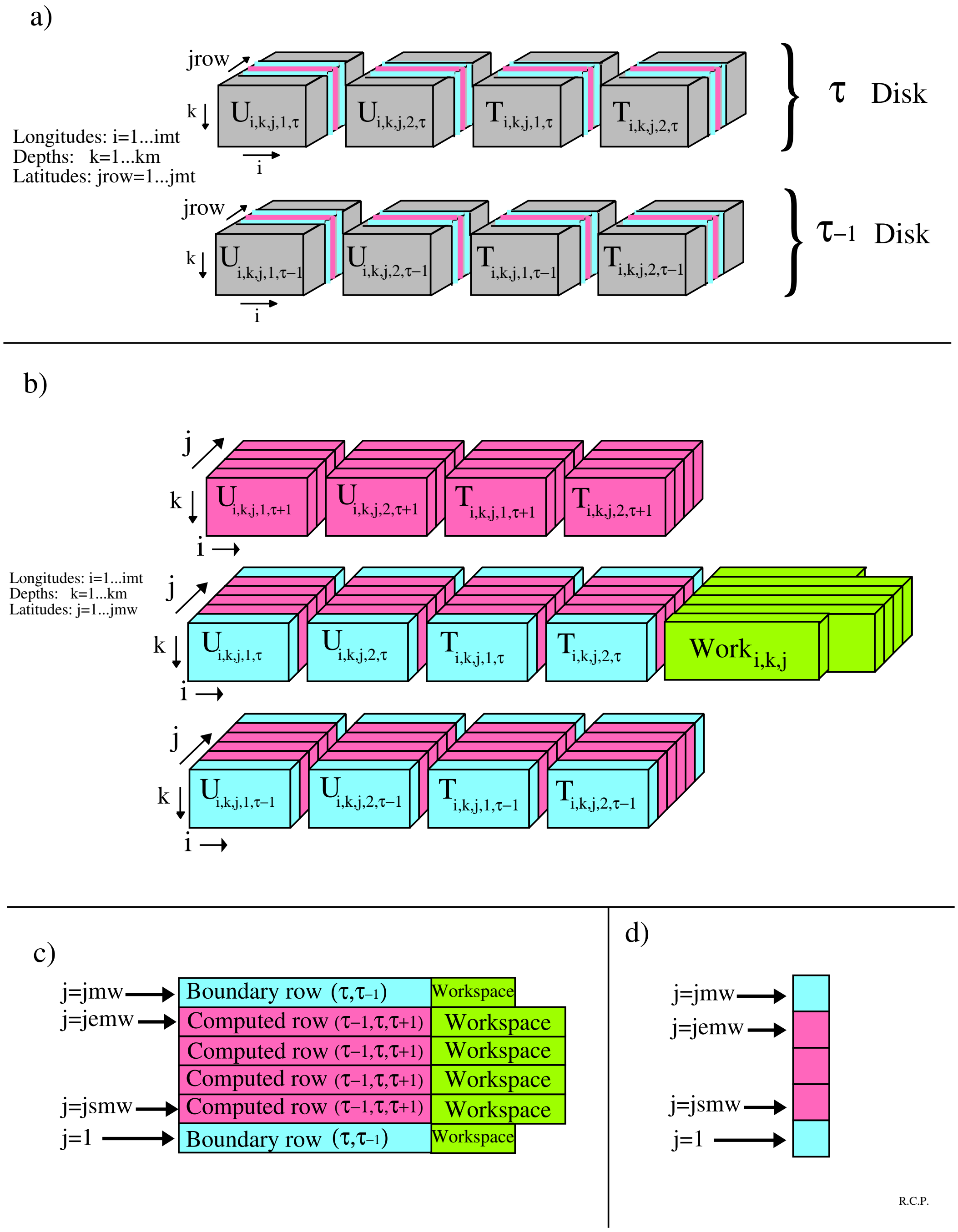


Figure 3.3: a)A slice through the volume of three dimensional prognostic variables on disk. Note that only two tracers ($nt = 2$) are assumed. For $nt > 2$, a new box is added for each tracer. b) Schematic of a memory window of size $jmw = 6$ holding the slice. c) Two dimensional schematic of the memory window d) One dimensional schematic of the memory window.

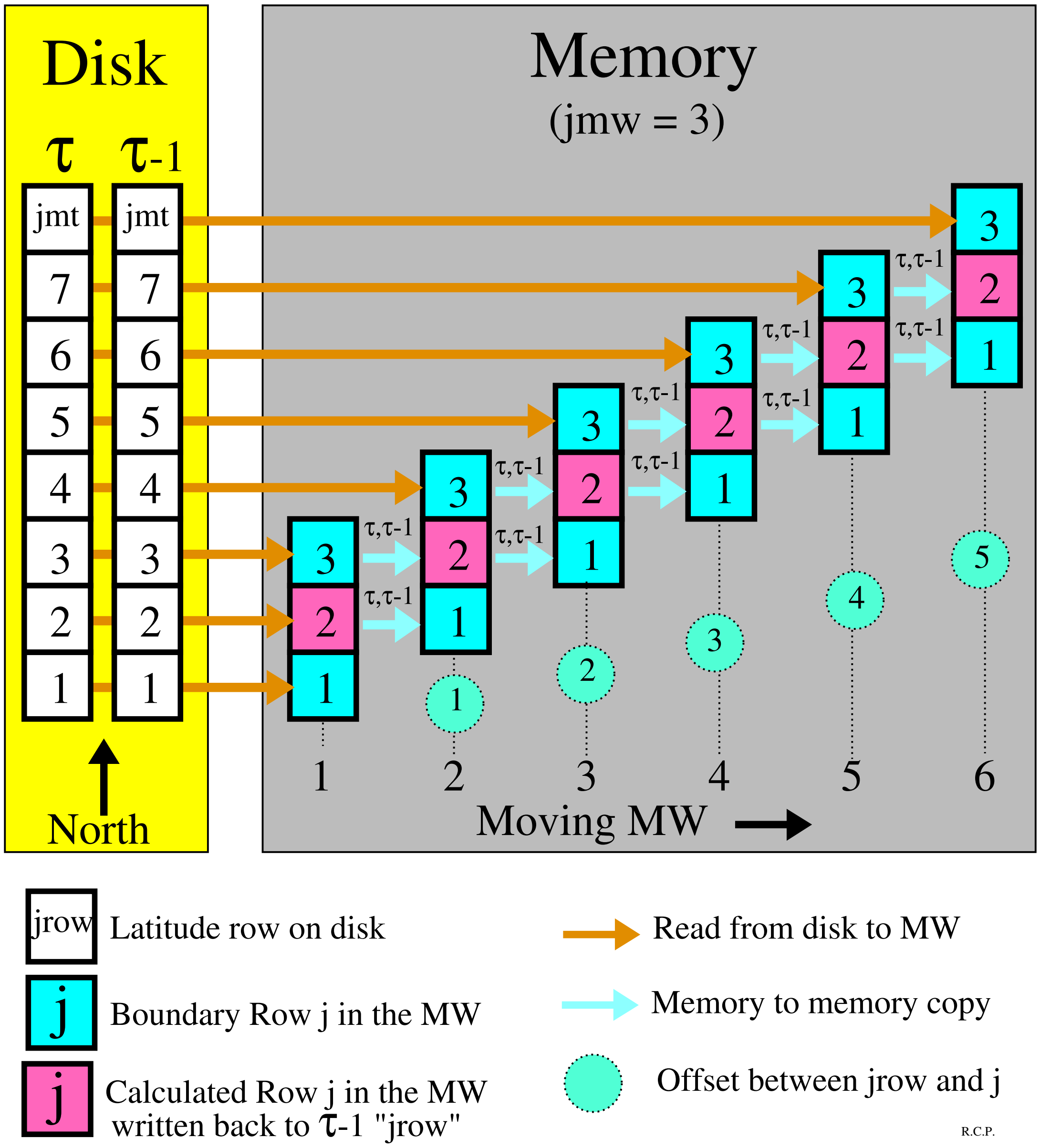


Figure 3.4: Schematic of dataflow between disk and memory for one timestep with a memory window size of $jmw = 3$ in MOM 2

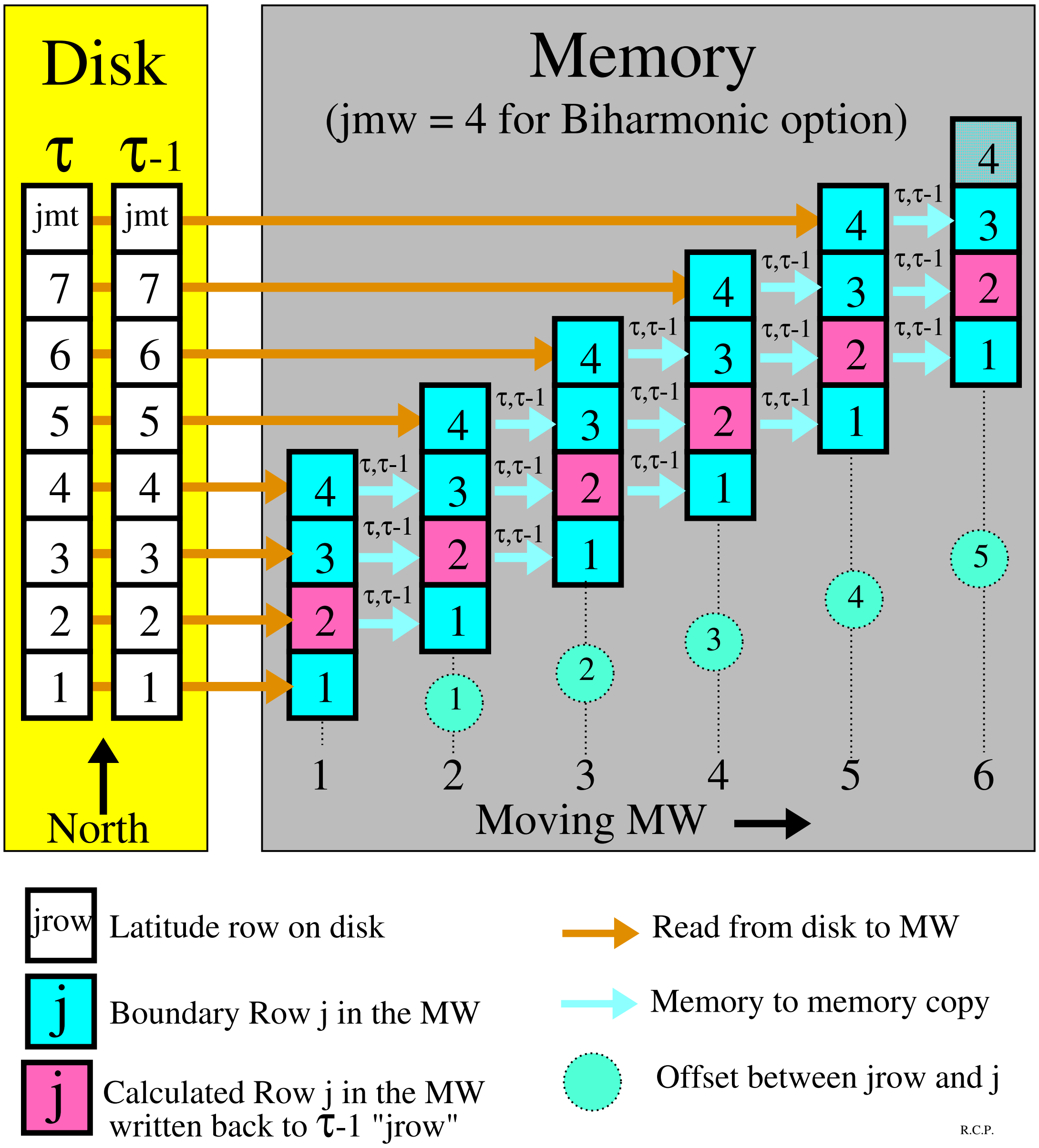


Figure 3.5: Example of dataflow between disk and memory for one timestep with a memory window size of $jmw = 4$ and the biharmonic option in MOM 2

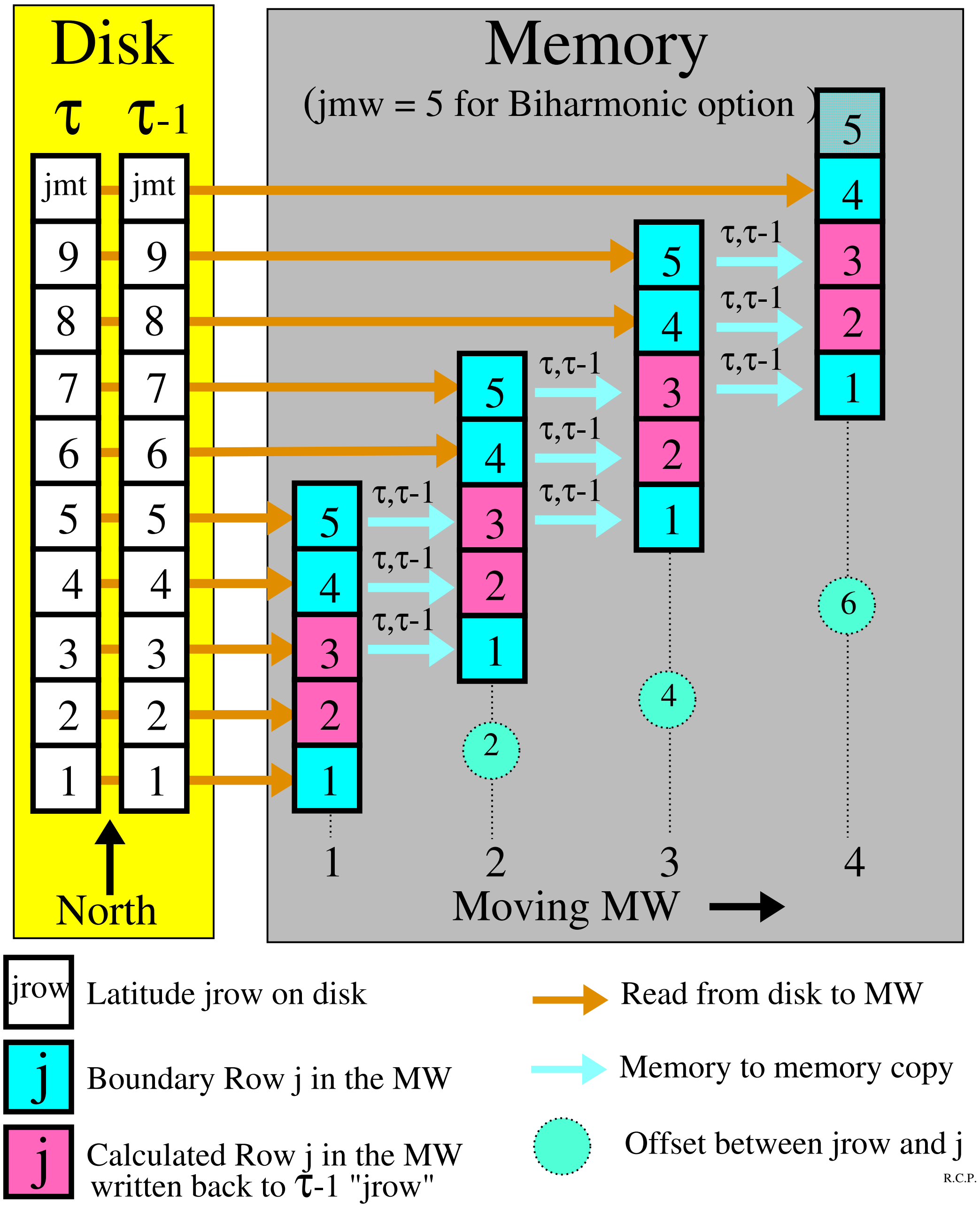


Figure 3.6: Example of dataflow between disk and memory for one timestep with a memory window size of $jmw = 5$ and the biharmonic option in MOM 2

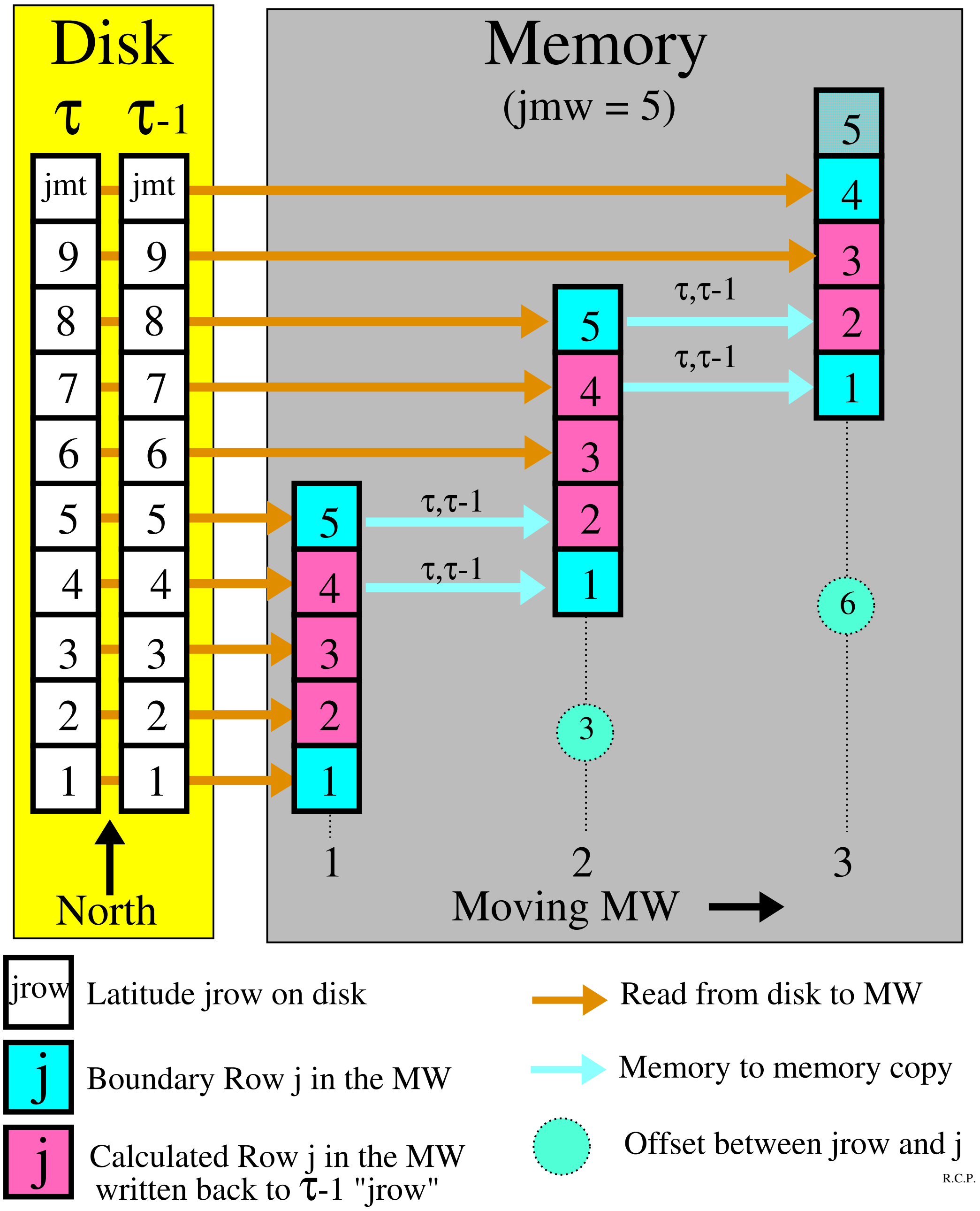


Figure 3.7: Schematic of dataflow between disk and memory for one timestep with a memory window size of $jmw = 5$ in MOM 2

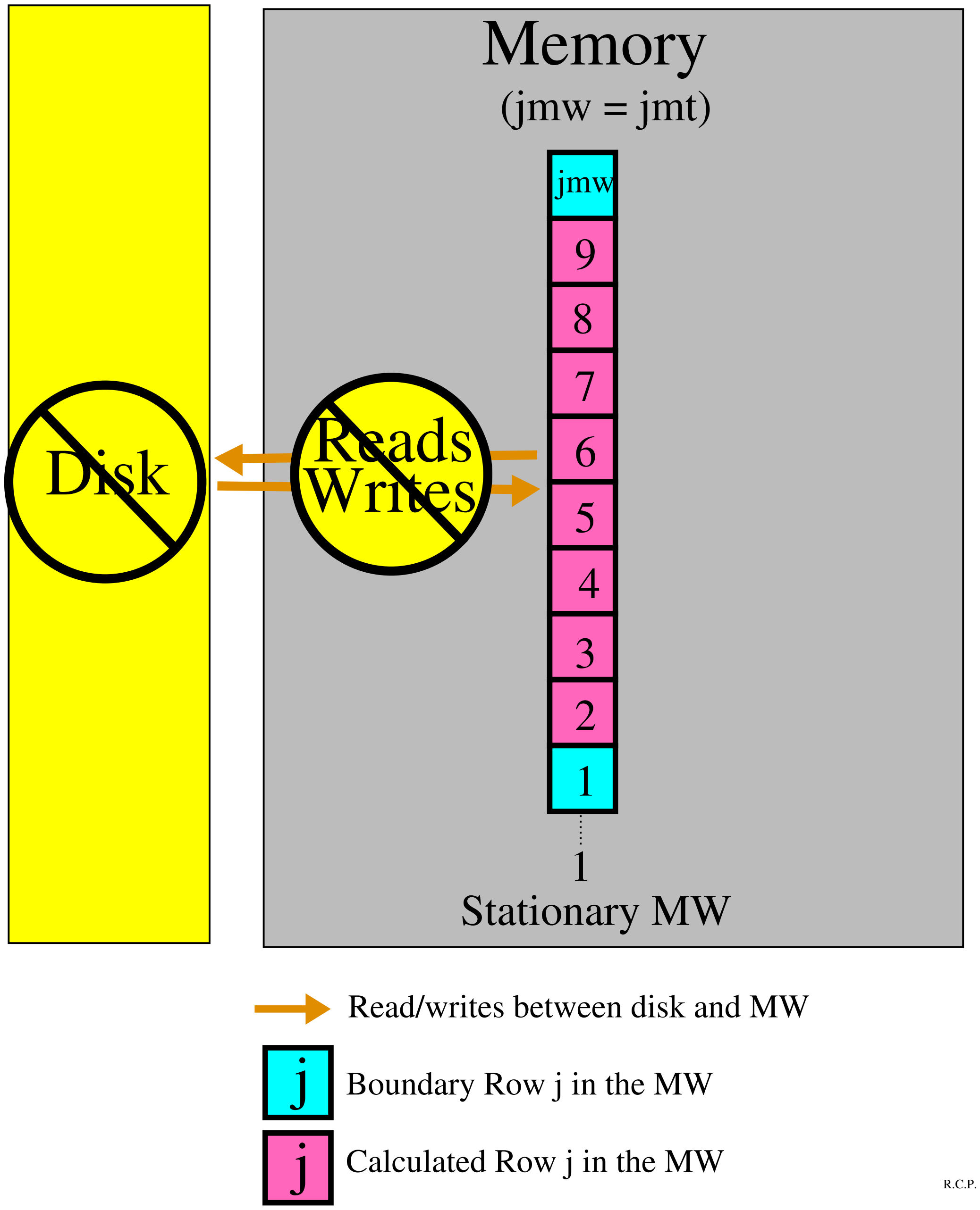


Figure 3.8: Schematic indicating that disk space, reads/writes, and memory to memory copies are not needed when the memory window size is $jmw = jmt$ in MOM 2

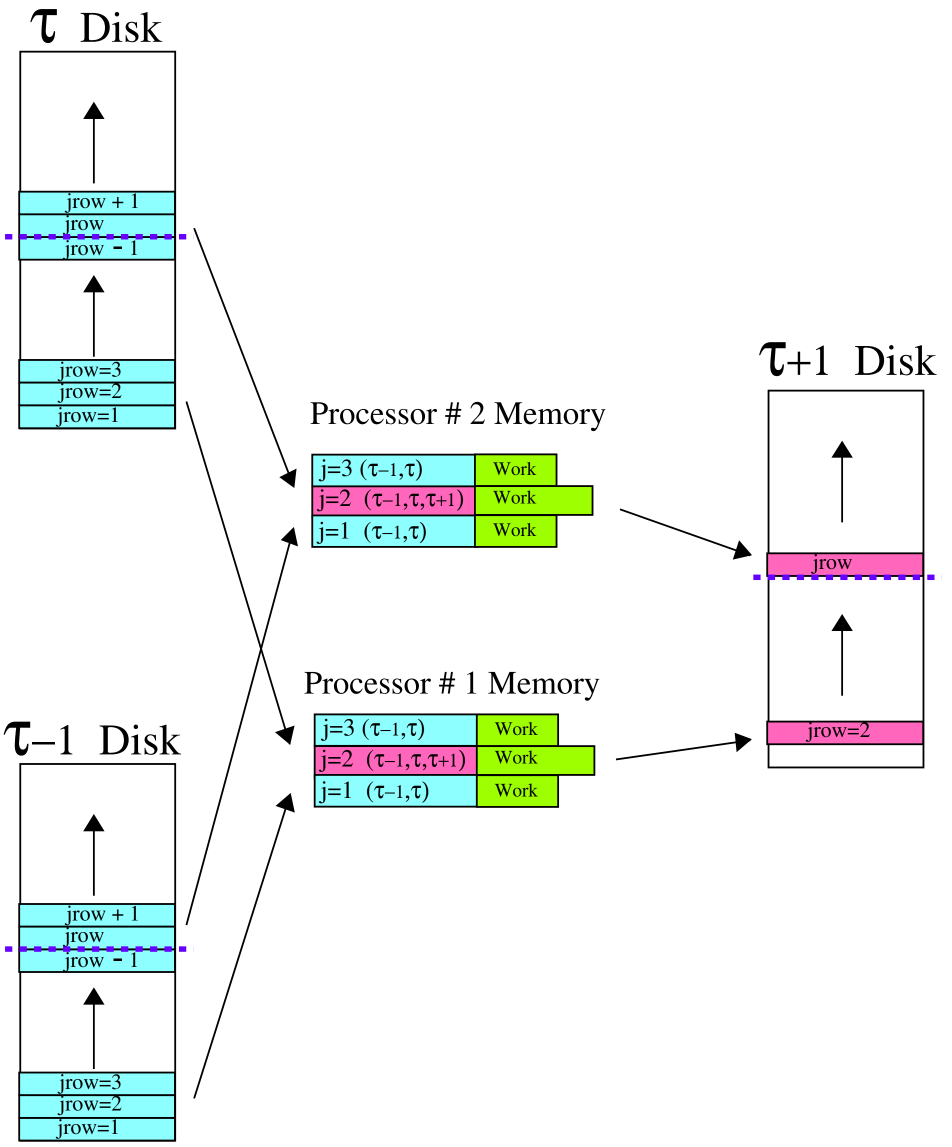
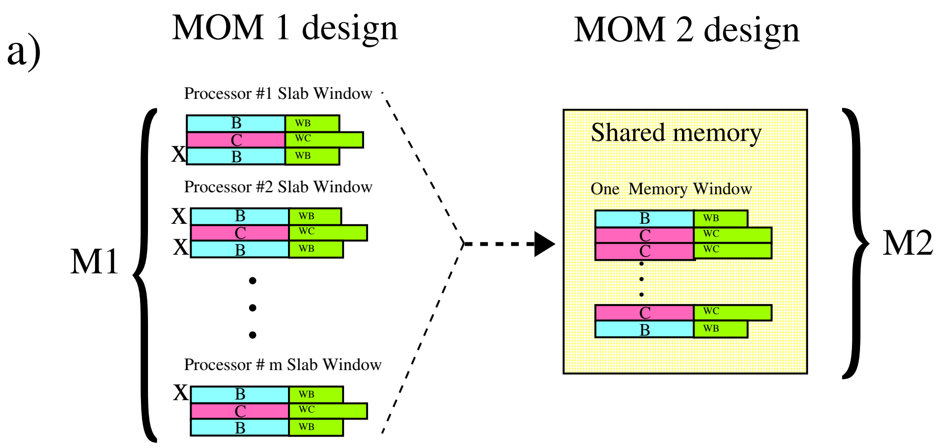


Figure 3.9: Loading the memory from disk, updating the central row and writing results back to disk. Note that disks are logically divided into two sections.



b)

$$\frac{M2}{M1} = \frac{2(1+\alpha\beta) + m(1+\beta)}{m[2(1+\alpha\beta) + (1+\beta)]} \quad \text{where } \alpha = WB/WC \text{ and } \beta = WC/C$$

M2 / M1 (α= 1/3)

"m" Processors ↓ \ β →	.75	1.0	1.25	1.5
1	1	1	1	1
2	.71	.71	.72	.73
3	.61	.62	.63	.64
4	.56	.57	.58	.59
5	.53	.54	.55	.56
10	.47	.49	.5	.51
100	.42	.43	.45	.46

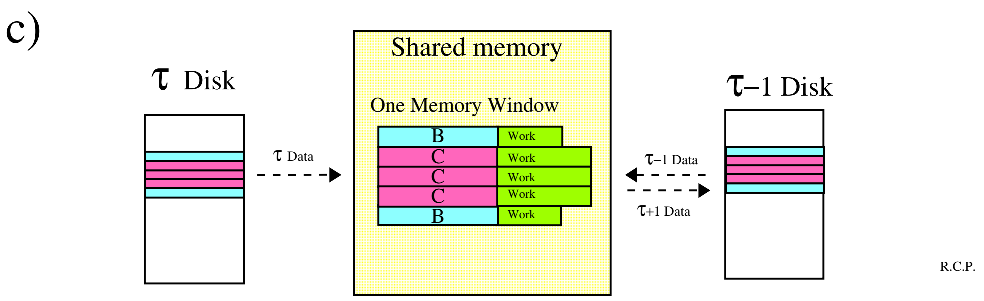


Figure 3.10: a) Redundant rows in coarse grained windows b) Comparing memory requirements between fine and coarse grained approaches c) Dataflow for fine grained approach

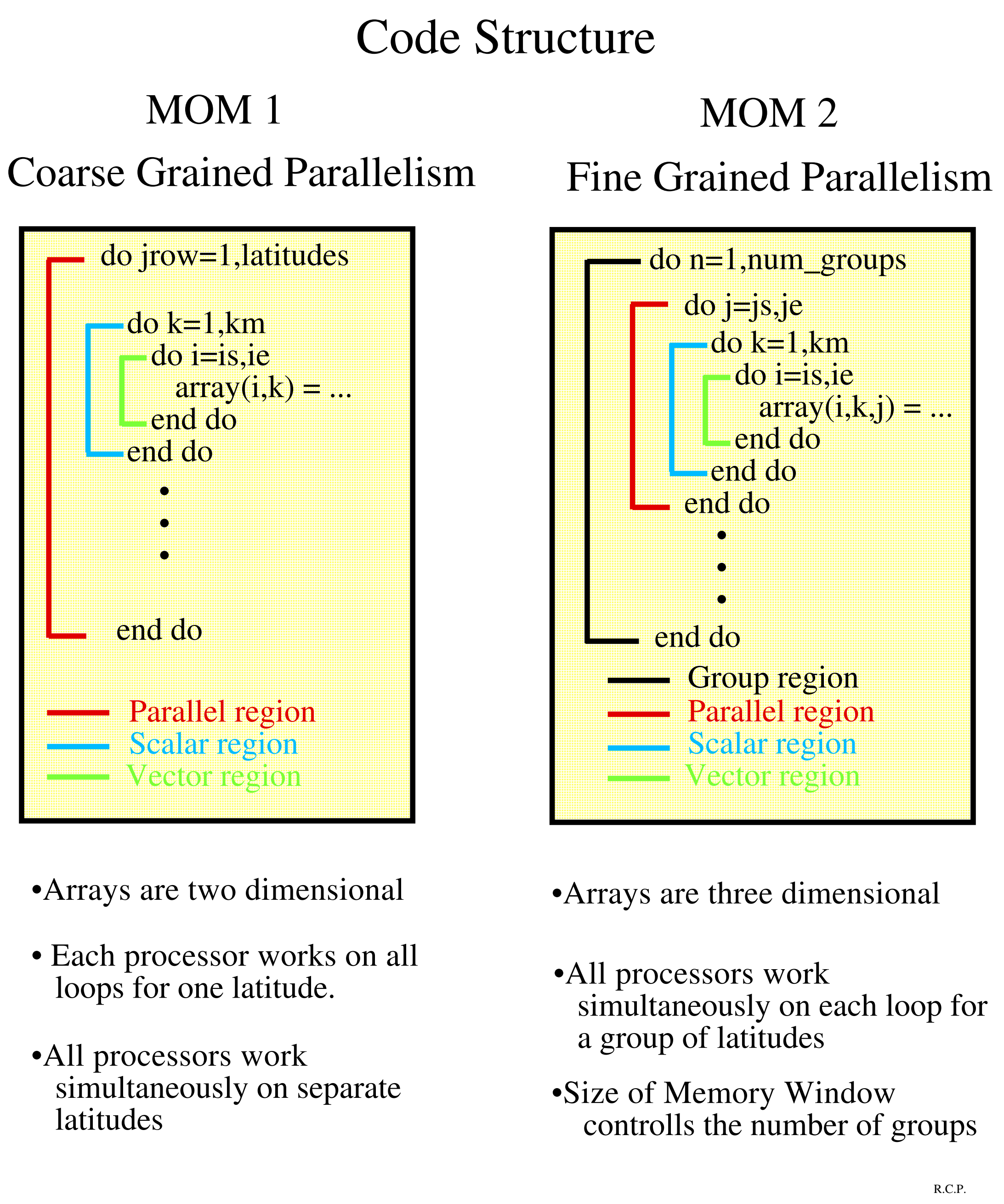


Figure 3.11: Differences in code structure between MOM 1 and MOM 2

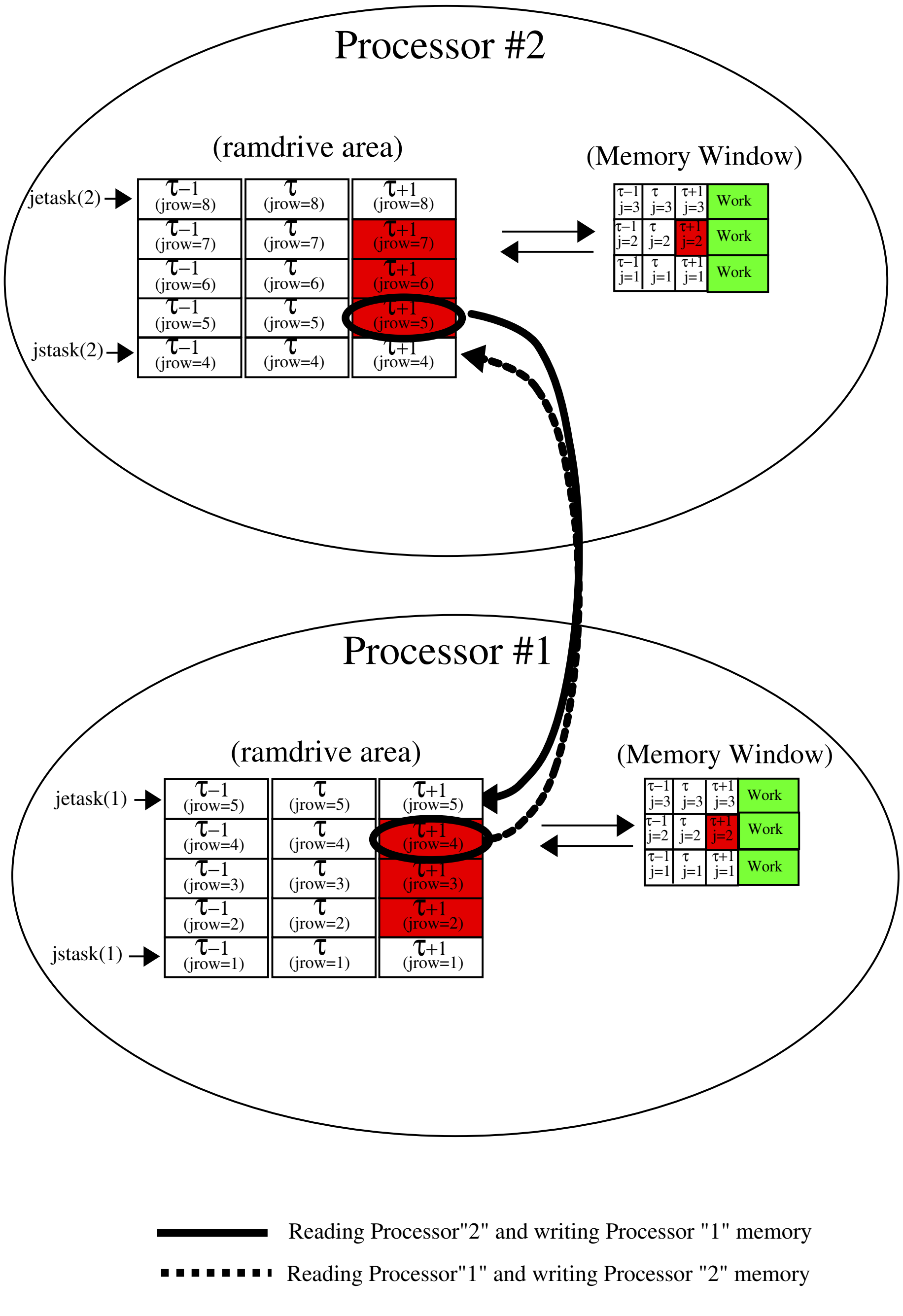


Figure 3.12: Inter-processor communication required for a second order memory window.

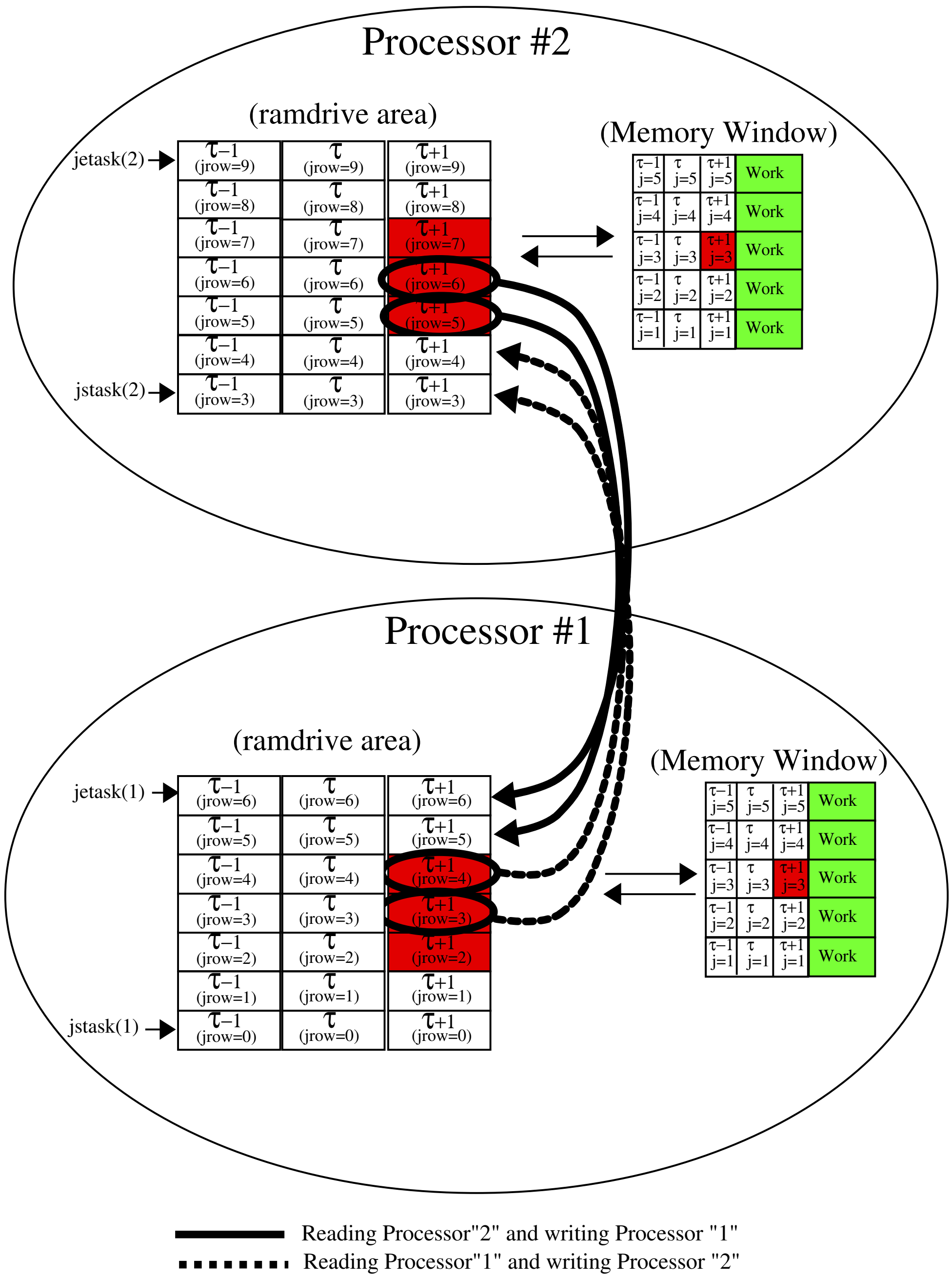


Figure 3.13: Inter-processor communication required for a fourth order memory window.

Chapter 4

Database

A database is included with MOM 2. The database is not needed to start using MOM 2, although use of MOM 2 is certainly limited without it. Refer to Chapter 19 for details on how to get this data and Section 19.6 for a description of programs which access it. This database is far from being complete. It should be considered as a starting point and act as a prototype for extensions using other datasets. GFDL is not a data center and there is no intention to extend this with other datasets.

4.1 Data files

The DATABASE contains approximately 160MB of IEEE 32bit data and the files are named as follows:

- *scripps.top* is the Scripps topography (265816 bytes).
- *hellerman.tau* is the Hellerman/Rosenstein (1983) windstress climatology (1757496 bytes).
- *oorts.air* is the Oort (1983) 1000mb air temperature climatology (294268 bytes).
- *levitus.mask* is the Levitus (1982) land/sea mask (8680716 bytes).
- *ann.temp* is the Levitus (1982) annual mean potential temperatures (8682036 bytes).
- *ann.salt* is the Levitus (1982) annual mean salinities (8682036 bytes).

  Then there are the Levitus (1982) monthly climatological potential temperatures. There are twelve monthly files where the three letter suffix gives the month.
- *jan.temp* is the Levitus (1982) monthly mean potential temperatures (4998748 bytes).

  The remaining eleven monthly temperature files are of the same size but use a different month as the suffix.

  Then there are the Levitus (1982) monthly climatological salinities. There are twelve monthly files where the three letter suffix gives the month.
- *jan.salt* is the Levitus (1982) monthly mean salinities (4998748 bytes).

  The remaining eleven monthly salinity files are of the same size but use a different month as the suffix.

Note that the file size for the monthly Levitus (1982) data is less than the file size for the annual means. This is because seasonal effects only penetrate down to 1000m or so and the monthly data only extends to this depth. Also, salinity has been interpolated from four seasonal values to twelve monthly values within this dataset. All Levitus (1982) data as well as Hellerman/Rosenstein (1983) and Oort (1983) data has been forced into land areas on their native grid resolutions by solving

$$\nabla^2 \xi_{i,jrow} = 0 \tag{4.1}$$

over land areas using values of $\xi_{i,jrow}$ over ocean areas as boundaries values where $\xi_{i,jrow}$ represents the various types of data. This method is similar to what was done with surface boundary conditions in Section 10.1.2.

Chapter 5

Variables

5.1 Naming convention for variables

Modern compilers no longer restrict variable names to 6 or fewer characters. This 6 character restriction is also relaxed in MOM 2 although economy of characters in choosing names is always desirable. In an attempt to unify naming of variables, a convention is introduced which uses abbreviated forms of the fundamental conceptual pieces of the numerical oceanography. This convention also gives guidance for building the names for variables in future parameterizations and keeps an overall consistency throughout the model. Constructing names for variables is then a matter of assembling these abbreviations in various ways. Note that when names of variables are spelled out, the naming convention should not be applied. However, when names seem cryptic, the convention is intended to help decipher their meaning. A list of abbreviations is given below:

u =	Velocity point	d =	Delta	adv =	Advective
t =	Tracer point	f =	Flux	diff =	Diffusive
n =	North face	p =	Pressure	visc =	Viscous
e =	East face	c =	Coefficient	psi =	Stream function
b =	Bottom face	v =	Velocity	ps =	Surface pressure

Variables described subsequently within this chapter are build in terms of these abbreviations. All variables are positioned relative to the arrangement of grid cells discussed in Chapter 7. When an unknown variable is encountered in MOM 2, it may be possible to determine its meaning from context or the naming convention. For example, the variable *adv_vnt* is an advective velocity on the north face of T cells. If this fails, it may be useful to search all *include files* for comments using the UNIX command: *grep* as described in Section 19.5.

Operators are used in MOM 2 to construct various terms in finite difference equations which are the counterpart to equations detailed in Chapter 2. Unlike arrays, operators are build using expressions which relate variables but require no explicit memory. The details of a particular operation are forced into the operator and buried from view which allows complicated equations to be written in terms of simple, easy to understand, pieces. At the same time, compilers can expand these pieces in-line into an un-intelligible mess (by human standards) and subsequently optimize it. In fact, there is no difference in speed on the CRAY (or any mature compiler) between writing an equation with operators and writing an equation with all terms expanded out. The big difference is that the former is easier to understand and work with for humans.

Instead of using operators, all terms in the equations could be computed once and stored in arrays. This would allow less redundant calculation but be prohibitive in memory usage. It is an interesting counter intuitive point that it is sometimes faster to do redundant calculation rather than save intermediate results in arrays. Depending on the speed of the load/store operations compared with the multiply/add operations, if redundant computation contains little work it may be faster to compute redundantly. In practice, this depends on the computer and compiler optimizations. This can be verified by executing script *run_timer* which exercises the timing utilities in module *timer* by solving a tracer equation in various ways. It is one measure of a mature compiler when the speed differences implied by solving the equations in various ways is relatively small. On the CRAY YMP the differences are about 10% whereas on the SGI workstation, they can exceed 50%.

Within operators, capital letters are reserved for the intended operation and small letters act as qualifiers. A list of abbreviations is given below:

ADV_T = Advection of tracer	x = longitude direction
DIFF_T = Diffusion of tracer	y = latitude direction
ADV_U = Advection of velocity	z = vertical direction
DIFF_U = Diffusion of velocity	

As an example, tracer diffusion in the longitude direction at a given point is given by operator DIFF_Tx(i,k,j). These operators are defined in include files and may be located using the UNIX command *grep* as indicated in Section 19.5.

5.2 The main variables

5.2.1 Relating j and jrow

In MOM 2, variables dimensioned by the number of rows in the memory window "jmw" are referenced with index *j*. So $A(i,k,j)$ would refer to an array dimensioned as $A(imt,km,jmw)$ where the allowable range for index *j* is from 1 through jmw. Variables dimensioned by the total number of latitude rows in the grid *jmt* are referenced with index *jrow*. So $B(i,jrow)$ would refer to an array dimensioned as $B(imt,jmt)$ where the allowable range for index *jrow* is from 1 through jmt. In this manual, $A(i,k,j)$ is written as $A_{i,k,j}$ and $B(i,jrow)$ is written as $B_{i,jrow}$ to save space and make the equations more readable. In general, *jrow = j + joff* where offset *joff* indicates how far the memory window has moved northward. Refer to Figure 3.4 and Section 3.3.2 for a description of how this works. In MOM 2, whether an array is indexed by *j* or by *jrow* is dictated entirely by whether it is dimensioned by *jmw* or *jmt*.

5.2.2 Cell faces

As described below, in addition to defining prognostic variables at grid points within T cells and U cells, it is useful to define other variables on the faces of these cells. This is because the equations are written in finite difference flux form to allow quantities to be conserved numerically. Each cell has six faces and certain variables must be defined on all faces. For instance, advective velocities (zonal, meridional, and vertical) are located on cell faces defined normal to those faces and positive in a direction of increasing longitude, latitude and decreasing depth[1]. Additional variables defined on cell faces are diffusive coefficients, viscous coefficients, diffusive fluxes, and advective fluxes. Note that names for variables on cell faces are only needed on the east, north, and bottom faces of cells. This is because the west, south, and top faces of

[1] Positive vertical velocity at the bottom of a cell points upward in the positive z direction

cell $T_{i,k,jrow}$ are the east face of cell $T_{i-1,k,jrow}$, the north face of cell $T_{i,k,jrow-1}$, and bottom face of cell $T_{i,k-1,jrow}$. A similar arrangement holds for U cells.

5.2.3 Model size parameters

The main dimensional parameters are:

- *imt* = number of longitudinal grid cells in the model domain. This is output by module *grids* when specifying a domain and resolution.
- *jmt* = number of latitudinal grid cells in the model domain. This is output by module *grids* when specifying a domain and resolution.
- *km* = number of grid cells between the ocean surface and the bottom in the model domain. This is output by module *grids* when specifying a domain and resolution.
- *jmw* = number of latitude rows within the memory window. The minimum is three and the maximum is jmt.
- *nt* = number of tracers in the model. Usually set to two. The first is for potential temperature and the second is for salinity as described in Section 5.2.8. Additional tracers (when $nt > 2$) are passive and initialized to 1.0 at level $k = 1$ amd 0.0 for levels $k > 1$.

Some secondary ones are:

- *mnisle* = maximum number of unconnected land masses (islands).
- *maxipp* = maximum number of island perimeter (coastal) points. This includes coastlines from all land masses.

5.2.4 T cells

A complete description of grid cells is given in Chapter 7. For cells on the T grid given by $T_{i,k,jrow}$, the primary grid distances in longitude and latitude are given by:

- xt_i = longitudinal coordinate of grid point within cell $T_{i,k,jrow}$ in degrees. Index i increases eastward with increasing longitude.
- yt_{jrow} = latitudinal coordinate of grid point within cell $T_{i,k,jrow}$ in degrees. This is written in the equations of this manual as ϕ^T_{jrow}. Index *jrow* increases northward with increasing latitude.
- dxt_i = longitudinal width of equatorial cell $T_{i,k,jrow}$ in cm.
- dyt_{jrow} = latitudinal width of cell $T_{i,k,jrow}$ in cm.
- cst_{jrow} = cosine of latitude at the grid point within cell $T_{i,k,jrow}$. This is written in the equations of this manual as $\cos\phi^T_{jrow}$.

5.2.5 U cells

A complete description of grid cells is given in Chapter 7. For cells on the U grid given by $U_{i,k,jrow}$, the primary grid distances in longitude and latitude are given by:

- xu_i = longitudinal coordinate of grid point within cell $U_{i,k,jrow}$ in degrees. Index i increases eastward with increasing longitude.
- yu_{jrow} = latitudinal coordinate of grid point within cell $U_{i,k,jrow}$ in degrees. This is written in the equations of this manual as ϕ^U_{jrow}. Index *jrow* increases northward with increasing latitude.
- dxu_i = longitudinal width of equatorial cell $U_{i,k,jrow}$ in cm.
- dyu_{jrow} = latitudinal width of cell $U_{i,k,jrow}$ in cm.
- csu_{jrow} = cosine of latitude at the grid point within cell $U_{i,k,jrow}$. This is written in the equations of this manual as $\cos\phi^U_{jrow}$.

5.2.6 Vertical spacing

A complete description of grid cells is given in Chapter 7. For T cells and U cells, the primary grid distances in the vertical are given by:

- zt_k = distance from the surface to the grid point in cell $T_{i,k,jrow}$ or $U_{i,k,jrow}$ in units of cm. The T cells and U cells are not staggered vertically. Note that although zt_k is positive downwards, the coordinate z is positive upwards. Index k increases downward.
- zw_k = distance from the surface to the bottom face of cell $T_{i,k,jrow}$ or $U_{i,k,jrow}$ in cm. The T cells and U cells are not staggered vertically.
- dzt_k = vertical thickness of cell $T_{i,k,jrow}$ or $U_{i,k,jrow}$ in cm.
- dzw_k = vertical distance between the grid point within cell $T_{i,k,jrow}$ and the grid point within cell $T_{i,k+1,jrow}$ in cm. It's the same for the U cells.

5.2.7 Time level indices

As explained in Section 11.1, indices are used to point to various time levels on disk and in the memory window. Note that all of these indices are integers.
The indices which point to time levels within the memory window are:

- *taum1* which points to data at time level $\tau - 1$.
- *tau* which points to data at time level τ.
- *taup1* which points to data at time level $\tau + 1$.

The indices which point to time levels on disk are:

- *taum1disk* which points to data at time level $\tau - 1$.
- *taudisk* which points to data at time level τ.
- *taup1disk* which points to data at time level $\tau + 1$.

Note that typically there are only two disks needed (unless multitasking with the coarse grained parallel approach). In this typical case, the “taup1disk” takes on the value of “taum1disk”.

5.2.8 3-D Prognostic variables

The three dimensional[2] prognostic variables are:

- $u_{i,k,j,n,\tau}$ = horizontal velocity components located at grid points within cell $U_{i,k,jrow}$
- $t_{i,k,j,n,\tau}$ = tracer components located at grid points within cell $T_{i,k,jrow}$

Here, subscript i is the longitude index of the cell from $i = 1$ (westernmost cell) to $i = imt$ (easternmost cell); subscript k is the depth index of the cells from $k = 1$ (surface cell) to $k = km$ (bottom cell); and subscript j is the latitude index of the cells from $j = 1$ (southernmost cell) to $j = jmw$ (northernmost cell)[3]. With regard to U cells, subscript n is the velocity component[4] in units of cm/sec. With regard to T cells, subscript n refers to a particular tracer [5]. Subscript τ in both cases refers to the discrete time level.

5.2.9 2-D Prognostic variables

The two dimensional prognostic quantities are:

- $psi_{i,jrow,\tau}$ = stream function defined at $i, jrow$ locations on the T grid in units of cm^3/sec, where $10^{12}cm^3/sec$ is one sverdrup. $psi_{i,jrow,\tau}$ is written as $\psi_{i,jrow,\tau}$ in this manual.
- $ps_{i,jrow,\tau}$ = prognostic surface pressure defined at $i, jrow$ locations on the T grid in units of $gram/cm/sec^2$. This variable is also used as the implicit free surface (in terms of pressure) and the units are the same.

5.2.10 3-D Workspace variables

These variables are determined diagnostically in the sense that they are computed from other prognostic quantities and are therefore not indexed by time level. The three dimensional diagnostic and workspace variables are listed below:

- Six variables are needed for advective velocities on T cells and U cells. Units are cm/sec

  1. $adv\_vet_{i,k,j}$ = advective velocity on the east face of cell $T_{i,k,jrow}$
  2. $adv\_vnt_{i,k,j}$ = advective velocity on the north face of cell $T_{i,k,jrow}$
  3. $adv\_vbt_{i,k,j}$ = advective velocity on the bottom face of cell $T_{i,k,jrow}$
  4. $adv\_veu_{i,k,j}$ = advective velocity on the east face of cell $U_{i,k,jrow}$
  5. $adv\_vnu_{i,k,j}$ = advective velocity on the north face of cell $U_{i,k,jrow}$
  6. $adv\_vbu_{i,k,j}$ = advective velocity on the bottom face of cell $U_{i,k,jrow}$

---

[2] Referring to three spatial dimensions.

[3] The j index need only be dimensioned for the size of the memory window. Refer to Chapter 1 for a description of the memory window and Section 5.2 for a description of $jrow$.

[4] n=1 is the zonal and n=2 is the meridional velocity component. Vertical velocity is not a prognostic variable. It is a diagnostic variable defined as an advective velocity at the bottom face of cells. Note that since there are T cells and U cells, there are vertical velocities associated with each. For diagnostic purposes, the vertical velocity on the bottom of T cells is output.

[5] n=1 is for potential temperature in units of *degrees C*, n=2 is for salinity in units of deviations from 0.035 grams of salt/cm^3 of water or parts per mill ppm -0.035. MOM 2 uses $\rho_\circ = 1.035\ gm/cm^3$ for the Boussinesq approximation. These units can be converted to parts per thousand (ppt) by adding 0.035 grams/cm^3 and multiplying by 1000. n > 2 is for additional passive tracers.

- Two variables are needed for density and hydrostatic pressure gradients

  1. $rho_{i,k,j}$ = density defined at T cell grid points. Units are in sigma units gm/cm^3 and represent departures from reference densities specific to each model level as described in Section 6.2.2. In this manual, $rho_{i,k,j}$ is written as $\rho_{i,k,j}$.
  2. $grad\_p_{i,k,j,n}$ = hydrostatic pressure gradient. n=1 is for the zonal component and n=2 is for the meridional component. Units are in cm/sec^2 assuming $\rho_\circ$ = 1.035 gm/cm^3.

- Six variables are needed for fluxes through the faces of T cells and U cells. Note that there is no explicit T or U cell suffix on these names because they are re-calculated for each prognostic variable in T cells and U cells. Note also that since these are re-calculated for each prognostic variable, they are <u>not</u> candidates for being moved as the memory window moves northward. Units are in *cgs* and specific to the quantity being advected or diffused.

  1. $adv\_fe_{i,k,j}$ = advective flux on the east face of a cell
  2. $adv\_fn_{i,k,j}$ = advective flux on the north face of a cell
  3. $adv\_fb_{i,k,j}$ = advective flux on the bottom face of a cell
  4. $diff\_fe_{i,k,j}$ = diffusive flux on the east face of a cell
  5. $diff\_fn_{i,k,j}$ = diffusive flux on the north face of a cell
  6. $diff\_fb_{i,k,j}$ = diffusive flux on the bottom face of a cell

- There are three coefficients for diffusion of tracers across T cells and another three for diffusion (viscous transfer of momentum) across U cells. They may or may not be triply subscripted. It depends on the physics parameterization. For instance, options *consthmix* and*constvmix* use coefficients which are constant throughout the grid and therefore require no subscripts. Units are cm^2/sec.

  1. $diff\_cet_{i,k,j}$ = diffusive coefficient on the east face of cell $T_{i,k,jrow}$
  2. $diff\_cnt_{i,k,j}$ = diffusive coefficient on the north face of cell $T_{i,k,jrow}$
  3. $diff\_cbt_{i,k,j}$ = diffusive coefficient on the bottom face of cell $T_{i,k,jrow}$
  4. $visc\_ceu_{i,k,j}$ = viscous coefficient on the east face of cell $U_{i,k,jrow}$
  5. $visc\_cnu_{i,k,j}$ = viscous coefficient on the north face of cell $U_{i,k,jrow}$
  6. $visc\_cbu_{i,k,j}$ = viscous coefficient on the bottom face of cell $U_{i,k,jrow}$

Additionally, other variables may be needed but these are dependent on enabled options and will not be listed here.

5.2.11 3-D Masks

- To promote vectorization, two fields are used to differentiate ocean and land cells.

  1. $tmask_{i,k,j}$ = mask for $T_{i,k,jrow}$. 1.0 for ocean, 0.0 for land
  2. $umask_{i,k,j}$ = mask for $U_{i,k,jrow}$. 1.0 for ocean, 0.0 for land

5.2.12 Surface Boundary Condition variables

The surface boundary condition quantities are contained in three arrays. Note that one is dimensioned by the total number of latitude rows (jmt) while two are dimensioned by the size of the memory window (jmw).

- $sbcocn_{i,jrow,m}$= surface boundary conditions which are dimensioned as *(imt,jmt,numsbc)* where *numsbc* is the number of surface boundary conditions. Refer to Section 10.3 for usage details.

- $smf_{i,j,n}$ = components of the surface momentum flux defined on the U grid for the latitude rows within the memory window. $n = 1$ is for zonal momentum flux (windstress in the zonal direction) and $n = 2$ is for meridional momentum flux (windstress in the meridional direction). This field is dimensioned by the memory window size as *smf(imt,jmw,2)*. Units are in $dynes/cm^2$.

- $stf_{i,j,n}$ = components of the surface tracer flux defined on the T grid for the latitude rows within the memory window. $n = 1$ is for heat flux in units of *langley/sec* where 1 *langley* = 1 cal/cm^2, $n = 2$ is for salt flux in units of $grams/cm^2/sec$, and $n > 2$ is for additional tracers. This field is dimensioned by the memory window size as *stf(imt,jmw,nt)*.

5.2.13 2-D Workspace variables

These two dimensional quantities are functions of geometry and topography. Being independent of time, they are computed only once and are dimensioned by (imt, jmt):

- $kmt_{i,jrow}$= number of T cells between the surface and bottom of the ocean defined at $i, jrow$ locations on the T grid. It is constructed by module *topog*.

- $kmu_{i,jrow}$= number of T cells between the surface and bottom of the ocean defined at $i, jrow$ locations on the U grid. It is given by Equation (9.1).

- $mskhr_{i,jrow}$= horizontal regional mask number used for certain diagnostics as described in Section 18.1.4.

- $mskvr_k$= vertical regional mask number used for certain diagnostics as described in Section 18.1.4. This is only a function of k but is included here because of its close association with $mskhr_{i,jrow}$.

- $h_{i,jrow}$ = depth from the surface to the bottom of the ocean defined at $i, jrow$ locations on the U grid in units of *cm*. In this manual, $h_{i,jrow}$ is written as $H_{i,jrow}$ and is computed by Equation (9.2).

- $hr_{i,jrow}$ = reciprocal of $h_{i,jrow}$ defined at $i, jrow$ locations on the U grid in units of $1/cm$. For masking purposes, $hr_{i,jrow} = 0$ whenever $h_{i,jrow} = 0$

The following two dimensional quantities are diagnosed from other prognostic quantities and are therefore recomputed every time step. They are dimensioned by (imt, jmt):

- $zu_{i,jrow,n}$ = vertical average of time derivatives of the momentum equation used as forcing for the external mode equations at $i, jrow$ locations on the U grid. Units are cm/sec^2.

- $ztd_{i,jrow}$ = curl of $zu_{i,jrow,n}$ defined at $i, jrow$ locations on the T grid. Units are $1/sec^2$.

- $res_{i,jrow}$ = residual from the elliptic solver defined at $i, jrow$ locations on the T grid. Units depend on whether streamfunction or rigid lid surface pressure or implicit free surface method is used. Refer to these variables for units.

- $ptd_{i,jrow}$ = time rate of change of stream function, rigid lid surface pressure, or implicit free surface defined at $i, jrow$ locations on the T grid. Refer to these variables for units.

- $cf_{i,jrow,-1:1,-1:1}$ = coefficient arrays for solving the external mode elliptic equation with 5 point or 9 point numerics. Units are $1/sec/cm^3$. Normally, this is independent of time except when solving the Coriolis part of the external mode implicitly. Since this computation is done very fast, it is computed every time step even when the Coriolis term is solved explicitly. The third and fourth indices are deviations about i and $jrow$ respectively. They give access to cell $(i, jrow)$ and all eight surrounding grid cells.

5.3 Operators

As outlined in Section 5.1, operators are used to construct various terms in the tracer and momentum equations. They are implemented as statement functions requiring no storage and their details are expanded out in Sections 11.11.5 and 11.10.7. It is important to note that for operators to work correctly, all items buried within the details of the operator must be defined correctly at the time when the operator is used. As with variables, operators also have a placement on the grid as explained below:

5.3.1 Tracer Operators

The details of the following operators used in solving the tracer equations are defined in file *fdift.h*.

- $ADV\_Tx_{i,k,j}$ = the flux form of the zonal (x) advection of tracer defined on the $T_{i,k,jrow}$ grid point. Units are in tracer units per second.

- $ADV\_Ty_{i,k,j}$ = the flux form of the meridional (y) advection of tracer defined on the $T_{i,k,jrow}$ grid point. Units are in tracer units per second.

- $ADV\_Tz_{i,k,j}$ = the flux form of the vertical (z) advection of tracer defined on the $T_{i,k,jrow}$ grid point. Units are in tracer units per second.

- $ADV\_Txiso_{i,k,j}$ = the counterpart to $ADV\_Tx_{i,k,j}$ using the Gent-McWilliams advective velocity which comes from parameterizing the effect of mesoscale eddies on isopycnals. Only used when option *gent_mcwilliams* is enabled. Units are in tracer units per second.

- $ADV\_Tyiso_{i,k,j}$ = the counterpart to $ADV\_Ty_{i,k,j}$ using the Gent-McWilliams advective velocity which comes from parameterizing the effect of mesoscale eddies on isopycnals. Only used when option *gent_mcwilliams* is enabled. Units are in tracer units per second.

- $ADV\_Tziso_{i,k,j}$ = the counterpart to $ADV\_Tz_{i,k,j}$ using the Gent-McWilliams advective velocity which comes from parameterizing the effect of mesoscale eddies on isopycnals. Only used when option *gent_mcwilliams* is enabled. Units are in tracer units per second.

- $DIFF\_Tx_{i,k,j}$ = the flux form of zonal (x) diffusion of tracer defined on the $T_{i,k,jrow}$ grid point. Units are in tracer units per second.

- *DIFF_Ty*$_{i,k,j}$ = the flux form of the meridional (y) diffusion of tracer defined on the $T_{i,k,jrow}$ grid point. Units are in tracer units per second.
- *DIFF_Tz*$_{i,k,j}$ = the flux form of the vertical (z) diffusion[6] of tracer defined on the $T_{i,k,jrow}$ grid point. Units are in tracer units per second.
- *DIFF_Tziso*$_{i,k,j}$ = the flux form[7] of the vertical component of isopycnal tracer diffusion defined on the $T_{i,k,jrow}$ grid point. Only used when option *isopycmix* is enabled. Units are in tracer units per second.

5.3.2 Momentum Operators

The details of the following operators used in solving the momentum equations are defined in file *fdifm.h*.

- *ADV_Ux*$_{i,k,j}$ = the flux form of the zonal (x) advection of momentum defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *ADV_Uy*$_{i,k,j}$ = the flux form of the meridional (y) advection of momentum defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *ADV_Uz*$_{i,k,j}$ = the flux form of the vertical (z) advection of momentum defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *ADV_metric*$_{i,k,j,n}$ = the metric term for momentum advection defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *DIFF_Ux*$_{i,k,j}$ = the flux form of the zonal (x) diffusion of momentum defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *DIFF_Uy*$_{i,k,j}$ = the flux form of the meridional (y) diffusion of momentum defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *DIFF_Uz*$_{i,k,j}$ = the flux form of the vertical (z) diffusion[8] of momentum defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *DIFF_metric*$_{i,k,j,n}$ = the metric term for momentum diffusion defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.
- *CORIOLIS*$_{i,k,j,n}$ = Coriolis term defined on the $U_{i,k,jrow}$ grid point. Units are in cm/sec^2.

5.4 Namelist variables

Although MOM 2 is configured in various ways using UNIX cpp options, the value of many of the variables and constants within MOM 2 and its parameterizations are defaulted and their values can be over-ridden using *namelist*[9] input. Included below are the variables input through namelists categorized by namelist name.

---

[6]This is the explicit portion of K^{33} indicated in 15.16.3 when option *isopyc* is enabled. If option *implicitvmix* is enabled, then it is the explicit part of the vertical diffusion.

[7]These are the K^{31} and K^{32} tensor components indicated in Section 15.16.3.

[8]If option *implicitvmix* is enabled, then it is the explicit part of the vertical diffusion.

[9]This is a non-standard Fortran 77 feature that is very useful. Most compilers support it. Refer to any Fortran manual for usage.

5.4.1 Time and date

Namelist */ictime/*

These variables are for use setting the time and date for initial conditions, referencing diagnostic calculations, and related items.

- *year0* = year of initial conditions (integer).
- *month0* = month of initial conditions (integer).
- *day0* = day of initial conditions (integer).
- *hour0* = hour of initial conditions (integer).
- *min0* = minute of initial conditions (integer).
- *sec0* = second of initial conditions (integer).
- *ryear* = user specified reference year (integer).
- *rmonth* = user specified reference month (integer).
- *rday* = user specified reference day (integer).
- *rhour* = user specified reference hour (integer).
- *rmin* = user specified reference min (integer).
- *rsec* = user specified reference sec (integer).
- *refrun* = Boolean used to specify that the time and date to be used for calculating diagnostic switches is the starting time and date of each submitted job. For example, suppose each job submission integrates for one month starting at the beginning of a month but the number of days per month changes. Setting "refrun" = true and setting "timavgint" = (days in month)/3 will give 3 averaged outputs per month at intervals of approximately 10 days each. The averaging period *timavgper* may be set less than *timavgint* for shorter averages but the output is still every *timavgint* days. The only restriction is that "timavgint" divided into the integration time for each job should work out to be an integral number (because this diagnostic is an average over time). If not, then "timavgint" is reset internally to insure this condition.
- *refinit* = Boolean used to specify that the time and date to be used for calculating diagnostic switches is the time and date of the initial conditions. For example, if term balances are desired every 20 days "trmbint" = 20.0, then they will be calculated and written out every 20 days starting from initial condition time.
- *refuser* = Boolean used to specify that the time and date to be used for calculating diagnostic switches is a user specified time and date given by (ryear, rmonth, rday, rhour, rmin, rsec) described above.for comparing diagnostics from various experiments with different initial condition times, *refuser* = *true* is appropriate. Note that setting *refuser* = *true* and choosing the reference time to be the initial condition time is the same as specifying *refinit* = *true*.

  Note that one of these reference time Booleans must be set *true* and the other two set *false*.

- *eqyear* = Boolean which forces all years to have the same number of days (no leap years). If *false*, then a Julian calendar is used.
- *eqmon* = Boolean which forces all months to have the same number of days. If *false*, then the actual number of days per month will be used. This is only used when *eqyear* is *true*.
- *monlen* = the length of each month in days when *eqmon* = *true*.

5.4.2 Integration control

Namelist */contrl/*

These variables are used for setting the integration time, when to initialize, and when to write restart files.

- *init* = Boolean for denoting whether the first time step of a run is to perform initializations such as reading in initial conditions, etc. When *init* = false, then the execution is started from a restart data file. Refer to *restrt* listed below.
- *runlen* = length of run in units given by *rununits*
- *rununits* = units for *runlen*. Either 'days', 'months' or 'years'.
- *segtim* = time in days for one segment of ocean or atmosphere. Only used with option *coupled*.
- *restrt* = Boolean for controlling whether a restart is to be written at the end of the run.
- *initpt* = Boolean for controlling whether particle trajectories are to be initialized on the first time step of a run. This is only used if option *trajectories* is enabled.

5.4.3 Surface boundary conditions

Namelist */mbcin/*

These variables are for surface boundary conditions.

- $mapsbc_m$ = indices for relating how surface boundary conditions are ordered. $m = 1$ to the maximum number of surface boundary conditions.
- $sbcname_m$ = character*10 names for surface boundary conditions.
- $dunits_m$ = character*15 names for units of surface boundary conditions.
- $coabc_m$ = conversion factors for converting surface boundary conditions from their dimensional units (in the model where they were constructed) to the other model's units. If option *coupled* is not enabled, then data is assumed to be in units required by MOM and no conversion is done.
- $crits_m$ = convergence criteria used to terminate solving the elliptic equation which extrapolates surface boundary conditions into land areas on the model grid. The extrapolation takes place on the native grid before interpolating to the other model grid. This only applies when option *coupled* is enabled.

- *numpas* = maximum number of iterations used to extrapolate data into land regions on the model grids where it was constructed. This only applies when option *coupled* is enabled.
- *bwidth* = blending zone width in degrees when using limited domain ocean models with global atmosphere models. This only applies when option *coupled* is enabled.
- *taux0* = zonal windstress in $dynes/cm^2$ for idealized equatorial studies. Refer to Section 15.9.2.
- *tauxy* = meridional windstress in $dynes/cm^2$ for idealized equatorial studies. Refer to Section 15.9.3.

5.4.4 Time steps

Namelist */tsteps/*

Historically, ocean time is measured in terms of tracer time steps. It should be noted that the equations in MOM 2 can be solved asynchronously (Bryan 1984) using one timestep for the internal mode *dtuv*, another for the external mode *dtsf*, and a third for the tracers *dtts*. Each are input through namelist.

Basically asynchronous timestepping can be done because the three processes have different time scales. Since the timescale for adjustment of density is much greater than that of velocity, it is reasoned that integrations to equilibrium can be speeded up by taking a large time step on the tracer equations (within CFL restrictions) and letting the velocities come into geostrophic adjustment with the density. If the problem is linear and only the equilibrium solution is sought, the equilibrium solution is unique and it doesn't matter how the integration gets there. However, if the solution is non-linear enough to have multiple equilibria or the transient response is of interest, all three time steps should be synchronous.

A similar argument can be made for the adjustment time scale of the deep layers being much greater than that of the surface layers. An acceleration with depth factor $dtxcel_k$, initialized to 1.0 for all levels, is used to increase the length of the tracer time step with depth to reach equilibrium sooner (Bryan 1984).

Given the resolution defined by module *grids*, a time step can be estimated from the linear CFL condition

$$\Delta t \leq \frac{\Delta_{min}}{2 \cdot c} \tag{5.1}$$

where Δ_{min} is the minimum grid cell width and c is the fastest wavespeed. The external gravity wave is the fastest wave with $c = \sqrt{grav \cdot H}$ but this has been filtered out of the equations by the rigid lid approximation. Next fastest are the low frequency external mode barotropic Rossby basin-scale waves with $c = -\frac{\beta}{k^2+l^2}$ where $k = \frac{2\pi}{L_x}$ and $l = \frac{2\pi}{L_y}$ are zonal and meridional wavenumbers. These might be expected to limit the time step but will not if sufficiently resolved by the grid as they are in most resolutions. Actually, Equation (5.1) applies only at the smallest grid scales ($2\Delta x$) and the eastward traveling short Rossby waves travel much slower. The next fastest waves to contend with are the high wavenumber internal mode gravity waves and the non-dispersive Kelvin wave. They are the appropriate wavespeeds for calculating timesteps. The fastest internal gravity wave speed is, $c \approx 3\ m/sec$. Note that the speed of the fluid is also dependent on subgrid scale parameters.

In some models, wavespeed is not the limiting factor for determining timestep length. In models with vertical resolution of approximately 10 meters thick, the time step may be limited by vertical velocity near the surface in regions such as the equator. In any event, it is recommended that diagnostic option *stability_tests* be enabled to show how close the model is to the local CFL condition and where that position is located. Large vertical diffusion coefficients can also limit the timestep length and when this is the case, option *implicitvmix* should be enabled to solve the vertical diffusion components implicitly.

Here are some rough examples from models run at GFDL.

- For a 2.4° by 2.4° mid latitude thermocline model, the following settings were used: $dtsf = dtuv = 3000.0$ sec; $dtts = 30000.0$ with acceleration of $dtts$ with depth using $dtxcel_1 = 1.0$ to $dtxcel_{km} = 5.0$.
- For a $\Delta_\lambda = 1°$ by $\Delta_\phi = 1/3°$ equatorial basin the following settings were used: $dtsf = dtuv = dtts = 3000.0$ sec.

The following variables set the time steps in seconds.

- *dtts* = time step length for tracers
- *dtuv* = time step length for internal mode velocities
- *dtsf* = time step length for external mode velocities

5.4.5 External mode

Namelist $/riglid/$

These variables are for setting limits for the elliptic solvers.

- *mxscan* = maximum number of iterations.
- *sor* = successive over-relaxation constant for use with option *oldrelax* and *hypergrid*.
- *tolrsf* = admissible error for the stream function in cm^3/sec. A reasonable starting point is 10^8.
- *tolrsp* = admissible error for the surface pressure in $gram/cm/sec^2$. A reasonable starting point is 10^{-4}.
- *tolrfs* = admissible error for the implicit free surface in $gram/cm/sec^2$. A reasonable starting point is 10^{-4}.

5.4.6 Mixing

Namelist $/mixing/$

These variables are for setting mixing and related values.

- *am* = lateral viscosity coefficient in cm^2/sec for option *consthmix*. Refer to Bryan, Manabe, Pacanowski (1975) for estimating a value.
- *ah* = lateral diffusion coefficient in cm^2/sec for option *consthmix*.
- *ambi* = absolute value of lateral viscosity coefficient in cm^4/sec for option *biharmonic*. Refer to Section 15.15.2 for estimating a value.

- *ahbi* = absolute value of lateral diffusion coefficient in cm^4/sec for option *biharmonic*. Refer to Section 15.15.2 for estimating a value.

- *kappa_m* = vertical viscosity coefficient in cm^2/sec for option *constvmix*.

- *kappa_h* = vertical diffusion coefficient in cm^2/sec for option *constvmix*.

- *cdbot* = bottom drag coefficient.

- *aidif* = implicit vertical diffusion factor. The vertical diffusion term is solved implicitly when option *implicitvmix* or *isopycmix* is enabled. Otherwise, *aidif* is not used! When solving implicitly, $0 \leq aidif \leq 1$ with $aidif = 1$ giving fully implicit and $aidif = 0$ giving fully explicit treatment. Why should semi-implicit vertical diffusion be used? The recommendation from Haltiner and Williams (1980) is for $aidif = 0.5$ when solving semi-implicitly. This gives the Crank-Nicholson implicit scheme which is always stable. This setting is supposed to be the most accurate one. However, this is not the case when solving a time dependent problem as discussed in Section 15.19.4. In cases where the vertical mixing coefficients severely limit the time step (because of fine vertical resolution or large coefficients) , this constraint on the time step can be relaxed by solving the vertical diffusion term implicitly.

- *ncon* = number of passes on the convective adjustment routine. Only meaningful when option *fullconvect* is not enabled and *implicitvmix* is not enabled.

- *nmix* = number of time steps between mixing time steps. A mixing time step is either a Forward or Euler Backward time step as opposed to the normal leapfrog time step.

- *eb* = Boolean for using Euler backward mixing scheme used on mixing time steps.

- *acor* = implicit Coriolis factor for treating the Coriolis term semi-implicitly. For semi-implicit treatment, $0.5 < acor < 1$ and option *damp_inertial_oscillation* must be enabled. Refer to Section 15.11.3 for a discussion of when this is appropriate.

- *dampts* = Newtonian damping time scale in days used with option *restorst*. Note that damping time scale may by set differently for each tracer.

- *dampdz* = Thickness in cm used to convert Newtonian damping to a flux when enabling option *restorst*. Note that damping thickness may by set differently for each tracer.

- *annlev* = Boolean for replacing seasonal sponge data by annual means when enabling option *sponges*.

5.4.7 Diagnostic intervals

Namelist /*diagn*/

These variables are used for setting diagnostic intervals and related items. The interval is a real number and has a Boolean variable (switch) associated with it which is set by the time manager every time step. The Boolean variable is set to *true* when the model integration time *mod* the interval is less than or equal to half a timestep. Otherwise, it is set *false*.

To add a switch, three variables must be added to the common block in *switch.h*: an interval (real numbner), a Boolean variable which acts as the logical switch, and an integer variable used as an index within module *tmngr*. Refer to include file *switch.h* to see the structure. Refer to the section where alarms are set within module *tmngr* for examples of how to implement new switches.

- *tsiint* = interval in days between writing time step integrals. This is used with option *time_step_monitor*.
- *tavgint* = interval in days between writing data for use with option *tracer_averages*.
- *itavg* = Boolean for writing regional mask when used with option *tracer_averages*. It should be set true on the first time step of the first run and false thereafter. This allows datasets from multiple runs to be concatenated without regional mask information being duplicated.
- *tmbint* = interval in days between writing data for option *meridional_tracer_budget*.
- *tmbper* = period in days for producing time averaged data for use with option *meridional_tracer_budget*. This averaging period may be set shorter than the interval *tmbint*.
- *itmb* = Boolean for writing "msktmb" when used with option *meridional_tracer_budget*.It should be set true on the first time step of the first run and false thereafter. This allows datasets from multiple runs to be concatenated without regional mask information being duplicated.
- *stabint* = interval in days between doing stability analysis for use with option *stability_tests*.
- *cflons* = starting longitude (deg) for computing data for use with option *stability_tests*.
- *cflone* = ending longitude (deg) for computing data for use with option *stability_tests*.
- *cflats* = starting latitude (deg) for computing data for use with option *stability_tests*.
- *cflate* = ending latitude (deg) for computing data for use with option *stability_tests*.
- *cfldps* = starting depth (cm) for computing data for use with option *stability_tests*.
- *cfldpe* = ending depth (cm) for computing data for use with option *stability_tests*.
- *maxcfl* = maximum number of CFL violations before quitting for use with option *stability_tests*.
- *zmbcint* = interval in days between writing data for option *show_zonal_mean_of_sbc*.
- *glenint* = interval in days between writing data for option *energy_analysis*.
- *trmbint* = interval in days between writing data for option *term_balances*.
- *itrmb* = Boolean for writing regional masks when used with option *term_balances*.
- *vmsfint* = interval in days between writing data for option *meridional_overturning*.
- *gyreint* = interval in days between writing data for option *gyre_components*.
- *exconvint* = interval in days between writing data for option *save_convection*.
- *cmixint* = interval in days between writing data for option *save_mixing_coeff*.

- *extint* = interval in days between writing data for option *show_external_mode*.
- *prxzint* = interval in days between writing data for use with option *matrix_sections*.
- *prlat* = latitudes for writing data for use with option *matrix_sections*.
- *prslon* = starting longitude (deg) for writing (xz) data for use with option *matrix_sections*.
- *prelon* = ending longitude (deg) for writing (xz) data for use with option *matrix_sections*.
- *prsdpt* = starting depth (cm) for writing (xz) data for use with option *matrix_sections*.
- *predpt* = ending depth (cm) for writing (xz) data for use with option *matrix_sections*.
- *slatxy* = starting latitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *elatxy* = ending latitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *slonxy* = starting longitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *elonxy* = ending longitude (deg) for writing (xy) data for use with option *matrix_sections*.
- *trajint* = interval in days between writing data for use with option *trajectories*.
- *dspint* = interval in days between writing data for use with option *diagnostic_surf_height*.
- *dspper* = period in days for producing time averaged data for use with option *diagnostic_surf_height*. This averaging period may be set shorter than the interval *dspint*.
- *snapint* = interval in days between writing data for use with option *snapshots*.
- *snapls* = starting latitude (deg) for writing data for use with option *snapshots*.
- *snaple* = ending latitude (deg) for writing data for use with option *snapshots*.
- *snapds* = starting depth (cm) for writing data for use with option *snapshots*.
- *snapde* = ending depth (cm) for writing data for use with option *snapshots*.
- *timavgint* = interval in days between writing data for use with option *time_averages*.
- *timavgper* = period in days for producing time averaged data for use with option *time_averages*. This averaging period may be set shorter than the interval *timavgint*.
- *xbtint* = interval in days between writing data for use with option *xbts*.
- *xbtper* = period in days for producing time averaged data for use with option *xbts*. This averaging period may be set shorter than the interval *xbtint*.

5.4.8 Directing output

Namelist */io/*

These variables are used for directing diagnostic output to either the model *printout* file which is in *ascii* or to 32 bit IEEE data files with suffixes *.dta*. These control variables will not direct output to NetCDF formatted files. NetCDF format is controlled by options described under each diagnostic in Chapter 18. Control variables are integers and exert control as follows:

- *If* control variable < 0 then output is written to unformatted IEEE file and *stdout*.
- *If* control variable > 0 *and* $\neq 6$ then output is written to unformatted IEEE file.
- *If* control variable $= 6$ then output is written to *stdout* which is file *fort.6*. The script *run_mom* redirects *stdout* to file *results* and copies it to a printout file.

The namelist variables are:

- *expnam* = character*60 experiment name.
- *iotavg* = control variable for writing output from option *tracer_averages*.
- *iotmb* = control variable for writing output from option *meridional_tracer_budget*.
- *iotrmb* = control variable for writing output from option *term_balances*.
- *ioglen* = control variable for writing output from option *energy_analysis*.
- *iovmsf* = control variable for writing output from option *meridional_overturning*.
- *iogyre* = control variable for writing output from option *gyre_components*.
- *ioprxz* = control variable for writing output from option *matrix_sections*.
- *ioext* = control variable for writing output from option *show_external_mode*.
- *iodsp* = control variable for writing output from option *diagnostic_surf_height*.
- *iotsi* = control variable for writing output from option *time_step_monitor*.
- *iozmbc* = control variable for writing output from option *show_zonal_mean_of_sbc*.
- *iotraj* = control variable for writing output from option *trajectories*.
- *ioxbt* = control variable for writing output from option *xbts*.

5.4.9 Isopycnal diffusion

Namelist */isopyc/*

These variables are for use with option *isopycmix*.

- *ahisop* = isopycnal diffusion coefficient in cm^2/sec.
- *athkdf* = isopycnal thickness diffusion coefficient in cm^2/sec. This is only used with option *gent_mcwilliams*.
- *slmx* = maximum slope of isopycnals.

5.4.10 Pacanowski/Philander mixing

Namelist */ppmix/*

These variables are for use with option *ppvmix*.

- *wndmix* = min value for mixing at surface to simulate high frequency wind mixing in cm^2/sec. (if absent in forcing).
- *frcmx* = maximum mixing in cm^2/sec.
- *diff_cbt_back* = background diffusion coefficient in cm^2/sec.
- *visc_cbu_back* = background viscosity coefficient in cm^2/sec.
- *diff_cbt_limit* = limiting diffusion coefficient in cm^2/sec. This is to be used in regions of negative Richardson number.
- *visc_cbu_limit* = limiting viscosity coefficient in cm^2/sec. This is to be used in regions of negative Richardson number.

5.4.11 Smagorinsky mixing

Namelist */smagnl/*

These variables are for use with option *smagnlmix*.

- *diff_c_back* = background diffusion coefficient which is added to predicted diffusion coefficient in cm^2/sec.

5.4.12 Bryan/Lewis mixing

Namelist */blmix/*

These variables are for use with option *bryan_lewis_vertical* and *bryan_lewis_horizontal*.

- Ahv_k = vertical diffusion coefficient for tracers as a function of depth in cm^2/sec.
- Ahh_k = horizontal diffusion coefficient for tracers as a function of depth in cm^2/sec.

5.4.13 Held/Larichev mixing

Namelist */hlmix/*

These variables are for use with option *held_larichev*.

- *hl_depth* = integration depth in cm.
- *hl_back* = background diffusion coefficient in cm^2/sec.
- *hl_max* = maximum allowable diffusion coefficient in cm^2/sec.

Chapter 6

Modules and Modularity

Modularity has various meanings depending upon who is consulted. For purposes of MOM 2, modules and modularity are key organizational tools used to minimize inter-connectivity between various model components so that the model's structure remains clear even when subjected to a large number of additions.

MOM 2 assumes an underlying structure which orchestrates the dataflow between disk and memory, the steps for solving prognostic equations, and where each possible option comes into play. This structure has enough generality to support many different parameterizations implemented as UNIX preprocessor "ifdef" options[1]. Conceptually, modularity has to do with minimizing contact between options and this underlying structure and also eliminating contact among options. It also implies something about how options are internally organized. In general, within each option, code is organized into a minimum number of non-contiguous segments. This is an important point because use of UNIX preprocessor "ifdefs" tend to encourage writing many non-contiguous code fragments resulting in "spaghetti code" which is best suited for land fill.

As Fortran 90 becomes more widely used, the idea of a Fortran 90 module will be incorporated into MOM 2. In the meantime, consider the construction of modules in MOM 2 as an approximation to the idea of a Fortran 90 module but which is compatible with Fortran 77.

6.1 What are modules

Modules are a method of organization which allows a large number of things to be managed in a reasonable way. There are of course many ways to organize and the best one depends not only on the problem but on the perspective of the researcher. For instance, a computational scientist's approach to numerical ocean modeling is not the same as an oceanographer's. Since MOM 2 is an oceanographic research tool, code is structured according to concepts that are important to the numerical oceanographer. The main idea is to increase modularity by arranging code into sections that fit the numerical oceanographer's conceptual model, isolating these code sections from each other as much as possible, and restricting how they interact.

As implemented in MOM 2, a module is a collection of subroutines which relate to the same concept and are bundled within a ".F" file. One useful feature of a module (as defined in MOM 2) is the capability of being executed in a stand alone mode apart from MOM 2 as well as from within MOM 2. Executing in a stand alone mode provides a simplified environment which aids in configuring the model, development work, and debugging. Many sections of MOM 2 are modular but cannot be run in a stand alone mode. In principle, they could become modules but

[1] When the number of options is large, an effective way is needed to point out incompatibilities and inconsistencies. This is the purpose of subroutine *checks*.

are so tied into the underlying structure of MOM 2 that providing a reasonable set of inputs is problematic. The diagnostics and subroutines that solve for the internal mode velocities or tracers are examples of modular code but not modules. Nevertheless, it is desirable to make modules where reasonable and an example of one is *grids* described in Chapter 7.

Refer to Figure 6.1 for a schematic of the anatomy of a module in MOM 2. As an example, consider this imaginary module named *example* which is contained within an imaginary file named *example.F*. Enabling a specific ifdef *option*[2] at compile time[3] would allow the module to run in stand alone mode. This means that one of its internal subroutines becomes a driver composed of code fragments and calls to the remaining internal subroutines in the module. The driver exercises these internal subroutines in a simplified environment by supplying a set of inputs and producing output that can be verified[4]. If the specific ifdef *option* were not enabled, the code for the driver would not be compiled and would add nothing to the load module. In this case, MOM 2 would assume the role of driver by supplying necessary input.

As indicated, subroutines may have an input/output list. In practice this may be an argument list, include files containing common blocks, or a combination of both. Deciding which one is best is a matter of judgement. When the input/output list is short, it makes sense to implement items as an argument list since this allows great portability[5]. However, when the input/output list is long, passing variables through many levels of subroutines is awkward at best and a nightmare at worst[6]. Common blocks allow easy access to variables from any subroutine. This is their greatest strength and also their greatest weakness. They expose all variables in the block to potential change by any subroutine that includes it whereas argument lists limit the exposure. Also, when exchanging subroutines between unrelated models, there is possibility for naming conflicts when common blocks are used. Nevertheless, experience with MOM 1 and MOM 2 indicates that the advantages of a well structured common block[7] outweigh the disadvantages.

Internal to each subroutine, details are pushed into lower level subroutines which are kept near the bottom of the module. If the overhead in calling the routine is not negligible, the routine should be inlined. On CRAY computers, this is a matter of supplying a list of routines to inline. Pushing details into lower level subroutines tends to make the structure of higher level routines more understandable. However, this process can be carried too far and judgement needs to be exercised. After applying this throughout MOM 2, certain lower level subroutines show up repeatedly. When this happens, they are extracted and put into a utility module. One example of this is *util.F*. This utility module can then be included as needed within other modules or MOM 2. The net effect is to end up with less redundant code and a restricted way in which subroutines interact within modules and modules interact with MOM 2.

6.2 List of Modules

The following sections contain a listing of modules by filename and a description of what they do. The run_scripts which activate these modules in a stand alone mode are all UNIX C shell scripts and should work on any workstation running UNIX. They are amenable to change and are intended as prototypes.

[2]Each module has a unique ifdef option which is given within the section which describes the module.

[3]This is usually done in the run_script. Enabling options can also be done another way by hardwiring them into the code using #define statements.

[4]Sometimes by the module itself. Other times by the judgement of the researcher based on output from the module.

[5]When this is done, the routine is termed "plug compatible".

[6]When code is being developed and constantly changing as in a research model.

[7]Variables are commented and organized into groups that make sense physically.

Recommendation

When experiencing problems in MOM 2 relating to modules, the recommendation is to run the module in isolation (stand alone mode) as described below in an attempt to reproduce the problem in a simpler environment. For example, if a problem is suspected with I/O, try to replicate it within the driver for the I/O manager *iomngr*. Or if a diagnostic is not being saved at the intended time, try to replicate this in the driver for the time manager module *tmngr*. For additional information, refer to related options in Chapter 15 as indicated below.

6.2.1 convect.F

File *convect.F* contains the convection module. It may be exercised in a stand alone mode by executing script *run_convect* which enables option *test_convect*. The driver is intended to show what happens to two bubbles in a one dimensional column of fluid when acted on by the old style explicit convection used in previous versions of MOM and a newer explicit convection scheme enabled by option *fullconvect*. Refer to Section 15.13 for details on both types of convection.

6.2.2 denscoef.F

File *denscoef.F* contains module *denscoef* which computes a set of coefficients $c_{k,\ell}$ used in a third order polynomial approximation to density given in Bryan and Cox (1972) as

$$\begin{aligned}[\rho(T,S,zt_k)-\rho_{0,k}]\cdot 10^3 &= c_{k,1}\delta T + c_{k,2}\delta S + c_{k,3}(\delta T)^2 + c_{k,4}(\delta S)^2 + c_{k,5}\delta T\delta S \\ &+ c_{k,6}(\delta T)^3 + c_{k,7}(\delta S)^2\delta T + c_{k,8}(\delta T)^2\delta S + c_{k,9}(\delta S)^3 \qquad (6.1)\end{aligned}$$

where δT and δS are departures of temperatures and salinities from mean values which are dependent on depth. In MOM 2, the polynomial can approximate either the Knudsen Formula (option *knudsen*) or UNESCO equation of state (the default) for sea water[8].

Generating the polynomial coefficients.

At each model depth zt_k, density is divided into 50 points between a minimum and maximum density. The minimum density is specified by the minimum in-situ temperature *tmin* and salinity *smin* and the maximum density is specified by the maximum in-situ temperature *tmax* and salinity *smax* based on observations (Levitus 1982). These temperature and salinity ranges are determined for the world ocean but may be further tightened to give increased accuracy for limited domains. The 50 densities are actually determined by dividing the range of specified in-situ temperatures into 10 equi-spaced temperatures and the range of specified salinities into 5 equi-spaced salinities. This yields 50 equations and 9 unknown polynomial coefficients at each model level. The system of equations is over determined and a best fit for the coefficients in the least squares sense is given by Hanson and Lawson (1969). In MOM 2, the solution is by a Jet Propulsion Laboratory subroutine "LSQL2". As a reality check, the specified in-situ temperature ranges are converted to potential temperature ranges for use within model diagnostics. When diagnostic option *stability_tests* is enabled, any predicted temperature or salinity outside of these ranges will be flagged.

Calculating density.

The polynomial approximation to density at any depth zt_k is calculated by first removing a reference potential temperature T_k^{ref} and reference salinity S_k^{ref} from the model potential temperature and salinity. The references are the means of the specified ranges described earlier.

[8] For the equations, refer to Gill (1982), Appendix Three.

The result is a potential density anomaly $\rho_{\tilde{t},\tilde{s},k}$ which is relative to a reference density ρ_k^{ref} implied by T_k^{ref}, S_k^{ref}, and depth zt_k. Only gradients of density are dynamically important and the horizontal gradient operator eliminates this reference. When vertical gradients of density are needed, both densities are referenced to the same local depth which again eliminates this reference. The equations are

$$\tilde{t} = t_{i,k,j,1,\tau} - T_k^{ref} \tag{6.2}$$

$$\tilde{s} = t_{i,k,j,2,\tau} - S_k^{ref} \tag{6.3}$$

$$\begin{aligned}\rho_{\tilde{t},\tilde{s},k} = {} & (c_{k,1} + (c_{k,4} + c_{k,7} * \tilde{s}) * \tilde{s} + \\ & (c_{k,3} + c_{k,8} * \tilde{s} + c_{k,6} * \tilde{t}) * \tilde{t}) * \tilde{t} + \\ & (c_{k,2} + (c_{k,5} + c_{k,9} * \tilde{s}) * \tilde{s}) * \tilde{s}\end{aligned} \tag{6.4}$$

The reason for calculating potential density anomaly is accuracy since the anomaly $\rho_{\tilde{t},\tilde{s},k} << \rho_k^{ref}$. If this were not done, three significant digits would be lost. The pressure effect with depth is incorporated into the polynomial coefficients as a function of model level k.

When computing horizontal pressure gradients, the potential density anomaly $\rho_{\tilde{t},\tilde{s},k}$ is integrated vertically using Equation (11.79) to construct pressure are level k. As noted above, it is of no consequence that the resulting pressure is an anomaly since horizontal derivatives eliminate ρ_k^{ref}. However, the effect of ρ_k^{ref} must be taken into account when constructing vertical derivatives. This is done by referencing temperature and salinity at both levels ($t_{i,k,j,n,\tau}$ and $t_{i,k+1,j,n,\tau}$) to the same ρ_k^{ref}

Normally, module *denscoef* is called from within a model execution to compute density coefficients. However, script *run_denscoef* enables option *drive_dencoef* and executes *denscoef.F* in a "stand alone mode"to produce a listing of the coefficients $c_{k,\ell}$ along with T_k^{ref}, S_k^{ref} and ρ_k^{ref}.

6.2.3 grids.F

File *grids.F* contains the *grids* module. Script *run_grids* uses option *drive_grids* to execute the module in a stand alone mode which is the recommended way to design a domain and grid resolution for MOM 2. The grid is specified in the USER INPUT section of file *grids.F.* When option *drive_grids* is not enabled, module *grids* is used by the model to install the specified grid information. Refer to Chapter 7 for a description of how to construct a grid.

6.2.4 iomngr.F

This is one of those places where the old code was overly complex with too many options but questionable gain. There is a newer, shorter, simpler, and easier to use version of the I/O manager tucked in at the top of file *iomngr.F.* This newer I/O manager works on SGI and CRAY platforms at GFDL. It can be tested by setting the appropriate computer PLATFORM variable in script *run_iomngr_new* and executing. For non-CRAY platforms use PLATFORM option "sgi". If executing on a Fortran 90 compiler, change the compile statement and add option *f90.* To access the newer I/O manager in other run scripts, just add option *new_iomngr.*

The purpose of the I/O manager is two fold: first, to find unit numbers which are not already attached to other files; and second to account for differences in the Fortran "open" statement depending on computer platform, Fortran 77, and Fortran 90. The older I/O manager will eventually become extinct and be replaced by the newer one. However, for now, the older one is still the default. To get the newer one in the model, add option *new_iomngr* to script *run_mom.*

If the newer I/O manager does not work properly on a particular platform, supply the needed fix and report what change was necessary. Enabling option *debug_iomngr* will help pinpoint a problem if it arises. If changes can be kept simple and clear, they will be incorporated at GFDL for all to use. Only "useful" basic file attributes used within MOM 2 are allowed in this newer I/O manager and should be built into the driver for testing with script *run_iomngr_new*. When, the older version is eliminated, file *iomngr.F* will be reduced in size and file *iomngr.h* will go away.

The interface to subroutines is much the same as in the older version except that no abbreviations are allowed. As an example, consider writing data to a file named "test.dta" which is to be a sequential unformatted file. The following call

```
call getunit (nu,'test.dta', 'sequential unformatted rewind')
```

finds a unit number "nu" which is not attached to any other opened file, assigns it to file "test.dta", and performs an "open" statement with the requested file attributes. After writing data to unit "nu", the unit can be closed and the unit number released with the following call

```
call relunit (nu)
```

The file may subsequently be opened with an "append" attribute using

```
call getunit (nu,'test.dta', 'sequential unformatted append')
```

to append data after which the file can again be closed and the unit number released with another call to "relunit(nu)". Possible file keyword attributes include:

- "sequential" for sequential files, or "direct" for direct access files, or "word " (note the blank at the end) for CRAY word I/O files.
- "formatted" or "unformatted".
- "rewind" or "append".
- "words=" for specifying record length in <u>words</u> for direct access files.
- "sds" for solid state disk on CRAY computers outfitted with solid state disk.
- "ieee" for 32bit IEEE format on CRAY computers
- "cray_float" for reading restart data written by a CRAY YMP or C90 but read on a CRAY T90 which has a 64bit IEEE native format.

Note that the argument lists are the same as in the older version but that abbreviations for file attributes are not allowed.

<u>Fortran 90</u>

When compiling under Fortran 90, option *f90* must be enabled to handle differences between Fortran 77 and Fortran 90. These differences occur only in file *iomngr.F*.

6.2.5 poisson.F

poisson.F contains the elliptic equation solver module. It is exercised in a stand alone mode by using script *run_poisson* which enables option *test_poisson*. Module *poisson* uses the grid defined by module *grids* and the topography and geometry defined by module *topog*.

Three methods of solution are provided and can be compared within module *poisson*: sequential relaxation enabled by option *oldrelax* in MOM 2, sequential relaxation alternately solving on the red/black squares of a checkerboard enabled by option *hypergrid* in MOM 2, and conjugate gradients enabled by option *conjugate_gradient* in MOM 2. These solvers form the basis of solving the external mode stream function and the newer rigid lid surface pressure and implicit free surface method developed by Dukowicz and Smith (1994). The first two methods require choosing a successive overrelaxation constant *sor* which may be input through namelist. Its optimal value is dependent on geography and topography. Refer to Section 5.4 for information on namelist variables. This module provides an easy way to optimize this constant. In principle, the conjugate gradient solver is the most efficient of the three and is the recommended one. The others are retained for compatibility reasons and as an alternative if the conjugate gradient solver should fail on a particular geometric configuration or value of the implicit Coriolis parameter *acor*.

The stopping criterion for convergence within the elliptic solvers is input through a namelist (*tolrsf* for stream function, *tolrsp* for surface pressure, and *tolrfs* for the implicit free surface). This criterion behaves differently than the corresponding variable *crit* which was used in MOM 1. In MOM 2, the sum of the truncated series of future corrections to the prognostic variable is estimated assuming geometric convergence and the iteration is terminated when this sum is within the requested tolerance. This means that the solution differs from the true solution to within an error given by the tolerance. The tolerance is given in the same units as the external mode prognostic variable. In MOM 1, when the residual was smaller than *crit* the iteration was stopped. This, however, did not mean that the solution was within *crit* of the true solution. Before using one of these solvers, it must be decided whether 5 point or 9 point numerics are to be used. For the rigid lid surface pressure and implicit free surface method, only the 9 point numerics are appropriate. For the stream function, either the 5 point (option *sf_5_point*) or the 9 point (option *sf_9_point*) numerics is appropriate. The recommendation is to use option *sf_9_point*. The generation of the coefficient matrix for the elliptic equation is described in Section 17.4. The resulting matrix is slightly different that the one calculated in previous versions of the model but presumed to be more accurate.

There are other differences besides the coefficient matrix when comparing the way the elliptic equation for the stream function was solved in MOM 1 and MOM 2. The result of the differences (as measured in the conjugate gradient) is that the solvers in MOM 2 converge faster and in less time than in MOM 1. How much is problem dependent but for the test cases measured, the difference is about 10 to 20% of the time taken by the external mode when solving to equivalent accuracy. Refer to Section 16.1.4 for details of the island equations.

6.2.6 ppmix.F

File *ppmix.F* contains the module which calculates vertical mixing coefficients based on the scheme of Pacanowski/Philander (1981). Script *run_ppmix* exercises this module in stand alone mode by enabling option *test_ppmix*. The driver uses a simplified 1-D equation configuration with Coriolis and vertical diffusion terms at a specific latitude to indicate how mixing penetrates vertically. The 1-D equations are

$$\partial_t u - f v = \partial_z(\kappa_m \cdot \partial_z(u)) \tag{6.5}$$

$$\partial_t v + f u = \partial_z(\kappa_m \cdot \partial_z(v)) \tag{6.6}$$

$$\partial_t T = \partial_z(\kappa_h \cdot \partial_z(T)) \tag{6.7}$$

with values of κ_m and κ_h being predicted by the scheme of Pacanowski/Philander (1981). Refer to Section 15.14.2 for details of the scheme.

6.2.7 timeinterp.F

File *timeinterp.F* contains the time interpolator module. It is exercised in a stand alone mode using script *run_timeinterp* which enables option *test_timeinterp*. The output indicates how surface boundary condition data (which may be monthly averages, daily averages, etc) is interpolated to the current time step in a simulated model integration. Keep in mind that interpolating surface boundary conditions in time only applies in the model when option *time_varying_sbc_data* is enabled.

Based on model time, the time interpolator decides when to read data from disk into memory buffers. When to read and which data to read is determined by the relation between the model time and the time at the centers of the data records. There are two memory buffers: one to hold data from the disk record whose centered time[9] precedes the model time (the previous data) and one to hold data from the disk record whose centered time has not yet been reached by the model time (the next data). Four alternative interpolation methods are supported as described in Section 10.2.

6.2.8 timer.F

File *timer.F* contains the timer module. It can be exercised in a stand alone mode by executing script *run_timer* which enables option *test_timer*. The module contains general purpose timing routines which are useful for optimizing any code. The timing routines consist of calls to"tic" and "toc" routines which are placed around code to be timed. Many levels of nesting are allowed as well as dividing times into categories and sub-categories. The driver (which is overly complex) simulates solving a tracer equation using various forms for calculating advection and diffusion. Timing results are given for each case. Experience shows there is no one form that is optimum on all computers, so if one wants to optimize speed for a particular environment, these routines will be useful. In MOM 2, most code sections are outfitted with calls to these timing routines. Enabling option *timing* in MOM 2 will give an indication of how much time is taken by various options and sections. These routines take time themselves to execute[10] and so should be disabled once results are obtained.

6.2.9 tmngr.F

File *tmngr.F* contains the time manager module. It can be exercised in stand alone mode by script *run_tmngr* which enables option *test_tmngr*. Based on settings outlined in the driver, a clock and calendar[11] are defined and used to decide when specific events will take place. To illustrate this, the driver sets a time step and specifies intervals for certain events. It then integrates time forward one time step at a time and indicates when the requested events take place.

In MOM 2, time is integrated in units of density time steps *dtts*. The input to the time manager is this density time step in seconds which is supplied as the sole explicit argument. The internal representation of time is kept as two integers: one for the number of whole days since December 31, 1899 at the start of the day at 00:00:00, and the other for the number of milliseconds within the last day. The benefit of doing it this way is no roundoff. At first, this may not seem important, but if not attended to, this will result in events happening unexpectedly. Time, as tracked by the time manager, is accurate to within one millisecond

[9]Date and time defined at the center of the data record. For instance, if the record was January (31 days long), the centered time would be at day 15.5 which is the center.

[10]This time is accounted for in the timer routines.

[11]The calendar may be realistic (Julian or Gregorian) or idealized.

for times up to about 5 million years using 32 bit integers. On 64 bit computers, time can be tracked to an accuracy of one millisecond for a period longer than the age of the universe.

The time manager is divided into two sections: one (*increment_time*) is a clock and calendar which integrates model time forward by one "time step" each time it is called; the other (*set_time_switches*) calculates when specific events (e.g. saving snapshot data, reading boundary condition data, etc.) will take place based on the model time and "intervals" or "averaging periods" set by the researcher.

Logical (Boolean) Switches.

Once model time corresponding to the current time step has been calculated by module *tmngr*, a long list of logical variables are set based on input variables specified by the researcher and the current model time. The logical variables act as switches which indicate whether specific events (diagnostics, mixing time steps, end of month, end of week, Tuesday, etc.) will happen or not during the current time step.

Include file *switch.h* contains all logical switches along with a long list of input variables used by the researcher to specify the interval in days between various requested events[12] or the period in days for averaging diagnostic quantities[13]. The list of input variables can be accessed through namelist and each input variable must have an associated logical switch. The logical switch is set to *true* when within half a time step of the requested interval or averaging period, otherwise it is set to *false*. Also associated with each switch is an internal variable containing the time when the logical switch will next be *true*.

Adding Switches.

In principle, any question regarding time can have a logical switch associated with it. It is easy to add new switches following the examples in file *switch.h*. As a naming convention, all logical switches end in *ts*. One example is *snapts* which is the logical for determining whether a snapshot of the model solution is to be taken during the current time step. Use the UNIX *grep snapts *.[Fh]* to see where the switch *snapts* is defined and calculated, then replicate the form. Before adding a new switch, check through file *switch.h* because the switch may already be there. The following examples further indicate how to add a logical switch for various purposes in MOM 2:

Switches based on an interval.

Suppose the intent was to compute global energetics every 15.0 days. The following three variables would be added (they are already there) to the common block in file *switch.h*. Let *glen* be the desired mnemonic for global energetics, then the naming convention used in *switch.h* implies the following

- glenint = an interval (real) for diagnostic output.
- glents = a switch (logical) corresponding to the interval.
- iglenint = internal (integer) variable needed by the time manager to support calculation of the logical switch.

The researcher specifies the interval [e.g., glenint] for diagnostic output in units of days through namelist. tmngr sets the corresponding logical switch [e.g., glents] every time step.

[12]The interval must be referenced to a starting time which may be specified as the beginning of the experiment, the beginning of a particular job, or a specific date in time.

[13]Each event may have its own interval and averaging period.

It is set to true when within half a time step of the requested interval, otherwise it is false. All decisions relating to the interval [e.g., glenint] are based on the logical switch [e.g., glents]. The following statement placed inside the time manager in the section for switches based on an interval will calculate the logical switch.

glents = alarm (iglenint, ihalfstep, glenint, iref)

In the above, variables *ihalfstep* and *iref* are calculated internally by the time manager. Switch *glents* is updated every time step and is available anywhere in MOM 2 by including file *switch.h* in the routine where the switch is needed.

Switches based on an interval and averaging period.

Suppose the intent was to compute 5.0 day averages of XBT's every 30.0 days. The following five variables would be added to the common block in file *switch.h*. Let *xbt* be the mnemonic for XBT's, then the naming convention used in *switch.h* implies

- xbtint = an interval (real) for diagnostic output
- xbtts = a switch (logical) corresponding to the interval
- xbtper = an averaging period (real)
- xbtperts = a switch (logical) corresponding to the averaging period
- ixbtint = internal variable needed by the time manager to support calculation of the logical switches

The researcher specifies the interval [e.g., xbtint] for diagnostic output and averaging period [e.g., xbtper] in units of days through namelist and *tmngr* sets the corresponding logical switches [e.g., xbtts, xbtperts] every time step. They are set to true when within half a time step of the requested interval or averaging period, otherwise they are false. In this example, *xbtts* will be *true* on the time step corresponding to day 30.0 and *xbtperts* will be *true* on all time steps within days 26.0 through 30.0. On the next interval, *xbtts* will be *true* on the time step corresponding to day 60.0 and *xbtperts* will be *true* on all time steps within days 56.0 through 60.0. The following statements placed inside the time manager in the section for switches based on interval and averaging period will calculate both switches.

xbtts = avg_alarm(ixbtint, ihalfstep, xbtint, xbtper, iref, 0)
xbtperts = on(ixbtint)

In the above, variables *ihalfstep* and *iref* are calculated internally by the time manager. Functionavg_alarm is internal to the time manager as well as array *on()*. Switches *xbtts* and *xbtperts* are updated every time step and are available anywhere in MOM 2 by including file *switch.h* in the routine where the switches are needed.

Switches based on the calendar or previously computed switches.

Suppose the intent was to compute 5.0 day averages of XBT's at the end of each month realizing that each month has a different number of days. The following four variables would be added to the common block in file *switch.h*. Let *test* be the mnemonic, then the naming convention used in *switch.h* implies

- testper = an averaging period (real)

- testts = a switch (logical) corresponding to the end of month
- testperts = a switch (logical) corresponding to the averaging period
- itestint = internal variable needed by the time manager to support calculation of the logical switches

The researcher specifies the averaging period [e.g., testper] in units of days through namelist and *tmngr* sets the corresponding logical switches [e.g., testts, testperts] every time step. They are set to true when within half a time step of the requested interval or averaging period, otherwise they are false. In this example, *testts* will be *true* on the time step corresponding to the end of the month and *testperts* will be *true* on all time steps within the last 5.0 days of the month. The following statements placed inside the time manager in the section for switches based on calendar or previous switch and averaging period will calculate both switches.

testts = avg_alarm(itestper, ihalfstep, 0, testper, iref, ieomon)
testperts = on(itestper)

In the above, variables *ihalfstep* and *iref* are calculated internally by the time manager. also, internal variable *ieom* is used to reference into the end of month time structure. Function-avg_alarm is internal to the time manager as well as array *on()*. Switches *testts* and *testperts* are updated every time step and are available anywhere in MOM 2 by including file *switch.h* in the routine where the switches are needed.

6.2.10 topog.F

File *topog.F* contains the topography module which is used to design a topography and geometry for the particular resolution specified by module *grids*. The module is exercised in stand alone mode by script *run_topog* which enables option *drive_topog*. This is the recommended way of generating topography and geometry. When option *drive_topog* is <u>not</u> enabled, the geometry and topography are constructed from with the model. Before executing module *topog*, the grid must be defined using module *grids*. Refer to Chapter 9 for a description of how to construct geometry and topography.

6.2.11 util.F

File *util.F* contains the utility module. It is exercised in stand alone mode by script *run_util* which enables option *test_util*. The module is a collection of various subroutines or utilities used throughout MOM 2. All output from these utilities is passed through argument lists. Input to the utilities is also passed through argument lists, except for dimensional information which is passed by including file *size.h* within each utility. The include file is used for purposes of parallelization. If it were not for this include file, this module would be truly "plug compatible" and highly portable. The following is a summary of subroutines within the utility module:

<u>indp</u>

Function *indp* (index of nearest data point) searches an array to find which element is closest to a specified value and returns the index of that array element.

<u>ftc</u>

Subroutine *ftc* (fine to coarse) interpolates data defined on a fine resolution grid to a coarser resolution grid. It does this by averaging together all cells on the fine grid which lie within each coarse cell. Partial overlapping areas are taken into account.

<u>ctf</u>

Subroutine *ctf* (coarse to fine) linearly interpolates data defined on a coarse resolution grid to a fine resolution grid.

<u>extrap</u>

Subroutine *extrap* extrapolates data defined at ocean cells to land cells on the same grid. The intent is to force oceanographic quantities into land areas to allow reasonable values for interpolations involving coastlines. This is important in air/sea coupled models where coastlines and resolution may not match between models.

<u>setbcx</u>

Subroutine *setbcx* sets boundary conditions on arrays.

<u>iplot</u>

Subroutine *iplot* plots an integer array with characters thereby giving a quick “contour” map of the data.

<u>imatrix</u>

Subroutine *imatrix* prints values of an integer array in matrix format.

<u>matrix</u>

Subroutine *matrix* prints values of an real array in matrix format.

<u>scope</u>

Subroutine *scope* interrogates an array for the minimum, maximum and simple unweighted average. It also list their positions within the array.

<u>sum1st</u>

Subroutine *sum1st* performs a simple sum on the first index of and array for each value of the second index.

<u>plot</u>

Subroutine *plot* contours an array by dividing the array values into bins and assigning a character to each bin. It then prints the characters to produce a quick “contour” map. The array must be real.

<u>print_checksum</u>

Subroutine *print_checksum* prints a checksum for a two dimensional array.

<u>checksum</u>

Function *checksum* calculates a checksum for a two dimensional array and returns the value. The value is not printed.

wrufio

Subroutine *wrufio* writes an array as an unformatted fortran write. The purpose is to speed up writing by eliminating the indexed list when writing sub-sections of arrays.

rrufio

Subroutine *rrufio* reads an array as an unformatted fortran read. The purpose is to speed up reading by eliminating the indexed list when reading into sub-sections of arrays.

tranlon

Subroutine *tranlon* is only used in limited domain models to move the Greenwich meridian outside of the limited grid domain. It translates data in longitude and redefines longitudes to accommodate interpolating data which starts and ends at the Greenwich meridian. Recall that for interpolating data, grid coordinates must be monotonically increasing with increasing index.

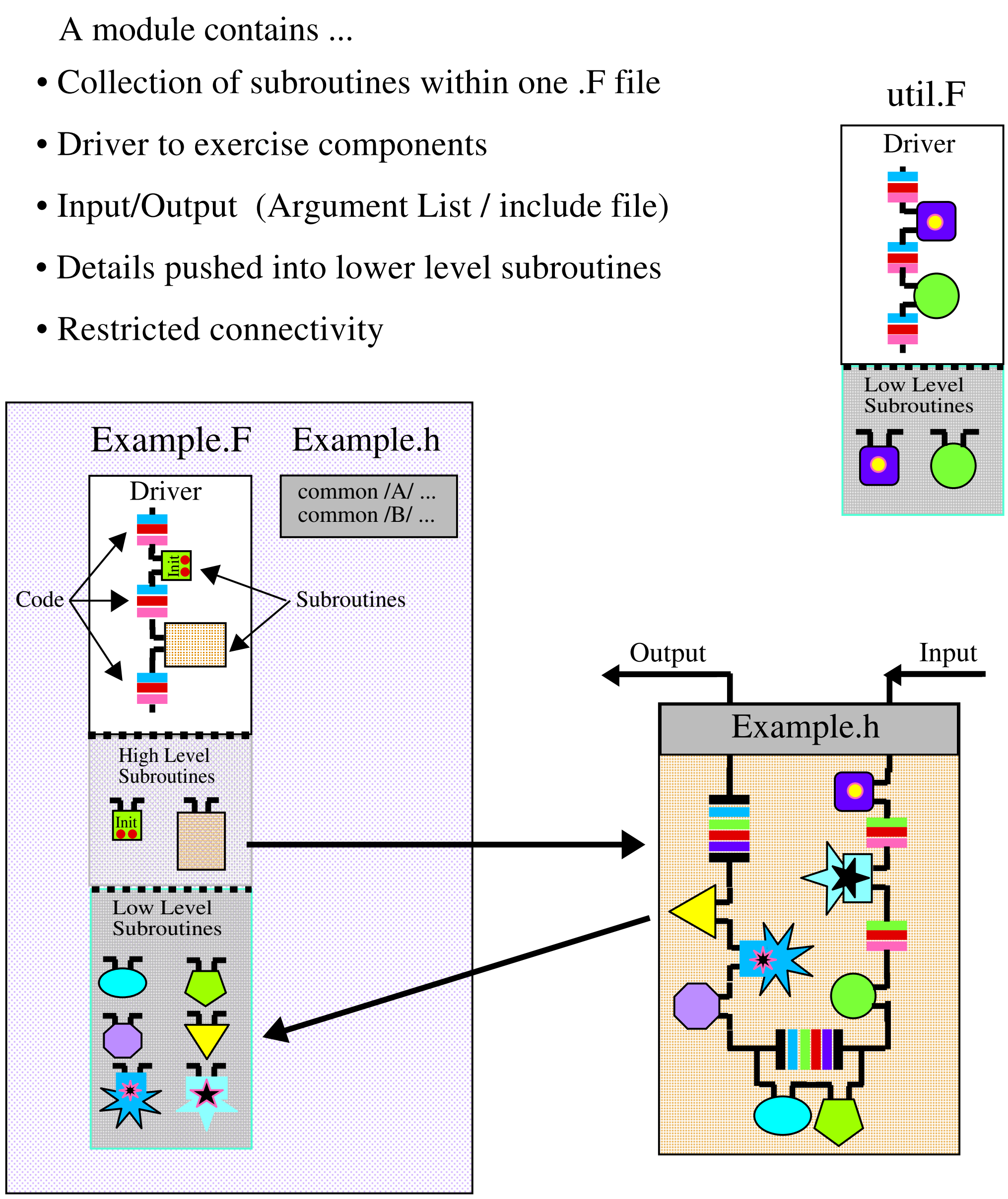


Figure 6.1: Schematic idealization of the anatomy of a module in MOM 2

Chapter 7

Grids

Before constructing surface boundary conditions, initial conditions, or executing a model, the model domain and resolution must be specified. The specification takes place within module *grids* which is contained in file grids.F.

7.1 Domain and Resolution

The model domain, which may be global, is defined as a thin shelled volume bounded by six coordinates: two latitudes, two longitudes, and two depths on the surface of a spherical earth. Embedded within this domain is a "rectangular" grid system aligned such that the principle directions are along longitude λ (measured in degrees east of Greenwich), latitude ϕ (measured in degrees north of the equator), and depth zt (measured in centimeters from the surface of the spherical shell to the ocean bottom). Note that the vertical coordinate z increases upwards.

7.1.1 Regions

The domain may be further sub-divided into regions; each of which is similarly bounded by six coordinates: two latitudes, two longitudes, and two depths. Within each region, resolution can be specified as constant or non-uniform along each of the coordinate directions. Although non-uniform resolution is permitted, it is not allowed in the sense of generalized curvilinear coordinates. Resolution along any coordinate is constrained to be a function of position along that coordinate. For instance, vertical thickness of grid cells may vary with depth but not with longitude or latitude.

7.1.2 Resolution

The resolution within a region is determined by the width of the region and resolution at the bounding coordinates. Along any coordinate, if resolution at the bounding coordinates is the same, then resolution is constant across the region, otherwise it varies continuously from one boundary to the other according to an analytic function (Treguier, Dukowicz, and Bryan 1995). The function describing the variation is prescribed to be a cosine. Although arbitrary, this function has two important properties: it allows the average resolution within any region to be calculated as an average of the two bounding resolutions; and it insures that the first derivative of the resolution vanishes at the region's boundaries. A vanishing first derivative allows regions to be smoothly joined. The only restriction is that there is an integral number of grid cells within a region.

To formalize these ideas, let a region be bounded along any coordinate direction (latitude, longitude, or depth) by two points α and β at which resolutions are Δ_α and Δ_β. The number of discrete cells N contained between α and β is given by

$$N = \frac{|\beta - \alpha|}{(\Delta_\alpha + \Delta_\beta)/2} \tag{7.1}$$

where N must be an integer and the resolution for any cell Δ_m is given by

$$\Delta_m = \frac{\Delta_\alpha + \Delta_\beta}{2} - \frac{\Delta_\beta - \Delta_\alpha}{2} \cos(\pi \frac{m - 0.5}{N}) \tag{7.2}$$

where $m = 1 \cdots N$. As an example, if α and β were longitudes, the western edge of the first cell would be a α and the eastern edge of cell N would be at β.

7.1.3 Describing a domain and resolution

A grid domain and resolution is built by specifying bounding coordinates and resolution at those coordinates for each region.

Example 1

Imagine a grid with a longitudinal resolution $\Delta_\lambda = 1°$ encircling the earth and a meridional resolution $\Delta_\phi = 1/3°$ equatorward of 10°N and 10°S, and a vertical grid spacing $\Delta_z = 10$ meters between the surface and a depth of 100 meters. This domain and resolution is specified in the following manner. Two bounding longitudes: one at 0°E and the other as 360°E with $\Delta_\lambda = 1°$ at both longitudes; two bounding latitudes: one at $-10°$ and the other at $+10°$ with $\Delta_\phi = 1/3°$ at both latitudes; and two bounding depths: one at 0cm and the other at $100\text{x}10^2$cm with $\Delta_z = 10\text{x}10^2$cm at both depths. These specifications imply 360 grid cells in longitude, 60 grid cells in latitude, and 10 grid cells in depth[1]

Example 2

In the preceeding example, if it were desired to extend the latitudinal domain poleward of 10°N and 10°S to 30°N and 30°S where the meridional resolution was to be $\Delta_\phi = 1°$, then the two previous bounding latitudes would need to be replaced by four: one at $-30°$ where $\Delta_\phi = 1°$, one at $-10°$ where $\Delta_\phi = 1/3°$, one at $+10°$ where $\Delta_\phi = 1/3°$ and one at $+30°$ where $\Delta_\phi = 1°$. Poleward of 10 degrees, meridional resolution would telescope from $\Delta_\phi = 1/3°$ to $\Delta_\phi = 1°$ over a span of 20°. The average meridional resolution in this region is calculated as the average of the bounding resolutions which is $\frac{1°+1/3°}{2} = 2/3°$. Therefore, there would be $\frac{20°}{2/3°} = 30$ additional grid cells in each hemisphere between latitudes 10° and 30°.

Example 3

Suppose it was desired to construct a square grid between latitude 60°S and 60°N with 1° resolution at the equator. The bounding longitudes would be set as in Example 1. To keep the grid cells square, the latitudinal resolution at 60° would be $\Delta_\phi = 1° \cdot \cos 60° = 0.5°$. Therefore, two latitudinal regions are required: The first is specified as being bounded by latitude 60°S where $\Delta_\phi = 0.5°$ and latitude 0°S where $\Delta_\phi = 1.0°$; the second is bounded by latitude 0°S where $\Delta_\phi = 1.0°$ and latitude 60°N where $\Delta_\phi = 0.5°$. Each region has a width of 60° and $N = \frac{60°}{(1°+0.5°)/2} = 80$ grid cells.

[1]Actually, two extra boundary cells are added to the grid domain in the latitude and longitude dimensions, but not in the vertical (historical reasons). Calculations range from $i = 2$ to $imt - 1$ in longitude, $jrow = 2$ to $jmt - 1$ in latitude, and $k = 1$ to km in depth where domain size is (imt x jmt x km) cells.

7.2 Grid cell arrangement

The grid system is a rectangular Arakawa staggered B grid (Bryan 1969) containing T cells and U cells. In order to visualize this arrangement, it will be helpful to refer to Figs 7.1, 7.2, and 7.3 which depict grid cells within horizontal and vertical surfaces. Within each T cell is a T grid point which defines the location of tracer quantities. Similarly, each U cell contains a U grid point which defines the location of the zonal and meridional velocity components.

7.2.1 Relation between T and U cells

Within a horizontal surface at depth level k, grid points and cells are arranged such that a grid point $U_{i,k,jrow}$ (where subscript i is the longitude index, *jrow* is the latitude index, and k is the depth index) is located at the northeast vertex of cell $T_{i,k,jrow}$. Conversely, the grid point $T_{i,k,jrow}$ is located at the southwest vertex of cell $U_{i,k,jrow}$. $\mathrm{T}_{i=1,k,jrow=1}$ is the southwestern most T cell and $\mathrm{T}_{i=imt,k,jrow=jmt}$ is the northeastern most T cell within the grid. This horizontally staggered grid system is replicated and distributed vertically between the ocean surface and bottom of the domain. $\mathrm{T}_{i,k=1,jrow}$ is the first cell below the surface and $\mathrm{T}_{i,k=km,jrow}$ is the deepest cell. Unlike in the horizontal, T cells and U cells are not staggered vertically so all T cells and U cells with index k are at the same depth.

7.2.2 Regional and domain boundaries

As mentioned in Section 7.1, when specifying bounding coordinates and resolution, there must be an integral number of cells contained between these coordinates in each of the coordinate directions. The integral number of cells refers specifically to T cells. In the horizontal plane, bounding surfaces of constant longitude and latitude define the location of U grid points and resolution at those surfaces is given to the corresponding U cells. Resolution varies continuously between bounding surfaces and is discretized, using Equations (7.1) and (7.2), to cell widths and heights. In the vertical plane, bounding surfaces are at the top or bottom of cells and resolution is discretized to cell thicknesses as in the horizontal plane. It should be noted that in the vertical, allowance is made for a stretching factor in the last region to provide for a more drastic fall off of resolution to the bottom if desired.

7.2.3 Non-uniform resolution

Within a region of constant resolution, all T and U grid points are located at the centers of their respective cells. When resolution is non-uniform, this is not the case. Within MOM 2, there are two methods to discretize non-uniform resolution onto T cells and U cells. Based on Equations (7.1) and (7.2) in section 7.1, and the averaging operator given by

$$\overline{(\xi)}^{m} = \frac{(\xi)_m + (\xi)_{m+1}}{2}, \tag{7.3}$$

the two methods are

1. $\Delta^T_m = \frac{\Delta^U_\alpha + \Delta^U_\beta}{2} - \frac{\Delta^U_\alpha - \Delta^U_\beta}{2}\cos(\pi\frac{m-0.5}{N})$; $\quad \Delta^U_m = \overline{\Delta^T_m}^{m}$
2. $\Delta^U_m = \frac{\Delta^U_\alpha + \Delta^U_\beta}{2} - \frac{\Delta^U_\alpha - \Delta^U_\beta}{2}\cos(\pi\frac{m}{N})$; $\quad \Delta^T_m = \overline{\Delta^U_m}^{m}$

where Δ^U_α and Δ^U_β are resolution of U cells at the bounding surfaces α and β, N is the number of cells given by Equation (7.1) in section 7.1, and the subscript m refers to longitude index i or latitude index j. These methods can also be applied to cells in the vertical, even though

T and U cells are not staggered vertically. Refer to Figure 7.4 and assume phantom W cells (of thickness dzw_k) staggered vertically such that the W grid point within cell W_k lies at the bottom of cell T_k and the T grid point within cell T_k lies at the top of cell W_k. Now replace U by W in the expressions for both methods given above.

The motivation for method 2 is first to notice that on a non-uniform grid, advective velocities are a weighted average of velocities but the denominator is not the sum of the weights as indicated in Section 11.5. This form of the advective velocities is implied by energy conservation arguments as given in Section 12.3. Secondly, the average of the quantity being advected is not defined coincident with the advecting velocity. Redefining the average operator in Equation (7.3) results in second moments not being conserved as indicated in Chapter 12. Method 2 remedies both problems by simply redefining the location of grid points within grid cells. All equations remain the same.

In method 1, U cell size is the average of adjacent T cell sizes. This means that T points are always centered within T cells, but U points are off center when the grid is non-uniform. This was the method used in model versions prior to MOM 2. In method 2, the construction is the other way around: T cell size is the average of adjacent U cell sizes. Accordingly, U points are always centered within U cells but T points are off center when the resolution is non-uniform. It should be noted that MOM 2 allows both methods. Although the default grid construction is by method 2, enabling option *centered_t*[2] when compiling will result in grid construction by method 1.

Accuracy of numerics

When resolution is constant, the finite difference numerics are second order accurate. In this case, grid cells and grid points are at the same locations regardless of whether they are constructed using method 1 or method 2. However, contrary to widespread belief, when resolution is non-uniform, numerics are still second order accurate if the stretching is based on a smooth analytic function. See Treguier, Dukowicz, and Bryan (1995).

Even though methods 1 and 2 are second order accurate, is one slightly better than the other? In particular, does the horizontal staggering of grid cells implied by method 2 give slightly better horizontal advection[3] of tracers while the staggering implied by method 1 give more accurate horizontal advection of momentum? Also, since T cells and U cells are not staggered in the vertical, does method 2 gives more accurate vertical advection of tracers and momentum than method 1?

The reasoning behind these questions can be seen by referring to Figure 7.4 and noting the placement of T grid points within T cells[4]. Advective fluxes are constructed as the product of advective velocities and averages of quantities to be advected.

$$advective\ \ flux = W_k \cdot \frac{T_k + T_{k+1}}{2} \tag{7.4}$$

When resolution is constant, both advective velocity and averaged quantities lie on cell faces, but when resolution is non-uniform, averaged quantities may lie off the cell faces. Whether or not this happens depends on the placement of grid points within grid cells. Method 2 defines tracer points off center in such a way that the averaged tracer given by Equation (7.3) is placed squarely on the cell faces. Recall from Chapter 12 that defining the average operator differently than in Equation (7.3) will not conserve second moments.

Simple one dimensional tests using a constant advection velocity of 5 cm/sec to advect a gaussian shaped waveform through a non-uniform resolution varying from 2 to 4 degrees and

[2] Centered refers to the centering of the t grid point within the T grid cell.

[3] Of course, this is not to be compared to the significant gains due to higher order advective schemes.

[4] Although the argument is given for the vertical direction, it applies in the horizontal as well.

back to 2 degrees suggests that method 2 is better than method 1. Also, a one dimensional thermocline model employing a stretched vertical coordinate indicates again that method 2 is better than method 1. In both cases, better means that the variance of the solution was closer to the variance of the analytic solution by a few percent. In short integrations using MOM 2, the effect showed up as less spurious creation of tracer extrema in grids constructed with method 2 as compared to method 1. Whether this difference is robust in all integrations has not been demonstrated. Nevertheless, in light of these results, the default grid construction in MOM 2 is method 2. Method 1 can be implemented by using option *centered_t*.

It should be stressed, that regardless of which method of grid construction is used, the equations don't change and first and second moments are conserved. Of primary importance when constructing a grid is whether the physical scales are adequately resolved by the number of grid points. Beyond this, grid construction by method 1 or method 2 is of secondary importance.

7.3 Constructing a grid

The domain and resolution is constructed in MOM 2 by calling module *grids* to construct a grid system. Executing MOM 2 to design a grid is similar to using a sledge hammer to work with tacks. Instead, module *grids* can be executed in a much simpler environment (without the rest of MOM 2) by using script *run_grids*. This is a UNIX C shell script which assumes an *f77* compiler which allows the script to run on workstations. To run on a CRAY, change the *f77* to *cf77* or *f90* if a Fortran 90 compiler is used. Although the script can be executed from the MOM_2 directory, it's a safer practice to use a MOM_UPDATES directory for each experiment as described in Section 19.6. Copy both *run_grids* and file *grids.F* into the MOM_UPDATES directory, and make changes there.

To define a grid, go to the USER INPUT section of module *grids*. After reading the information in this chapter and looking at the examples given in module *grids*, implementing a grid design should be straightforward. About the only potential problem might be that a particular specification leads to a non-integral number of T cells within a region. Recall that the number of T cells can be found by dividing the span of the region by the average resolution. Either the position of the bounding coordinates or the resolution at these coordinates may be changed to resolve the problem.

In the USER INPUT section, the bounding coordinates are specified by variables *x_lon*, *y_lat*, and *z_depth*. Variable *x_lon* is dimensioned by parameter *nxlons* which gives the number of bounding longitudes to define one or more regions in longitude. Units are in degrees of longitude measured at the equator and these points define the longitude of U grid points. Refer to Figs 7.1, 7.2, and 7.3 which indicate U grid points on bounding coordinates with an integral number of T cells inbetween. Similarly, variable *y_lat* is dimensioned by parameter *nylats* which gives the number of bounding latitudes to define one or more regions in latitude. These coordinates mark the position on U points in latitude. Variable *z_depth* is dimensioned by parameter *nzdepths* which gives the number of bounding depth coordinates to define one or more regions in depth. Units are in *cm* and mark where the position of the bases of T and U cells will be. Note that the bottoms of T and U cells define where the vertical velocity points are located.

Associated with variables *x_lon*, *y_lat*, and *z_depth* are variables which define the respective resolution *dx_lon*, *dy_lat*, and *dz_depth* of cells at the bounding coordinates. Units are in degrees for *dx_lon* and *dy_lat* but in *cm* for *dz_depth*. Note the variable *stretch_z* provides additional stretching for the last region in the vertical. When *stretch_z*= 1.0, there is no additional stretching. However, when *stretch_z* > 1.0 additional stretching is applied. To see how it works, set *stretch_z* $= 1.1$ and gradually increase it. It can be used to approximate an exponential fall off in the vertical.

When executed, script *run_grids* creates a sub directory and compiles module *grids* using option *drive_grids* to activate a driver as the main program. After executing, results are copied to a file in the directory from which script *run_grids* was executed and the sub-directory is eliminated. View the file *results_grids* with an editor. When satisfied, copy *size.h* from the MOM_2 directory and change the parameters according to the indicated directions. Any component of MOM 2 which accesses the updated module *grids* and *size.h* will get the new grid. All components of MOM 2 which use module *grids* and *size.h* perform a consistency check. If there is an inconsistency, MOM 2 will give an error message and stop.

7.3.1 Grids in two dimensions

When the grid system is contracted to a minimum along one dimension, MOM 2 is essentially reduced to a two dimensional model. For instance, if it is desirable to have a two dimensional model that is a function of latitude ϕ and depth z, then setting $nxlons = 2$ and specifying dx_lon_1, dx_lon_2 such that there are two grid cells between bounding surfaces x_lon_1, x_lon_2 will generate $imt = 4$ grid cells in the longitudinal direction. Two T cells $i = 2$ and $i = 3$ will be calculated and the two extra T cells $i = 1$ and $i = 4$ are for boundaries. Only one U cell, $i = 2$, is not in the boundary. If option *cyclic* is enabled, then the domain is zonally re-entrant. If the forcing and initial conditions are independent of longitude λ, then the solution is independent of λ and the model is two dimensional in ϕ and z. Obviously the relevance of this model depends on the scientific question being posed. This is just to demonstrate how the longitudinal dimension can be contracted to a minimum of $imt = 4$ cells. The memory window should be opened to $jmw = jmt$ to make it as efficient as possible but this will not be as fast as the two dimensional model in λ and z discussed below because of the short vector lengths in longitude.

A similar contraction can be performed in the latitudinal direction to end up with a two dimensional model in longitude λ and depth z. Here again, the minimum number of latitudes is $jmt = 4$ with $jrow = 2$ and $jrow = 3$ being ocean T cells and $jrow = 1$ and $jrow = 4$ being land cells. Note that there is only one latitude of ocean U cells. If these U cells are placed at the equator (with the two ocean T cell latitudes placed symmetrically about the equator) and there is no meridional variation in initial conditions or forcing, then the model is two dimensional in λ and z. Again, the scientific question being posed needs to be suited to this design. Note that this type of model really flies computationally because of the long vectors in the longitude dimension. The memory window should be opened to $jmw = jmt$ to make it as efficient as possible.

In the vertical, the minimum number of ocean levels is 2. Therefore the bounding surfaces and resolutions can be set to yield $km = 2$. Note that $kmt_{i,jrow} < 2$ is not allowed.

7.4 Summary of options

The following options are used by module *grids* and are enabled by compiling with options of the form *-Doption1 -Doption1* ...

- *drive_grids* turns on a small driver which allows the module *grids* to be executed in a simple environment. This is only used for designing grids with the script *run_grids*. It is not appropriate when executing MOM 2.

- *generate_a_grid* generates a grid based on specifications in the USER INPUT section of module *grids*. It is used both in the script *run_grids* and when executing MOM 2.

- *read_my_grid* allows grids developed elsewhere to be imported into MOM 2.
- *write_my_grid* writes a copy of the grid information to file *grid.dta.out*.
- *centered_t* constructs a grid using method 1 from Section 7.2.3. If not enabled, the grid is constructed using method 2 from Section 7.2.3 which is different than the way it has been constructed in previous versions of the model.

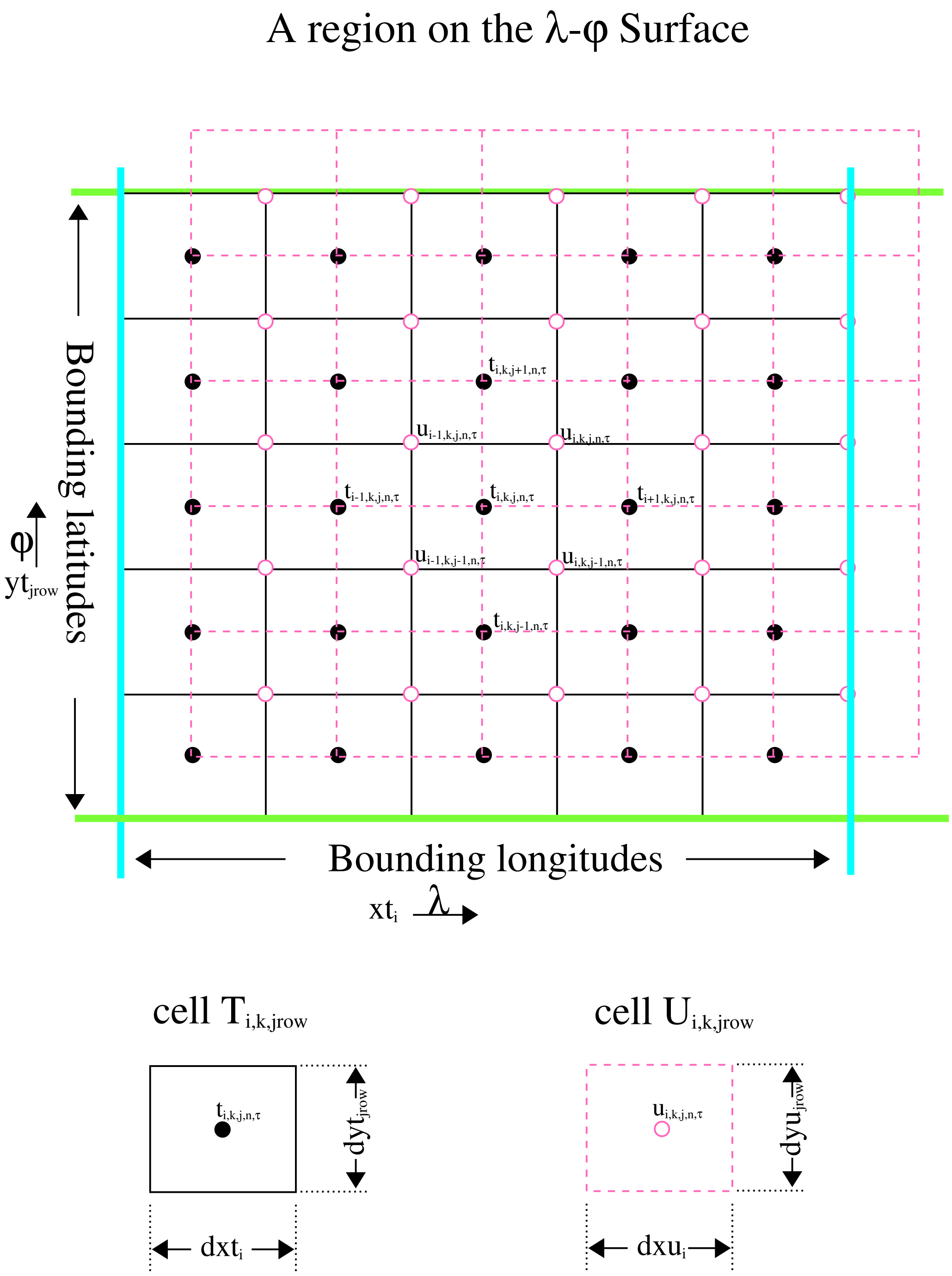


Figure 7.1: Grid cell arrangement on a horizontal longitude-latitude surface.

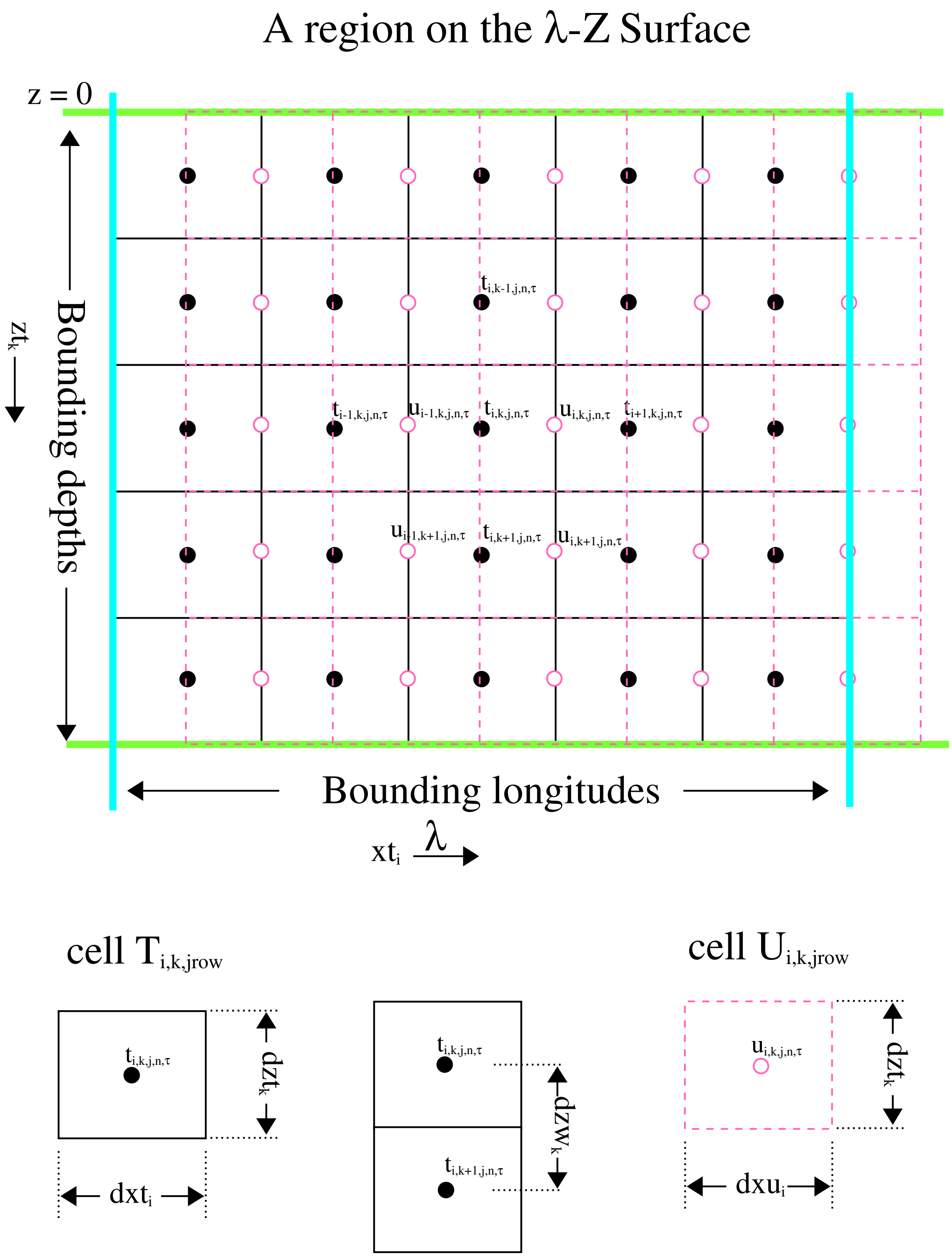


Figure 7.2: Grid cell arrangement on a vertical longitude-depth surface.

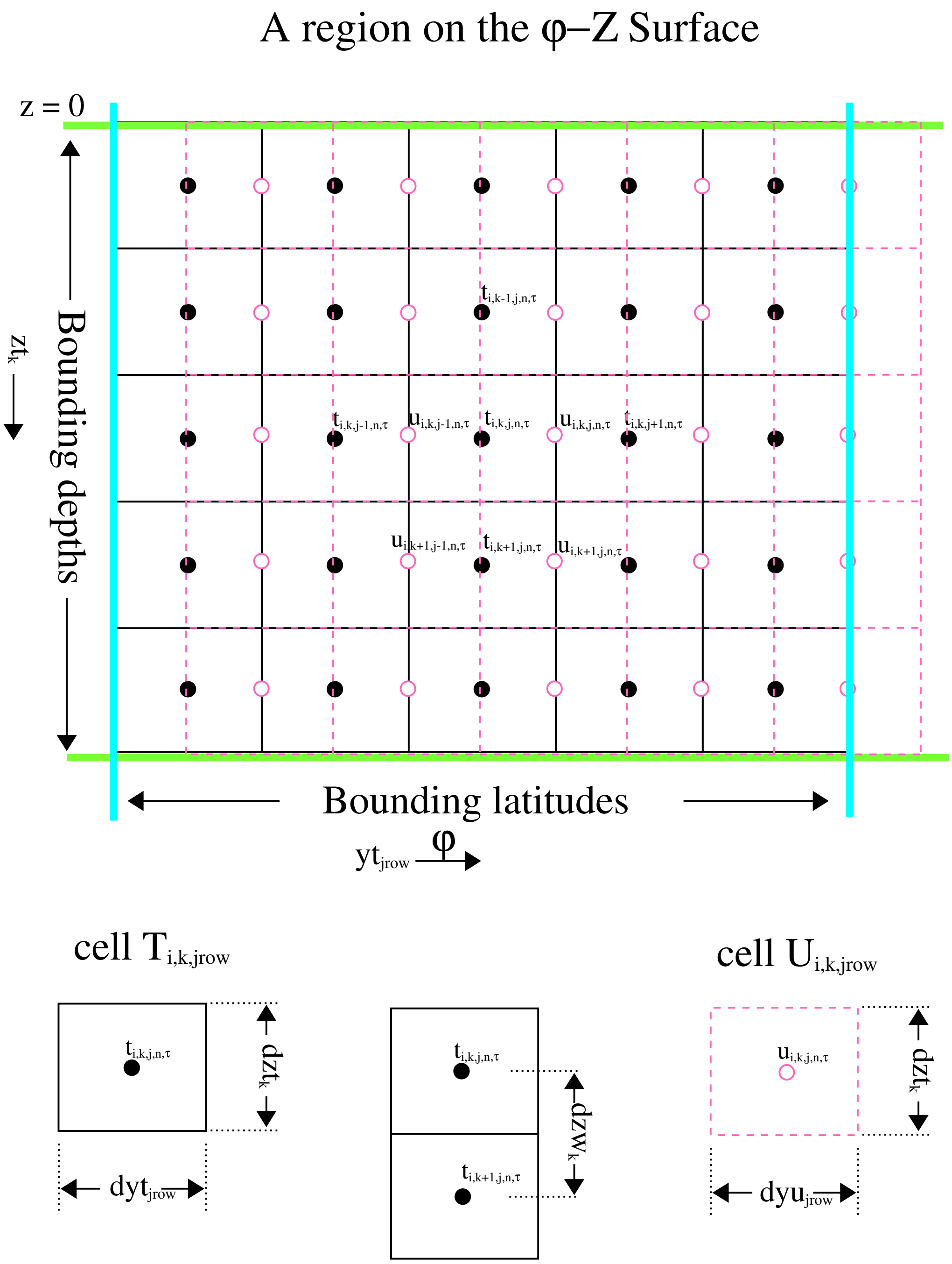


Figure 7.3: Grid cell arrangement on a vertical latitude-depth surface.

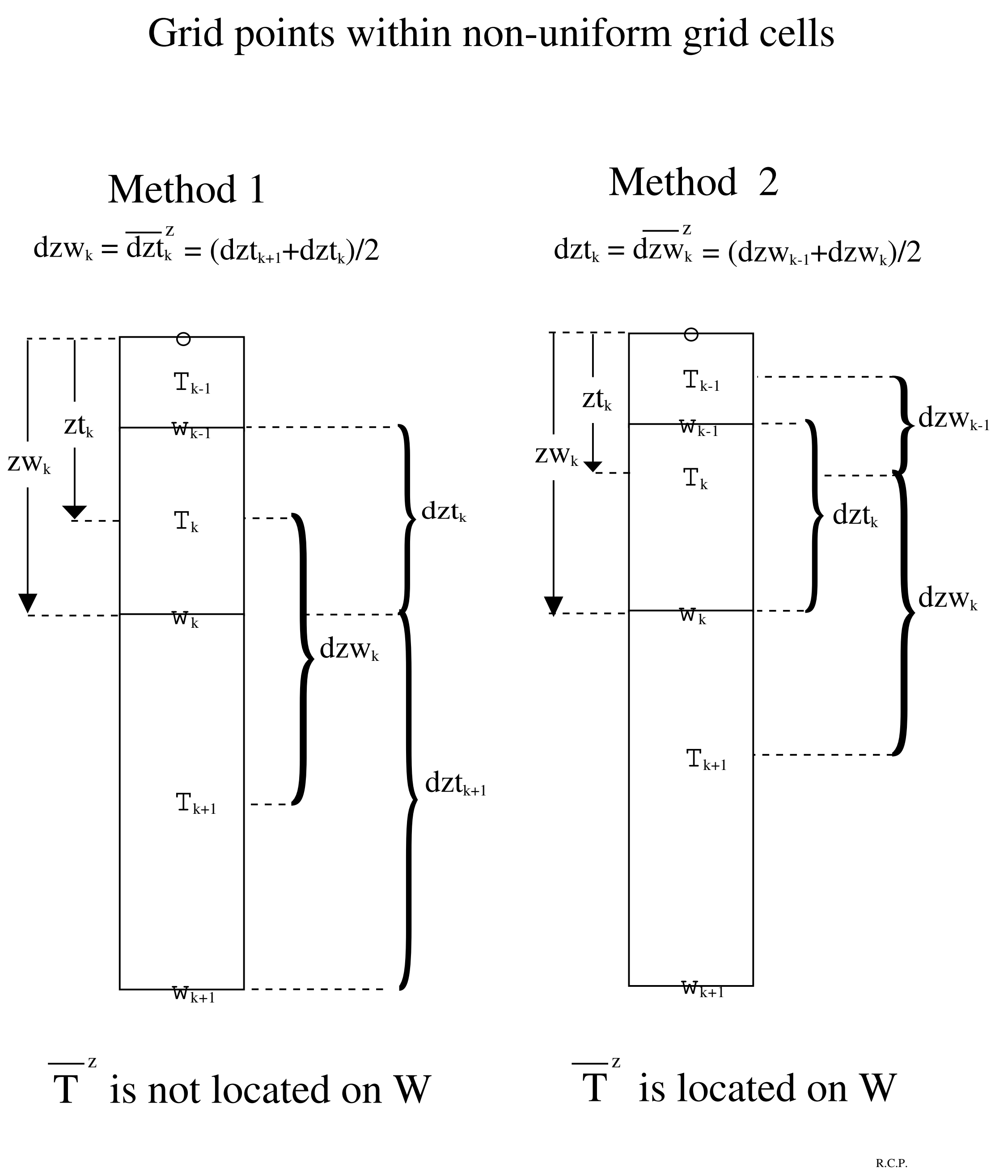


Figure 7.4: Comparison of two grid cell construction methods applied in the vertical dimension.

Chapter 8

Grid Rotation

A grid rotation is one of the simplest types of grid transformations that can be applied to a spherically gridded model. This transform is particularly useful for studies of high latitude oceans where the convergence of lines of longitude may limit time steps or where the ocean contains a pole (as in the Arctic). The idea is to define a new grid in which the area of interest is far from the grid poles. In limited domain models, the pole can be rotated outside of the domain. For global models, one possibility is rotate the North Pole to (40° W,78° N) which puts the North pole in Greenland and keeps the South pole in Antarctica. Other uses include rotating the grid to match the angle of a coastline or to provide more flexibility in specifying lateral boundary conditions.

Option *rot_grid* modifies the Coriolis term to handle a rotated model grid which is specified using three Euler angles for solid body rotation. The Euler angles are computed by defining the geographic latitude and longitude of the rotated north pole and a point on the prime meridian as described below.

To make this option easier to use, several rotation routines are provided within module *rotation.F*. The driver may be used to help define the rotation, write a file of geographic latitudes and test the rotation with idealized data. Other routines demonstrate how to interpolate scalar and vector data from a geographically gridded data set, to a rotated model grid. To run the driver, use the script *run_rotation*.

8.1 Defining the rotation

Any spherical grid rotation can be specified by defining three solid body rotations. The angles which define the rotations are usually referred to as Euler angles (see "Classical Mechanics" by Goldstein, 1950 or a similar text). First, define the Z axis to be through the poles such that the X-Y plane defines the equator and the X axis runs through the prime meridian. In the routines (in rotation.F), the rotation angles are called *phir*, *thetar* and *psir*. The angle *phir* is defined as a rotation about the original Z axis. Angle *thetar* is defined as a rotation about the new X axis (after the first rotation) and angle *phir* is defined as a rotation about the final Z axis.

It is helpful to have a globe to look at when thinking about this. Imagine that the globe has a clear sphere surrounding it, with only grid lines of latitude and longitude. By moving the outer sphere, the grid poles can be moved to line up with different points on the globe. Once the new poles are located, two of the rotation angles can be defined as follows. The definition for *phir* is 90 degrees minus the geographic longitude of the new north pole. This rotates the Y axis under the new pole. To move the Z axis down, *thetar* is defined to be 90 degrees minus the geographic latitude of the new north pole. This places the original Z axis though the new north pole position.

To completely define the grid, a third rotation about the new Z axis, must be specified. The rotated grid longitude of any point on the geographic grid is still undefined. To specify this last rotation, choose a point on the geographic grid (the globe) to locate the rotated grid's prime meridian. Set angle *psir* to zero and calculate the longitude of this point on the rotated grid. This longitude is the final angle *psir*, the angle needed to rotate the point back to the prime meridian. The definition of *psir* is usually not very important since the new grid longitude is arbitrary, but it does make a difference in defining exactly where the new grid starts. This may be important if it is desirable to line up grids for nesting. It may appear that all of the angle definitions are of the opposite sign to what they should be, but this comes from thinking about rotating the axes rather than rotating the rigid body.

Generally, the idea is to move the poles so that they are 90 degrees away from the area of interest. For example, to set up a model with an equatorial grid over the Arctic and North Atlantic, the rotated grid north pole could be positioned at 0 N, 110 W and a prime meridian point at 0 N, O E. This defines a grid rotation in which the new grid equator is along the 20 W and 160 E meridians. The rotated grid longitude is east, north of the geographic equator and west to the south. On the rotated grid, North America is in the north and Europe in the south, and the geographic north pole is at 0 N, 90 E. It is more difficult if you want to specify an arbitrary grid rotation, but usually a few trials is enough to locate the necessary pole position.

8.2 Rotating Scalars and Vectors

Code changes to mom are limited to modifying the calculation and dimension of the Coriolis variable *cori*. Running the rotation driver will create the latitude data file needed to calculate the correct Coriolis terms, but the creation of all other data files is left to the researcher. The subroutine *rot_intrp_sclr* may be used to interpolate scalars (such as depths, surface tracers, etc.) while subroutine *rot_intrp_vctr* can be used to interpolate and correct the angles for vectors (such as wind stress).

8.3 Considerations

Although additional execution time used by this option is negligible, the option may complicate subsequent analysis by the researcher. Rotated model results can be interpolated back to the geographic grid for comparison, but this may involve recalculating many diagnostics (such as zonal integrals) since all diagnostics within MOM 2 are only computed on the rotated model grid.

Section 15.6.1 contributed by
Michael Eby
eby@uvic.ca

Chapter 9

Topography and geometry

As described in Chapter 7, the model domain is filled with a rectangular arrangement of T cells and U cells. In general, some of these grid cells will be land cells and others will be ocean cells. Geometry and topography are implemented as an array of integers $kmt_{i,jrow}$ indicating the number of ocean cells stacked vertically between the surface and ocean bottom for each longitude and latitude coordinate xt_i and yt_{jrow} on the T grid. On the U grid, there is a corresponding field of integers $kmu_{i,jrow}$ derived as the minimum of the four surrounding $kmt_{i,jrow}$ values.

$$kmu_{i,jrow} = min(kmt_{i,jrow}, kmt_{i+1,jrow}, kmt_{i,jrow+1}, kmt_{i+1,jrow+1}) \tag{9.1}$$

In continental areas and regions where land rises above the ocean surface, $kmt_{i,jrow} = 0$. Vertical levels are indexed from the uppermost at $k = 1$ to the bottom of the domain at $k = km$. Beneath the ocean surface, land cells exist where $k > kmt_{i,jrow}$ on the T grid and where $k > kmu_{i,jrow}$on the U grid. The depth from the ocean surface to the bottom of the ocean is defined at the longitude and latitude of U cells as

$$H_{i,jrow} = zw_{k=kmu_{i,jrow}} \tag{9.2}$$

where zw_k is the depth of the bottom of the kth vertical level as given by module *grids*. Although module *topog* may be executed as part of MOM 2, designing $kmt_{i,jrow}$ by executing MOM 2 is similar to using a sledge hammer to work with tacks. The preferred method is to do the construction by executing script *run_topog* from a MOM_UPDATES directory as described for module *grids* in Section 7.3. Script *run_topog* will execute module *topog* in a stand alone mode by enabling option *drive_topog*. Once topography and geometry are judged satisfactory, module *topog* along with the resulting topography and geometry can be used from MOM 2 during model execution.

9.1 Constructing the KMT field

At each T cell longitude and latitude index $(i, jrow)$, ocean depth is discretized into the number of vertical grid cell thicknesses (levels) that most nearly approximates the ocean depth. The resulting field of model levels is the base $kmt_{i,jrow}$ field. To specify geometry and topography, one (and only one) of the options described below must be enabled:

- *rectangular_box* constructs a flat bottomed rectangular box with $\mathrm{kmt}_{i,jrow} = km$ (deepest level) for all interior points on the grid ($i = 2, imt-1$ and $jrow = 2, jmt-1$) while setting

$\text{kmt}_{i,jrow} = 0$ on all boundary cells. Enabling option *cyclic* turns the box into a zonally re-entrant channel.

- *idealized_kmt* constructs an idealized version of the earth's geometry. Continental features are very coarse[1] but map onto whatever grid resolution is specified by module *grids*. They are built with the aid of subroutine *setkmt* which approximates continental shapes by filling in trapezoidal areas of $kmt_{i,jrow}$ with *zero*. The bottom topography has a sinusoidal variation which is totally unrealistic. It is intended only to provide a test for the numerics of MOM 2. Since this option generates geometry and topography internally, no external data is required and therefore the DATABASE (explained in Chapter 19) is not needed. Typically, this option is useful when researching idealized geometries and topographies since arbitrary ones can be easily constructed. Also, this is useful when porting MOM 2 to other computer platforms since no data files need be considered.

- *scripps_kmt* reads the *scripps.top* 1° x 1° topography file (from the MOM 2 DATABASE explained in Chapter 19) and interpolates to $kmt_{i,jrow}$ for the grid defined by module *grids*. Subroutine *scripp* within module *topog* makes an educated guess as to whether resolution specified by module *grids* is coarser or finer than the native Scripps resolution. If finer, it does a linear interpolation from Scripps data, otherwise it uses area averaging of Scripps data to estimate the value of $kmt_{i,jrow}$ for the resolution required by module *grids*.

- *etopo_kmt* reads the file *ETOPO5.NGDCunformat_ieee* which is a 1/12° x 1/12° topography dataset. It may be purchased from the Marine Geology and Geophysics Division of the National Geophysical Data Center and is not included in the MOM 2 DATABASE. Depths are interpolated to a $kmt_{i,jrow}$ field for the grid defined by module *grids*. Subroutine *etopo* within module *topog* does the interpolation. If model resolution is finer than 1/12°, a linear interpolation from the dataset is used, otherwise area averaging over the dataset is used to estimate the value of $kmt_{i,jrow}$ for the resolution required by module *grids*.

- *read_my_kmt* allows importing $kmt_{i,jrow}$ fields into MOM 2 which have been exported from module *topog* using option *write_my_kmt*. Any $kmt_{i,jrow}$ field may be exported from module *topog*. The purpose of importing is to provide a hook for reading $kmt_{i,jrow}$ fields which have been constructed outside the MOM 2 environment. However, imported $kmt_{i,jrow}$ fields are subjected to the same consistency tests for violations as are constructed $kmt_{i,jrow}$ fields. It's a good idea to verify checksums when importing $kmt_{i,jrow}$ fields to make sure that something wasn't inadvertently changed. Importing and exporting is also useful with larger topographic datasets such as the 1/12° ETOPO5. Bringing these gigantic datasets into model runs can adversely affect model turn around time. It may be better to export the resulting $kmt_{i,jrow}$ field, then import it into a model execution instead of using ETOPO5 within the model.

Note that the $kmt_{i,jrow}$ field only needs to be constructed or imported into the model once at the beginning or time of initial conditions. After that, it is incorporated as part of the restart file *restart.dta*. Regardless of how $kmt_{i,jrow}$ is generated, module *topog* produces a checksum which can be used to verify that the $kmt_{i,jrow}$ field produced by script *run_topog* is the one being used by MOM 2.

[1] Their scale is about 10 degrees in latitude and longitude.

9.2 Modifications to KMT

The base $kmt_{i,jrow}$ field constructed above may have problems as illustrated in Figure 9.1. These include perimeter overlaps, minimum depth violations, and potentially troublesome areas such as isolated cells. In addition, the researcher may wish to modify the $kmt_{i,jrow}$ field for various reasons and these modifications may themselves introduce problems. All problems are remedied by iterating over a sequence of steps using guidance given in the form of options. Iteration is required because changes made in any step may affect other steps. Usually, after a few iterations, all changes are consistent. The steps are:

- If option *fill_isolated_cells* is enabled, isolated T cells are made into land cells. Isolated T cells are deeper than nearest neighbor T cells. Think of them as potholes. Also affected are T cells where the four surrounding U cells are land cells. They cannot communicate with horizontal neighbors through advection because all surrounding velocities are zero. MOM 2 will run with these conditions, but sometimes they cause problems.

- Specific researcher changes to the $kmt_{i,jrow}$ field are added. These may take the form of hard wiring $kmt_{i,jrow}$ to specific values for any number of reasons. They should be put in the USER INPUT section of module *topog*. Setting large areas of $kmt_{i,jrow}$ may be handled using routine *setkmt* as is done when generating the idealized dataset.

- Limit the minimum number of vertical levels in the ocean to parameter *kmt_min = 2*. This parameter may be increased but not decreased. There is some choice for altering $kmt_{i,jrow}$ to meet this condition. The choice is made by enabling one of the following options:

  - *fill_shallow* makes land cells where there are fewer than *kmt_min* vertical levels
  - *deepen_shallow* sets $kmt_{i,jrow}$ = *kmt_min* where there are fewer than *kmt_min* vertical levels
  - *round_shallow* sets $kmt_{i,jrow}$ to either 0 or *kmt_min* depending on which is closest

- Search for perimeter violations between land masses. Basically, there must be at least two T cells separating distinct land masses. If not, there is ambiguity in defining a stream function. Subroutine *isleperim* does this analysis by finding all land masses and their ocean perimeter cells. Violations can be corrected according to one of the following options:

  - *fill_perimeter_violations* builds a land bridge to connect two land masses by setting $\text{kmt}_{i,jrow} = 0$ at the first T cell discovered to be in the perimeter of both land masses. Each land bridge reduces the number of islands by one.(*This is the default action.*)
  - *widen_perimeter_violations* removes land cells from one of the land masses until a separation between land masses of at least two T cells is achieved. The values of $kmt_{i,jrow}$ on these newly created ocean cells are set to the average of the values of $kmt_{i,jrow}$ on adjacent ocean cells.

It sometimes happens that steep topographic gradients as indicated in Killworth (1987) are another source of problems. Because this requires knowledge of time step length and mixing coefficients, these problems are not detected in module *topog*, and analysis is left to MOM 2. In many cases the instability that results can be controlled by changing time step length or diffusion coefficients. In extreme cases, consideration might be given to smoothing[2] the entire topography field before interpolating to $kmt_{i,jrow}$.

[2] Not an option as of this writing. If a smoothing is implemented be aware that coastlines may change.

9.3 Viewing results

The recommended way of viewing the resulting topography, $kmt_{i,jrow}$ field, and f/H field is to save these results using option *topog_netcdf*. This is described in Section 18.2.20. A good way to visualize results is with Ferret which is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL (email: ferret@pmel.noaa.gov URL: http://www.pmel.noaa.gov/ferret/home.html).

As an alternative, a map of the resulting $kmt_{i,jrow}$ field is printed out as as part of the results from executing module *topog*. For gigantic domains, it may be desirable to limit the printed output and directions for doing this are given in the output file.

9.4 Fine tuning *kmt*

Although many of the problems with geometry and topography can be handled in a routine fashion, sometimes it is desirable to interact with the modification process. At one point, module *topog* contained interactive editors but these accounted for approximately 3500 lines of code which was hardly justifiable. These editors along with intermediate files for storing changes have been dropped. The current approach is to save output with option *topog_netcdf* which involves relatively little code and view the results with Ferret (described above). Alternatively, the $kmt_{i,jrow}$ field is written to the results file and can be viewed with an editor. Regions of $kmt_{i,jrow}$ needing changes are easily identified and changes can be hard wired into the USER INPUT section of module *topog*. Large areas of changes (like blocking out areas) can be handled easily using calls to subroutine *setkmt*. Look at the way idealized topography is built within module *topog*.

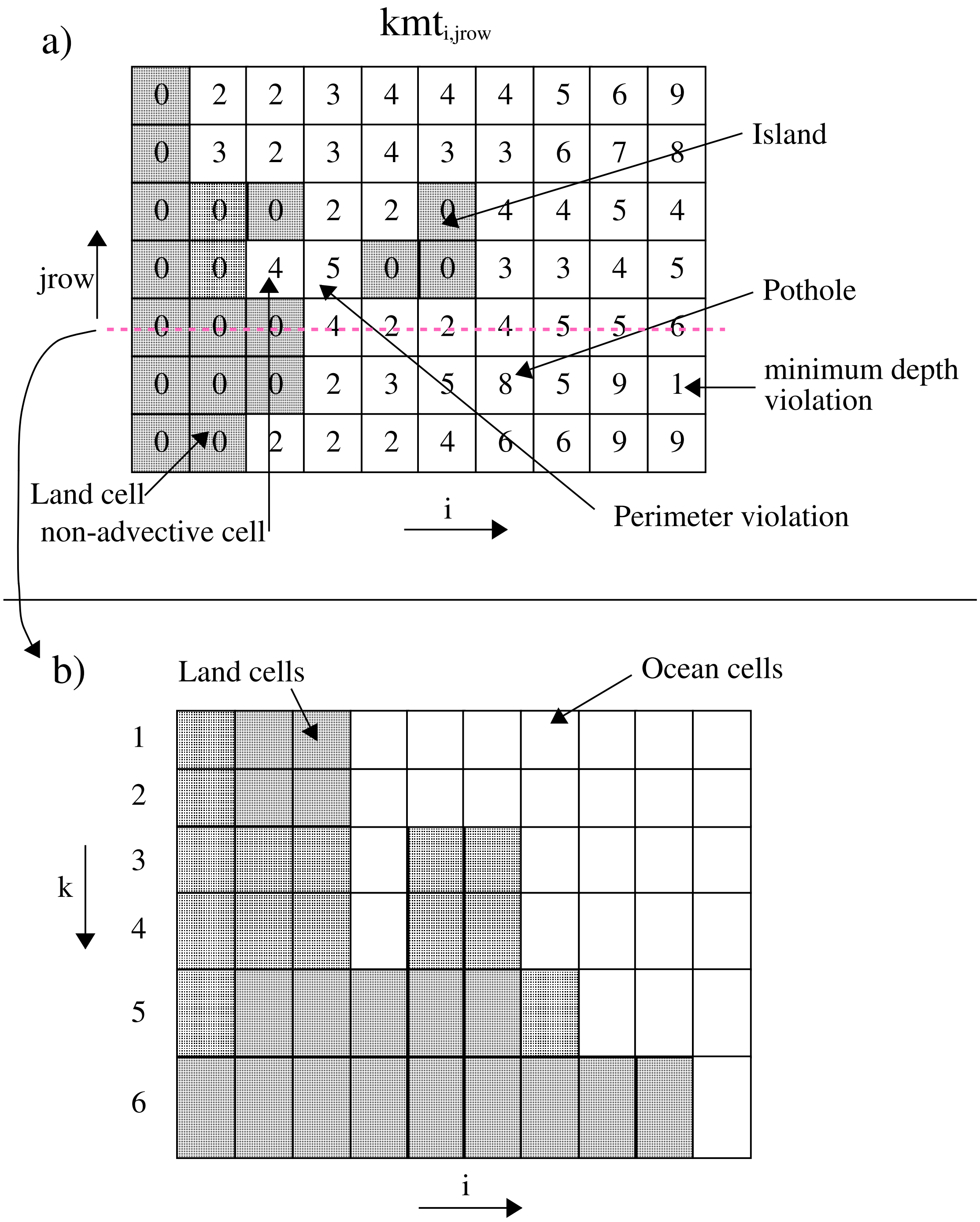


Figure 9.1: a) A portion of the *kmt* field indicating ocean and land areas with potential problems. b) Topography along the slice indicated in a)

Chapter 10

Generalized Surface Boundary Condition Interface

The ocean is driven by surface boundary conditions[1] which come from the atmosphere and the atmosphere is in turn driven by surface boundary conditions[2] which come from the ocean. The difficulty in understanding the coupled system is that each component influences the other. For purposes of MOM 2, the atmosphere may be thought of as a hierarchy of models ranging from a simple idealized wind dataset fixed in time through a complicated atmospheric GCM[3]. Regardless of which is used, MOM 2 is structured to accommodate both extremes as well as atmospheres of intermediate complexity as discussed in the following sections.

10.1 Coupling to atmospheric models

In this section, the general case will be considered where the atmosphere is assumed to be a GCM with domain and resolution differing from that of MOM 2 and two-way coupling between MOM 2 and the atmosphere model will be allowed. Option *coupled* configures MOM 2 for this general case and the test case prototype is described in Sections 19.6 and 19.7 as CASE=3.

A flowchart of *driver*[4] is given in Figure 10.2. It begins by calling subroutine *setocn*[5] to perform initializations for *mom*[6]. Included in the list are such things as initializing variables, reading namelists to over-ride defaults, setting up the grid, topography, initial conditions, region masks, etc. In short, everything that needs to be done only once per model execution. Following this, a call is made to subroutine *setatm*[7] which completes whatever setup is required by *atmos*[8].

At this point, the integration is ready to begin. Integration time is divided into a number of equal length time segments[9] which determine the coupling period. In practice, this interval should always be chosen short enough to adequately resolve time scales of coupled interaction. Typically this value would be one day[10]. Within the segment loop, *atmos* and *mom* are

[1]Wind, rain, heatflux, etc.

[2]Primarily SST.

[3]General circulation model.

[4]Contained within file driver.F. This is the main program for MOM 2.

[5]Contained within file setocn.F.

[6]Contained within file mom.F. This is the subroutine that does the time integration for the ocean model.

[7]Contained within file setatm.F in the MOM_2/SBC directories.

[8]Contained within file atmos.F in the SBC directories. This is the subroutine that does the time integration for the atmosphere model.

[9]The length of one time segment should be divisible by the length on one ocean time step to allow an integral number of calls to subroutine *mom*. The same holds true for the atmosphere time step.

[10]If the diurnal cycle is included, the coupling period needs to be reduced to allow adequately resolution in

alternately integrated for each time segment while holding surface boundary conditions fixed. Note that this may require multiple calls to each model. In Figure 10.2, the loop variable *ntspas* stands for the number of time steps per atmosphere segment and the loop variable *ntspos* stands for the number of time steps per ocean segment. Products of each atmosphere segment include surface boundary conditions[11] for the ocean averaged over that segment. These are held fixed and applied to *mom* while it integrates over the same segment. Products of integrating *mom* include surface boundary conditions for the atmosphere which are also averaged over this segment. Subsequently, they are held fixed and applied to *atmos* on the following segment. This process is represented schematically as a function of time in Figure 10.1 and continues until all time segments are completed. Using asynchronous time segments[12] is possible with a small code modification but this is left to the researcher.

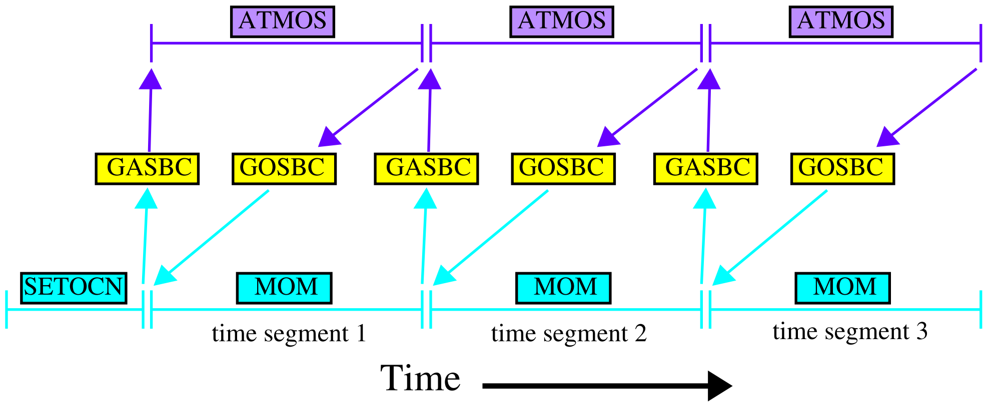


Figure 10.1: Schematic of two way coupling between an atmosphere model (ATMOS) and an ocean model (MOM) showing time segments.

In MOM 2 and in the descriptions that follow, index *j* refers to any variable dimensioned by the number of rows in the memory window and index *jrow* refers to any variable dimensioned by the total number of latitude rows. They are related by an offset *jrow* = *j* + *joff* which indicates how far the memory window has moved northward. Refer to Section 5.2 and also 3.3.2 with Figure 3.4 for a more complete description.

10.1.1 GASBC

Getting surface boundary conditions for the atmosphere model defined on the atmosphere grid is the purpose of subroutine *gasbc*[13] which is an acronym for *get atmosphere surface boundary conditions*. All of the surface boundary conditions are two dimensional fields defined in longitude and latitude at model grid points. In the ocean, all quantities which are to be used as surface boundary conditions for the atmosphere are defined on the $T_{i,jrow}$ grid[14] in array $sbcocn_{i,jrow,m}$

time.

[11]The first set of surface boundary conditions for the atmosphere are products of *setocn*.

[12]Where, for example, an ocean segment is much longer than an atmosphere segment. This assumes the coupled system is linear with one equilibrium. Exercise caution if contemplating this!

[13]Contained in file gasbc.F.

[14]At grid locations given by xt_i and yt_{jrow}.

where subscript m refers to the ordering described in Section 10.3. Typically, only SST is needed but it may be desirable to let the atmosphere sense that the ocean surface is moving in which case the horizontal velocity components might also be used (Pacanowski 1987). In any event, these quantities are accumulated in $sbcocn_{i,jrow,m}$ within MOM 2 and averaged at the end of each time segment. Basically what needs to be done in *gasbc* is to interpolate the time averaged $sbcocn_{i,jrow,m}$ fields to $sbcatm_{i',j',m}$ which is an array of the same surface boundary conditions except defined on the atmosphere boundary condition grid[15] $\mathrm{A}_{i',j'}$. One of the duties of *setatm* is to define $\mathrm{A}_{i',j'}$ as the atmospheric boundary condition grid which includes extra boundary points along the borders to facilitate these interpolations.

SST outside Ocean domain

There is a complication if the ocean is of limited extent: SST must be prescribed outside the ocean domain as a surface boundary condition for the atmosphere on the $\mathrm{A}_{i',j'}$ grid. To accommodate this, the ocean domain $T_{i,jrow}$ must be known in terms of $\mathrm{A}_{i',j'}$ and a buffer or blending zone must be established. Within this zone, SST from *mom* is blended with the prescribed SST outside the ocean domain as indicated in Figure 10.3. As a simplification, SST is prescribed outside the domain of *mom* as a constant which is obviously unrealistic and should be prescribed by the researcher as a function of space and time appropriately along with the width of the blending zone. SST within the blending zone is a linear interpolation between the two regions.

Interpolations to atmos grid

Each surface boundary condition is interpolated one at a time using essentially the following steps:

- For limited ocean domains, prescribe SST in $sbcatm_{i',j'}$ outside the *mom* domain as described above. When ocean and atmosphere domains match, this is not necessary.

- Extrapolate $sbcocn_{i,jrow,m}$ into land areas on the *mom* grid. This is done by solving $\nabla^2(SST_{i,jrow}) = 0$ over land cells using ocean $SST_{i,jrow}$as boundary conditions[16]. The idea is to get reasonable $SST_{i,jrow}$in land adjacent to coastlines but not necessarily to produce accurate $SST_{i,jrow}$in the middle of continents. Where land and ocean areas on the ocean and atmosphere grids are mismatched, this extrapolation will ameliorate the sensing of erroneous $SST_{i,jrow}$by the atmosphere. Such situations are common when coupling spectral atmospheres to MOM 2 which has an Arakawa B-grid.

- Interpolate $sbcocn_{i,jrow,m}$ to $sbcatm_{i',j',m}$. The ocean resolution is typically higher than that of the atmosphere because the Rossby radius is larger in the atmosphere. To prevent aliasing, the interpolation is an area average of $sbcocn_{i,jrow,m}$ for those $i, jrow$ cells which fall within each $\mathrm{A}_{i',j'}$ grid cell[17]. The interpolation is carried out using subroutine *ftc*[18] which can easily be replaced by subroutine *ctf*[19] (or something else) if ocean resolution is less than atmosphere resolution.

- Set cyclic conditions on $sbcatm_{i',j',m}$ and convert to units expected by the atmosphere.

[15] Grid locations given by $abcgx_{i'}$ and $abcgy_{j'}$.

[16] This method is arbitrary. However, when solving this equation iteratively, it is important not to zero out SST from previous solutions over land areas. Their purpose is to act as a good initial guess to limit the iterations needed for subsequent solutions.

[17] Partial ocean cells are accounted for to conserve the interpolated value.

[18] Acronym for fine to coarse resolution. It is an interpolation utility in module *util*.

[19] Acronym for coarse to fine resolution. It is an interpolation utility in module *util*.

- Compute global mean of $sbcatm_{i',j',m}$

The structure of *gasbc* is arranged such that components can be removed or replaced with more appropriate ones if desired.

Caveat

In the process of constructing $sbcatm_{i',j',m}$ no attempt has been made at removing small scale spatial features from the grid which could potentially be a source of noise for the atmosphere model.

10.1.2 GOSBC

Getting surface boundary conditions for the ocean model defined on the grid of *mom* is the purpose of subroutine *gosbc*[20] which is an acronym for *get ocean surface boundary conditions* . This is the counterpart of *gasbc*. All of the surface boundary conditions are two dimensional fields as described in Section 10.1.1. In the atmosphere, all quantities which are to be used as surface boundary conditions for MOM 2 are defined on the $A_{i',j'}$ grid in array $sbcatm_{i',j',m}$ where subscript m again refers to the ordering described in Section 10.3. Typically, they are quantities like windstress components, heatflux and precipitation minus evaporation. These must be accumulated in $sbcatm_{i',j',m}$ within the atmosphere model and averaged at the end of each time segment. Basically what needs to be done in *gosbc* is to interpolate the time averaged $sbcatm_{i',j',m}$ to $sbcocn_{i,jrow,m}$ which is defined on the ocean grid.

Interpolations to ocean grid

Unlike in Section 10.1.1, there are no complications since the ocean domain is assumed to be contained within the atmosphere domain as shown in Figure 10.4. Each surface boundary condition is interpolated one at a time with the essential steps being:

- Set cyclic conditions of $sbcatm_{i',j',m}$. This assumes a global atmosphere domain.

- Extrapolate $sbcatm_{i',j',m}$ into land areas on the atmosphere grid. This is done as in Section 10.1.1 by solving $\nabla^2\xi_{i',j'} = 0$ over land areas using values of $\xi_{i',j'}$ over non-land areas as boundaries where $\xi_{i',j'}$ represents windstress components, heatflux, etc[21]. The idea, as on the ocean grid, is to get reasonable $\xi_{i',j'}$ adjacent to coastlines while not trying to produce accurate $\xi_{i',j'}$ in the middle of continents. Where land and ocean areas on the ocean and atmosphere grids are mismatched, this will ameliorate the sensing of erroneous $\xi_{i',j'}$ by the ocean.

- Interpolate $sbcatm_{i',j',m}$ to $sbcocn_{i,jrow,m}$. Since ocean resolution is typically higher than that of the atmosphere due to the difference in Rossby radius scales, the interpolation is linear. The interpolation is carried out using subroutine *ctf* from file *util.F*. Note that linear interpolation does not exactly conserve the quantity being interpolated. However, linear interpolation has been used without causing noticable drift in coupled integrations (without flux correction) of up to 20 years at GFDL.

  For integrations simulating thousands of years, linear interpolation may not be good enough. To conserve fluxes exactly, one possibility (although drastic) is to design the

[20] Contained within file gosbc.F.

[21] This method is arbitrary. However, when solving this equation iteratively, it is important not to zero out ξ from previous solutions over land areas. Their purpose is to act as a good initial guess to limit the iterations needed for subsequent solutions.

grids such that an integral number of ocean cells overlay each atmospheric grid cell. Then the quantity to be interpolated can simply be broadcast from each atmospheric cell to all underlying ocean cells. However, this is really not necessary. Assume that the value of the quantity being interpolated is constant over the entire atmospheric grid cell. For exact conservation, the value in each ocean cell is just an integral of the atmospheric quantity over the area of the ocean cell. It would be a matter of writing a subroutine to do it and substituting this subroutine for the call to *ctf*.

- Set cyclic conditions on $sbcocn_{i,jrow,m}$ and convert to units expected by the ocean. MOM 2 expects units of *cgs*.

- Compute global mean of $sbcocn_{i,jrow,m}$

Caveat

In the process of constructing $sbcocn_{i,jrow,m}$ no attempt has been made at removing small scale spatial features from the grid which could potentially be a source of noise for the ocean model.

10.2 Coupling to datasets

In Section 10.1, the general case of two way coupling to an atmospheric GCM was considered. It is also useful to drive *mom* with atmospheric datasets representing simpler idealized atmospheres. One problem with doing this is that datasets act as infinite reservoirs of heat capable of masking shortcomings in parameterizations which only become apparent when two way coupling is allowed. Nevertheless, driving ocean models with surface boundary conditions derived from datasets is useful.

These datasets can be thought of as simple atmospheres prepared *a priori* such that data is defined at grid locations expected by *mom* for surface boundary conditions. All spatial interpolations are done beforehand, as described in Section 19.6. In this case, subroutines *gasbc* and *goabc* are bypassed because spatial interpolation to the *mom* grid is not needed. Also, the length of a time segment reduces to the length of one ocean time step. In general, the datasets contain either time mean conditions or time varying conditions and there are three options which configure these types of surface boundary conditions for *mom*:

- *simple_sbc* allows for the simplest of atmospheric datasets. All surface boundary conditions are a function only of latitude. No time dependence here although it could easily be incorporated. The test case prototype is described in Sections 19.6 and 19.7 as CASE=0. In this case, there is no dataset on disk and $sbcocn_{i,jrow,m}$ is not needed since all surface boundary conditions are generated internally. This is an appropriate option for building forcing functions for idealized surface boundary conditions .

- *time_mean_sbc_data* uses an atmosphere dataset defined in SBC/TIME_MEAN. The test case prototype is described in Sections 19.6 and 19.7 as CASE=1. The dataset resides on disk and each surface boundary condition is a function of latitude and longitude. The dataset should be prepared for the grid in MOM 2 using scripts in PREP_DATA as described in Section 19.6. The surface boundary condition data is read into the surface boundary condition array $sbcocn_{i,jrow,m}$ once in *atmos* for all latitude rows where m gives the ordering of the boundary conditions as described in Section 10.3. On every time step, data from $sbcocn_{i,jrow,m}$ is loaded into the surface tracer flux array $stf_{i,j,n}$ and the surface momentum flux array $smf_{i,j,n}$. This is done in subroutine *setvbc* for rows $j = jsmw, jemw$ in the memory window[22].

- *time_varying_sbc_data* uses an atmosphere dataset defined in SBC/MONTHLY. The test case prototype is described in Sections 19.6 and 19.7 as CASE=2. The dataset resides on disk as a series of records. Each record is a climatological monthly mean surface boundary condition as a function of latitude and longitude. This dataset should be prepared for the grid in *mom* using scripts in PREP_DATA as described in Section 19.6. The complication is one of interpolating the monthly values to the model time in *mom* for each time step. Basically, the model time must be mapped into the dataset to find which two months (data records) straddle the current model time[23]. For the purpose of interpolation, months are defined at the center of their averaging periods. Both months are read into additional boundary condition arrays in *atmos* and used to interpolate to array $sbcocn_{i,jrow,m}$ for the current model time. When the model time crosses the midpoint of a month, the next month is read into a boundary condition array[24] and the process continues indefinitely assuming the dataset is specified as periodic. If it is not, *mom* will stop when the ocean integration time reaches the end of the dataset. Although data is read into the additional boundary condition arrays only when ocean model time crosses the mid month boundary, data is interpolated into $sbcocn_{i,jrow,m}$ on every timestep. There are four allowable methods for interpolating these datasets in time:

  Method=0 is for no interpolation; the average value is used for all times within the averaging period. This is the simplest interpolation. It preserves the integral over averaging periods, but is discontinuous at period boundaries.

  Method=1 is for linear interpolation between the middles of two adjacent averaging periods. It is continuous but does not preserve integrals for unequal averaging periods.

  Method=2 is for equal linear interpolation. It assumes that the value on the boundary between two adjacent averaging periods is the unweighted average of the two average values. Linearly interpolates between the midperiod and period boundary. It is continuous but does not preserve integral for unequal periods.

  Method=3 is for equal area (midperiod to midperiod) interpolation. It chooses a value for the boundary between two adjacent periods such that linear interpolation between the two midperiods and this value will preserve the integral midperiod to midperiod.

  To explore how time interpolation works in a very simple environment, one can execute script *run_timeinterp* which exercises *timeinterp*[25] as a stand alone program.

  As mentioned above, two additional boundary condition arrays are needed for each surface boundary condition. The memory increase may get excessive with large resolution models. In that case, option *minimize_sbc_memory* reduces these arrays from being dimensioned as (imt,jmt) to being dimensioned as (imt,jmw). The trade off is increased disk access which may be prohibitive if using rotating disk. If efficiency is an issue, asynchronous

[22] Refer to Chapters 1 and 5. In CASE=0, the surface boundary conditions are zonally averaged time means. Since they are only functions of latitude, array $sbcocn_{i,jrow,m}$ does not exist and surface boundary conditions are set directly into arrays $stf_{i,j,n}$ and $smf_{i,j,n}$.

[23] It should be noted that there is enough generality to accommodate datasets with other periods such as daily, hourly, etc and treat them as climatologies (periodic) or real data (non periodic). Also datasets with differing periods may be mixed. For example: climatological monthly SST may be used with hourly winds.

[24] The one which is no longer needed.

[25] Contained in file timeinterp.F. This is the subroutine that does interpolations in time.

reads with look ahead capability would be worth trying. This look ahead feature has not been implemented.

10.2.1 Bulk parameterizations

Section 10.2 describes three simple atmospheres requiring very little computation. In general, the atmosphere is sensitive to SST but not sea surface salinity. Since fresh water flux into the ocean is not known very accurately, it is reasonable to damp sea surface salinity back to climatological values on some Newtonian time scale for a surface boundary condition. As mentioned in Section 10.3, restoring SST and sea surface salinity to data can be done by enabling option *restorst*. Note that damping time scale and thickness may by set differently for each tracer. However, instead of restoring SST, the next simplest atmosphere in the hierarchy may be thought of as being parameterized with bulk formulae as given by the idealized version in Philander/Pacanowski (1986) or the more complete version in Rosati/Miyakoda (1988).

Although these bulk parameterizations are not included in MOM 2 as of this writing, they are easy to implement. The largest uncertainties are due to clouds. One must be cautious with this type of atmosphere since the global integral of heatflux into the ocean averaged over one seasonal cycle may be non-zero which will lead to a drift in the ocean heat content with time.

10.3 Surface boundary condition details

As mentioned in the previous sections, surface boundary conditions are contained within array $sbcocn_{i,jrow,m}$ defined on the *mom* grid. If option *coupled* is enabled, they are also contained within array $sbcatm_{i',j',m}$ defined on the atmosphere surface boundary condition grid $\mathrm{A}_{i',j'}$. If option *coupled* is not enabled, then array $sbcatm_{i',j',m}$ does not exist. The total number of surface boundary conditions *numsbc* is divided into *numosbc* for the ocean and *numasbc* for the atmosphere. The ordering of surface boundary conditions in both arrays is the same and is specified by a mapping array *mapsbc*. Why not just specify an order and forget about *mapsbc*? The reason is that array *mapsbc* allows new surface boundary conditions to be added or old ones to be removed from the list in a relatively easy way. The default ordering is given below: These five surface boundary conditions for the ocean come <u>from</u> the atmosphere

1. mapsbc(1) references $\vec{\tau} \cdot \hat{\lambda}$ which is the eastward windstress exerted on the ocean in units of $dynes/cm^2$. If the wind in the atmosphere model is positive (blowing towards the east) then the ocean and land exert a westward stress which decelerates the atmosphere. From the atmosphere point of view, this stress is negative. However, the ocean and land are being accelerated by the transfer of momentum through the boundary layer. Therefore the correct sign for the stress acting on the ocean is positive (towards the east).

2. mapsbc(2) references $\vec{\tau} \cdot \hat{\phi}$ which is the northward windstress exerted on the ocean in units of $dynes/cm^2$. A northward windstress is positive.

3. mapsbc(3) references heat flux in units of *langley/sec* where 1 $langley = 1\ cal/cm^2$. A positive heatflux means heat is being pumped into the ocean.

4. mapsbc(4) references a salinity flux into the ocean in units of $grams/cm^2/sec$. In MOM 2, the rigid lid approximation implies that ocean volume is constant[26]. Therefore, when it

[26] When using option *implicit_free_surface* the ocean volume is allowed to change. This allows for a fresh water flux to be used directly. This has not been implemented as of this writing. The recommendation is to set the salinity flux to zero and add fresh water flux directly to the free surface elevation.

rains, salt must change since ocean volume cannot. If the atmosphere model supplies a fresh water flux, this should be converted to a salt flux = -(P-E+R) $\cdot\rho_o\cdot$ S_{ref} where P-E+R represents a precipitation minus evaporation plus runoff rate in *cm/sec*, ρ_o is taken to be 1.035 gm/cm^3, and S_{ref} is a reference salinity in units of grams of salt per gram of water (units of *parts per part* such as 0.035). Depending upon the application of interest, S_{ref} may be a constant over the entire model domain or the locally predicted salinity of the uppermost model level. If the intent is for a global average P-E+R flux of zero to imply a zero trend in the ocean salt content, then S_{ref} must be chosen as a constant. A positive precipitation minus evaporation into the ocean means that fresh water is being added to the ocean which decreases the salinity. This implies that the salt flux is negative.

5. mapsbc(5) references solar short wave flux in units of *langley/sec* where 1 *langley* = 1 cal/cm^2. Normally this effect is included in the surface heat flux. However, solar short wave penetration into the ocean is a function of wavelength. The clear water case assumes energy partitions between two exponentials as follows: 58% of the energy decays with a 35 cm e-folding scale; 42% of the energy decays with a 23 m e-folding scale. If the thickness of the first ocean level $dzt_{k=1}$ = 50 meters, then shortwave penetration wouldn't matter. However, for $dzt_{k=1}$ = 10 meters, the effect can be significant and may be particularly noticeable in the summer hemisphere. See Paulson and Simpson (1977), Jerlov (1968) and Rosati (1988).

When option *restorst* is enabled, surface tracers are restored to prescribed data $t^{\star}_{i,j,n,time}$[27] using a Newtonian damping time scale *dampts* in units of days and a thickness *dampdz* in units of *cm*. Both are input through a namelist. Refer to Section 5.4 for information on namelist variables. Note that damping time scale and thickness may by set differently for each surface tracer. These are used to convert the Newtonian damping term into a surface tracer flux as described under option *restorst* in Section 15.9.7.
These four surface boundary conditions for the atmosphere come <u>from</u> the ocean.

1. mapsbc(6) references SST in units of degrees C.

2. mapsbc(7) references sea surface salinity which is typically used for damping surface salinity back to prescribed climatological values. This is used to compute an effective salt flux.

3. mapsbc(8) references the zonal component of the velocity from the first level in the vertical in units of *cm/sec*. Typically, the ocean surface is assumed to be rigid within atmospheric models. However, in certain regions surface velocities can exceed 50 *cm/sec* which is enough to effect bulk transfer estimates of evaporation and surface stress. The stress is related to the relative difference between wind speed and the ocean currents as in Pacanowski (1987).

4. mapsbc(9) references the meridional component of velocity from the first level in the vertical in units of *cm/sec*.

<u>Adding or removing surface boundary conditions</u>

[27] A product of PREP_DATA scripts operating on the DATABASE in CASE=1 and CASE=2. CASE=0 uses an idealized data generated internally as a function of latitude only. Interpolations to model time are explained under option *time_varying_sbc_data* in Section 10.2.

As an example, suppose one wanted the first six surface boundary conditions but not the rest. Use the UNIX command *grep numosbc *.h* to find *numosbc* and set parameters to $numosbc = 5$ and $numasbc = 1$. This reduces memory requirements and no other changes are necessary. However, suppose one wanted to remove the short wave boundary condition from the previous example. The following steps would be needed:

- *grep numosbc *.h* to find and set numosbc=4, numasbc=1
- *grep "mapsbc(5)" *.F* to find and remove references for mapsbc(5)
- *grep "mapsbc(6)" *.F* to change references for mapsbc(6) to mapsbc(5)

This is much cleaner than hard wiring numbers to specify particular surface boundary conditions throughout the code.

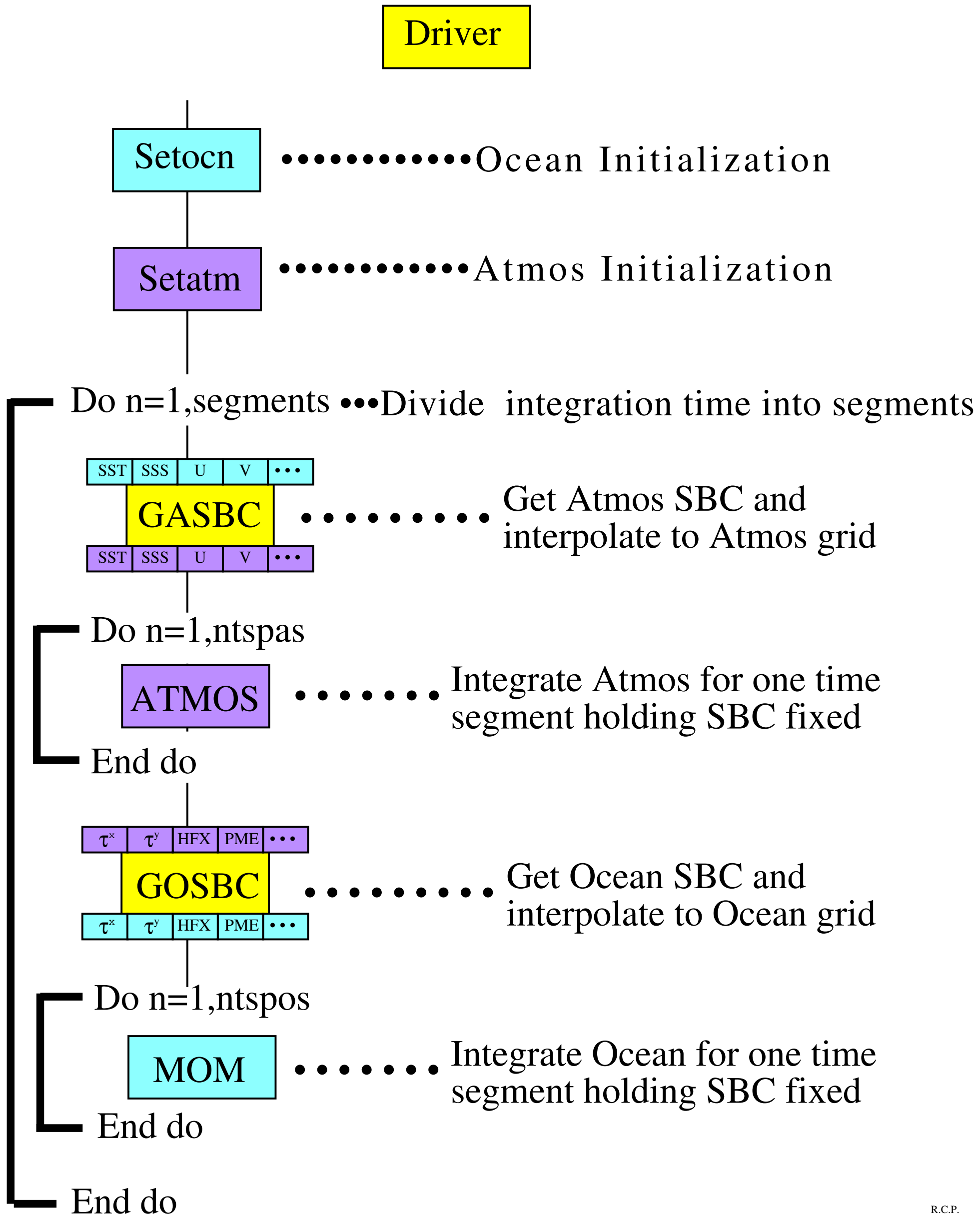


Figure 10.2: Flowchart for main program *driver.F* which controls surface boundary conditions in MOM 2

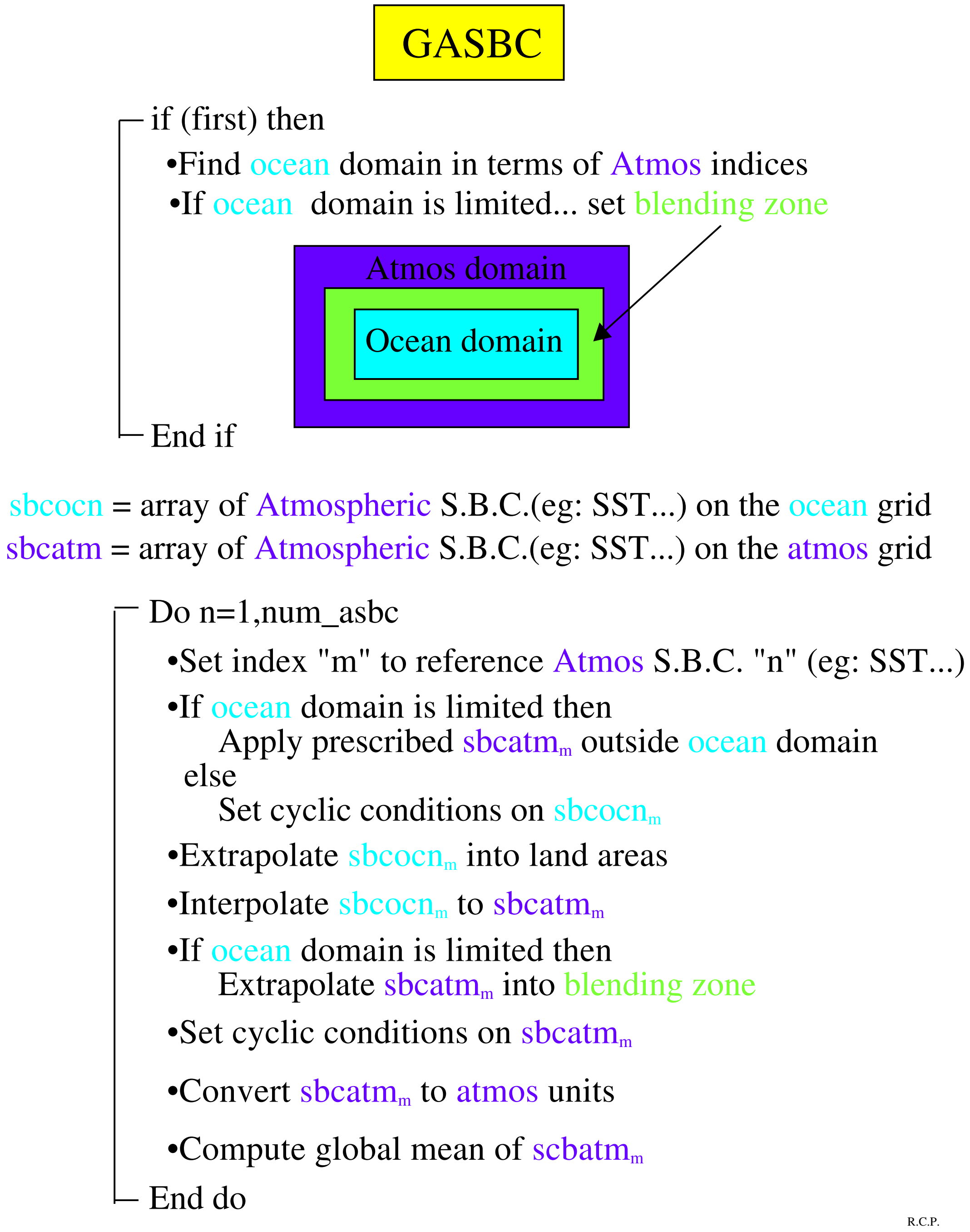


Figure 10.3: Flowchart for subroutine *gasbc.F*

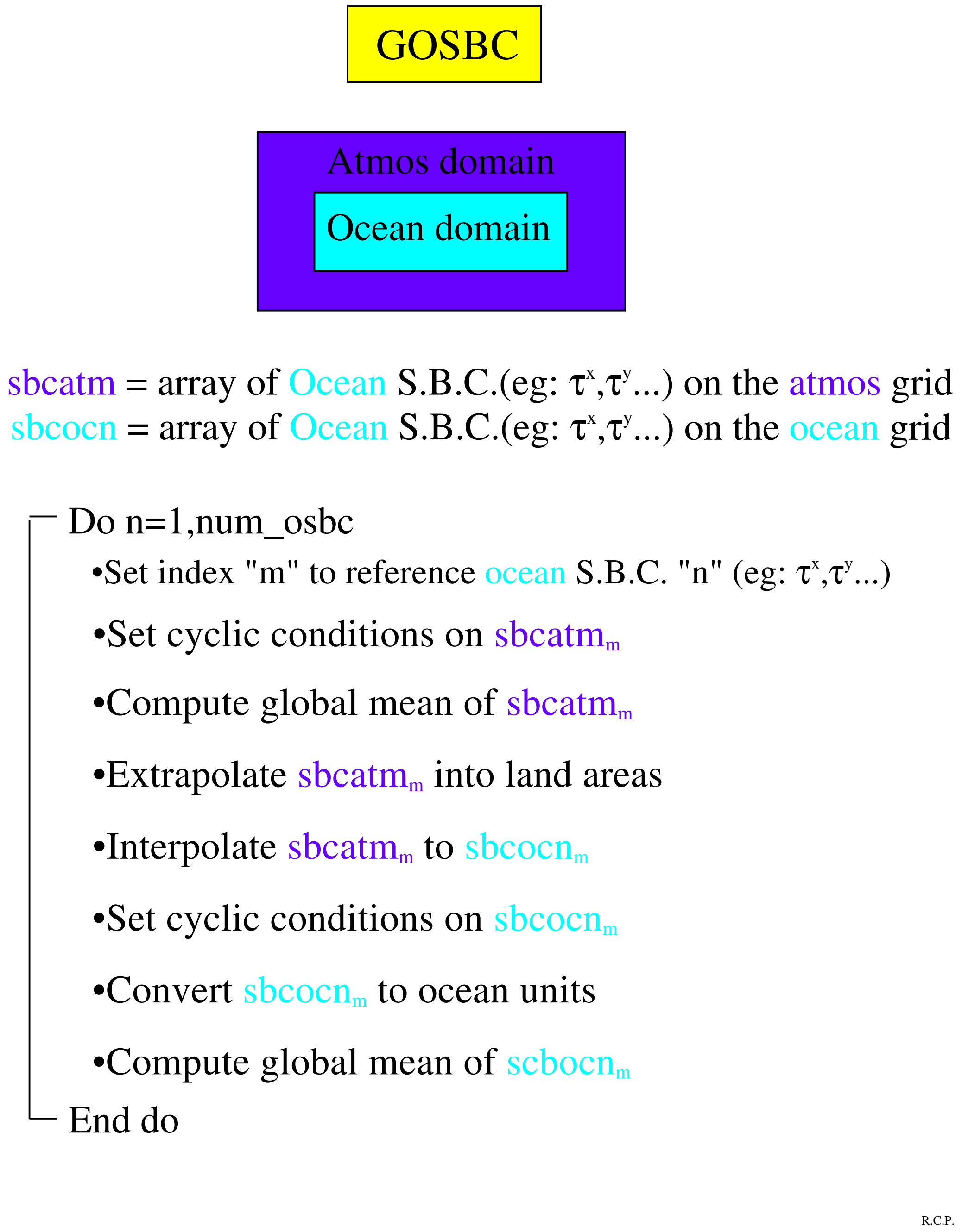


Figure 10.4: Flowchart for subroutine *gosbc.F*

Chapter 11

Discrete equations

Once surface boundary conditions are available for the ocean as described in Chapter 10, the ocean equations are integrated for one time step with each call to subroutine *mom*[1]. As indicated schematically in Figure 10.2 and described further in Section 10.1, *mom* may need to be called repeatedly until integration has been carried out for one time segment. For a flowchart indicating the calling sequence within *mom* for one time step, refer to Figure 11.1. The process starts by incrementing the ocean time step counter *itt* by one

$$itt = itt + 1 \tag{11.1}$$

and calling module *tmngr*[2] which increments ocean time by the number of seconds in one time step. Refer to Section 5.4.4 for choosing a time step length. Module *tmngr* additionally calculates a time and date as described in Section 6.2.9 and determines which events are to be activated on the current time step and which are not. Each event has an associated logical switch which is kept in file *switch.h*. An event might be something like writing a particular diagnostic for analysis, or doing a mixing time step, etc. After determining all logical switches, a call is made to the diagnostic initialization subroutine *diagi* which performs initializations for various diagnostics when required[3].

11.1 Time Stepping Schemes

Numerically, there are three types of time step schemes used within *mom*. The normal one is a leapfrog scheme which is centered in time. The others are mixing schemes intended to damp time splitting characteristic of schemes centered in time. Whether it's a mixing time step or not is determined by a logical switch *mixts* set within module *tmngr*. The type of mixing scheme is input through namelist as logical variable *eb* which if true indicates an Euler backward mixing time step, otherwise a forward mixing time step will be used. Refer to Section 5.4 for information on namelist variables. Of the two mixing schemes, the Euler backward is more diffusive than the forward. It also damps spatial scales whereas the forward scheme does not (Haltiner and Williams 1980). The focus here will be to explain how these schemes are implemented within *mom*. Mixing timesteps in the predecessors of MOM have been set at multiples of 17 since prehistoric times. This number seems to satisfy most people and has been empirically established long time ago.

[1]Contained in file *mom.F*.
[2]Contained in file tmngr.F.
[3]Assuming these diagnostics have been enabled by their options at compile time.

The following description assumes that the idea of a memory window as outlined in Section 3.3.1 is understood. Refer to Figure 11.2 and note the four columns: *Time Step*, *Type of Time Step*, a partially opened memory window with two disk areas indicated bu column $jmw < jmt$, and a fully opened memory window with no disk area indicated by column $jmw{=}jmt$.

Two sets of indices are required to act as pointers to specific areas on disk and in the memory window: one set points to $\tau - 1$, τ, and $\tau + 1$ locations on disk and these indices are named taum1disk, taudisk, and taup1disk; the other set points to $\tau - 1$, τ, and $\tau + 1$ locations within the memory window and these indices are named taum1, tau, and taup1.

At the beginning of each time step, memory window indices are updated as shown schematically in the *Type of Time Step* column. The whole idea is to get data positioned properly inside the memory window so that the equations can be solved for various types of time steps with only minimal changes[4].

To actually see how disk indices are cycled for various types of time steps, enable option *trace_indices* as described in Chapter 18.

11.1.1 Leapfrog

Consider the partially opened memory window shown in the third column. On leapfrog time steps, the updating arrows indicate that $\tau - 1$ variables in the memory window are being filled with what was τ disk data on the previous time step. Likewise, τ variables in the memory window are being filled with what was $\tau + 1$ disk data on the previous time step. The equation for a typical prognostic variable h indicates a central difference or leapfrog scheme in time. Note that the forcing term F is a function of τ for advective processes and $\tau - 1$ for diffusive processes. Before the leapfrog step, disk indices are set as follows:

- taum1disk = $mod(itt + 1, 2) + 1$
- tau = $mod(itt, 2) + 1$
- taup1disk = taum1disk

This formulation cyclically exchanges disk pointer indices every time step to assure that the correct disk data is read into and written from the proper memory window locations as described above. Memory window indices are always as follows when the memory window is partially opened:

- taum1 = 1
- tau = 2
- taup1 = 3

When the time step is complete, $\tau + 1$ data is written back over the $\tau - 1$ disk area. Looking to the fourth column, the fully opened memory window has no arrows therefore no movement of data. The disk pointers are not used and instead of being fixed in time, the memory pointers are cyclicly updated according to

- taum1 = $mod(itt + 0, 3) + 1$
- tau = $mod(itt + 1, 3) + 1$
- taup1 = $mod(itt + 2, 3) + 1$

which accomplishes renaming of areas within the memory window without moving data. Comparing time step n with $n + 1$ will clarify this.

[4] The only change is whether a time step length is $\Delta\tau$ or $2\Delta\tau$

11.1.2 Forward

On forward time steps when the memory window is partially opened as indicated by time step *n+2*, both $\tau - 1$ and τ variables are loaded with what was $\tau + 1$ disk data on the previous time step. In the third column, memory pointers are set to reflect this. The equation is the same as for the leapfrog time steps except that $2\Delta\tau$ is now replaced by $\Delta\tau$.

11.1.3 Euler Backward

Euler backward steps are comprised of two half steps. The first is identical to a forward step and the second fills τ variables within the memory window with $\tau + 1$ data from the first step. When the memory window is opened all the way, an *euler shuffle* is required to get data properly aligned in preparation for the next time step. This is the only data movement required when the memory window is fully opened[5].

11.1.4 Others

Another possibility is to do *Robert* time filtering every time step. Schematically, this amounts to:

$$h^\tau = h^\tau + \frac{\varepsilon}{2} \cdot (h^{\tau+1} - 2h^\tau + h^{\tau-1}) \tag{11.2}$$

where ε is an adjustable parameter. This is not an efficient operation when the memory window is only partially opened because it requires re-cycling through all latitudes on disk every time step. It's trivial when the memory window is fully opened however, this scheme is not implemented as of this writing.

11.2 Time and Space discretizations

Equations are solved for a group of latitude rows within the memory window. The group may be as few as one[6] or as many as $(jmt - 2)$ latitudes[7]. The number of groups depends on the size of the memory window and each group is solved one at a time until all groups have been solved. The subroutines called for each group of latitudes as outlined in Figure 11.1 are explained in the following sections. The reader is referred to Chapter 5 for a description of the variables, Chapter 7 for a description of the grid system, and Chapter 3 for a description of the memory window.

The following finite difference averaging and derivative operators[8] are used to discretize equations on the grid system. Before going further, the grid system should be firmly in mind.

<u>Averaging operators</u>

$$\overline{\alpha_{i,k,j}}^{\lambda} = \frac{\alpha_{i+1,k,j} + \alpha_{i,k,j}}{2} \tag{11.3}$$

$$\overline{\alpha_{i,k,j}}^{\phi} = \frac{\alpha_{i,k,j+1} + \alpha_{i,k,j}}{2} \tag{11.4}$$

$$\overline{\alpha_{i,k,j}}^{z} = \frac{\alpha_{i,k+1,j} + \alpha_{i,k,j}}{2} \tag{11.5}$$

[5]This can be eliminated by recalculating the pointers differently on the time step after the Euler backward step. However, the calculation gets tricky and has complications.

[6]Requiring three rows in the memory window.

[7]Requiring jmt rows in the memory window.

[8]The subscript j in the operators is replaced by *jrow* when α is dimensioned by the total number of latitudes.

where α is any variable defined on grid points within T cells or U cells. It should be noted that the average is defined midway between the variables being averaged. The derivative operators in space and time are defined as:

<u>Derivative operators</u>

$$\delta_\lambda(\alpha_{i,k,j}) = \frac{\alpha_{i+1,k,j} - \alpha_{i,k,j}}{a\Delta\lambda_i} \tag{11.6}$$

$$\delta_\phi(\alpha_{i,k,j}) = \frac{\alpha_{i,k,j+1} - \alpha_{i,k,j}}{a\Delta\phi_{jrow}} \tag{11.7}$$

$$\delta_z(\alpha_{i,k,j}) = -\frac{\alpha_{i,k+1,j} - \alpha_{i,k,j}}{\Delta z_k} \tag{11.8}$$

$$\delta_\tau(\beta_\tau) = \frac{\beta_{\tau+1} - \beta_{\tau-1}}{2\Delta\tau} \tag{11.9}$$

where grid distances (measured in cm) are determined by the distance between variables as indicated in Figures 7.1, 7.2, and 7.3 and discussed in Chapter 5. When $\alpha_{i,k,j}$ is defined at $T_{i,k,jrow}$grid points, then $a\Delta\lambda_i = dxu_i$ (a = 6370 x 10^5 cm) and when $\alpha_{i,k,j}$ is defined at $U_{i,k,jrow}$grid points, then $a\Delta\lambda_i = dxt_{i+1}$. Similarly, $a\Delta\phi_{jrow} = dyu_{jrow}$ when $\alpha_{i,k,j}$ is defined at $T_{i,k,jrow}$grid points and $a\Delta\phi_{jrow} = dyt_{jrow+1}$ when $\alpha_{i,k,j}$ is defined at $U_{i,k,jrow}$grid points. Note the negative sign in the vertical derivative. This is because z increases upwards while k increases downwards. The negative sign in Equation (11.8) is usually absorbed by reversing the indexing to give

$$\delta_z(\alpha_{i,k,j}) = \frac{\alpha_{i,k,j} - \alpha_{i,k+1,j}}{\Delta z_k} \tag{11.10}$$

In the finite difference approximation to the time derivative, Equation (11.9) is appropriate for the normal leapfrog time steps where $2\Delta\tau$ is in seconds. As indicated in Section 11.1, on mixing time steps, the denominator is replaced by $\Delta\tau$.

11.2.1 Key to understanding finite difference equations

The grid distances (measured in cm) are determined by the distance between variables as indicated in Figures 7.1, 7.2, and 7.3. Before looking at any finite difference equations, the reader should convince him/herself of the following relations.

- If $\alpha_{i,k,jrow}$ is defined at the T cell grid point given by $T_{i,k,jrow}$, then the operation $\delta_\lambda(\overline{\alpha_{i,k,jrow}}^\phi)$ results in a quantity defined at the U cell grid point with index $U_{i,k,jrow}$.
- If $\beta_{i,k,jrow}$ is defined at the U cell grid point given by $U_{i,k,jrow}$, then the operation $\delta_\lambda(\overline{\beta_{i-1,k,jrow-1}}^\phi)$ results in a quantity defined at the T cell grid point given by $T_{i,k,jrow}$.

These operations reflect the nature of the staggered B grid. Once convinced of the above, the second thing to be aware of is that it's not sufficient to only know what a quantity is anymore; where it's defined is just as important. The <u>where</u> information is usually built into the name of the variable by the naming convention described in Chapter 5. There are a specific number of interesting places on the grid. The grid point within a grid cell is one; the east, north, and bottom face of a cell are others. A quantity indexed by (i, k, j) may be defined at the grid point in $cell_{i,k,j}$ or on the east, north, or bottom face of $cell_{i,k,j}$. Note that if a variable

$\alpha_{i,k,j}$ is defined on the east face of $cell_{i,k,j}$, its value on the west face of the cell is $\alpha_{i-1,k,j}$. Similarly, if defined on the north face of $cell_{i,k,j}$, then its value on the south face of the cell is $\alpha_{i,k,j-1}$. Likewise, if defined on the bottom face of $cell_{i,k,j}$, then its value on the top face of the cell is $\alpha_{i,k-1,j}$.

Note that this convention is a departure from the indexing used in Bryan (1969) where the faces of cells were referenced by half indexes (i.e., $i + \frac{1}{2}$). Half indexes do not map well into Fortran and are not used in this manual. The idea is that by looking at this manual it should be possible to determine if the code is wrong (or vice versa).

Rules for using operators

In general, finite difference derivative and average operators don't commute unless the grid resolution is constant. Also, the finite difference derivative operator doesn't distribute unless the grid resolution is constant. Assuming that $\alpha_{i,k,j}$ is defined at grid points within T cells, then the above conditions are illustrated as

$$\delta_\lambda(\overline{\alpha_{i,k,j}}^\lambda) \neq \overline{\delta_\lambda(\alpha_{i,k,j})}^\lambda \tag{11.11}$$

$$\delta_\lambda(\alpha_{i,k,j} + \alpha_{i+1,k,j}) \neq \delta_\lambda(\alpha_{i,k,j}) + \delta_\lambda(\alpha_{i+1,k,j}) \tag{11.12}$$

How is a term like $\delta_\lambda(\overline{\alpha_{i,k,j}}^\lambda)$ evaluated? It can be expanded from the inside out as

$$\begin{aligned}
\delta_\lambda(\overline{\alpha_{i,k,j}}^\lambda) &= \delta_\lambda(\frac{\alpha_{i,k,j} + \alpha_{i+1,k,j}}{2}) \\
&= \frac{\frac{\alpha_{i+1,k,j}+\alpha_{i+2,k,j}}{2} - \frac{\alpha_{i,k,j}+\alpha_{i+1,k,j}}{2}}{dxt_{i+1}} \\
&= \frac{\alpha_{i+2,k,j} - \alpha_{i,k,j}}{2 \cdot dxt_{i+1}}
\end{aligned} \tag{11.13}$$

or from the outside in as

$$\begin{aligned}
\delta_\lambda(\overline{\alpha_{i,k,j}}^\lambda) &= \frac{\overline{\alpha_{i+1,k,j}}^\lambda - \overline{\alpha_{i,k,j}}^\lambda}{dxt_{i+1}} \\
&= \frac{\frac{\alpha_{i+1,k,j}+\alpha_{i+2,k,j}}{2} - \frac{\alpha_{i,k,j}+\alpha_{i+1,k,j}}{2}}{dxt_{i+1}} \\
&= \frac{\alpha_{i+2,k,j} - \alpha_{i,k,j}}{2 \cdot dxt_{i+1}}
\end{aligned} \tag{11.14}$$

however, evaluating from the outside in requires careful attention to where quantities are defined. This is particularly true for grid distances (note the use of dxt_{i+1}). Also, it is worth remembering that the results of operators are displaced by the distance of a half cell width. For example, the single operator $\delta_\lambda(\alpha_{i,k,j})$ results in a quantity defined on the eastern face[9] of cell $T_{i,k,jrow}$ and the double operator $\delta_\lambda(\delta_\lambda(\alpha_{i,k,j}))$ results in a quantity defined at the grid point within $T_{i+1,k,jrow}$. These results easily extend to two and three dimensions. Mixed double operators such as $\delta_\lambda(\overline{\alpha_{i,k,j}}^\phi)$ results in a quantity defined on the grid point within cell $U_{i,k,jrow}$.

Formal manipulations

There is no ambiguity in manipulating finite difference objects. As noted in Bryan (1969), there are formal rules and some are given below. They can be verified by substituting the basic

[9]This is the the longitude of the grid point within $U_{i,k,jrow}$.

finite difference derivative and averaging operators and expanding the terms. For illustrative purposes, consider one dimensional quantities α_i and γ_i (defined on longitudes of T cell grid points) and β_i (defined on longitudes of U cell grid points).

$$\overline{\alpha_i \gamma_i}^{\lambda} = \overline{\alpha_i}^{\lambda}\overline{\gamma_i}^{\lambda} + \frac{1}{4} dxu_i^2 \delta_\lambda(\alpha_i)\delta_\lambda(\gamma_i) \tag{11.15}$$

$$\delta_\lambda(\alpha_i\gamma_i) = \overline{\alpha_i}^{\lambda}\delta_\lambda(\gamma_i) + \overline{\gamma_i}^{\lambda}\delta_\lambda(\alpha_i) \tag{11.16}$$

$$dxt_{i+1}\cdot\delta_\lambda(\overline{\alpha_i}^{\lambda}\beta_i) = \overline{\beta_i \cdot dxu_i \cdot \delta_\lambda(\alpha_i)}^{\lambda} + \alpha_{i+1}\cdot dxt_{i+1}\cdot\delta_\lambda(\beta_i) \tag{11.17}$$

$$\overline{\overline{\alpha_i}^{\lambda}\beta_i}^{\lambda} = \alpha_{i+1}\cdot\overline{\beta_i}^{\lambda} + \frac{1}{4} dxt_{i+1}\delta_\lambda(\beta_i\cdot dxu_i\cdot\delta_\lambda(\alpha_i)) \tag{11.18}$$

These expressions also hold along other dimensions. In particular if ϕ is substituted[10] for λ or if z is substituted for λ. Consider further the two dimensional case where α_i is defined at T cell grid points, and β_i is defined at U cell grid points. The following rule may be derived by combining Equations (11.17) and (11.18)

$$\begin{aligned} & \alpha_{i,k,j}\cdot dxt_i\cdot\delta_\lambda(\overline{\beta_{i-1,k,j-1}}^{\phi}) + \overline{\beta_{i-1,k,j-1} dxu_{i-1}\cdot\overline{\delta_\lambda(\alpha_{i-1,k,j-1})}^{\phi}}^{\phi\lambda} \\ = \; & dxt_i\cdot\delta_\lambda(\overline{\alpha_{i-1,k,j}}^{\lambda}\overline{\beta i-1,k,j-1}^{\phi}) \\ + \; & \frac{1}{4} dyt_{jrow}\delta_\phi(\overline{\beta_{i-1,k,j-1}\cdot dyu_{jrow-1}\delta_\phi(dxu_{i-1}\delta_\lambda(\alpha_{i-1,k,j-1})))}^{\lambda}) \end{aligned} \tag{11.19}$$

For further details, refer to Appendix B.

11.3 Start of computation within Memory Window

Baroclinic velocities and tracers are solved for one group of latitudes at a time. This group is contained within the memory window as detailed in Section 3.3.1. The number of memory windows or groups of latitude rows is controlled by a “do loop” loop wrapping around the following sections with a scope extending to Section 11.12.

11.4 loadmw (loads the memory window)

Subroutine *loadmw*[11] begins by moving the memory window northward if necessary[12]. This operation involves copying data from the top two rows into the bottom two rows of the memory window as was described in Section 3.3.2.

Next, land/sea masks $tmask_{i,k,j}$ and $umask_{i,k,j}$ are constructed from $kmt_{i,jrow}$ and $kmu_{i,jrow}$. Their purpose is to promote vectorization throughout the calculations. Note that when the memory window is fully opened ($jmw = jmt$), this calculation is only done once per run instead of redundantly every time step.

After building the land/sea vectorization masks, a group of latitude rows are read from the τ and $\tau-1$ disks into the memory window. Time level indices depend on whether it is a normal leapfrog time step or a mixing time step as described in Section 11.1.

[10] The grid distances must also be replaced accordingly.

[11] Contained in file loadmw.F.

[12] It is necessary for all but the first group of latitudes. If the memory window is opened all the way (when $jmw = jmt$), then no movement is necessary.

Once the memory window has been loaded with variables, the external mode velocity is added to the internal mode velocity to construct the full velocity as in Equations (2.13) and (2.14). The finite difference counterpart to the external mode given in Equations (2.15) and (2.16) is

$$\overline{u}_{i,jrow,1} = -\frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau}}^{\lambda}) \tag{11.20}$$

$$\overline{u}_{i,jrow,2} = \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau}}^{\phi}) \tag{11.21}$$

The full velocity needs to be constructed for both time levels at this point because only the internal mode velocities[13] are kept in the latitude rows on disk.

$$u_{i,k,j,1,\tau-1} = \hat{u}_{i,k,j,1,\tau-1} - \frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau-1}}^{\lambda}) \tag{11.22}$$

$$u_{i,k,j,2,\tau-1} = \hat{u}_{i,k,j,2,\tau-1} + \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau-1}}^{\phi}) \tag{11.23}$$

$$u_{i,k,j,1,\tau} = \hat{u}_{i,k,j,1,\tau} - \frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau}}^{\lambda}) \tag{11.24}$$

$$u_{i,k,j,2,\tau} = \hat{u}_{i,k,j,2,\tau} + \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau}}^{\phi}) \tag{11.25}$$

where $\hat{u}$ is the internal mode velocity, ψ is the stream function, and H is the ocean depth defined over U cells by Equation (9.2).

Finally, the density $rho_{i,k,j}$ is computed as a function of $t_{i,k,j,1,\tau}$, $t_{i,k,j,2,\tau}$, and depth of level k based on a third order polynomial fit to the equation of state for sea water. The density is actually a deviation from a reference density ρ_k^{ref} as detailed in Section 6.2.2.

11.5 adv_vel (computes advective velocities)

Subroutine *adv_vel*[14] constructs advective velocities on the north, east and bottom faces of $T_{i,jrow}$ cells within the memory window. These advective velocities are then suitably averaged to construct advective velocities on the north, east and bottom faces of $U_{i,k,jrow}$ cells. These are the finite difference counterpart of the advective velocities in Equation (2.10) and are developed as follows. Advective velocities are defined in a direction normal to the cell face. Refer to Figure 11.3a which indicates a T cell surrounded by four U cells in the horizontal plane. Advective velocity on the eastern face of a T cell is a weighted average in the meridional direction of the zonal velocities at the vertices of the face. A similar relation holds for the advective velocity on the northern face of a T cell but it involves a zonal average of the meridional velocities. In both cases the average is in a direction normal to the flow

$$adv\_vet_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot dyu_{jrow} + u_{i,k,j-1,1,\tau} \cdot dyu_{jrow-1}}{2 \cdot dyt_{jrow}} \tag{11.26}$$

[13]After each group of internal mode velocities are solved within a time step, they are written back to disk. The external mode is unknown until all rows have been solved and subroutine *tropic* solves for the external mode. The only way to get the full velocity on disk is to read all rows back into memory after the external mode has been solved. This is why the external mode is added at the beginning of each time step for time levels τ and $\tau - 1$.

[14]Contained within file adv_vel.F.

$$adv\_vnt_{i,k,j} = \frac{u_{i,k,j,2,\tau} \cdot dxu_i + u_{i-1,k,j,2,\tau} \cdot dxu_{i-1}}{2 \cdot dxt_i} \tag{11.27}$$

This form is not arbitrary and comes from the condition that the work done by horizontal pressure forces must equal the work done by buoyancy. The details are given in Section 12.3. Note that with non-uniform resolution, the denominator is not equal to the sum of the weights if the grid is constructed as:

$$dxu_i = \frac{dxt_i + dxt_{i+1}}{2} \tag{11.28}$$

$$dyu_{jrow} = \frac{dyt_{jrow} + dyt_{jrow+1}}{2} \tag{11.29}$$

which was done in MOM 1 and earlier implementations[15]. However, MOM 2 allows grid construction[16] by

$$dxt_i = \frac{dxu_i + dxu_{i-1}}{2} \tag{11.30}$$

$$dyt_{jrow} = \frac{dyu_{jrow} + dyu_{jrow-1}}{2} \tag{11.31}$$

which guarantees that the denominator always equals the sum of the weighting factors. This is presumed to allow more accurate advective velocities when the grid is stretched in the horizontal. Refer to Chapter 7 for a further discussion on non-uniform grids and advection.

From the incompressibility condition expressed by Equation (2.5), the advective velocity on the bottom face of each cell is defined as the vertical integral of the divergence of the horizontal advective velocities from the surface down to the level of the particular cell. The rigid lid assumption sets the advective velocity at the top face of the first cell to zero. If option *implicit_free_surface* is enabled, the advective velocity at the top face of the first cell is $\frac{1}{rho_0 \cdot grav} \cdot \delta_\tau(ps_{i,jrow})$. Actually, since there is no prognostic equation for vertical velocity, the vertical velocity is calculated diagnostically as the advective velocity at the bottom face of each cell.

$$\begin{aligned} adv\_vbt_{i,k,j} &= \frac{1}{\cos\phi^T_{jrow}} \cdot \sum_{m=1}^{k} \Big(\delta_\lambda(adv\_vet_{i-1,m,j}) + \delta_\phi(adv\_vnt_{i,m,j-1})\Big) \cdot dzt_m \\ &= \frac{1}{\cos\phi^T_{jrow}} \cdot \sum_{m=1}^{k} \Big(\frac{adv\_vet_{i,m,j} - adv\_vet_{i-1,m,j}}{dxt_i} + \\ &\qquad \frac{adv\_vnt_{i,m,j} - adv\_vnt_{i,m,j-1}}{dyt_{jrow}}\Big) \cdot dzt_m \end{aligned} \tag{11.32}$$

where ϕ^T_{jrow} refers to the latitude at point $T_{i,k,jrow}$. To calculate advective velocities on faces of U cells within the memory window, the averaging approach indicated by Dukowicz and Smith (1994) and subsequently described by Webb (1995) is used, which involves averaging the advective velocities from T cells onto U cells. Their formulas are:

[15] This was the motivation for exploring grid construction by method 2 as detailed in Section 7.2.3

[16] The old construction method 1 is enabled by option *centered_t*. Refer to Chapter 7 for a description of grid design.

$$adv\_veu_{i,k,j} = \overline{adv\_vet_{i,k,j}}^{\lambda\phi} \tag{11.33}$$
$$adv\_vnu_{i,k,j} = \overline{adv\_vnt_{i,k,j}}^{\lambda\phi} \tag{11.34}$$
$$adv\_vbu_{i,k,j} = \overline{adv\_vbt_{i,k,j}}^{\lambda\phi} \tag{11.35}$$

This works when the grid cells are square. For MOM 2, a more general form is used to preserve fluxes when the grid resolution is non-uniform. The idea is to take a weighted average in a direction normal to the flow and a linear interpolation in a direction parallel to the flow. Referring to Figure 11.3b and using the indicated grid distances, gives:

$$adv\_vnu_{i,k,j} = \frac{1}{dyt_{jrow+1} \cdot dxu_i} \cdot \Big[(adv\_vnt_{i,k,j} \cdot duw_i + adv\_vnt_{i+1,k,j} \cdot due_i) \cdot dus_{jrow+1} + (adv\_vnt_{i,k,j+1} \cdot duw_i + adv\_vnt_{i+1,k,j+1} \cdot due_i) \cdot dun_{jrow}\Big] \tag{11.36}$$

$$adv\_veu_{i,k,j} = \frac{1}{dyu_{jrow} \cdot dxt_{i+1}} \cdot \Big[(adv\_vet_{i,k,j} \cdot dus_{jrow} + adv\_vet_{i,k,j+1} \cdot dun_{jrow}) \cdot duw_{i+1} + (adv\_vet_{i+1,k,j} \cdot dus_{jrow} + adv\_vet_{i+1,k,j+1} \cdot dun_{jrow}) \cdot due_i\Big] \tag{11.37}$$

$$\begin{aligned} adv\_vbu_{i,k,j} = & \frac{1}{dyu_{jrow} \cdot dxu_i \cdot \cos\phi^U_{jrow}} \cdot \Big[adv\_vbt_{i,k,j} \cdot dus_{jrow} \cdot duw_i \cdot \cos\phi^T_{jrow} \\ + & \quad adv\_vbt_{i+1,k,j} \cdot dus_{jrow} \cdot due_i \cdot \cos\phi^T_{jrow} \\ + & \quad adv\_vbt_{i,k,j+1} \cdot dun_{jrow} \cdot duw_i \cdot \cos\phi^T_{jrow+1} \\ + & \quad adv\_vbt_{i+1,k,j+1} \cdot dun_{jrow} \cdot due_i \cdot \cos\phi^T_{jrow+1}\Big] \end{aligned} \tag{11.38}$$

This weighting eliminates the problem of numerical de-coupling between the advective velocities on the bottom of T cells and U cells in the presence of flow over topographic gradients as in MOM 1 and previous implementations. It should also be noted that the weighting factors dus, dun, due, and duw are always equal if grid construction is by method 1 as described in Chapter 7. A gain in speed can be realized by removing these factors from the above equations if method 2 is not to be used. Refer to Sections 11.11.5 and 11.10.7 for details on how these advective velocities are used.

11.6 isopyc (computes isopycnal mixing tensor components)

Subroutine *isopyc*[17] computes the isopycnal mixing tensor components and the isopycnal advection velocities which parameterize the effect of mesoscale eddies on the isopycnals. This only applies when option *isopycmix* is enabled. If option *gent_mcwilliams* is also enabled, the Gent-McWilliams (1990) advection velocities are computed. Refer to Sections 15.16.3, 15.16.4, and 15.16.5 for the details.

[17]Contained in file isopyc.F.

11.7 vmixc (computes vertical mixing coefficients)

Subroutine *vmixc*[18] computes vertical mixing coefficients for momentum ($diff\_cbu_{i,k,j}$) and tracers ($diff\_cbt_{i,k,j}$) according to the scheme selected by options at compile time. Subscripts for the mixing coefficients depend on which option was enabled. Options are *constvmix* for constant vertical mixing coefficients, *ppvmix* for Richardson dependent vertical mixing coefficients as in Pacanowski/Philander (1981), and *tcvmix* for the second order turbulence closure of Mellor/Yamada as given in Rosati and Miyakoda (1988). However, *tcvmix* is not available as of this writing but is being implemented by Rosati at GFDL. One of these must be enabled and all adjustable values are input through namelist. Refer to Section 5.4 for information on namelist variables.

Additionally, there are two hybrid options allowed which supply mixing coefficients in the vertical for tracers but not momentum. Option *isopycmix* is for diffusion of tracers along isopycnals as described in Sections 11.6 and 15.16.3 and option *bryan_lewis_vertical* (Bryan/Lewis 1979) sets constant values as a function of depth for the vertical mixing coefficient. When option *isopycmix* is enabled, mixing coefficients for tracers from other options such as *constvmix*, *ppvmix*, or *bryan_lewis_vertical* are used as background values and added to the vertical diffusion coefficient[19] from option *isopycmix*.

Also, for use with option *isopycmix* is option *held_larichev* as in Held/Larichev (1995) which predicts variable isopycnal mixing coefficients. As of this writing, option *held_larichev* is incomplete. Its diffusion coefficient should be limited to regions of possible baroclinic instability. All adjustable values are input through namelist. Refer to Section 5.4 for information on namelist variables. For details on specific schemes, consult their options.

11.8 hmixc (computes horizontal mixing coefficients)

Subroutine *hmixc*[20] computes horizontal mixing coefficients for momentum ($diff\_ceu_{i,k,j}$ and $diff\_cnu_{i,k,j}$) and tracers ($diff\_cet_{i,k,j}$ and $diff\_cnt_{i,k,j}$) according to the scheme selected by options at compile time. Currently, options include *consthmix* which use constant horizontal mixing coefficients appropriate for ∇^2 mixing, *biharmonic* which use constant mixing coefficients appropriate for higher order ∇^4 mixing, and *smagnlmix* for Smagorinsky nonlinear mixing coefficients after Smagorinsky (1963).

As in Section 11.7, coefficients from hybrid schemes for mixing of tracers but not mixing of momentum are also allowed. They are *bryan_lewis_horizontal*, *isopycmix* and *held_larichev* as referred to in Section 11.7. When option *isopycmix* is enabled, mixing coefficients for tracers from *consthmix*, *smagnlmix*, or *bryan_lewis_vertical* are used as background values and added to the horizontal diffusion coefficients[21] from option *isopycmix*. For details on specific schemes, consult their options.

[18] Contained in file vmixc.F.

[19] This is effectively done by adding diffusive fluxes from the basic scheme to the isopycnal flux across the bottom face of the T cell.

[20] Contained in file hmixc.F.

[21] This is effectively done by adding diffusive fluxes from the basic scheme to the isopycnal flux across the northern and eastern face of the T cell.

11.9 setvbc (set vertical boundary conditions)

Subroutine *setvbc*[22] sets vertical boundary conditions for momentum $smf_{i,j,n}$ and tracers $stf_{i,j,n}$ at the ocean surface and their counterparts defined at the ocean bottom $bmf_{i,j,n}$ and $btf_{i,j,n}$. Also, $bmf_{i,j,n}$ can be set to zero (free slip) or a linear bottom drag condition with the drag coefficient being input through namelist. Refer to Section 5.4 for information on namelist variables. Refer to Chapter 10 for further discussion on surface boundary conditions.

11.10 tracer (computes tracers)

Subroutine *tracer* is contained in file tracer.F and computes tracers given by $t_{i,k,j,n,\tau+1}$ for $n = 1$ to nt, where nt is the number of tracers. Index $n = 1$ corresponds to potential temperature in Equation (2.3) and index $n = 2$ is for salinity in Equation (2.4). These first two tracers are dynamically active in the sense that they determine density. Other tracers may be added by setting parameter $nt > 2$ but the additional tracers are passive. In previous versions of the model, tracers were solved after the baroclinic velocities. However, in order to accommodate option *pressure_gradient_average* tracers are now computed before baroclinic velocities.

11.10.1 Tracer components

Each tracer is considered as a component of one tracer equation and solved separately by a "do $n = 1, nt$" loop extending to Section 11.10.10. As with the finite difference momentum equations given in Section 11.11.5, the finite difference tracer equation is written in flux form to conserve first and second moments as discussed in Chapter 12.

11.10.2 Advective and Diffusive fluxes

Advective and diffusive fluxes across the north, east, and bottom faces of T cells within the memory window are calculated for use by the advection and diffusion operators described in Section 11.10.7. The canonical forms are given below. The actual form of the diffusive flux may differ depending on which option is enabled. Note that advective flux is missing a factor of $\frac{1}{2}$ which is incorporated into the advection operator for reasons of speed. Also, a cosine factor has been absorbed into the definition of $adv\_vnt_{i,k,j}$ for reasons of speed. Limits on j are typically $jsmw = 2$ and $jemw = jmw - 1$, except for the last memory window, which may only be partially full. Meridional operators require meridional flux to be computed for $j = 1$, which is why the lower limit of the j index is not identical for all fluxes. Refer to Figures 3.3 and 3.7 for clarification of the latitude indexing limits.

$$
\begin{aligned}
adv\_fe_{i,k,j} &= 2adv\_vet_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau}}^{\lambda} \\
&= adv\_vet_{i,k,j} \cdot \left(t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}\right), j = jsmw, jemw
\end{aligned}
\tag{11.39}
$$

$$
\begin{aligned}
adv\_fn_{i,k,j} &= 2adv\_vnt_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau}}^{\phi} \\
&= adv\_vnt_{i,k,j} \cdot \left(t_{i,k,j,n,\tau} + t_{i,k,j+1,n,\tau}\right), j = 1, jemw
\end{aligned}
\tag{11.40}
$$

$$
\begin{aligned}
adv\_fb_{i,k,j} &= 2adv\_vbt_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau}}^{z} \\
&= adv\_vbt_{i,k,j} \cdot \left(t_{i,k,j,n,\tau} + t_{i,k+1,j,n,\tau}\right), j = jsmw, jemw
\end{aligned}
\tag{11.41}
$$

[22] Contained in file setvbc.F.

$$\begin{aligned} diff\_fe_{i,k,j} &= \frac{diff\_cet_{i,k,j}}{\cos\phi^T_{jrow}}\delta_\lambda(t_{i,k,j,n,\tau-1}) \\ &= diff\_cet_{i,k,j}\cdot\frac{t_{i+1,k,j,n,\tau-1}-t_{i,k,j,n,\tau-1}}{\cos\phi^T_{jrow}dxu_i}\ ,j=jsmw,jemw \end{aligned} \tag{11.42}$$

$$\begin{aligned} diff\_fn_{i,k,j} &= diff\_cnt_{i,k,j}\cdot\cos\phi^U_{jrow}\delta_\phi(t_{i,k,j,n,\tau-1}) \\ &= diff\_cnt_{i,k,j}\cdot\cos\phi^U_{jrow}\frac{t_{i,k,j+1,n,\tau-1}-t_{i,k,j,n,\tau-1}}{dyu_{jrow}}\ ,j=1,jemw \end{aligned} \tag{11.43}$$

$$\begin{aligned} diff\_fb_{i,k,j} &= diff\_cbt_{i,k,j}\cdot\delta_z(t_{i,k,j,n,\tau-1}) \\ &= diff\_cbt_{i,k,j}\cdot\frac{t_{i,k,j,n,\tau-1}-t_{i,k+1,j,n,\tau-1}}{dzw_k}\ ,j=jsmw,jemw \end{aligned} \tag{11.44}$$

At the top and bottom, boundary conditions are applied to the vertical fluxes where the bottom cell is at level $kb = kmt_{i,jrow}$. Also, the advective flux through the bottom of the last level $kb = kmt_{i,jrow}$ is zero[23]. Note that the advective flux through the bottom of the last vertical cell $k = km$ is set to zero. Also, when option *stream_function* is enabled, $adv\_vbt_{i,0,j} = 0$ because of the rigid lid but $adv\_vbt_{i,0,j} \neq 0$ when option *implicit_free_surface* is enabled.

$$kb = kmt_{i,jrow} \tag{11.45}$$
$$diff\_fb_{i,0,j} = stf_{i,j,n} \tag{11.46}$$
$$diff\_fb_{i,kb,j} = btf_{i,j,n} \tag{11.47}$$
$$adv\_fb_{i,0,j} = adv\_vbt_{i,0,j}\cdot(t_{i,1,j,n,\tau}+t_{i,1,j,n,\tau}) \tag{11.48}$$
$$adv\_fb_{i,km,j} = 0.0 \tag{11.49}$$

11.10.3 Isopycnal fluxes

When option *isopycmix* is enabled, the horizontal components of the isopycnal diffusive flux in Section 15.16.3 are added to the horizontal diffusive flux in Equations (11.42) and (11.43). The complete diffusive flux across the northern and eastern face of a T cell then is given by

$$diff\_fe_{i,k,j} = diff\_fe^{old}_{i,k,j} + diff\_fet^{iso}_{i,k,j} \tag{11.50}$$
$$diff\_fn_{i,k,j} = diff\_fn^{old}_{i,k,j} + diff\_fnt^{iso}_{i,k,j} \tag{11.51}$$

where the isopycnal diffusion fluxes $diff\_fet^{iso}_{i,k,j}$ and $diff\_fnt^{iso}_{i,k,j}$ are given in Section 15.16.3. The background fluxes are given by

$$diff\_fe^{old}_{i,k,j} = diff\_cet_{i,k,j}\cdot\frac{t_{i+1,k,j,n,\tau-1}-t_{i,k,j,n,\tau-1}}{\cos\phi^T_{jrow}dxu_i} \tag{11.52}$$
$$diff\_fn^{old}_{i,k,j} = diff\_cnt_{i,k,j}\cdot\cos\phi^U_{jrow}\frac{t_{i,k,j+1,n,\tau-1}-t_{i,k,j,n,\tau-1}}{dyu_{jrow}} \tag{11.53}$$

where $diff\_cet_{i,k,j}$ and $diff\_cnt_{i,k,j}$ are the coefficients from whatever subgrid scale mixing parameterization option has been enabled. The $[K^{31}]$ and $[K^{32}]$ parts of the diffusive flux across

[23] Zero to within roundoff. Note that unlike adv_vbt$_{i,kb,j}$, adv_vbu$_{i,kb,j}$ can be non-zero if there is a bottom slope.

the bottom face of the T cell is given as $diff\_fbiso_{i,k,j}$ which is the vertical flux term $-F^z_{i,k,j}$ from Section 15.16.3 minus the $[K^{33}]$ piece.

$$diff\_fbiso_{i,k,j} \quad = \quad -F^z_{i,k,j} - K^{33}_{i,k,j}\delta_z T_{i,k,j} \tag{11.54}$$

Note that the isopycnal diffusion coefficient A_I has been absorbed into the tensor components $[K^{31}]$, $[K^{32}]$, $[K^{33}]$. The $[K^{33}]$ component is handled implicitly as indicated in Section 11.10.7.

11.10.4 Source terms

It is possible to introduce sources into the tracer equation by enabling option *source_term*. If this option is enabled, the source term is initialized to zero for each component of the tracer equation.

$$source_{i,k,j} = 0.0 \tag{11.55}$$

Adding new sources or sinks to the tracer equations is a matter of calculating them and adding to $source_{i,k,j}$ as indicated in Sections 11.10.5 and 11.10.6.

11.10.5 Sponge boundaries

If option *sponges* is enabled, a Newtonian damping term is added to the tracer equation through the source term near northern and southern boundaries where γ^{-1} is a Newtonian damping time scale input through a namelist and $t^\star_{i,k,j,n}$ is interpolated in time from data prepared by the scripts in PREP_DATA. If option *equatorial_sponge* is enabled, then the profile from Equation (15.1) is used for $t^\star_{i,k,j,n}$ instead of data prepared in PREP_DATA. Refer to Section 5.4 for information on namelist variables. The time interpolation method is the same as described in Section 10.2.

$$source_{i,k,j} = source_{i,k,j} - \gamma \cdot (t_{i,k,j,n,\tau-1} - t^\star_{i,k,j,n}) \tag{11.56}$$

11.10.6 Shortwave solar penetration

If option *shortwave* is enabled, the divergence of shortwave penetration is also added to the source term using

$$source_{i,k,j} = source_{i,k,j} + sbcocn_{i,jrow,m_{i,jrow,isw}} \cdot divpen_k \tag{11.57}$$

where subscript *isw* points to the shortwave surface boundary condition. This only applies for $n = 1$. Refer to Section 15.9.8 for more details about the divergence of shortwave penetration.

11.10.7 Tracer operators

Finite difference tracer operators as described in Section 5.1 are used for the various terms when solving for $t_{i,k,j,n,\tau+1}$ and parallel those used in the solution of the internal mode velocities. These operators are also used in diagnostics throughout MOM 2. They are defined in file *fdift.h* and their canonical forms are given by:

$$DIFF\_Tx_{i,k,j} = \frac{diff\_fe_{i,k,j} \cdot tmask_{i+1,k,j} - diff\_fe_{i-1,k,j} \cdot tmask_{i-1,k,j}}{\cos\phi^T_{jrow} \cdot dxt_i} \tag{11.58}$$

$$DIFF\_Ty_{i,k,j} = \frac{diff\_fn_{i,k,j} \cdot tmask_{i,k,j+1} - diff\_fn_{i,k,j-1} \cdot tmask_{i,k,j-1}}{\cos\phi^T_{jrow} \cdot dyt_{jrow}} \tag{11.59}$$

$$DIFF\_Tz_{i,k,j} = \frac{diff\_fb_{i,k-1,j} - diff\_fb_{i,k,j}}{dzt_k} \tag{11.60}$$

$$ADV\_Tx_{i,k,j} = \frac{adv\_fe_{i,k,j} - adv\_fe_{i-1,k,j}}{2\cos\phi^T_{jrow} dxt_i} \tag{11.61}$$

$$ADV\_Ty_{i,k,j} = \frac{adv\_fn_{i,k,j} - adv\_fn_{i,k,j-1}}{2\cos\phi^T_{jrow} dyt_{jrow}} \tag{11.62}$$

$$ADV\_Tz_{i,k,j} = \frac{adv\_fb_{i,k-1,j} - adv\_fb_{i,k,j}}{2dzt_k} \tag{11.63}$$

where the vertical diffusion operator needs no masking because the masking effect has been built into boundary conditions $stf_{i,j,n}$ and $btf_{i,j,n}$ at the top and bottom of the column.

The finite difference counterpart to Equations (2.3) and (2.4) is given as

$$\mathcal{L}^T(t_{i,k,j,n,\tau}) = ADV\_Tx_{i,k,j} + ADV\_Ty_{i,k,j} + ADV\_Tz_{i,k,j} \tag{11.64}$$

Implicit vertical diffusion

When option *implicitvmix* is enabled, the vertical diffusion operator in Equation (11.60) becomes

$$DIFF\_Tz_{i,k,j} = (1 - aidif) \cdot \frac{diff\_fb_{i,k-1,j} - diff\_fb_{i,k,j}}{dzt_k} \tag{11.65}$$

where the implicit factor $aidif$ is used to separate the term into explicit and implicit parts.

Isopycnal mixing

When option *isopycmix* is enabled, the vertical diffusion operator in Equation (11.60) becomes

$$\begin{aligned} DIFF\_Tz_{i,k,j} &= (1 - aidif) \cdot \frac{diff\_fb_{i,k-1,j} - diff\_fb_{i,k,j}}{dzt_k} + \\ &\quad \frac{diff\_fbiso_{i,k-1,j} - diff\_fbiso_{i,k,j}}{dzt_k} \end{aligned} \tag{11.66}$$

where the implicit factor is $aidif$ and the K^{33} term is within $diff\_fb_{i,k-1,j}$ which is solved implicitly. The K^{31} and K^{32} terms are handled explicitly (no implicit treatment) within $diff\_fbiso_{i,k,j}$ which is defined by Equation (11.54). The horizontal isopycnal fluxes are given by Equations (11.50) and (11.51). Refer to Section 15.16.3 for further details on isopycnal mixing.

Gent-McWilliams advection velocities

If option *isopycmix* and option *gent_mcwilliams* are enabled, the advective operators for Equation (11.72) are given by

$$ADV\_Txiso_{i,k,j} = \frac{adv\_vetiso_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau-1}}^{\lambda} - adv\_vetiso_{i-1,k,j} \cdot \overline{t_{i-1,k,j,n,\tau-1}}^{\lambda}}{\cos\phi^T_{jrow} dxt_i} \tag{11.67}$$

$$ADV\_Tyiso_{i,k,j} = \frac{adv\_vntiso_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau-1}}^{\phi} - adv\_vntiso_{i,k,j-1} \cdot \overline{t_{i,k,j-1,n,\tau-1}}^{\phi}}{\cos\phi^T_{jrow} dyt_{jrow}} \tag{11.68}$$

$$ADV\_Tziso_{i,k,j} = \frac{adv\_fbiso_{i,k-1,j} - adv\_fbiso_{i,k,j}}{2dzt_k} \tag{11.69}$$

where twice the advective flux at the bottom of the T cell is

$$adv\_fbiso_{i,k,j} = 2(adv\_vbiso_{i,k,j} \cdot \overline{t_{i,k,j,n,\tau-1}}^{z}) \tag{11.70}$$

As with the normal advective flux, the factor of two is for reasons of computational speed. Note also, that a cosine factor has been absorbed within $adv\_vntiso_{i,k,j}$ to mimic the regular meridional advective velocity. The horizontal components are treated differently than the vertical component to save memory at the expense of speed. If memory is not a problem, the advective flux across the north and east face of the T cell can be computed separately to eliminate the redundancy. The total advection becomes the regular advection plus the Gent-McWilliams advection given as

$$\mathcal{L}^{total}(t_{i,k,j,n,\tau}) = \mathcal{L}^T(t_{i,k,j,n,\tau}) + \mathcal{L}^{gm}(t_{i,k,j,n,\tau-1}) \tag{11.71}$$

where the Gent-McWilliams advective operator is

$$\mathcal{L}^{gm}(t_{i,k,j,n,\tau-1}) = ADV\_Txiso_{i,k,j} + ADV\_Tyiso_{i,k,j} + ADV\_Tziso_{i,k,j} \tag{11.72}$$

It should be noted that when option *fct* is enabled, Equation 11.72 is not used in *tracer.F*. Instead, the Gent-McWilliams advection velocities are added to the regular advective velocities within the flux corrected transport calculations.

11.10.8 Solving for the tracer

The tracer equation is solved using a variety of time differencing schemes (leapfrog, forward, Euler backward) as outlined in Section 11.1. A complication arises due to the vertical diffusion term which may be solved explicitly or implicitly. The implicit treatment is appropriate when large vertical diffusion coefficients (or small vertical grid spacing) limit the time step. The following discussion pertains to the leapfrog scheme. The other schemes amount to changing $2\Delta t$ to Δt in what follows.

Explicit vertical diffusion

Using the above operators, the tracer is computed directly as:

$$\begin{aligned} t_{i,k,j,n,\tau+1} = t_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot (\ & DIFF\_Tx_{i,k,j} + DIFF\_Ty_{i,k,j} + DIFF\_Tz_{i,k,j} \\ - \quad & ADV\_Tx_{i,k,j} - ADV\_Ty_{i,k,j} - ADV\_Tz_{i,k,j} \\ + \quad & source_{i,k,j}) \cdot tmask_{i,k,j} \end{aligned} \tag{11.73}$$

When options *isopycmix* and *gent_mcwilliams* are enabled, the right hand side of Equation (11.73) also includes the flux form of the advection terms[24] given by Equations (11.67), (11.68), and (11.69). In effect, combining Equations (11.64) and (11.72) gives the total advection as in Equation (11.71).

Implicit vertical diffusion

In general, vertical diffusion is handled implicitly when using option *implicitvmix* or option *isopycmix*. In either case, Equation (11.73) is solved in two steps. The first calculates an explicit piece $t^{\star}_{i,k,j,n}$ using all terms except the portion of vertical diffusion which is to be solved implicitly. The equation is

$$
\begin{aligned}
t^{\star}_{i,k,j,n} = t_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot (\; & DIFF\_Tx_{i,k,j} + DIFF\_Ty_{i,k,j} + (1 - aidif) \cdot DIFF\_Tz_{i,k,j} \\
- \quad & ADV\_Tx_{i,k,j} - ADV\_Ty_{i,k,j} - ADV\_Tz_{i,k,j} \\
+ \quad & source_{i,k,j}) \cdot tmask_{i,k,j}
\end{aligned} \tag{11.74}
$$

where *aidif* is the implicit vertical diffusion factor. Setting *aidif = 1.0* gives full implicit treatment and setting *aidif = 0* gives full explicit treatment for the vertical diffusion term. Intermediate values give semi-implicit treatment. The second step involves solving the implicit equation

$$
t_{i,k,j,n,\tau+1} = t^{\star}_{i,k,j,n} + 2\Delta\tau \cdot \delta_z(aidif \cdot diff\_cbt_{i,k,j} \cdot \delta_z(t_{i,k,j,n,\tau+1})) \tag{11.75}
$$

Notice that $t_{i,k,j,n,\tau+1}$ appears on both sides of the equation. Solving this involves inverting a tri-diagonal matrix and the technique is taken from pages 42 and 43 of Numerical Recipes in Fortran (1992). For details, refer to Section 15.19.4.

11.10.9 Diagnostics

At this point, diagnostics are computed for tracer component n. For a description of the diagnostics, refer to Chapter 18.

11.10.10 End of tracer components

At this point, the "do $n = 1, nt$" loop issued in Section 11.10.1 is ended and all tracers have been computed.

11.10.11 Explicit Convection

The condition for gravitational instability is given by

$$
\delta_z(\rho_{i,k,j}) < 0 \tag{11.76}
$$

If option *implicitvmix* is enabled, the instability is handled by implicit vertical diffusion using huge vertical diffusion coefficients *diff_cbt_limit* input through namelist. Refer to Section 5.4 for information on namelist variables. Otherwise, when option *implicitvmix* is not enabled, the temperature and salinity within each vertical column of T cells is stabilized where needed by one of two explicit convection methods. These methods involve mixing predicted temperature

[24]These could have been added directly to the advective fluxes computed previously. For now they are kept separately for diagnostic reasons.

and salinity between two adjacent levels ($t_{i,k,j,n,\tau+1}$ and $t_{i,k+1,j,n,\tau+1}$ for $n = 1, 2$). If option *fullconvect* is enabled, the convection scheme of Rahmstorf (1993) is used. Otherwise, the original scheme involving alternate mixing of odd and even pairs of levels is used. Refer to Section 15.13. The mixing in the latter case may be incomplete and multiple passes through the scheme is allowed by variable *ncon* which is also input through namelist. If explicit convection is active, additional diagnostics are computed. Refer to Section 6.2.1 for the details.

11.10.12 Filtering

If the time step constraint imposed by convergence of meridians is to be relaxed, $t_{i,k,j,n,\tau+1}$ is filtered in longitude by one of two techniques: Fourier filtering (Bryan, Manabe, Pacanowski 1975) enabled by option *fourfil* or finite impulse response filtering enabled by option *firfil*. Both should be used with caution and only when necessary at high latitudes. *firfil* is much faster than *fourfil*.

11.10.13 Accumulating $sbcocn_{i,jrow,m}$

Finally, if required as a surface boundary condition for the atmosphere, $t_{i,1,j,n,\tau}$ is accumulated and averaged over one time segment. The result is stored in $sbcocn_{i,jrow,m}$ where m relates the ordering of n in the array of surface boundary conditions as discussed in Section 10.3.

11.11 clinic (computes internal mode velocities)

Subroutine *clinic*[25] computes the baroclinic or internal mode velocities $\hat{u}_{i,k,j,n,\tau+1}$ for velocity components $n = 1$ (the zonal velocity) and 2 (the meridional velocity). They are the finite difference counterparts of the velocities in Equations (2.1) and (2.2) after the vertical means have been removed. The finite difference equations given below are written in flux form to allow conservation of first and second moments as discussed in Chapter 12. The solution starts by computing the hydrostatic pressure gradient terms.

11.11.1 Hydrostatic pressure gradient terms

Both horizontal pressure gradient terms in Equations (2.1) and (2.2) are computed first. The unknown barotropic surface pressure gradients exerted by the rigid lid at the ocean surface are not needed since only the internal mode velocities are being solved for[26]. The discretized forms are given by

$$-\frac{1}{\rho_\circ a \cdot \cos\phi} p_\lambda \approx -grad\_p_{i,k,j,1} = -\frac{1}{\rho_\circ \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{p_{i,k,j}}^{\phi}) \tag{11.77}$$

$$-\frac{1}{\rho_\circ a} p_\phi \approx -grad\_p_{i,k,j,2} = -\frac{1}{\rho_\circ} \delta_\phi(\overline{p_{i,k,j}}^{\lambda}) \tag{11.78}$$

The discrete form of the hydrostatic Equation (2.6) is given by

$$\delta_z(p_{i,k,j}) = -grav \cdot \overline{\rho_{i,k,j}}^{z} \tag{11.79}$$

[25] Contained in file clinic.F.

[26] Indeed, lack of knowledge about these barotropic surface pressure gradients is the very reason the internal mode velocities are being solved for.

with $grav = 980.6$ cm$/sec^2$ and is used to eliminate p from Equations (11.77) and (11.78). Note that the hydrostatic pressure is an anomaly because $\rho_{i,k,j}$ is an anomaly as described in Section 6.2.2. The hydrostatic pressure gradient terms at the depth of the grid point in the first level are

$$grad\_p_{i,1,j,1} = \frac{grav \cdot dzw_0}{\rho_\circ \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{\rho_{i,1,j}}^{\phi}) \tag{11.80}$$

$$grad\_p_{i,1,j,2} = \frac{grav \cdot dzw_0}{\rho_\circ} \delta_\phi(\overline{\rho_{i,1,j}}^{\lambda}) \tag{11.81}$$

which assumes that density at the ocean surface is the same as at depth $zt_{k=1}$[27]. Pressure gradient terms at successive levels 2 through $kmu_{i,jrow}$ are built by partial sums

$$grad\_p_{i,k,j,1} = \sum_{m=2}^{k} \frac{grav \cdot dzw_{m-1}}{\rho_\circ \cdot \cos\phi^U_{jrow}} \delta_\lambda(\overline{\frac{\rho_{i,m,j} + \rho_{i,m-1,j}}{2}}^{\phi}) \tag{11.82}$$

$$grad\_p_{i,k,j,2} = \sum_{m=2}^{k} \frac{grav \cdot dzw_{m-1}}{\rho_\circ} \delta_\phi(\overline{\frac{\rho_{i,m,j} + \rho_{i,m-1,j}}{2}}^{\lambda}) \tag{11.83}$$

When option *pressure_gradient_average* is enabled, ρ in Equations (11.79) through (11.83) is replaced by the time averaged value $\tilde{\rho}$ given by Equation (15.180) and discussed further in Section 15.19.2. Note that since the integration is to levels of constant depth, hydrostatic pressure gradients at depth are zero when ρ is a function of z only. Note also, that the pressure gradients are different for grid construction methods 1 and 2 as discussed in Chapter 7.2.3.

11.11.2 Momentum components

After constructing both components of the hydrostatic pressure gradients, the remaining terms on the right hand side of the momentum equations are computed first for the zonal component of momentum given by Equation (2.1) and then for the meridional component of momentum given by Equation (2.2). This is accomplished by a "do $n = 1, 2$" loop extending to Section 11.11.9. All workspace arrays used in the momentum operators are re-computed for each component of momentum. Therefore all momentum operators can only be used from within this n loop which implies all diagnostics using these operators are issued from within this n loop.

11.11.3 Advective and Diffusive fluxes

Advective and diffusive fluxes across the north, east, and bottom faces of U cells within the memory window are calculated for use by the advection and diffusion operators described in Section 11.11.5. These are the canonical forms. The actual form for the diffusive flux may differ depending upon which option is enabled. Note that advective flux is missing a factor of $\frac{1}{2}$ which is incorporated in the advection operator for speed reasons. Also for reasons of speed, a cosine factor has been absorbed into the definition of $adv\_vnu_{i,k,j}$. Limits on j are usually $jsmw = 2$ and $jemw = jmw - 1$, except for the last memory window, which may be only partially full. Meridional operators require meridional flux to be computed for $j = 1$, which is why the lower limit of the j index is not identical for all fluxes. Refer to Figures 3.3 and 3.7:

[27] This is the depth of the grid point within the first T cell and U cell.

$$\begin{aligned} adv\_fe_{i,k,j} &= 2adv\_veu_{i,k,j} \cdot \overline{u_{i,k,j,n,\tau}}^{\lambda} \\ &= adv\_veu_{i,k,j} \cdot (u_{i,k,j,n,\tau} + u_{i+1,k,j,n,\tau}) \, , j = jsmw, jemw \end{aligned} \tag{11.84}$$

$$\begin{aligned} adv\_fn_{i,k,j} &= 2adv\_vnu_{i,k,j} \cdot \overline{u_{i,k,j,n,\tau}}^{\phi} \\ &= adv\_vnu_{i,k,j} \cdot (u_{i,k,j,n,\tau} + u_{i,k,j+1,n,\tau}) \, , j = 1, jemw \end{aligned} \tag{11.85}$$

$$\begin{aligned} adv\_fb_{i,k,j} &= 2adv\_vbu_{i,k,j} \cdot \overline{u_{i,k,j,n,\tau}}^{z} \\ &= adv\_vbu_{i,k,j} \cdot (u_{i,k,j,n,\tau} + u_{i,k+1,j,n,\tau}) \, , j = jsmw, jemw \end{aligned} \tag{11.86}$$

$$\begin{aligned} diff\_fe_{i,k,j} &= \frac{diff\_ceu_{i,k,j}}{\cos\phi^U_{jrow}} \delta_\lambda(u_{i,k,j,n,\tau-1}) \\ &= diff\_ceu_{i,k,j} \cdot \frac{u_{i+1,k,j,n,\tau-1} - u_{i,k,j,n,\tau-1}}{\cos\phi^U_{jrow} dxt_{i+1}} \, , j = jsmw, jemw \end{aligned} \tag{11.87}$$

$$\begin{aligned} diff\_fn_{i,k,j} &= diff\_cnu_{i,k,j} \cdot \cos\phi^T_{jrow+1} \delta_\phi(u_{i,k,j,n,\tau-1}) \\ &= diff\_cnu_{i,k,j} \cdot \cos\phi^T_{jrow+1} \frac{u_{i,k,j+1,n,\tau-1} - u_{i,k,j,n,\tau-1}}{dyt_{jrow+1}} \, , j = 1, jemw \end{aligned} \tag{11.88}$$

$$\begin{aligned} diff\_fb_{i,k,j} &= diff\_cbu_{i,k,j} \cdot \delta_z(u_{i,k,j,n,\tau-1}) \\ &= diff\_cbu_{i,k,j} \cdot \frac{u_{i,k,j,n,\tau-1} - u_{i,k+1,j,n,\tau-1}}{dzw_k} \, , j = jsmw, jemw \end{aligned} \tag{11.89}$$

The top and bottom boundary conditions are applied to the vertical fluxes where the bottom cell is at level $kb = kmu_{i,jrow}$. Also, the advective flux through the bottom of the last level $k = km$ is set to zero since there can be no bottom slope at the bottom of the deepest level.

$$kb = kmu_{i,jrow} \tag{11.90}$$

$$diff\_fb_{i,0,j} = smf_{i,j,n} \tag{11.91}$$

$$diff\_fb_{i,kb,j} = bmf_{i,j,n} \tag{11.92}$$

$$adv\_fb_{i,0,j} = adv\_vbu_{i,0,j} \cdot (u_{i,1,j,n,\tau} + u_{i,1,j,n,\tau}) \tag{11.93}$$

$$adv\_fb_{i,km,j} = 0.0 \tag{11.94}$$

11.11.4 Source terms

There is the possibility of introducing sources into the momentum equation by enabling option *source_term*. If this option is enabled, the source term is initialized to zero for each component of the momentum equations.

$$source_{i,k,j} = 0.0 \tag{11.95}$$

Adding new sources or sinks to the momentum equations is a matter of calculating new terms and adding them to $source_{i,k,j}$. For instance, suppose it was desirable to add a Rayleigh damping in some region of the domain. The damping could be calculated as

$$rayleigh_{i,k,j,n} = -\gamma \cdot u_{i,k,j,n,\tau-1} \cdot ray\_mask_{i,jrow} \tag{11.96}$$

Where $ray\_mask_{i,jrow}$ is a mask of ones and zeros to limit the region of damping and γ is the reciprocal of the damping time scale in seconds. This term could be added to the momentum equations using

$$source_{i,k,j} = source_{i,k,j} + rayleigh_{i,k,j,n} \tag{11.97}$$

and additional sources and sinks could be stacked in a likewise manner.

11.11.5 Momentum operators

Finite difference momentum operators as described in Section 5.1 are used for various terms when solving for the time derivatives in Equations (2.1) and (2.2). These operators are also used in diagnostics throughout MOM 2. Note that the arrays embedded within the operators must be defined when the operators are invoked. The operators are defined in file *fdifm.h* and their canonical forms are given by:

$$\begin{aligned}
DIFF_Ux_{i,k,j} &= \frac{diff_fe_{i,k,j} - diff_fe_{i-1,k,j}}{\cos\phi^U_{jrow} \cdot dxu_i} && (11.98)\\
DIFF_Uy_{i,k,j} &= \frac{diff_fn_{i,k,j} - diff_fn_{i,k,j-1}}{\cos\phi^U_{jrow} \cdot dyu_{jrow}} && (11.99)\\
DIFF_Uz_{i,k,j} &= \frac{diff_fb_{i,k-1,j} - diff_fb_{i,k,j}}{dzt_k} && (11.100)\\
DIFF_metric_{i,k,j,n} &= A_m \frac{1-\tan^2\phi^U_{jrow}}{radius^2} u_{i,k,j,n,\tau-1} && (11.101)\\
&\mp A_m \frac{2\sin\phi^U_{jrow}}{radius \cdot \cos^2\phi^U_{jrow}} \frac{u_{i+1,k,j,3-n,\tau-1} - u_{i-1,k,j,3-n,\tau-1}}{dxt_i + dxt_{i+1}} &&\\
ADV_Ux_{i,k,j} &= \frac{adv_fe_{i,k,j} - adv_fe_{i-1,k,j}}{2\cos\phi^U_{jrow} dxu_i} && (11.102)\\
ADV_Uy_{i,k,j} &= \frac{adv_fn_{i,k,j} - adv_fn_{i,k,j-1}}{2\cos\phi^U_{jrow} dyu_{jrow}} && (11.103)\\
ADV_Uz_{i,k,j} &= \frac{adv_fb_{i,k-1,j} - adv_fb_{i,k,j}}{2dzt_k} && (11.104)\\
ADV_metric_{i,k,j,n} &= \mp\frac{\tan\phi^U_{jrow}}{radius} u_{i,k,j,1,\tau} \cdot u_{i,k,j,3-n,\tau} && (11.105)\\
& && (11.106)
\end{aligned}$$

where the earth's rotation rate[28] $\Omega = \frac{\pi}{43082.0} sec^{-1}$, and the mean radius of the earth is given by $radius = 6370.0x10^5 cm$. The $\mp$ indicates a minus sign for $n = 1$ and a plus sign for $n = 2$.

In terms of the above operators, the finite difference counterpart to Equation (2.10) plus the advection metric term in Equations (2.1) and (2.2) is given as

$$\mathcal{L}^U(u_{i,k,j,n,\tau}) = ADV\_Ux_{i,k,j} + ADV\_Uy_{i,k,j} + ADV\_Uz_{i,k,j} + ADV\_metric_{i,k,j,n} \tag{11.107}$$

Coriolis treatment

[28] The 43082.0 sec is arrived at assuming approximately $86400 * (1 - \frac{1}{366}) = 86164$ seconds in one sidereal day. The 366 is used to account for a 1 day co-rotation.

The Coriolis term may be handled explicitly or semi-implicitly. Refer to Section 15.11.3 for a discussion of the semi-implicit treatment of the Coriolis term and when it is warranted. When solving explicitly, the Coriolis term is given by

$$CORIOLIS_{i,k,j,n} = \pm 2\Omega \sin \phi^{U}_{jrow} \cdot u_{i,k,j,3-n,\tau} \tag{11.108}$$

When solving implicitly, option *damp_inertial_oscillation* is enabled and the semi-implicit Coriolis factor *acor* is used. In the semi-implicit treatment of the Coriolis term, the explicit part is given by

$$CORIOLIS_{i,k,j,n} = \pm(1 - acor) \cdot 2\Omega \sin \phi^{U}_{jrow} \cdot u_{i,k,j,3-n,\tau-1} \tag{11.109}$$

and the handling of the implicit part is detailed below in Section 11.11.10. For both explicit and semi-implicit treatment, the $\pm$ indicates a plus sign for $n = 1$ and a minus sign for $n = 2$.

11.11.6 Solving for the time derivative of velocity

The vertical diffusion term in the momentum equations may be solved explicitly or implicitly. The implicit treatment is appropriate when large diffusion coefficients (or small vertical grid spacing) limit the time step.

Explicit vertical diffusion

Using the above operators (and flux terms calculated previously), the time derivative of velocity is computed by including all terms except the unknown surface pressure gradients exerted by the rigid lid at the ocean surface. If the vertical diffusion term is handled explicitly, then the time derivative of velocity is given by:

$$\begin{aligned}\delta_\tau(u^{\star}_{i,k,j,n}) = \ & (DIFF\_Ux_{i,k,j} + DIFF\_Uy_{i,k,j} + DIFF\_Uz_{i,k,j} \\ & + DIFF\_metric_{i,k,j,n} - ADV\_Ux_{i,k,j} - ADV\_Uy_{i,k,j} \\ & - ADV\_Uz_{i,k,j} + ADV\_metric_{i,k,j,n} - grad\_p_{i,k,j,n} \\ & + CORIOLIS_{i,k,j,n} + source_{i,k,j}) \cdot umask_{i,k,j}\end{aligned} \tag{11.110}$$

Implicit vertical diffusion

When solving the vertical diffusion implicitly, option *implicitvmix* must be enabled and the implicit diffusion factor *aidif* is used. The vertical diffusion operator in Equation (11.100) is replaced by

$$DIFF\_Uz_{i,k,j} = (1 - aidif) \cdot \frac{diff\_fb_{i,k-1,j} - diff\_fb_{i,k,j}}{dzt_k} \tag{11.111}$$

and the time derivative of velocity is computed in two steps. The first computes the derivative as

$$\begin{aligned}\delta_\tau(u^{\star\star}_{i,k,j,n}) = \ & (DIFF\_Ux_{i,k,j} + DIFF\_Uy_{i,k,j} + DIFF\_Uz_{i,k,j} \\ & + DIFF\_metric_{i,k,j,n} - ADV\_Ux_{i,k,j} - ADV\_Uy_{i,k,j} \\ & - ADV\_Uz_{i,k,j} + ADV\_metric_{i,k,j,n} - grad\_p_{i,k,j,n} \\ & + CORIOLIS_{i,k,j,n} + source_{i,k,j}) \cdot umask_{i,k,j}\end{aligned} \tag{11.112}$$

and an intermediate velocity $u^{\star\star}_{i,k,j,n}$ is constructed as

$$u^{\star\star}_{i,k,j,n} = u_{i,k,j,n,\tau-1} + 2\Delta\tau \cdot \delta_t(u^{\star\star}_{i,k,j,n}) \tag{11.113}$$

The next step solves the following equation by inverting a tri-diagonal matrix as given in pages 42 and 43 of Numerical Recipes in Fortran (1992)[29].

$$u^{\star}_{i,k,j,n,\tau+1} = u^{\star\star}_{i,k,j,n} + 2\Delta\tau \cdot \delta_z(aidif \cdot diff\_cbu_{i,k,j} \cdot \delta_z(u^{\star}_{i,k,j,n,\tau+1})) \tag{11.114}$$

The time derivative of velocity is then given by

$$\delta_\tau(u^{\star}_{i,k,j,n}) = \frac{u^{\star}_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau} \tag{11.115}$$

The recommendation from Haltiner and Williams (1980) is for $aidif = 0.5$ which gives the Crank-Nicholson implicit scheme which is always stable. This setting is supposed to be the most accurate one. However, this is not the case when solving a time dependent problem as discussed in Section15.19.4. Note that in the model code, $\delta_\tau(u^{\star}_{i,k,j,n})$ is temporarily stored in $u_{i,k,j,n,\tau+1}$ as a space saving measure until the internal modes are computed as indicated below.

11.11.7 Diagnostics

At this point, diagnostics are computed for velocity component n using previously defined operators. For a description of the diagnostics, refer to Chapter 18.

11.11.8 Vertically averaged time derivatives of velocity

The vertical average of the time derivative of the velocity components is computed as the last item within the loop over velocity components. The vertical averages are constructed as

$$zu_{i,jrow,n} = \frac{1}{H_{i,jrow}} \cdot \sum_{k=1}^{km} dzt_k \cdot \delta_\tau(u^{\star}_{i,k,j,n}) \tag{11.116}$$

and later used to construct the forcing for the external mode.

11.11.9 End of momentum components

At this point, the "do $n = 1,2$" loop issued in Section 11.11.2 is ended. After completion of this loop, all right hand terms in Equations (2.1) and (2.2) have been computed.

11.11.10 Computing the internal modes of velocity

The internal modes of velocity are solved using a variety of time differencing schemes (leapfrog, forward, Euler backward) as outlined in Section 11.1. When the time derivatives of both velocity components have been computed using Equations (11.110) through (11.115), the uncorrected velocities[30] $u'_{i,k,j,1,\tau+1}$ and $u'_{i,k,j,2,\tau+1}$ are computed. A complication arises due to the vertical diffusion term which may be solved explicitly or implicitly. The implicit treatment is appropriate

[29] The surface boundary condition $smf_{i,j,n}$ should be at $\tau+1$ but that value isn't known. Instead, the τ value is used. In MOM 1, The Richardson number in *ppmix.F* was calculated at $tau-1$ in the implicit case. It should be at $\tau+1$ but that is unknown. In MOM 2, the Richardson number can easily be changed to τ. The effect of this on the solution should be small but has not been explored.

[30] Velocities with incorrect vertical means due to the missing unknown surface pressure.

when large vertical viscosity coefficients (or small vertical grid spacing) limit the time step. The following discussion pertains to the leapfrog scheme. The other schemes amount to changing $2\Delta t$ to Δt in what follows.

Explicit Coriolis treatment

If the Coriolis term is treated explicitly, the solution is given by

$$u'_{i,k,j,1,\tau+1} = u_{i,k,j,1,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,1}) \tag{11.117}$$
$$u'_{i,k,j,2,\tau+1} = u_{i,k,j,2,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,2}) \tag{11.118}$$

Semi-implicit Coriolis treatment

If the Coriolis term is treated semi-implicitly (refer to Section 15.11.3 for a discussion of the semi-implicit treatment and when it is warranted), the equations are

$$u'_{i,k,j,1,\tau+1} - 2\Delta\tau \cdot acor \cdot f_{jrow} \cdot u'_{i,k,j,2,\tau+1} = u_{i,k,j,1,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,1}) \tag{11.119}$$
$$u'_{i,k,j,2,\tau+1} + 2\Delta\tau \cdot acor \cdot f_{jrow} \cdot u'_{i,k,j,1,\tau+1} = u_{i,k,j,2,\tau-1} + 2\Delta\tau \cdot \delta_\tau(u^\star_{i,k,j,2}) \tag{11.120}$$

and the solution, after a little algebra, is given by

$$u'_{i,k,j,1,\tau+1} = u_{i,k,j,1,\tau-1} + 2\Delta\tau \cdot \frac{\delta_\tau(u^\star_{i,k,j,1}) + (2\Delta\tau \cdot f \cdot acor)\cdot \delta_\tau(u^\star_{i,k,j,2})}{1 + (2\Delta\tau \cdot f \cdot acor)^2} \tag{11.121}$$
$$u'_{i,k,j,2,\tau+1} = u_{i,k,j,2,\tau-1} + 2\Delta\tau \cdot \frac{\delta_\tau(u^\star_{i,k,j,2}) - (2\Delta\tau \cdot f \cdot acor)\cdot \delta_\tau(u^\star_{i,k,j,1})}{1 + (2\Delta\tau \cdot f \cdot acor)^2} \tag{11.122}$$

The pure internal modes $\hat{u}_{i,k,j,n,\tau+1}$ are then calculated by removing the incorrect vertical means

$$\hat{u}_{i,k,j,n,\tau+1} = u'_{i,k,j,n,\tau+1} - \frac{1}{H_{i,jrow}} \cdot \sum_{k=1}^{km} dzt_k \cdot u'_{i,k,j,n,\tau+1} \tag{11.123}$$

In the code, $\hat{u}_{i,k,j,n,\tau+1}$ is stored in $u_{i,k,j,n,\tau+1}$. The full velocity is actually not available until the next timestep when the external mode velocity is added in subroutine *loadmw*. This completes the formal solution of the internal mode velocities.

11.11.11 Filtering

If the time step constraint imposed by convergence of meridians is to be relaxed, the internal modes are filtered by one of two techniques: Fourier filtering (Bryan, Manabe, Pacanowski 1975) enabled by option *fourfil*, or finite impulse response filtering enabled by option *firfil*. Both should be used with caution and only when necessary at high latitudes. *firfil* is much faster than *fourfil*.

11.11.12 Accumulating $sbcocn_{i,jrow,m}$

Finally, if required as a surface boundary condition for the atmosphere (Pacanowski 1987)[31], surface velocity components $u_{i,1,j,n,\tau}$ are accumulated and averaged over one time segment. The result is stored in $sbcocn_{i,jrow,m}$where m relates the ordering of n in the array of surface boundary conditions as discussed in Section 10.3.

[31]Although the paper explored the affect on SST primarily through Ekman divergence, there may also be an impact on evaporative flux in certain regions where the ocean surface velocities are large.

11.12 End of computation within Memory Window

As each group of latitudes is solved within the memory window, the updated baroclinic velocities and tracers are written to disk as explained in Section 3.3.1 and the next group issued in Section 11.3 is started. After all memory windows have been solved, the forcing for the barotropic velocity has been constructed in Section 11.11.8 and the external mode equations can now be solved.

11.13 tropic (computes external mode velocities)

After the baroclinic velocities have been computed for all latitude rows in the domain, the barotropic or external mode velocity can be computed. Subroutine *tropic*[32] computes the barotropic velocity $\bar{u}_{i,k,j,n,\tau+1}$ for $n = 1$ and 2 by one of three options: *stream_function* described in Section 16.1, *prognostic_surface_pressure* described in Section 16.2, or *implicit_free_surface* described in Section 16.3. Refer to the above sections for details.

11.14 diago

After the external mode has been solved, the remaining diagnostics can be calculated and written out. This is done in subroutine *diago*[33] and described in Chapter 18.

[32] Contained in file tropic.F.
[33] Contained in file diago.F.

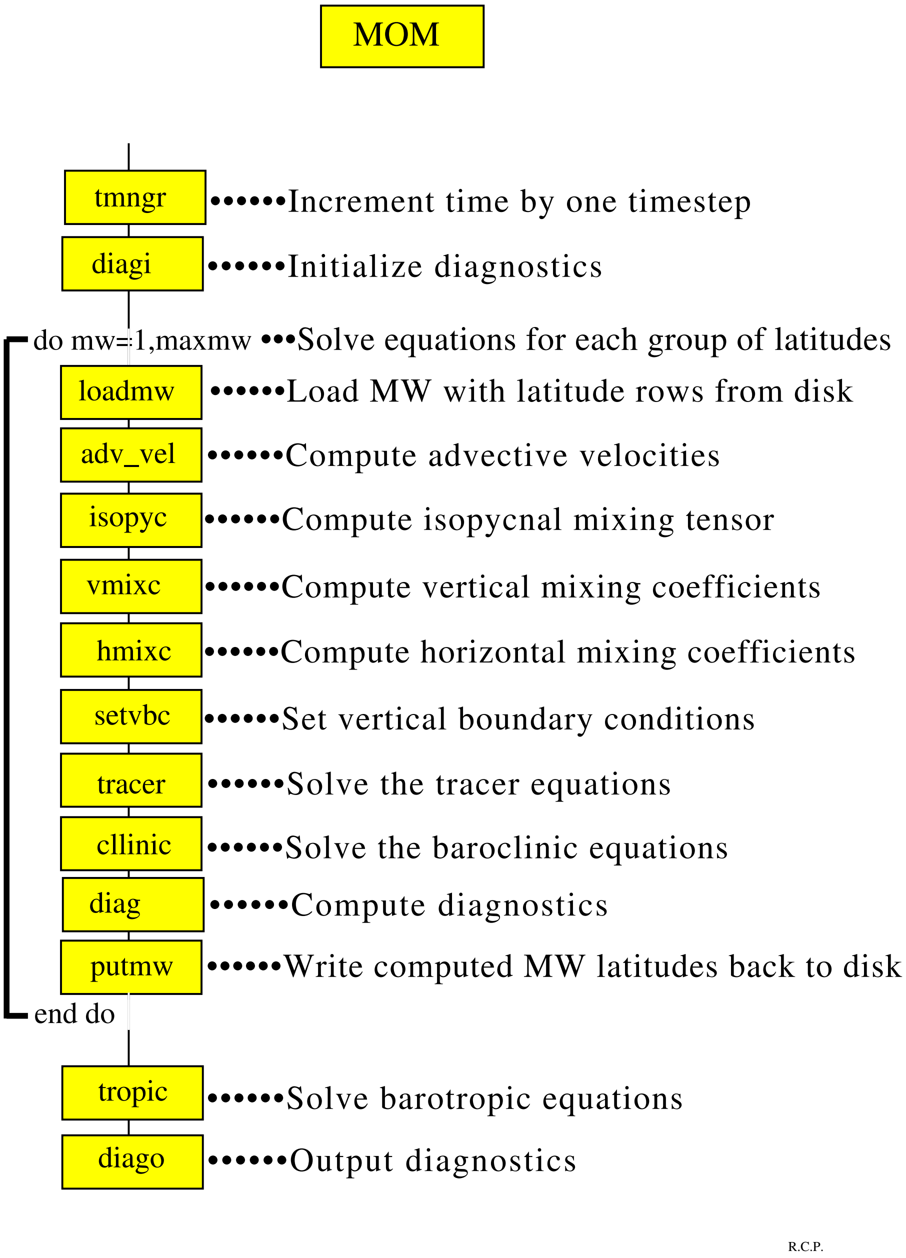


Figure 11.1: Flowchart for subroutine *mom.F* showing order of components

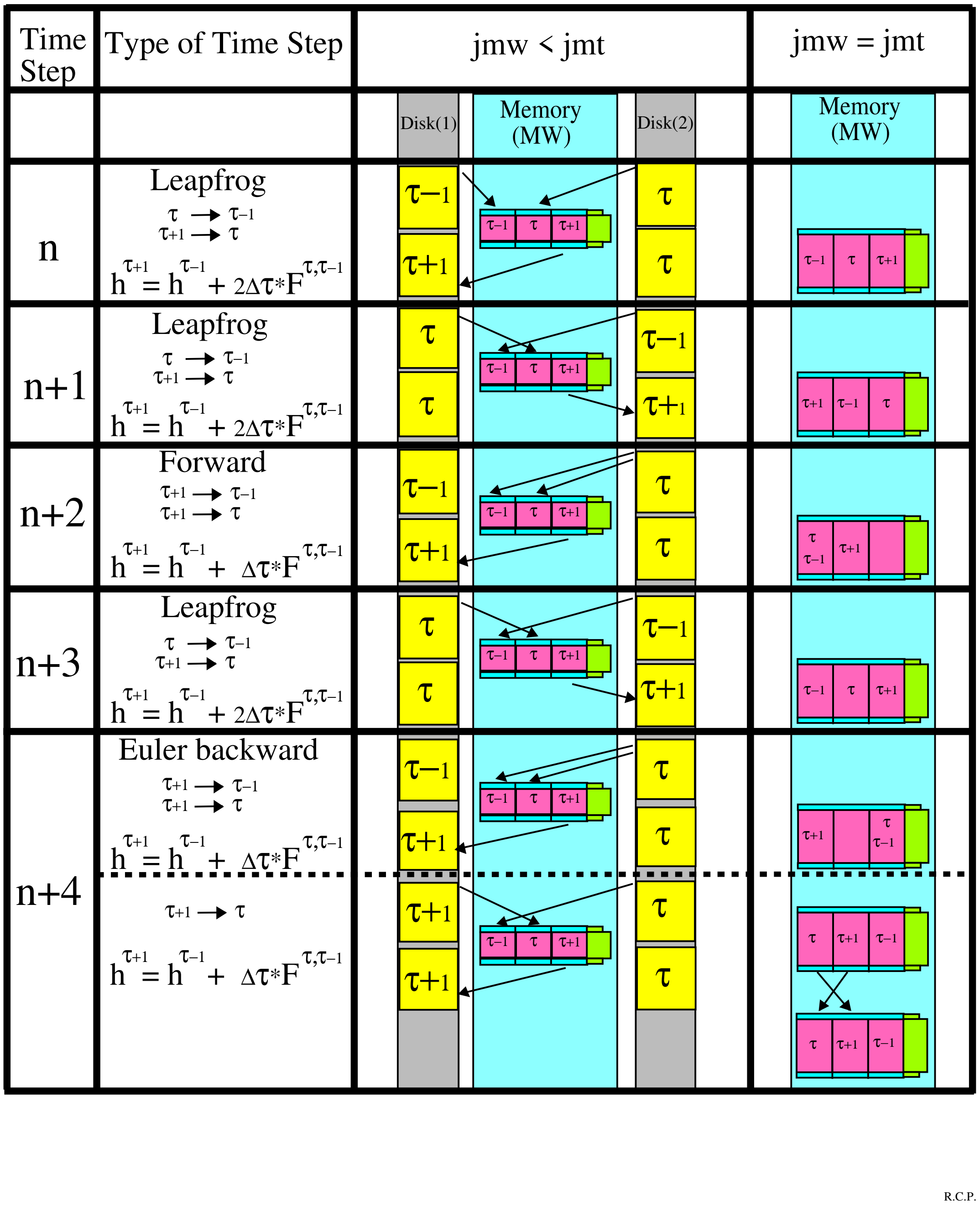


Figure 11.2: Time discretization used in MOM 2 when memory window is partially opened and fully opened. components

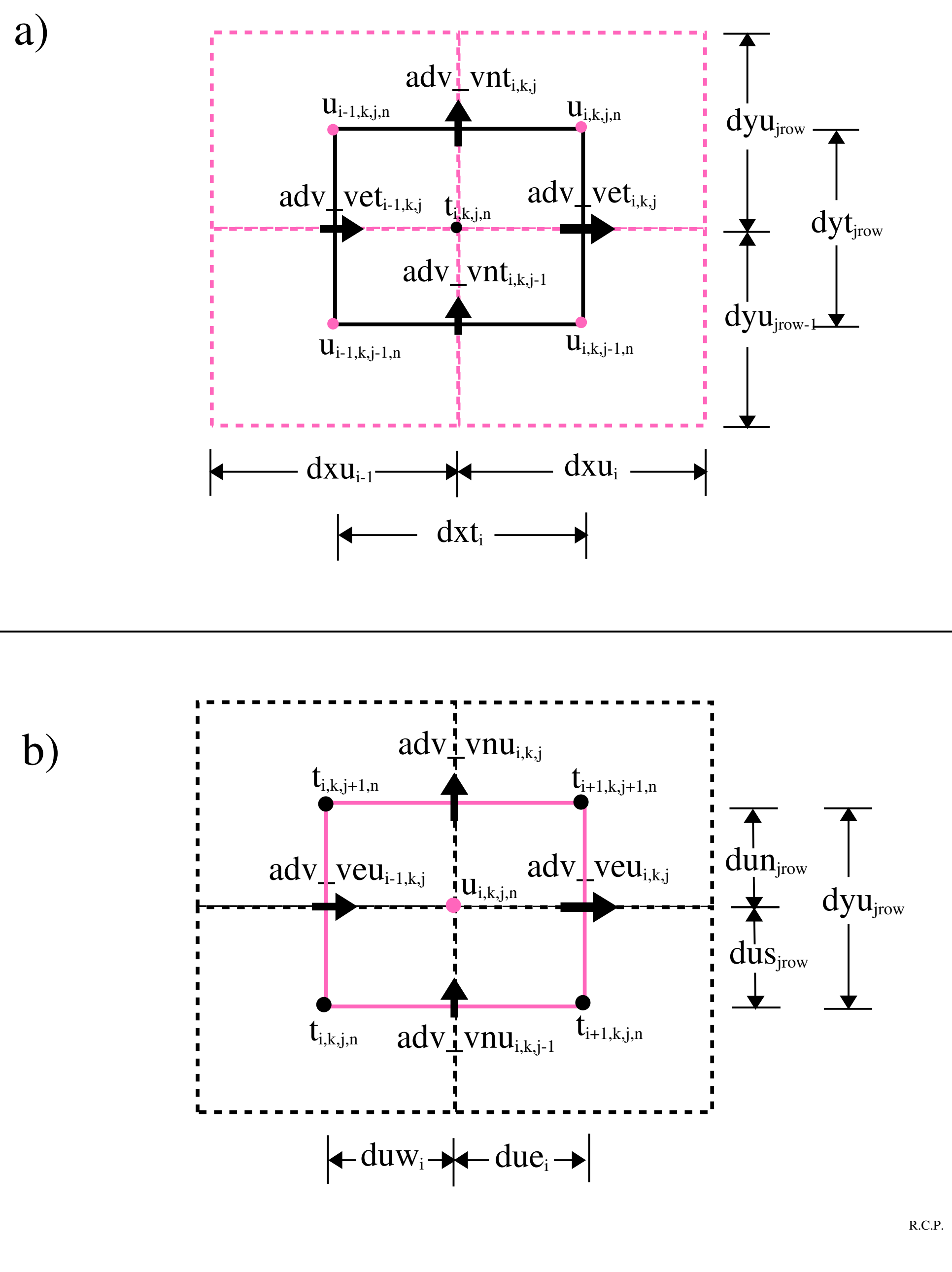


Figure 11.3: a) Advective velocities on a T cell. b) Advective velocities on a U cell.

Chapter 12

Discrete energetics

Bryan (1969) noted that it is desirable to conserve first and second moments in the absence of dissipative effects to prevent systematic errors that accumulate in time and to avoid non-linear instability. In Section 2.2 it was shown that energy is conserved by the non-linear terms in the continuous equations. In Section 12.1, the finite difference counterpart of that argument relating to the ideas of Bryan on conservation of first and second moments will be given. The argument follows that given by Bryan(1969) and Semtner(1974).

In addition to non-linear terms conserving energy, it must be shown that the work done by the pressure terms is equal to the work done by buoyancy. This result is given in Section 12.3.

12.1 Non-linear terms

Let q be any three dimensional prognostic quantity at the centers of grid cells $q_1, q_2, q_3, \cdots q_N$ with volumes $\alpha_1, \alpha_2, \alpha_3, \cdots \alpha_N$ respectively. What will be shown is that the finite difference analogue of the advection equation

$$\partial_t(q) + \nabla \cdot (\vec{v}q) = 0 \tag{12.1}$$

will conserve the two quantities

$$\sum_{n=1}^{N} \alpha_n q_n \qquad \sum_{n=1}^{N} \alpha_n q_n^2 \tag{12.2}$$

in time. Using Stokes' theorem to change volume integrals into surface integrals, the finite difference form of the volume integral of Equation (12.1) is

$$\sum_{n=1}^{N} (\alpha_n \frac{dq_n}{dt} + \sum_{i=1}^{6} A_n^i V_n^i \,\overline{q}_n^i) = 0 \tag{12.3}$$

with

$$\overline{q}_n^i = \frac{(q_n + q_n^i)}{2} \tag{12.4}$$

where q_n is the value in the nth cell, A_n^i is the area of the ith face of that cell, V_n^i is the velocity normal to the ith face, and $\overline{q}_n^i$ is the average value of q on the ith face with q_n^i being the value in the other cell that shares that face. Using Equations (12.3) and (12.4), the time change of the volume integral of q_n when integrated over all cells can be written as.

$$\frac{d}{dt}\sum_{n=1}^{N}\alpha_n q_n = \sum_{n=1}^{N}\alpha_n \frac{dq_n}{dt} = -\sum_{n=1}^{N}\sum_{i=1}^{6} A_n^i V_n^i \frac{(q_n + q_n^i)}{2} = 0 \tag{12.5}$$

To obtain this result, the normal velocity V_n^i must be zero at land boundaries. Note that V_n^i is anti-symmetric on the common face between adjacent cells due to the change in sign of the normal vector, whereas the average $\overline{q}_n^i$ is symmetric with respect to adjacent cells. This result indicates that integrals of first moments are conserved by Equation (12.4). It should be further noted that constructing $\overline{q}_n^i$ by any linear interpolation of q_n and q_n^i will also conserve first moments.

Using similar techniques, the time change of the volume integral of q^2 is given by:

$$\frac{d}{dt}\sum_{n=1}^{N}\alpha_n q_n^2 = 2\sum_{n=1}^{N}\alpha_n q_n \frac{dq_n}{dt} = -\sum_{n=1}^{N} q_n^2 (\sum_{i=1}^{6} A_n^i V_n^i) - \sum_{n=1}^{N}\sum_{i=1}^{6} A_n^i V_n^i q_n q_n^i = 0 \tag{12.6}$$

The third term in Equation (12.6) vanishes because the fluid is incompressible and continuity applies at each cell. The fourth term vanishes because $q_n q_n^i$ is symmetric and V_n^i is anti-symmetric for pairs of adjacent cells (again due to the change in sign of the normal vector) and $V_n^i = 0$ on all land boundaries. It should be noted here that, in particular, constructing $\overline{q}_n^i$ as $\beta \cdot q_n + \gamma \cdot q_n^i$ where $\beta \neq \gamma \neq 1/2$ will <u>not</u> conserve second moments because neither the third nor fourth term will vanish. MOM 2 uses Equation (12.4) to construct averaged quantities and therefore conserves first and second moments.

Since q in the above equations stands for any prognostic quantity, non-linear terms are conserved for each component of momentum. However, in sperical coordinates both components must be considered to eliminate the work done by the non-linear metric term. For example, let $q_n = u_{i,k,k,n,\tau}$ and construct the volume integral of the work done by the non-linear terms using operators defined in Section 11.11.5.

$$A = \sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2} u_{i,k,j,n,\tau}\Big[\, ADV\_Ux_{i,k,j} + ADV\_Uy_{i,k,j} + ADV\_Uz_{i,k,j} + ADV\_metric_{i,k,j,n}\Big]\cos\phi_{jrow}^U dxu_i\ dyu_{jrow}\ dzt_k \tag{12.7}$$

Expanding the advection operators yield the work done in terms of fluxes on all faces of the U cells.

$$A = \sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2} u_{i,k,j,n,\tau}\Big[\, \frac{adv\_fe_{i,k,j} - adv\_fe_{i-1,k,j}}{2\cos\phi_{jrow}^U dxu_i} + \frac{adv\_fn_{i,k,j} - adv\_fn_{i,k,j-1}}{2\cos\phi_{jrow}^U dyu_{jrow}} + \frac{adv\_fb_{i,k-1,j} - adv\_fb_{i,k,j}}{2dzt_k} \mp \frac{\tan\phi_{jrow}^U}{radius} u_{i,k,j,n,\tau}\cdot u_{i,k,j,3-n,\tau}\Big]\cos\phi_{jrow}^U dxu_i\ dyu_{jrow}\ dzt_k \tag{12.8}$$

Expanding the advective fluxes further leads to

$$A = \sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2} u_{i,k,j,n,\tau}\Big[\, (adv\_veu_{i,k,j}\frac{u_{i,k,j,n,\tau} + u_{i+1,k,j,n,\tau}}{2}$$

$$
\begin{aligned}
- \quad & adv\_veu_{i-1,k,j}\frac{u_{i,k,j,n,\tau}+u_{i-1,k,j,n,\tau}}{2})dyu_{jrow} \\
+ \quad & (adv\_vnu_{i,k,j}\frac{u_{i,k,j,n,\tau}+u_{i,k,j+1,n,\tau}}{2} \\
- \quad & adv\_vnu_{i,k,j-1}\frac{u_{i,k,j,n,\tau}+u_{i,k,j-1,n,\tau}}{2})dxu_i \\
+ \quad & (adv\_vbu_{i,k,j}\frac{u_{i,k,j,n,\tau}+u_{i,k+1,j,n,\tau}}{2} \\
- \quad & adv\_vbu_{i,k-1,j}\frac{u_{i,k,j,n,\tau}+u_{i,k-1,j,n,\tau}}{2})\cos\phi^U_{jrow}dxu_i dyu_{jrow} \\
& \mp\frac{\tan\phi^U_{jrow}}{radius}u_{i,k,j,n,\tau}\cdot u_{i,k,j,3-n,\tau}\cos\phi^U_{jrow}dxu_i\ dyu_{jrow}\ dzt_k \\
& \Big]
\end{aligned}
\tag{12.9}
$$

where the minus sign in the mertic term is used when $n = 1$ and the plus sign is used when $n = 2$. Using the condition that the normal velocity on land cells is zero allows rearrangement of terms yielding the form

$$
\begin{aligned}
A = \frac{1}{2}\sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2}u_{i,k,j,n,\tau}\Big[\quad & (adv\_veu_{i,k,j} - adv\_veu_{i-1,k,j})dyu_{jrow} \\
+ \quad & (adv\_vnu_{i,k,j} - adv\_vnu_{i,k,j-1})dxu_i \\
+ \quad & (adv\_vbu_{i,k-1,j} - adv\_vbu_{i,k,j})\cos\phi^U_{jrow}dxu_i dyu_{jrow}\Big] \\
+u_{i,k,j,n,\tau} \quad \Big[\quad & u_{i+1,k,j,n,\tau}\cdot adv\_veu_{i+1,k,j} - u_{i-1,k,j,n,\tau}\cdot adv\_veu_{i-1,k,j} \\
+ \quad & u_{i,k,j+1,n,\tau}\cdot adv\_vnu_{i,k,j+1} - u_{i,k,j-1,n,\tau}\cdot adv\_vnu_{i,k,j-1} \\
+ \quad & u_{i,k-1,j,n,\tau}\cdot adv\_vbu_{i,k-1,j} - u_{i,k+1,j,n,\tau}\cdot adv\_vbu_{i,k,j} \\
\Big] \quad & \cos\phi^U_{jrow}dxu_i dyu_{jrow}dzt_k \\
= \quad & 0
\end{aligned}
\tag{12.10}
$$

Notice the similarity between Equations (12.10) and (12.6). The first term in brackets vanishes because mass continuity applies to each cell. Summing the second term in brackets results in zero if the normal velocity component is zero on land cells and the previous arguments involving symmetry are applied. It should be noted that if $u_{i,k,j,n,\tau}$ in Equation (12.7) is replaced by internal and external modes $\hat{u}_{i,k,j,n,\tau}$ and $\overline{u}_{i,j,n,\tau}$, then Equation (12.10) will not vanish for the internal mode component of the work. Nor will it vanish for the external mode component of the work. Both components must be added for the sum to vanish.

12.2 Work done by Coriolis terms

There is no contribution from integrating the work done by the Coriolis force. The summation is

$$
C = \sum_{i=2}^{imt-1}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{n=1}^{2}u_{i,k,j,n,\tau}\cdot 2\Omega\sin\phi^U_{jrow}\cdot u_{i,k,j,3-n,\tau} = 0 \tag{12.11}
$$

Again, only use of the full velocity guarantees that the sum will vanish.

12.3 Work done by pressure terms

In Section 2.2.1, it was shown that the work done by pressure forces equals the work done by buoyancy in the continuous equations when integration is over the entire domain. The finite difference result is also equal to the work done by buoyancy. For this derivation, the definition of variables, indices and the relation between *jrow* and j as given in Section 5.2.1 will be used. In the terminology of MOM 2, the work done by presure forces summed over all ocean U cells is given by

$$-\frac{1}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\left(u_{i,k,j,1,\tau}\cdot\delta_\lambda(\overline{p_{i,k,j}}^{\phi})+u_{i,k,j,2,\tau}\cdot\delta_\phi(\overline{p_{i,k,j}}^{\lambda})\right)\cdot dxu_i\cdot\cos\phi^U_{jrow}\cdot dyu_{jrow}\cdot dzt_k \tag{12.12}$$

p is defined on T cells. Expanding these terms using Equations (11.6), (11.7), (11.3), (11.4) and taking the area element of a U cell as $A_{i,jrow} = dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow}$ yields

$$\begin{aligned}
-\frac{1}{\rho_\circ}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\Big(& u_{i,k,j+1,1,\tau}\cdot A_{i,jrow+1}\cdot dzt_k\cdot\frac{p_{i+1,j+2}+p_{i+1,j+1}-p_{i,j+2}-p_{i,j+1}}{2dxu_i\cdot\cos\phi^U_{jrow+1}}\\
+ & u_{i+1,k,j+1,1,\tau}\cdot A_{i+1,jrow+1}\cdot dzt_k\cdot\frac{p_{i+2,j+2}+p_{i+2,j+1}-p_{i+1,j+2}-p_{i+1,j+1}}{2dxu_{i+1}\cdot\cos\phi^U_{jrow+1}}\\
+ & u_{i,k,j,1,\tau}\cdot A_{i,jrow}\cdot dzt_k\cdot\frac{p_{i+1,j+1}+p_{i+1,j}-p_{i,j+1}-p_{i,j}}{2dxu_i\cdot\cos\phi^U_{jrow}}\\
+ & u_{i+1,k,j,1,\tau}\cdot A_{i+1,jrow}\cdot dzt_k\cdot\frac{p_{i+2,j+1}+p_{i+2,j}-p_{i+1,j+1}-p_{i+1,j}}{2dxu_i\cdot\cos\phi^U_{jrow}}\\
+ & u_{i,k,j+1,2,\tau}\cdot A_{i,jrow+1}\cdot dzt_k\cdot\frac{p_{i+1,j+2}+p_{i,j+2}-p_{i+1,j+1}-p_{i,j+1}}{2dyu_{jrow+1}}\\
+ & u_{i+1,k,j+1,2,\tau}\cdot A_{i+1,jrow+1}\cdot dzt_k\cdot\frac{p_{i+2,j+2}+p_{i+1,j+2}-p_{i+2,j+1}-p_{i+1,j+1}}{2dyu_{jrow+1}}\\
+ & u_{i,k,j,2,\tau}\cdot A_{i,jrow}\cdot dzt_k\cdot\frac{p_{i+1,j+1}+p_{i,j+1}-p_{i+1,j}-p_{i,j}}{2dyu_{jrow}}\\
+ & u_{i+1,k,j,2,\tau}\cdot A_{i+1,jrow}\cdot dzt_k\cdot\frac{p_{i+2,j+1}+p_{i+1,j+1}-p_{i+2,j}-p_{i+1,j}}{2dyu_{jrow}}\\
\Big) &
\end{aligned} \tag{12.13}$$

Because the normal component of velocity is zero on all boundaries, the terms in Equation (12.13) can be rearranged around p. Collecting terms for $p_{i+1,jrow+1}$ and multiplying numerator and denominator by the area of a T cell $dxt_{i+1}\cdot\cos\phi^T_{jrow+1}\cdot dyt_{jrow+1}$ results in

$$\begin{aligned}
&\frac{1}{\rho_\circ}\sum_{jrow=0}^{jmt-1}\sum_{k=1}^{km}\sum_{i=0}^{imt-1} p_{i+1,k,j+1}\cdot dxt_{i+1}\cos\phi^T_{jrow+1}\cdot dyt_{jrow+1}dzt_k\cdot\\
&\left[\frac{adv_vet_{i+1,k,j+1}-adv_vet_{i,k,j+1}}{dxt_{i+1}\cdot\cos\phi^T_{jrow+1}}+\frac{adv_vnt_{i+1,k,j+1}-adv_vnt_{i+1,k,j}}{dyt_{jrow+1}\cdot\cos\phi^T_{jrow+1}}\right]
\end{aligned} \tag{12.14}$$

where the limits of the sum have been adjusted to cover all cells and the advective velocity on the eastern face of the T cell is given by

$$adv\_vet_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot dyu_{jrow} + u_{i,k,j-1,1,\tau} \cdot dyu_{jrow-1}}{2dyt_{jrow}} \tag{12.15}$$

and the advective velocity on the northern face of the T cell is given by

$$adv\_vnt_{i,k,j} = \frac{u_{i,k,j,2,\tau} \cdot dxu_i + u_{i-1,k,j,2,\tau} \cdot dxu_{i-1}}{2dxt_i} \cdot \cos\phi^U_{jrow} \tag{12.16}$$

Note that the finite difference counterpart of the incompressibility Equation (2.5) is arrived at by using the advective velocities on the horizontal faces of T cells

$$\frac{adv\_vet_{i,k,j} - adv\_vet_{i-1,k,j}}{dxt_i \cdot \cos\phi^T_{jrow}} + \frac{adv\_vnt_{i,k,j} - adv\_vnt_{i,k,j-1}}{dyt_{jrow} \cdot \cos\phi^T_{jrow}} + \frac{adv\_vbt_{i,k-1,j} - adv\_vbt_{i,k,j}}{dxt_k} = 0 \tag{12.17}$$

Substituting Equation (12.17) into Equation (12.14) leads to

$$\frac{1}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} p_{i,k,j} \cdot dxt_i \cos\phi^T_{jrow} \cdot dyt_{jrow} dzt_k \cdot \frac{adv\_vbt_{i,k-1,j} - adv\_vbt_{i,k,j}}{dzt_k} \tag{12.18}$$

where the transformation of indices $i+1 \rightarrow i$ and $j+1 \rightarrow j$ is used to center the kernal on $T_{i,k,jrow}$. Because the advective velocity at the surface $adv\_vbt_{i,0,j}$ and bottom face of the deepest ocean cell $adv\_vbt_{i,k=kmt_{i,jrow},j}$ is zero, Equation (12.18) can be re-arranged to yield

$$\frac{1}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=2}^{km} \sum_{i=2}^{imt-1} adv\_vbt_{i,k-1,j} \cdot dxt_i \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot (p_{i,k-1,j} - p_{i,k,j}) \tag{12.19}$$

where once again the limits have been adjusted to cover all cells in the vertical. Using the discrete hydrostatic Equation (11.79) reduces Equation (12.19) to

$$-\frac{1}{\rho_\circ} \sum_{jrow=2}^{jmt-1} \sum_{k=2}^{km} \sum_{i=2}^{imt-1} grav \cdot adv\_vbt_{i,k-1,j} \cdot dxt_i \cos\phi^T_{jrow} \cdot dyt_{jrow} dzt_k \cdot \frac{\rho_{i,k-1,j} + \rho_{i,k,j}}{2} \tag{12.20}$$

which is the work done by buoyancy and the counterpart to Equation (2.33).

Chapter 13

Stevens Open Boundary Conditions

Open boundary conditions have been tested successfully in the *FRAM* model (Stevens 1991) and in the *community modeling effort (CME)*, (e.g. Döscher and Redler 1995). They have also been used in a GFDL-model of the North Atlantic in the framework of *MAST2-DYNAMO* (Dynamo 1994) as well as in a regional model of the subpolar North Atlantic (Redler and Böning 1996). The above mentioned experiments performed at Kiel were based on René Redler's[1] implementation in MOM 1. This chapter was written by Arne Biastoch[2] and describes an implementation of these open boundary conditions at the northern, southern, eastern and/or western boundary for a basin in MOM 2. The approach is based on the methodology of Stevens (1990).

There are two types of open boundary conditions: 'active' in which the interior is forced at inflow points by data prescribed at the boundary and 'passive' in which there is no forcing at the boundary and phenomena generated within the domain can propagate outward without disturbing the interior solution.

At open boundaries, baroclinic velocities are calculated using linearized horizontal momentum equations and the streamfunction is prescribed from other model results or calculated transports (e.g. directly or indirectly from the Sverdrup relation). Thus, the vertical shear of the current is free to adjust to local density gradients. Heat and salt are advected out of the domain if the normal component of the velocity at the boundary is directed outward. When the normal component of the velocity at the boundary is directed inward, heat and salt are either restored to prescribed data ('active' open boundary conditions) or not ('passive' open boundary conditions).

In contrast to the above described 'active' open boundary conditions, 'passive' ones are characterized by not restoring tracers at inflow points. Additionally, a simple Orlanski radiation condition (Orlanski 1976) is used for the streamfunction.

In what follows, the northern and southern open boundaries are detailed. Open boundaries at the eastern and western end of the domain are handled in a similar manner except where noted.

WARNING: Tests for 'active' and 'passive' open boundary conditions are included as options and described below. However, open boundary conditions is not an option which can be simply enabled to see what it does. Not only is data along the open boundaries needed (in the case of 'active' open boundaries) but topography along open boundary points must be chosen carefully. Additionally, prescribing a net tranport along the open boundary (via ψ_b), requires that changes be made in the stream function calculation. Although open boundary

[1] Check the mailing list for e-mail address.

[2] Dept. Theoretical Oceanography, Institut für Meereskunde, Düsternbrooker Weg 20, 24105 Kiel, Germany. Check the mailing list for e-mail address.

conditions have been tested in simple cases, attention must be given to details to insure that specific configurations are working properly. For more details the reader is referred to Stevens (1990) and a 'TO-DO' list given at the end of this chapter.

13.1 Boundary specifications

Geographically, open boundaries are defined at latitude rows $jrow = 2$ and $jrow = jmt - 1$ (or at longitudes given by $i = 2$ and $i = imt - 1$). This is in contrast to Stevens (1990) where the open boundaries reside directly at the sides of the domain. The advantage of this definition made here is that memory management, diagnostics, and the external mode Poisson solver remain unchanged.

During outflow conditions new tracer values are calculated on the boundaries using

$$\frac{\partial T}{\partial t} + \frac{v_{ad} + c_T}{a}\frac{\partial T}{\partial \phi} = F^T \tag{13.1}$$

where T represents the tracer, v_{ad} meridional velocity, c_T phase speed of tracer, a radius of the Earth and ϕ latitude.

During inflow conditions, nothing is done unless the open boundary is 'active'. If 'active', tracer values at the boundary are set using a Newtonian restoring condition on a time scale defined by the researcher.

The baroclinic velocity components at $jrow = 2$ and $jrow = jmt - 2$ are calculated by a linearized momentum equation with only the advective terms being omitted.

Solving for the external mode stream function is similar to solving in a closed domain with walls at $jrow = 1$ and jmt. After each solution, two boundary rows are set ('active' open boundary conditions) or calculated ('passive' open boundary conditions) using a radiation condition after Orlanski (1976)

$$\frac{\partial \psi}{\partial t} = -\frac{c_\psi}{a}\frac{\partial \psi}{\partial \phi} \tag{13.2}$$

Because the Poisson solver calculates the updated stream function from $jrow = 3$ to $jrow = jmt - 2$ (In Stevens (1990) ψ is calculated from from $jrow = 2$ to $jrow = jmt - 1$) the first and the last two boundary rows are equal in ψ. This is desirable because there should be no barotropic velocity tangent to the boundary and a zero derivative across the first two rows assures this.

13.2 Options

The following options are available as *ifdefs*.

obc_south open boundary at the southern wall ($jrow = 2$).

obc_north open boundary at the northern wall ($jrow = jmt - 1$).

obc_west open boundary at the western wall ($i = 2$).

obc_east open boundary at the eastern wall ($i = imt - 1$).

obc is automatically defined if any of the above *ifdefs* are enabled.

orlanski Orlanski radiation condition for 'passive' open boundary conditions . If not set: 'active' open boundary conditions are assumed.

obcparameter write open boundary conditions related parameters into snapshot file (only if **obc_south** and/or **obc_north** are enabled)

obctest configuration for 'passive' open boundary conditions test from Chap.5 of Stevens (1990). Requires **obc_north**, **obc_south** and **orlanski**. The new option **no_sbc** (no surface boundary conditions) must also be enabled.

obctest2 configuration for 'active' open boundary conditions test from Chap.6 of Stevens (1990). Requires **obc_south** . Data for the southern boundary is taken from a run with a larger domain without **obc_south** enabled.

The following *options* have been tested with open boundary conditions

cray_ymp, generate_a_grid, read_my_grid, generate_a_kmt, read_my_kmt, flat_bottom, rectangular_box, timing, restorst, source_term, sponges, simple_sbc, constvmix, consthmix, biharmonic, fullconvect, stream_function, sf_9_point, conjugate_gradient

The following *options* do **not** work:

symmetry with **obc_north,obc_south**; **cyclic** with **obc_west,obc_east**

13.3 New Files

`cobc.F`

This subroutine is used instead of `tracer.F` at eastern and western open boundaries. At boundary cells, new tracer values are calculated using (13.1) for outflow conditions and restored for inflow conditions (only for 'active' open boundary conditions).

- calculate advection velocity vad. If there is flow into the region vad is set to zero which indicates restoring (only for 'active' open boundary conditions).
- calculate phase speed for tracer $c_T = c1s, c1n$. For phase speed into the region c_T is set to zero. A maximum for c_T is limited by the CFL-Criterion, e.g. $-a\Delta\phi \leq c_T \leq 0$ for the southern boundary.
- For northern open boundary conditions $var1$ is calculated in `tracer.F`; otherwise it would pointing outside of the memory window.
- calculate quantities for vertical and lateral diffusion. In either case a laplacian formulation is used at the boundary (a_h is taken from the 'mixing' namelist). Therefore a new definition $DIFF\_Ty\_obc(i,k,j)$ is introduced. $diff\_fe$ and $diff\_fb$ are calculated explicitly.
- calculate values for the meridional advection: ADV_Ty_obc.
- new subroutine `obcsponge` (only active obc): similar to `sponge`, but restoring holds only for flux into the domain (vad).
- calculate new tracer values at the boundaries using equation (13.1)

- diagnostics as in `tracer.F`

`cobc2.F`

This subroutine is used instead of `tracer.F` at northern and southern open boundaries and is similar in most respects to `cobc.F`.

- calculate advection velocity uad.
- calculate phase speed for tracer $c_T = c1w, c1e$.
- Introduce $DIFF\_Ty\_obc(i,k,j)$. $diff\_fe$ and $diff\_fb$ are calculated explicitly.
- calculate values for the meridional advection: ADV_Tx_obc.
- new subroutine `obcsponge2` (only 'active' open boundary conditions).
- calculate new tracer values at the boundaries using a similar equation to (13.1) but for meridional boundaries.
- diagnostic as in `tracer.F`

`addobcpsi.F`

This subroutine is used for prescribing values of ψ at open boundaries in the case of 'active' open boundary conditions.

- decide if a new timestep of prescribed psi values has to be read from the disk (based on `sponge.F`).
- if a time interpolation has to be done then the values are prescribed.

`cobc.h`

Coefficient definitions are given for advective and phase velocities at open boundaries

`obc_data.h`

Definitions are given for open boundary data, pointers and damping coefficients (based on `sponge.h`)

`Extra add-in files`

The following files are included by preprocessor commands only if open boundary conditions are enabled. They are kept in separate files to keep the structure of the model clear.

- `clinic_obc.inc`
- `loadmw_obc.inc`
- `mom_obc.inc`
- `setocn_obc.inc`
- `tracer_obc.inc`
- `tropic1_obc.inc`
- `tropic2_obc.inc`

13.4 Changes in existing subroutines

`iounit.h`

New units for open boundaries have been added.

`size.h`

- set parameters for options *obctest* and *obctest2*
- option *obc* is defined as a general *ifdef* for open boundary conditions
- a minimum for the memory window is set as $jmw = 4$ (only zonal open boundary conditions)

`bcest.F`

Define wind stress and tracer surface values for restoring with option *obctest2*.

`checks.F`

Add consistency checks for allowed combinations of *ifdefs*.

`clinic.F`

Linearize velocity components at boundaries.

- at $jrow = 2$ and $jmt - 2$ add-in `clinic_obc.inc`
- at $i = 2$ and $i = imt - 2$: add-in `clinic_obc.inc`

`grids.F`

Set grid definitions for options *obctest* and *obctest2*.

`hmixc.F`

Set ahc_north, amc_north (for *obc_north*),ahc_south, amc_south (for *obc_south*) to zero at open boundaries. This is done as a precaution to prevent diffusion beyond the open boundary.

`isleperim.F`

Set open boundary cells to land values for Poisson solver. ψ is calculated as for closed boundaries. Values at $jrow = 1, 2, jmt - 1$ and jmt are added later. Similar handling for $i = 1, 2, imt - 1$ and imt.

`loadmw.F`

- set velocity values at $jrow = 1$ and $jrow = jmt - 1$ to prevent diffusion over the boundaries: $u_{i,k,1,n} = u_{i,k,2,n}, u_{i,k,jmt,n} = u_{i,k,jmt-1,n} = u_{i,k,jmt-2,n}$ (for τ and $\tau - 1$): add-in `loadmw_obc.inc`
- set tracer values at $jrow = 1$ and $jrow = jmt$ to prevent diffusion over the boundaries: $t_{i,k,1,n} = t_{i,k,2,n}, t_{i,k,jmt,n} = t_{i,k,jmt-1,n}$ (for τ and $\tau - 1$): add-in `loadmw_obc.inc`
- set rho values at $jrow = jmt$ to prevent errors in the calculation of the hydrostatic pressure gradients at the northern boundary: $\rho_{i,k,jmt,n} = \rho_{i,k,jmt-1,n}$: add-in `loadmw_obc.inc`

`mom.F`

- include `cobc.h`
- include calculations for boundaries: add-in `mom_obc.inc`
  - first memory window (example for options *obc_south* and *obc_north*): for options *obc_west* and *obc_east*, changes are in the routines itself):
    * call `clinic` as before (changes are **in** the subroutine itself)
    * `cobc` for $jrow = 2$
    * `tracer` for the rest of the memory window
  - last memory window:
    * call `clinic` as before (changes are **in** the subroutine itself)
    * `tracer` for the first part of the window
    * `cobc` for $jrow = jmt - 1$

`setocn.F`

- include `cobc.h`, `obc_data.h`
- define option lists for open boundary units: $opt\_obc1, opt\_obcpsi1$
- define *annlevobc* (logical) for annual boundary data (no time interpolation), include in namelist `mixing` and initialize.
- initialize coefficients $var1, c1s, c1n, vad, c1ps, c1pn$: add-in `setocn_obc.inc`
- read open boundaries for T and S (`obc1.mom`) and put into unit `obc1`: add-in `setocn_obc.inc`
- read open boundaries for ψ (`obcpsi1.mom`) and put into unit `obcpsi1`: add-in `setocn_obc.inc`
- subroutine `rowi`: initialize Tracer for options *obctest* and *obctest2*

`setvbc.F`
Set surface boundary conditions to zero for option *obctest* (option *no_sbc* for use with option *simple_sbc*).

`snapit.F`

- include `cobc.h`
- write open boundary conditions parameter: $c1s, c1n, vads, vadn$ (option *obcparameter*, only for zonal open boundary conditions)

`topog.F`
Because the open boundaries resides at $j = 2$ and $j = jmt - 1$ (i similar) for each open boundary condition an extra has to be introduced, identically to the virtual boundary row. So a duplex boundary must enclose the domain — either open or closed but always identically.

- two rows next to the open boundaries are set equal to the boundary row itself. Otherwise in the case of sloping bottom from the boundary to the interior the calculation of the phase velocities would point into bottom and would deliver wrong values! Actually a minimum depth is chosen, e.g. $kmt_{i,2} = kmt_{i,3} = kmt_{i,4} = \min(kmt_{i,2}, kmt_{i,3}, kmt_{i,4})$ for option *obc_south*, but the rearcher is free to change this.
- subroutine `kmtbc`: extrapolate kmt onto open boundaries, e.g. $kmt(i,1) = kmt(i,2)$ or $kmt(i,jmt) = kmt(i,jmt-1)$

`tracer.F`

- include `cobc.h` (only with option *bc_north*)
- at $jmt-2$ calculate $var1$ for `cobc` (only with option *bc_north*): add-in `tracer_obc.inc`
- before calculating new tracer values call `cobc2` for $i = 2$ (option *obc_west*): add-in `tracer_obc.inc`
- calculate new tracer values only for non-boundary rows
- after calculating new tracer values call `cobc2` for $i = imt-1$ (option *obc_east*): add-in `tracer_obc.inc`

`tropic.F`

For the Orlanski (1976) radiation condition, new stream function values at the open boundary are calculated using (13.2). For active open boundaries values for ψ are prescribed.

- for Orlanski radiation deliver *dtts*
- include `cobc.h`, `obc_data.h`
- for Orlanski radiation calculate phase velocities $c_\psi = c1ps, c1pn$: add-in `tropic1_obc.inc`
- calculate new ψ at $jrow = 1,2$ and/or $jrow = jmt-1, jmt$ with equation (13.2) ('passive' open boundary conditions) or prescribe ('active' open boundary conditions, `addobcpsi.F`). For second case *psiwall_south* and/or *psiwall_north* (of `obcpsi`) are used: add-in `tropic2_obc.inc`

`util.F`

Subroutine `setbcx`: extrapolate values onto open boundaries (only with options *obc_west* and *obc_east*)

13.5 Data Preparation Routines

`obc.F`

This routine is based on the PREP_DATA routine `sponge.F` and prepares the open boundary values for T and S. For input it uses the standard Levitus data (prepared by `ic.F`) or ASCII data written by FERRET (Hankin and Denham 1994). The damping factors have to be specified (search for `USER INPUT`). Two files, `obc1.mom` for zonal and `obc2.mom` for meridional open boundaries, are written. Following *ifdefs* can be chosen

makeobc driver

readferret use FERRET-data as input (formatted listing); otherwise Levitus-data are used

obc_north, obc_south, obc_west, obc_east

write_netcdf additional netCDF output

`obcpsi.F`

This routine prepares open boundary values for ψ. It uses output (formatted listing) of FERRET. Alternatively values for psi must be given directly. A file `obcpsi.mom` is written. Following *ifdefs* can be chosen:

makeobcpsi driver

readferret use FERRET-data as input (formatted listing); otherwise data must be given directly.

obc_north, obc_south, obc_west, obc_east

write_netcdf additional netCDF output

13.6 TO-DO List (How to set up open boundaries)

1. Choose regions of open boundaries carefully at locations where the barotropic velocity is mostly perpendicular to the boundary.

2. Open the grid one cell further for open boundaries, so the open boundary conditions reside at the second row (column) in the domain. The first row is only a dummy which is not real. Topography is set automatically by subroutine `topog` similar to the open boundary conditions row. If the researcher does not want the drastic preparation of setting the next three rows identically, the corresponding lines in `topog` must be commented out and it is up to the researcher to insure that the bottom topography is not sloping upward within the next two rows away from the boundary. The virtually row is set in any case.

3. Choose restoring time scale (`USER INPUT` in `obc.F`) and create tracer values at the open boundaries either from Levitus or FERRET written data.

4. Choose ψ values at the open boundaries in `obcpsi.F` or read FERRET written data to create ψ values.

5. If ψ values at the open boundaries indicate a net transport through the domain, land masses have to be set to a non-zero value. In the following example open western and eastern boundaries are chosen, transporting $130\,Sv$ ($psimax$) through the domain. $psiwall\_west$ and $psiwall\_east$ rise from zero at the northern end to $130\,Sv$ at the southern end ($jpsimax$). To guide the current through the domain all southern land masses must set to $130\,Sv$ which can be done at the end of `tropic2_obc.inc`:

```
do jrow=1,jpsimax
  do i=1,imt
    if (map(i,jrow) .ne. 0) then
```

```
        psi(i,jrow,1) = 130.e12
        psi(i,jrow,2) = 130.e12
      endif
    enddo
  enddo
```

Chapter contributed by
Arne Biastoch
abiastoch@ifm.uni-kiel.de

Chapter 14

Options for testing modules

14.1 Testing modules

Modules and the usage of modules are described in Chapter 6. Within each module is a driver intended to exercise the module in a simple environment. This means executing in a "stand alond mode" as opposed to from within a model execution. Associated with each module is a *run_script* which activates the included driver with an option. These options are listed below and are not to be enabled from within a model execution (because the model becomes the driver).

14.1.1 test_convect

This option activates a driver for the convection module in file convect.F. Refer to Sectionsubsection:convect.F for details.

14.1.2 drive_denscoef

When executing the model, required density coefficients are automatically computed[1] from within by calling module *denscoef*. However, script *run_denscoef* activates a driver in "stand alone mode" which gives details about density coefficients. Executing *run_denscoef* uses information from module *grids* and file *size.h* to construct density coefficients which depend on the vertical discretization of the grid cells. Refer to Section 6.2.2 for details on density.

14.1.3 drive_grids

This option activates a driver for the grids module in file grids.F. Refer to Sectionsubsection:grids.F for details.

14.1.4 test_iomngr

This option activates a driver for the I/O manager module in file iomngr.F. Refer to Sectionsubsection:iomngr.F for details.

14.1.5 test_poisson

This option activates a driver for the poisson module in file poisson.F. For more details refer to Sectionsubsection:poisson.F.

[1]It is no longer required to construct density coefficients before executing the model.

14.1.6 test_ppmix

This option activates a driver for the Richardson based vertical mixing module in file ppmix.F. Refer to Sectionsubsection:ppmix.F for details.

14.1.7 test_rotation

The model grid may be rotated using a set of Euler angles for solid body rotation. The Euler angles are computed by defining the geographic latitude and longitude of the rotated north pole and a point on the prime meridian. This option activates the driver which gives results from a sample rotation of the model grid. It indicates how scalars and vectors are interpolated and rotated before being used on the rotated model grid. Refer to Section 15.6.1 for details.

14.1.8 test_timeinterp

This option activates a driver for the the time interpolation module in file timeinterp.F. Refer to Sectionsubsection:timeinterp.F for details.

14.1.9 test_timer

This option activates a driver for the timing module in file timer.F. For more details refer to Sectionsubsection:timer.F.

14.1.10 test_tmngr

This option activates a driver for the time manager module in file tmngr.F. Refer to Section-subsection:tmngr.F for details.

14.1.11 drive_topog

This option activates a driver for the topography and geometry module in file topog.F. Refer to Sectionsubsection:topog.F for details.

14.1.12 test_util

This option activates a driver for the utility module in file util.F. Refer to Sectionsubsec-tion:util.F for details.

14.1.13 drive_mwsim

This option activates a driver which simulates how the memory window behaves using various combinations of processors and options when option *coarse_grained_parallelism* is enabled. The run script is *run_mwsim* and the source file is *mwsim.F* which can be modified to investigate various scenarios.

Chapter 15

General Model Options

MOM 2 is configured in various ways through the use of options which are enabled by setting preprocessor type directives. The directives are usually placed on a preprocessor or compile statement in a run_script and take the form *-Doption1 -Doption2* etc. Note that the *-D* must be used as a prefix to the option name. For an example, refer to script *run_mom*. Care should be exercised when enabling options because there is no check for mispelled options. When enabled, options eliminate or include portions of code thereby implementing desired features. As MOM 2 executes, a summary of all enabled options is given.

Some options are incompatible with others. Subroutine *checks*[1] looks for all conflicting options and if any are found, writes all error messages to the printout file before stopping. Various additional checking is done and warning messages may also be issued. These are less serious than error messages and allow MOM 2 to continue. This does not mean they should be ignored. They often give information about possibly inappropriate values being used. In fact, it is a good idea to search the printout file to locate any "warning" messages and verify they are harmless before continuing. Refer to Section 19.5 for details on how to perform searches.

In the rest of this chapter, allowable options will be grouped into catagories and given brief explanations aloong with guidelines for useage. Diagnostic options are discussed separately in Chapter 18.

15.1 Computer platform

Options are used to invoke minor changes in MOM 2 which are computer platform specific thereby enabling MOM 2 to execute on various platforms. These changes tend to be concentrated in the I/O manager, timing, and NetCDF routines. Only one platform must be specified.

15.1.1 cray_ymp

This option configures MOM 2 for the CRAY YMP which no longer exists at GFDL. If using a CRAY YMP with an operating system prior to Unicos 9, then the "f90" command in script *run_mom* must be changed to "cf77" with appropriate options.

15.1.2 cray_c90

This option configures MOM 2 for the CRAY C90 which no longer exists at GFDL. If using a CRAY YMP with an operating system prior to Unicos 9, then the "f90" command in script *run_mom* must be changed to "cf77" with appropriate options.

[1] contained in file checks.F

15.1.3 cray_t90

This option configures MOM 2 for the CRAY T90 which is the current computational workhorse at GFDL. The operating system is Unicos 9.1.0.1 which does not have a "cf77" compiler. Therefore all scripts which execute on this platform use the Fortran 90 compiler "f90".

15.1.4 sgi

This option configures MOM 2 for Silicon graphics workstations at GFDL. This should work on any system whose Fortran *open* statement conforms to Fortran 77 and whose extensions are in compliance with Fortran 90.

15.1.5 distributed_memory

This option is to be used in conjunction with options *ramdrive* and *coarse_grained_parallelism* to configure MOM 2 for distributed systems. This is an experimental option which has not yet been completed. It will most likely be changing until the Cray T3E is installed at GFDL. Refer to Section 3.5.3 for further details.

15.2 Compilers

Run scripts which are typically executed on SGI workstations use a "f77" command. Run scripts which typically execute on the CRAY T90 at GFDL use a "f90" command. On other platforms, the appropriate compiler command should be used.

15.2.1 f90

When compiling under a Fortran 90, the option *f90* must be enabled to handle differences between Fortran 77 and Fortran 90. This option only affects the I/O manager and should <u>not</u> be enabled when compiling under Fortran 77.

15.3 Dataflow I/O Options

Various options for handling I/O between disk and the memory window are available. Each has its advantages and disadvantages as described in the following sections. One and only one of these options must be enabled. It should be noted that regardless of which option is used, details are implemented at the lowest level routines. This allows the higher level code structure, in particular the reading and writing of data, to remain the same for all options. The I/O manager assigns unit numbers for each file with appropriate attributes. Refer to Section 6.2.4 for details.

15.3.1 ramdrive

This one is similar in concept to the usage of the word *ramdrive* in the IBM PC world. It concerns configuring a portion of memory to behave like disk which speeds up programs that are I/O intensive particularly when disks are relatively slow. The catch is that there must be enough memory available to contain the disk data. In MOM 2, this option defines a huge memory array and replaces all reads and writes to disk with copying variables into or out of locations in this huge array. This is done at the lowest subroutine levels so that the higher level routines look the same regardless of whether this I/O option is used or not. This option is not specific to any computer platform and therefore makes MOM 2 highly portable.

15.3.2 crayio

This option uses the CRAY specific *wordio* package to read and write data to and from disk. As currently configured, option *crayio* directs the *iomngr* to assign scatch files to CRAY solid state disk. So solid state disk is assumed. If the "sds" attribute is removed from the file specification, then the files will reside on rotating disk.

15.3.3 fio

This option uses Fortran direct access I/O to read and write data to and from disk. It works on CRAY, SGI workstations and presumably any other platform which supports Fortran direct access I/O. Record size is specified in words which is converted to bytes within the *iomngr*.

15.4 Parallelization

Currently, there are two ways to parallelize MOM 2. The first is with 'autotasking' which gives fine grained parallelism at the do loop level. All that is required is to set the desired number of processors and open the memory window up as described in Section 3.5.1. The second is coarse grained parallelism which is enabled with an option.

15.4.1 coarse_grained_parallelism

Coarse grained parallelism is high level parallelism enforced at the level of the latitude rows rather than at the do loop level as in fine grained parallelism. For details, refer to Section 3.5.2.

15.5 Grid generation

How to design grids is detailed in Chapter 7 and the options are summarized below.

15.5.1 drive_grids

Script *run_grids* uses option *drive_grids* to execute the grids module in "stand alone mode". This is the recommended method for designing grids. Option *drive_grids* is not appropriate when executing a model because the model itself becomes the driver.

15.5.2 generate_a_grid

This option is used for generating a grid domain and resolution as opposed to importing one. Either this one or option *read_my_grid* must be enabled.

15.5.3 read_my_grid

This is the option to use when importing a domain and resolution generated from outside of the MOM 2 environment. Either this one or option *generate_a_grid* must be enabled.

15.5.4 write_my_grid

For exporting a domain and resolution generated by module *grids*.

15.5.5 centered_t

This is for constructing a grid by method 1 as described in Chapter 7. If this option is not enabled, grid construction is by method 2 which is the default method for MOM 2.

15.6 Grid Transformations

15.6.1 rot_grid

Option *rot_grid* allows the grid system defined by module *grids* to be rotated so that the pole is outside the area of interest. Refer to Chapter 8 for details.

15.7 Topography and geometry generation

The options available for constructing topography and geometry are covered in detail in Chapter 9 and summarized below. One and only one of the following options must be enabled: *rectangular_box*, *idealized_kmt*, *scripps_kmt*, *etopo_kmt*, or *read_my_kmt*. All others are optional.

15.7.1 rectangular_box

This constructs a flat bottomed rectangular domain on the grid resolution specified by module *grids*. All interior ocean $kmt_{i,jrow}$ are set to level *km*. If used with option *cyclic* it configures a zonally re-entrant channel. If used with option *solid_walls* then all boundary $kmt_{i,jrow}$ cells are land cells.

15.7.2 idealized_kmt

This produces an idealized map of the world's geometry with a bottom depth that is not realistic. It requires no data and is highly portable. This is also good for investigations where idealized geometries need to be constructed. The geometry and topography are interpolated to the resolution specified by module *grids*.

15.7.3 scripps_kmt

This generates geometry and topography interpolated from SCRIPPS data (in the DATABASE described in Section 19.6) to the resolution specified by module *grids*.

15.7.4 etopo_kmt

This generates geometry and topography interpolated from a 1/12° x 1/12° dataset[2] to the resolution specified by module *grids*.

15.7.5 read_my_kmt

This option imports a topography and geometry $kmt_{i,jrow}$ field from outside of the MOM 2 environment or one produced from option *write_my_kmt*.

[2]This dataset may be purchased from the Marine Geology and Geophysics Division of the National Geophysical Data Center and is not included in the MOM 2 DATABASE.

15.7.6 write_my_kmt

This option exports a topography and geometry $kmt_{i,jrow}$ field by writing it to disk.

15.7.7 flat_bottom

To deepen all ocean $kmt_{i,jrow}$ cells to the maximum number of levels given by level km, enable this option. It works with option *idealized_kmt* or *scripps_kmt*.

15.7.8 fill_isolated_cells

This option converts all isolated ocean T cells into land cells.

15.7.9 fill_shallow

This option converts ocean cells to land cells in regions where the ocean depth is shallower than what is allowed.

15.7.10 deepen_shallow

This option increases the number of ocean cells in regions where the ocean depth is shallower than what is allowed.

15.7.11 round_shallow

This option rounds the number of ocean cells to zero or the minimum number of allowable ocean levels in regions where the ocean depth is shallower than what is allowed.

15.7.12 fill_perimeter_violations

This option builds a land bridge between two distinct land masses which are seperated only by a distance of one ocean T cell. There must be at least two ocean T cells separating distinct land masses.

15.7.13 widen_perimeter_violations

This option removes land cells between two distinct land masses which are seperated only by a distance of one ocean T cell. There must be at least two ocean T cells separating distinct land masses.

15.8 Initial Conditions

Typically, at initial condition time, the model is at rest with a specified density distribution. This is accomplished by zeroing all velocities (internal and external mode) and setting potential temperature and salinity at each grid cell using one of the options described below. All passive tracers are initialized to unity in the first level ($k = 1$) and zero for all other levels. In principle, initial conditions could also come from a particular MOM 2 restart file although this is not the intent of the following options. One and only one of these options must be enabled.

15.8.1 equatorial_thermocline

An idealized equatorial thermocline from Philander and Pacanowski (1980) is approximated by a function dependent on depth but not latitude or longitude. Salinity is held constant at 35 ppt and the temperature profile is given by

$$\bar{t}_k = T_\circ(1 - \tanh(\frac{zt_k - H_\circ}{Z_\circ})) + T_1(1 - \frac{zt_k}{zt_{km}}) \tag{15.1}$$

where $H_\circ = 80\text{x}10^2$ cm, $Z_\circ = 30\text{x}10^2 cm$, $T_\circ = 7.5\ ^\circ C$ and $T_1 = 10.0\ ^\circ C$. The intent is to initialize idealized equatorial models. It is also useful to use option *sponges* which damps the solution back to Equation (15.1) along the northern and southern boundaries to kill off Kelvin waves. Simple idealized windstress can also be set using options *equatorial_taux* and *equatorial_tauy*. Refer to these options for more details.

15.8.2 idealized_ic

This constructs an idealized temperature and salinity distribution based on zonal averages of annual means from the Levitus (1982) data. The distribution is only a function of latitude and depth and is contained totally within MOM 2 so no external datasets are needed. This option makes MOM 2 easily portable to various computer platforms.
Recommendation: As a matter of strategy, it is recommended that any new model be first set up using option *idealized_ic* to eliminate problems which could potentially come from more realistic initial conditions. Only after the model sucessfully executes and the solution is judged reasonable should more realistic initial conditions be considered.

15.8.3 levitus_ic

This option accesses three dimensional temperature and salinity data which have previously been prepared by scripts in PREP_DATA for the resolution specified by module *grids* as discussed in Section 19.6. The particular month from the Levitus (1982) climatology which is to be used as initial conditions must be accessed by pathname from within script *run_mom*. The researcher should supply the pathname.

15.9 Surface Boundary Conditions

As detailed in Chapter 10, surface boundary conditions are viewed as comming from a hierarchy of atmospheric models: the simplest of which are datasets of varying complexity and the most complicated ones are the GCM's. The following is a summary of the various options relating to surface boundary conditions with further details being left to Chapter 10. One and only one option of the following options must be enabled: *simple_sbc*, *time_mean_sbc_data*, *time_varying_sbc_data*, or *coupled*.

15.9.1 simple_sbc

These are based on zonal averages of Hellerman and Rosenstein annual mean windstress (1983). If option *restorst* is enabled, the surface temperature and salinity are damped back to initial conditions on a time scale given by *dampts* in days as given by Equation (15.3) in Section 15.9.7. This option together with options *idealized_ic* and *restorst* allow MOM 2 to be easily portable to various computer platforms.

Recommendation: When setting up a new model for the first time, it is recommended that option *simple_sbc* be used. Only after verifying that results are reasonable should more suitable options be considered.

15.9.2 equatorial_taux

This option is meant to specify a constant windstress of $taux0 \; dynes/cm^2$ in the zonal direction for idealized equatorial studies. It will replace the zonal windstress given by option *simple_sbc*. and its default value of $taux0 = -0.5$ can be changed via namelist. Refer to Section 5.4

15.9.3 equatorial_tauy

This option is meant to specify a constant windstress of $tauy0 \; dynes/cm^2$ in the meridional direction for idealized equatorial studies. It will replace the meridional windstress given by option *simple_sbc* and its default value of $tauy0 = 0.0$ can be changed via namelist. Refer to Section 5.4

15.9.4 time_mean_sbc_data

This option is used to supply annual mean Hellerman and Rosenstein windstress and annual mean Levitus (1982) sea surface temperature and salinity for use with option *restorst*. Data is prepared for the resolution in *mom* as described in Section 19.6. Refer to Section 10.2 for further discussion. If option *restorst* is enabled, surface temperature and salinity are damped back to this annual mean data as given by Equation (15.3) in Section 15.9.7.

15.9.5 time_varying_sbc_data

This option is used to supply monthly mean Hellerman and Rosenstein windstress and monthly mean Levitus (1982) sea surface temperature and salinity for use with option *restorst*. Data is prepared for the resolution specified in module *grids* as described in Section 19.6. It is interpolated to the time step level in *mom* as described in Section 10.2. If option *restorst* is enabled, surface temperature and salinity are damped back to this monthly mean data as given by Equation (15.3) in Section 15.9.7.

15.9.6 coupled

This option is used to couple *mom* to an atmospheric model assumed to have resolution differing from that of *mom*. It allows for two way coupling as described in Section 10.1.

15.9.7 restorst

Ocean tracers are driven directly by surface tracer fluxes. In some cases, specification of surface tracer flux may lead to surface tracers drifting far away from observed data. This may be due to errors in the data or in paramaterizations not capturing the proper physics. When this happens, it is desirable to restrict how closely sea surface tracers remain to prescribed data by use of a restoring term. The prescribed data typically comes from options *simple_sbc*, *time_mean_sbc_data*, or *time_varying_sbc_data* as described above. Enabeling option *restorst* provides the restoring by means of Newtonian damping.

The damping is actually converted to a sea surface tracer flux and enters the tracer equation as a top boundary condition for vertical diffusion. The amount of damping is defined by a time scale *dampts* in days and thickness *dampdz* in cm which are input through namelist (Refer to

Section 5.4). The thickness is redundant but its use is explained below. Note that the damping time scale and thickness may by set differently for each tracer.

A Newtonian damping term applied to the first vertical level may be converted into a surface tracer flux by vertically summing as in

$$\sum_{k=1}^{km}(\kappa t_z)_z\Delta z = -\sum_{k=1}^{km}\delta^{1,k}\cdot\frac{1}{dampts_n\cdot 86400.0}(t-t^\star)\Delta z \tag{15.2}$$

where t is temperature and $\delta^{1,k}$ is the Kronecker delta ($\delta^{1,1}=1$ and $\delta^{1,k>1}=0$). The left hand side of Equation (15.2) equates to $stf - bmf$ where stf is the surface tracer flux and bmf is the bottom tracer flux (which is taken as zero). Applying the indexing terminology of MOM 2, the equation becomes

$$stf_{i,j,n} = -\frac{dzt_{k=1}}{dampts_n\cdot 86400.0}(t_{i,1,j,n,\tau-1}-t^\star_{i,j,n,time}) \tag{15.3}$$

where $t^\star_{i,j,n,time}$ represents prescribed surface data for tracer n and $dampts_n$ is the time scale. However, by introducing another factor $dampdz_n$, the equation can be re-written as

$$stf_{i,j,n} = -\frac{dampdz_n}{\frac{dampdz_n}{dzt_{k=1}}dampts_n\cdot 86400.0}(t_{i,1,j,n,\tau-1}-t^\star_{i,j,n,time}) \tag{15.4}$$

which gives a modified time scale of $\frac{dampdz_n}{dzt_{k=1}}dampts_n$ days. The purpose[3] of this awkward complication of re-defining the time scale is to remind researchers that when converting to a flux, the flux is a function of vertical thickness of the first model level $dzt_{k=1}$. For instance, consider two models. The first with $dzt_{k=1}$ = 10x10^{2}cm and the second with $dzt_{k=1}$ = 50x10^{2}cm. Specifying $dampts_n = 50$ days and $dampdz_n = dzt_{k=1}$ in both models implies a very different tracer flux into each. The tracer flux is only comparable if $dampdz_n$ has the same value in each model (but this means the damping time scales are different).

15.9.8 shortwave

Heatflux at the ocean surface is composed of latent, sensible, longwave, and shortwave components. When applied as an upper boundary condition at the ocean surface, all of the flux is absorbed within the first vertical model level. If vertical resolution is less than 25m, this may lead to too much heating within the first level resulting in SST that is too warm. Option *shortwave* allows some of the solar shortwave energy to penetrate below the first level. This penetration below the surface is a function of wavelength. For the case of clear water, it is assumed that energy partitions between two exponentials with a vertical dependency given by

$$pen_k = A\cdot e^{-zw_k/\ell_1} + (1-A)\cdot e^{-zw_k/\ell_2} \quad for\ k = 1\ to\ km \tag{15.5}$$

Coefficients are set for the case of clear water using $A = 0.58$, $\ell_1 = 35$ cm, and $\ell_2 = 2300$ cm. This means that 58% of the energy decays with a 35 cm e-folding scale and 42% of the energy decays with a 23 m e-folding scale. If the thickness of the first ocean level $dzt_{k=1} = 50$ meters, then penetration of solar shortwave wouldn't matter. However, for the case where $dzt_{k=1} = 10$ meters, the effect can be significant and may be particularly noticeable as warm SST in the summer hemisphere.

The divergence of the vertical penetration pen_k is calculated as

[3] Note that $dampdz_n$ does not mean that depth over which tracers are restored is different than $dzt_{k=1}$.

$$divpen_k = (pen_{k-1} - pen_k)/dzt_k, \quad for\ k = 1\ to\ km \tag{15.6}$$

and the sub-surface heating due to the divergence is added to the temperature component of the tracer equation through a source term (refer to Section 11.10.5) given by

$$source_{i,k,j} = source_{i,k,j} + sbcocn_{i,jrow,m_{i,jrow,isw}} \cdot divpen_k \tag{15.7}$$

where subscript isw points to the shortwave surface boundary condition[4]. In general, since surface heatflux $stf_{i,j,1}$ already contains a solar shortwave component, the penetration function at the ocean surface is set to zero ($pen_0 = 0$) to prevent the shortwave component from being added in twice: once through the surface boundary condition ($stf_{i,j,1}$) for vertical diffusion (given by Equation (11.46)) and once through the source term given above. For further information refer to Paulson and Simpson (1977), Jerlov (1968) and Rosati (1988).

15.9.9 minimize_sbc_memory

This option re-dimensions buffer arrays used with option *time_varying_sbc_data* for time interpolations. Instead of being dimensioned as (imt,jmt) these arrays are dimensioned as (imt,jmw) and there are two buffer fields for each surface boundary condition. The resulting savings in memory is approximately $2 \cdot numobc \cdot imt \cdot jmt$ where $numobc$ is the number of ocean surface boundary conditions . However, the price to be paid is increased disk access which will significantly slow execution if conventional rotating disk is used. The intent of this option is for very high resolution models which cannot fit into available memory and require a very conservative approach to memory usage. For additional space saving measures, refer to Section 15.19.11.

15.10 Lateral Boundary Conditions

Neumann conditions are assumed for heat and salt on lateral boundaries (no-flux across boundaries). For momentum, a Dirichlet condition is assumed (no-slip). There are some minor variations on these at the domain boundaries as described in the following sections. Note that land cells are where $kmt_{i,jrow}$= 0. One and only one of the following options must be enabled.

15.10.1 cyclic

This makes a semi-infinite domain. It implements cyclic conditions in longitude. Whatever flows out of the (eastern,western) end of the basin enters the (western,eastern) end. Normally, the computations in longitude proceed from $i = 2$ to $imt - 1$ after which cyclic conditions are imposed. In latitude, the computations proceed from $jrow = 2$ to $jmt-1$. The cyclic boundary condition is

$$\begin{aligned} kmt_{1,jrow} &= kmt_{imt-1,jrow} \\ kmt_{imt,jrow} &= kmt_{2,jrow} \end{aligned} \tag{15.8}$$

Note, there is no option for the doubly cyclic case (cyclic in latitude also) and on the northern and southernmost cells

$$\begin{aligned} kmt_{i,1} &= 0 \\ kmt_{i,jmt} &= 0 \end{aligned} \tag{15.9}$$

[4] Note that solar shortwave data is not supplied with MOM 2 and so must be supplied by the researcher.

15.10.2 solid_walls

This implements solid walls at domain boundaries in longitude and latitude. The condition is

$$\begin{aligned} kmt_{1,jrow} &= 0 \\ kmt_{imt,jrow} &= 0 \\ kmt_{i,1} &= 0 \\ kmt_{i,jmt} &= 0 \end{aligned} \tag{15.10}$$

The domain is finite and closed.

15.10.3 symmetry

This implements a symmetric boundary condition across the equator. Specifically, the second last row of velocity points must be defined on the equator ($\phi^U_{jmt-1} = 0.0$). Note that the equator is at the northern end of the domain. The condition applies when j in the memory window corresponds to $jrow = jmt - 1$ and is given by

$$\begin{aligned} t_{i,k,j+1,n} &= t_{i,k,j,n} \\ u_{i,k,j+1,1} &= u_{i,k,j,1} \\ u_{i,k,j+1,2} &= -u_{i,k,j,2} \\ psi_{i,jrow+1} &= -psi_{i,jrow} \\ kmt_{i,jrow+1} &= kmt_{i,jrow} \end{aligned} \tag{15.11}$$

15.10.4 sponges

This implements a poor man's open boundary condition along the northernmost and southernmost artificial walls in a limited domain basin. A Newtonian damping term is added to the tracer equations which damps the solution back to data within a specified width from the walls. The form is given by Equation 11.56 and the data is generated by script *run_sponge*[5] in PREP_DATA.

If option *equatorial_thermocline* is enabled, then the profile from Equation (15.1) is used instead of data prepared in PREP_DATA. The meridional width of the sponge layer *spng_width* is hard wired to 3 degrees and the reciprocal of the damping time scale *spng_damp* is hard wired to 1/5 days. Their purpose is to damp Kelvin waves in idealzed equatorial models. As indicated in Section 19.5, use UNIX *grep* to find their location if changes are to be made.

The current implementation uses data defined at the latitude of the northern and southern walls as the data to which the solution is damped. This data varies monthly, but the annual mean values can be used instead by setting variable *annlev* in the namelist. Refer to Section 5.4 for information on namelist variables. The width of the sponge layers is determined by a Newtonain damping time scale that is a function of latitude and set in subroutine *sponge* which is executed by script *run_sponge*.

If a more realistic sponge layer is desired, data from latitude rows within the sponge layers needs to be saved instead of just the data at the latitude of the walls. This is a bit more I/O intensive and is not an option as of this writing. Refer to Sections 19.6 and 11.10.5 for further details.

[5] Note that this script can only be run after script *run_ic* which prepares temperature and salinity data for all latitude rows.

15.10.5 obc

Open boundary conditions are based on the methodology of Stevens (1990). There are two types of open boundary conditions: 'active' in which the interior is forced by data prescribed at the boundary and 'passive' in which there is no forcing at the boundary and phenomena generated within the domain can propagate outward without disturbing the interior solution.

Open boundaries may be placed along the northern, southern, eastern and/or western edges of the domain. At open boundaries, baroclinic velocities are calculated using linearized horizontal momentum equations and the streamfunction is prescribed from other model results or calculated transports (e.g. directly or indirectly from the Sverdrup relation). Thus, the vertical shear of the current is free to adjust to local density gradients. Heat and salt are advected out of the domain if the normal component of the velocity at the boundary is directed outward. When the normal component of the velocity at the boundary is directed inward, heat and salt are either restored to prescribed data ('active' open boundary conditions) or not ('passive' open boundary conditions).

In contrast to the above described 'active' open boundary conditions, 'passive' ones are characterized by not restoring tracers at inflow points. Additionally, a simple Orlanski radiation condition (Orlanski 1976) is used for the streamfunction.

Refer to Chapter 13 for all the options and details.

15.11 Filtering

In general, the equations of motion are filtered in various ways to remove components from the solution which are physically unimportant but nevertheless limit the length of the time step.

Convergence of meridians

Because the model is formulated in spherical coordinates, convergence of meridians near the earth's poles reduces the effective grid size in longitude. At high latitude, the solution may start to become unstable because of too large a time step. Instead of decreasing the time step, an alternative is to filter the solution at high latitudes to remove high wavenumber components. There are two filtering methods described below for suppressing these components. Option *fourfil* uses Fourier filtering and option *firfil* uses a symmetric finite impulse response filter to accomplish the same thing. Either one of these is typically applied polward of a critical latitude determined by the researcher.

An alternative to filtering is to rotate the grid so that the convergence of meridians takes place outside to the model domain. Refer to Section 15.6.1 for details.

Inertial period

Another component involves the inertial period. The time step for coarse global models is severly restricted by having to resolve the inertial period near the poles. Option *damp_inertial_oscillation*, described below, filters the Coriolis term and removes this restriction.

15.11.1 fourfil

Due to the convergence of meridians on a sphere, longitudinal grid resolution decreases to zero as the poles are approached. For global domains this may severely limit the length of the time step due to the CFL constraint

$$\Delta\tau < \frac{\Delta x}{2 \cdot u} \tag{15.12}$$

where $\Delta\tau$ is the time step. The instability can be removed by filtering (Bryan, Manabe, and Pacanowski, 1975 and Takacs, Balgovind, 1983) the highest wave numbers out of the solution since they are the most unstable ones.

Filtering is not to be encouraged and is not recommended unless absolutely necessary. When necessary, it should be restricted to the northernmost and southernmost latitudes in the domain. Typically, the filtering works on strips of latitude defined by the researcher. The latitudes filtered in the southern hemisphere are from *rjfrst* to *rjft1* for tracers and from *rjfrst* to *rjfu1* for velocity components. These latitudes are actually converted to the nearest model latitudes before being used. The reason for *rjfrst* is to skip over the latitudes containing Antartica. In the northern hemisphere, filtering operates on latitudes from *rjft2* to yt_{jmt-1} for tracers and from *rjfu2* to yu_{jmt-3} for velocities.

Although the intent was for demonstration purposes only, latitude ranges for filtering could have been set more conservatively in the MOM 1 test case. In the MOM 2 test cases, the same latitude ranges are kept for compatibility reasons. In practice, there is probably no need to filter southern hemisphere latitudes because of Antarctica[6]. In the northern hemisphere, it might be necessary to filter poleward of 70° or 75°N and the latitude ranges should be set accordingly. These latitudes are set in subroutine *setocn*.

This option does a Fourier smoothing of prognostic variables in the longitudinal direction by knocking out wavenumbers larger than a critical one which is dependent on latitude. The critical wavenumber is given as

$$N = im \frac{\cos\phi^T_{jrow}}{\cos\phi^T_{jft0}} \tag{15.13}$$

where N is the number of waves to retain, im is the length of an ocean strip, and $jft0$ is a reference latitude row[7]. Because of geometry and topography, each filtering latitude is composed of strips defined by starting and stopping longitudes and each strip is a function of latitude and depth.

Any variable α_i where $i = 1, im$ can be filtered by a discrete Fourier transform

$$\tilde{\alpha}_i = A_\circ + \sum_{\ell=1}^{N/2} A_\ell \cos\frac{2\pi i\ell}{im} + \sum_{\ell=1}^{N/2-1} B_\ell \sin\frac{2\pi i\ell}{im} \tag{15.14}$$

where $\tilde{\alpha}_i$ is the filtered value composed of N wavenumbers given by Equation (15.13) and the even and odd Fourier coefficients A_ℓ and B_ℓ are given as

$$A_\circ = \frac{1}{im}\sum_{i=1}^{N}\alpha_i \tag{15.15}$$

$$A_\ell = \frac{2}{im}\sum_{i=1}^{N}\alpha_i\cos\frac{2\pi i\ell}{im} \quad for\ \ell = 1, 2, \cdots \le \frac{im}{2} - 1 \tag{15.16}$$

$$A_{im/2} = \frac{1}{im}\sum_{i=1}^{im}\alpha_i\cos(i\pi) \tag{15.17}$$

$$B_\ell = \frac{2}{im}\sum_{i=1}^{N}\alpha_i\sin\frac{2\pi i\ell}{im} \quad for\ \ell = 1, 2, \cdots \le \frac{im}{2} - 1 \tag{15.18}$$

[6] Unless an *aqua planet* is being investigated.

[7] When filtering velocities the reference latitude row is given by $jfu0$.

Even Fourier coefficients are used to filter tracers because the boundary condition at edges of the strips is no-flux. Odd Fourier coefficients are used for velocity components because of the no-slip boundary condition at the edges of the strips. Vorticity ($ztd_{i,jrow}$) is filtered with even Fourier coefficients[8]. Note that there is no windowing in the transform. Also, this method is a real time burner and FFT's are of little use because of the arbitrary strip widths. The time burning aspect was the motivation for option *firfil*.

15.11.2 firfil

For reasons outlined in Section 15.11.1, prognostic variables in polar latitudes may have to be filtered. This option uses multiple passes with a simple symmetric finite impulse response filter (Hamming 1977) to accomplish the filtering. Its advantage over option *fourfil* is speed. As with fourier filtering, the amount of filtering can be a function of latitude and is controlled by the number of passes of the filter on the data. The number of passes is arbitrairly given by

$$numflt = \frac{\cos\phi^T_{jft0}}{\cos\phi^T_{jrow}} \tag{15.19}$$

where $jft0$[9] is some reference latitude and *numflt* is the number of filter applications per prognostic variable per latitude. *numflt* is for variables on T cell latitudes and there is a corresponding number of passes used for variables on U cell latitudes given by *numflu*. Equation (15.19) is arbitrary and should be adjusted by the researcher to get the desired effect. It is set in subroutine *setocn*. This filter differs from the Fourier filter in that it filters across land using masks to set boundary conditions. It does not need the strips and is much faster than the Fourier filter.

A symmetric filter can be written as

$$\tilde{\alpha}_i = \sum_{\ell=-M}^{M} c_\ell \alpha_{i-\ell} \tag{15.20}$$

and its transfer function is

$$H(\omega) = c_\circ + 2\sum_{\ell=1}^{M} C_\ell \cos(\omega) \;\; for\ \omega = 0\ to\ \pi \tag{15.21}$$

A very simple one is used with $M = 1$ with weights

$$\tilde{\alpha}_i = \frac{1}{4}\alpha_{i-1} + \frac{1}{2}\alpha_i + \frac{1}{4}\alpha_{i+1} \tag{15.22}$$

and the transfer function is

$$H(\omega) = \frac{1}{2}(1 + \cos(\omega)) \tag{15.23}$$

The effect of multiple applications or passes with this filter can be easily calculated. If the filter is applied $numflt$ times then the transfer function is $H^{numflt}(\omega)$.

[8] Strips for the vorticity do not include coastal ocean cells because of the island integrals.

[9] When filtering velocities the reference latitude is given by *jfu0*.

15.11.3 damp_inertial_oscillation

Option *damp_inertial_oscillation* damps inertial oscillations by treating the Coriolis term semi-implicitly. Why treat the Coriolis terms semi-implicitly? It only makes sense in coarse resolution ($\Delta_x \geq 5°$) global models where the time step allowed by the CFL condition does not resolve the inertial period which is 1/2 day at the poles. Treating the Coriolis term semi-implicitly damps the inertial oscillation and allows a longer time step. For global models with $\Delta_x \leq 2°$, the time step allowed by the CFL condition is typically small enough (less than 2 hours) to resolve the inertial period at the poles and so semi-implicit treatment is not needed. Consider the simple system for inertial oscillations

$$\begin{aligned} u_t - fv &= 0 && (15.24)\\ v_t + fu &= 0 && (15.25)\end{aligned}$$

where $f = 2\Omega\sin\phi$, u is zonal velocity, and v is meridional velocity. When discretizing and solving the Coriolis term explicitly, the solution is given by

$$\begin{aligned} u^{\tau+1} &= u^{\tau-1} + 2\Delta\tau\cdot fv^{\tau} && (15.26)\\ v^{\tau+1} &= v^{\tau-1} - 2\Delta\tau\cdot fu^{\tau} && (15.27)\end{aligned}$$

When treating the Coriolis term semi-implicitly, an implicit Coriolis factor $0.5 \leq acor < 1$ is used and the system becomes

$$\begin{aligned} u^{\tau+1} - 2\Delta\tau\cdot acor\cdot fv^{\tau+1} &= u^{\tau-1} + 2\Delta\tau\cdot(1-acor)\cdot fv^{\tau-1} && (15.28)\\ v^{\tau+1} + 2\Delta\tau\cdot acor\cdot fu^{\tau+1} &= v^{\tau-1} - 2\Delta\tau\cdot(1-acor)\cdot fu^{\tau-1} && (15.29)\end{aligned}$$

The solution, after a little algebra, is given by

$$u^{\tau+1} = u^{\tau-1} + 2\Delta\tau\cdot\frac{fv^{\tau-1} - (2\Delta\tau\cdot acor\cdot f)\cdot fu^{\tau-1}}{1+(2\Delta\tau\cdot acor\cdot f)^2} \qquad (15.30)$$

$$v^{\tau+1} = v^{\tau-1} - 2\Delta\tau\cdot\frac{fu^{\tau-1} + (2\Delta\tau\cdot acor\cdot f)\cdot fv^{\tau-1}}{1+(2\Delta\tau\cdot acor\cdot f)^2} \qquad (15.31)$$

Note that option *damp_inertial_oscillation* will also damp external Rossby waves although the fastest ones are of largest spatial scales, well resolved by the grid, and do not limit the time step. Refer to Section 5.4.4 for information on choosing time steps.

15.12 Linearizations

15.12.1 linearized_density

Instead of the third order polynomial approximation to the equation for density, a linear form given by

$$\rho_{i,k,j} = -\alpha\cdot t_{i,k,j,1,\tau} \qquad (15.32)$$

is used where α is arbitrarily set to 2.0x10^{-4}. Note that there is no dependence on depth and the usual form $\rho = \rho_\circ(1-\alpha\cdot t)$ is not used. The reason is for numerical accuracy. Adding the constant part removes three significant digits of accuracy from the resulting density. Since only gradients of density are dynamically important, the constant is of no physical importance. Also, the values of T_k^{ref} and S_k^{ref} (refer to Section 6.2.2) are typically subtracted from temperature and salinity throughout the code so are set to zero.

15.12.2 linearized_advection

It is sometimes useful to linearize about a state of no motion as in Philander and Pacanowski (1980). The state of no motion is given by

$$t = t(z) \tag{15.33}$$

$$\rho = \rho(t) \tag{15.34}$$

where t is temperature and the density ρ is linearized as in option *linearized_density*. If the advective velocities are thought of as being composed of a mean and deviation, then linearizing about this state eliminates the advective terms in the momentum equations. In the temperature equation, only the advective term $w \cdot \bar{t}_z$ remains where $\bar{t}$ is an initial equatorial stratification given by Equation 15.1. Note that this stratification is the initial condition. Options *levitus_ic* and *idealized_ic* are therefore incompatible and must not be enabled with option *linearized_advection*. It is not necessary to enable option *equatorial_thermocline* for linearized advection.

15.13 Explicit Convection

Convection in MOM 2 can happend in one of two ways: either implicitly by enabling option *implicitvmix* and using huge vertical mixing coefficients between ocean cells that are gravitationally unstable or explicitly by not enabling option *implicitvmix*. There are two choices for explicit convection. If option *fullconvect* is enabled, all instabilities in the water column are removed on every time step by the method of Rahmstorf which is described below. Otherwise, the old style convective adjustment takes multiple passes through the water column alternately looking for instability on odd and even model levels. When an instability is found, tracers are mixed preserving their means (weighted by cell thickness). This process may induce further instability and therefore more than one pass through the water column may be needed to remove all instability. The number of passes through the water column is controlled by variable "ncon" which is input through namelist. Refer to Section 5.4 for information on namelist variables. Both explicit convective schemes can be tested in a one dimensional model. Refer to Section 6.2.1 for details.

Explicit convection takes place only when option *implicitvmix* is not enabled. When option *implicitvmix* is enabled, instead of explicitly trying to remove a gravitational instability, large vertical diffusion coefficients are used locally to eliminate it through vertical diffusion. These large coefficients severly limit the time step and so vertical diffusion is solved implicitly to relax the restriction. As implemented, explicit convection operates only on tracers whrereas option *implicitvmix* affects both tracers and momentum.

Explicit vertical mixing schemes and option *implicitvmix* are mutually exclusive. In the following discussion the assumption is that option *implicitvmix* is not enabled. When active, explicit convection happens in one of two ways. If option *fullconvect* is enabled, the scheme of Rahmstorf is used, otherwise the older style explicit convection is used. In either case, when option *save_convection* is enabled, the results of explicit convection can be subsequently analyzed. Both explicit convection schemes are explained below.

15.13.1 fullconvect

The following is taken from notes received from Stefan Rahmstorf on "A fast and complete convection scheme for ocean models".

Imagine having three half-filled glasses of wine lined up in front of you. On the left a German Riesling, in the middle a French Burgundy and on the right a Chardonnay from New Zealand. Imagine further that you're not much of a connoisseur, so you want to mix the three together to a refreshing drink, with exactly the same mixture in each glass. The trouble is, you can only mix the contents of two adjacent glasses at a time. So you start off by mixing the Riesling with the Burgundy, then you mix this mixture with the Chardonnay, then... How often do you need to repeat this process until you get an identical mix in all glasses?

Incidentally, putting this question to a friend is a good test to see whether she (or he) is a mathematician or a physicist. A mathematician would answer "an infinite number of times", while a physicist would be well aware that there is only a finite number of molecules involved, so you can get your perfect drink with a finite mixing effort (only you would have no way to tell whether you've got it or not).

In any case, the number of times you need to mix is very large, and this is the problem of the standard convection scheme of the GFDL ocean model (Cox 1984), which mixes two adjacent levels of the water column if they are statically unstable. The model includes the option to repeat this mixing process a number of times at each time step, as an iteration process towards complete removal of static instabilities. The minimum number of iterations needed to mix some of the information from layer 1 down to layer n is n-1.

To avoid this problem, one needs to relax the condition that only two levels may be mixed at a time. To achieve complete mixing, a convection scheme is required that can mix the whole unstable part of the water column in one go. I have been using such a scheme back in 1983 in a one-dimensional mixing model for the Irish Sea, and I'm sure many other people have been using similar ones. Marotzke (1991) introduced such a scheme into the GFDL ocean model. It appears that it hasn't been taken up as enthusiastically as it might have been, and an implicit convection scheme (which increases the vertical diffusivity at unstable parts of the water column) has been preferred because of lower computational cost (e.g. Weaver et al, 1993). However, it is not difficult to set up a complete convection scheme which uses less computer time than the implicit scheme.

The standard scheme. Since the GFDL model works the grid row by row, we'll only discuss how one grid-row is treated. Here's how:

1. Compute the densities for all grid cells in the row. Two adjacent levels are always referenced to the same pressure in order to get the static stability of this pair of levels.

2. Mix all unstable pairs.

3. Since we have now only compared and mixed "even" pairs (i.e. levels 1 & 2; levels 3 & 4; etc), repeat steps (1) and (2) for "odd" pairs (i.e. levels 2 & 3; levels 4 & 5; etc).

4. Repeat steps (1)-(3) a predetermined number of times.

There are a couple of problems here. We've already said that strictly speaking this never leads to complete mixing of an unstable water column. So the process is repeated several times at each time step to approximate complete mixing. But each time all grid cells are checked for instabilities again, even those we already found to be stable. Each density calculation requires evaluation of a third order polynomial (Cox 1972) in T and S. This is where the cpu time is eaten up.

Marotzke's scheme. This scheme works as follows:

1. Same as step (1) above, except that the stability of all pairs of grid cells is checked, odd and even pairs (so that the density of interior levels is computed twice, for two different reference pressures).
2. Don't mix yet: just mark all unstable pairs and find continuous regions of the water column which are unstable (neutral stability is treated as unstable).
3. Mix the unstable regions.
4. If there was instability in any column, repeat steps (1) to (3). Those columns which were completely stable in the previous round are not dealt with again in (2) and (3), but the densities are still recomputed for the entire grid row. Repeat until no more instabilities are found.

So Marotzke relaxed the condition that only two levels are mixed at a time, and complete mixing will be achieved with at most k-1 passes through the water column, if k is the number of model levels. However, if only one grid point of a row requires n iterations, the densities for the entire grid row will be recomputed n times, so it still doesn't look too good in terms of cpu efficiency.

The fast way.

1. Compute all densities like in (1) of Marotzke.
2. Compare all density pairs to find instabilities.

   From here on, deal column by column with those grid points where an instability was found, performing the following steps:
3. Mix the uppermost unstable pair.
4. Check the next level below. If it is less dense than the mixture, mix all three. Continue incorporating more levels in this way, until a statically stable level is reached.
5. Then check the level above the newly mixed part of the water column, to see whether this has become unstable now. If so, include it in the mixed part and go back to (3). If not, search for more unstable regions below the one we just mixed, by working your way down the water column comparing pairs of levels; if you find another unstable pair, go to (3).

Note that levels which have been mixed are from then on treated as a unit. This scheme has a slightly more complicated logical structure; it needs a few more integer variables and if statements to keep track of which part of the water column we have already dealt with. The advantage is that we only recompute the densities of those levels we need; levels which are not affected by the convection process are only checked once. The scheme includes diagnostics which allow to plot the convection depth at each grid point.

Discussion
Perhaps these schemes are best discussed with an example. Imagine a model with five levels. At one grid point levels 2 & 3 and levels 3 & 4 are statically unstable. The standard scheme will, at the first pass, mix 3 & 4 and then 2 & 3. It will repeat this ncon times. Marotzke's

scheme will mark the unstable pairs and then mix 2-4 in one go. It will then return to this column for a second pass and check all levels once more. My scheme will mix 2 & 3; then compare the densities of 3 & 4 and (if unstable) mix 2-4 like Marotzke's scheme. It will then recompute the density of level 4, compare levels 4 & 5 and mix 2-5 if unstable. Finally it will compare 1 & 2 again, since the density of 2 has changed in the mixing process, so level 1 might have become statically unstable. Only the density of 2 is recalculated for this.

Note that Marotzke's scheme handles the initial mixing of levels 2-4 more efficiently. Probably my scheme could be made slightly faster still by including the "marking" feature from Marotzke's scheme (the schemes were developed independently). However, in the typical convection situation only levels 1 & 2 are initially unstable, due to surface cooling. In this situation marking doesn't help. My scheme saves time by "remembering" which parts of the water column we already know to be stable, and rechecking only those levels necessary.

There is a subtlety that should be mentioned: due to the non-linear equation of state the task of removing all static instability from the water column may not have a unique solution. In the example above, mixing 2 & 3 could yield a mixture with a lower density than level 4, in spite of 3 being denser than 4, and 2 being denser than 3 originally. In this case, my scheme would only mix 2 & 3, while Marotzke's scheme would still mix 2-4. So both schemes are not strictly equivalent, though for all practical purposes they almost certainly are.

I performed some test runs with the GFDL modular ocean model (MOM) in a two-basin configuration (the same as used by Marotzke and Willebrand 1991). The model has ca. 1000 horizontal grid points and 15 levels, and was integrated for 1 year (time step 1.5 h) on a Cray YMP. Three different model states were used: (A) a state with almost no static instability, achieved by strong uniform surface heating; (B) a state with convection occurring at about 15% of all grid points; (C) a state with convection at 30% of all grid points. The latter two were near equilibrium, with permanent convection. I compared the overall cpu time consumed by these runs with different convection schemes. The standard scheme was tried for three different numbers of iterations ncon. The results are summarized in the table; the overall cpu time is given relative to a run with no convection scheme.

Convection scheme	relative cpu time		
	A	B	C
No convection scheme	1	1	1
standard, ncon=1	1.13	1.13	1.13
standard, ncon=7	1.88	1.89	1.92
standard, ncon=10	2.25	2.27	2.32
implicit	1.52	1.52	1.52
complete	1.12	1.20	1.36

It was surprising to find that the few innocent-looking lines of model code that handle the convection consume a large percentage of the overall processing time. The numbers are probably an upper limit; a model with realistic topography and time-dependent forcing will use a bigger chunk of the cpu time for iterations in the relaxation routine for the stream function, so that the relative amount spent on convection will be lower. In my test runs, the standard scheme adds 13% cpu time per pass. My complete convection scheme used as much time as 1-3 iterations of the standard scheme, depending on the amount of convection. For zero convection it is as fast as one pass of the standard scheme, because it does the same job in this case. Additional cpu time is only used at those grid points where convection actually occurs. My scheme is considerably faster than the implicit scheme, especially for models where convection happens only at a few grid points, or only part of the time. I did not have Marotzke's scheme available for the test, but in his 1991 paper he mentions a comparison where the computation time with the implicit scheme was 60% of that with his scheme. This would give Marotzke's scheme a relative cpu time of about 2.5 in the table, with strong dependence on the amount of convective activity.

Surface heat fluxes looked identical in the runs with the implicit and complete schemes. The standard scheme showed significant deviations, however, in the surface flux as well as the convective heat flux at different depths. This is not surprising, since the rate at which heat is brought up by convection will be reduced if mixing is incomplete. The runs with ncon=7 and ncon=10 still differed noticeably from each other, and from the complete mixing case. It is possible that this could affect the deep circulation, which is driven by convective heat loss, but I didn't do long integrations to test this. The problem gets worse for longer time steps; with the standard scheme the rate of vertical mixing depends on the time step length. If acceleration techniques are used ("split time stepping", Bryan 1984), the final equilibrium could differ from one without acceleration due to this unwanted time-step dependence. Marotzke (1991) reports a case where the choice of convective scheme had a decisive influence on the deep circulation. The intention of this note is not to examine these problems any further; it is to provide an efficient alternative.

Conclusion

A convection scheme which completely removes static instability from the water column in one pass has been described, and which is much faster than the implicit scheme of the GFDL model. This scheme avoids possible problems resulting from the incomplete mixing in the standard scheme, while only using as much computer time as 1-3 iterations of the standard scheme.

15.14 Vertical sub-grid scale mixing schemes

The following options parameterize the way in which momentum and tracers are mixed vertically in the ocean. One and only one of these options must be enabled. Refer to Section 15.16 for additional hybrid mixing schemes (ones that mix tracers but not momentum).

15.14.1 constvmix

This is a basic mixing scheme that uses constant values for vertical mixing coefficients κ_m and κ_h in Equations (2.1), (2.2), (2.3), and (2.4) which amounts to using the following form for mixing coefficients at the bottom of U and T cells

$$diff\_cbu_{i,k,j} = \kappa_m \tag{15.35}$$
$$diff\_cbt_{i,k,j} = \kappa_h \tag{15.36}$$

If implicit vertical mixing is used by enabling option *implicitvmix* then mixing coefficients in regions of gravitational instability are set to their maximum values using

$$diff\_cbt_{i,k,j} = diff\_cbt\_limit \tag{15.37}$$

Typically, the value of $diff\_cbt\_limit$ is set to $10^6 cm^2/sec$. Note that $visc\_cbu\_limit$ is not used to limit $diff\_cbu_{i,k,j}$ but could be. What value to use for $visc\_cbu\_limit$ in convective regions needs further study. Diffusive fluxes at the bottom of cells take the form

$$diff\_fb_{i,k,j} = diff\_cbu_{i,k,j}(u_{i,k,j,n,\tau-1} - u_{i,k+1,j,n,\tau-1})/dzw_k \tag{15.38}$$

for momentum and

$$diff\_fb_{i,k,j} = diff\_cbt_{i,k,j}(t_{i,k,j,n,\tau-1} - t_{i,k+1,j,n,\tau-1})/dzw_k \tag{15.39}$$

for tracers. The mixing coefficients κ_m and κ_h are independent of time and are input through namelist. Refer to Section 5.4 for information on namelist variables.

15.14.2 ppvmix

This is a basic vertical mixing scheme which calculates Richardson dependent values of mixing coefficients κ_m and κ_h based on the formulation of Pacanowski and Philander (1981). In previous discretizations, the Richardson number was first computed at the base of U cells then averaged onto T cells after which the three dimensional viscosity coefficients $visc\_cbu_{i,k,j}$ were computed using Richardson numbers at the base of U cells and the three dimensional diffusivity coefficients $diff\_cbt_{i,k,j}$ were computed using Richardson numbers at the base of T cells. Actually, this is only one of many ways to do the discretization. Instead of averaging Richardson numbers, another way is to compute Richardson numbers, viscosity, and diffusivity coefficients at the base of U cells and then average the diffusivity coefficients onto T cells. A third way is to compute separate Richardson numbers at the base of T cells and U cells after which the mixing coefficients are computed. However, all three approaches have a a computational null mode and are therefore discarded as potentially troublesome. Refer to Appendix C for a discussion of null modes.

Two remaining approaches are as follows: The first involves computing Richardson numbers at the base of T cells, averaging to the base of U cells, then computing mixing coefficients at the base of T cells and U cells. The second involves computing Richardson numbers at the base of T cells, computing both mixing coefficients at the base of T cells, and then averaging the viscosity coefficients onto U cells. Based on a combination of analytic and numerical results at GFDL by Anand Gnanadesikan, the latter way is most accurate, has no computational null mode, and is therefore the approach that is followed below.

When option *ppvmix* is enabled, module *ppmix* is called to compute the vertical mixing coefficients. The Richardson numbers and mixing coefficients are calculated at the bottom of T cells.

$$rit = \frac{-grav \cdot \delta_z(\rho_{i,k,j,\tau-1})}{\overline{(\delta_z(u_{i-1,k,j-1,1,\tau-1}))^2 + (\delta_z(u_{i-1,k,j-1,2,\tau-1}))^2}^{\lambda\phi}} \tag{15.40}$$

$$diff\_cbt_{i,k,j} = \frac{fricmax}{(1+5\cdot rit)^3} + diff\_cbt\_back \tag{15.41}$$

$$visc\_cbt_{i,k,j} = \frac{fricmax}{(1+5\cdot rit)^2} + visc\_cbu\_back \tag{15.42}$$

where $grav = 980.6 cm/sec^2$. In regions of vertical instability, the mixing coefficients are set to their limiting values.

$$visc\_cbt_{i,k,j} = visc\_cbu\_limit \;\; (rit < 0) \tag{15.43}$$

$$diff\_cbt_{i,k,j} = diff\_cbt\_limit \;\; (rit < 0) \tag{15.44}$$

Typically, $diff\_cbt\_limit$ is set to $10^6 cm^2/sec$ when option *implicitvmix* is enabled. Otherwise it is set to $fricmax$. The value of $visc\_cbu\_limit$ is always set to $fricmax$. The viscosity coefficients are then averaged onto the bottom of U cells.

$$visc\_cbu_{i,k,j} = \overline{visc\_cbt_{i,k,j}}^{\lambda\phi} \tag{15.45}$$

To account for the effect of high frequency wind mixing near the surface (which is absent in climatological monthly mean wind stress), the mixing coefficients at the base of level one are taken as the maximum of the predicted mixing coefficients and parameter *windmix*.

$$diff\_cbt_{i,k=1,j} = max(windmix, diff\_cbt_{i,k=1,j}) \tag{15.46}$$
$$visc\_cbu_{i,k=1,j} = max(windmix, visc\_cbu_{i,k=1,j}) \tag{15.47}$$

Choosing coefficients

A reasonable range for the background diffusion coefficient *diff_cbt_back* is from molecular values of 0.00134 cm^2/sec to bulk values of about 0.1 cm^2/sec. For background viscosity coefficient *visc_cbu_back*, a reasonable range is from molecular values of 0.0134 cm^2/sec to bulk values of about 1.0 cm^2/sec. Based on Pacanowski and Philander (1981), reasonable values for the maximum mixing coefficient *fricmax* range from about 50 cm^2/sec to 100 cm^2/sec. The Prandtl number is about 10 for stable regions and 1 for regions of strong mixing.

Choosing *diff_cbt_back* too large tends to erode the thermocline away. Values of *visc_cbu_back* occur in regions of high Richardson number and have an affect on limiting the speed of the Equatorial Undercurrent: lower (higher) values resulting in faster (slower) speeds. To work reasonably well, this scheme needs about 10 meter or finer resolution in the vertical. It may not do well off the equator if shortwave penetration isn't taken into account because vertical shear may not be enough to overcome buoyancy when all the heat flux is absorbed within the first vertical cell. In simulations, maximum values for mixing coefficients occur in regions of low Richardson number such as surface mixed layers. Philander speculates that *fricmax* should be a function of windspeed and this is being explored. The *windmix* parameter is arbitrary and is meant to simulate the high frequency wind bursts that are absent (due to the averaging process) from climatological forcing. Typically, *windmix* has been set at 15 cm^2/sec.

Various parameters for this option may be changed through namelist. Refer to Section 5.4 for information on namelist variables.

15.14.3 kppmix

This is a basic scheme which supplies values for vertical mixing coefficients κ_m and κ_h based on Large et al. (1994). The scheme is being implemented by Tony Rosati but is not yet ready. Questions should be directed to Tony (ar@gfdl.gov).

Section 15.14.3 contributed by
Tony Rosati
ar@gfdl.gov

15.14.4 tcvmix

This is a basic scheme which supplies values for vertical mixing coefficients κ_m and κ_h based on the second order turbulence closure scheme of Mellor and Yamada level 2.5 as given in Rosati and Miyakoda (1988). The scheme is being implemented by Tony Rosati but is not yet ready. Questions should be directed to Tony (ar@gfdl.gov).

Section 15.14.4 contributed by
Tony Rosati
ar@gfdl.gov

15.15 Horizontal sub-grid scale mixing schemes

The following options parameterize the way in which momentum and tracers are mixed horizontally in the ocean. One and only one of these options must be enabled. Refer to Section 15.16 for additional hybrid mixing schemes (ones that mix tracers but not momentum).

15.15.1 consthmix

This is a basic mixing parameterization that uses constant values for horizontal mixing coefficients A_m and A_h in Equations (2.8), (2.9), (2.3), and (2.4). These constants are implemented on the eastern and northern faces of U and T cells as

$$diff\_ceu = A_m \tag{15.48}$$
$$diff\_cnu = A_m \tag{15.49}$$
$$diff\_cet = A_h \tag{15.50}$$
$$diff\_cnt = A_h \tag{15.51}$$

Note that there are no subscripts[10] on the mixing coefficients since these are independent of space. For reasons of computational speed, these mixing coefficients are incorporated with grid factors when computing diffusive fluxes which take the form

$$diff\_fe_{i,k,j} = am\_csudxtr_{i,j}(u_{i+1,k,j,n,\tau-1} - u_{i,k,j,n,\tau-1}) \tag{15.52}$$

for momentum and

$$diff\_fe_{i,k,j} = ah\_cstdxur_{i,j}(t_{i+1,k,j,n,\tau-1} - t_{i,k,j,n,\tau-1}) \tag{15.53}$$

for tracers where

$$am\_csudxtr_{i,j} = \frac{diff\_ceu}{\cos\phi^U_{jrow} dxt_{i+1}} \tag{15.54}$$
$$ah\_cstdxur_{i,j} = \frac{diff\_cet}{\cos\phi^T_{jrow} \cdot dxu_i} \tag{15.55}$$

There is a similar absorption of the mixing coefficients *diff_cnt* and *diff_cnu* in the northward diffusion of tracers and momentum. The coefficients A_m and A_h are also independent of time and are input through namelist. Refer to Section 5.4 for information on namelist variables.

Spatially varying mixing coefficients

It may be desirable to set horizontal mixing coefficients as a function of grid size when the grid is non-uniform. This is useful when large mixing coefficients are needed for numerical

[10] To save memory although this results in complications with different numbers of subscripts being used for various mixing parameterizations.

stability in coarse regions (outside the area of interest) but are too large for regions of high resolution (within the area of interest). Currently, there is no option for doing this. For tracer diffusion coefficients the change is easy. For example, suppose the grid is stretched in latitude. Adding a variable diffusion coefficient is a matter of substituting $A_h \cdot \Gamma_{jrow}$ for $diff\_cet$ in Equation (15.53) where Γ_{jrow} is a scale factor depending on grid size. Variable mixing coefficients for momentum are a bit more difficult. In addition to making a similar substitution in Equation (15.52), additional terms given in Wajsowicz (1993) must be included. Note that option *smagnlmix* does include such terms.

15.15.2 biharmonic

This is a basic mixing parameterization which is best thought of as another form of constant horizontal mixing that replaces the ∇^2 terms by ∇^4 terms and uses biharmonic mixing coefficients A_{mbi} and A_{hbi} instead of A_m and A_h in Equations (2.8), (2.9), (2.3), and (2.4). A rough scaling argument is given by

$$A_m \nabla^2(\alpha) \approx -A_{mbi} \nabla^4(\alpha) \tag{15.56}$$

which leads to

$$A_{mbi} \approx -\Delta^2 \cdot A_m \tag{15.57}$$

where Δ is the grid scale.

To compute ∇^4 mixing terms for each tracer component, second order ∇^2 mixing terms are computed first using the diffusion operators given by Equations (11.58) and (11.59). The second order mixing term is given by

$$del2_{i,k,j,n+2} = DIFF\_Tx_{i,k,j} + DIFF\_Ty_{i,k,j} \tag{15.58}$$

for $n = 1, nt$ tracer components assuming the northern and eastern fluxes of tracer have been computed using Equations (11.42) and (11.43). The offset of 2 is intended to leave room for two velocity components. Fluxes of $del2_{i,k,j,n+2}$ on the eastern and northern sides of T cells are then computed using

$$\begin{aligned} diff\_fe_{i,k,j} &= \frac{diff\_cet_{i,k,j}}{\cos\phi^T_{jrow}} \delta_\lambda(del2_{i,k,j,n+2}) \\ &= diff\_cet_{i,k,j} \cdot \frac{del2_{i+1,k,j,n+2} - del2_{i,k,j,n+2}}{\cos\phi^T_{jrow} dxu_i}, j = jsmw, jemw \end{aligned} \tag{15.59}$$

$$\begin{aligned} diff\_fn_{i,k,j} &= diff\_cnt_{i,k,j} \cdot \cos\phi^U_{jrow} \delta_\phi(del2_{i,k,j,n+2}) \\ &= diff\_cnt_{i,k,j} \cdot \cos\phi^U_{jrow} \frac{del2_{i,k,j+1,n+2} - del2_{i,k,j,n+2}}{dyu_{jrow}}, j = 1, jemw \end{aligned} \tag{15.60}$$

Using these fluxes in operators given by Equations (11.58) and (11.59), the biharmonic mixing terms are construced. A similar process is repeated for velocity components $n = 1, 2$ using

$$del2_{i,k,j,n} = DIFF\_Ux_{i,k,j} + DIFF\_Uy_{i,k,j} + DIFF\_metric_{i,k,j,n} \tag{15.61}$$

where the northern and eastern fluxes of velocity have been computed using Equtaions (11.88) and (11.89). Fluxes of $del2_{i,k,j,n}$ on the eastern and northern sides of U cells along with the metric term are then computed using

$$
\begin{aligned}
diff\_fe_{i,k,j} &= \frac{diff\_ceu_{i,k,j}}{\cos\phi^U_{jrow}}\delta_\lambda(del2_{i,k,j,n}) \\
&= diff\_ceu_{i,k,j}\cdot\frac{del2_{i+1,k,j,n}-del2_{i,k,j,n}}{\cos\phi^U_{jrow}dxt_{i+1}}\ , j = jsmw, jemw \qquad (15.62)
\end{aligned}
$$

$$
\begin{aligned}
diff\_fn_{i,k,j} &= diff\_cnu_{i,k,j}\cdot\cos\phi^T_{jrow+1}\delta_\phi(del2_{i,k,j,n}) \\
&= diff\_cnu_{i,k,j}\cdot\cos\phi^T_{jrow+1}\frac{del2_{i,k,j+1,n}-del2_{i,k,j,n}}{dyt_{jrow+1}}\ , j = 1, jemw
\end{aligned}
\qquad (15.63)
$$

$$
\begin{aligned}
DIFF\_metric^{new}_{i,k,j,n} &= A_m\frac{1-\tan^2\phi^U_{jrow}}{radius^2}del2_{i,k,j,n} \\
&\mp A_m\frac{2\sin\phi^U_{jrow}}{radius\cdot\cos^2\phi^U_{jrow}}\frac{del2_{i+1,k,j,3-n}-del2_{i-1,k,j,3-n}}{dxt_i+dxt_{i+1}}
\end{aligned}
\qquad (15.64)
$$

Two extra boundary conditions are required and are taken from conditions on ∇^2 at land boundaries. Mixing coefficients are independent of space and time and the *biharmonic* option requires option *consthmix* to be enabled as well. Note that since this is a fourth order scheme, the minimum size of the memory window is $jmw = 4$ instead of $jmw = 3$ required for second order differences. As with all fourth order schemes, option *fourth_order_memory_window* must be enabled to handle this option. This is automatically done when option *biharmonic* is enabled. Mixing coefficients are input through namelist. Refer to Section 5.4 for information on namelist variables.

15.15.3 smagnlmix

This is a basic mixing scheme that formulates the mixing of momentum and tracers in terms of functions of local stress and strain in Equations (2.8), (2.9), (2.3), and (2.4). The mixing coefficients are functions of space and time as given in Smagorinsky (1963), and Deardorff (1973). The method is summarized in Rosati and Miyakoda (1988) and the additional viscous terms of Wajsowicz (1993) are accounted for. Values required for this option are input through namelist. Refer to Section 5.4 for information on namelist variables.

In this formulation, the momentum mixing terms given by Equations (2.8) and (2.9) are replaced by the finite difference forms for Equations (2.18) and (2.19) given in Rosati and Miyakoda (1988). To solve these finite difference equations, tension ($strain_{i,k,j,1}$) and shearing ($strain_{i,k,j,2}$) rates of strain are computed on the northern face of U cells as

$$
strain_{i,k,j,1} = \frac{1}{\cos\phi^T_{jrow+1}}\overline{\delta_\lambda(\overline{u_{i-1,k,j,1,\tau-1}}^\phi)}^\lambda - \cos\phi^T_{jrow+1}\delta_\phi(\frac{u_{i,k,j,2,\tau-1}}{\cos\phi^U_{jrow}}) \qquad (15.65)
$$

$$
strain_{i,k,j,2} = \frac{1}{\cos\phi^T_{jrow+1}}\overline{\delta_\lambda(\overline{u_{i-1,k,j,2,\tau-1}}^\phi)}^\lambda + \cos\phi^T_{jrow+1}\delta_\phi(\frac{u_{i,k,j,1,\tau-1}}{\cos\phi^U_{jrow}}) \qquad (15.66)
$$

and used to compute viscosity coefficients on the north face of U cells

$$
[A^\lambda_M]_{i,k,j} = c^\lambda|D| \qquad (15.67)
$$

$$
[A^\phi_M]_{i,k,j} = c^\phi|D| \qquad (15.68)
$$

where the the brackets are only intended to isolate the subscripts. The effective anisentropic wavenumbers[11] for diffusing turbulence and total deformation are given by

$$
\begin{aligned}
c &= 0.14 && (15.69)\\
c^{\lambda} &= \frac{(c \cdot \cos\phi^U_{jrow} dxu_i)^2}{\sqrt{2}} && (15.70)\\
c^{\phi} &= \frac{(c \cdot dyu_{jrow})^2}{\sqrt{2}} && (15.71)\\
|D| &= \sqrt{2(strain^2_{i,k,j,1} + strain^2_{i,k,j,2})} && (15.72)
\end{aligned}
$$

Rosati (personal communication) indicates that a value of $c = 0.14$ leads to under-diffused solutions and he has used $c = 0.28$. When applied to the ocean, it is unknown how well this number c is known or even to what extent it may be a function of resolution. In terms of the above equations, the viscous terms given by Equations (2.8) and (2.9) are formulated in discrete form as

$$
\begin{aligned}
F^u &= \frac{1}{\cos\phi^U_{jrow}}[\delta_\lambda([A^\lambda_M]_{i,k,j} \cdot strain_{i,k,j,1})\\
&+\frac{1}{\cos\phi^U_{jrow}}\delta_\phi([A^\phi_M]_{i,k,j} \cdot strain_{i,k,j,2} \cdot \cos^2\phi^T_{jrow+1})] && (15.73)\\
F^v &= \frac{1}{\cos\phi^U_{jrow}}[\delta_\lambda([A^\lambda_M]_{i,k,j} \cdot strain_{i,k,j,2}) - \delta_\phi([A^\phi_M]_{i,k,j} \cdot strain_{i,k,j,1} \cdot \cos^2\phi^T_{jrow+1})\\
&+\frac{\sin\phi^U_{jrow}}{radius}[A^\lambda_M]_{i,k,j} \cdot strain_{i,k,j,1}] && (15.74)
\end{aligned}
$$

The second term in Equation (15.73) is not in flux form. To put it in flux form leads to more calculation than handling it as a metric term and so it is computed as one. For diagnostic purposes, the viscosity coefficients on the northern and eastern face of U cells can be estimated as

$$
visc\_cnu_{i,k,j} = [A^\phi_M]_{i,k,j} \quad (15.75)
$$

$$
visc\_ceu_{i,k,j} = \overline{[A^\lambda_M]_{i,k,j-1}}^{\lambda,\phi} \quad (15.76)
$$

although these viscous coefficients are only used for diagnostic purposes. The first term in Equations (15.73) and (15.74) is in flux form and the flux across the eastern face of the U cell is given by

$$
diff\_fe_{i,k,j} = \overline{[A^\lambda_M]_{i,k,j-1} \cdot strain_{i,k,j-1,n}}^{\lambda,\phi} \quad (15.77)
$$

where $n = 1$ is for the Equation (15.73) and $n = 2$ is for Equation (15.74). The second term in Equation (15.74) is in flux form but the second term in Equation (15.73) is not which yields

[11] Equations (2.28) and (2.29) in Rosati/Miyakoda (1988) used 2 in the denominator instead of $\sqrt{2}$ which was also used in MOM 1. Also, m in Equation (2.28) should be m^{-1}.

the following form for the meridional flux through the northern face of the U cell. When $n = 1$, the form is

$$diff\_fn_{i,k,j} = 0 \tag{15.78}$$

and when $n = 2$, the form is

$$diff\_fn_{i,k,j} = \delta_\phi(-[A_M^\phi]_{i,k,j-1} \cdot strain_{i,k,j-1,1} \cdot \cos\phi_{jrow}^T) \tag{15.79}$$

Treating the second term in Equation (15.73) as a metric term yields the following form when $n = 1$

$$smag\_metric_{i,k,j} = \frac{1}{\cos^2\phi_{jrow}^U}\delta_\phi([A_M^\phi]_{i,k,j-1} \cdot strain_{i,k,j-1,2} \cdot \cos^2\phi_{jrow}^T) \tag{15.80}$$

and when $n = 2$, the form is

$$smag\_metric_{i,k,j} = \frac{\sin\phi_{jrow}^U}{radius \cdot \cos\phi_{jrow}^U}\overline{[A_M^\lambda]_{i,k,j-1} \cdot strain_{i,k,j-1,1}}^{\phi} \tag{15.81}$$

For diffusion of tracers, there is no metric term and the fluxes across T cell faces can be computed directly as

$$diff\_fe_{i,k,j} = [A_M^\lambda]_{i,k,j-1} + diff\_c\_back \tag{15.82}$$

$$diff\_fn_{i,k,j} = \overline{[A_M^\phi]_{i-1,k,j-1}}^{\lambda,\phi} + diff\_c\_back \tag{15.83}$$

where $diff\_c\_back$ is an arbitrary background value. Note that the fourth m in Equation (2.34) of Rosati/Miyakoda (1988) should be m^{-1}.

15.16 Hybrid mixing schemes

Hybrid mixing schemes affect mixing of tracers but not momentum. They may be enabled concurrently with one basic mixing option.

15.16.1 bryan_lewis_vertical

This is a hybrid mixing scheme (Bryan, Lewis 1979) which specifies the vertical diffusion coefficient for tracers κ_h in Equations (2.3), and (2.4) as a depth dependent function given by

$$diff\_cbt_{i,k,j} = afkph + \frac{dfkph}{\pi}\arctan(sfkph \cdot (zw_k - zfkph)) \tag{15.84}$$

where $afkph = 0.8\ cm^2/sec$, $dfkph = 1.05\ cm^2/sec$, $sfkph = 4.5\text{x}10^{-5}$, and $zfkph = 2500.0\text{x}10^2\ cm$. These values imply a diffusivity coefficient ranging from 0.3 cm^2/sec near the surface of the ocean to 1.3 cm^2/sec near the bottom. Values required for this option can be changed through namelist. Refer to Section 5.4 for information on namelist variables.

15.16.2 bryan_lewis_horizontal

This is a hybrid mixing scheme (Bryan, Lewis 1979) which specifies the horizontal diffusion coefficient for tracers A_h in Equations (2.3), and (2.4) to be a function of depth given by

$$diff\_cet_k = diff\_cnt_k = (ahb + (ahs - ahb)\exp(-\frac{zt_k}{50000.0})) \cdot 1.0\text{x}10^4 \tag{15.85}$$

with $ahs = 5.0\text{x}10^3\ cm^2/sec$ and $ahs = 1.0\text{x}10^3\ cm^2/sec$. Note that $diff\_cet_k$ and $diff\_cnt_k$ are a function of k only (to conserve memory). It should be noted that the functional relationship can easily be changed by the researcher. Values required for this option can be changed through namelist. Refer to Section 5.4 for information on namelist variables.

15.16.3 isopycmix

Option *isopycmix* enables a hybrid mixing scheme which mixes tracers (but not momentum) along locally defined neutral directions by means of vertical and horizontal mixing in Equations (2.3), and (2.4). One basic horizontal mixing scheme (e.g. option *consthmix*) and one basic vertical mixing scheme (e.g. option *constvmix*) must also be enabled for use with option *isopycmix*. Diffusion coefficients for tracers from the basic mixing schemes are used as background coefficients in the horizontal and vertical but these may be set to zero. The isopycnal mixing coefficient A_I (typically $O(10^7 cm^2/sec)$) along with the maximum allowable isopycnal slope S_{max} (typically 1/100 and referred to as variable $slmx$ in the model) are input through namelist. Refer to Section 5.4 for information on namelist variables.

Sub-options

The following options are available for use with option *isopycmix*:

- Option *full_tensor* includes all terms in the Redi tensor. If this option is not used, then the small angle approximation to the Redi tensor is used. Note that the full tensor is very computationally expensive but is most accurate in regions of strongly sloping isopycnals.

- Option *dm_taper* enables the hyperbolic tangent re-scaling for steep slopes by Danabasoglu and McWilliams (NCAR 1996). Where isopycnal slopes exceed the critical slope $slmx$, the re-scaling factor is

$$scaling = 0.5(1 - \tanh(\frac{|S| - del^{dm}}{S^{dm}})) \tag{15.86}$$

  where S is the local isopycnal slope, and del^{dm} and S^{dm} are namelist input parameters. Refer to Section 5.4 for information on namelist variables. If option *dm_taper* is not enabled, the re-scaling of Gerdes et al (1991) is used

$$scaling = (\frac{slmx}{|S|})^2 \tag{15.87}$$

  Neither of these ad-hoc re-scalings apply to option *full_tensor* which is re-scaled as given in Section B.3.3.

- Option *gent_mcwilliams* enables the Gent-McWilliams parameterization for the effect of mesoscale eddies on isopycnals as given in Section 15.16.4.

- Option *held_larichev* enables a higher order scheme for prediciting the isopycnal mixing coefficient as explained in Section 15.16.5. However, this one is not ready yet.

Details

This section presents a summary of the discretization of the isopycnal diffusion fluxes. The full details for the derivation and explanation of the labels are presented in Appendix B. Further reference should be made to the manuscript in preparation by Griffies et al. (1997).

The discretization for the isopycnal mixing tensor given below assumes the default grid construction used in MOM 2. It is not correct for the grid option *centered_t* which always centers grid points within T cells as in MOM 1. Also, details of the Fortran coding may change in an attempt to reduce memory requirements and increase computational speed.

Zonal Flux

The isopycnal component of the diffusive flux through the eastern face of a T cell $diff\_fet^{iso}_{i,k,j}$ for the full Redi tensor is given by

$$\begin{aligned}
diff_fet^{iso}_{i,k,j} \equiv -F^x_{i,k,j} = \frac{1}{4dyt_j \cos\phi^T_j} \sum_{jq=0}^{1} dyu_{j-1+jq} \cos\phi^U_{j-1+jq} \sum_{ip=0}^{1} A^o_I \times \\
\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq} \left(\frac{\delta_x T_{i,k,j}\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq} - \delta_x\rho^{(i+ip,k,j)}_{i,k,j}\delta_y T_{i+ip,k,j-1+jq}}{(\delta_x\rho^{(i+ip,k,j)}_{i,k,j})^2 + (\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2 + .5\sum_{kr=0,1}(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \right) \\
+ \frac{1}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{ip=0}^{1} A^{(i+ip,k,j)}_{ez}(i,k,j|i+ip,k-1+kr,j) \times \\
\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j} \left(\frac{\delta_x T_{i,k,j}\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j} - \delta_z T_{i+ip,k-1+kr,j}\delta_x\rho^{(i+ip,k,j)}_{i,k,j}}{(\delta_x\rho^{(i+ip,k,j)}_{i,k,j})^2 + .5\sum_{jq=0,1}(\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2 + (\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \right)
\end{aligned} \tag{15.88}$$

This discretization should be compared to the continuum version in equation (B.18). The convergence of the discrete form in the continuum limit is manifest. It is not possible to identify the off-diagonal components of the Redi tensor independently of the tracers. However, it is possible to identify the non-negative diagonal component K^{11}

$$\begin{aligned}
K^{11}_{i,k,j} = \frac{1}{4dyt_j \cos\phi^T_j} \sum_{jq=0}^{1} dyu_{j-1+jq} \cos\phi^U_{j-1+jq} \sum_{ip=0}^{1} A^o_I \\
\frac{(\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2}{(\delta_x\rho^{(i+ip,k,j)}_{i,k,j})^2 + (\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2 + .5\sum_{kr=0,1}(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \\
+ \frac{1}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{ip=0}^{1} A^{(i+ip,k,j)}_{ez}(i,k,j|i+ip,k-1+kr,j) \\
\frac{(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2}{(\delta_x\rho^{(i+ip,k,j)}_{i,k,j})^2 + .5\sum_{jq=0,1}(\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2 + (\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2},
\end{aligned} \tag{15.89}$$

which brings the x-flux to the form

$$\begin{aligned}
-F^x_{i,k,j} \equiv diff_fet^{iso}_{i,k,j} = K^{11}_{i,k,j}\delta_x T_{i,k,j} \\
- \frac{1}{4dyt_j \cos\phi^T_j} \sum_{jq=0}^{1} dyu_{j-1+jq} \cos\phi^U_{j-1+jq} \sum_{ip=0}^{1} A^o_I \; \delta_y T_{i+ip,k,j-1+jq} \times \\
\left(\frac{\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq} \; \delta_x\rho^{(i+ip,k,j)}_{i,k,j}}{(\delta_x\rho^{(i+ip,k,j)}_{i,k,j})^2 + (\delta_y\rho^{(i+ip,k,j)}_{i+ip,k,j-1+jq})^2 + .5\sum_{kr=0,1}(\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j})^2} \right)
\end{aligned}$$

$$
- \frac{1}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{ip=0}^{1} A_{ez}^{(i+ip,k,j)}(i,k,j|i+ip,k-1+kr,j)\, \delta_z T_{i+ip,k-1+kr,j} \times
\left(\frac{\delta_x \rho_{i,k,j}^{(i+ip,k,j)} \delta_z \rho_{i+ip,k-1+kr,j}^{(i+ip,k,j)}}{(\delta_x \rho_{i,k,j}^{(i+ip,k,j)})^2 + .5 \sum_{jq=0,1} (\delta_y \rho_{i+ip,k,j-1+jq}^{(i+ip,k,j)})^2 + (\delta_z \rho_{i+ip,k-1+kr,j}^{(i+ip,k,j)})^2} \right). \quad (15.90)
$$

The small slope limit is found by taking the limit $|\delta_x \rho|, |\delta_y \rho| << |\delta_z \rho|$. The diagonal component of the Redi tensor in this limit is

$$
K_{i,k,j}^{11\, small} = \frac{1}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{ip=0}^{1} A_{ez}^{(i+ip,k,j)}(i,k|i+ip,k-1+kr,j), \quad (15.91)
$$

where the diffusion coefficient $Ax^{(i+ip,k,j)}(i,k|i+ip,k-1+kr,j)$ is now determined by the small slope constraints discussed in Section B.2.7 in the Appendix. The small slope diffusive flux then takes the form

$$
-F_{i,k,j}^{x\, small} \equiv \mathit{diff\_fet}_{i,k,j}^{iso} = K_{i,k,j}^{11\, small} \delta_x T_{i,k,j}
- \frac{1}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{ip=0}^{1} A_{ez}^{(i+ip,k,j)}(i,k|i+ip,k-1+kr,j)\, \delta_z T_{i+ip,k-1+kr,j} \times
\left(\frac{\delta_x \rho_{i,k,j}^{(i+ip,k,j)}}{\delta_z \rho_{i+ip,k-1+kr,j}^{(i+ip,k,j)}} \right). \quad (15.92)
$$

<u>Meridional Flux</u>

The isopycnal component of the diffusive flux through the northern face of a T cell $diff\_fnt_{i,k,j}^{iso}$ for the full Redi tensor is given by

$$
\mathit{diff\_fnt}_{i,k,j}^{iso} \equiv -F_{i,k,j}^{y} = \cos\phi_j^U\, K_{i,k,j}^{22} \delta_y T_{i,k,j} - \frac{\cos\phi_j^U}{4dxt_i} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{jq=0}^{1} A_I^o\, \delta_x T_{i-1+ip,k,j+jq} \times
\left(\frac{\delta_x \rho_{i-1+ip,k,j+jq}^{(i,k,j+jq)}\ \delta_y \rho_{i,k,j}^{(i,k,j+jq)}}{(\delta_x \rho_{i-1+ip,k,j+jq}^{(i,k,j+jq)})^2 + (\delta_y \rho_{i,k,j}^{(i,k,j+jq)})^2 + .5 \sum_{kr=0,1} (\delta_z \rho_{i,k-1+kr,j+jq}^{(i,k,j+jq)})^2} \right)
- \frac{\cos\phi_j^U}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{jq=0}^{1} A_{nz}^{(k,j+jq)}(i,k,j|i,k-1+kr,j+jq)\, \delta_z T_{i,k-1+kr,j+jq}
\left(\frac{\delta_z \rho_{i,k-1+kr,j+jq}^{(i,k,j+jq)}\ \delta_y \rho_{k,j}^{(i,k,j+jq)}}{.5 \sum_{ip=0,1} (\delta_x \rho_{i-1+ip,k,j+jq}^{(i,k,j+jq)})^2 + (\delta_y \rho_{i,k,j}^{(i,k,j+jq)})^2 + (\delta_z \rho_{i,k-1+kr,j+jq}^{(i,k,j+jq)})^2} \right), \quad (15.93)
$$

where the non-negative diagonal component K^{22} of the Redi tensor is given by

$$
K_{i,k,j}^{22} = \frac{1}{4dxt_i} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{jq=0}^{1} A_I^o
\frac{(\delta_x \rho_{i-1+ip,k,j+jq}^{(i,k,j+jq)})^2}{(\delta_x \rho_{i-1+ip,k,j+jq}^{(i,k,j+jq)})^2 + (\delta_y \rho_{i,j}^{(i,k,j+jq)})^2 + .5 \sum_{kr=0,1} (\delta_z \rho_{i,k-1+kr,j+jq}^{(i,k,j+jq)})^2}
+ \frac{1}{4dzt_k} \sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{jq=0}^{1} A_{nz}^{(i,k,j+jq)}(i,k,j|i,k-1+kr,j+jq)
$$

$$\frac{(\delta_z \rho^{(i,k,j+jq)}_{i,k-1+kr,j+jq})^2}{.5\sum_{ip=0,1}(\delta_x \rho^{(i,k,j+jq)}_{i-1+ip,k,j+jq})^2 + (\delta_y \rho^{(i,k,j+jq)}_{i,k,j})^2 + (\delta_z \rho^{(i,k,j+jq)}_{i,k-1+kr,j+jq})^2}. \tag{15.94}$$

The small angle limit of these results yields

$$K^{22\ small}_{i,k,j} = \frac{1}{4dzt_k}\sum_{kr=0}^{1} dzw_{k-1+kr}\sum_{jq=0}^{1} A^{(i,k,j+jq)}_{nz}(i,k,j|i,k-1+kr,j+jq) \tag{15.95}$$

$$\begin{aligned} -F^{y\ small}_{i,k,j} &\equiv \mathit{diff\_fnt}^{iso}_{i,k,j} = \cos\phi^U_j K^{22}_{i,k,j}\delta_y T_{i,k,j} \\ - &\frac{\cos\phi^U_j}{4dzt_k}\sum_{kr=0}^{1} dzw_{k-1+kr}\sum_{jq=0}^{1} A^{(i,k,j+jq)}_{nz}(i,k,j|i,k-1+kr,j+jq)\,\delta_z T_{i,k-1+kr,j+jq}\times \\ &\left(\frac{\delta_y \rho^{(i,k,j+jq)}_{i,k,j}}{\delta_z \rho^{(i,k,j+jq)}_{i,k-1+kr,j+jq}}\right). \end{aligned} \tag{15.96}$$

Vertical Flux

The isopycnal component of the diffusive flux through the bottom face of a T cell $diff\_fbt^{iso}_{i,k,j}$ for the full Redi tensor is broken into two parts: the $K^{33}_{i,k,j}$ component and the off diagonal component $diff\_fbiso_{i,k,j}$ which contains $K^{31}_{i,k,j}$ and $K^{32}_{i,k,j}$ pieces. The vertical diffusion term for tracers is also broken into two parts for isopycnal mixing: the part containing $diff\_fbiso_{i,k,j}$ is solved explicitly with all other explicit components and the part containing the $K^{33}_{i,k,j}$ component is solved implicitly due to restrictions on stability. Refer to Section 11.10.3 for details.

$$\begin{aligned} &K^{33}_{i,k,j}\delta_z T_{i,k,j} + \mathit{diff\_fbiso}_{i,k,j} \equiv -F^z_{i,k,j} = K^{33}_{i,k,j}\delta_z T_{i,k,j} \\ - &\frac{1}{4dxt_i}\sum_{ip=0}^{1} dxu_{i-1+ip}\sum_{kr=0}^{1} A^{(i,k+kr,j)}_{bx}(i-1+ip,k+kr,j|i,k,j)\,\delta_x T_{i-1+ip,k+kr,j}\times \\ &\left(\frac{\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j}\ \delta_z \rho^{(i,k+kr,j)}_{i,k,j}}{(\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + .5\sum_{jq=0,1}(\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr,j)}_{i,k,j})^2}\right) \\ - &\frac{1}{4\cos\phi^T_j\, dyt_j}\sum_{jq=0}^{1}\cos\phi^U_{j-1+jq} dyu_{j-1+jq}\sum_{kr=0}^{1} A^{(i,k+kr,j)}_{by}(i,k+kr,j-1+jq|i,k,j)\,\delta_y T_{i,k+kr,j-1+jq}\times \\ &\left(\frac{\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq}\ \delta_z \rho^{(i,k+kr,j)}_{i,k,j}}{.5\sum_{ip=0,1}(\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + (\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr,j)}_{i,k,j})^2}\right). \end{aligned} \tag{15.97}$$

where the the non-negative K^{33} component of the Redi tensor is given by

$$\begin{aligned} K^{33}_{i,k,j} = &\frac{1}{4dxt_i}\sum_{ip=0}^{1} dxu_{i-1+ip}\sum_{kr=0}^{1} A^{(i,k+kr,j)}_{bx}(i-1+ip,k+kr,j|i,k,j) \\ &\frac{(\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2}{(\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + .5\sum_{jq=0,1}(\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr,j)}_{i,k,j})^2} \\ + &\frac{1}{4\cos\phi^T_j\, dyt_j}\sum_{jq=0}^{1}\cos\phi^U_{j-1+jq} dyu_{j-1+jq}\sum_{kr=0}^{1} A^{(i,k+kr,j)}_{by}(i,k+kr,j-1+jq|i,k,j) \\ &\frac{(\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2}{.5\sum_{ip=0,1}(\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j})^2 + (\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq})^2 + (\delta_z \rho^{(i,k+kr,j)}_{i,k,j})^2}. \end{aligned} \tag{15.98}$$

The small angle limit of these results yields

$$
\begin{aligned}
K^{33}_{i,k,j} &= \frac{1}{4dxt_i} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} A_{bx}^{(i,k+kr,j)}(i-1+ip,k+kr,j|i,k,j) \times \\
&\left(\frac{\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j}}{\delta_z \rho^{(i,k+kr,j)}_{i,k,j}} \right)^2 \\
+ &\frac{1}{4\cos\phi^T_j \, dyt_j} \sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} A_{by}^{(i,k+kr,j)}(i,k+kr,j-1+jq|i,k,j)\, \delta_y T_{i,k+kr,j-1+jq} \times \\
&\left(\frac{\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq}}{\delta_z \rho^{(i,k+kr,j)}_{i,k,j}} \right)^2, \qquad (15.99) \\
-F^z_{i,k,j} &= K^{33}_{i,k,j} \delta_z T_{i,k,j} \\
- &\frac{1}{4dxt_i} \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{kr=0}^{1} A_{bx}^{(i,k+kr,j)}(i-1+ip,k+kr,j|i,k,j) \delta_x T_{i-1+ip,k+kr,j} \times \\
&\left(\frac{\delta_x \rho^{(i,k+kr,j)}_{i-1+ip,k+kr,j}}{\delta_z \rho^{(i,k+kr,j)}_{i,k,j}} \right) \\
- &\frac{1}{4\cos\phi^T_j \, dyt_j} \sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} A_{bx}^{(i,k+kr,j)}(i,k+kr,j-1+jq|i,k,j)\, \delta_y T_{i,k+kr,j-1+jq} \times \\
&\left(\frac{\delta_y \rho^{(i,k+kr,j)}_{i,k+kr,j-1+jq}}{\delta_z \rho^{(i,k+kr,j)}_{i,k,j}} \right). \qquad (15.100)
\end{aligned}
$$

Section 15.16.3 contributed by
Stephen M. Griffies
smg@gfdl.gov
Last revised October 1996

15.16.4 gent_mcwilliams

Option *gent_mcwilliams* applies only when option *isopycmix* is enabled. It calculates the effect of mesoscale eddies on isopycnals in terms of eddy induced transport (advective) velocities. This section assumes familiarity with Section 15.16.3 and is based on the notes of Gokhan Danabasoglu which have been re-written for the terminology and implementation in MOM 2. For a commentary on the general properties of tracer mixing, refer to Appendix A. Values required for this option are input through namelist. Refer to Section 5.4 for information on namelist variables.

The eddy-induced advective velocities, as with the regular advection velocities in MOM 2, are computed at the centers of the eastern, northern, and bottom faces of the cells. The velocities are given by $adv\_vetiso_{i,k,j}$, $adv\_vntiso_{i,k,j}$, and $adv\_vbtiso_{i,k,j}$ respectively. In MOM 2 version 1, the eddy induced transport velocities were discretized based on the notes of Gokhan Danabasoglu who implemented the Gent/McWilliams (1990) isopycnal parameterization as

$$adv\_vetiso_{i,k,j} = -\delta_z\left(\overline{\frac{A_{ITH}}{A_I}[K^{13}]_{i,k-1,j}}^{z}\right) \qquad (15.101)$$

$$adv\_vntiso_{i,k,j} = -\delta_z(\overline{\frac{A_{ITH}}{A_I}[K^{23}]_{i,k-1,j}}^{z}) \tag{15.102}$$

However, the above form contains a null mode and has been replaced by the following

$$adv\_vetiso_{i,k,j} = -\delta_z(A_{ITH} \cdot S^{xb}_{i,k-1,j}) \tag{15.103}$$
$$adv\_vntiso_{i,k,j} = -\delta_z(A_{ITH} \cdot S^{yb}_{i,k-1,j}) \tag{15.104}$$

where A_{ITH} is the isopycnal thickness diffusion coefficient and the neutral slope in the zonal direction at the bottom of the eastern face of a T grid cell is given by

$$S^{xb}_{i,k,j} = -\frac{\overline{\alpha_{i,k,j}}^{\lambda,z}\overline{\delta_\lambda(t_{i,k,j,1,\tau-1})}^{z} + \overline{\beta_{i,k,j}}^{\lambda,z}\overline{\delta_\lambda(t_{i,k,j,2,\tau-1})}^{z}}{\overline{\alpha_{i,k,j}}^{\lambda,z}\overline{\delta_z(t_{i,k,j,1,\tau-1})}^{\lambda} + \overline{\beta_{i,k,j}}^{\lambda,z}\overline{\delta_z(t_{i,k,j,2,\tau-1})}^{\lambda}} \tag{15.105}$$

and the neutral slope in the meridional direction at the bottom of the northern face of a T grid cell is given by

$$S^{yb}_{i,k,j} = -\frac{\overline{\alpha_{i,k,j}}^{\phi,z}\overline{\delta_\phi(t_{i,k,j,1,\tau-1})}^{z} + \overline{\beta_{i,k,j}}^{\phi,z}\overline{\delta_\phi(t_{i,k,j,2,\tau-1})}^{z}}{\overline{\alpha_{i,k,j}}^{\phi,z}\overline{\delta_z(t_{i,k,j,1,\tau-1})}^{\phi} + \overline{\beta_{i,k,j}}^{\phi,z}\overline{\delta_z(t_{i,k,j,2,\tau-1})}^{\phi}} \tag{15.106}$$

where the $\alpha_{i,k,j}$ and $\beta_{i,k,j}$ are defined as in Section B.2.6.
The vertical component of the isopycnal advection velocity is obtained by vertically integrating the divergence of the horizontal isopycnal advection velocities as is done in the notes of Gokhan Danabasoglu. Note that the requirement of zero vertical isopycnal advection velocity at the top face of $cell_{i,k=1,j}$ and bottom face of $cell_{i,k=bottom,j}$ indicates that S^{xb} and S^{yb} must be zero at these surfaces.

$$adv\_vbtiso_{i,k,j} = \frac{1}{\cos\phi^T_{jrow}} \sum_{m=1}^{k} \left[\delta_\lambda(adv\_vetiso_{i-1,m,j}) + \delta_\phi(adv\_vntiso_{i,m,j-1})\right] \cdot dzt_m \tag{15.107}$$

The eddy-induced advection terms are discretized as:

$$\mathcal{L}^{gm}(\gamma_{i,k,j}) = \frac{1}{\cos\phi^T_{jrow}}\left[\delta_\lambda(adv\_vetiso_{i-1,k,j}\overline{\gamma_{i-1,k,j}}^{\lambda}) + \delta_\phi(adv\_vntiso_{i,k,j-1}\overline{\gamma_{i,k,j-1}}^{\phi})\right]$$
$$- \quad \delta_z(adv\_vbtiso_{i,k-1,j}\overline{\gamma_{i,k-1,j}}^{z}) \tag{15.108}$$

where $adv\_vntiso_{i,k,j-1}$ contains an embedded cosine factor as does $adv\_vnt_{i,k,j}$. Refer to Section 11.10.7 for a definition of the Gent/McWilliams advective operators.

15.16.5 held_larichev

This option calculates isopycnal mixing coefficients (A_I and A_{ITH}) based on vertically integrated Richardson numbers as given in Held and Larichev (1995). This is experimental and as of this writing needs more work to restrict the mixing to the levels where baroclinic instability is possible. Therefore, this scheme is not suitable for use at this time. Values required for this option are input through namelist. Refer to Section 5.4 for information on namelist variables.

15.17 Eddy interaction parameterizations

15.17.1 neptune

Option *neptune* implements the following. Based on statistical mechanics arguments, Holloway (1992) proposed that interaction between mesoscale eddies and topography results in a stress on the ocean with two important consequences: first, the ocean is not driven towards a state of rest and secondly, the resulting motion may have scales much larger than the scale of the eddies. Somewhat suprisingly, this interaction[12] can generate coherent mean flows on the scale of the topography. The magnitude of this topographic stress is dependent on the correlation between pressure p and topographic gradients ∇H which is largely unknown but even if the correlation is 0.1, the resulting topographic stress would be comparable in magnitude to that of the surface wind.

If the view is taken that equations of motion are solved for moments of probable flow (because of imperfect resolution) then those moments are forced in part by derivatives of the distribution entropy with respect to the realized moments. The entropy gradient is estimated as begin proportional to a departure of the realized moments from a state in which the entropy gradient is weak. This latter state is approximated by a transport streamfunction $\psi^\star$ and maximum entropy velocity $u^\star$ given by

$$\psi^\star = -fL'H \tag{15.109}$$
$$u^\star = \hat{z} \times \nabla \psi^\star \tag{15.110}$$

where f is the Coriolis term, H is depth, and L' is O (10 km). If model resolution is coarse relative to the first deformation radius, $u^\star$ is independent of depth. Instead of eddy viscosity driving flow towards rest, flow is driven towards $u^\star$ using an eddy viscosity of the form $A\nabla^2(u^\star - u)$. Note that topographic influence on flow[13] is not strongest near bottom topography. Instead, the flow implied by $\psi^\star$ only approximates a maximum entropy system given eddies and topography. Since this approximation is admittedly crude, further refinements are open to researchers.

There is legitimate concern about the stepwise resolution of bottom topography in level models such as MOM 2 and its predecessors. Option *neptune* is an attempt to instruct the model about physical consequences due to topography and eddies which are nearly unachievable even at the most ambitious resolutions. The hope is that if the model can be suitably informed about the effect of topography, it matters little if that topography is only "approximately" represented.

15.18 Advection schemes

The advection of momentum always uses second order accurate center differencing in space and time which conserves first and second moments. In addition to this second order accurate scheme, others schemes are available for the advection of tracers. If none are enabled, then the second order accurate scheme is also used for tracers.

[12] Which is missing or at best poorly represented in numerical models at any resolution.

[13] This is referred to as the Neptune effect because when Greg Holloway described coastal currents that persistantly flow againgst both wind forcing and pressure gradient, the response was that it must be due to King Neptune. Who else?

15.18.1 second_order_tracer_advection

This option is automatically enabled in file *size.h* if no other advective schemes are enabled. Do not directly enable it with a preprocessor directive. The advective flux on the eastern, northern, and bottom sides of cell $T_{i,k,jrow}$is given by

$$
\begin{aligned}
adv\_fe_{i,k,j} &= adv\_vet_{i,k,j} \cdot (t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau})/2 && (15.111)\\
adv\_fn_{i,k,j} &= adv\_vbt_{i,k,j} \cdot (t_{i,k,j,n,\tau} + t_{i,k,j+1,n,\tau})/2 && (15.112)\\
adv\_fb_{i,k,j} &= adv\_vbt_{i,k,j} \cdot (t_{i,k,j,n,\tau} + t_{i,k+1,j,n,\tau})/2 && (15.113)
\end{aligned}
$$

Note that twice the advective flux is actually computed in the model by eliminating the division by 2 for reasons of speed optimization.

15.18.2 fourth_order_tracer_advection

Option *fourth_order_tracer_advection* replaces the second order advective scheme with a fourth order scheme and requires option *fourth_order_memory_window* to be enabled. This is automatically done when option *fourth_order_tracer_advection* is enabled. Consider any quantity q_i for $i = 1\ to\ N$ points. Expanding q_{i+1}, q_{i-1}, q_{i+2}, and q_{i-2} in a Taylor series about q_i yields

$$
\begin{aligned}
q_{i+1} &= q_i + \frac{\partial q_i}{\partial i} + \frac{1}{2}\frac{\partial^2 q_i}{\partial^2 i} + \frac{1}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{1}{64}\frac{\partial^4 q_i}{\partial^4 i} + \cdots && (15.114)\\
q_{i-1} &= q_i - \frac{\partial q_i}{\partial i} + \frac{1}{2}\frac{\partial^2 q_i}{\partial^2 i} - \frac{1}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{1}{64}\frac{\partial^4 q_i}{\partial^4 i} + \cdots && (15.115)\\
q_{i+2} &= q_i + 2\frac{\partial q_i}{\partial i} + 2\frac{\partial^2 q_i}{\partial^2 i} + \frac{8}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{16}{64}\frac{\partial^4 q_i}{\partial^4 i} + \cdots && (15.116)\\
q_{i-2} &= q_i - 2\frac{\partial q_i}{\partial i} + 2\frac{\partial^2 q_i}{\partial^2 i} - \frac{8}{6}\frac{\partial^3 q_i}{\partial^3 i} + \frac{16}{64}\frac{\partial^4 q_i}{\partial^4 i} + \cdots && (15.117)
\end{aligned}
$$

The above expansions can be combined to yield

$$
\begin{aligned}
(q_{i+1} + q_i) - (q_i + q_{i-1}) &\approx 2\frac{\partial q_i}{\partial i} + \frac{1}{3}\frac{\partial^3 q_i}{\partial^3 i} && (15.118)\\
(q_{i+2} + q_{i-1}) - (q_{i+1} + q_{i-2}) &\approx 2\frac{\partial q_i}{\partial i} + \frac{7}{3}\frac{\partial^3 q_i}{\partial^3 i} && (15.119)
\end{aligned}
$$

Multiplying Equation (15.118) by 7, subtracting Equation (15.119) and changing to grid spacing Δx yields

$$
\begin{aligned}
\partial q_i/\partial x &\approx \frac{F_i - F_{i-1}}{\Delta x} && (15.120)\\
F_i &= A(q_{i+1} + q_i) - B(q_{i+2} + q_{i-1}) && (15.121)
\end{aligned}
$$

where $A = 7/12$ and $B = 1/12$ for fourth order accuracy and $A = 1/2$ and $B = 0$ for second order accuracy. Since the argument is the same for zonal, meridional, and vertical directions, consider the zonal direction. A fully fourth order advective scheme would set

$$q_{i,k,j} = \overline{adv\_vet_{i-1,k,j}}^{\lambda} \cdot t_{i,k,j,n,\tau} \qquad (15.122)$$

However, as implemented in MOM 2, the scheme is only psuedo fourth order[14] because the advecting velocity is left second order while the average tracer on the cell faces is expanded to fourth order using

$$F_i = adv\_vet_{i,k,j} \cdot (A(t_{i+1,k,j,n,\tau} + t_{i,k,j,n,\tau}) - B(t_{i+2,k,j,n,\tau} + t_{i-1,k,j,n,\tau})) \quad (15.123)$$

Note that for ocean T cells adjacent to land cells, the scheme is reduced to second order. This is easily accomplished by using a land/sea mask to select the appropriate coefficients A and B for each T cell. In principle, a fully fourth order scheme could easily be implemented by expanding fluxes according to Equation (15.122) but this has not been done. The reason is that fourth order schemes, although more accurate on small scales, still produce undershoots and overshoots which can have significant effect on local areas of the ocean. These can be eliminated using flux corrected transport methods as described in Section 15.18.4.

15.18.3 quicker

This is a third order advection scheme for tracers which significantly reduces the over-shooting inherent in the second order center differenced advection scheme. The cost is less than a 10% increase in overall time. It is enabled by option *quicker* and is based on the "quick" scheme of Leonard (1979) but has been modified to lag the upstream correction by putting it on time level $\tau - 1$. This was first recommended (personal communication) by Jeurgen Willebrand who demonstrated it in a one dimensional advection diffusion model. The advantage of this is that it allows the same time step as for second order advection. If this is not done, the scheme is unstable unless the time step is reduced by about one half. The discretization of Farrow and Stevens (1995) has been followed but not their predictor corrector method since the lagged correction mentioned above solves the stability problem. The formulation given in NCAR (1996) is recovered by changing the $\tau - 1$ to τ in all upstream corrective terms. This is done by additionally enabling option *ncar_upwind3*. Each direction is discretized independently of others and so the scheme is un-compensated for multi-dimensions. In the zonal direction, twice the advective flux out of the eastern face of the T cell is given by

$$\begin{aligned} u^+ &= (adv\_vet_{i,k,j} + |adv\_vet_{i,k,j}|)/2 && (15.124)\\ u^- &= (adv\_vet_{i,k,j} - |adv\_vet_{i,k,j}|)/2 && (15.125)\\ adv\_fe_{i,k,j} &= adv\_vet_{i,k,j} \cdot (quick^x_{i,1} \cdot t_{i,k,j,n,\tau-1} + quick^x_{i,2} \cdot t_{i+1,k,j,n,\tau-1}) \\ &- u^+ \cdot (curv^{x+}_{i,1} \cdot t_{i+1,k,j,n,\tau-1} + curv^{x+}_{i,2} \cdot t_{i,k,j,n,\tau-1} + curv^{x+}_{i,3} \cdot t_{i-1,k,j,n,\tau-1} \\ &- u^- \cdot (curv^{x-}_{i,1} \cdot t_{i+2,k,j,n,\tau-1} + curv^{x-}_{i,2} \cdot t_{i+1,k,j,n,\tau-1} + curv^{x-}_{i,3} \cdot t_{i,k,j,n,\tau-1}) \\ & && (15.126) \end{aligned}$$

where the coefficients are given by

$$\begin{aligned} quick^x_{i,1} &= 2 \cdot dxt_{i+1}/(dxt_{i+1} + dxt_i) && (15.127)\\ quick^x_{i,2} &= 2 \cdot dxt_i/(dxt_{i+1} + dxt_i) && (15.128)\\ curv^{x+}_{i,1} &= 2 \cdot dxt_i * dxt_{i+1}/((dxt_{i-1} + 2 \cdot dxt_i + dxt_{i+1}) \cdot (dxt_i + dxt_{i+1})) && (15.129)\\ curv^{x+}_{i,2} &= -2 \cdot dxt_i * dxt_{i+1}/((dxt_i + dxt_{i+1}) \cdot (dxt_{i-1} + dxt_i)) && (15.130) \end{aligned}$$

[14]The idea of a psuedo fourth order technique was taken from the GFDL SKYHI stratospheric GCM.

$$curv^{x+}_{i,3} = 2 \cdot dxt_i * dxt_{i+1}/((dxt_{i-1} + 2 \cdot dxt_i + dxt_{i+1}) \cdot (dxt_{i-1} + dxt_i)) \quad (15.131)$$
$$curv^{x-}_{i,1} = 2 \cdot dxt_i * dxt_{i+1}/((dxt_i + 2 \cdot dxt_{i+1} + dxt_{i+2}) \cdot (dxt_{i+1} + dxt_{i+2})) \quad (15.132)$$
$$curv^{x-}_{i,2} = -2 \cdot dxt_i * dxt_{i+1}/((dxt_{i+1} + dxt_{i+2}) \cdot (dxt_i + dxt_{i+1})) \quad (15.133)$$
$$curv^{x-}_{i,3} = 2 \cdot dxt_i * dxt_{i+1}/((dxt_i + 2 \cdot dxt_{i+1} + dxt_{i+2}) \cdot (dxt_i + dxt_{i+1})) \quad (15.134)$$

In the meridional direction, twice the advective flux out of the northern face of the T cell is given by

$$v^+ = (adv\_vnt_{i,k,j} + |adv\_vnt_{i,k,j}|)/2 \quad (15.135)$$
$$v^- = (adv\_vnt_{i,k,j} - |adv\_vnt_{i,k,j}|)/2 \quad (15.136)$$
$$\begin{aligned} adv\_fn_{i,k,j} &= adv\_vnt_{i,k,j} \cdot (quick^y_{jrow,1} \cdot t_{i,k,j,n,\tau-1} + quick^y_{jrow,2} \cdot t_{i,k,j+1,n,\tau-1}) \\ &- v^+ \cdot (curv^{y+}_{jrow,1} \cdot t_{i,k,j+1,n,\tau-1} + curv^{y+}_{jrow,2} \cdot t_{i,k,j,n,\tau-1} + curv^{y+}_{jrow,3} \cdot t_{i,k,j-1,n,\tau-1} \\ &- v^- \cdot (curv^{y-}_{jrow,1} \cdot t_{i,k,j+2,n,\tau-1} + curv^{y-}_{jrow,2} \cdot t_{i,k,j+1,n,\tau-1} + curv^{y-}_{jrow,3} \cdot t_{i,k,j,n,\tau-1}) \end{aligned} \quad (15.137)$$

where the coefficients are given by

$$quick^y_{jrow,1} = 2 \cdot dyt_{jrow+1}/(dyt_{jrow+1} + dyt_{jrow}) \quad (15.138)$$
$$quick^y_{jrow,2} = 2 \cdot dyt_{jrow}/(dyt_{jrow+1} + dyt_{jrow}) \quad (15.139)$$
$$curv^{y+}_{jrow,1} = 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow-1} + 2 \cdot dyt_{jrow} + dyt_{jrow+1}) \cdot (dyt_{jrow} + dyt_{jrow+1})) \quad (15.140)$$
$$curv^{y+}_{jrow,2} = -2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow} + dyt_{jrow+1}) \cdot (dyt_{jrow-1} + dyt_{jrow})) \quad (15.141)$$
$$curv^{y+}_{jrow,3} = 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow-1} + 2 \cdot dyt_{jrow} + dyt_{jrow+1}) \cdot (dyt_{jrow-1} + dyt_{jrow})) \quad (15.142)$$
$$curv^{y-}_{jrow,1} = 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow} + 2 \cdot dyt_{jrow+1} + dyt_{jrow+2}) \cdot (dyt_{jrow+1} + dyt_{jrow+2})) \quad (15.143)$$
$$curv^{y-}_{jrow,2} = -2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow+1} + dyt_{jrow+2}) \cdot (dyt_{jrow} + dyt_{jrow+1})) \quad (15.144)$$
$$curv^{y-}_{jrow,3} = 2 \cdot dyt_{jrow} * dyt_{jrow+1}/((dyt_{jrow} + 2 \cdot dyt_{jrow+1} + dyt_{jrow+2}) \cdot (dyt_{jrow} + dyt_{jrow+1})) \quad (15.145)$$

Note that the indices in the above expressions require that option *fourth_order_memory_window* be enabled. This is automatically done when option *quicker* is enabled. Also, for ocean cells next to land cells (and the surface), the correction term on the face parallel to the land boundary is dropped thereby reducing the flux to second order. Normal flux on faces shared by land cells is set to zero. Masking (not shown) is used to enfore this.

In the vertical direction, twice the advective flux through the bottom face of the T cell is given by

$$w^+ = (adv\_vbt_{i,k,j} + |adv\_vbt_{i,k,j}|)/2 \quad (15.146)$$
$$w^- = (adv\_vbt_{i,k,j} - |adv\_vbt_{i,k,j}|)/2 \quad (15.147)$$

$$\begin{aligned} adv\_fb_{i,k,j} &= adv\_vbt_{i,k,j} \cdot (quick^z_{k,1} \cdot t_{i,k,j,n,\tau-1} + quick^z_{k,2} \cdot t_{i,k+1,j,n,\tau-1}) \\ &- w^- \cdot (curv^{z+}_{k,1} \cdot t_{i,k+1,j,n,\tau-1} + curv^{z+}_{k,2} \cdot t_{i,k,j,n,\tau-1} + curv^{z+}_{k,3} \cdot t_{i,k-1,j,n,\tau-1} \\ &- w^+ \cdot (curv^{z-}_{k,1} \cdot t_{i,k+2,j,n,\tau-1} + curv^{z-}_{k,2} \cdot t_{i,k+1,j,n,\tau-1} + curv^{z-}_{k,3} \cdot t_{i,k,j,n,\tau-1}) \end{aligned} \tag{15.148}$$

Note the way w^- and w^+ are used to account for a z axis which is positive upwards. The coefficients are given by

$$quick^z_{k,1} = 2 \cdot dzt_{k+1}/(dzt_{k+1} + dzt_k) \tag{15.149}$$
$$quick^z_{k,2} = 2 \cdot dzt_k/(dzt_{k+1} + dzt_k) \tag{15.150}$$
$$curv^{z+}_{k,1} = 2 \cdot dzt_k * dzt_{k+1}/((dzt_{k-1} + 2 \cdot dzt_k + dzt_{k+1}) \cdot (dzt_k + dzt_{k+1})) \tag{15.151}$$
$$curv^{z+}_{k,2} = -2 \cdot dzt_k * dzt_{k+1}/((dzt_k + dzt_{k+1}) \cdot (dzt_{k-1} + dzt_k)) \tag{15.152}$$
$$curv^{z+}_{k,3} = 2 \cdot dzt_k * dzt_{k+1}/((dzt_{k-1} + 2 \cdot dzt_k + dzt_{k+1}) \cdot (dzt_{k-1} + dzt_k)) \tag{15.153}$$
$$curv^{z-}_{k,1} = 2 \cdot dzt_k * dzt_{k+1}/((dzt_k + 2 \cdot dzt_{k+1} + dzt_{k+2}) \cdot (dzt_{k+1} + dzt_{k+2})) \tag{15.154}$$
$$curv^{z-}_{k,2} = -2 \cdot dzt_k * dzt_{k+1}/((dzt_{k+1} + dzt_{k+2}) \cdot (dzt_k + dzt_{k+1})) \tag{15.155}$$
$$curv^{z-}_{k,3} = 2 \cdot dzt_k * dzt_{k+1}/((dzt_k + 2 \cdot dzt_{k+1} + dzt_{k+2}) \cdot (dzt_k + dzt_{k+1})) \tag{15.156}$$

15.18.4 fct

The main disadvantage of the widely used central differences advection scheme (or other higher order schemes) is the numerical dispersion that is most noticeable near large gradients in the advected quantity. Non-physical oscillations or "ripples" (under and overshoots) and negative concentrations of positive definite quantities may occur. Addition of explicit diffusion in the coordinate directions is required to reduce or eliminate this problem. The one-dimensional advection diffusion equation

$$U\frac{\partial T}{\partial x} = A\frac{\partial}{\partial x}(\frac{\partial T}{\partial x}) \tag{15.157}$$

discretized with central differences

$$U\frac{U}{2\Delta x}(T_{i+1} - T_{i-1}) = \frac{A}{\Delta x^2}(T_{i+1} - 2T_i + T_{i-1}) \tag{15.158}$$

has solutions of the form $T_i = \xi^i$ which when put into Equation (15.158) results in a quadratic equation with roots

$$\xi = 1 \quad and \quad \xi = (2 + Pe)/(2 - Pe) \tag{15.159}$$

where $Pe = U\Delta x/A$ is the Péclèt number for the grid scale Δx. The second solution changes sign from grid point to grid point (two grid point noise) unless the grid Péclèt number is less than two. Simple estimates demonstrate that the required diffusion in a typical ocean model is rather large. For a current of 10 cm/s and a grid distance of 100 km, a diffusion coefficient of 5x10^7 cm^2/s is implied. A moderate vertical velocity of 10^{-5} cm/s and a grid distance of 100m would require a vertical diffusion coefficient of 0.25 cm^2/s. Note that in the deep ocean grid distances are usually much larger and vertical velocities can easily be one or two orders of magnitude larger. In the standard case of constant diffusion coefficients the numerical requirements in regions of strong currents determine the magnitude of the coefficients. In quiet regions this implies a diffusively dominated tracer balance that is not physically justified.

The above analysis only consideres a one dimensional advective diffusive balance and the requirements on diffusion to suppress two grid point noise can be relaxed somewhat in three dimensions. However, the problem is indeed of great practical importance as shown, among others, by Weaver and Sarachik 1990, Gerdes et al. 1991, Farrow and Stevens 1995.

The upstream scheme is an equally simple advection scheme that is free from the dispersive effects mentioned above. However, it has very different numerical errors. The main disadvantage of the only first order accurate scheme is its large amount of implicit diffusion. Here one-sided upstream differences are used to calculate the advective fluxes. The upstream discretized advective diffusive balance in one dimension is

$$\frac{U+|U|}{2\Delta tx}(T_i - T_{i-1}) + \frac{U-|U|}{2\Delta x}(T_{i+1} - T_i) = \frac{A}{\Delta x^2}(T_{i+1} - 2T_i + T_{i-1}) \tag{15.160}$$

and the solution is as given above for the central differences scheme except that the grid Péclèt number is replaced by

$$Pe' = (2U\Delta x)/(2A + |U|\Delta x) \tag{15.161}$$

that is always less than two. The truncation error of the advection scheme in multi-dimensions is

$$\sum_i \frac{\partial}{\partial_i} 0.5(|u_i|\Delta x_i - \Delta t u_i^2) + \sum_{i\neq j} 0.5\Delta t u_i u_j \frac{\partial T}{\partial x_j} \tag{15.162}$$

which can be interpreted as implicit diffusion with diffusion coefficients given by

$$A^i_{impl} = 0.5(|u_i|\Delta x_i - \Delta t u_i^2) \tag{15.163}$$

For small time steps (small compared to the maximum time step allowed by the CFL criterion) the effective diffusion is such that the grid Péclèt number is two. It should be noted that the tracer balance is thus always advectively dominated. Therefore the upstream scheme might actually be less diffusive than the central differences scheme with explicit diffusion in the larger part of the model domain.

Central differences and upstream algorithms represent, in a sense, opposite extremes, each minimizing one kind of error at the expense of another. A linear compromise between both schemes may be useful in certain cases (e.g. Fiadeiro and Veronis 1977) but will in general exhibit dispersive effects. A nonlinear compromise is the flux-corrected transport (FCT) algorithm (Boris and Book 1973; Zalesak 1979. A comparison of the central differences, upstream and FCT schemes for (oceanographic) two- and three-dimensional examples is given in Gerdes et al. 1991). Here the flux difference (anti-diffusive flux) between a central difference scheme (or any other higher order scheme) and an upstream scheme is computed. Adding the anti-diffusive flux in full to the upstream flux would maximally reduce diffusion but introduce dispersive ripples. The central idea is to limit the anti-diffusive flux locally such that no under and overshoots are introduced.

One possible criterion is to insist that from one time step to the next no new maxima or minima around one grid cell are created by advection. As remarked by Rood (1987), the FCT is more a philosophy rather than an explicit algorithm, as the crucial limiting step is essentially left to the user's discretion. Depending on the choice of the limiting step the results will be closer to those of either the upstream or the central differences scheme. The amount of implicit mixing does, therefore, depend on a subjective choice. With this limitation in mind, the FCT algorithm may be regarded as a way to find the minimum mixing that is consistent with the thermodynamical constraint. With the FCT scheme, the model can be run without any explicit diffusion and the author suggests running a case with all tracer diffusion coefficients set to zero to appreciate the effects of the advection scheme. This offers an opportunity to study cases where the tracer balance is advectively dominated everywhere. Furthermore, the scheme allows use of physically motivated mixing that will not be swamped by numerically necessary explicit diffusion or large implicit diffusion of the advection scheme.

The recommended options (all of which should be specified) for the FCT scheme are *fct*, *fct_dlm1*, and *fct_3d*. An alternative option to *fct_dlm1* is *fct_dlm2* which specifies the limiter as originally proposed by Zalesak (1979). Changes in tracer due to FCT, (i.e. FCT minus central differences), are written in NetCDF format to file *fct.dta.nc* when the diagnostic option *fct_netcdf* is enabled. Two additional options *tst_fct_his* and *tst_fct_los* were introduced mainly for debugging purposes. Option *tst_fct_his* suppresses the limitation of the anti-diffusive fluxes and thus results in the high-order scheme. With this option, the model should reproduce the results of the central differences scheme. Option *tst_fct_los* forces a complete limitation of the anti-diffusive fluxes thus realizing the upstream scheme. However, all intermediate steps of the algorithm are performed. So for performance reasons, it is not recommended to use *tst_fct_los* to implement an upstream scheme.

Option fct

With option *fct* enabled, the advective fluxes are calculated in subroutine *fctflx*. The implementation closely follows the FCT algorithm as given by Zalesak (1979). The low-order (upstream) fluxes are calculated first and a preliminary upstream solution is given by

$$t^{low}_{i,k,j,n} = t_{i,k,j,n,\tau-1} - 2\Delta t(ADV\_Tx_{i,k,j} + ADV\_Ty_{i,k,j} + ADV\_Tz_{i,k,j}) \tag{15.164}$$

where the advective operator is defined by Equations (11.61), (11.62), (11.63) except that the fluxes are given by

$$adv\_fe^{ups}_{i,k,j} = adv\_vet_{i,k,j}(t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}) + |adv\_vet_{i,k,j}|(t_{i,k,j,n,\tau} - t_{i+1,k,j,n,\tau}) \tag{15.165}$$

$$adv\_fn^{ups}_{i,k,j} = adv\_vnt_{i,k,j}(t_{i,k,j,n,\tau} + t_{i,k,j+1,n,\tau}) + |adv\_vnt_{i,k,j}|(t_{i,k,j,n,\tau} - t_{i,k,j+1,n,\tau}) \tag{15.166}$$

$$adv\_fb^{ups}_{i,k,j} = adv\_vbt_{i,k,j}(t_{i,k,j,n,\tau} + t_{i,k+1,j,n,\tau}) + |adv\_vbt_{i,k,j}|(t_{i,k+1,j,n,\tau} - t_{i,k,j,n,\tau}) \tag{15.167}$$

A forward time step over $2\Delta t$ is used because the usual forward time step turned out to be unstable in the ocean model. For stability reasons, discretization by Equation (15.164) is not allowed with option *damp_inertial_oscillation* which treats the Coriolis term implicitly.

Options fct_dlm1 and fct_dlm2

The next step involves limitation of anti-diffusive fluxes using options *fct_dlm1* or *fct_dlm2*. The anti-diffusive fluxes are limited for each coordinate direction separately. Flux limitation in three dimensions is (optionally) done afterwards. This procedure was recommended by Zalesak (1979) in the case that a tracer is transported in a direction perpendicular to a large gradient in the tracer. In ocean models, the possible range of the solution is frequently given by a large variation in the vertical direction while the largest anti-diffusive fluxes occur in the horizontal direction. Experience shows that using only three-dimensional limiting results in very noisy fields although the solution is free from overshoots and undershoots[15].

As an example, the following presents details of the algorithm for the one-dimensional limiter in the x-direction. The procedure is the same for the other coordinate directions. Assume that the solution is required to stay within bounds given by Tr_i^{max} and Tr_i^{min}. There are currently two different ways to calculate these bounds and are selected by options *fct_dlm1* and *fct_dlm2*. For option *fct_dlm1* these bounds are specified as

$$\begin{aligned} Tr_i^{max} &= max(\frac{t_{i-1,k,j,n,\tau} + t_{i,k,j,n,\tau}}{2}, \frac{t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}}{2}, t^{low}_{i,k,j,n}) \\ Tr_i^{min} &= min(\frac{t_{i-1,k,j,n,\tau} + t_{i,k,j,n,\tau}}{2}, \frac{t_{i,k,j,n,\tau} + t_{i+1,k,j,n,\tau}}{2}, t^{low}_{i,k,j,n}) \end{aligned} \quad (15.168)$$

while option *fct_dlm2* employs

$$\begin{aligned} Tr_i^{max} &= max(t^{low}_{i-1,k,j,n}, t^{low}_{i,k,j,n}, t^{low}_{i+1,k,j,n}) \\ Tr_i^{min} &= min(t^{low}_{i-1,k,j,n}, t^{low}_{i,k,j,n}, t^{low}_{i+1,k,j,n}) \end{aligned} \quad (15.169)$$

which is the original formula of Zalesak (1979). The upstream solution at all neighbouring points enters the version given by Equation (15.169) which requires additional storage for the meridional direction. With the version given by Equation (15.168), the current values of the tracers at neighbouring points are used instead of the upstream solution that enters only at the central point. Experimentally, the author has found that differences in solutions using Equations (15.168) and (15.169) are very small and thus option *fct_dml1* is recommended in general. Obviously, Equations (15.168) and (15.169) are not the only possible choices for upper and lower bounds on the solution. Narrower bounds will make the solution more diffusive. Specification of Tr_i^{max} and Tr_i^{min} can be used to keep the solution within a certain range (always positive for example).

To calculate the limiters, the possible change of the solution in either direction is determined by considering the sum of anti-diffusive fluxes into and out of the grid cell. For the x-direction:

$$P^+_{i,k,j} = max(0, A\_e_{i-1,k,j}) - min(0, A\_e_{i,k,j}) \quad (15.170)$$

$$P^-_{i,k,j} = max(0, A\_e_{i,k,j}) - min(0, A\_e_{i-1,k,j}) \quad (15.171)$$

where

$$A\_e_{i,k,j} = 2\Delta t \frac{anti\_fe_{i,k,j}}{2\cos\phi^T_{jrow} dxt_i} \quad (15.172)$$

and

$$anti\_fe_{i,k,j} = adv\_fe_{i,k,j} - adv\_fe^{ups}_{i,k,j} \quad (15.173)$$

is the anti-diffusive flux at the eastern edge of the tracer cell. The maximum permitted positive or negative changes in the solution due to the divergence of the delimited anti-diffusive fluxes are

[15] Experimentation with the limitation process can be useful: The combination of two-dimensional limiting in the horizontal and one-dimensional limiting in the vertical is likely to generate less implicit diffusion than the implemented scheme.

$$\begin{aligned} Q^+_{i,k,j} &= Tr^{max}_i - t^{low}_{i,k,j} \\ Q^-_{i,k,j} &= t^{low}_{i,k,j} - Tr^{min}_i \end{aligned} \tag{15.174}$$

so that with the ratios[16]

$$\begin{aligned} R^+_{i,k,j} &= min(1, \frac{Q^+_{i,k,j}}{P^+_{i,k,j} + \epsilon}) \\ R^-_{i,k,j} &= min(1, \frac{Q^-_{i,k,j}}{P^-_{i,k,j} + \epsilon}) \end{aligned} \tag{15.175}$$

the limiters can be defined as

$$\begin{aligned} C\_e_{i,k,j} &= min(R^+_{i+1,k,j}, R^-_{i,k,j}) \quad for \quad A\_e_{i,k,j} \geq 0 \\ C\_e_{i,k,j} &= min(R^+_{i,k,j}, R^-_{i+1,k,j}) \quad for \quad A\_e_{i,k,j} < 0 \end{aligned} \tag{15.176}$$

Option fct_3d

Limitation of the anti-diffusive fluxes in the coordinate directions separately does not guarantee that the solution stays in the permitted range. To assure that no undershoots and overshoots appear a three-dimensional limitation of the anti-diffusive fluxes must be performed. This is accomplished with option *fct_3d*. This option is, however, not automatically enabled because the one-dimensional limitation is sufficient in many cases and has slightly less implicit diffusion than the full scheme.

The one-dimensional scheme shown above easily generalizes to multiple dimensions. For instance, the possible increase in the solution by anti-diffusive fluxes into the grid cell becomes

$$\begin{aligned} P^+_{i,k,j} &= max(0, A\_e_{i-1,k,j}) - min(0, A\_e_{i,k,j}) \\ &+ max(0, A\_b_{i,k,j}) - min(0, A\_b_{i,k-1,j}) \\ &+ max(0, A\_n_{i,k,j-1}) - min(0, A\_n_{i,k,j}) \end{aligned} \tag{15.177}$$

where the "A"s are defined analogously to Equation (15.172). For option *fct_dlm2*, the upper bound for the solution is

$$Tr^{max}_{i,k,j} = max(t^{low}_{i-1,k,j,n}, t^{low}_{i+1,k,j,n}, t^{low}_{i,k-1,j,n}, t^{low}_{i,k+1,j,n}, t^{low}_{i,k,j-1,n}, t^{low}_{i,k,j+1,n}, t^{low}_{i,k,j,n}) \tag{15.178}$$

The additional computational load due to the three-dimensional limiter is moderate because most of the needed maxima and minima have already been computed during the calculation of the one-dimensional limiters. Total advective fluxes are given by

$$\begin{aligned} adv\_fe_{i,k,j} &= adv\_fe^{ups}_{i,k,j} + C\_e_{i,k,j} \cdot anti\_fe_{i,k,j} \\ adv\_fn_{i,k,j} &= adv\_fn^{ups}_{i,k,j} + C\_n_{i,k,j} \cdot anti\_fn_{i,k,j} \\ adv\_fb_{i,k,j} &= adv\_fb^{ups}_{i,k,j} + C\_b_{i,k,j} \cdot anti\_fb_{i,k,j} \end{aligned} \tag{15.179}$$

[16] Where ϵ is a small value $O(10^{-25})$ to avoid division by zero.

In comparing the program code with this description, it will noticed that most quantities are computed for row *j+1* instead of row *j*. The anti-diffusive flux $anti\_fn_{j+1}$ is needed to compute R_{j+1} which enters Equation (15.176) for the meridional direction. If option *fct_dlm2* is used, the low-order solution is needed in Equation (15.169). For economic reasons, zonal and vertical fluxes for row *j+1* are already calculated and delimited at row *j*. The meridional flux $anti\_fn_{j+1}$ is calculated but not yet limited at row j while $anti\_fn_j$ has already been calculated the row before and is now delimited at row *j*.

The scheme is expensive in terms of computer time. The time spent in subroutine tracer increases by a factor 2.5. In a typical coarse resolution model with potential temperature and salinity subroutine tracer may require 40% of the total time, in which case the CPU time of the ocean model with FCT will be 1.6 times larger than without.

Section 15.18.4 contributed by
Ruediger Gerdes
$rgerdes@AWI - Bremerhaven.DE$

15.19 Miscellaneous

This section contains various options which haven't been placed into other catagories.

15.19.1 knudsen

Option *knudsen* computes density coefficients according to the Knudsen formulation. If this option is not enabled, then density coefficients are computed according to the UNESCO formulation. Refer to Section 6.2.2 for details.

15.19.2 pressure_gradient_average

Option *pressure_gradient_average* implements the pressure gradient averaging technique of Brown and Campana (1978) which can allow the time step to be increased by up to a factor of two in certain circumstances! This applies when the time step is limited by internal gravity waves. The actual time step, which should always be determined empirically, will typically be somewhat less than the theoretical factor of two limit. It should be noted that this is not a damping scheme. The amplification factor $|\lambda|$ in the stability analysis given by Brown and Campana (1978) is unity within the region of stable solutions.

Basically, the way it works is that instead of using ρ^τ in the hydrostatic pressure gradient, a semi-implicit density given by

$$\tilde{\rho} = alpha'(\rho^{tau+1} + \rho^{tau-1}) + (1 - 2\alpha')\rho^\tau \tag{15.180}$$

is used with $alpha' = 1/4$. Brown and Campana (1978) also discuss three computational modes which are introduced by this technique. They are handled by either reducing $alpha'$ slightly or applying additional time averaging to other prognostic variables. Both methods sharply reduce the maximum allowable time step. In MOM 2, the Euler backward mixing time step damps the computational modes. For a discussion on when a semi-implicit pressure gradient is applicable, refer to Killworth, Smith, and Gill (1984). Their analysis indicates that a semi-implicit pressure gradient is applicable for coarse and medium resolution ($\geq$ 1 deg) studies but may not be applicable for high resolution ($\leq$ 30 km) ones.

In order to apply this scheme for one or more rows within the memory window, there must be one additional row of tracers calculated before the internal modes of velocity can be calculated. This is because the pressure gradient, which is defined on U cell latitude rows, requires an average of four surrounding densities which are defined on adjacent T cell latitude rows. Although, strictly not a fourth order option, this extra computed tracer row requires that option *fourth_order_window* must also be enabled. This is automatically done when option *pressure_gradient_average* is enabled.

For the minimum fourth order window configuration with jmw=4, tracers are computed for rows 2 and 3 while velocities are only computed on row 2. To accommodate this, starting and ending rows for tracer calculations are given by

$$jstrac = 2 \tag{15.181}$$
$$jetrac = min(jsmw + ncrows, jmt - 1 + joff) \tag{15.182}$$

where function "*min*" limits $jetrac$ to memory window rows corresponding to latitude rows less than jmt. Refer to the formal treatment of dataflow in the memory window given in Section 3.3.2 where the starting and ending rows are given. The situation is further complicated when this option is used in conjunction with a fourth order option such as option *biharmonic*. The minimum configuration for the memory window is then jmw=5. This time, tracers are again computed for rows 2 and 3, while velocities are computed for row 2 but an extra fifth row is needed for the biharmonic computation for tracers on row 3. Yes, it works!

15.19.3 fourth_order_memory_window

The memory window typically has a minimum size of three latitude rows (jmw=3) which is appropriate for second order accurate numerics. Some options use fourth order numerics or, in some cases, averaging operators which require information from two cells away. All fourth order schemes require option *fourth_order_memory_window* which is automatically enabled in file *size.h* when any of the existing fourth order schemes are enabled. These schemes require the minimum size of the memory window to be four latitude rows (jwm=4). Some combination of schemes require the minimum memory window to be of size $jmw = 5$. However, more is required than simply opening up the window to jmw=4 rows. In the minimum size fourth order window, prognostic equations are solved only on row 2. In a second order window with $jmw = 4$, they are solved for rows 2 and 3. For a further description of how this works, refer to Section 3.3.3.

Calculations always proceed up to latitude row $jrow = jmt - 1$ even with higher order schemes!. There are no out of bounds references because meridional indexing is limited to a maximum at latitude $jrow = jmt$ and a maximum corresponding memory window row given by $j = min(j + joff, jmt) - joff$. To accommodate higher order schemes when a fully open memory window $jmw = jmt$ is used, meridional fluxes are set to zero at latitude $jrow = jmt$ which allows calculations to proceed through latitude row $jrow = jmt - 1$.

15.19.4 implicitvmix

Option *implicitvmix* allows the vertical diffusion of momentum and tracers to be solved implicitly under control of an implicit vertical diffusion factor *aidif* which is input through namelist. Refer to Section 5.4 for information on namelist variables. Setting *aidif = 1.0* gives full implicit treatment and setting *aidif = 0* gives full explicit treatment.

Pages 42 and 43 of Numerical Recipes (1992) recommend a setting of *aidif = 0.5* which is the Crank-Nicholson scheme followed by a few fully implicit steps using *aidif = 1.0*. The reason, according to Anand Gnanadesikan, is because when $R = K_v \delta t / \Delta z^2 > 1$, the explicit treatment of diffusion produces wiggles in the solution, which are then smoothed out by the implicit treatment. Essentially, using the fully implicit treatment gives more smoothing than may be realistic for $R > 1$. However, using a few fully implicit steps at the end of the integration allows for proper averaging in cases where $R >> 1$. In MOM however, the interval of interest is each timestep and without a fully implicit treatement, the wiggles produced by the explicit treatment are left. In order for the most accurate solution of the diffusion term it is necessary to set $aidif = 1.0$ for all cases where $R > 1$.

Option *implicitvmix* solves both the vertical diffusion of tracers and vertical diffusion of horizontal velocity components implicitly. Otherwise, the vertical diffusion of horizontal velocity components is always solved explicitly. In the case where option *implicitvmix* is <u>not</u> enabled, vertical diffusion of tracers may be solved implicitly or explicitly depending on whether option *isopycmix* is enabled or not. When option *isopycmix* is enabled, the vertical diffusion of tracers is always solved implicitly.

Explicit convection of tracers as described in Section 15.13 is activated only when option *implicitvmix* is <u>not</u> enabled or option *isopycmix* <u>is</u> enabled.

Assume that the one dimensional vertical diffusion equation to be solved implicitly is given by

$$\xi_k = \xi_k^\star + aidif \cdot 2\Delta\tau\delta_z(dcb_k \cdot \delta_z(\xi_k)) \tag{15.183}$$

for levels $k = 1$ to $k = km$ where ξ_k is the vertical profile of a tracer or a horizontal velocity component at time level $\tau + 1$. The given quantities are dcb_k which is the diffusion coefficient at the bottom of T or U cells, $\Delta\tau$ which is the time step, $aidif$ which is the implicit factor, and $\xi_k^\star$. If ξ is a tracer , $\xi_k^\star$ is known from the solution of Equation (11.74). If ξ is a horizontal component of velocity, $\xi_k^\star$ is the solution from Equation (11.114).

Equtation (15.183) is solved subject to flux boundary conditions at the top of the first cell at $k = 1$ (given as $sflux$) and base of the bottom cell at $k = kz$ (given as $bflux$). The solution follows from pages 42 and 43 of Numerical Recipes (1992). Note that when ξ is a horizontal component of velocity, $sflux = smf$ and $bflux = bmf$. When ξ is a tracer, $sflux = stf$ and $bflux = btf$. Equation (15.183) can be written as

$$\xi_k = \xi_k^\star + aidif \cdot 2\Delta\tau(dcb_{k-1}\frac{\xi_{k-1} - \xi_k}{dzt_k \cdot dzw_{k-1}} - dcb_k\frac{\xi_k - \xi_{k+1}}{dzt_k \cdot dzw_k}) \tag{15.184}$$

which can be re-arranged to

$$A_k \cdot \xi_{k-1} + B_k \cdot \xi_k + C_k \cdot \xi_{k+1} = \xi_k^\star \tag{15.185}$$

where

$$A_k = -\frac{aidif \cdot 2\Delta\tau \cdot dcb_{k-1}}{dzt_k \cdot dzw_{k-1}} \tag{15.186}$$

$$C_k = -\frac{aidif \cdot 2\Delta\tau \cdot dcb_k}{dzt_k \cdot dzw_k} \tag{15.187}$$

$$B_k = 1 - A_k - C_k \tag{15.188}$$

At $k = 1$, $A_k = 0$ and at $k = kz$, $C_k = 0$. Note that the last ocean level kz may be less than bottom level km to accommodate bottom topography. The boundary conditions at the top $k = 1$ and bottom $k = kz$ are imposed by setting

$$F_{k=1} = \xi^{\star}_{k=1} + \frac{sflux \cdot aidif \cdot 2\Delta\tau}{dzt_{k=1} \cdot dzw_{k=0}} \tag{15.189}$$

$$F_k = \xi^{\star}_k \tag{15.190}$$

$$F_{k=kz} = \xi^{\star}_{k=kz} - \frac{bflux \cdot aidif \cdot 2\Delta\tau}{dzt_{k=kz} \cdot dzw_{k=kz}} \tag{15.191}$$

The solution is arrived at by performing a decomposition and forward substitution using

$bet = B_{k=1}$
$\xi_{k=1} = F_{k=1}/bet$
`do k=2,kz`
    $E_k = C_{k-1}/bet$
    $bet = B_k - A_k \cdot E_k$
    $\xi_k = (F_k - A_k \cdot \xi_{k-1})/bet$
`enddo`

then a back substitution using

`do k=kz-1,1,-1`
    $\xi_k = \xi_k - E_{k+1} \cdot \xi_{k+1}$
`enddo`

15.19.5 beta_plane

Normally, the equations in MOM 2 are formulated in spherical coordinates. This option turns the model into a beta plane: $f = f_\circ + \beta \cdot y$ where $\beta = \partial f/\partial_y$ and $f_\circ$ is taken at the latitude given by $\phi^U_{jrow=1}$.

15.19.6 f_plane

Normally, the equations in MOM 2 are formulated in spherical coordinates. This option turns the model into a f plane as in option *beta_plane* with $\beta = 0$. Choosing $f_\circ$ as the equator sets a flat space cartesian grid.

15.19.7 source_term

This option allows adding source terms to the momentum and tracer equations as indicated in Sections 11.11.4 and 11.10.4.

15.19.8 readrmsk

This option allows importing region masks $mskhr_{i,jrow}$and $mskvr_k$into MOM 2 for use with certain diagnostics.

15.19.9 show_details

When enabled, this option allows details from various parts of the setup calculations to be printed to file *stdout*. When MOM 2 executed, the printout indicates where this option will give more information if enabled. When enabled, it leads to <u>lots</u> of printout and is left disabled unless the missing details are needed.

15.19.10 timing

This option allows any Fortran source code or portions of the code to be timed using a simple set of timing routines. Executing script *run_timer* will exercise these routines by solving a tracer equation in various ways. This is useful when trying to optimize speed for a particular computer platform. Refer to Section 6.2.8 for details.

15.19.11 equivalence_mw

This option hides the nine two dimension fields of the coefficient matrix for inverting the external mode elliptic equation over the memory window space. This can be done since the external mode and internal mode are essentially orthogonal calculations and the space required by one can be used by the other.

Chapter 16

External Mode Options

There are three ways to solve for external mode (depth independent) velocities and they are described in the following sections.

16.1 stream_function

Option *stream_function* enables the time honored standard approach which basically eliminates the unknown surface pressure from the momentum equations by vertically integrating and taking the curl. Boundary conditions are Dirichlet which necessitate solving island equations as given in Section 16.1.4. Changes[1] to the coefficient matrices and in the handling of islands in the elliptic solvers have resulted in faster convergence rates than in MOM 1. The external mode velocities given by the stream function approach are guaranteed to be divergence free even if the solution for change in stream function $\Delta\psi$ is not accurate . The accuracy of the solution for $\Delta\psi$ is governed by "tolrsf" which is input through namelist. Typically, the value is 10^8 cm^3/sec.

<u>Remarks</u>

Use of polar filtering on the forcing term of the elliptic equation for the implicit free surface method has been shown to lead to problems. Removing the filtering eliminates the problem. Speculation is that the polar filering would not be needed on the forcing term of the elliptic equation for the stream function method either. However, this has not yet been investigated.

16.1.1 The equation

The stream function equation is generated by taking the curl of the vertically averaged momentum equations to knock out the unknown surface pressure terms. This is done by vertically averaging the Equations (2.1) and (2.2), expressing the averaged velocities in terms of a stream function ψ, and taking the $\hat{\mathbf{k}}\cdot\nabla\times$ of these equations yielding:

$$\nabla\cdot(\frac{1}{H}\cdot\nabla\psi_t) - J(acor\cdot\frac{f}{H},\psi) = \hat{\mathbf{k}}\cdot\nabla\times F \tag{16.1}$$

where J is the Jacobian[2], $acor$ is the implicit Coriolis factor[3], the Coriolis term $f = 2\Omega\sin\phi$, and F is the vertically averaged forcing computed in subroutine *clinic*. The boundary condition

[1] Worked out by Charles Goldberg.

[2] The Jacobian is given as $J(A,B) = \frac{1}{a^2\cos\phi}(\frac{\partial A}{\partial\lambda}\frac{\partial B}{\partial\phi} - \frac{\partial B}{\partial\lambda}\frac{\partial A}{\partial\phi})$.

[3] $acor$ = zero implies that the Coriolis term is handled explicitly. Otherwise, $0.5 \leq acor < 1.0$ implies implicit handling of the Coriolis term. This is useful for coarse models with global domains where the time step is limited by the inertial period $1/f$.

is that the normal component of the gradient of the stream function $\hat{n} \cdot \nabla\psi = 0$ on lateral boundaries. Actually, since the viscous terms in the vertically averaged forcing are of the form $\nabla^2 u$ and $\nabla^2 v$, another boundary condition is necessary. The additional boundary condition is that the tangential component of the stream function $\hat{t} \cdot \nabla\psi = 0$ on lateral boundaries.

Bryan(1969) gives the discretization of Equation (16.1) in terms of five point numerics. This means that the discretized equation at grid point with subscripts $(i, jrow)$ involves the four nearest neighboring points with subscripts $(i+1, jrow)$,$(i-1, jrow)$, $(i, jrow+1)$, and $(i, jrow-1)$. Semtner (1974) derives the nine point equivalent of Bryan's external mode equation which additionally involves the four nearest neighboring corner points $(i+1, jrow+1)$,$(i-1, jrow+1)$, $(i+1, jrow-1)$, and $(i-1, jrow-1)$. A similar approach[4] is given below.

The finite difference counterpart of Equation (16.1) is arrived at by starting with the vertically averaged finite differenced momentum equations

$$\frac{\bar{u}_{i,k,j,1,\tau+1} - \bar{u}_{i,k,j,1,\tau-1}}{2\Delta\tau} - \tilde{f}_{jrow} \cdot (\bar{u}_{i,k,j,2,\tau+1} - \bar{u}_{i,k,j,2,\tau-1}) = \frac{-1}{\rho_\circ \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{p^s_{i,jrow}}^{\phi}) + zu_{i,jrow,1} \tag{16.2}$$

$$\frac{\bar{u}_{i,k,j,2,\tau+1} - \bar{u}_{i,k,j,2,\tau-1}}{2\Delta\tau} + \tilde{f}_{jrow} \cdot (\bar{u}_{i,k,j,1,\tau+1} - \bar{u}_{i,k,j,1,\tau-1}) = \frac{-1}{\rho_\circ} \cdot \delta_\phi(\overline{p^s_{i,jrow}}^{\lambda}) + zu_{i,jrow,2} \tag{16.3}$$

where p^s is the unknown surface pressure and $zu_{i,jrow,n}$ contains the vertically averaged advection, diffusion, hydrostatic pressure gradients, and explicit part of the Coriolis term. The implicit part of the Coriolis term is given by

$$\tilde{f}_{jrow} = acor \cdot 2\Omega \sin\phi^U_{jrow} \tag{16.4}$$

When the finite difference curl is taken, particular attention must be taken to assure that the unknown surface pressure terms are eliminated even when the grid is non-uniform. Here is an outline of the steps needed to do this. Starting with Equations (16.2) and (16.3), make the following substitutions:

$$\bar{u}_{i,k,j,1,\tau-1} = -\frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau-1}}^{\lambda}) \tag{16.5}$$

$$\bar{u}_{i,k,j,2,\tau-1} = \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau-1}}^{\phi}) \tag{16.6}$$

$$\bar{u}_{i,k,j,1,\tau+1} = -\frac{1}{H_{i,jrow}} \cdot \delta_\phi(\overline{\psi_{i,jrow,\tau+1}}^{\lambda}) \tag{16.7}$$

$$\bar{u}_{i,k,j,2,\tau+1} = \frac{1}{H_{i,jrow} \cdot \cos\phi^U_{jrow}} \cdot \delta_\lambda(\overline{\psi_{i,jrow,\tau+1}}^{\phi}) \tag{16.8}$$

$$\Delta\psi_{i,jrow} = \psi_{i,jrow,\tau+1} - \psi_{i,jrow,\tau-1} \tag{16.9}$$

and multiply the equations as follows:

[4]This approach was first worked out by Charles Goldberg (personal communication) using algebraic manipulations. The derivation given here is in terms of finite difference operators.

$$dxu_i \cdot \cos\phi^U_{jrow} \cdot (Eqn\, 16.2\,) \tag{16.10}$$
$$dyu_{jrow} \cdot (Eqn\, 16.3\,) \tag{16.11}$$

Then take the finite difference equivalent of the curl operation using:

$$-\, dyt_{jrow} \cdot \delta_\phi(\overline{Eqn16.10}^{\lambda}) + dxt_i \cdot \delta_\lambda(\overline{Eqn16.11}^{\phi}) \tag{16.12}$$

After completing the above operations, the resulting nine point equivalent of Bryan's external mode equation is obtained. The finite difference counterpart of Equation (16.1) for non-uniform grids is given as

$$\begin{aligned}
&\frac{1}{2\Delta\tau}\cdot \Bigg[dyt_{jrow}\cdot\delta_\phi\left(\overline{(\frac{dxu_{i-1}\cdot\cos\phi^U_{jrow-1}}{H_{i-1,jrow-1}\cdot dyu_{jrow-1}})\cdot dyu_{jrow-1}\cdot\delta_\phi(\overline{\Delta\psi_{i-1,jrow-1}}^{\lambda})}^{\lambda}\right)\\
&\qquad + dxt_i\cdot\delta_\lambda\left(\overline{(\frac{dyu_{jrow-1}}{H_{i-1,jrow-1}\cdot\cos\phi^U_{jrow-1}\cdot dxu_{i-1}})\cdot dxu_{i-1}\cdot\delta_\lambda(\overline{\Delta\psi_{i-1,jrow-1}}^{\phi})}^{\phi}\right)\Bigg]\\
&\quad + \Bigg[dyt_{jrow}\cdot\delta_\phi\left(\overline{\frac{\tilde{f}_{jrow-1}}{H_{i-1,jrow-1}}\cdot dxu_{i-1}\cdot\delta_\lambda(\overline{\Delta\psi_{i-1,jrow-1}}^{\phi})}^{\lambda}\right)\\
&\qquad - dxt_i\cdot\delta_\lambda\left(\overline{\frac{\tilde{f}_{jrow-1}}{H_{i-1,jrow-1}}\cdot dyu_{jrow-1}\cdot\delta_\phi(\overline{\Delta\psi_{i-1,jrow-1}}^{\lambda})}^{\phi}\right)\Bigg]\\
&\quad = \Big[-dyt_{jrow}\cdot\delta_\phi(\overline{dxu_{i-1}\cdot\cos\phi^U_{jrow-1}\cdot zu_{i-1,jrow-1,1}}^{\lambda})\\
&\qquad + dxt_i\cdot\delta_\lambda(\overline{dyu_{jrow-1}\cdot zu_{i-1,jrow-1,2}}^{\phi})\Big]
\end{aligned} \tag{16.13}$$

where $\tilde{f}_{jrow}$ is given by Equation (16.4). Comparing this to Equation (16.1), $\nabla\cdot(\frac{1}{H}\cdot\nabla\psi_t)$ corresponds to the first bracket, $-J(acor\cdot\frac{f}{H},\psi)$ corresponds to the second bracket and $\hat{\mathbf{k}}\cdot\nabla\times F$ corresponds to the third bracket which is

$$\begin{aligned}
ztd_{i,jrow} &= -dyt_{jrow}\cdot\delta_\phi(\overline{dxu_{i-1}\cdot\cos\phi^U_{jrow-1}\cdot zu_{i-1,jrow-1,1}}^{\lambda})\\
&+\; dxt_i\cdot\delta_\lambda(\overline{dyu_{jrow-1}\cdot zu_{i-1,jrow-1,2}}^{\phi})
\end{aligned} \tag{16.14}$$

If the time step constraint imposed by convergence of meridians needs to be relaxed, $ztd_{i,jrow}$ is filtered in longitude by one of two techniques: Fourier filtering (Bryan, Manabe, Pacanowski 1975) enabled by option *fourfil* or finite impulse response filtering enabled by option *firfil.* Both should be used with caution[5] and only when necessary at high latitudes. *firfil* is much faster than *fourfil.*
The boundary condition is no slip on lateral boundaries. This is expressed as

$$\delta_\lambda(\overline{\psi_{i,jrow}}^{\phi}) = 0 \tag{16.15}$$
$$\delta_\phi(\overline{\psi_{i,jrow}}^{\lambda}) = 0 \tag{16.16}$$

[5]This filtering induces spurious vertical velocities.

which implies that $\psi_{i,jrow} = constant$ in space on boundaries. In domains containing disconnected land masses (islands), if there is circulation around an island, then the value of $\psi_{i,jrow}$ on the island is a different constant than the value of $\psi_{i,jrow}$ on the continent. Refer to Appendix 16.1.4 for details relating to solving for $\psi_{i,jrow}$ on islands.

16.1.2 The coefficient matrices

Equation (16.13) involves nine values of $\Delta\psi$ centered at $\Delta\psi_{i,jrow}$ which may be written as

$$\sum_{i'=-1}^{1}\sum_{j'=-1}^{1} \mathit{coeff}_{i,jrow,i',j'} \Delta\psi_{i+i',jrow+j'} = ztd_{i,jrow} \tag{16.17}$$

where the 3rd and 4th subscripts on the coefficient matrix refer to coefficients on neighboring cells. For example, $i' = -1$ and $j' = 0$ refers to the coefficient of $\Delta\psi_{i-1,jrow}$ (the value on the western neighboring cell). The coefficient of $\Delta\psi_{i+1,jrow+1}$ on the northeast neighboring cell is given by $i' = 1$ and $j' = 1$. When option *sf_9_point* is enabled, MOM 2 calculates the coefficient matrix $\mathit{coeff}_{i,jrow,i',j'}$ for Equation (16.17) using summation formulas given in Section 17.4. This nine point coefficient matrix differs slightly from the one in MOM 1 and is more accurate. The elliptic solvers also converge in fewer iterations using this coefficient matrix. The $\bar{u}_{i,k,j,1,\tau+1}$ and $\bar{u}_{i,k,j,2,\tau+1}$ derived from the solution of Equation (16.13) are exact solutions[6] of the finite difference vertically averaged momentum Equations (16.2) and (16.3).

When option *sf_5_point* is enabled, the nine point coefficient matrix is approximated by a five point coefficient matrix involving five non-zero coefficients $\mathit{coeff}_{i,jrow,i',j'}$ for $i' = 0$ or $j' = 0$ as in Bryan (1969). It is arrived at by averaging terms used to construct the nine point coefficient matrix in a different way resulting in a coefficient matrix that differs from the one used by Bryan (1969). The five point coefficient matrix is not as accurate as the nine point matrix although the nine point matrix has a checkerboard null mode. Refer to Appendix C for a discussion on null modes. However, in the stream function, this null mode is largely suppressed because $\psi_{i,jrow}$ is constant along boundaries. The five point matrix does not have this checkerboard null mode. However, both five and nine point operators have a constant null mode. Arbitrarily, one land mass is chosen and all stream function values are referenced to this land mass value to eliminate the constant null mode. Either the five point or nine point operator must be chosen by enabling options *sf_5_point* or *sf_9_point*.

16.1.3 Solving the equation

After choosing whether five point numerics enabled by option *sf_5_point* or nine point numerics enabled by option *sf_9_point* is to be used with Equation (16.13), the elliptic equation is inverted by one of three methods: the preferred method is enabled by option *conjugate_gradient*; a sequential relaxation method is enabled by option *oldrelax*; and a more highly vectorized version of the sequential relaxation method is enabled by option *hypergrid*. The conjugate gradient solver may fail to converge for large values of the implicit Coriolis parameter *acor*, which desymmetrize the equations.

16.1.4 Island equations

When solving Equations (16.2) and (16.2) by the method of stream function, Dirichlet boundary conditions on velocity are used. Specifically, both components of vertically integrated velocity are set to zero on all land U cells. This implies

[6] To within roundoff.

$$\delta_\lambda(\overline{\psi_{i,jrow}}^{\phi}) = 0 \quad (16.18)$$

$$\delta_\phi(\overline{\psi_{i,jrow}}^{\lambda}) = 0 \quad (16.19)$$

which further implies that $\psi_{i,jrow} = constant$ on all T cells within land masses and surrounding ocean coastal perimeters. At this point it is useful to change to a directional notation letting cell $(i, jrow)$ be referred to as cell ℓ. The central coefficient $coeff_{i,jrow,0,0}$ for cell ℓ from Equation (16.17) is referred to as C_ℓ°, the coefficient $coeff_{i,jrow,1,0}$ at the eastern face of cell ℓ is referred to as C_ℓ^e, the coefficient $coeff_{i,jrow,1,1}$ at the northeastern corner of cell ℓ is referred to as C_ℓ^{ne}, and so forth. Using this notation, Equation (16.17) may be written for the ℓ'th T cell as

$$\begin{aligned} C_\ell^n \cdot \psi_\ell^n + C_\ell^e \cdot \psi_\ell^e + C_\ell^s \cdot \psi_\ell^s + C_\ell^w \cdot \psi_\ell^w + C_\ell^\circ \cdot \psi_\ell^\circ + \\ C_\ell^{ne} \cdot \psi_\ell^{ne} + C_\ell^{nw} \cdot \psi_\ell^{nw} + C_\ell^{se} \cdot \psi_\ell^{se} + C_\ell^{sw} \cdot \psi_\ell^{sw} = ztd_\ell^\circ \end{aligned} \quad (16.20)$$

where superscript n indicates the cell to the north of cell ℓ with index $(i, jrow{+}1)$, superscript ne indicates the cell to the northeast of cell ℓ with index $(i+1, jrow+1)$ and so forth. Superscript $\circ$ refers to the the ℓ'th cell with index $(i, jrow)$.
The central coefficient C_ℓ° is related to the surrounding coefficients by

$$c_\ell^\circ = -(C_\ell^n + C_\ell^s + C_\ell^e + C_\ell^w + C_\ell^{ne} + C_\ell^{nw} + C_\ell^{se} + C_\ell^{sw}) \quad (16.21)$$

An island equation is generated by summing Equation (16.20) over all cells within an island including coastal ocean perimeter cells. At each cell ℓ within the island proper, the left hand side of Equation (16.20) is zero because of Equation (16.21) and the condition that

$$\psi_\ell^\circ = \psi_\ell^n = \psi_\ell^s = \psi_\ell^e = \psi_\ell^w = \psi_\ell^{ne} = \psi_\ell^{nw} = \psi_\ell^{se} = \psi_\ell^{sw} = constant \quad (16.22)$$

After summing over all cells within the ocean perimeter and island proper, the only locations with non-zero contributions to the left hand side sum are ocean perimeter cells. This can be expressed as the sum of L times nine products where subscript ℓ runs from 1 to L which is the number of island perimeter cells.

$$\begin{aligned} \sum_{\ell=1}^{L}(C_\ell^n \cdot \psi_\ell^n + C_\ell^e \cdot \psi_\ell^e + C_\ell^s \cdot \psi_\ell^s + C_\ell^w \cdot \psi_\ell^w + C_\ell^\circ \cdot \psi^\circ + \\ C_\ell^{ne} \cdot \psi_\ell^{ne} + C_\ell^{nw} \cdot \psi_\ell^{nw} + C_\ell^{se} \cdot \psi_\ell^{se} + C_\ell^{sw} \cdot \psi_\ell^{sw}) = \sum_{\ell=1}^{L} ztd_\ell^\circ \end{aligned} \quad (16.23)$$

Without loss of generality, all coefficients adjacent to land cells on the left hand side may be set to zero which leaves only the island stream function ψ° and values of ψ exterior to the island perimeter. Indeed, this is the very reason that reciprocals of H are set to zero on land (which zeroes the coefficients there) in the code of MOM 1 and MOM 2.

Recall from Equation (16.14) that $ztd_{i,jrow}$ contains sums and differences involving $zu_{i-1,jrow-1,n}$, $zu_{i-1,jrow,n}$, $zu_{i,jrow-1,n}$, and $zu_{i,jrow,n}$. It is in fact the finite difference version of the curl of $zu_{i,jrow,n}$. The details of what is inside of $zu_{i,jrow,n}$ are unimportant. By Stokes theorem, summing this curl over any area leaves only values of $zu_{i,jrow,n}$ at the outer boundary. These

outer boundary cells are by definition exterior to the island perimeter and contain only known values of $zu_{i,jrow,n}$.
The island stream function $\psi_\circ$ can be expressed as a linear combination of all $\psi_{i,jrow}$ and $zu_{i,jrow}$ immediately outside the island perimeter.

$$\psi_\circ = (\sum_{\ell=1}^{L} ztd_\ell^\circ - \sum_{\ell=1}^{L}(C_\ell^n \cdot \psi_\ell^n + C_\ell^e \cdot \psi_\ell^e + \cdots))/\sum_{\ell=1}^{L} C_\ell^\circ \tag{16.24}$$

In the code of MOM 2, $zu_{i,jrow,n}$ is set to zero on land cells so that $\sum_{\ell=1}^{L} ztd_\ell^\circ$ picks up contributions only from values of $zu_{i,jrow,n}$ in the ocean.

The solution of Equation (16.1) is determined only to within an arbitrary additive constant (null mode). If the domain is multiply connected by two or more distinct land masses (islands), the value of the stream function can be chosen arbitrarily on one of the land masses. In MOM 1, the value on the main continent is held fixed at zero and each iteration involved calculating an integral around each other island. In MOM 2, this option is retained, but it has been determined that the solution converges more quickly if the stream function values on all land masses are allowed to "float". Afterwards, the entire solution is adjusted to make the stream function zero on the main continent. Choosing a stream function value of zero on the main continent <u>and</u> on an island amounts to an over specification of the problem and should not be done.

<u>Another approach</u>

Another approach suggested by Charles Goldberg leads to the same result. Equation (17.12) indicates that the right hand side of Equation (16.13) centered at a land or ocean perimeter cell $T_{i,jrow}$ contains values of $zu_{i,jrow,n}$ on land that are not available. In fact, $forc_{i,jrow}$ is the line integral of $zu_{i,jrow,n}$ around the boundary of cell $T_{i,jrow}$. The island equation for land mass m is formed by summing Equation (16.13) over all land and ocean perimeter T cells of land mass m.

Since each right hand side is a line integral around the boundary of one T cell, in the sum of such line integrals over all of land mass m and its surrounding ocean perimeter cells, all contributions from interior edges cancel, leaving only known values of $zu_{i,jrow,n}$ at ocean U cells.

On the left side of an island equation, Equations (17.9) and (17.10) and the fact that $\psi_{i,jrow} = \psi_m$ on all land and ocean perimeter T cells of land mass m imply that all contributions from land T cells are zero as follows. At a land T cell, all nine values of ψ are the constant value ψ_m, so by pulling all terms that don't involve i' or j' out of the i' and j' summations, Equation (17.9) reduces to

$$\begin{aligned}
&\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddyt_{i'',j''} \cdot \frac{dxu_{i+i''} \cdot \cos\phi^U_{jrow+j''}}{H_{i+i'',jrow+j''} \cdot dyu_{jrow+j''} \cdot 2\Delta\tau}\right]\Delta\psi_m \sum_{i'=0}^{1}\sum_{j'=0}^{1} cddyu_{i',j'} \\
&+ \sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddxt_{i'',j''} \cdot \frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''} \cdot dxu_{i+j''} \cdot \cos\phi^U_{jrow+j''} \cdot 2\Delta\tau}\right]\Delta\psi_m \sum_{i'=0}^{1}\sum_{j'=0}^{1} cddxu_{i',j'} \\
&= 0
\end{aligned} \tag{16.25}$$

since the sum of the partial derivative coefficients $cddxu$ and $cddyu$ are zero. Similarly, at land points, the contributions to the island equation from the implicit Coriolis terms is also zero because Equation (17.10) reduces to

$$
\begin{aligned}
&\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[-cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\right]\Delta\psi_m\sum_{i'=0}^{1}\sum_{j'=0}^{1}cddxu_{i',j'}\\
&+\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[-cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\right]\Delta\psi_m\sum_{i'=0}^{1}\sum_{j'=0}^{1}cddyu_{i',j'}\\
&=0
\end{aligned}
\tag{16.26}
$$

Because of these simplifications, the island equation for land mass m may be calculated by summing Equations (16.13) only over the ocean perimeter T cells of land mass m.

In fact, in the Fortran code of MOM 2 , the full island equation never appears. Instead, each set of contributions to the island equation from an island perimeter cell $T_{i,jrow}$ is stored in $coeff_{i,jrow,i'',j''}$. The sum over the island perimeter is done in the elliptic solver.

16.1.5 Symmetry in the stream function equation

Conjugate gradient solvers work by transforming the system of equations

$$\mathbf{Ax} = \mathbf{b}$$

into minimizing the quadratic form

$$\mathbf{Q}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

This transformation is justified as long as $\mathbf{A}$ is symmetric, that is, $\mathbf{A}_{\alpha,\beta} = \mathbf{A}_{\beta,\alpha}$ for all α and β.

Symmetry of the explicit equations

In Equation (16.13), the three major brackets are expanded into summation notation as given by Equations (17.9), (17.10), and (17.12). Interpreting these equations in the formalism given above, the "vector" $\mathbf{x}$ is $\Delta\Psi_{i,jrow}$ at all mid-ocean points and island values $\Delta\Psi_m$, the linear operator $\mathbf{A}$ is the array $coeff_{i,jrow,i'+i'',j'+j''}$, and the subscripts are $\alpha = (i, jrow)$ and $\beta = (i^\star, j^\star)$, where $i^\star = i + i' + i''$ and $j^\star = jrow + j' + j''$ for some values of $i', j' \in \{0, 1\}$ and $i'', j'' \in \{-1, 0\}$. More simply, $\beta = (i^\star, j^\star)$ is one of the eight nearest neighbors of $\alpha = (i, jrow)$ and $\alpha = (i, jrow)$ is one of the eight nearest neighbors of $\beta = (i^\star, j^\star)$. This section shows that the contributions to the coefficients arising from the first brackets, Equation (17.9), are symmetric.

If $i^\star \neq i$, then i' and i'' must both be at their upper limits or both must be at their lower limits. In either case, $i' = i''+1$. Similarly, if $j^\star \neq j$, then $j' = j''+1$. If both are unequal, that is, if $(i, jrow)$ and $(i^\star, j^\star)$ are diagonal neighbors, then $i+i'' = i+i'+i''-i' = i^\star-i' = i^\star+(-1-i'')$, and similarly $jrow+j'' = j^\star+(-1-j'')$. Note that the expressions $(i^\star)' = 1-i'$, $(j^\star)' = 1-j'$, $(i^\star)'' = -1-i''$, and $(j^\star)'' = -1-j''$ describe the transition in the opposite direction, so the relations $i+i'' = i^\star+(-1-i'')$ and $jrow+j'' = j^\star+(-1-j'')$ show that the evaluation point of most of the factors in Equation (17.9) is the same going both ways. The remaining factors, $cddyu_{i',j'}$, $cddyt_{i'',j''}$, $cddxu_{i',j'}$, and $cddxt_{i'',j''}$ all change sign when i' is replaced by $1-i'$, j' is replaced by $1-j'$, i'' is replaced by $-1-i''$, and j'' is replaced by $-1-j''$. Thus the product of any two of these is unchanged, and each diagonal coeffiecient at $(i, jrow)$ is equal to the opposite diagonal coefficient at $(i^\star, j^\star)$.

If $i = i^\star$, there are two possibilities: either $i' = i'' = 0$ or $i' = -i'' = 1$. Assume that $j \neq j^\star$. Otherwise, $(i, jrow)$ and $(i^\star, j^\star)$ are not neighbors, but identical. The transitions back to $(i, jrow)$ are in this case: $i = i^\star = i^\star + i' + i''$ and $jrow = j^\star + (1 - j') + (-1 - j'')$, so $i + i'' = i^\star + i''$ and, as above, $jrow + j'' = j^\star + (-1 - j'')$. This time, only j' changes to $1 - j'$ and j'' changes to $-1 - j''$. As a result, the factors $cddyu_{i',j'}$ and $cddyt_{i'',j''}$ change sign, but the factors $cddxu_{i',j'}$, and $cddxt_{i'',j''}$ remain unchanged. All other factors remain evaluated at the same points, so again symmetry holds in that the northern or southern coefficient in Equation (17.9) centered at $(i, jrow)$ is the same as the opposite coefficient in Equation (17.9) centered at $(i^\star, j^\star)$. The case where $j = j^\star$ is proved similarly.

Antisymmetry of the implicit Coriolis terms

If one applies the same arguments to the implicit Coriolis terms in Equation (17.10), the result is a proof of antisymmetry rather than symmetry. First, the diagonal coefficients in the implicit Coriolis terms are zero, so this case need not be considered. In the northern and southern terms (i.e., when $i = i^\star$), the evaluation points given by $(i + i'', jrow + j'') = (i^\star + i''), j^\star + (-1 - j''))$ still remain the same, but each term has one *cddx* coefficient and one *cddy* coefficient, so the product of these two factors changes sign. The antisymmetry of the eastern and western implicit Coriolis coefficients is proved similarly.

Since conjugate gradient solvers are derived under the assumption of symmetry of coefficients, the larger the implicit Coriolis terms, the less suitable a conjugate gradient solver is for the elliptic Equations (16.13). For large values of the implicit Coriolis parameter *acor*, the conjugate gradient solver will not converge.

Island equations and symmetry

If one of the T cells, $\alpha = (i, jrow)$ or $\beta = (i^\star, j^\star)$ is an island perimeter cell, the arguments in the above section on symmetry of the explicit equations must be made more carefully, since each island equation is a sum of Equations (16.13). Note that it cannot happen that both α and β are island perimeter T cells because islands must be separated by at least two ocean T cells.

Without loss of generality, we may assume that β is an island perimeter cell and the α is not. In this case, β may not be the only island perimeter cell of land mass m that is a nearest neighbor of α. The coefficient of the island perimeter value $\Delta\Psi_m$ in the elliptic Equation (16.13) centered at α is the sum of the contributions from all nearest neighbors of α that are in the island perimeter of land mass m. Each contribution will be shown equal to a corresponding contribution of $\Delta\Phi_\alpha$ to the island equation for land mass m. Individually, each coefficient[7] of the equation centered at cell α, say, the coefficient in the direction of cell β, is equal to the opposite coefficient in the contribution to the island equation arising from β. Since multiple island perimeter cells as nearest neighbors of α lead to a sum of their individual contributions, and since the island equation is the sum of the individual equations centered at island perimeter points β, both summations lead to the same result[8]. Thus even the coefficient values involving islands satisfy the symmetry relation $\mathbf{A}_{\alpha,\beta} = \mathbf{A}_{\beta,\alpha}$ if the implicit Coriolis parameter is zero.

Asymmetry of the barotropic equations in MOM 1

The barotropic equations presented to the solvers in MOM 1 were not symmetric, even when the implicit Coriolis parameter *acor* was set to zero. In an attempt to optimize the code to save a few floating point operations in the calculation of $\mathbf{Ax}$, each equation was divided by its diagonal coefficient, $coeff_{i,jrow,0,0}$. Since these diagonal coefficients depend on topography and grid factors, they are not equal at neighboring cells, and the resulting equations are not

[7] Only the first brackets are being done here. The implicit Coriolis terms stand no chance of being symmetric.

[8] Note that remote island perimeter cells neither appear in the equation centered at α, nor do they contain references to cell α in their (up to) nine terms.

symmetric. The asymmetry between a mid-ocean cell and a neighboring island perimeter is likely to be especially severe. It is possible that some of the problems arising with the use of conjugate gradient solvers in MOM 1 may be attributed to MOM 1's "normalization" of the elliptic equations and the resulting de-symmetrization of the coefficient array.

Section 16.1.5 contributed by
Charles Goldberg
chg@gfdl.gov

16.2 rigid_lid_surface_pressure

Option *rigid_lid_surface_pressure* enables the method developed by Dukowicz, Smith, and Malone (1992) which is hereafter referred to as DSM. Instead of taking the curl of the vertically integrated momentum equations to drop the surface pressure, the divergence is taken which yields an elliptic equation for the surface pressure instead of a stream function. Its main advantage is that Neumann boundary conditions apply instead of Dirichlet boundary conditions, and there is no need to solve island integrals which perform poorly on SIMD[9] parallel computers. However, elliptic equation solvers converge very slowly using this method.

16.2.1 The equations

Basically, the DSM approach is to define an auxiliary velocity $\hat{U}$ and $\hat{V}$ in terms of a time difference of surface pressure Δps and vertically averaged velocities U and V as follows

$$\hat{U}_{i,jrow} = U^{\tau+1}_{i,jrow} + 2\Delta\tau \cdot \frac{1}{\cos\phi^U_{jrow}} \delta_\lambda(\overline{\Delta ps_{i,jrow}}^{\phi}) \tag{16.27}$$
$$\hat{V}_{i,jrow} = V^{\tau+1}_{i,jrow} + 2\Delta\tau \cdot \delta_\phi(\overline{\Delta ps_{i,jrow}}^{\lambda}) \tag{16.28}$$
$$\Delta ps_{i,jrow} = ps^{\tau}_{i,jrow} - ps^{\tau-1}_{i,jrow} \tag{16.29}$$

The momentum equations can then be re-written in terms of $\hat{U}$ and $\hat{V}$ as

$$\hat{U}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} - 2\Delta\tau\omega\delta_\phi(\overline{\Delta ps_{i-1,jrow-1}}^{\lambda}) \tag{16.30}$$
$$\hat{V}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} + 2\Delta\tau\omega\delta_\lambda(\overline{\Delta ps_{i-1,jrow-1}}^{\phi}) \tag{16.31}$$
$$\tilde{U}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,1} - \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(\overline{ps^{\tau-1}_{i-1,jrow-1}}^{\phi})) \tag{16.32}$$
$$\tilde{V}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,2} - \delta_\phi(\overline{ps^{\tau-1}_{i-1,jrow-1}}^{\lambda})) \tag{16.33}$$
$$zu_{i,jrow,1} = f(gcor \cdot V^{\tau}_{i,k,j} + (1-gcor)\cdot V^{\tau-1}_{i,k,j}) + G_{i,jrow,1} \tag{16.34}$$
$$zu_{i,jrow,2} = -f(gcor \cdot U^{\tau}_{i,k,j} + (1-gcor)\cdot U^{\tau-1}_{i,k,j}) + G_{i,jrow,2} \tag{16.35}$$

where $\omega = 2\Delta\tau \cdot acor \cdot f$ and the forcing terms $G_{i,jrow,1}$ and $G_{i,jrow,2}$ contain all remaining terms which are known. If solving the Coriolis term explicitly ($acor = 0$), then $gcor = 1$ otherwise $gcor = 0$ for implicit treatment ($acor > 1/2$).

[9] Single Instruction stream–Multiple Data streams

Equations (16.30) and (16.31) can be solved if terms involving Δps are dropped. The justification given by DSM is that these terms are the same order of magnitude as the time truncation error which is O(τ^3). This is the operator splitting technique of DSM which leads to a self-adjoint elliptic equation and therefore a symmetric coefficient matrix which can be solved with efficient conjugate gradient techniques. Multiplying Equations (16.27) and (16.28) by the total depth H and taking the divergence gives the second order elliptic equation for Δps in terms of known quantities $\hat{U}$ and $\hat{V}$.

$$\begin{aligned}
&\delta_\lambda(\overline{\frac{H_{i-1,jrow-1}}{\cos\phi^U_{jrow-1}}\delta_\lambda(\overline{\Delta ps_{i-2,jrow-2}}^\phi)}^\phi) + \delta_\phi(\overline{H_{i-1,jrow-1}\cos\phi^U_{jrow-1}\delta_\phi(\overline{\Delta ps_{i-2,jrow-2}}^\lambda)}^\lambda) \\
= \; & \frac{1}{2\Delta\tau}(\delta_\lambda(\overline{H_{i-1,jrow-1}\hat{U}_{i-1,jrow-1}}^\phi) + \delta_\phi(\cos\phi^U_{jrow-1}\overline{H_{i-1,jrow-1}\hat{V}_{i-1,jrow-1}}^\lambda)) \qquad (16.36)
\end{aligned}$$

Equation (16.36) is solved using option *conjugate_gradient* with option *sf_9_point*. Note that the number of islands is set to zero ($nislsp = 0$) because no island equations are to be solved. After solving for Δps, a checkerboard null space and mean are removed after which the barotropic velocities at time level $\tau + 1$ are given by

$$U^{\tau+1}_{i,jrow} = \hat{U}_{i,jrow} - \frac{2\Delta\tau}{\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta ps_{i-1,jrow-1}}^\phi) \qquad (16.37)$$

$$V^{\tau+1}_{i,jrow} = \hat{V}_{i,jrow} - 2\Delta\tau \cdot \delta_\phi(\overline{\Delta ps_{i-1,jrow-1}}^\lambda) \qquad (16.38)$$

and the predicted surface pressure is

$$ps^{\tau+1}_{i,jrow} = ps^{\tau-1}_{i,jrow} + \Delta ps_{i,jrow} \qquad (16.39)$$

The above equations are written with a leapfrog time step in mind. During mixing time steps (forward or first pass of an Euler backward), quantities at time level $\tau - 1$ are replaced by their values at τ and $2\Delta\tau$ is replaced by $\Delta\tau$. The second pass of an Euler backward mixing time step is as described in Section 11.1 and indicated in Figure 11.2.

16.2.2 Remarks

It should be noted that Equation (16.36) only requires a Neumann boundary condition at the boundaries instead of the Direchlet boundary condition required for the elliptic equation of the stream function method. The implication is that there are no island equations to be solved and hence this method should be faster than the stream function method on massively parallel computers.

The second point to be made is that Equation (16.36) contains a factor $H_{i,jrow}$ whereas the elliptic equation for the stream function contains the factor $1/H_{i,jrow}$ which should make this method less prone to stability problems than the stream function equation when topography contains steep slopes.

The third point to be noted is that the barotropic velocities $U^{\tau+1}_{i,jrow}$ and $V^{\tau+1}_{i,jrow}$ given by this method are not non-divergent. The degree of non-divergence is related to how accurately Equation (16.36) is solved for the change in surface pressure Δps and the accuracy depends on the tolerance variable *tolrsp* which is input through namelist and typically set to $10^{-4} gram/cm/sec^2$. Refer to Section 5.4 for information on namelist variables.

Use of polar filtering on $\hat{U}$ and $\hat{V}$ leads to a problem. Removing the filtering eliminates the problem. So the filtering has been removed for this method.

16.3 implicit_free_surface

Option *implicit_free_surface* enables the method developed by Dukowicz and Smith (1994) which is hereafter referred to as DS. This work is an extension of their earlier work described in Section 16.2. The rigid lid assumption is replaced by a free surface which exerts a surface pressure at $z = 0$. As in the rigid lid surface pressure approach, an elliptic equation can be derived for the change in surface pressure Δps but the resulting equation is more diagonally dominant than the rigid lid surface pressure equation. The implication is that the DS method converges faster than the rigid lid surface pressure method. As with the rigid lid surface pressure approach, DS is well suited for SIMD parallel computers because it requires Neumann boundary conditions and thus needs no island integrals.

16.3.1 The equations

The equations are given as (E1), (E2) and (E3) in DM and will be repeated here for convienence. They are

$$
\begin{aligned}
(I + \alpha'\tau B)(\hat{u} - u^{n-1}) &= \tau(F - g\cdot G(\tilde{\gamma}\eta^n + (1-\tilde{\gamma})\eta^{n-1}) - B(\tilde{\gamma}'u^n + (1-\tilde{\gamma}')u^{n-1})) && (16.40)\\
(DHG - \frac{2}{\alpha\theta g\tau^2}I)\Delta\eta &= \frac{1}{\alpha\theta g\tau}DH(\theta\hat{u} + (1-\theta)u^{n-1} + u^n) && (16.41)\\
u^{n+1} &= \hat{u} - \alpha\tau g G\Delta\eta && (16.42)
\end{aligned}
$$

where B is the Coriolis operator, D is the divergence operator, and G is the gradient operator. The centering coefficients $(\alpha, \alpha', \gamma, \gamma', \theta)$ and the quantities $I, u, \Delta\eta, g$ and τ are as defined in DM. This set of equations is very similar to the set in Dukowicz, Smith, and Malone (1992) except there is an extra divergence term and three centering coefficients needed to damp two computational modes. Because of the similarities, the surface pressure and implicit free surface methods share much of the same code in MOM 2.

In the terminology of MOM 2, the centering coefficients $(alph, gam, theta)$ are set to $(1, 0, 1)$ for the rigid lid surface pressure method and $(1/3, 1/3, 1/2)$ for the implicit free surface method. The relevant equations corresponding to Equations (16.40), (16.41), and (16.42) are as follows

$$
\begin{aligned}
\hat{U}_{i,jrow} &= U^{\tau+1}_{i,jrow} + 2\Delta\tau\cdot\frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta ps_{i,jrow}}^{\phi}) + U^{\tau}_{i,jrow} && (16.43)\\
\hat{V}_{i,jrow} &= V^{\tau+1}_{i,jrow} + 2\Delta\tau\cdot\delta_\phi(\overline{\Delta ps_{i,jrow}}^{\lambda}) + V^{\tau}_{i,jrow} && (16.44)\\
\Delta ps_{i,jrow} &= ps^{\tau}_{i,jrow} - ps^{\tau-1}_{i,jrow} && (16.45)\\
\hat{U}_{i,jrow} &= \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} + U^{\tau}_{i,jrow} && (16.46)\\
\hat{V}_{i,jrow} &= \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} + V^{\tau}_{i,jrow} && (16.47)\\
\tilde{U}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,1} \\
&- \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda((1-gam)\cdot\overline{ps^{\tau-1}_{i-1,jrow-1}}^{\phi} + gam\cdot\overline{ps^{\tau}_{i-1,jrow-1}}^{\phi})) && (16.48)\\
\tilde{V}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,2} \\
&- \delta_\phi((1-gam)\cdot\overline{ps^{\tau-1}_{i-1,jrow-1}}^{\lambda} + gam\cdot\overline{ps^{\tau}_{i-1,jrow-1}}^{\lambda})) && (16.49)\\
zu_{i,jrow,1} &= f(gcor\cdot V^{\tau}_{i,k,j} + (1-gcor)\cdot V^{\tau-1}_{i,k,j}) + G_{i,jrow,1} && (16.50)
\end{aligned}
$$

$$zu_{i,jrow,2} \quad = \quad -f(gcor \cdot U^{\tau}_{i,k,j} + (1-gcor)\cdot U^{\tau-1}_{i,k,j}) + G_{i,jrow,2} \tag{16.51}$$

where U and V are vertically averaged velocity components and terms involving $\Delta ps_{i-1,jrow-1}$ have been dropped from Equations (16.46) and (16.47) to obtain the operator splitting as discussed in DM. Also, $\omega = 2\Delta\tau \cdot acor \cdot f$ and the forcing terms $G_{i,jrow,1}$ and $G_{i,jrow,2}$ contain all remaining terms which are known. The elliptic equation takes the form

$$\begin{aligned}
&\delta_\lambda(\overline{\frac{H_{i-1,jrow-1}}{\cos\phi^U_{jrow-1}}\delta_\lambda(\overline{\Delta ps_{i-2,jrow-2}}^{\phi})}^{\phi}) + \delta_\phi(\overline{H_{i-1,jrow-1}\cos\phi^U_{jrow-1}\delta_\phi(\overline{\Delta ps_{i-2,jrow-2}}^{\lambda})}^{\lambda})\\
-&\ \frac{\cos\phi^T_{jrow}dyt_{jrow}}{apgr\cdot 2\Delta\tau^2\cdot grav}\\
=&\ \frac{1}{apgr\cdot 2\Delta\tau}(\delta_\lambda(\overline{H_{i-1,jrow-1}\hat{U}_{i-1,jrow-1}}^{\phi}) + \delta_\phi(\cos\phi^U_{jrow-1}\overline{H_{i-1,jrow-1}\hat{V}_{i-1,jrow-1}}^{\lambda}))
\end{aligned} \tag{16.52}$$

Note that the piece $-\frac{\cos\phi^T_{jrow}dyt_{jrow}}{apgr\cdot 2\Delta\tau^2\cdot grav}$ is included only when the implicit free surface method is used. Also, the coefficient *apgr* is set to *alph* for leapfrog time steps and *theta* for mixing time steps. Equation (16.52) is solved using option *conjugate_gradient* with option *sf_9_point* with the number of islands *nislsp* set to zero because no island equations are being solved. After solving for Δps, the barotropic velocities at time level $\tau+1$ are given by

$$U^{\tau+1}_{i,jrow} \quad = \quad \hat{U}_{i,jrow} - \frac{apgr\cdot\Delta\tau}{\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta ps_{i-1,jrow-1}}^{\phi}) - U^{\tau}_{i,jrow} \tag{16.53}$$

$$V^{\tau+1}_{i,jrow} \quad = \quad \hat{V}_{i,jrow} - apgr\cdot\Delta\tau\delta_\phi(\overline{\Delta ps_{i-1,jrow-1}}^{\lambda}) - V^{\tau}_{i,jrow} \tag{16.54}$$

where the terms $U^{\tau}_{i,jrow}$ and $V^{\tau}_{i,jrow}$ are included only in the implicit free surface method. The predicted surface pressure is given by

$$ps^{\tau+1}_{i,jrow} = ps^{\tau-1}_{i,jrow} + \Delta ps_{i,jrow} \tag{16.55}$$

and the free surface elevation at time level $\tau+1$ is

$$\eta_{i,jrow} = \rho_\circ \cdot grav \cdot ps^{\tau+1}_{i,jrow} \tag{16.56}$$

but η is not explicitly needed in the code and so is not calculated. The vertical velocity at the top of the free surface is

$$adv\_vbt_{i,k=0,j} = (ps^{tau+1}_{i,jrow} - ps^{tau-1}_{i,jrow})/(g\Delta\tau) \tag{16.57}$$

The following setting are needed on various types of time steps:

Leapfrog time steps

The equations are as given above and the centering coefficients are given as

- implicit free surface method: Centering coefficient $apgr = alph$. If solving the Coriolis term explicitly ($acor = 0$), then centering coefficient $gcor = 1$. If solving the Coriolis term implicitly ($acor \neq 0$), then $acor$ is reset to $acor = alph$ and centering coefficient $gcor = gam$.

- rigid lid surface pressure method: Centering coefficient $apgr = alph$. If solving the Coriolis term explicitly ($acor = 0$), then centering coefficient $gcor = 1$ otherwise $gcor = 0$.

Mixing time steps (Forward and Euler backward)

The equations for forward mixing time steps and the first step of an Euler backward are modified to:

$$\hat{U}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} \tag{16.58}$$
$$\hat{V}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} \tag{16.59}$$
$$\tilde{U}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,1} - \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(\overline{ps^\tau_{i-1,jrow-1}}^\phi)) \tag{16.60}$$
$$\tilde{V}_{i,jrow} = 2\Delta\tau(zu_{i,jrow,2} - \delta_\phi(\overline{ps^\tau_{i-1,jrow-1}}^\lambda)) \tag{16.61}$$

The equations for the second step of an Euler backward are modified to:

$$\hat{U}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{U}_{i,jrow} + \omega\tilde{V}_{i,jrow}) + U^{\tau-1}_{i,jrow} \tag{16.62}$$
$$\hat{V}_{i,jrow} = \frac{1}{1+\omega^2}(\tilde{V}_{i,jrow} - \omega\tilde{U}_{i,jrow}) + V^{\tau-1}_{i,jrow} \tag{16.63}$$
$$\begin{aligned}\tilde{U}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,1} \\ &- \frac{1}{\cos\phi^U_{jrow}}\delta_\lambda((1-theta)\cdot\overline{ps^\tau_{i-1,jrow-1}}^\phi + theta\cdot\overline{ps^{\tau+1}_{i-1,jrow-1}}^\phi))\end{aligned} \tag{16.64}$$
$$\begin{aligned}\tilde{V}_{i,jrow} &= 2\Delta\tau(zu_{i,jrow,2} \\ &- \delta_\phi((1-theta)\cdot\overline{ps^\tau_{i-1,jrow-1}}^\lambda + theta\cdot\overline{ps^{\tau+1}_{i-1,jrow-1}}^\lambda))\end{aligned} \tag{16.65}$$

In both cases of mixing time steps, the time step factor $2\Delta\tau$ is replaced by $\Delta\tau$ (also in ω) and the centering coefficients are given as

- implicit free surface method: Centering coefficient $apgr = theta$. Centering coefficient $gcor = 0$. For implicit treatment of the Coriolis term, $acor$ is reset to $acor = theta$. Also on the second pass of an Euler backward time step the term $\frac{dyt_{jrow}\cos\phi^T_{jrow}dxt_i}{apgr\cdot g\cdot 2\Delta\tau^2}(ps^{\tau+1}_{i,jrow} - ps^\tau_{i,jrow})$ must be added to the right hand side of Equation (16.52).
- rigid lid surface pressure method: Centering coefficient $apgr = theta$. Centering coefficient $gcor = 1$ when the Coriolis term is handled explicitly but $gcor = 0$ when the Coriolis term is handled implicitly.

16.3.2 Remarks

It should be noted that Equation (16.36) only requires a Neumann boundary condition at the boundaries instead of the Direchlet boundary condition required for the elliptic equation of the stream function method. The implication is that there are no island equations to be solved and hence this method should be faster than the stream function method on massively parallel computers. Also, this method does not have a checkerboard null space as does the rigid lid surface pressure method. Whether the improved[10] *stream_function* approach or the *implicit_free_surface* approach is fastest is likely to be problem and computer platform dependent. This has yet to be explored.

The implicit free surface allows for prescription of a fresh water flux surface boundary condition on salinity rather than a salt flux which is required with the stream function or rigid lid surface pressure approach. However, provision for this has not been implemented in MOM 2. The implicit free surface method uses a salt flux upper boundary condition as do the other methods.

The rigid lid approximation essentially makes the speed of all external gravity waves infinite and therefore equilibrates them at all scales. This is reasonable in mid and high latitudes where there is a large time scale separation between gravity waves and Rossby waves. However, this separation of time scales is not the case in the equatorial domain. The implicit free surface method resolves both Rossby and gravity waves within the equatorial region while equilibrating the higher frequency gravity waves at mid and high latitudes. It remains to be shown if this difference between rigid lid and implicit free surface is significant.

Another point to be made is that Equation (16.36) contains a factor $H_{i,jrow}$ whereas the elliptic equation for the stream function contains the factor $1/H_{i,jrow}$. Therefore, the implicit free surface method should be less prone to stability problems than the stream function method when topography contains steep gradients. This has yet to be verified.

It should also be noted that the barotropic velocities $U_{i,jrow}^{\tau+1}$ and $V_{i,jrow}^{\tau+1}$ given by this method are not non-divergent. The degree of non-divergence is related to how accurately Equation (16.36) is solved for the change in surface pressure Δps and the accuracy depends on the tolerance variable *tolrfs* which is input through namelist and typically set to $10^{-4} gram/cm/sec^2$. Refer to Section 5.4 for information on namelist variables.

Use of polar filtering on $\hat{U}$ and $\hat{V}$ leads to problems. Removing the filtering eliminates the problems. So the filtering has been removed for this method. Whether it can be eliminated for the stream function method has yet to be determined.

16.4 explicit_free_surface

This is a version of the Killworth explicit free surface being implemented in MOM 2 by Martin Schmidt during the beta test period. Any questions should be directed to Martin at mschmidt@paula.io-warnemuende.de.

Section 16.4 contributed by
Martin Schmidt
mschmidt@paula.io − warnemuende.de

[10] Better numerics than in MOM 1.

Chapter 17

Elliptic Equation Solver Options

There are three methods for solving the external mode elliptic equation and two forms for the numerics of the coefficient matrix. One and only one of the methods should be chosen and one form of the coefficient matrix. All are described in the following sections.

17.1 conjugate_gradient

This option selects a conjugate gradient technique to invert the elliptic equation for all external mode options using either five point or nine point operators, selected by options *sf_9_point* or *sf_5_point*. The algorithm below is a Fortran 90 version of the algorithm in Smith, Dukowicz, and Malone (1992).

```
subroutine congrad (A, guess, forc, dpsi, iterations, epsilon)

use matrix_module

intent (in)     ::  A, guess, forc, epsilon
intent (out)    ::  dpsi, iterations

type(dpsi_type) ::  guess, dpsi, Zres, s
type(res_type)  ::  res, As, forc
type(operator)  ::  A
type(inv_op)    ::  Z
dimension (0:max_iterations) ::  dpsi, res, s, As, beta, alpha

dpsi(0) = guess
res(0)  = forc - A ⋆ dpsi(0)
beta(0) = 1
s(0)    = zerovector()
do k = 1 to max_iterations
       Zres(k-1) = Z ⋆ res(k-1)
       beta(k)   = res(k-1) ⋆ Zres(k-1)
       s(k)      = Zres(k-1) + (beta(k)/beta(k-1)) ⋆ s(k-1)
       As(k)     = A ⋆ s(k)
       alpha(k)  = beta(k) / (s(k) ⋆ As(k))
       dpsi(k)   = dpsi(k-1) + alpha(k) ⋆ s(k)
```

```
        res(k)    = res(k-1) - alpha(k) * As(k)
        estimated_error = err_estimate(k, alpha(k), s(k))
        if (estimated_error) < epsilon) exit
end do
if (k > max_iterations) then
        print *, 'did not converge in ',k,' iterations'
        stop 'congrad'
end if

iterations = k
dpsi = dpsi(k)

end
```

In MOM 2, all land masses are usually treated as islands when using this option with option *stream_function* because it has been found that this treatment speeds convergence. Island sums are turned off when solving for surface pressure by setting *nisle*=0. The test for convergence has also been changed from that used in MOM 1 and previous versions. In MOM 2, the sum of all estimated future corrections to the prognostic variable is calculated assuming geometric decrease of the maximum correction, and the iteration is terminated when this sum is within the requested tolerance. Tests indicate that this solver converges significantly faster than the one in MOM 1[1].

Section 17.1 contributed by
Charles Goldberg
chg@gfdl.gov

17.2 oldrelax

This is a sequential over-relaxation method which requires an over-relaxation constant. This constant is dependent on geometry and topography and the optimum one can be found by using script *run_poisson* to exercise the elliptic solvers. This method uses the coefficient matrices given by options *sf_9_point* or *sf_5_point*. This is not the recommended method and is retained for compatibility reasons and in case option *conjugate_gradient* should fail on a particular configuration.

17.3 hypergrid

This is a more highly vectorized version of *oldrelax* using the coefficient matrices given by options *sf_9_point* or *sf_5_point*. This is not the recommended method and is retained for compatibility reasons and in case option *conjugate_gradient* should fail on a particular configuration.

17.4 sf_9_point

This option uses the full nine point numerics for the coefficient matrix in Equation (16.17) when inverting the external mode elliptic Equation (16.13). This matrix is slightly different than the

[1]Because of the difference in meaning between the MOM 1 tolerance *crit* and the MOM 2 tolerance *tolrsf*, care must be taken to assure that both are converging to the same tolerance when doing these tests.

one used in MOM 1 and earlier versions. Particular attention has been given to assuring that the operator remains symmetric with respect to islands. This form exactly eliminates[2] the surface pressure term from the momentum equations when used with option *stream_function*. It has a checkerboard null space which in most cases is not a problem because of the Dirichlet boundary conditions.

Options *rigid_lid_surface_pressure* and *implicit_free_surface* require the use of a nine point coefficient matrix[3], and in these cases, option *sf_9_point* must also be enabled. The null space is a problem when option *rigid_lid_surface_pressure* is chosen, and it must be removed. Adding the missing divergence from the *rigid_lid_surface_pressure* to the central term constructs the *implicit_free_surface* which suppresses the null space. Details of these methods appear later.

Constructing the coefficient matrix for Equation (16.13) can be difficult. The mathematician among us (Goldberg) points out that the finite difference operators can be written in terms of matrices as follows:

$$dxt_i \cdot \delta_\lambda(\overline{\alpha_{i,jrow}}^{\phi}) = \sum_{j'=-1}^{0} \sum_{i'=-1}^{0} cddxt_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{17.1}$$

$$dyt_{jrow} \cdot \delta_\phi(\overline{\alpha_{i,jrow}}^{\lambda}) = \sum_{j'=-1}^{0} \sum_{i'=-1}^{0} cddyt_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{17.2}$$

$$dxu_i \cdot \delta_\lambda(\overline{\alpha_{i,jrow}}^{\phi}) = \sum_{j'=0}^{1} \sum_{i'=0}^{1} cddxu_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{17.3}$$

$$dyu_{jrow} \cdot \delta_\phi(\overline{\alpha_{i,jrow}}^{\lambda}) = \sum_{j'=0}^{1} \sum_{i'=0}^{1} cddyu_{i',j'}\ \alpha_{i+i',jrow+j'} \tag{17.4}$$

where $cddxt_{i',j'}$, $cddyt_{i',j'}$, $cddxu_{i',j'}$, and $cddyu_{i',j'}$ are defined as 2x2 matrices:

$$cddxt_{i',j'} = (a_{i',j'}) = \begin{pmatrix} a_{-1,0} & a_{0,0} \\ a_{-1,-1} & a_{0,-1} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{17.5}$$

$$cddyt_{i',j'} = (a_{i',j'}) = \begin{pmatrix} a_{-1,0} & a_{0,0} \\ a_{-1,-1} & a_{0,-1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \tag{17.6}$$

$$cddxu_{i',j'} = (a_{i',j'}) = \begin{pmatrix} a_{0,1} & a_{1,1} \\ a_{0,0} & a_{1,0} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{17.7}$$

$$cddyu_{i',j'} = (a_{i',j'}) = \begin{pmatrix} a_{0,1} & a_{1,1} \\ a_{0,0} & a_{1,0} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \end{pmatrix} \tag{17.8}$$

Using Equations (17.1) – (17.4), the first brackets in Equation (16.13) can be rewritten as

[2]To within roundoff.

[3]However, in these cases, the elliptic equations for the prognostic surface pressure p^s are, of course, different than those given above for the prognostic stream function Ψ.

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\Bigg[cddyu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^U_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}$$
$$+\,cddxu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^U_{jrow+j''}\cdot 2\Delta\tau}\Bigg]\Delta\psi_{i+i'+i'',jrow+j'+j''} \tag{17.9}$$

and in a similar fashion, the implicit Coriolis contributions from the second brackets in Equation (16.13) can be rewritten as:

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\Bigg[-cddxu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}$$
$$-cddyu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\Bigg]\Delta\psi_{i+i'+i'',jrow+j'+j''} \tag{17.10}$$

which provides a very compact and efficient way of calculating the coefficient matrix $coeff_{i,jrow,i^\star,j^\star}$.

$$coeff_{i,jrow,i^\star,j^\star} =$$
$$\sum_{i'=0}^{1}\sum_{i''=-1}^{0}\sum_{j'=0}^{1}\sum_{j''=-1}^{0}\delta_{i'+i'',i^\star}\delta_{j'+j'',j^\star}\Bigg[cddyu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^U_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}$$
$$+\,cddxu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^U_{jrow+j''}\cdot 2\Delta\tau}$$
$$-cddxu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}$$
$$-cddyu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\Bigg] \tag{17.11}$$

where $\delta_{i'+i'',i^\star}$ is the Kronecker Delta function which is 1 when $i'+i''=i^\star$ and 0 when $i'+i''\neq i^\star$. The entire coefficient array is calculated by six nested loops on i, $jrow$, i', j', i'', and j'', with each iteration adding the terms above to the proper coefficient bucket[4].

The matrix for right hand side of Equation (16.13) can be written as:

$$forc_{i,jrow}=\sum_{i'=0}^{1}\sum_{j'=0}^{1}\Big[-cddyt_{i',j'}\cdot zu_{i+i',jrow+j',1}\cdot dxu_{i+i'}\cdot\cos\phi^U_{jrow+j'}$$
$$+cddxt_{i',j'}\cdot zu_{i+i',jrow+j',2}\cdot dyu_{jrow+j'}\Big] \tag{17.12}$$

The coefficient matrix is symmetric[5] except for the implicit Coriolis term (the second bracket in Equation (16.13)). The preferred method of solving Equation (16.13) is by conjugate gradients as described in Dukowicz, Smith and Malone (1993). Refer to Section 17.1 for more detail on the elliptic solvers and to Appendix 16.1.4 for more details on the island equations.

[4]We have come to call these loops "Amtrack Normal Form," because the indentation of the DO loops and the long executable statement in the middle resemble the "Amtrack" logo. Properly ordered, these nested loops vectorize very well on a Cray YMP.

[5]Refer to Section 16.1.5.

17.5 sf_5_point

This form approximates averages of contributions to the corner points of the nine point operator to produce a five point operator that has coefficients only in the center, north, east, south, and west places in the coefficient matrix for inverting the external mode elliptic equation. This option can be used with option *stream_function* but is not appropriate for options *rigid_lid_surface_pressure* or *implicit_free_surface* because of energy leakage.

In Equation (17.9), corresponding to the first bracket of Equation (16.1), all 32 contributions to the elliptic operator are calculated as in the 9 point operator, except that in the first group, i.e., in the terms originating in a second difference in the δ_ϕ direction, the coefficient of the northeast variable $\Delta\phi_{i,jrow,1,1}$ is averaged with the coefficient of the northwest variable $\Delta\phi_{i,jrow,-1,1}$, and both are applied to the northern variable $\Delta\phi_{i,jrow,0,1}$. This averaging centers the coefficient on the northern edge of the central T cell, so that continuous derivatives in the ϕ direction are well approximated. Similarly, the southeast and southwest contributions from the first group are applied to the southern variable $\Delta\phi_{i,jrow,0,-1}$. The second group of terms originate in a second difference in the δ_λ direction, and here the northeastern and southeastern contributions are averaged and applied to the eastern variable $\Delta\phi_{i,jrow,1,0}$, and the northwestern and southwestern contributions are averaged and applied to the western variable $\Delta\phi_{i,jrow,-1,0}$, giving good approximations to the continuous derivatives in the λ direction. The resulting summations are

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddyu_{i',j'}\cdot cddyt_{i'',j''}\cdot\frac{dxu_{i+i''}\cdot\cos\phi^U_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dyu_{jrow+j''}\cdot 2\Delta\tau}\right]\Delta\psi_{i,jrow+j'+j''}$$
$$+\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\left[cddxu_{i',j'}\cdot cddxt_{i'',j''}\cdot\frac{dyu_{jrow+j''}}{H_{i+i'',jrow+j''}\cdot dxu_{i+j''}\cdot\cos\phi^U_{jrow+j''}\cdot 2\Delta\tau}\right]\Delta\psi_{i+i'+i'',jrow} \tag{17.13}$$

Bryan (1969) uses a five point approximation to $(\frac{1}{H}\cdot\nabla\psi_t)$ which as implemented in MOM 1 takes the form

$$\frac{1}{\cos\phi^T_{jrow}dxt_i\; dyt_{jrow}}\left[dxt_i\cdot dyt_{jrow}\cdot\delta_\phi\left(\frac{\cos\phi^U_{jrow-1}}{\overline{H_{i-1,jrow-1}}^{\lambda}\cdot dyu_{jrow-1}}\,dyu_{jrow-1}\cdot\delta_\phi\Delta\psi_{i,jrow-1}\right)\right.$$
$$\left.+\,dyt_{jrow}\cdot dxt_i\cdot\delta_\lambda\left(\frac{1}{\overline{H_{i-1,jrow-1}}^{\phi}\cdot\cos\phi^T_{jrow}\;dxu_{i-1}}\,dxu_{i-1}\cdot\delta_\lambda\Delta\psi_{i-1,jrow}\right)\right] \tag{17.14}$$

In this notation, the five point approximation used in MOM 2 takes the form

$$\frac{1}{2\Delta\tau}\cdot\left[dyt_{jrow}\cdot\delta_\phi\left(\overline{\left(\frac{dxu_{i-1}\cdot\cos\phi^U_{jrow-1}}{H_{i-1,jrow-1}\cdot dyu_{jrow-1}}\right)}^{\lambda}\cdot dyu_{jrow-1}\cdot\delta_\phi\Delta\psi_{i,jrow-1}\right)\right.$$
$$\left.+\,dxt_i\cdot\delta_\lambda\left(\overline{\left(\frac{dyu_{jrow-1}}{H_{i-1,jrow-1}\cdot\cos\phi^U_{jrow-1}\cdot dxu_{i-1}}\right)}^{\phi}\cdot dxu_{i-1}\cdot\delta_\lambda\Delta\psi_{i-1,jrow}\right)\right] \tag{17.15}$$

For comparison with MOM 2, Bryan's form must be multiplied by $(\cos\phi^T_{jrow} dxt_i dyt_{jrow})/(2\Delta\tau)$. The principal differences between these two forms arise from Bryan's use of averages of reciprocals of the form $\frac{2}{H_{i,jrow} + H_{i-1,jrow}}$ where MOM 2 uses $\frac{1}{2}(\frac{1}{H_{i,jrow}} + \frac{1}{H_{i-1,jrow}})$. These differ by a factor of two in the second order term in $H_{i,jrow} - H_{i-1,jrow}$. Other differences arise on a nonuniform grid.

The second bracket, the implicit Coriolis terms, may be seen to already be in five point form because each corner coefficient consists of two terms of equal magnitude, but opposite sign.

$$\sum_{i'=0}^{1}\sum_{j'=0}^{1}\sum_{i''=-1}^{0}\sum_{j''=-1}^{0}\Bigg[-cddxu_{i',j'} \cdot cddyt_{i'',j''} \cdot \frac{-\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}} - cddyu_{i',j'} \cdot cddxt_{i'',j''} \cdot \frac{\tilde{f}_{jrow+j''}}{H_{i+i'',jrow+j''}}\Bigg] \Delta\psi_{i+i'+i'',jrow+j'+j''} \tag{17.16}$$

The right hand side of Equation (16.13) also remains the same in the five point equations as in the nine point equations.

$$forc_{i,jrow,} = \sum_{i'=0}^{1}\sum_{j'=0}^{1}\Big[-cddyt_{i',j'} \cdot zu_{i+i',jrow+j',1} \cdot dxu_{i+i'} \cdot \cos\phi^U_{jrow+j'} + cddxt_{i',j'} \cdot zu_{i+i',jrow+j',2} \cdot dyu_{jrow+j'}\Big] \tag{17.17}$$

Although the five point operator approximates the continuous differential equation (16.1) well, its solutions are not exact solutions of the finite difference momentum equations (16.2) and (16.3). Moreover, the time saved by calculating a five point operator instead of a nine point operator in two places per iteration in a conjugate gradient solver is not large, and the nine point equations usually take fewer iterations to converge.

Section 17.5 contributed by
Charles Goldberg
chg@gfdl.gov

Chapter 18

Diagnostic Options

18.1 Design

MOM 2 is instrumented with a variety of diagnostic options. Some are useful for diagnosing model problems while others are aimed towards providing information to help resolve questions of a more scientific nature. All are independent of each other and each is activated with its own option at compile time. For added flexibility, each diagnostic has an associated interval, control, and possibly an averaging period variable which are input through namelist. To see all namelist variables, refer to Section 5.4.

When not enabled, a diagnostic requires neither memory nor computational time. When enabled, some require large amounts of memory, disk, or cpu time so it is important to use them cautiously with specific goals in mind. The amount of time generally depends on the particular configuration of MOM 2, which diagnostics are enabled, the interval between diagnostic output, and the averaging period (if applicable). An assessment of the computational time[1] can easily be made by enabling the required diagnostics along with option *timing*[2] in a short model execution.

In addition to the diagnostics listed in the following sections, there are numerous "debug" options in critical areas of the source code which can be enabled to give more information for debugging purposes. An example of one of these options would be *debug_adv_vel* near the bottom of file *adv_vel.F* which computes advective velocities. When enabled, *debug_adv_vel* gives the components of the divergence of advective velocities for all T cells and U cells in the vertical at any $i, jrow$ location. These "debug" options can also be found using UNIX *grep* as described in Section 19.5.

Recommendation

MOM 2 can produce many types of diagnostics and all are described within this chapter. How to organize and view results from these diagnostics has been a major problem in the past. With the adoption of a NetCDF standard, results can now be viewed almost without effort. A good way to visualize results is to use Ferret which is a graphical analysis tool developed by Steve Hankin (1994) at NOAA/PMEL (email: ferret@pmel.noaa.gov URL: http://www.pmel.noaa.gov/ferret/home.html)

[1]Note that when enabling lots of diagnostics, substantial reduction in computational time can be realized by opening up the memory window even on a single processor. As an example, execute the test case script *run_mom* and open the memory window wider than the minimum (jmw¿3 in file *size.h*). Of course, the price to be paid is an increase in the required memory.

[2]After timing a particular configuration, option *timing* should always be disabled because the act of timing takes non-negligible time!

18.1.1 NetCDF formatted data

In general, the format of diagnostic output files can be either in 32bit unformatted IEEE as described in Section 18.1.2 or as NetCDF. Saving diagnostic data in NetCDF format is the preferred approach since diagnostic output is immediately accessible to visualization packages which support NetCDF. This means that there is no need to write and maintain code for manipulating diagnostic data to visualize it. NetCDF format also allows data to be passed between various computer platforms easily and is therefore the preferred method for sharing data. The option for NetCDF format can be specified in two ways: either by enabling the catch all option *netcdf* which makes all enabled diagnostics save their data in NetCDF format or by selectively enabling a NetCDF option for individual diagnostics as described under each diagnostic. Refer to option *netcdf* or particular diagnostic options for more details. For suggestions on executing in double precision on workstations and using NetCDF, refer to Section 19.11.

When data is written as unformatted 32 bit IEEE, the output file is given the suffix *.dta*[3]. to distinguish it from NetCDF format files which have the suffix *.dta.nc*.

18.1.2 IEEE formatted data

The preferred method of saving diagnostic data is NetCDF as described in Secton 18.1.1. Prior to NetCDF availibility, diagnostic data was written using 32 bit IEEE format. This section describes the record structure of data written with 32bit unformatted IEEE writes. To access this data for visualization, analysis code must be written. In order to do this, the record structure must be known. When data is written as unformatted 32 bit IEEE, the output file is given the suffix *.dta* to distinguish it from NetCDF format files which have the suffix *.dta.nc*. If NetCDF format is not specified, then diagnostic output is written as: 32 bit IEEE unformatted data, formatted ascii text, or both depending on the value of a control variable specific to the particular diagnostic.

The record structure of any non-netcdf diagnostic file can be found by examining the sections of MOM 2 which write the data. This information is also embedded in header records. When writing native unformatted data records in MOM 2, each data record is preceded by a header record of the form

```
write (iounit) stamp, iotext, expnam
```

Where *stamp* is a 32 character specification of the model date and time corresponding to the time step[4] when the data was written, *iotext* is an 80 character description of what is in the data record and how it is to be read, and *expnam* is a 60 character experiment name which identifies an experiment that wrote the data. All 32 bit IEEE unformatted diagnostic datasets in MOM 2 have this structure. This makes it easy to decipher any unformatted output from MOM 2. The following program is intended as an example of how to decphfer any MOM 2 unformatted diagnostic output file and is intended to run on a workstation:

```
Program decifer

character*32 stamp
character*80 iotext
character*60 expnam
character*30 filename
iounit = 21
filename = 'snapshots.dta'                         !  Set the file name
```

[3]Other files have the same *.dta* suffix but are not 32 bit IEEE data. Refer to Section 19.7 for details.

[4]For datasets which are averaged through time, this stamp corresponds to the end of each averaging period. The length of each averaging period is also saved in this case.

```
open(iounit,FILE=filename,FORM='unformatted',ACCESS='sequential')
rewind iounit
do n=1,100000
read (iounit, end=110) stamp, iotext, expnam  !  read the header record
write (*,'(1x,a32,1x,a80)') stamp, iotext     !  show the header record
read (iounit)                                 !  skip the data record
enddo
110   continue
write (*,*) " End of file on ",filename," on unit ",iounit
stop
end
```

Note that when printed, the header records give the Fortran statement[5] needed to read the following data record. In principal, this presents an interesting concept. Program *decifer* could write the Fortran program needed to read the data! However, details have not been worked out because there is a better approach and that is NetCDF format.

Recommendation

Move over to saving NetCDF formatted data as soon as possible. Eventually, in an effort to minimize redundancy, code for saving data in 32 bit IEEE format will be removed. Newly added diagnostics do not have an option to write 32 bit IEEE format.

18.1.3 Sampling data

Depending on the particular diagnostic, output may consist of instantaneous or time averaged data. Instantaneous data is written out periodically on those time steps that fall nearest to the end of a specified interval. For instance, specifying a 30 day interval means that results are written out every 30 days from some specified reference time. The reference time is also specified through namelist and is the same for all diagnostics.

Averaged data starts out being accumulated over all time steps within a specified interval. It is then averaged and written out periodically at the end of the interval. If the interval were 30 days, results would be output every 30 days from the reference time and would represent 30 day averages. It is possible to produce sub sampled averages by specifying an averaging period as less than an interval. The averaging period is also input through namelist. For example, if the interval was set to 30 days and the averaging period was set to 2.0 days, then 2 day averages would be written out every 30 days and the averaging would be over days 29 and 30 of each interval.

How often is it necessary to sample model generated data? It should be sampled often enough to resolve the shortest time scale which is of interest. Of course, it follows that longer time scales will also be resolved. For example, if data is sampled at an interval of once per month, the implicit assumption is that time scales with periods of four months and longer are being resolved.

What happens when the data has energy at periods of one week? If data is sample instantaneously once per month, then sampled data will be aliased by the shorter period. One way to remove this alias is to produce a monthly climatology. This is done by saving data for many years and averaging all Januaries together, all Februaries together and so forth to produce twelve climatological or "mean" months. The alias error reduces as more and more years are included in the climatology.

[5]Sometimes only symbolically.

A better way is to filter out energy at periods of one week by constructing monthly averages which are then saved once per month. A dataset of monthly averages represents estimates that are statistically more stable than a dataset of instantaneous values. If there is little energy at periods shorter than the period of interest, then saving instantaneous samples is essentially the same as saving monthly averages.

The averaging period need not always equal the interval at which results are saved. This applies when frequencies are widely separated. For instance, suppose that the amplitude of a diurnal period (one cycle per day) is not negligible compared to the amplitude of an annual period (one cycle per year). The diurnal period can be effectively removed and the annual period resolved by saving a three or four day average at the end of every month.

18.1.4 Regional masks

For use with certain diagnostics calculations, the model domain may be sub-divided into a number of regions over which calculations are averaged. An arbitrary number of non-overlapping regions of areal extent are defined by setting a horizontal region mask number $mskhr_{i,jrow}= m$ where $(i, jrow)$ is in region m for $m = 1 \cdots nhreg$ and $nhreg$ is the number of regions. One way to do this is to use the subroutine *sethr* which assigns $mskhr_{i,jrow}$to a region number within a rectangular region. Regions may be built up from calls to *sethr* but in general need not be simply rectangular.

In a similar fashion, the vertical region mask number $mskvr_k= \ell$ where k is in vertical region ℓ for $\ell = 1 \cdots nvreg$ and $nvreg$ is the number of vertical regions. Unlike $mskhr_{i,jrow}$which may contain a region which is multiply connected, the k indices within a vertical region ℓ must be contiguous.

Regional volumes are constructed in subroutine *setocn* by the union of $mskhr_{i,jrow}$and $mskvr_k$such that the regional volume numbers are given by

$$nreg = nhreg \cdot (mskvr_k - 1) + mskhr_{i,jrow} \tag{18.1}$$

18.2 List of diagnostic options

18.2.1 cross_flow_netcdf

How much of the flow is along isopycnal surfaces and how much cuts across isopycnal surfaces? Option *cross_flow_netcdf* computes the projection of a flow field into an along isopycnal component and a cross isopycnal (diapycnal) component. Let the unit vector normal to the isopycnal surface at cell $T_{i,k,jrow}$be given by

$$\hat{S}_{i,k,j} = \frac{\vec{S}_{i,k,j}}{|\vec{S}_{i,k,j}|} \tag{18.2}$$

where

$$\vec{S}_{i,k,j} = \frac{1}{\cos\phi^T_{jrow}} \overline{\delta_\lambda(\rho_{i-1,k,j})}^{\lambda} \cdot \hat{x} \; + \; \overline{\delta_\phi(\rho_{i,k,j-1})}^{\phi} \cdot \hat{y} \; + \; \overline{\delta_z(\rho_{i,k-1,j})}^{z} \cdot \hat{z} \tag{18.3}$$

and $\rho_{i,k,j}$ is the local in-situ density which is used to approximate the isopycnal surface at time level τ, and $\hat{x}$, $\hat{y}$, and $\hat{z}$ are local unit vectors in longitude, latitude, and depth. Note that both in-situ and potential density fields are saved when diagnostic option *density_netcdf* is enabled. If the velocity field at cell $T_{i,k,jrow}$is approximated as

$$\tilde{V}_{i,k,j} = \overline{u_{i-1,k,j-1,1,\tau}}^{\lambda\phi} \cdot \hat{x} \;+\; \overline{u_{i-1,k,j-1,2,\tau}}^{\lambda\phi} \cdot \hat{y} \;+\; \overline{adv\_vbt_{i,k-1,j}}^{z} \cdot \hat{z} \tag{18.4}$$

then the instantaneous diapycnal $\vec{D}$ and along isopycnal $\vec{A}$ velocities are given by

$$\vec{D}_{i,k,j} = \tilde{V}_{i,k,j} \bullet \hat{S}_{i,k,j} \tag{18.5}$$
$$\vec{A}_{i,k,j} = \tilde{V}_{i,k,j} - \vec{D}_{i,k,j} \tag{18.6}$$

Output in 32 bit IEEE unformatted data is not an option for this diagnostic. The three velocity components (zonal, meridional, and vertical) of $\vec{D}$ and $\vec{A}$ are output only in NetCDF format to file *cross.dta.nc* and the interval between output is specified by namelist variable *crossint* in units of days. Refer to Section 5.4 for information on namelist variables.

18.2.2 density_netcdf

It is useful to construct both a locally referenced potential density $\rho^{full}_{i,k,j}$ and potential density $\sigma_{i,k,j}$ referenced to a specific depth. The superscript *in* is used to differentiate between potential density anomaly (used throughout MOM 2) and the full potential density. To construct the full density $\rho^{full}_{i,k,j}$ in units of gm/cm^3 at each T cell, the mean reference density must be added to the deviation

$$\rho^{full}_{i,k,j} = \rho_{i,k,j} + \rho^{ref}_{k} \tag{18.7}$$

where ρ^{ref}_{k} is described in Section 6.2.2. When option *potential_density* is enabled, σ_0 (referenced to the surface), σ_1 (referenced to 1000m), σ_2 (referenced to 2000m), and σ_3 (referenced to 3000m) are also constructed and saved.

Using, $\rho_{i,k,j}$, the hydrostatic pressure is given by integrating Equation 11.79 to yield

$$p_{i,1,j} = grav \cdot dzw_0 \cdot \rho_{i,1,j} \quad for\ k = 1 \tag{18.8}$$
$$p_{i,k,j} = \sum_{m=2}^{k} grav \cdot dzw_{m-1} \overline{\rho_{i,m-1,j}}^{z} \quad for\ k = 2\ to\ km \tag{18.9}$$

The internal mode pressures in units of $gm/cm/sec^2$ are given by removing the vertical mean

$$p^{int}_{i,k,j} = p_{i,k,j} - \frac{1}{zw_{k=kmt(i,jrow)}} \sum_{k=1}^{kmt(i,jrow)} p_{i,k,j} \cdot dzt_k \tag{18.10}$$

The external mode pressure is available from diagnostic *diagnostic_surf_height*. Note that output from this diagnostic is only available in NetCDF format in file *density.dta.nc* and the interval between output is specified by namelist variable *densityint* in units of days. Refer to Section 5.4 for information on namelist variables.

18.2.3 diagnostic_surf_height

Option *diagnostic_surf_height* constructs an average sea surface elevation from the prognostic stream function. It does this by accumulating a forcing term in time, then averaging over a specified interval (because this is a linear problem) to produce an average forcing X and solving an elliptic equation of the form

$$\nabla \cdot H \nabla \overline{p^s} = X \tag{18.11}$$

for average surface pressure $\overline{p^s}_{i,jrow}$ which is then converted into height using

$$dsp_{i,jrow} = \frac{\overline{p^s}_{i,jrow}}{\rho_0 \cdot grav} \tag{18.12}$$

where the height $dsp_{i,jrow}$ is in units of *cm*.

The process begins after solving for the external mode stream function. The surface pressure gradient terms can then be reconstructed from Equations (16.2) and (16.3) as

$$\begin{aligned} \frac{1}{\rho_0 \cos\phi^U_{jrow}} \delta_\lambda(\overline{p^s_{i,jrow}}^{\phi}) &= \frac{1}{2\Delta\tau}\Big(\frac{1}{H_{i,jrow}}\delta_\phi(\overline{\Delta\psi_{i,jrow}}^{\lambda})\Big) + \frac{\tilde{f}_{jrow}}{H_{i,jrow}\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta\psi_{i,jrow}}^{\phi}) \\ &\quad + zu_{i,jrow,1} \end{aligned} \tag{18.13}$$

$$\begin{aligned} \frac{1}{\rho_0} \delta_\phi(\overline{p^s_{i,jrow}}^{\lambda}) &= \frac{1}{2\Delta\tau}\Big(-\frac{1}{H_{i,jrow}\cdot\cos\phi^U_{jrow}}\delta_\lambda(\overline{\Delta\psi_{i,jrow}}^{\phi})\Big) + \frac{\tilde{f}_{jrow}}{H_{i,jrow}}\delta_\phi(\overline{\Delta\psi_{i,jrow}}^{\lambda}) \\ &\quad + zu_{i,jrow,2}` \end{aligned} \tag{18.14}$$

where $\Delta\psi_{i,jrow}$ is given by Equation (16.9) and $\tilde{f}_{jrow}$ is given by Equation (16.4). Since the coefficient matrix required for solving Equation (18.11) is generated within MOM 2 for purposes of solving the prognostic surface pressure and implicit free surface methods, this coefficient matrix can be easily generated. It is done by essentially replacing $1/H$ with H in the method given within Section 17.4. The average forcing X is computed as

$$X = \frac{1}{L}\sum_{\ell=1}^{L}\delta_\lambda\left(\overline{\frac{H_{i-1,jrow-1}}{\cos\phi^U_{jrow-1}}\cdot\delta_\lambda(\overline{p^s_{i-1,jrow-1}}^{\phi})}^{\phi}\right) + \delta_\phi\left(\overline{H_{i-1,jrow-1}\cdot\cos\phi^U_{jrow-1}\cdot\delta_\phi(\overline{p^s_{i-1,jrow-1}}^{\lambda})}^{\lambda}\right) \tag{18.15}$$

with the aid of Equations (18.13) and (18.14) where L is the number of timesteps in the averaging period. Equation (18.11) is then inverted for $\overline{p^s}_{i,jrow}$ by the method of conjugate gradients.

Equations (18.13) and (18.14) may be solved directly by line integrals to construct $\overline{p^s}_{i,jrow}$. However, the integration paths must be chosen carefully so as not to propagate errors introduced by a non-zero tolerance when solving the stream function equation. Choosing nine point numerics by enabling option *sf_9_point* minimizes these errors compared to using option *sf_5_point*.

The solution $\overline{p^s}_{i,jrow}$ is thus an average and may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *diag_surf.dta*. If option *netcdf* or *diagnostic_surf_height_netcdf* is enabled, data is written in NetCDF format to file *diag_surf.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *dspint* and the control is specified by variable *iodsp*. The averaging period is typically the same as the interval but may be specified shorter by variable *dspper*. These variables are input through namelist. Refer to Section 5.4 for information on namelist variables.

18.2.4 energy_analysis

Option *energy_analysis* computes the instantaneous scalar product of $\vec{u}$ and the momentum Equations (2.1) and (2.2) integrated over the entire domain to verify energy conservation as demonstrated for the continuous equations in Section 2.2. The finite discrete equivalent is demonstrated in Chapter 12. From Equations (2.13) and (2.14) the total velocity $\vec{u}$ is divided into internal mode $\hat{u}$ and external mode $\overline{u}$ components. The work done by each term in Equations (2.1) and (2.2) is broken into internal and external mode contributions. Note that this approach is different than that given in Cox (1984). Units are in $gm/cm/sec^3$.

Define two operators for integrating the scalar product of $\vec{u}$ and any terms in Equations (2.1) and (2.2). In finite difference form, they are

$$< \overline{\alpha_{i,k,j,n}}^{ext} >= \frac{1}{Vol^U} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \overline{u}_{i,j,n,\tau} \cdot (\alpha_{i,k,j,n}) \Delta^U_{i,k,jrow} \quad (18.16)$$

$$< \overline{\alpha_{i,k,j,n}}^{int} >= \frac{1}{Vol^U} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \hat{u}_{i,k,j,n,\tau} \cdot (\alpha_{i,k,j,n}) \Delta^U_{i,k,jrow} \quad (18.17)$$

$$(18.18)$$

$< \overline{\alpha_{i,k,j,n}}^{ext} >$ represents the external mode component of the work done by $\alpha_{i,k,j,n}$, $< \overline{\alpha_{i,k,j,n}}^{int} >$ represents the internal mode component of the work done by $\alpha_{i,k,j,n}$, and the relation between j and *jrow* is as described in Section 5.2. The volume element and total volume are given by

$$\Delta^U_{i,k,jrow} = umask_{i,k,jrow} \cdot dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \quad (18.19)$$

$$Vol^U = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^U_{i,k,jrow} \quad (18.20)$$

Work done by each term in the momentum equations is computed using operators described in Sections 11.11.5.

Total change in kinetic energy

$$E =< \overline{\frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{ext} > + < \overline{\frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{int} > \quad (18.21)$$

Non-linear terms

The work done by non-linear terms on the velocity is broken into two parts: G1 is the horizontal part including the metric term and G2 is the vertical part. G1 + G2 should be zero. The "nonlinear error" = G1 + G2 which should be roundoff to within machine precision. On a Cray YMP with 64 bit words, this amounts to the error being about 1x10^{-12} smaller than the leading term in the computation of G1 and G2.

$$\begin{aligned} G1 &= < \overline{ADV\_Ux_{i,k,j} + ADV\_Uy_{i,k,j} + ADV\_metric_{i,k,j,n}}^{ext} > \\ &\quad + < \overline{ADV\_Ux_{i,k,j} + ADV\_Uy_{i,k,j} + ADV\_metric_{i,k,j,n}}^{int} > \quad (18.22) \\ G2 &= < \overline{ADV\_Uz_{i,k,j}}^{ext} > \\ &\quad + < \overline{ADV\_Uz_{i,k,j}}^{int} > \quad (18.23) \end{aligned}$$

Buoyancy

$$B = < \overline{grav \cdot adv\_vbt_{i,k,j} \cdot \overline{\rho_{i,k,j}}^z}^{ext} > + < \overline{grav \cdot adv\_vbt_{i,k,j} \cdot \overline{\rho_{i,k,j}}^z}^{int} > \tag{18.24}$$

Horizontal pressure gradients

$$G = < \overline{grad\_p_{i,k,j,n}}^{ext} > + < \overline{grad\_p_{i,k,j,n}}^{int} > \tag{18.25}$$

The work done by the horizontal pressure gradients should equal the work done by buoyancy. G − B should be zero. The imbalance G − B is the "energy conversion error" which should be roundoff within machine precision. On a Cray YMP with 64 bit words, this amounts to the error being about 1x10^{-12} smaller than either term.

Mixing

The work done by viscous mixing terms on the velocity is broken into two parts

$$\begin{aligned} D1 &= < \overline{DIFF\_Ux_{i,k,j} + DIFF\_Uy_{i,k,j} + DIFF\_metric_{i,k,j,n}}^{ext} > \\ &\quad + < \overline{DIFF\_Ux_{i,k,j} + DIFF\_Uy_{i,k,j} + DIFF\_metric_{i,k,j,n}}^{int} > && (18.26) \\ D2 &= < \overline{DIFF\_Uz_{i,k,j}}^{ext} > + < \overline{DIFF\_Uz_{i,k,j}}^{int} > && (18.27) \end{aligned}$$

Wind and Bottom drag

$$\begin{aligned} W1 &= < \overline{smf_{i,j,n}}^{ext} > + < \overline{smf_{i,j,n}}^{int} > && (18.28) \\ W2 &= < \overline{bmf_{i,j,n}}^{ext} > + < \overline{bmf_{i,j,n}}^{int} > && (18.29) \end{aligned}$$

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *energy_int.dta*. If option *netcdf* or *energy_analysis_netcdf* is enabled, data is written in NetCDF format to file *energy_int.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *glenint* and the control is specified by variable *ioglen*.

18.2.5 fct_netcdf

Option *fct_netcdf* is intended to show the difference between second order advection of tracers and the flux corrected transport. In other words, where the FCT is diffusing tracers with the lower order upwind advection. This is done by saving the change in tracer per time step due to the flux corrected transport scheme minus a second order advection term.

Output from this diagnostic is only available in NetCDF format. If option *fct_netcdf* is enabled, data is written in NetCDF format to file *fct.dta.nc*. The interval between output is specified by variable *fctint* and the data is instantaneous.

18.2.6 gyre_components

Option *gyre_components* computes instantaneous values of various components of northward tracer transport. The longitudinally averaged temperature and meridional velocity are constructed for the latitude of ϕ^U_{jrow}as a function of depth. Also, the vertically averaged temperature and meridional velocity are constructed for the latitude of ϕ^U_{jrow}as a function of longitude

$$< \overline{T}^{\phi}_{k,jrow} >^{\lambda} = \frac{1}{Volx^T_{k,jrow}} \sum_{i=2}^{imt-1} \overline{t_{i,k,j,n,\tau}}^{\phi} \Delta^T_i \tag{18.30}$$

$$< V_{k,jrow} >^{\lambda} = \frac{1}{Volx^U_{k,jrow}} \sum_{i=2}^{imt-1} u_{i,k,j,2,\tau} \Delta^U_i \tag{18.31}$$

$$< \overline{T}^{\phi}_{i,jrow} >^{z} = \frac{1}{Volz^T_{i,jrow}} \sum_{k=1}^{km} \overline{t_{i,k,j,n,\tau}}^{\phi} \Delta^T_k \tag{18.32}$$

$$< V_{i,jrow} >^{z} = \frac{1}{Volz^U_{i,jrow}} \sum_{k=1}^{km} adv\_vnt_{i,k,j} \Delta^U_k \tag{18.33}$$

Note that a factor of $\cos\phi^U_{jrow}$is built into $adv\_vnt_{i,k,j}$. The volume elements and volume of the latitude strip as a function of depth are

$$\Delta^T_i = tmask_{i,k,j} \cdot tmask_{i,k,j+1} \cdot dxt_i \tag{18.34}$$

$$\Delta^U_i = dxui \cdot \cos\phi^U_{jrow} \tag{18.35}$$

$$\Delta^T_k = tmask_{i,k,j} \cdot tmask_{i,k,j+1} \cdot dzt_k \tag{18.36}$$

$$\Delta^U_k = dzt_k \tag{18.37}$$

$$Volx^T_{k,jrow} = \sum_{i=2}^{imt-1} \Delta^T_i \tag{18.38}$$

$$Volx^U_{k,jrow} = \sum_{i=2}^{imt-1} \Delta^U_i \tag{18.39}$$

$$Volz^T_{i,jrow} = \sum_{k=1}^{km} \Delta^T_k \tag{18.40}$$

$$Volz^U_{i,jrow} = \sum_{k=1}^{km} \Delta^U_k \tag{18.41}$$

$$\tag{18.42}$$

The canonical form of the northward components of tracer transport by various means and deviations is given as

$$ttn_{1,jrow,n} = \sum_{k=1}^{km} < \overline{T}^{\phi}_{k,jrow} >^{\lambda} \cdot < V_{k,jrow} >^{\lambda} \cdot dzt_k \tag{18.43}$$

$$ttn_{2,jrow,n} = ttn_{6,jrow,n} - ttn_{1,jrow,n} \tag{18.44}$$

$$ttn_{3,jrow,n} = \sum_{i=2}^{imt-1} < \overline{T}^{\phi}_{i,jrow} >^{z} \cdot < V_{i,jrow} >^{z} \cdot dxt_i \tag{18.45}$$

$$
\begin{aligned}
ttn_{4,jrow,n} &= ttn_{6,jrow,n} - ttn_{3,jrow,n} - ttn_{5,jrow,n} && (18.46)\\
ttn_{5,jrow,n} &= \sum_{i=2}^{imt-1} -(\overline{smf_{i-1,j,1} \cdot dxu_{i-1}}^{\lambda}) \cdot (\overline{t_{i,1,j,n,\tau}}^{\phi} - < \overline{T}^{\phi}_{i,jrow} >^{z}) \cdot \frac{\cos\phi^U_{jrow}}{f_{jrow}} && (18.47)\\
ttn_{6,jrow,n} &= \sum_{k=1}^{km}\sum_{i=2}^{imt-1} 0.5 \cdot adv\_fn_{i,k,j} \cdot \Delta^T_i \cdot dzt_k && (18.48)\\
ttn_{7,jrow,n} &= \sum_{k=1}^{km}\sum_{i=2}^{imt-1} diff\_fn_{i,k,j} \cdot \Delta^T_i \cdot dzt_k && (18.49)\\
ttn_{8,jrow,n} &= ttn_{6,jrow,n} + ttn_{7,jrow,n} && (18.50)
\end{aligned}
$$

Note the factor of 0.5 which is needed to correct the advective flux of tracer as described in Section 11.10.2. These terms may also be broken down as a function of latitude within $mskhr_{i,jrow}$.

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *gyre_comp.dta*. If option *netcdf* or *gyre_components_netcdf* is enabled, data is written in NetCDF format to file *gyre_comp.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *gyreint* and the control is specified by variable *iogyre*.

18.2.7 matrix_sections

The purpose of option *matrix_sections* is mainly for debugging. It is useful when trying to look at numbers as the model integrates. Various quantities are printed in matrix form as a function of longitude x and depth z for specific latitudes. The latitudes and ranges of longitudes and depths for limiting the printout are input through namelist. Refer to Section 5.4 for information on namelist variables. The output from this diagnostic is written as ascii to the model *printout* file and there is no NetCDF capability. The interval between output is specified by variable *prxzint*.

18.2.8 meridional_overturning

Option *meridional_overturning* computes instantaneous values of a meridional mass transport stream function in units of cm^3/sec. This is useful in determining aspects of the thermohaline circulation.

$$vmsf_{jrow,k} = \sum_{m=1}^{k}\sum_{i=2}^{imt-1} u_{i,m,j,2,\tau} \cdot \cos\phi^U_{jrow} dxu_i \; dzt_m \qquad (18.51)$$

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *overturn.dta*. If option *netcdf* or *meridional_overturning_netcdf* is enabled, data is written in NetCDF format to file *overturn.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *vmsfint* and the control is specified by variable *iovmsf*.

18.2.9 meridional_tracer_budget

What are the dominant meridional balances (if any) in the model and how do they depend on time? Option *meridional_tracer_budget* contracts the tracer equation into a one dimensional equation in latitude by averaging over longitude and depth cells. Using operators described in Section 11.10.7, this yields

$$\frac{1}{Vol^T_{jrow}} \sum_{i=2}^{imt-1} \sum_{k=1}^{km} \Delta^T_{i,k,jrow} \quad \cdot \quad \Big[\delta_t(T_{i,k,j,n,\tau}) + ADV\_Tx_{i,k,j} + ADV\_Ty_{i,k,j} + ADV\_Tz_{i,k,j} = DIFF\_Tx_{i,k,j} + DIFF\_Ty_{i,k,j} + DIFF\_Tz_{i,k,j} + source_{i,k,j}\Big] \tag{18.52}$$

where n is the tracer and the relation between j and $jrow$ is as described in Section 5.2. The volume element and total volume as a function of latitude are given by

$$\Delta^T_{i,k,jrow} = tmask_{i,k,jrow} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{18.53}$$

$$Vol^T_{jrow} = \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow} \tag{18.54}$$

The quantity $tmask_{i,k,jrow}$ is 1 for ocean cells and 0 for land cells. Applying boundary conditions in depth and longitude to Equation (18.52) yields

$$< \delta_t(T_{i,k,j,n,\tau}) >^{\lambda,z} + < ADV\_Ty_{i,k,j} >^{\lambda,z} = < stf_{i,j,n} >^{\lambda} + < DIFF\_Ty_{i,k,j} >^{\lambda,z} + < source_{i,k,j} >^{\lambda,z} \tag{18.55}$$

where $< \; >^{\lambda,z}$ indicates an average over all λ and z. Each term in Equation (18.55) is then averaged in time to produce stable estimates which can indicate the dominant meridional balances as a function of time. The meridional tracer equation becomes

$$\frac{1}{L}\sum_{\ell=1}^{L}\Big[< \delta_t(T_{i,k,j,n,\tau}) >^{\lambda,z} + < ADV\_Ty_{i,k,j} >^{\lambda,z} = < stfi,j,n >^{\lambda} + < DIFF\_Ty_{i,k,j} >^{\lambda,z} + < source_{i,k,j} >^{\lambda,z}\Big] \tag{18.56}$$

where ℓ is the time step counter and L is the number of time steps in the averaging period described below. The individual terms in Equation (18.56) are given as

$$tstor_{jrow,n} = \frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol^T_{jrow}}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\frac{t_{i,k,j,n,\tau+1} - t_{i,k,j,n,\tau-1}}{2\Delta\tau}\Delta^T_{i,k,jrow}\Big] \tag{18.57}$$

$$tdiv_{jrow,n} = -\frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol^T_{jrow}}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} ADV\_Ty_{i,k,j} \cdot \Delta^T_{i,k,jrow}\Big] \tag{18.58}$$

$$tflux_{jrow,n} = -\frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Area^T_{jrow}}\sum_{i=2}^{imt-1} stf_{i,j,n} \cdot A^T_{i,jrow}\Big] \tag{18.59}$$

$$tsorc_{jrow,n} = \frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol^T_{jrow}}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} source_{i,k,j} \cdot \Delta^T_{i,k,jrow}\Big] \tag{18.60}$$

$$tdif_{jrow,n} = -\frac{1}{L}\sum_{\ell=1}^{L}\Big[\frac{1}{Vol^T_{jrow}}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} DIFF\_Ty_{i,k,j} \cdot \Delta^T_{i,k,jrow}\Big] \tag{18.61}$$

where the area element and total area of a latitude are given by

$$A^T_{i,jrow} = tmask_{i,1,jrow} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \tag{18.62}$$

$$Area^T_{jrow} = \sum_{i=2}^{imt-1} A^T_{i,jrow} \tag{18.63}$$

$$\tag{18.64}$$

The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *tracer_bud.dta*. If option *netcdf* or *meridional_tracer_budget_netcdf* is enabled, data is written in NetCDF format to file *tracer_bud.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *tmbint* and the control is specified by variable *iotmb*. The averaging period is typically the same as the interval but may be specified shorter by variable *tmbper*. These variables are input through namelist. Refer to Section 5.4 for information on namelist variables. Note further that this dataset may be contracted in latitude and region space to yield a zero dimensional model which describes the total heat content of one tremendously gigantic cell as a function of time.

$$< tstor_{jrow,n} >^{\phi} = \frac{1}{Area^T} \sum_{jrow=2}^{jmt-1} \left[tflux_{jrow,n} \right] \Delta^T_{jrow} \tag{18.65}$$

where the area element is

$$Area^T = \sum_{jrow=2}^{jmt-1} \Delta^T_{jrow} \tag{18.66}$$

However, this can only be done if the averaging period equals the interval for writing the output. But, when the averaging period equals the sampling interval this diagnostic is a significant time burner. For most cases, it may be adequate to specify a short averaging period at the end of the sampling interval (e.g. in the limit, a one time step average at the end of every sampling interval.). The terms in the above analysis may be further decomposed into regions based on mask $msktmb_{i,jrow}$ along the lines of Section 18.1.4. However, $msktmb_{i,jrow}$ =1 and it is left to the researcher to define other regions if desired. Note that the regional mask $msktmb_{i,jrow}$ is also written to file *tracer_bud.dta* when the initialization boolean *itmb* is true. It should be set true on the first run but false thereafter.

18.2.10 netcdf

When enabled, this option saves diagnostic output in a NetCDF format instead of unformatted 32 bit IEEE format. This is a catch all option which applies to <u>all</u> enabled diagnostics. Many diagnostics also have an option for enabling NetCDF format independent of other diagnostics. Refer to the particular diagnostic for further information. For information on using double precision on systems with 32 bit word length, refer to Sections 19.11 and 19.12. <u>Note</u> that early versions of the NetCDF libraries at GFDL were very inefficient. Newer versions reduce previous wall clock time by over two orders of magnitude! Routines used in MOM 2 require use of the UNIDATA NetCDF library. If not already in place, this will have to be built using information available from http://www.unidata.ucar.edu

Note that the directory MOM_2/NETCDF contains utility routines written by John Sheldon at GFDL (for purposes other than MOM 2) to provide an intermediate level interface to the low level "netcdf" library routines. Also, the file *netcdf.inc* within MOM_2/NETCDF is platform specific. At GFDL, this file is referenced by pathname "/usr/local/include/netcdf.inc". If platforms other than these are used, get the proper "netcdf.inc" from the address given above. Typically, routines in MOM/NETCDF do not have to be explicitly used (or understood) when writing diagnostics in MOM 2. There are higher level interface routines in file *util_netcdf.F* which are described below and used to facilitate writing diagnostic output in NetCDF format. All diagnostics within MOM 2 use these higher level interface routines to provide uniformity between diagnostics and to minimize the amount of coding needed in what otherwise would be a long tedious process. Once the idea is grasped, any diagnostic can be used as a template to add NetCDF format capability to new diagnostics.

Routines within directory MOM_2/NETCDF need to know which computer platform they are being used on. This information is supplied through preprocessor *ifdef* options as in MOM 2. Scripts *run_mom* and *run_mom_sgi* set the computer platform option and (based on this choice) adds options needed for routines within directory MOM_2/NETCDF.

Using the high level interface

In general, each NetCDF file needs dimensional information indicating the total number of variables to be written to the NetCDF file, the total number of axes used, and the size of the longest axis. For instance, if arrays A(imt,jmt) and B(imt,jmt) were to be written periodically in time, then the total number of variables would be two, the total number of axes would be three (longitude, latitude, and time), and the longest axes would be either the first or second depending on the sizes of *imt* and *jmt*. The time axis is infinitely extendible and doesn't count in this respect. Typically, this information is prescribed for each diagnostic by the following code:

```
parameter (ndimsout=4, nvarsout = 2, mxdimout = max(imt,jmt,km))
```

where *ndimsout* is the total number of axes to be used (longitude, latitude, depth, time, etc), *nvarsout* is the total number of variables to be written, and *mxdimout* is the size of the longest axis (imt, jmt, km). If a mistake is made in defining any of the above parameters, an error message will point back to the offending parameter with a simple traceback. Basically three things are needed to prepare for writing each variable to a NetCDF data file.

1. The axes information must be defined by making a call to routine *def_axis* which temporarily stores the specified information. One call to *def_axis* is required for each axis. When all axes have been defined, definitions are retrieved by making one call to routine *get_axes* for each call to routine *def_axis*. This is typically done by one compact loop. Ordering is important. Retrieval must be done in the same order as definitions.

2. All variables are defined by a set of properties (name, units, etc). Properties of each variable must be specified using a call to routine *def_var* which temporarily stores the definition. After all variables have been defined, the definitions are retrieved by making one call to routine *get_def* for each call to routine *def_var*. Again, this is done in one compact loop. Ordering is important. Retrieval must be done in the same order as definitions.

3. The NetCDF file is opened with the specified information for variables and axes by a call to routine *ncsetup*. This need be done only once. Subsequent appending of data to the file only requires a re-opening.

The above three steps need only be done once per diagnostic per run. When writing, variables must be written to the NetCDF file in the same order as they were defined. This is done by one call to routine *ncput* for each variable. After all variables have been written, the NetCDF file is closed. Subsequently, it may be reopened by a call to routine *ncreo1* and appended to.

As an example, consider the *save_convection* diagnostic. The following code defines four axes (longitude,latitude,depth,time) and numbers them as (1,2,3,4).

```
      num  = 0
      call caller_id ('conv_netcdf')
      call def_axis (1, num, 'X', +1, 'xt_i', 'Longitude of T points'
  &,                  'degrees_E', xt, imt, dimvals, lendims,
  mxdimout, ndimsout)

      call def_axis (2, num, 'Y', +1, 'yt_j', 'Latitude of T points'
  &,                  'degrees_N', yt, jmt, dimvals, lendims,
  mxdimout, ndimsout)

      call def_axis (3, num, 'Z', -1, 'zt_k', 'Depth of T points'
  &,                  'cm', zt, km, dimvals, lendims, mxdimout,
  ndimsout)

      call timestr_netcdf (time_since)
      call def_axis (4, num, 'T', +1,'Time','Time since initial cond'
  &,                  'years', 0.0, 0, dimvals, lendims,
  mxdimout, ndimsout)
```

If additional axes were required, they could be added. The arguments for the call to *def_axis* are: a number to associate with the axis, a counter for checking consistency, a label, a direction for which way is positive, a short name, a long name, units, an array defining points along the axis, the number of points along the axis, an internal netcdf array which will be set within *def_axis*, an internal netcdf variable which will be set within *def_axis*, max length of any axis, and the maximum number of axes. The internal netcdf variables are set by the call to *get_axis*. Finish up by getting the definition of axes into the proper NetCDF variables with the following:

```
      do n=1,num
        call get_axis (n, cart_axis(n), ipositive(n)
  &,                   cdimnam(n), cdimlnam(n), cdimunits(n))
      enddo
```

This particular diagnostic requires writing two variables periodically and their properties must be uniquely defined. The first variable (named *convect*) will be defined to be organized in the NetCDF file by axes '1234'. Given the above definition of axes, this means that NetCDF views the data as if it were dimensioned by (x,y,z,t). The second variable (named *period*) is defined on axis '4' so is only a function of time. Actually, this diagnostic only outputs instantaneous data, so the period is zero. The following code accomplishes the definition of properties for these two variables.

```
      num = 0
      call def_var (num, 'convect', 'Rate of convection', 'dec C/sec'
     &,                 '1234',  0.0, 1.e6, .false., 0.0,  32, nvarsout)

      call def_var (num, 'period','Averaging period','days'
     &,                 '4', 0.0, 1.e20, .false., 0.0, 32, nvarsout)
```

The arguments for *def_var* are: a variable for consistency checking, a short name, a long name, units, character string indicating which axes contain data, smallest valid number for the data, largest valid number for the data, logical for special value, special value used to signify missing data, number of bits used for precision for packing purposes, and maximum allowable number of variables. The following call to *get_def* retrieves these properties and sets the appropriate internal netcdf variables.

```
      do n=1,num
        call get_def (n, cvarnam(n), cvarlnam(n)
     &,                 cvarunits(n),  nvdims(n), idimindx(1,n)
     &,                 validr(1,n), lspval(n), vspval(n), nbits(n))
      enddo
```

All of the above information is encapsulated into a NetCDF file named *fname* using the following call to *ncseti* which calls the MOM_2/NETCDF routine *ncsetup*. Routine *ncseti* is an interface routine provided to allow MOM 2 to be compiled in double precision on workstations. It then converts data to single precision which is expected by the Netcdf routines. Refer to Section 19.11.

```
      iverbose = 0
      icaltype = 0
      call  ncseti (fname, lclobber, gtitle, lgspval, gspval
     &,             ndimsout, lendims, cdimnam, cdimlnam
     &,             cdimunits, cart_axis, ipositive
     &,             dimvals, mxdimout, time
     &,             nvarsout, cvarnam, cvarlnam, cvarunits, nbits
     &,             nvdims, idimindx, ndimsout
     &,             validr, lspval, vspval, iverbose)
```

Note that *ncseti* need only be called once for each diagnostic. Afterwards, all properties of the data are uniquely defined. Filename *fname* is now ready to receive data. Assume array *excnv1* is dimensioned by (imt,km) and is filled with convection data for every *jrow*. The following lines store the data from *excnv1* into latitude row *jrow* in the NetCDF file.

```
c     set the starting indices and length

      istart(1) = 1
      icount(1) = imt
      istart(2) = jrow
      icount(2) = 1
```

```
istart(3) = 1
icount(3) = km
istart(4) = num_conv
icount(4) = 1

num_var = 0
call ncput (num_var, istart, icount, excnv1(1,1), timrec)

period    = 0.0
istart(1) = num_conv
icount(1) = 1
call ncput (num_var, istart, icount, period, timrec)

call release_netcdf ('conv_netcdf', num, num_var)
```

When actually writing data, information stating how much data is to be written along each axis must be given. In the above code, *istart*(n) is the starting location for axis n in the NetCDF file and *icount*(n) is how many elements are to be written along that axis starting at location *istart*(n) from array *excnv1*. Note that array *excnv1* is dimensioned (imt,km) in memory but the NetCDF data looks as if it were dimensioned (imt,jmt,km,time). Each write accounts for a sub-section in this four dimensional space. Variable *num_conv* is the index for the time axis and represents the number of times this diagnostic was called while variable *timrec* is the corresponding time in years. After all data is written, the file is released.

Diagnostic output is typically written periodically. Having been set up once, there is no future need to re-specify the setup information by calling *ncsetup*. The file need only be re-opened to be appended.

After collecting a number of NetCDF formatted diagnostic files which have been written by separate jobs, the files cannot be concatenated with the UNIX *cat* command. Rather, the NetCDF *nccat* utility is needed.

18.2.11 save_convection

When convection is being computed explicitly (option *implicitvmix* is <u>not</u> enabled), the instantaneous value of the component of the time rate of change of temperature due to explicit convection can be saved three dimensionally. If cell $T_{i,k,jrow}$is a land cell, its value is set to a flag value $= 10^{-20}$ to denote land. If no convection has taken place in cell $T_{i,k,jrow}$, the convection for that cell is zero. Otherwise

$$convect_{i,k,j} = \frac{t_{i,k,j,1,\tau+1}^{after\ convection} - t_{i,k,j,1,\tau+1}^{before\ convection}}{2\Delta\tau} \tag{18.67}$$

The output from this diagnostic is written only as 32 bit IEEE unformatted data to file *cvct.dta*. If option *netcdf* or *save_convection_netcdf* is enabled, data is written in NetCDF format to file *cvct.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *exconvint* and there is no control variable since only the unformatted data is written.

18.2.12 save_mixing_coeff

Much of the physics in MOM 2 is distilled into mixing coefficients. Mixing coefficients can be computed in a variety of ways depending on the combinations of horizontal and vertical subgrid scale mixing parameterization options which have been enabled. In some cases, the mixing coefficients are not explicitly computed. However, the flux across faces of cells is always computed. Option *save_mixing_coeff* estimates mixing coefficients from flux across cell faces. For momentum, the coefficients on east, north, and bottom faces of U cells is estimated by

$$ce_{i,k,j,1} = \frac{diff\_fe_{i,k,j}}{\frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(u_{i,k,j,1,\tau-1}) + \epsilon} \tag{18.68}$$

$$cn_{i,k,j,1} = \frac{diff\_fn_{i,k,j}}{\delta_\phi(u_{i,k,j,1,\tau-1}) + \epsilon} \tag{18.69}$$

$$cb_{i,k,j,1} = \frac{diff\_fb_{i,k,j}}{\delta_z(u_{i,k,j,1,\tau-1}) + \epsilon} \tag{18.70}$$

where $\epsilon = 10^{-20}$ to keep from dividing by zero where no gradient in velocity exists. For tracers, the coefficients on east, north, and bottom faces of T cells is estimated by

$$ce_{i,k,j,2} = \frac{diff\_fe_{i,k,j}}{\frac{1}{\cos\phi^U_{jrow}}\delta_\lambda(t_{i,k,j,1,\tau-1}) + \epsilon} \tag{18.71}$$

$$cn_{i,k,j,2} = \frac{diff\_fn_{i,k,j}}{\delta_\phi(t_{i,k,j,1,\tau-1}) + \epsilon} \tag{18.72}$$

$$cb_{i,k,j,2} = \frac{diff\_fb_{i,k,j}}{\delta_z(t_{i,k,j,1,\tau-1}) + \epsilon} \tag{18.73}$$

where ϵ plays the same role as for velocities. If option *isopycmix* is enabled, the $K11_{i,k,j}$, $K22_{i,k,j}$, $K33_{i,k,j}$ elements of the isopycnal mixing tensor along with the suitably averaged mixing coefficients $A^{ez}_{i,k,j}$, $A^{nz}_{i,k,j}$, $A^{bx}_{i,k,j}$, and $A^{by}_{i,k,j}$ are additionally output.

The output from this diagnostic is written as 32 bit IEEE unformatted data to file *cmix.dta*. If option *netcdf* or *save_mixing_coeff_netcdf* is enabled, data is written in NetCDF format to file *cmix.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *cmixint* and there is no control variable since only the unformatted data is written.

18.2.13 show_external_mode

Option *show_external_mode* saves instantaneous values (at time level $\tau + 1$) of either the stream function, prognostic surface pressure, or implicit free surface depending on which option is enabled. Stream function is in units of cm^3/sec, surface pressure and implicit free surface is in units of $gram/cm/sec^2$. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *psi.dta*, *surf_press.dta*, or *ifree_surf.dta*. If option *netcdf* or *show_external_mode_netcdf* is enabled, data is written in NetCDF format to file *psi.dta.nc*, *surf_press.dta.nc*, or *ifree_surf.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *extint* and the control is specified by variable *ioext*.

18.2.14 show_zonal_mean_of_sbc

Option *show_zonal_mean_of_sbc* saves instantaneous zonal means of all surface boundary conditions as a function of latitude. This is useful to verify that the surface boundary conditions are reasonable, at least in the zonal mean sense.

$$zmsm_{jrow,n} = \frac{1}{Vol^U_{jrow}} \sum_{i=2}^{imt-1} u_{i,1,j,n,\tau} \cdot \Delta^U_{i,jrow} \tag{18.74}$$

$$zmsmf_{jrow,n} = \frac{1}{Vol^U_{jrow}} \sum_{i=2}^{imt-1} smf_{i,j,n} \cdot \Delta^U_{i,jrow} \tag{18.75}$$

$$zmst_{jrow,n} = \frac{1}{Vol^T_{jrow}} \sum_{i=2}^{imt-1} t_{i,1,j,n,\tau} \cdot \Delta^T_{i,jrow} \tag{18.76}$$

$$zmstf_{jrow,n} = \frac{1}{Vol^T_{jrow}} \sum_{i=2}^{imt-1} stf_{i,j,n} \cdot \Delta^T_{i,jrow} \tag{18.77}$$

where the volume elements and total volumes on U cells and T cells are

$$\Delta^U_{i,jrow} = umask_{i,1,jrow} \cdot dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{18.78}$$

$$\Delta^T_{i,jrow} = tmask_{i,1,jrow} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{18.79}$$

$$Vol^U_{jrow} = \sum_{i=2}^{imt-1} \Delta^U_{i,jrow} \tag{18.80}$$

$$Vol^T_{jrow} = \sum_{i=2}^{imt-1} \Delta^T_{i,jrow} \tag{18.81}$$

and the relation between *j* and *jrow* is as described in Section 5.2. Surface heat flux is converted to $watts/m^2$, windstress is in $dyne/cm^2$ precip minus evaporation is in mm/day, velocity is in cm/sec, temperature is in $degC$ and salinity is in $ppt - 35.0$. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *zmean_sbc.dta*. If option *netcdf* or *show_zonal_mean_of_sbc_netcdf* is enabled, data is written in NetCDF format to file *zmean_sbc.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *zmbcint* and the control is specified by variable *iozmbc*.

18.2.15 snapshots

Option *snapshots* saves <u>instantaneous</u> values of prognostic and associated variables. The variables are tracers $t_{i,k,j,n,\tau}$ for $n = 1, nt$, horizontal velocities $u_{i,k,j,n,\tau}$ for $n = 1, 2$, vertical velocity at the base of T cells $adv\_vbt_{i,k,j}$, surface tracer flux $stf_{i,j,n}$ for $n = 1, nt$, surface momentum flux $smf_{i,j,n}$ for $n = 1, 2$, and the external mode which is given by either[6] $psi_{i,jrow,\tau}$ or $ps_{i,jrow,\tau}$.

This output data may be restricted to certain contiguous latitude and depth ranges using variables input through namelist. Refer to Section 5.4 for information on namelist variables. Note that on time step = itt (which corresponds to $\tau + 1$), data is written from time level τ rather than $\tau + 1$ because the $\tau + 1$ external mode is unknown at the time when data is written. Therefore, $u_{i,k,j,n,\tau}$ is a total velocity containing both internal and external modes. The output from this diagnostic is written as 32 bit IEEE unformatted data to file *snapshots.dta*. If option *netcdf* or *snapshots_netcdf* is enabled, data is written in NetCDF format to file *snapshots.dta.nc* rather than in unformatted IEEE. The structure of this file is very similar to the one from option *time_averages*. The interval between output is specified by variable *snapint*.

[6] Depending on whether option *stream_function*, *rigid_lid_surface_pressure* or *implicit_free_surface* is enabled.

18.2.16 stability_tests

Option *stability_tests* computes various stability criteria and related items within a portion or all of the model domain. The limits of volume of domain to be considered when testing are set through namelist. Refer to Section 5.4 for information on namelist variables. If MOM 2 blows up, this diagnostic is useful in finding where it went unstable. The following are computed:

1. Based on local velocities within each cell, a maximum local time step is computed for the three principal directions as

$$\Delta\tau^x_{i,k,j} = \frac{\cos\phi^U_{jrow} \cdot dxu_i}{2 \cdot u_{i,k,j,1,\tau}} \tag{18.82}$$

$$\Delta\tau^y_{i,k,j} = \frac{dyu_{jrow}}{2 \cdot u_{i,k,j,2,\tau}} \tag{18.83}$$

$$\Delta\tau^z_{i,k,j} = \frac{dzw_k}{2 \cdot adv\_vbt_{i,k,j}} \tag{18.84}$$

   These local time steps are compared with the model specified time step and the location of the largest is chosen as the position of the most unstable cell. If the local time step exceeds the model time step by an amount which can be set through namelist, then a CFL violation is detected. The number of times a CFL violation is allowed may also be set through namelist. Variables within the local neighborhood of the offending cells are shown and when the number of offenses exceeds the allowable number, the model is brought down. Refer to Section 5.4 for choosing time step lengths and setting a region over which stability calculations are performed. The default region is the entire domain.

2. The local Reynolds number is estimated along each of the principle directions as

$$ray^x_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot dxu_i}{visc\_ceu_{i,k,j}} \tag{18.85}$$

$$ray^y_{i,k,j} = \frac{u_{i,k,j,2,\tau} \cdot dyu_{jrow}}{visc\_cnu_{i,k,j}} \tag{18.86}$$

$$ray^z_{i,k,j} = \frac{adv\_vbu_{i,k,j} \cdot dzw_k}{visc\_cbu_{i,k,j}} \tag{18.87}$$

   and the location of the maximum is found and printed.

3. The local Peclet number is estimated along each of the principle directions as

$$pecl^x_{i,k,j} = \frac{u_{i,k,j,1,\tau} \cdot dxu_i}{diff\_cet_{i,k,j}} \tag{18.88}$$

$$pecl^y_{i,k,j} = \frac{u_{i,k,j,2,\tau} \cdot dyu_{jrow}}{diff\_cnt_{i,k,j}} \tag{18.89}$$

$$pecl^z_{i,k,j} = \frac{adv\_vbt_{i,k,j} \cdot dzw_k}{diff\_cnt_{i,k,j}} \tag{18.90}$$

   and the location of the maximum is found and printed.

4. The locations where numerics are breaking down and producing spurious tracer extrema are determined. This is done by searching the immediate neighborhood of cell (i, k, j) for extrema in temperature at τ and $\tau - 1$. If $t_{i,k,j,1,\tau+1}$ exceeds this extrema by an amount which may be specified through namelist, then there is numerical truncation at (i, k, j). Note that this statement can only be made because of the incompressibility condition. If there are more than 100 locations exhibiting numerical truncation, only the first 100 locations are shown.

5. The locations are shown where predicted temperatures or salinities are outside the bounds of temperatures and salinities which were used for the construction of density coefficients. If there are more than 100 locations where this occurs, only the first 100 locations are shown.

6. The location of maximum error in continuity is calculated considering all U cells and T cells separately.

7. The maximum error in vertical velocity at the ocean bottom on T cells is computed. This is the residual error from integrating Equation (11.32) vertically from the surface to the ocean bottom.

8. The maximum vertical velocity at the ocean bottom on U cells from Equation (11.38) is computed. This is non-zero if the bottom has a slope since the bottom flow is required to parallel the bottom slope.

As described above, if more than a specified number of CFL violations are found when this diagnostic is active (only at times specified by the interval), the integration will stop indicating where the most unstable locations are with matrix printouts of variables in the neighborhoods. If a violation is not found, statistics will be printed indicating how close the integration is to violating the CFL condition along with other information described above. The output from this diagnostic may only be written as ascii to the model *printout*. The interval between output is specified by variable *stabint*.

18.2.17 term_balances

Option *term_balances* constructs <u>instantaneous</u> spatial averages of all terms[7] in the prognostic equations over arbitrary regional volumes defined by horizontal and vertical region masks as described in Section 18.1.4. Regional volumes may be set up as indicated by the test case example in file *setocn.F*. They may not overlap one another. This diagnostic is useful when trying to understand the dominant balances within regional volumes of the model domain. Be aware that aliasing may occur because results are <u>not</u> averaged in time. This diagnostic is relatively slow and can appreciably degrade the speed of MOM 2. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *term_bal.dta*. If option *netcdf* or *term_balances_netcdf* is enabled, data is written in NetCDF format to file *term_bal.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *trmbint* and the control is specified by variable *iotrmb*. Note that the regional masks $mskhr_{i,jrow}$and $mskvr_k$are also written to file *term_bal.dta* when the initialization boolean *itrmb* is true. It should be set true on the first run but false thereafter.

Partial sums are taken over all *(i,k,jrow)* within the domain to contract quantities into regional volumes given as

[7] Plus a few extra ones.

$$nreg = (mskvr_k - 1) \cdot nhreg + mskhr_{i,jrow} \tag{18.91}$$

where masks $mskvr_k$and $mskhr_{i,jrow}$are defined as in Section 18.1.4 and the range of $nreg$ is from 0 to $nhreg \cdot nvreg$. Two operators are defined to perform these contractions separately for quantities defined on T cells and U cells. They are given by

$$\overline{\alpha_{i,k,j}}^{\mathcal{U}(nreg)} = \frac{1}{Vol^{\mathcal{U}(nreg)}} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \alpha_{i,k,j} \cdot \Delta^U_{i,k,jrow} \tag{18.92}$$

$$\overline{\beta_{i,k,j}}^{\mathcal{T}(nreg)} = \frac{1}{Vol^{\mathcal{T}(nreg)}} \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \sum_{n=1}^{2} \beta_{i,k,j} \cdot \Delta^T_{i,k,jrow} \tag{18.93}$$

$$\tag{18.94}$$

where $\alpha_{i,k,j}$ is defined on U cells, $\beta_{i,k,j}$ is defined on T cells, and the respective volume elements and total volumes for each region are

$$\Delta^U_{i,k,jrow} = umask_{i,k,jrow} \cdot dxu_i \cdot \cos\phi^U_{jrow} \cdot dyu_{jrow} \cdot dzt_k \tag{18.95}$$

$$Vol^{\mathcal{U}(nreg)} = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^U_{i,k,jrow} \tag{18.96}$$

$$Vol^{\mathcal{U}(0)} = \sum_{nreg=1}^{nhreg} Vol^{\mathcal{U}(nreg)} \tag{18.97}$$

$$\Delta^T_{i,k,jrow} = tmask_{i,k,jrow} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \cdot dzt_k \tag{18.98}$$

$$Vol^{\mathcal{T}(nreg)} = \sum_{jrow=2}^{jmt-1} \sum_{k=1}^{km} \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow} \tag{18.99}$$

$$Vol^{\mathcal{T}(0)} = \sum_{nreg=1}^{nhreg} Vol^{\mathcal{T}(nreg)} \tag{18.100}$$

with the relation between j and $jrow$ as described in Section 5.2.

Term balance for Momentum Equations

Using arrays and operators described in Section 11.11.5, all components of the momentum Equation are contracted into regional volumes with the canonical forms given below. One way to think of this is that all cells within a regional volume $\mathcal{U}(nreg)$are replaced by one gigantic U cell and the value of all terms in the momentum equation are given for this one cell. Subscript $n = 1$ refers to the zonal velocity component and $n = 2$ refers to the meridional component.

$$termbm_{1,n,nreg} = \overline{\frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{\mathcal{U}(nreg)} \tag{18.101}$$

$$termbm_{2,n,nreg} = \overline{-grad\_p_{i,k,j,n}}^{\mathcal{U}(nreg)} \tag{18.102}$$

$$termbm_{3,n,nreg} = \overline{-ADV\_Ux_{i,k,j}}^{\mathcal{U}(nreg)} \tag{18.103}$$

$$termbm_{4,n,nreg} = \overline{-ADV\_Uy_{i,k,j}}^{\mathcal{U}(nreg)} \tag{18.104}$$

$$\begin{aligned}
termbm_{5,n,nreg} &= \overline{-ADV_Uz_{i,k,j}}^{\mathcal{U}(nreg)} && (18.105)\\
termbm_{6,n,nreg} &= \overline{DIFF_Ux_{i,k,j}}^{\mathcal{U}(nreg)} && (18.106)\\
termbm_{7,n,nreg} &= \overline{DIFF_Uy_{i,k,j}}^{\mathcal{U}(nreg)} && (18.107)\\
termbm_{8,n,nreg} &= \overline{DIFF_Uz_{i,k,j}}^{\mathcal{U}(nreg)} && (18.108)\\
termbm_{9,n,nreg} &= \overline{DIFF_metric_{i,k,j,n}}^{\mathcal{U}(nreg)} && (18.109)\\
termbm_{10,n,nreg} &= \overline{CORIOLIS_{i,k,j,n}}^{\mathcal{U}(nreg)} && (18.110)\\
termbm_{11,n,nreg} &= \overline{source_{i,k,j}}^{\mathcal{U}(nreg)} && (18.111)\\
termbm_{12,n,nreg} &= \overline{-grad_p^s_{i,k,j,n}}^{\mathcal{U}(nreg)} && (18.112)\\
termbm_{13,n,nreg} &= \overline{ADV_metric_{i,k,j,n}}^{\mathcal{U}(nreg)} && (18.113)\\
& && (18.114)
\end{aligned}$$

In Equation (18.112), the quantity $grad\_p^s_{i,k,j,n}$ is calculated from Equations (18.13) and (18.14). The equation for the gigantic grid cell in each regional volume $\mathcal{U}(nreg)$is given by

$$termbm_{1,n,nreg} = \sum_{\ell=2}^{13} termbm_{\ell,n,nreg} \tag{18.115}$$

Note that terms 3,4, and 5 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the true advection in λ, ϕ, z because they contain divergent components. The true advection in λ, ϕ, z is given below by terms 14,15, and 16.

$$\begin{aligned}
termbm_{14,n,nreg} &= \overline{u_{i,k,j,n,\tau} \cdot \frac{adv_veu_{i,k,j} - adv_veu_{i-1,k,j}}{dxu_i \cdot \cos\phi^U_{jrow}} - ADV_Ux_{i,k,j}}^{\mathcal{U}(nreg)} && (18.116)\\
termbm_{15,n,nreg} &= \overline{u_{i,k,j,n,\tau} \cdot \frac{adv_vnu_{i,k,j} - adv_vnu_{i,k,j-1}}{dyu_{jrow} \cdot \cos\phi^U_{jrow}} - ADV_Uy_{i,k,j}}^{\mathcal{U}(nreg)} && (18.117)\\
termbm_{16,n,nreg} &= \overline{u_{i,k,j,n,\tau} \cdot \frac{adv_vbu_{i,k-1,j} - adv_vnu_{i,k,j}}{dzt_k} - ADV_Uz_{i,k,j}}^{\mathcal{U}(nreg)} && (18.118)\\
termbm_{17,n,nreg} &= \overline{u_{i,k,j,n,\tau}}^{\mathcal{U}(nreg)} && (18.119)\\
avgw_{nreg} &= \overline{\frac{adv_vbu_{i,k-1,j} + adv_vbu_{i,k,j}}{2}}^{\mathcal{U}(nreg)} && (18.120)\\
smflx_{n,nreg} &= \overline{smf_{i,j,n}}^{\mathcal{U}(nreg)} && (18.121)
\end{aligned}$$

Term 17 represents the average horizontal velocity components within the cell and *avgw* is the average vertical component. *smflx* represents the windstress acting on the top of the near surface cells and Equation (18.121) is only averaged over the regional volumes at the ocean surface.

Term balance for Tracer Equation

Using operators and arrays described in Section 11.10.7, all components of the tracer equation are contracted into regional volumes with the canonical forms given below. As with the momentum contraction above, one way to think of this is that all cells within a regional volume $\mathcal{T}(nreg)$are replaced by one gigantic T cell and the value of all terms in the tracer equation are given for this one cell. Subscript $n = 1$ refers to the temperature component and $n = 2$ refers to the salinity.

$$
\begin{aligned}
termbt_{1,n,nreg} &= \overline{\frac{t_{i,k,j,n,\tau+1} - t_{i,k,j,n,\tau-1}}{2\Delta\tau}}^{\mathcal{T}(nreg)} && (18.122)\\
termbt_{2,n,nreg} &= \overline{-ADV\_Tx_{i,k,j}}^{\mathcal{T}(nreg)} && (18.123)\\
termbt_{3,n,nreg} &= \overline{-ADV\_Ty_{i,k,j}}^{\mathcal{T}(nreg)} && (18.124)\\
termbt_{4,n,nreg} &= \overline{-ADV\_Tz_{i,k,j}}^{\mathcal{T}(nreg)} && (18.125)\\
termbt_{5,n,nreg} &= \overline{DIFF\_Tx_{i,k,j}}^{\mathcal{T}(nreg)} && (18.126)\\
termbt_{6,n,nreg} &= \overline{DIFF\_Ty_{i,k,j}}^{\mathcal{T}(nreg)} && (18.127)\\
termbt_{7,n,nreg} &= \overline{DIFF\_Tz_{i,k,j}}^{\mathcal{T}(nreg)} && (18.128)\\
termbt_{8,n,nreg} &= \overline{source_{i,k,j}}^{\mathcal{T}(nreg)} && (18.129)\\
termbt_{9,n,nreg} &= \overline{explicit\ \ convection}^{\mathcal{T}(nreg)} && (18.130)\\
& && (18.131)
\end{aligned}
$$

When options *isopycmix* and *gent_mcwilliams* are enabled, then Equations (18.123), (18.124), and (18.125) also include the flux form of the advection terms from option *gent_mcwilliams* given by $-ADV\_Txiso_{i,k,j}$, $-ADV\_Tyiso_{i,k,j}$, and $-ADV\_Tziso_{i,k,j}$ respectively. In Equation (18.130), the quantity *explicit convection* comes from solving Equation (18.122) before and after explicit convection. The equation for the gigantic grid cell in each regional volume $\mathcal{T}(nreg)$is given by

$$termbt_{1,n,nreg} = \sum_{\ell=2}^{9} termbt_{\ell,n,nreg} \qquad (18.132)$$

Note that terms 2,3, and 4 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the true advection in λ, ϕ, z because they contain divergent components. The canonical form for the true advection in λ, ϕ, z is given below by terms 11,12, and 13.

$$
\begin{aligned}
termbt_{11,n,nreg} &= \overline{t_{i,k,j,n,\tau} \cdot \frac{adv\_vet_{i,k,j} - adv\_vet_{i-1,k,j}}{dxt_i \cdot \cos\phi^T_{jrow}} - ADV\_Tx_{i,k,j}}^{\mathcal{T}(nreg)} && (18.133)\\
termbt_{12,n,nreg} &= \overline{t_{i,k,j,n,\tau} \cdot \frac{adv\_vnt_{i,k,j} - adv\_vnt_{i,k,j-1}}{dyt_{jrow} \cdot \cos\phi^T_{jrow}} - ADV\_Ty_{i,k,j}}^{\mathcal{T}(nreg)} && (18.134)\\
termbt_{13,n,nreg} &= \overline{t_{i,k,j,n,\tau} \cdot \frac{adv\_vbt_{i,k-1,j} - adv\_vnt_{i,k,j}}{dzt_k} - ADV\_Tz_{i,k,j}}^{\mathcal{T}(nreg)} && (18.135)\\
termbt_{15,n,nreg} &= \overline{t_{i,k,j,n,\tau}}^{\mathcal{T}(nreg)} && (18.136)\\
stflx_{n,nreg} &= \overline{stf_{i,j,n}}^{\mathcal{T}(nreg)} && (18.137)
\end{aligned}
$$

When options *isopycmix* and *gent_mcwilliams* are enabled, Equations (18.133), (18.134), and (18.135) also include the advective or transport velocities from option *gent_mcwilliams* given by $adv\_vetiso_{i,k,j}$, $adv\_vntiso_{i,k,j}$, and $adv\_vbtiso_{i,k,j}$ respectively. Term 15 represents the average tracer within the cell and term *stflx* represents the surface tracer flux acting on the top of the surface cells. Equation (18.137) is only averaged over the regional volumes at the ocean surface.

18.2.18 time_averages

Option *time_averages* saves the same variables as does option *snapshots*. However, the variables are time averaged for stability and are defined on an averaging grid which can be the model grid or a sparse subset of the model grid. Often, for analysis purposes, it is not necessary to have data at every grid point or for the entire domain. When this is the case, the size of the archive disk space may be significantly reduced. The variables are tracers $t_{i,k,j,n,\tau}$ for $n = 1, nt$, horizontal velocities $u_{i,k,j,n,\tau}$ for $n = 1, 2$, vertical velocity at the base of T cells $adv\_vbt_{i,k,j}$, surface tracer flux $stf_{i,j,n}$ for $n = 1, nt$, surface momentum flux $smf_{i,j,n}$ for $n = 1, 2$, and the external mode which is given by either[8] $psi_{i,jrow,\tau}$ or $ps_{i,jrow,\tau}$.

Before using this diagnostic, the averaging grid must be constructed by first executing script *run_timeavgs* with option *drive_timeavgs* enabled. This runs the averaging grid generator in a stand alone mode. To define the averaging grid, follow the example in the USER INPUT section of *timeavgs.F*. It is possible to arbitrarily pick which model grid points will be on the averaging grid. However, the code is set to specify a constant spacing between points on the averaging grid in longitude, latitude, and depth. Model grid points nearest to this spacing will be chosen as the averaging grid. To incorporate this averaging grid into MOM 2, follow the directions given when script *run_timeavgs* executes.

The output from this diagnostic is written as 32 bit IEEE unformatted data to file *time_mean.dta*. If option *netcdf* or *time_averages_netcdf* is enabled, data is written in NetCDF format to file *time_mean.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *timavgint*. The averaging period is controlled by variable *timavgper*. Both variables are input through namelist. Refer to Section 5.4 for information on namelist variables. Typically the averaging period is set equal to the interval for output but it may be specified as less than the interval. Refer to Section 18.1.3 for a discussion of when this is appropriate.

If option *time_averages* can produce the same output as option *snapshots* then isn't option *snapshots* redundant? It would be except for the fact that option *time_averages* needs storage space to accumulate data to produce averages. This space is equal to the size of the data being averaged. Typically $5(imt \cdot jmt \cdot km + imt \cdot jmt)$ words are needed. If option *time_averages_disk* is enabled, disk space is used for the storage area, otherwise the storage space is in memory! If memory is insufficient or solid state disk space is insufficient, option *snapshots* is the only way to save the data.

18.2.19 time_step_monitor

Option *time_step_monitor* computes instantaneous values of total kinetic energy[9] per unit volume averaged over the entire domain volume in units of $gm/cm/sec^2$. It also computes first and second moments of tracer quantities. The mean tracer is the first moment and tracer variance[10] is the second moment about the mean in units of tracer squared. Additionally, the quantity

[8] Depending on whether option *stream_function*, *rigid_lid_surface_pressure* or *implicit_free_surface* is enabled.

[9] Neglecting the vertical velocity component on the basis of scale analysis.

[10] Tracer variance is not conserved if explicit convection is active or there is a non zero surface tracer flux. It is also not conserved when diffusion is present. Refer to Appendix A for further discussion.

$|\partial T/\partial t|$ in units of $degC/sec$ (for n=1) and *(grams of salt per grams of water)/sec* (for n=2) averaged over the domain volume is computed. The explicit forms are given by

$$< ke > = \frac{\rho_\circ}{2}\frac{1}{Vol^U}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\sum_{n=1}^{2} u^2_{i,k,j,n,\tau}\cdot\Delta^U_{i,k,jrow} \tag{18.138}$$

$$< t_{i,k,j,n,\tau} > = \frac{1}{Vol^T}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} t_{i,k,j,n,\tau}\cdot\Delta^T_{i,k,jrow} \tag{18.139}$$

$$tracer\_variance = < t^2_{i,k,j,n,\tau} > - < t_{i,k,j,n,\tau} >^2 \tag{18.140}$$

$$< |\delta_\tau(t_{i,k,j,n,\tau})| > = \frac{1}{Vol^T}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\frac{|t_{i,k,j,n,\tau+1}-t_{i,k,j,n,\tau-1}|}{2\Delta\tau}\cdot\Delta^T_{i,k,jrow} \tag{18.141}$$

where

$$< t^2_{i,k,j,n,\tau} >= \frac{1}{Vol^T}\sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1} t^2_{i,k,j,n,\tau}\cdot\Delta^T_{i,k,jrow} \tag{18.142}$$

and the volume elements for U cells and T cells are

$$\Delta^U_{i,k,jrow} = umask_{i,k,jrow}\cdot dxu_i\cdot\cos\phi^U_{jrow}\cdot dyu_{jrow}\cdot dzt_k \tag{18.143}$$

$$\Delta^T_{i,k,jrow} = tmask_{i,k,jrow}\cdot dxt_i\cdot\cos\phi^T_{jrow}\cdot dyt_{jrow}\cdot dzt_k \tag{18.144}$$

The masks $umask_{i,k,jrow}$ and $tmask_{i,k,jrow}$ are 1.0 on ocean U cells and T cells but 0.0 on land U cells and T cells. Also, the relation between *j* and *jrow* is as described in Section 5.2. The total volumes are constructed as

$$Vol^U = \sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\Delta^U_{i,k,jrow} \tag{18.145}$$

$$Vol^T = \sum_{jrow=2}^{jmt-1}\sum_{k=1}^{km}\sum_{i=2}^{imt-1}\Delta^T_{i,k,jrow} \tag{18.146}$$

In addition to the above quantities, the iteration count from the elliptic solver is saved. When option *netcdf* is enabled, the output from this diagnostic has a NetCDF format and is written to file *ts_integrals.dta.nc*. Otherwise, the output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *ts_integrals.dta*. If option *netcdf* or *time_step_monitor_netcdf* is enabled, data is written in NetCDF format to file *ts_integrals.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *tsiint* and the control is specified by variable *iotsi*.

18.2.20 topog_netcdf

The output from option *topog_netcdf* is ocean depth at T cells given by

$$HT_{i,jrow} = zw_{kmt_{i,jrow}} \tag{18.147}$$

in units of cm and $f/HT_{i,jrow}$ in units of $cm^{-1}sec^{-1}$ where the Coriolis term f is defined at the latitude of T cells. When this option is enabled, output is written in a NetCDF format to file *topog.dta.nc*. There is no 32 bit IEEE unformatted data option. Therefore, the only way to get it is to enable option *topog_netcdf* (using option *netcdf* will not do it). Although this option is intended to be used when executing the topography module in stand alone mode from script *run_topog*, it can also be enabled in a model run using script *run_mom*. There is no associated interval or control variable.

18.2.21 trace_coupled_fluxes

This diagnostic is useful as an aid to diagnosing problems with coupling to atmosphere models using option *coupled*. It gives some gross statistics for the surface boundary conditions at various key places as they are being constructed for both atmosphere and ocean models. Output from this diagnostic is only written to file printout and is not intended for post processing.

18.2.22 trace_indices

This diagnostic is useful when trying to understand how the memory window operates for various settings of *jmw*. Refer to Section 3.3.2 for a description of how the memory window works. It gives a trace of the latitude loop indices indicating the dataflow from disk to memory window as MOM 2 executes. Also indicated are the latitude rows being worked on by various subroutines in preparation for solving the equations on rows in the memory window. As the memory window is moved northward, a listing of "which rows are copied where" is given. As new code is added to MOM 2, it is strongly recommended that this option be included as an aid in verifying that the code is correct. When using this option, disable all diagnostics and execute the model for only a limited number of time steps since the output can get voluminous. Refer also to Section 14.1.13.

18.2.23 tracer_averages

Option *tracer_averages* constructs <u>instantaneous</u> spatial averages of tracers over regional areas described by $mskhr_{i,jrow}$in Section 18.1.4. This diagnostic is useful when trying to examine how the model equilibrates with area and depth. The average of tracer n as a function of depth level k and region number m is constructed as

$$\overline{T}_{k,m,n} = \frac{1}{Area^T_{k,m}} \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} t_{i,k,j,n,\tau} \cdot A^T_{i,k,jrow} \tag{18.148}$$

$$m = mskhr_{i,jrow} \tag{18.149}$$

where the area element and area of each region at level k are given by

$$A^T_{i,k,jrow} = tmask_{i,k,jrow} \cdot dxt_i \cdot \cos\phi^T_{jrow} \cdot dyt_{jrow} \tag{18.150}$$

$$Area^T_{k,m} = \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} A^T_{i,k,jrow} \tag{18.151}$$

(18.152)

and the relation between j and *jrow* are as described in Section 5.2. The average of tracer n over all regions as a function of depth is given by

$$\overline{T}^m_{k,n} = \frac{1}{Area^T_{k,0}} \sum_{m=1}^{nhreg} \overline{T}_{k,m,n} \cdot Area^T_{k,m} \tag{18.153}$$

$$Area^T_{k,0} = \sum_{m=1}^{nhreg} Area^T_{k,m} \tag{18.154}$$

where $Area^T_{k,0}$ is the total area of all regions at level k. In a likewise manner, the tracer flux through the ocean surface for tracer n as a function of region number m is given by

$$\overline{stf}_{m,n} = \frac{1}{Area^T_{1,m}} \sum_{i=2}^{imt-1} \sum_{jrow=2}^{jmt-1} stf_{i,j,n} \cdot A^T_{i,1,jrow} \tag{18.155}$$

$$\overline{stf}^m_n = \frac{1}{Area^T_{1,0}} \sum_{m=1}^{nhreg} \overline{stf}_{m,n} \cdot Area^T_{1,m} \tag{18.156}$$

The output from this diagnostic is written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *tracer_avg.dta.* If option *netcdf* or *tracer_averages_netcdf* is enabled, data is written in NetCDF format to file *tracer_avg.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *tavgint* and the control is specified by variable *iotavg.* Note that the regional mask $mskhr_{i,jrow}$is also written to file *tracer_avg.dta* when the initialization boolean *itavg* is true. It should be set true on the first run but false thereafter.

18.2.24 tracer_yz

Option *tracer_yz* computes instantaneous values of the zonal integral of the tracer and tracer equation, term by term for each component. Contraction of the tracer equation along longitude yields a two dimensional equation as a function of latitude and depth. Using operators described in Section 11.10.7 yields

$$\begin{aligned}
\frac{1}{L^T_{k,jrow}} \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow} \cdot \delta_t(T_{i,k,j,n,\tau}) = & - \frac{1}{L^T_{k,jrow}} \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow}(ADV_Tz_{i,k,j} + ADV_Ty_{i,k,j}) \\
& + \frac{1}{L^T_{k,jrow}} \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow}(DIFF_Tz_{i,k,j} + DIFF_Ty_{i,k,j}) \\
& + \frac{1}{L^T_{k,jrow}} \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow} \cdot source_{i,k,j} \qquad (18.157)
\end{aligned}$$

where n is the tracer and the relation between *j* and *jrow* is as described in Section 5.2. The zonal advection and diffusion are not shown because they are eliminated by the summing operation. The length element and total length are given by

$$\Delta^T_{i,k,jrow} = tmask_{i,k,jrow} \cdot dxt_i \tag{18.158}$$

$$L^T_{k,jrow} = \sum_{i=2}^{imt-1} \Delta^T_{i,k,jrow} \tag{18.159}$$

Each sum in Equation (18.157) is output as one data field along with the zonally averaged tracer. The output from this diagnostic is only written in NetCDF format (no option for IEEE) to file *tracer_yz.dta.nc*. The interval between output is specified by variable *tyzint*.

18.2.25 trajectories

Option *trajectories* integrates particles along trajectories[11] using a forward time step and a particle velocity determined by instantaneous linear interpolation every time step.
Let n particles be placed at positions $\mathcal{P}_1(x,y,z), \mathcal{P}_2(x,y,z), \mathcal{P}_3(x,y,z) \cdots \mathcal{P}_n(x,y,z)$ at time $t0$. The position of the nth particle at time $t1$ is given as

$$\mathcal{P}_n(x,y,z) = \int_{t0}^{t1} V_n(x,y,z) \cdot dt \tag{18.160}$$

which is discretized as

$$\mathcal{P}_n^{\tau+1}(x,y,z) = \mathcal{P}_n^{\tau}(x,y,z) + \Delta\tau \cdot V_n^{\tau}(x,y,z) \tag{18.161}$$

where V_n^τ is the particle velocity at $\mathcal{P}_n^\tau(x,y,z)$ arrived at by linear interpolation from $u_{i,k,j,1,\tau}$, $u_{i,k,j,2,\tau}$, and $adv\_vbt_{i,k,j}$.

Particles are neutrally buoyant and are initially spread uniformly within an arbitrary volume as indicated by the example in *ptraj.F*. If option *lyapunov* is enabled, the deformation rate matrix em is also calculated[12] as the particles are integrated (Pierrehumpert, Yang 1993). The Lyapunov exponent λ is useful in quantifying the dispersion of the particle cloud and can be computed from the eigenvalues of this matrix as follows. Let

$$c = (em_{2,2} - em_{1,1})^2 + 4(em_{1,2} \cdot em_{2,1}) \tag{18.162}$$

Then the Lyapunov coefficient is given by

$$\lambda = \log(|\theta|)/T \tag{18.163}$$

If $c >= 0$ then

$$|\theta| = \frac{|(em_{1,1} + em_{2,2})^2 \pm \sqrt(c)|}{2} \tag{18.164}$$

Otherwise

$$|\theta| = \frac{\sqrt{(em_{1,1} + em_{2,2})^2} + |c|}{2} \tag{18.165}$$

This diagnostic is useful for investigating the evolution of water masses, three dimensional flow structure, and mixing properties of currents and waves (Stokes drift, chaotic mixing, etc).

[11] On the fly while MOM 2 integrates
[12] Only in two dimensions: longitude and latitude.

The storage requirement is six times[13] the number of particles and the computation is relatively fast. How many particles are reasonable? Start with about 10,000 and depending on how the results look, make adjustments. Particle positions are saved to the restart file to provide the necessary starting point for integrating the particles for arbitrary lengths of time. The output from Equation (18.161) which consists of an (x,y,z) position and a set of 3 integers per particle may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *particles.dta*. If option *netcdf* or *trajectories_netcdf* is enabled, data is written in NetCDF format to file *particles.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *trajint* and the control is specified by variable *iotraj*.

18.2.26 xbts

Originally, option *xbts* sampled temperatures and salinities at various latitude and longitude locations on the model grid down to some prescribed level and produced time averages of these quantities. In time, it grew to construct time averages of <u>all terms</u> in the prognostic equations[14] at each model level down to a specified depth for a set of stations. Station locations and depth at each station can be specified by looking at the USER INPUT section of *xbt.F* and following the examples. This diagnostic is useful when trying to understand the time evolution of dominant balances at specific locations in MOM 2. For instance, deploying a group of XBTs can elucidate how waves propagate or currents meander. It is also useful for planning where to deploy instrumented arrays or moorings in ship based experiments. *xbts* is similar to *term_balances* but instead of averaging over space in various regions of the domain, it averages over time at specific stations to prevents aliasing. Unlike *term_balances* this diagnostic is fast although it can get expensive in memory[15]. The output from this diagnostic may be written as ascii to the model *printout* or as 32 bit IEEE unformatted data to file *xbt.dta*. If option *netcdf* or *xbts_netcdf* is enabled, data is written in NetCDF format to file *xbt.dta.nc* rather than in unformatted IEEE. The interval between output is specified by variable *xbtint* and the control is specified by variable *ioxbt*. The averaging period is typically the same as the interval but may be specified shorter by variable *xbtper*. These variables are input through namelist. Refer to Section 5.4 for information on namelist variables.
Define m stations with coordinates (λ_m, ϕ_m) which are sampled to a depth of z cm. Discretize these locations to the nearest model grid points ($xbtlon_m$, $xbtlat_m$, $xbtdpt_m$).

<u>XBTs for the Momentum Equations</u>

Using operators and arrays described in Section 11.11.5, all components of the Momentum Equation are averaged in time for each station m and at each level k. The averaging period L corresponds to the number of time steps in the average. Subscript $n = 1$ refers to the zonal velocity component and $n = 2$ refers to the meridional component.

The canonical form of the terms for the mth station at level k are given as

$$uxbt_{1,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{u_{i,k,j,n,\tau+1} - u_{i,k,j,n,\tau-1}}{2\Delta\tau} \tag{18.166}$$

$$uxbt_{2,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -grad\_p_{i,k,j,n} \tag{18.167}$$

[13] Nine times if option *lyapunov* is enabled.
[14] This option has outgrown its name. Think of it as an enhanced XBT.
[15] 67 items are computed for each T and U grid cell combination

$$uxbt_{3,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV\_Ux_{i,k,j} \tag{18.168}$$

$$uxbt_{4,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV\_Uy_{i,k,j} \tag{18.169}$$

$$uxbt_{5,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV\_Uz_{i,k,j} \tag{18.170}$$

$$uxbt_{6,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_Ux_{i,k,j} \tag{18.171}$$

$$uxbt_{7,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_Uy_{i,k,j} \tag{18.172}$$

$$uxbt_{8,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_Uz_{i,k,j} \tag{18.173}$$

$$uxbt_{9,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_metric_{i,k,j,n} \tag{18.174}$$

$$uxbt_{10,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} CORIOLIS_{i,k,j,n} \tag{18.175}$$

$$uxbt_{11,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} source_{i,k,j} \tag{18.176}$$

$$uxbt_{12,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -grad\_p^{s}_{i,k,j,n} \tag{18.177}$$

$$uxbt_{13,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} ADV\_metric_{i,k,j,n} \tag{18.178}$$

$$\tag{18.179}$$

In Equation (18.167), the quantity $grad\_p^{s}_{i,k,j,n}$ is calculated from Equations (18.13) and (18.14). The equation for the kth grid cell at the mth station is given by

$$uxbt_{1,k,n,mth} = \sum_{\ell=2}^{13} uxbt_{\ell,k,n,mth} \tag{18.180}$$

Note that terms 3,4, and 5 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the physical advection in λ, ϕ, z because they contain divergent components. The canonical form of the physical advection in λ, ϕ, z is given below by terms 14,15, and 16.

$$uxbt_{14,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \cdot \frac{adv\_veu_{i,k,j} - adv\_veu_{i-1,k,j}}{dxu_i \cdot \cos\phi^U_{jrow}} - ADV\_Ux_{i,k,j} \tag{18.181}$$

$$uxbt_{15,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \cdot \frac{adv\_vnu_{i,k,j} - adv\_vnu_{i,k,j-1}}{dyu_{jrow} \cdot \cos\phi^U_{jrow}} - ADV\_Uy_{i,k,j} \tag{18.182}$$

$$uxbt_{16,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \cdot \frac{adv\_vbu_{i,k-1,j} - adv\_vnu_{i,k,j}}{dzt_k} - ADV\_Uz_{i,k,j} \quad (18.183)$$

$$uxbt_{17,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} u_{i,k,j,n,\tau} \quad (18.184)$$

$$xbtw_{mth} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{adv\_vbu_{i,k-1,j} + adv\_vbu_{i,k,j}}{2} \quad (18.185)$$

$$uxbtsf_{n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} smf_{i,j,n} \quad (18.186)$$

Term 17 represents the average horizontal velocity components within the cell and *xbtw* is the average vertical component. *uxbtsf* represents the windstress acting on the top of the near surface cell.

XBTs for the Tracer Equation

Using operators and arrays described in Section 11.10.7, all components of the Tracer Equation are averaged in time for each station m and at each level k. The averaging period L corresponds to the number of time steps in the average. Subscript $n = 1$ refers to the temperature component and $n = 2$ refers to the salinity.

$$txbt_{1,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} \frac{t_{i,k,j,n,\tau+1} - t_{i,k,j,n,\tau-1}}{2\Delta\tau} \quad (18.187)$$

$$txbt_{2,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV\_Tx_{i,k,j} \quad (18.188)$$

$$txbt_{3,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV\_Ty_{i,k,j} \quad (18.189)$$

$$txbt_{4,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} -ADV\_Tz_{i,k,j} \quad (18.190)$$

$$txbt_{5,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_Tx_{i,k,j} \quad (18.191)$$

$$txbt_{6,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_Ty_{i,k,j} \quad (18.192)$$

$$txbt_{7,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} DIFF\_Tz_{i,k,j} \quad (18.193)$$

$$txbt_{8,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} source_{i,k,j} \quad (18.194)$$

$$txbt_{9,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} explicit\ \ convection \quad (18.195)$$

$$(18.196)$$

When options *isopycmix* and *gent_mcwilliams* are enabled, then Equations (18.188), (18.189), and (18.190) also include the flux form of the advection terms from option *gent_mcwilliams* given by $-ADV\_Txiso_{i,k,j}$, $-ADV\_Tyiso_{i,k,j}$, and $-ADV\_Tziso_{i,k,j}$ respectively. In Equation (18.195), the quantity *explicit convection* comes from solving Equation (18.122) before and after explicit convection. The equation for the kth grid cell in station m is given by

$$txbt_{1,k,n,mth} = \sum_{\ell=2}^{9} txbt_{\ell,k,n,mth} \tag{18.197}$$

Note that terms 2,3, and 4 are the flux form of advection. When summed up, they represent the physical advection in the cell. However, when taken separately, they do not represent the physical advection in λ, ϕ, z because they contain divergent components. The physical advection in λ, ϕ, z is given below by terms 11,12, and 13.

$$txbt_{11,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \cdot \frac{adv\_vet_{i,k,j} - adv\_vet_{i-1,k,j}}{dxt_i \cdot \cos\phi^T_{jrow}} - ADV\_Tx_{i,k,j} \tag{18.198}$$

$$txbt_{12,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \cdot \frac{adv\_vnt_{i,k,j} - adv\_vnt_{i,k,j-1}}{dyt_{jrow} \cdot \cos\phi^T_{jrow}} - ADV\_Ty_{i,k,j} \tag{18.199}$$

$$txbt_{13,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \cdot \frac{adv\_vbt_{i,k-1,j} - adv\_vnt_{i,k,j}}{dzt_k} - ADV\_Tz_{i,k,j} \tag{18.200}$$

$$txbt_{15,k,n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} t_{i,k,j,n,\tau} \tag{18.201}$$

$$txbtsf_{n,mth} = \frac{1}{L}\sum_{\ell=1}^{L} stf_{i,j,n} \tag{18.202}$$

When options *isopycmix* and *gent_mcwilliams* are enabled, Equations (18.198), (18.199), and (18.200) also include the advective or transport velocities from option *gent_mcwilliams* given by $adv\_vetiso_{i,k,j}$, $adv\_vntiso_{i,k,j}$, and $adv\_vbtiso_{i,k,j}$ respectively. Term 15 represents the average tracer within the cell and term *txbtsf* represents the surface tracer flux acting on the top of the surface cells.

Chapter 19

Getting Started

This chapter describes what is needed to start using MOM 2.

19.1 System Requirements

A Fortran 90 (preferable[1]) compiler "f90" or a Fortran 77 compiler "f77" is required along with a preprocessor such as UNIX "cpp" or its equivalent. Run scripts are written in UNIX C shell so the ability to handle this is also required. For executing MOM 2, a high end workstation (SGI, SUN, DEC, etc.) or supercomputer (CRAY YMP, CRAY C90, CRAY T90, etc.) is required.

19.2 Changes for Fortran 77 users

Since the arrival of the CRAY T90 in September 1996, the Unicos operating system at GFDL only supports a Fortran 90 compiler. If using a Fortran 77 compiler, note the following changes from previous versions of MOM 2:

- Namelist input data has been made compliant with the Fortran 90 form. The script *run_mom* now stores each namelist input record as a separate file instead of concatenating all namelists into one file. This is fine for Fortran 77 also but, instead of terminating namelist records with an "&end", each namelist record is now terminated with a "/". This terminator must be changed back to "&end" if executing under a Fortran 77 environment.

- Run scripts for executing the model (e.g. *run_mom* described below) and for preparing data (e.g. *run_sbc* etc. also described below) use the "f90" command since they are typically executed on the CRAY T90. If using Fortran 77, the appropriate "f77" command must be substituted. Look at script *run_mom_sgi* which is the counterpart to *run_mom* on the SGI workstations at GFDL.

- Option *f90* must be added to the list of MOM 2 options when compiling with a Fortran 90 compiler. This is needed to configure the Fortran "open" statement properly. Remove this option if compiling with a Fortran 77 compiler.

- Diagnostic routines which save data in NetCDF format require a parameter setting to indicate the size of the largest model dimension which is typically "imt", "jmt", or "km". Fortran 90 allows the use of a "max" function within a parameter statement which conveniently automates the choice. Functions were not allowed in parameter statements under

[1]Because "namelist" and the "open" statement are standard features as opposed to non-uniform extensions.

cf77 on the CRAY YMP or CRAY C90. On these and other such systems, the "max" function needs to be replaced with the maximum of "imt", "jmt", and "km". Which one to use is of course dependent on model configuration.

In summary, if a Fortran 77 compiler is being used, the namelist terminator within the run scripts must be changed back to "&end", the "f90" command must be changed back to the appropriate "f77" command with appropriate compiler options, and the option "-Df90" must be removed from the option list.

19.3 Special request for beta testers

A newer I/O manager is available which is much smaller, simpler, and easier to use and make changes to than the older one. Refer to Section 6.2.4 for a description and details. It can be tested in a "stand alone mode" by setting the PLATFORM variable in script *run_iomngr_new* and executing. The older I/O manager is marked for extinction so it is important to test this newer one on various platforms which do not exist at GFDL.

19.4 Accessing MOM 2

MOM 2 is kept on the anonymous ftp at GFDL. It can be accessed with the following steps:

ftp ftp.gfdl.gov	To reach GFDL's anon ftp.
	use *ftp* as your login name and your *e-mail* address as your password
cd pub/GFDL_MOM2	Change to the pub/GFDL_MOM2 directory
get MOM2.2tar.Z	Copy the version of MOM 2 to a directory at your site
quit	disconnect from the ftp
uncompress MOM2.2tar.Z	Expand to MOM2.2tar
tar xvf MOM2.2tar	Extract and build the MOM_2 directory structure on your disk

MOM2.2tar.Z is the code file and takes approximately 1.5MB of diskspace. The uncompressed version takes approximately 5MB. Note that directory MOM_2 (not MOM_2.2) will be built when the tar file is extracted.

A database is also included as part of MOM 2. While in the pub/GFDL_MOM2 directory, do a *cd* DATABASE to get into the DATABASE directory. This DATABASE directory contains approximately 160MB of IEEE 32bit data files which are described in Chapter 4. Either selected files or all files can be retrieved with the *get* command as in the anonymous ftp example given above. It is best to copy the files one at a time. The amount of data may take a non-trivial time to copy so be prepared to go to lunch.

As an alternative, the database for MOM 2 is available on exabyte tape (Data Grade 8mm 112m). This method of access is <u>not</u> encouraged since it is relatively easy for GFDL to get flooded with requests and this presents a problem. If a copy of the database on exabyte tape is requested, the name of the person requesting the data will be noted. Please be willing to pass on the favor by making a copy for someone else if asked. Also, please send a blank exabyte tape with the request. Files can be extracted from the exabyte tape with the following steps:

mkdir DATABASE	make a directory to hold the climatological DATABASE
cd DATABASE	change to the database directory
tar xvf /dev/rst0	extract the database files from tape on exabyte drive rst0

19.5 How to find things in MOM 2

Finding things within a large model like MOM 2 is no problem with the aid of UNIX utilities such as *grep*. For example, suppose all areas within the model having anything to do with isopycnal mixing are to be located. Searching for option *isopycmix* with the following command

```
grep -i isopycmix *.[Fh]
```

will find all such sections. The "-i" option is useful because it ignores upper/lower case distinctions. Searching for names of variables can likewise show every place where they are used. Definitions for variables can be seen by searching all ".h" files. Another very useful UNIX utility is "diff" as described in Section 19.16.

19.6 Directory Structure

First, refer to Figure 19.1 for a schematic view of how the directory structure of MOM_2 is organized at GFDL. The structure is divided between two file systems: the CRAY file system contains the data part and the workstation file system contains all code and *run_scripts*.

On the CRAY file system, there is an ARCHIVE/MOM_2/DATABASE directory. The DATABASE contains Hellerman and Rosenstein (1983) monthly climatological wind stress on a 2° grid, Oort (1983) Monthly Surface air temperature on a 5° grid, Levitus (1982) monthly temperature and salinity on a 1° grid, and Scripps topography on a 1° grid[2]. There is also an ARCHIVE/MOM_2/EXP directory where interpolated data from the DATABASE and results[3] from various experiments are stored, each under their own sub-directory[4]. The only sub-directory included is ..EXP/TEST_CASE which (after executing run scripts described below) will contain an interpolated version[5] of the DATABASE appropriate for the domain and resolution of the test case which is described below.

On the workstation file system, there is also a MOM_2 directory containing code, *run_scripts*, and four sub-directories: MOM_2/PREP_DATA for preparing surface boundary conditions and initial conditions, MOM_2/SBC for handling various types of surface boundary conditions, MOM_2/NETCDF containing routines to interface to the netcdf library, and MOM_2/EXP which in general contains a sub-directory for each experiment. Details of these sub-directories are given below:

- PREP_DATA contains subroutines and CRAY T90 *run_scripts* for extracting data[6] from the DATABASE and interpolating it to arbitrary resolution for use as surface boundary conditions and initial conditions within MOM 2. Before this can be done, the domain and resolution must first be specified in module *grids* as discussed in Chapter 7. The run scripts are:

[2] In principle, this DATABASE could be expanded to include other datasets but this has not been done as of this writing

[3] Model output may be composed of a printout file, diagnostic files and restart data.

[4] For example, EXP/ATLANTIC, EXP/PACIFIC, EXP/GLOBE.

[5] Note that these interpolated datasets are only needed for test cases #1 and #2. Test cases #0 and #3 rely on internally generated data.

[6] All DATABASE data consists of a header record preceding each data record. Included in each header is a time stamp. It contains the time corresponding to the instantaneous time at the end of the averaging period. It also contains a period which refers to the length of the time average. As an example, a time stamp of: m/d/y= 2/ 1/1900,h:m:s= 0: 0: 0. points to the beginning of the 1st day of Feb on year 1900. A period of 31 days for this record means that the data is average over the preceding 31 days; i.e, it is an average for January.

1. *run_sbc* reads unformatted climatological monthly Hellerman stress (1983) and Oort (1983) surface air temperature and interpolates to the grid defined by module *grids*. Look for the USER INPUT section to choose the type of interpolation appropriate for the grid resolution. This script uses file *sbc.F* which is included.
2. *run_ic* reads unformatted monthly Levitus (1982) temperature[7] and salinity data and generates monthly climatological initial conditions along with surface temperature and salinity for the grid defined by module *grids*. Look for the USER INPUT section to choose the type of interpolation appropriate for the grid resolution. This script uses file *ic.F* which is included.
3. *run_sponge* reads output files produced by *run_ic* to construct sponge rows for damping model predicted temperature and salinity back to these data near northern and southern artificial walls. This is only appropriate for use in limited domain models and is the poor mans open boundary condition. This script uses file *sponge.F* which is included. The width of the sponge layers and the variation of Newtonian damping time scale within the sponge layer may be set within file *sponge.F*.
4. *run_read_levitus* is a simple workstation script showing how to read the Levitus (1982) data (with Levitus land/sea masks) on a workstation. It assumes the Levitus (1982) data has been copied to the workstation's local disk.
5. *run_obc* is a run script which uses file *obc.F* for constructing data needed for open boundary conditions. This was done by Arne Biastoch (abiastoch@ifm.uni-kiel.de) but has not been converted to the CRAY T90 at GFDL at this point.
6. *run_obcpsi* is a run script which uses file *obcpsi.F* for constructing data needed for open boundary conditions. This was done by Arne Biastoch (abiastoch@ifm.uni-kiel.de) but has not been converted to the CRAY T90 at GFDL at this point.

- SBC contains three sub-directories for supplying various types of surface boundary conditions to MOM 2. Each is located in a separate sub-directory:

1. TIME_MEAN contains subroutines which supply the time mean Hellerman and Rosenstein climatological winds (1983) along with the time mean Levitus (1982) SST and sea surface salinity climatologies which are used by the test case to compute effective heat and salt fluxes given a damping time scale and thickness which can be input from a MOM 2 namelist. Refer to Section 5.4 for information on namelist variables. Note that the time scale and thickness can be different for restoring temperature and salinity. These time means are assumed to have been created using scripts from PREP_DATA so they are appropriately defined as functions of latitude and longitude on the domain and resolution specified by module *grids*. The option used to configure this type of surface boundary condition for MOM 2 is *time_mean_sbc_data* which is described further in Chapter 10.
2. MONTHLY contains subroutines which supply monthly mean Hellerman and Rosenstein climatological winds along with monthly mean Levitus (1982) climatological SST and sea surface salinity which are used by the test case to compute effective monthly mean heat and salt fluxes given a damping time scale which can be input from a MOM 2 namelist. Refer to Section 5.4 for information on namelist variables. Note that the time scale and thickness can be different for restoring temperature and salinity. All are assumed to have been created by scripts from PREP_DATA so they

[7]These are potential temperatures.

are monthly averages appropriately defined as functions of latitude and longitude on the domain and resolution specified by module *grids*. Each dataset is defined by an averaging period and time stamp which marks the end of the averaging period. As the model integrates, the datasets are used to interpolate to the time corresponding to each model time step. It should be noted that there is enough generality to accommodate datasets with other periods (daily, hourly, etc) and treat them as climatologies (periodic) or real data (non periodic). Also datasets with differing periods may be mixed (example: climatological monthly SST may be used with hourly winds from other datasets). The option used to configure this type of surface boundary condition for MOM 2 is *time_varying_sbc_data* which is described further in Chapter 10. There are four methods for interpolating these datasets to the time step level required by MOM 2 as described in Section 10.2 .

3. <u>ATMOS</u> contains subroutines that prototype what must be done to couple MOM 2 to an atmosphere model for the general case of two way coupling when resolution and land/sea areas do not match. The atmosphere model is unrealistic. It is intended only to show that essentially two things must be done: a boundary condition grid must be defined to match the atmospheric grid (which is assumed to be different from the MOM 2 grid resolution) and boundary conditions such as winds and heat flux must be accumulated in arrays as indicated. The option used to configure this type of surface boundary condition for MOM 2 is *coupled* which is explained further in Section 10.1.

- NETCDF contains fortran routines written by John Sheldon at GFDL for interfacing to lower level *netcdf* routines. These lower level routines are resolved by linking to the appropriate NetCDF libraries which will be site specific. The proper linking to these libraries at GFDL is given in script *run_mom*. For other sites, the appropriate links will have to be made by the researcher. MOM 2 uses wrapper utilities of its own which access Sheldon's files to further simplify and unify writing NetCDF related code. These utilities are in file *util_netcdf.F* within the MOM 2 directory. The NetCDF section of any diagnostic can be used as a template to add NetCDF capability to new diagnostics. For further information, refer to Section 18.2.10.

- EXP contains one sub-directory for each experimental design but only EXP/TEST_CASE is indicated. If there were others, they would have the same structure. EXP also contains printout files from the four test cases described later. They were produced on the CRAY T90 at GFDL and are named *printout.0.gfld*, *printout.1.gfld*, *printout.2.gfld*, and *printout.3.gfld*. These files can be used for comparison with results generated elsewhere and are described further in Section 19.8. Under the EXP/TEST_CASE are two sub-directories:

1. <u>MOM_UPDATES</u> contains <u>only</u> code and run_scripts from the MOM_2 directory which need to be altered to define an experiment (e.g. the test case). Actually, no fortran code is included here because the basic MOM_2 files are already configured for the test case. Typically though, the following would be a minimum set of useful ones: module *grids* and *run_grids* which are used to design the grid, *size.h* which is used to implement the grid size, and module *topog* and *run_topog* which are used to design the topography and geometry. Also, any other subroutine requiring changes <u>must</u> be placed in this directory because Cray script *run_mom* looks to this MOM_UPDATES directory for all updated code.
2. <u>PREP_UPDATES</u> contains only code and CRAY T90 run_scripts from the PREP_DATA directory which would have be altered to define the test case. Actually, none are

here since the ones in PREP_DATA are already setup to do the test case. Typically though, only run_scripts need be copied into this directory to alter pathnames (near the beginning of the scripts) which point to where interpolated initial conditions and surface boundary conditions are to be written. The scripts are then executed from this directory on the CRAY T90 to build the interpolated DATABASE appropriate for the resolution specified by module *grids*.

19.7 The MOM 2 Test Cases

MOM 2 is executed by a CRAY T90 script *run_mom* which is in directory MOM_2 on the workstation. A corresponding script *run_mom_sgi* is included for SGI workstations. The script executes a test case global domain with 4° longitude by 3° latitude by 15 vertical level resolution using idealized geometry and topography and exercises only a few of the available options for MOM 2. Test cases #0, #1, #2, and #3 use various types of surface boundary conditions and are selected by setting the CASE variable within *run_mom* as follows:

- CASE=0 uses idealized surface boundary conditions which are a function of latitude only and independent of time: zonally averaged annual mean Hellerman and Rosenstein (1983) wind stress with surface temperature and salinity damped back to initial conditions on a time scale of 50 days using a thickness of about 25 meters. Initial conditions are no motion and an idealized temperature (function of latitude and depth) and salinity (constant) structure[8]. All required data is generated internally and therefore the DATABASE is <u>not</u> needed. This is similar to the test case for MOM 1. The results are in file *EXP/TEST_CASE/printout.0.gfdl.*

- CASE=1 is similar to CASE=0 except uses time mean surface boundary conditions from SBC/TIME_MEAN which are assumed to have been prepared using scripts *run_sbc* and *run_ic* in PREP_DATA. These surface boundary conditions are a function of longitude and latitude but independent of time. The results are in file *EXP/TEST_CASE/printout.1.gfdl.*

- CASE=2 is similar to CASE=0 except uses time varying surface boundary conditions from SBC/MONTHLY as described in Section 19.6 which are assumed to have been prepared using scripts *run_sbc* and *run_ic* in PREP_DATA. The surface boundary conditions are linearly interpolated to each time step as the integration proceeds. The results are in file *EXP/TEST_CASE/printout.2.gfdl.*

- CASE=3 is similar to CASE=0 except uses surface boundary conditions supplied by an idealized atmospheric model as described in Section 19.6. This illustrates coupling MOM 2 to an atmospheric GCM. The results are in file *EXP/TEST_CASE/printout.3.gfdl.*

19.7.1 The run_mom and run_mom_sgi scripts

As mentioned previously, script *run_mom* is a UNIX C shell script which executes the MOM 2 four test cases (#0, #1, #2, and #3) on the CRAY T90 at GFDL. A corresponding script *run_mom_sgi* is included for SGI workstations. Near the beginning of script *run_mom*, pathnames point to where all required directories are located at GFDL. They will have to be changed at each installation. Control forwWhich test case executes is given by C shell variable CASE. CASE=0 is for test case 0 and so forth.

[8] If Levitus (1982) SST and sea surface salinity are to be used as initial conditions, option *levitus_ic* must be enabled as discussed in Section 15.8.

When *run_mom* executes, it copies all Fortran code from directory MOM_2 into a working directory followed by all code from either MOM_2/SBC/TIME_MEAN (if CASE = 1), MOM_2/SBC/MONTHLY (if CASE = 2), or MOM_2/SBC/ATMOS (if CASE = 3). If any NetCDF option is on, all files from MOM_2/NETCDF are also copied. Lastly, it copies all Fortran code from the EXP/TEST_CASE/MOM_UPDATES directory thereby installing all changes necessary (if any) to build the particular model.

Various ways of configuring MOM 2 are controlled by options. This includes enabling diagnostics as described in Chapters 18 and 15. Options are set within the script using *cpp* preprocessor commands of the form *-Doption1*, *-Doption2* and so forth. These options eliminate or include various portions of code to construct a model having the desired components. They are also used to enable diagnostics and whether output is in NetCDF format or not. Note also, that the computer platform is specified within *run_mom*. Currently, the list includes -Dcray_ymp, -Dcray_c90, -Dcray_t90, and -Dsgi. Based on which setting is chosen, appropriate platform options are added for routines in MOM_2/NETCDF.

There is no makefile supplied for compiling MOM 2. If compile time is an issue, then one can be constructed by the researcher. In the compiling section of the script, there is provision for enabling a bounds checker. This is strongly recommended as standard operating practice for verifying that subscripts do not exceed array bounds in newly developed code. Afterwards, the bounds checker should not be used since it significantly slows execution.

The compiling and linking to an executable is done in one step under "f90". If using "f77" refer to Section 19.2 and make the changes. After compiling, separate namelist files are constructed which contain specifications to reset various defaulted quantities. Refer to Section 5.4 for a list. These namelist files are read by subroutine *setocn* and other initialization subroutines specific to individual parameterizations which have been enabled. Note that each namelist ends with a "/" in compliance with Fortran 90 and is written to a separate file. The ending "/" should be changed to an "&end" for use with Fortran 77 compilers.

At this point, the executable is executed and output is redirected to file *results* which is later copied to the appropriate printout file. Except for the *printout* file which is ASCII, all diagnostic data is written to separate files as either *IEEE 32 bit* data (having a ".dta" suffix) or NetCDF formatted data (having a ".nc" suffix) as described in Chapter 18.

There are some additional files. The files *document.dta* and *restart.dta* also have a ".dta" suffix but are not diagnostic files. File *document.dta* is a formatted file containing all namelist settings plus some additional information. File *restart.dta* contains data needed to restart the integration from the point where it last ended. To write a restart, variable *restrt* must be set to *true* in the namelist. Within an actual integration, pathnames would be changed so that these files would be copied to the experiment directory (e.g. *cp *.nc* EXP/ATLANTIC) on the supercomputer archive. Refer to Section 18.1.1 for post processing the results.

The script *run_mom* is set to execute CASE=0. All test cases have a heavy load of diagnostics enabled for demonstration purposes. Look at the timing estimates at the end of the printout to see what they cost. Turn off the ones not needed by removing them from the option list in script *run_mom*.

19.8 Sample printout files

As mentioned previously, there are four printout files corresponding to four test cases which were executed at GFDL on a CRAY T90. Results from CASE=0 are in file *EXP/TEST_CASE/printout.0.gfdl*, from CASE=1 are in file *EXP/TEST_CASE/printout.1.gfdl*, and so forth. These cases are not intended to be scientifically meaningful. Rather they are included as examples of how to use various types of surface boundary conditions and provide a means of checking that MOM 2 is behaving as intended. The following is a brief tour of file *printout.0.gfdl*.

File *EXP/TEST_CASE/printout.0.gfdl* begins by listing various surface boundary condition names and units followed by the version number of MOM 2 and the values of namelist variables. If any variable was not included in the *run_mom* script namelist section, then it retains its initialized value. Otherwise, the new value is given. Immediately below, there is the file sizes in megawords needed for two dimensional fields (kflds) and the three dimensional fields (latdisk1 and latdisk2) where the latitude rows reside. If option *ramdisk* is enabled, these files are actually stored in memory but behave as if stored on disk.

Next, there is output from the *grids* module detailing everything relevant to grid cells. All this is summarized by a checksum which is essentially a sum over all grid cell information. Following this is a summary of temperature and salinity ranges used to compute density coefficients and a checksum of the coefficients[9]. Afterwards, output from the topography module *topog* supplies information about the $kmt_{i,jrow}$ field. Changes to the $kmt_{i,jrow}$ field are best done in a stand alone mode using script *run_topog*. Checking for violations is done iteratively so this section rambles on for awhile. If everything is as it should be then some basic statistics relating to geometry and topography are given with a map of land masses and island perimeters followed by a map of $kmt_{i,jrow}$. Along with these maps is information on how to suppress their printing since they can take up lots of space. The section finishes with a checksum for the topography and geometry.

After constructing a checksum over initial conditions, various initializations are indicated. The time manager module *tmngr* gives details on the calendar and time at initial conditions as well as information on the reference time for diagnostic switches. This is followed by information on damping surface tracers back to data which is given when option *restorst* is enabled.

Since the test case is of global extent, filtering of latitude rows is enabled (by option *fourfil* or *firfil*) and information is given as to which latitudes are filtered and by how much. Refer to Section 15.11 for a discussion of filtering.

Next comes some statistics on regions which have been arbitrarily set for diagnostic purposes along with a map detailing where the regions are. A detailed listing of filtering indices is suppressed but may be switched on as indicated. The time step multipliers are all set to unity indicating that there is no timestep acceleration with depth.

If option *time_averages* has been enabled, information on the grid over which data will be time averaged is given and if option *xbts* is enabled, the XBT station locations are given. Depending on enabled options, other initializations may give output here after which a general consistency check is done involving all enabled options. Two levels of messages are given: error and warning checks. After all messages are listed, if one or more error messages is present the model will stop. Warning messages will not stop the model, but the researcher should be aware of them and convinced they are harmless before continuing.

The preliminary setup finishes with a summary of enabled options after which a breakdown of the number of time steps per ocean segment and number of segments per integration is given. This is of interest only when option *coupled* is enabled.

A heavy battery of diagnostics are enabled but only for illustrative purposes. They are fully described in Chapter 18 and the I/O control variables are set to save data to unformatted files with suffix .dta as well as formatted to the printout file. At the end of the integration, all files are listed and a timing analysis is given detailing times taken by various sections of MOM 2. This is useful but once the information is digested, the timing should be turned off by not enabling option *timing*.

[9]Density coefficients are computed by a call to module *denscoef.F* from within the model.

The other printout files are very similar except for case #3 which has more output because option *trace_coupled_fluxes* has been enabled to show details of surface boundary conditions as they are being interpolated from ocean to atmosphere and visa versa.

19.9 How to set up a model

As an example, assume an Atlantic model is to be set up. Once familiar with the directory structure as outlined in Section 19.6 and illustrated in Fig 19.1, the following steps[10] may be used:

1. Add a sub-directory under EXP with two additional sub-directories for containing updates or changes which when applied to the base code in MOM_2 will defined the new model. For example, the new experiment might be named EXP/ATLANTIC and the two additional sub-directories: EXP/ATLANTIC/MOM_UPDATES and EXP/ATLANTIC/PREP_UPDATES.

2. Copy file *grids.F* and script *run_grids* from MOM 2 into EXP/ATLANTIC/MOM_UPDATES. If not executing on a CRAY T90, add option *sgi* to script *run_grids*. Specify a domain and grid resolution by entering changes in the USER INPUT of module *grids* as described in Chapter 7. Execute script *run_grids*. Examine the output and when satisfied, copy *size.h* from MOM 2 into the EXP/ATLANTIC/MOM_UPDATES directory and make the indicated parameter changes. This domain and grid resolution will now be available to other modules and MOM 2.

3. Copy file *topog.F* and script *run_topog* from MOM_2 into EXP/ATLANTIC/MOM_UPDATES. The model geometry and topography will be generated by executing script *run_topog* with *options* outlined in Chapter 9. If not executing on a CRAY T90, add option *sgi* to script *run_topog*. The domain and resolution will be defined from module *grids*. Note that the $kmt_{i,jrow}$ field is printed out. Decide which if any changes are needed and enter them in the USER INPUT section of module *topog*. The $kmt_{i,jrow}$ field can also be viewed with option *topog_netcdf* which may be more convenient.

   Recommendation: When setting up a model for the first time, use options *idealized_kmt* with *flat_bottom* to generate a flat bottomed idealized geometry for the region of interest. After becoming comfortable with the way the process works, enable a more appropriate option (e.g., option *scripps_kmt*).

4. Select options from a list of available ones described in Chapter 15. Enable selected options by including them on the compile statement in script *run_mom* to configure the model. Diagnostics are the analysis tools used to help understand the model solution. Select appropriate diagnostics from the ones described in Chapter 18 and enable them by including them on the compile statement in script *run_mom*.

   Recommendation: When setting up a new model, use options *idealized_ic*, *simple_sbc*, and *restorst* which will simplify initial conditions and boundary conditions. Also, choose the simplest mixing schemes using options *consthmix* and *constvmix*. Only after verifying that results are as expected should consideration be given to moving on to more appropriate options. In general, progress by enabling and verifying one option at a time until the desired configuration is reached. If problems occur, simlify the configuration to help pinpoint the cause.

[10] Note that it is no longer necessary to construct density coefficients prior to executing the model.

5. Certain options require input variables to be set. All are set to default values but these values may not be appropriate for the particular model being used. Their values may be changed to more appropriate ones through namelist. For setting input variables, read through Section 5.4 for a description of the namelist variables. Also, guidance is often given within the description of the option and this information should be read.

The following steps are optional. They apply if the DATABASE is to be used or option *time_averages* is enabled.

1. If it is desirable to use data from the DATABASE, copy the scripts from PREP_DATA into the EXP/ATLANTIC/PREP_UPDATES directory, change pathnames to point appropriately, and execute *run_sbc* followed by scripts *run_ic* and *run_sponge*. These will build a copy of the DATABASE appropriate to the domain and resolution specified in module *grids*.

2. If it is desirable to produce time averages during the integration, copy script *run_timeavgs* and file *timeavgs.F* to EXP/ATLANTIC/MOM_UPDATES. The grid used for producing time averages must be defined by modifying the USER INPUT section of file *timeavgs.F*. The entire model grid or a subset of grid points may be chosen. If after executing this script, a change is made to module *grids*, then this script must be executed again to re-establish the averaging grid. If examining the *results file* indicates that everything is as intended, copy file *timeavgs.h* from MOM_2 to EXP/ATLANTIC/MOM_UPDATES and make the indicated parameter changes.

19.10 Executing the model

Once the steps in Section 19.9 have been taken, make a copy of script *run_mom* (or script *run_mom_sgi* if executing on a workstation), change pathnames to point appropriately and add the desired options from Chapter 15. Any options used in scripts *run_grids* and *run_topog* must also be included in the *run_mom* script. If not executing on a CRAY T90, use option *sgi*, option *cray_ymp*, or option *cray_c90* in script *run_mom*. To keep things simple, make a short test run with options *time_step_monitor,snapshots* and *netcdf* to produce a snapshot of the data. Have a look at the data using Ferret (Section 18.1). After a successful test run, enable the desired diagnostic options and disable option *timing*.

As mentioned previously, this script was written for a CRAY T90 at GFDL which assumes an "f90" compiler. If not using an "f90" compiler, refer to Section 19.2. Changes required to run on a workstation are relatively few. Script *run_mom_sgi* is included as an example of a script which runs on SGI Indigo workstations using a Fortran 77 compiler at GFDL. Compare *run_mom* and *run_mom_sgi* to see the differences.

Production scripts

Production scripts are left to the researcher although they can be modeled after *run_mom*. At the minimum, commands must be added to allow for automatic reloading, archiving of results, and handling problems associated with long running experiments which may be site specific.

19.11 Executing on 32 bit workstations

If executing on a workstation with a 32 bit word length, it is recommended that double precision (usually a compiler option) be used otherwise numerical truncation may significantly limit accuracy. On an SGI Indigo 4000 workstation, the options include "-O2 -mips2 -r8 -align64 -Olimit 2160". For test cases #1 and #2, recall that data in the DATABASE is in 32 bit IEEE format. Routines for reading this data (e.g. *ic.F* and *sbc.F* in PREP_DATA) and preparing it for the model can be compiled with 32 bit word length "-O2" and the write statements changed to output real*8 data. This is left to the researcher.

19.12 NetCDF on 32 bit workstations

Generating NetCDF formatted output from MOM 2 is as easy as specifying an option. Of course, the UNIDATA NetCDF libraries must be installed. Refer to Chapter 18 and Section 18.2.10 for specifics.

NetCDF routines are very platform dependent. MOM 2 with NetCDF options works well on CRAY platforms (64bit word length) and SGI platforms (32bit word length) at GFDL. Each platform uses its own NetCDF library. However, NetCDF will only work with double precision on SGI workstations when the model (all files except those in MOM_2/NETCDF) is compiled in double precision <u>and</u> the NetCDF libraries and all routines in MOM_2/NETCDF are compiled in single precision (32bit word length) <u>and</u> the option *sgi* is used. To accomplish this mismatch in precision, double precision data is converted to single precision just for the NetCDF routines in two places: routines *ncseti* and *ncput*. Look at script *run_mom_sgi* to see how the compile is done.

19.13 Distributed memory systems

Originally, a CRAY T3E with 40 processors was scheduled to arrive at GFDL in November 1996. Latest word was that the system will not arrive until sometime in 1997 and work on extending MOM 2 for distributed systems will not be completed until the CRAY T3E has been installed. However, the groundwork has been done and it is anticipated that MOM 2 will be able to execute efficiently on the CRAY T90 and CRAY T3E without much change to the coding.

This feature is currently under development and is not yet stable. Refer to Section 3.5.3 for details.

19.14 MOM 1 and upgrading to MOM 2

MOM 1 included an upgrade script for incorporating changes. Since there is a major architectural difference between MOM 1 and MOM 2, there are too many changes to offer a meaningful upgrade approach from MOM 1. The only recourse is to bite the bullet and switch. It should be obvious that the appropriate time for switching is at the beginning of an experiment but not in the middle of one.

19.15 Upgrading from older to newer versions of MOM 2

MOM 2 version 1.0 included a script "run_upgrade_sgi" for incorporating local changes into newer versions of MOM 2. This approach has since been discarded in favor of a much better one: reliance on the directory structure in MOM 2 and a new utility ... the graphical difference analyzer "gdiff" which exists on Silicon Graphics workstations and makes easy, painless work out of what was once a difficult, time consuming, and complicated task. Presumably, similar software exists from other vendors. If not, an alternative method outlined below will still work, only not as easily. The "gdiff" utility has quickly become an indispensable tool for development work at GFDL.

Standard operating practice

First and foremost, as a matter of operating practice, NEVER change a routine within the MOM_2 directory. Copy it first into an UPDATES directory and make changes there. A different UPDATES directory should be associated with each experiment. Variants of routines within each UPDATES directory can be kept in sub directories (e.g. UPDATES/TEST1, UPDATES/TRIAL2, UPDATES/TEST1/QX etc.) and in this way a hierarchy of changes can be maintained. Selecting which sub-directories to copy and in which order will allow any combination of updates to be combined and this should be done in the run script. Before starting, move the whole existing MOM_2 directory structure to MOM_2_OLD using

```
mv  MOM_2  MOM_2_OLD
```

Then install the newer MOM_2 directory by uncompressing and extracting the tar file after retrieving it from the GFDL anonymous ftp. Assume all local updates are in MOM_2_OLD/EXP/BOX/MOM_UPDATES For illustrative purposes, this will be referred to as OLD_UPDATES. It is important to realize that although many of the routines in the newer version of MOM 2 may have changed, only routines in OLD_UPDATES will have to be examined. Make a directory under MOM_2 with the same structure using

```
mkdir  MOM_2/EXP/BOX/MOM_UPDATES
```

In what follows, let the above sub-directory be referred to as NEW_UPDATES.

The recommended method:

Use "gdiff" to compare each routine in OLD_UPDATES with the corresponding one in MOM_2. Within "gdiff', click the right mouse button and select the option to PICK all changes from the routine in MOM_2. Then use the mouse to mark each change to be added from the routine in OLD_UPDATES. Select WRITE FILE, type in the filename using a pathname to NEW_UPDATES, and the file written will have all marked changes merged together. In the event that part of a local change overlaps a change in MOM_2, an editor can be used to make the resulting code as intended.
As newer releases of MOM 2 become available, the above strategy for upgrading is strongly recommended. This method has been in use at GFDL over the past year to incorporate new changes into the development version of MOM 2 as well as to upgrade researchers from older versions to the development version.

An alternative method:

If there is no access to a "gdiff" utility, compare each routine in OLD_UPDATES with the corresponding one in MOM_2_OLD. Do this using "diff" with the "-e" option. For example:

```
diff -e  MOM_2_OLD/grids.F  OLD_UPDATES/grids.F > changes
```

Inspect the "changes" file to view personal modifications. Copy the new grids.F from MOM_2 to NEW_UPDATES. Use an editor to view, OLD_UPDATES/grids.F, "changes" and the new grids.F in NEW_UPDATES. While viewing all three, cut and paste from the "changes" file into NEW_UPDATES/grids.F.

19.16 Bug Fixes: How to get the latest MOM 2

As changes, bug fixes, and new parameterizations are incorporated into MOM 2, a new file *MOM2.2latesttar.Z* containing all changes to date (use "ls -laF MOM2.2latesttar.Z" to see the date) will be placed along side *MOM2.2tar.Z* on the GFDL anonymous ftp server. To upgrade to this latest version, refer to Section 19.15. To inspect what changes have been made, create a NEW directory, then uncompress, and extract the tar file *MOM2.2latesttar.Z* from within this NEW directory to build the latest MOM_2 directory structure. Differences between each pair of directories can be inspected using

```
diff -e  MOM_2 NEW/MOM_2 > diffs_mom2
diff -e  MOM_2/PREP_DATA NEW/MOM_2/PREP_DATA > diffs_prep_data
diff -e  MOM_2/SBC/TIME_MEAN NEW/MOM_2/SBC/TIME_MEAN > diffs_time_mean
diff -e  MOM_2/SBC/MONTHLY NEW/MOM_2/SBC/MONTHLY > diffs_monthly
diff -e  MOM_2/SBC/ATMOS NEW/MOM_2/SBC/ATMOS > diffs_atmos
diff -e  MOM_2/NETCDF NEW/MOM_2/NETCDF > diffs_netcdf
diff -e  MOM_2/EXP/TEST_CASE/MOM_UPDATES NEW/MOM_2/EXP/TEST_CASE/MOM_UPDATES > diffs_momupdates
diff -e  MOM_2/EXP/TEST_CASE/PREP_UPDATES NEW/MOM_2/EXP/TEST_CASE/PREP_UPDATES > diffs_prepupdates
```

How to apply changes from "diff -e"

It is sometimes useful to construct a "newfile" from an "oldfile" plus "changes". If the "changes" were constructed from

```
diff -e oldfile newfile > changes
```

then the "newfile" can be constructed from the "oldfile" using the following C shell script:

```
#!  /bin/csh -f
if ($3 == "") then
echo 'update builds "newfile" from "oldfile" using "changes"'
     echo usage:  update  oldfile  changes  newfile
     exit
endif
set oldfile = $1
set chgs = $2
set newfile = $3
set work = .temp
cat $chgs > $work
echo "w $newfile" >> $work
echo "q $newfile" >> $work
ed $oldfile < $work
/bin/rm $work
```

19.17 Registration for MOM 2

MOM 2 is free. Basically nothing to do except send in your e-mail address if you are not already on the mailing list. The mailing list is kept in the same directory as MOM 2 and will be updated as needed. The purpose of the list is to inform researches of changes to MOM 2. If no longer interested, please request that your name be removed from the list.

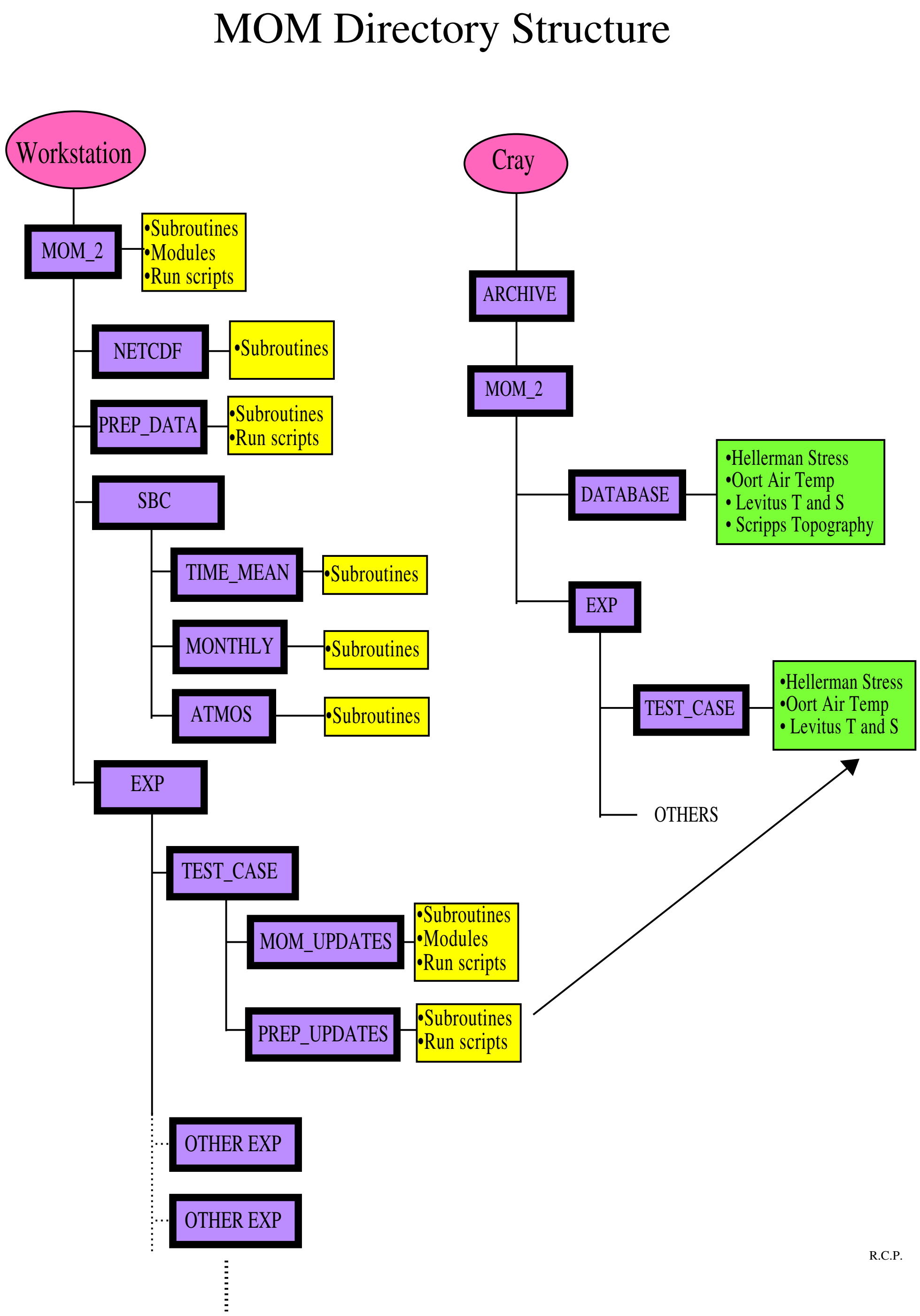


Figure 19.1: Directory structure for MOM 2 at GFDL

Appendix A

Tracer mixing kinematics

In the process of arriving at a new formulation for the isopycnal diffusion scheme in MOM 2, it was useful to understand the basic kinematical properties of tracer mixing when parameterized by a second order mixing tensor. This appendix serves to document certain of these properties which, although perhaps well known by some, were not found to be readily accessible in the literature. In this appendix, only continuum equations will be discussed, and coordinates are referenced to a frame tangent to the geopotential [i.e., (x, y, z)]. The MOM code uses spherical coordinates and the coordinate transformations are straightforward (e.g., Haltiner and Williams, Chapter 1).

The basic equation to be considered here is the tracer mixing equation

$$(\partial_t + \vec{u} \cdot \nabla)\, T = R(T), \tag{A.1}$$

where the mixing operator $R(T)$ is given in an orthogonal coordinate frame by

$$R(T) = \partial_m (J^{mn} \partial_n T). \tag{A.2}$$

In these expressions, the Einstein summation convention is assumed in which repeated indices (m, n) are summed over the three spatial directions. **J** is a second order tensor whose contravariant components are written as J^{mn}. The three-dimensional velocity field $\vec{u}$ is assumed to be divergence-free ($\nabla \cdot \vec{u} = 0$). T represents any tracer such as potential temperature, salinity, or a passive tracer.

In general, this equation represents a parameterization of mixing due to many different processes occurring over a wide range of scales. Therefore, the explicit form for the tensor **J** depends on the particular phenomenon being parametrized. Two basic forms of mixing are considered here as distinguished by the symmetry of the tensor. Either the process is purely dissipative, as represented by a symmetric positive semi-definite diffusion tensor, or purely advective, as represented by an anti-symmetric tensor.

A.1 Basic properties

Consider a mixing process governed by a tensor **J**. This tensor in general contains both symmetric and anti-symmetric components[1] given by

$$J^{mn} = \frac{1}{2}(J^{mn} + J^{nm}) + \frac{1}{2}(J^{mn} - J^{nm}) \equiv K^{mn} + A^{mn}, \tag{A.3}$$

[1] The anti-symmetric component is sometimes called the *skew-symmetric* component in the literature.

where the symmetric part of the tensor is written K^{mn} and the anti-symmetric part as $A^{mn} = -A^{nm}$. For diffusive or dissipative mixing, K^{mn} is symmetric and positive semi-definite. Note that anti-symmetry implies the diagonal terms in A^{mn} vanish. Additionally, anti-symmetry is a frame invariant property of a tensor.

A.1.1 Kinematics of an anti-symmetric tensor

Anti-symmetry introduces constraints on the form of mixing described by anti-symmetric tensors. All these constraints originate from the property that the tensor contraction of a symmetric tensor with an anti-symmetric tensor vanishes. Explicitly, consider a symmetric tensor ($S^{mn} = S^{nm}$) and its contraction (under a flat Euclidean metric) with an anti-symmetric tensor ($A^{mn} = -A^{nm}$).

$$\begin{aligned} S_{mn}A^{mn} &= S^{12}A^{12} + S^{21}A^{21} + S^{13}A^{13} + S^{31}A^{31} + S^{23}A^{23} + S^{32}A^{32} && \text{(A.4)} \\ &= S^{12}A^{12} - S^{12}A^{12} + S^{13}A^{13} - S^{13}A^{13} + S^{23}A^{23} - S^{23}A^{23} && \text{(A.5)} \\ &= 0, && \text{(A.6)} \end{aligned}$$

where the symmetry of S^{mn} and the anti-symmetry of A^{mn} were used. This property implies, for example, that

$$\begin{aligned} A^{mn}\partial_m T \partial_n T &= 0 && \text{(A.7)} \\ A^{mn}\partial_m \partial_n T &= 0 && \text{(A.8)} \\ \partial_m \partial_n A^{mn} &= 0. && \text{(A.9)} \end{aligned}$$

<u>Effective advection velocity</u>

Consider the tracer mixing operator constructed from an anti-symmetric tensor

$$R(T)_A = \partial_m (A^{mn} \partial_n T). \tag{A.10}$$

Property (A.8) implies

$$R(T)_A = (\partial_m A^{mn}) \partial_n T, \tag{A.11}$$

which allows for the identification of a velocity

$$u_A^n \equiv -\partial_m A^{mn}, \tag{A.12}$$

and which brings the anti-symmetric mixing process into the form of an advection

$$R(T)_A = -\vec{u}_A \cdot \nabla T. \tag{A.13}$$

It is important to note that the advection velocity $\vec{u}_A$ is divergence-free

$$\begin{aligned} \nabla \cdot \vec{u}_A &= \partial_n u_A^n && \text{(A.14)} \\ &= -\partial_n \partial_m A^{mn} && \text{(A.15)} \\ &= 0, && \text{(A.16)} \end{aligned}$$

where the last identity used relation (A.9) above. Therefore, tracer mixing as parameterized with an anti-symmetric tensor

$$\frac{DT}{Dt} \equiv (\partial_t + \vec{u} \cdot \nabla)\, T = R(T)_A \tag{A.17}$$

can be written

$$[\partial_t + (\vec{u} + \vec{u}_A) \cdot \nabla] \, T = 0, \tag{A.18}$$

which allows for the identification of an *effective advective transport velocity* (Plumb and Mahlman 1987)

$$\vec{U} \equiv \vec{u} + \vec{u}_A. \tag{A.19}$$

Skew or anti-symmetric flux

Rhines (1986) and Middleton and Loder (1989) discuss the *skew flux* instead of the effective advection velocity. This flux is defined as

$$F_A^m \equiv A^{mn} \partial_n T, \tag{A.20}$$

which allows the skew-symmetric mixing process to be written as

$$(\partial_t + \vec{u} \cdot \nabla) T = \nabla \cdot \vec{F_A}. \tag{A.21}$$

The skew flux is directed orthogonal to the local tracer gradient since

$$\vec{F_A} \cdot \nabla T = F_A^m \partial_m T = A^{mn} \partial_n T \partial_m T = 0. \tag{A.22}$$

A.1.2 Tracer moments

Multiplying the tracer equation

$$(\partial_t + \vec{u} \cdot \nabla) T = \partial_m (J^{mn} \partial_n T) \tag{A.23}$$

by T^{N-1}, where N is some positive integer, yields

$$\begin{aligned}
\frac{1}{N} \partial_t T^N + \frac{1}{N} \nabla \cdot (\vec{u} T^N) &= T^{N-1} \partial_m (J^{mn} \partial_n T) && \text{(A.24)} \\
&= \partial_m (T^{N-1} J^{mn} \partial_n T) - \partial_m T^{N-1} \partial_n T J^{mn} && \text{(A.25)} \\
&= \frac{1}{N} \partial_m (J^{mn} \partial_n T^N) - (N-1) T^{N-2} \partial_m \partial_n J^{mn}. && \text{(A.26)}
\end{aligned}$$

The time tendency is therefore given by

$$\begin{aligned}
\partial_t T^N &= -\nabla \cdot (\vec{u} T^N) + \partial_m (J^{mn} \partial_n T^N) - N(N-1) T^{N-2} \partial_m T \partial_n T J^{mn} && \text{(A.27)} \\
&= \nabla \cdot (\vec{F} T^{N-1} - \vec{u} T^N) - N(N-1) T^{N-2} \partial_m T \partial_n T K^{mn}, && \text{(A.28)}
\end{aligned}$$

where the flux

$$\begin{aligned}
F^m &= J^{mn} \partial_n T && \text{(A.29)} \\
&= K^{mn} \partial_n T + A^{mn} \partial_n T && \text{(A.30)} \\
&\equiv F_K^m + F_A^m && \text{(A.31)}
\end{aligned}$$

was introduced. Note that the above steps assumed a divergence-free velocity field ($\nabla \cdot \vec{u} = 0$) and the identity (A.7) ($A^{mn} \partial_m T \partial_n T = 0$) was employed.

Integrating the previous equation over some domain yields

$$\partial_t \int d\vec{x} \, T^N = \int d\vec{x} \, \left[\nabla \cdot (\vec{F} T^{N-1} - \vec{u} T^N) - N(N-1) T^{N-2} K^{mn} \partial_m T \partial_n T \right]. \tag{A.32}$$

Stokes' Theorem brings this expression to

$$\partial_t \int d\vec{x}\, T^N = \int dA\, \hat{n} \cdot (\vec{F} T^{N-1} - \vec{u} T^N) - N(N-1) \int d\vec{x}\, T^{N-2} K^{mn} \partial_m T \partial_n T, \tag{A.33}$$

where $\hat{n}$ is the outward normal to the boundary. Assuming the normal velocity vanishes at the domain boundary ($\vec{u} \cdot \hat{n} = 0$) yields for the time tendency of the globally integrated $N'th$ tracer moment

$$\partial_t \int d\vec{x}\, T^N = \int dA\, \hat{n} \cdot (\vec{F}_K + \vec{F}_A)\, T^{N-1} - N(N-1) \int d\vec{x}\, T^{N-2} K^{mn} \partial_m T \partial_n T. \tag{A.34}$$

It is of interest to consider some special cases. First, consider the case with no diffusive mixing, which means there is a zero symmetric component to the tracer mixing tensor ($K^{mn} = F^m_K = 0$). Therefore, all moments of the tracer are conserved when there is no normal skew flux at the boundaries ($\vec{F}_A \cdot \hat{n} = 0$), or equivalently there is zero normal induced velocity ($\vec{u}_A \cdot \hat{n} = 0$) at the boundaries. For example, the skew flux associated with the Gent-McWilliams effective transport velocity satisfies these conditions and so preserves all tracer moments (Gent and McWilliams 1990).

For the case with zero anti-symmetric but nonzero symmetric tracer mixing tensor, the first moment, or the total tracer, is conserved in source free regions where there is no tracer flux normal to the boundary; i.e., if $\hat{n} \cdot \vec{F}_K \equiv \hat{n}_m K^{mn} \partial_n T = 0$ at the domain boundaries. In the absence of sources, the tracer's second moment satisfies the evolution equation

$$\partial_t \int d\vec{x}\, T^2 = -2 \int d\vec{x}\, \partial_m T\, K^{mn} \partial_n T. \tag{A.35}$$

Therefore, this mixing tensor dissipates the tracer variance [2] over the source-free domain if the tensor K^{mn} is a positive semi-definite tensor (i.e., the right hand side is negative semi-definite if K^{mn} is positive semi-definite). This property of diffusion is fundamental, and is taken as the foundation for the numerical discretization of isopycnal diffusion in MOM 2. All higher moments of the tracer are also dissipated by diffusion assuming the usual case of a non-negative tracer concentration.

A.2 Horizontal-vertical diffusion

The previous discussion is now specialized to particular forms of mixing used in MOM. The first form is Cartesian or *horizontal-vertical diffusion* in which the principle axes are assumed to be defined by the local tangent to the geopotential surface; i.e., the constant *Eulerian* (x, y, z) frame (the term *z-level* is used in the subsequent). The tracer mixing tensor is diagonal in the (x, y, z) coordinate system and there is no anti-symmetric component. Additionally, the components of the diffusion tensor in the horizontal directions are generally taken to be roughly 10^7 times larger than the vertical components. This large anisotropy arises from the strong vertical stratification in most regions of the ocean which suppresses vertical mixing.

A.3 Isopycnal diffusion

From the standpoint of tracer mixing, the natural set of coordinates are those defined with respect to the isopycnal or neutral directions. These coordinates define what is termed here

[2] The variance of the tracer is given by $var(T) = V^{-1}[\int d\vec{x}\; T^2 - V^{-1}(\int d\vec{x}\; T)^2] \geq 0$, with $V = \int d\vec{x}$ the domain volume. Reducing $\int d\vec{x}\, T^2$ is therefore equivalent to reducing $var(T)$.

the *orthonormal isopycnal frame*. Diffusion in this frame is assumed to be diagonal in the formulation of Redi (1982). The along isopycnal diffusion is on the order of 10^7 times greater than the *diapycnal* diffusion. Refer to Redi (1982), McDougall (1987), Gent and McWilliams (1990) for discussions motivating such diffusive mixing.

This section presents some technical details related to representing the isopycnal diffusion tensor, which is diagonal and thus simple in the *orthogonal isopycnal frame*, in the *Eulerian* (x, y, z) frame which is relevant for MOM. Some of these results are derived by Redi (1982) but using a different formalism.

Basis vectors

Consider a first order tensor, or a vector $\vec{V}$. This object has any number of representations determined by the particular frame of reference. For example, a basis for two frames of interest yield the representations

$$\vec{V} = V^m \vec{e}_m, \tag{A.36}$$

$$= V^{\overline{m}} \vec{e}_{\overline{m}}, \tag{A.37}$$

where the space-time dependent functions V^m and $V^{\overline{m}}$ are the *coordinates* for the vector as represented in the respective frame. There are two sets of basis vectors which define the frames considered here:

$$\vec{e}_1 = \hat{x} \tag{A.38}$$

$$\vec{e}_2 = \hat{y} \tag{A.39}$$

$$\vec{e}_3 = \hat{z}, \tag{A.40}$$

which is the familiar Cartesian unit basis for the z-level frame, and

$$\vec{e}_{\overline{1}} = \frac{\hat{z} \times \nabla\rho}{|\hat{z} \times \nabla\rho|} \tag{A.41}$$

$$\vec{e}_{\overline{2}} = \vec{e}_{\overline{3}} \times \vec{e}_{\overline{1}}, \tag{A.42}$$

$$\vec{e}_{\overline{3}} = \frac{\nabla\rho}{|\nabla\rho|}, \tag{A.43}$$

which defines the orthonormal isopycnal frame determined by the fluctuating geometry of a locally referenced isopycnal surface.

Transforming from the z-level frame to the orthonormal isopycnal frame requires a linear transformation. As a tensor equation, the transformation is written $\vec{e}_{\overline{m}} = \Lambda^m{}_{\overline{m}} \vec{e}_m$. For the purposes of organizing the components of the transformation, this equation can be written as

$$(\vec{e}_{\overline{1}} \;\; \vec{e}_{\overline{2}} \;\; \vec{e}_{\overline{3}}) = (\vec{e}_1 \;\; \vec{e}_2 \;\; \vec{e}_3) \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \tag{A.44}$$

where $\vec{S} = \nabla_\rho z = -z_\rho \nabla_z \rho = (S_x, S_y, 0) = (-\rho_x/\rho_z, -\rho_y/\rho_z, 0)$ is the isopycnal slope vector with magnitude S. Since the transformation is between two orthonormal frames, this transformation matrix is a rotation (unit determinant and inverse given by the transpose) and so can be interpreted in terms of Euler angles (e.g., Redi 1982).

Orthonormal isopycnal frame

The relevance of the orthonormal isopycnal frame arises from the diagonal nature of diffusion within this frame, as assumed by Redi (1982). In general, diffusion is thought to occur predominantly along the two orthogonal directions $\vec{e}_{\overline{1}}$ and $\vec{e}_{\overline{2}}$, which define the neutral directions that are tangent to the locally referenced isopycnal surface. Diffusion in the diapycnal direction $\vec{e}_{\overline{3}}$ typically occurs with a diffusion coefficient which is on the order of 10^{-7} times smaller. Therefore, in this frame the symmetric diffusion tensor takes on the diagonal form

$$K^{\overline{mn}} = \begin{pmatrix} A_I & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_D \end{pmatrix}, \tag{A.45}$$

where A_I are the along isopycnal diffusion coefficients and $A_D \approx 10^{-7} A_I$ is the diapycnal diffusion coefficient [3]. The diffusion tensor written in terms of projection operators takes the form

$$K^{\overline{mn}} = A_I(\delta^{\overline{mn}} - e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}}) + A_D e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}} \tag{A.46}$$

with $\vec{e}_{\overline{3}}$ having the components $(0,0,1)^T$ in the orthonormal isopycnal frame. Explicitly, the diffusion operator in these coordinates is

$$R(T) = \partial_{\vec{e}_{\overline{1}}}(A_I \partial_{\vec{e}_{\overline{1}}} T) + \partial_{\vec{e}_{\overline{2}}}(A_I \partial_{\vec{e}_{\overline{2}}} T) + \partial_{\vec{e}_{\overline{3}}}(A_D \partial_{\vec{e}_{\overline{3}}} T) \tag{A.47}$$

where the diffusion coefficients are generally nonconstant.

z-level frame

The orthogonal isopycnal frame allowed for a simple prescription of the isopycnal diffusion process. In order to describe such a diffusion process in other frames of reference, the components of K must be transformed. Such rules of transformation are standard (e.g., Aris 1962). The diagonal components $K^{\overline{mn}}$ from the orthogonal isopycnal frame are transformed to the z-level frame through

$$K^{mn} = \Lambda^{m}{}_{\overline{m}} K^{\overline{mn}} \Lambda^{n}{}_{\overline{n}}, \tag{A.48}$$

where the transformation matrix is given by equation (A.44). Written as matrices with Λ having components $\Lambda^{n}{}_{\overline{n}}$, this transformation takes the form $K = \Lambda \overline{K} \Lambda^T$, where $\overline{K}$ has the diagonal form given in equation (A.45). Performing the matrix multiplication

$$K^{mn} = \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \begin{pmatrix} A_I & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_D \end{pmatrix} \begin{pmatrix} \frac{S_y}{S} & -\frac{S_x}{S} & 0 \\ \frac{S_x}{S\sqrt{1+S^2}} & \frac{S_y}{S\sqrt{1+S^2}} & \frac{S}{\sqrt{1+S^2}} \\ -\frac{S_x}{\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix}, \tag{A.49}$$

yields the components of the diffusion tensor in the z-level system given by Redi (1982)

$$K^{mn} = \frac{A_I}{(1+S^2)} \begin{pmatrix} 1 + \frac{\rho_y^2 + \epsilon\rho_x^2}{\rho_z^2} & (\epsilon-1)\frac{\rho_x\rho_y}{\rho_z^2} & (\epsilon-1)\frac{\rho_x}{\rho_z} \\ (\epsilon-1)\frac{\rho_x\rho_y}{\rho_z^2} & 1 + \frac{\rho_x^2+\epsilon\rho_y^2}{\rho_z^2} & (\epsilon-1)\frac{\rho_y}{\rho_z} \\ (\epsilon-1)\frac{\rho_x}{\rho_z} & (\epsilon-1)\frac{\rho_y}{\rho_z} & \epsilon + S^2 \end{pmatrix}, \tag{A.50}$$

[3] Since this frame is flat (metric is the unit tensor), there is no distinction between lower and upper labels on the components of a tensor. In general, the convention for tensors such as $K^{\overline{mn}}$, when written as a matrix, is that the first index indicates the row and the second index is for the column.

which can also be written

$$K^{mn} = \frac{A_I}{(1+S^2)} \begin{pmatrix} 1+S_y^2+\epsilon S_x^2 & (\epsilon-1)S_xS_y & (1-\epsilon)S_x \\ (\epsilon-1)S_xS_y & 1+S_x^2+\epsilon S_y^2 & (1-\epsilon)S_y \\ (1-\epsilon)S_x & (1-\epsilon)S_y & \epsilon+S^2 \end{pmatrix}. \tag{A.51}$$

Note that Redi uses the symbol δ instead of S in her expression. The ratio of the diapycnal to isopycnal diffusivities defines the typically small number $\epsilon = A_D/A_I \approx 10^{-7}$. An equivalent form is suggested by writing the tensor in the orthonormal isopycnal frame as

$$K^{\overline{mn}} = A_I \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + A_I(\epsilon-1) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{A.52}$$

which separates the effects of the anisotropy between the along and across isopycnal directions. This expression can be written in the compact form

$$K^{\overline{mn}} = K_{\overline{mn}} = A_I[\delta_{\overline{mn}} - (\vec{e}_{\overline{3}})_{\overline{m}}(\vec{e}_{\overline{3}})_{\overline{n}}] + A_D(\vec{e}_{\overline{3}})_{\overline{m}}(\vec{e}_{\overline{3}})_{\overline{n}}\,, \tag{A.53}$$

as noted in equation (A.46). Transforming to the z-level frame yields

$$K^{mn} = A_I \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \frac{A_I(\epsilon-1)}{\rho_x^2+\rho_y^2+\rho_z^2} \begin{pmatrix} \rho_x^2 & \rho_x\rho_y & \rho_x\rho_z \\ \rho_x\rho_y & \rho_y^2 & \rho_y\rho_z \\ \rho_x\rho_z & \rho_y\rho_z & \rho_z^2 \end{pmatrix}, \tag{A.54}$$

which can also be written

$$K^{mn} = K_{mn} = A_I\delta_{mn} + A_I(\epsilon-1)\frac{\partial_m\rho\,\partial_n\rho}{|\nabla\rho|^2} = A_I(\delta_{mn} - \hat{\rho}_m\hat{\rho}_n) + A_D\hat{\rho}_m\hat{\rho}_n, \tag{A.55}$$

where $\hat{\rho}_m = \partial_m\rho/|\nabla\rho|$ are the components of the diapycnal unit vector $\vec{e}_{\overline{3}}$ written in the z-level frame [see equation (A.43)]. Note that equation (A.55) could have been written down immediately once equation (A.53) was hypothesized, and the unit vectors (A.41), (A.42), and (A.43) were determined.

Small angle approximation

There have been no small angle approximations (i.e., $S << 1$) made in computing the representation K^{mn}. Therefore, for modeling the physical process of diffusion employing the isopycnal diffusive mixing hypothesis in the z-level frame, K^{mn} given here is *the* form of the diffusion tensor to be used. In so doing, the physics of diapycnal diffusion is isolated in the parameter ϵ. The small angle approximation of Cox (1987) is an additional statement regarding the behavior of the bulk of the ocean (that isopycnals have only a rather small slope except in convection regions). This approximation recovers the form quoted in Gent and McWilliams (1990)

$$K^{mn}_{\text{small}} = A_I \begin{pmatrix} 1 & 0 & (\epsilon-1)\frac{\rho_x}{\rho_z} \\ 0 & 1 & (\epsilon-1)\frac{\rho_y}{\rho_z} \\ (\epsilon-1)\frac{\rho_x}{\rho_z} & (\epsilon-1)\frac{\rho_y}{\rho_z} & \epsilon+S^2 \end{pmatrix} = A_I \begin{pmatrix} 1 & 0 & (1-\epsilon)S_x \\ 0 & 1 & (1-\epsilon)S_y \\ (1-\epsilon)S_x & (1-\epsilon)S_y & \epsilon+S^2 \end{pmatrix} \tag{A.56}$$

Note that Cox (1987) originally retained the $(1,2) = (2,1)$ elements equal to $-S_xS_y$. In the small angle approximation, however, this term is negligible. Additionally, and most crucially, if this term is retained, the small angle tensor will diffuse locally referenced potential density whereas the full tensor will not. Hence, it is important to use the physically consistent form of the small angle approximation which drops the $(1,2) and (2,1)$ elements.

As the small angle approximation is commonly used, it is perhaps interesting to ask the following question: Given the small angle approximation representation of the diffusion tensor in the z-level frame, what is the representation of this tensor in the orthonormal isopycnal frame? This tensor cannot be the diagonal form given in equation (A.45) since that form transformed into the full non-small angle representation of equation (A.50). Using the full transformation back to the orthonormal isopycnal frame, $K^{\overline{mn}}_{\text{small}} = \Lambda^{\overline{m}}{}_{m} K^{mn}_{\text{small}} \Lambda^{\overline{n}}{}_{n}$, or in matrix form

$$K^{\overline{mn}}_{\text{small}} = A_I \begin{pmatrix} \frac{S_y}{S} & -\frac{S_x}{S} & 0 \\ \frac{S_x}{S\sqrt{1+S^2}} & \frac{S_y}{S\sqrt{1+S^2}} & \frac{S}{\sqrt{1+S^2}} \\ -\frac{S_x}{\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \times \tag{A.57}$$

$$\begin{pmatrix} 1 & 0 & (1-\epsilon)S_x \\ 0 & 1 & (1-\epsilon)S_y \\ (1-\epsilon)S_x & (1-\epsilon)S_y & \epsilon + S^2 \end{pmatrix} \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \tag{A.58}$$

yields the orthonormal isopycnal frame representation of the small angle approximated tensor[4]

$$K^{\overline{mn}}_{\text{small}} = \begin{pmatrix} A_I & 0 & 0 \\ 0 & A_I(1+S^2) & 0 \\ 0 & 0 & A_D \end{pmatrix} + \frac{A_D S^2}{1+S^2} \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & S \\ 0 & S & 1 \end{pmatrix}. \tag{A.59}$$

The small angle approximation is seen to add a small amount of along isopycnal mixing (the $(2,2)$ term $A_I(1+S^2)$) as well as a term proportional to the generally small number $A_D S^2$. Dropping these terms is consistent with the small angle approximation, which then recovers the purely diagonal mixing tensor $K^{\overline{mn}} = A_I(\delta^{\overline{mn}} - e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}}) + A_D e^{\overline{m}}_{\overline{3}} e^{\overline{n}}_{\overline{3}}$.

Errors with z-level mixing

Consider the traditional diffusive mixing (Section A.2) with some tensor **I** that is diagonal in the z-level frame. Diffusive mixing with **I** is quite different than diffusive mixing with **K**, as can be seen clearly by transforming **I** to the orthonormal isopycnal frame

$$I^{\overline{mn}} = \begin{pmatrix} \frac{S_y}{S} & -\frac{S_x}{S} & 0 \\ \frac{S_x}{S\sqrt{1+S^2}} & \frac{S_y}{S\sqrt{1+S^2}} & \frac{S}{\sqrt{1+S^2}} \\ -\frac{S_x}{\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix} \begin{pmatrix} A_H & 0 & 0 \\ 0 & A_H & 0 \\ 0 & 0 & A_V \end{pmatrix} \begin{pmatrix} \frac{S_y}{S} & \frac{S_x}{S\sqrt{1+S^2}} & -\frac{S_x}{\sqrt{1+S^2}} \\ -\frac{S_x}{S} & \frac{S_y}{S\sqrt{1+S^2}} & -\frac{S_y}{\sqrt{1+S^2}} \\ 0 & \frac{S}{\sqrt{1+S^2}} & \frac{1}{\sqrt{1+S^2}} \end{pmatrix}, \tag{A.60}$$

which yields

$$I^{\overline{mn}} = \frac{A_H}{1+S^2} \begin{pmatrix} 1+S^2 & 0 & 0 \\ 0 & 1+\tilde{\epsilon}S^2 & -S(1-\tilde{\epsilon}) \\ 0 & -S(1-\tilde{\epsilon}) & \tilde{\epsilon}+S^2 \end{pmatrix}, \tag{A.61}$$

where $\tilde{\epsilon} = A_V/A_H$ is the ratio of the vertical to horizontal diffusion coefficient.

Therefore, diffusive mixing with **I** in the z-level frame introduces first order in slope errors in the off diagonal terms, whereas the diagonal terms contain second order in slope errors. The error in the $(3,3)$ component, however, is the most relevant as it represents an added source of

[4]Note the rotation need not transform the $\hat{x} \leftrightarrow \hat{y}$ symmetry present in the (x,y,z) form of the small angle mixing tensor into a $\vec{e}_{\overline{1}} \leftrightarrow \vec{e}_{\overline{2}}$ symmetry in the $(\vec{e}_{\overline{1}}, \vec{e}_{\overline{2}}, \vec{e}_{\overline{3}})$ form. The (x,y) coordinate symmetry, however, is preserved.

diapycnal mixing (i.e., a *false diapycnal mixing*). For example, with the usual diffusivity ratio $\tilde{\epsilon} \approx 10^{-7}$, modest slopes $S \approx 3\times10^{-4}$ are sufficient to add diapycnal mixing through the S^2 term which is on the same order as $\tilde{\epsilon}$. It is for this reason that horizontal mixing, especially in regions of the ocean with larger than modest slopes but still within the small slope approximation, is incompatible with the hypothesis that mixing is predominantly along the isopycnal directions.

As seen above, the distinction between horizontal and along isopycnal mixing is quite important. However, the distinction between vertical mixing and diapycnal mixing is not generally important except for extremely large slopes. The reason for this ambiguity can be easily understood by looking at the small angle approximation to the isopycnal tensor [equation (A.56], and setting A_I to zero in order to focus on the diapycnal piece

$$K^{mn}(A_I = 0)_{\text{small}} = A_D \begin{pmatrix} 0 & 0 & -S_x \\ 0 & 0 & -S_y \\ -S_x & -S_y & 1 \end{pmatrix}. \tag{A.62}$$

The off diagonal terms represent the difference between vertical mixing and diapycnal mixing. These terms are down by a single factor of the slope. To completely determine the error introduced by neglecting these terms, it is useful to write the diffusion operator $R(T)$ arising from this tensor and performing a scaling within the small slope approximation. For this purpose, consider the horizontal direction of steepest slope to be the x-direction. The y-direction will therefore be ignored. Also assume a constant diapycnal mixing coefficient A_D. With these approximations, the diffusion operator takes the form

$$\begin{aligned} R(T) &= A_D[\partial_z(S_x\partial_x T) + \partial_z\partial_z T] \\ &\approx A_D\left(\frac{\Delta T}{(\Delta x)^2} + \frac{\Delta T}{(\Delta z)^2}\right) \\ &\approx A_D\frac{\Delta T}{(\Delta z)^2}, \end{aligned} \tag{A.63}$$

where the last approximation assumed $\Delta z << \Delta x$. Hence, the off-diagonal pieces in the diffusion tensor are quite negligible in the small slope approximation. Therefore, with this scaling, the distinction between vertical and diapycnal mixing is negligible as well.

A.4 Symmetric and anti-symmetric tensors

The horizontal-vertical and isopycnal mixing of the previous two sections employed symmetric positive semi-definite tensors representing dissipative mixing processes. The presence of an anti-symmetric component to the mixing tensor, as seen in Section A.1, can be thought of as an added advective transport of tracers [see in particular the equations leading up to (A.19)]. The transport velocity defined by such a tensor is divergence-free [equation (A.16)] and preserves all tracer moments if it has zero normal value on the boundaries [discussion following equation (A.34)]. The parameterization of Gent and McWilliams (1990) has been reformulated in terms of such an effective transport velocity by Gent et. al., (1995). Other types of mixing can be parameterized by anti-symmetric mixing tensors. For example, Middleton and Loder (1989) summarize the effects of mixing by non-zero angular momentum waves which gives rise to skew-fluxes.

A.5 Summary

The traditional manner of parameterizing tracer mixing in ocean models is through the use of a second order mixing tensor. The basic mathematical properties of such mixing have been reviewed in this appendix. In general, the mixing tensor contains both a symmetric positive-semi-definite component and an anti-symmetric component. The symmetric component represents dissipative processes associated with diffusive mixing. The numerical discretization of this process is discussed further in Appendix B. The anti-symmetric component can be formulated in terms of an advective velocity which, in addition to the usual Eulerian velocity, contributes to conservative tracer mixing.

Because there will always be mixing processes too small to resolve in ocean models, there will always be a need to rationally parameterize these processes. Determining the nature of mixing tensors relevant for these parameterizations, if such objects exist, is proving to be one of the most challenging problems in physical oceanography today.

Appendix A contributed by
Stephen M. Griffies
smg@gfdl.gov
Last revised October 31, 1996

Appendix B

Isopycnal diffusion

This appendix provides details for the discretization of iso-neutral or isopycnal diffusion of tracers in MOM 2. Although some motivation is given here, the reader is referred to a manuscript currently under preparation by Griffies, Gnanadesikan, Pacanowski, and Larichev (hereafter, Griffies et al., 1997) for a more complete discussion. A draft of this manuscript will be made available sometime late Autumn 1996.

In arriving at the formulation given here, extensive benefit was derived from interactions with John Dukowicz and Rick Smith at Los Alamos. They developed much of the *Functional Formalism* used here and applied it to the discretization of the small angle isopycnal diffusion tensor in the Los Alamos Parallel Ocean Program (POP).[1] However, there are some important differences between the POP implementation and that in MOM2 which have to do with the computation of the neutral directions and the discretization of the diffusion coefficients.

Given the importance of isopycnal diffusion in ocean models and the fundamental changes that the new scheme represents, it is useful to provide the modelling community with complete details of the derivation. Therefore, it is hoped that those who are interested will carefully consider the discussion given in Griffies et al (1997) and in this appendix in order to expose any problems which may exist. At this stage of numerically testing the scheme, all appears with it.

Please note that in this Appendix, diagonal diffusive fluxes are considered positive if they are down the tracer gradient. The code in MOM 2, however, employs the opposite convention. The diffusion operator in MOM 2 picks up a minus sign which renders it the same as discussed in this Appendix.

B.0.1 Functional formalism

The fundamental mathematical property that is exploited in this formalism is that the diffusion operator can be associated with a negative semi-definite functional (Courant and Hilbert). For example, Laplacian diffusion in an isotropic media, $R(T) = \nabla^2 T$, is identified with the functional derivative $R(T) = \delta \mathbf{S}/\delta T$, where $\mathbf{S} \equiv -\int |\nabla T|^2 d\vec{x} \leq 0$. As shown in Section B.1, the negative semi-definiteness of the functional $\mathbf{S}$ is related to the dissipative property of the diffusion operator; i.e., one implies the other. On the lattice, not every consistent[2] discrete diffusion operator corresponds to a negative semi-definite discrete functional. Therefore, a consistent numerical diffusion operator does not necessarily possess the dissipative properties of the continuum operator. For the Laplacian in an isotropic media, it is trivial to produce

[1]Besides the unpublished notes of Dukowicz and Smith, I have only been able to find discussion of this approach for finite difference equations in the Soviet literature: see Goloviznin et al. (1977), Tishkin et al. (1979), and Korshiya et al. (1980).

[2]Consistent in that the discretization reduces to the correct continuum operator as the grid size goes to zero.

a dissipative numerical operator. In the anisotropic case, such as isopycnal diffusion, it is nontrivial. Indeed, the original discretization of the isopycnal diffusion operator in the GFDL model is numerically consistent but not dissipative. The approach taken in the derivation of the new discretization of this operator is to focus on discretizing the functional first and then to take the discrete version of the functional derivative in order to derive the discrete diffusion operator. This approach ensures that the discretized operator is dissipative, no matter how the functional is discretized.

B.0.2 Neutral directions

The functional formalism provides a powerful framework to discretize the isopycnal diffusion operator. Within this framework, provision is made for a discretization of the diffusion fluxes which are aligned according to the best approximation of the neutral directions. These considerations imply that the density gradients must be evaluated in terms of the active tracer gradients in order to provide for a zero along isopycnal flux of locally referenced potential density. Special care must be taken when choosing the reference points for evaluating these gradients, and the details are given in Section B.2.6. As shown in Griffies etal (1997), without a proper discretization which guarantees a zero flux of locally referenced potential density, the isopycnal diffusion operator will be unstable, even if it ensures variance does not increase. So both properties are essential.

B.0.3 Full isopycnal diffusion tensor

Traditionally, the implementation of isopycnal diffusion has been with the small slope approximation made to the full tensor (Cox 1987, Gent and McWilliams 1990). This approximation is justified in a large part of the world ocean since slopes larger than 1/100 are thought to be uncommon. However, the restrictions placed on the numerical realization of slopes larger than 1/100 are *ad hoc* and do affect the solution's integrity. In particular, the assumptions one makes about the slope checking can significantly influence the rates of ventilation. As models become more advectively dominant, as can now be realized with advection schemes such as FCT, the slopes associated with strong currents can reach greater than 1/100 in many reagions not associated with convection. Additionally, recent studies indicate that the effects of mesoscale eddies are important in regions near areas of active deep convection (e.g., Send and Marshall, 1995). Therefore, it may prove important to allow for a full realization of isopycnal diffusion for any slope. The discretization of the full Redi diffusion tensor provides a tool to help understand the physics of steep-slope mixing of tracers, without making any *ad hoc* assumptions about the strength of the along isopycnal diffusion. Namely, for certain grids, the discretized full tensor allows for stable diffusion without slope checking. For these reasons, both the full and small angle isopycnal diffusion tensors have been implemented in MOM2. It is hoped that the discretization of both tensors will allow the physics of tracer mixing in regions of intermediate to steep isopycnal slopes to be accessible to rational numerical investigations with MOM2.

B.1 Functional formalism in the continuum

Before proceding to the discretization, it is useful to discuss some functional calculus in the continuum.

B.1.1 The functional for a general diffusion operator

The diffusion operator can be written in two ways

$$\begin{aligned} R(T) &= \partial_m (K^{mn} \partial_n T) && \text{(B.1)} \\ &= -\nabla \cdot \vec{F}. && \text{(B.2)} \end{aligned}$$

The first form emphasizes the symmetric positive semi-definite diffusion tensor K^{mn} and the second emphasizes the diffusion fluxes $\vec{F}$ which are directed down the tracer gradient. The corresponding functional is written

$$\begin{aligned} \mathbf{S} &= -\int d\vec{x}\, \partial_m T\, K^{mn} \partial_n T && \text{(B.3)} \\ &= \int d\vec{x}\, \nabla T \cdot \vec{F} && \text{(B.4)} \\ &\equiv 2 \int d\vec{x}\, \mathbf{L}, && \text{(B.5)} \end{aligned}$$

where the last expression introduces the negative semi-definite quadratic form

$$\mathbf{L} = -\frac{1}{2} \partial_m T\, K^{mn} \partial_n T = \frac{1}{2} \nabla T \cdot \vec{F}. \tag{B.6}$$

Note that $\mathbf{L} \le 0$ is another way of stating that the diffusive flux $\vec{F}$ is directed down the tracer gradient ∇T. This result does not imply that each component of the diffusive flux $F^n = -\partial_m T K^{mn}$ is directed down the corresponding tracer gradient. Namely, for any particular direction, say the x-direction, $F^x \partial_x T > 0$, which means the flux is directed up the x-gradient of the tracer. This particular upgradient transfer is compensated by stronger downgradient transfers in the orthogonal directions which provides for the scalar product, representing the full projection of the flux vector onto the gradient of the tracer, to be negative semi-definite.

It is illustrative to verify the correspondence between the functional $\mathbf{S}$ and the diffusion operator $R(T)$. The first variation of $\mathbf{S}$ under the effects of an infinitesimal variation of the tracer $T \to T + \delta T$, where $\delta T = 0$ on the boundaries, is given by (Courant and Hilbert)

$$\delta\, \mathbf{S} = \int d\vec{x}\, [\frac{\delta \mathbf{L}}{\delta T} - \partial_m (\frac{\delta \mathbf{L}}{\delta T_m})] \delta T, \tag{B.7}$$

implying that the functional derivative is given by

$$\frac{\delta \mathbf{S}}{\delta T} = \frac{\delta \mathbf{L}}{\delta T} - \partial_m (\frac{\delta \mathbf{L}}{\delta T_m}). \tag{B.8}$$

Using the explicit form for the quadratic form $\mathbf{L} = -\frac{1}{2} \partial_m T K^{mn} \partial_n T$ renders

$$\frac{\delta \mathbf{S}}{\delta T} = \partial_m (K^{mn} \partial_n T) = R(T). \tag{B.9}$$

B.1.2 Functional as the source for variance tendency

In addition to providing a direct connection to the diffusion operator through its functional derivative, the functional $\mathbf{S}$ can be seen to be proportional to the tracer variance. To see this connection, consider the tendency for the squared tracer

$$\begin{aligned} \frac{1}{2} \partial_t T^2 &= -\nabla \cdot (T \vec{F}) + \nabla T \cdot \vec{F} && \text{(B.10)} \\ &= \partial_m (T\, K^{mn} \partial_n T) + 2\mathbf{L}. && \text{(B.11)} \end{aligned}$$

Integrating the squared tracer equation over a source-free domain yields an expression for the time tendency of the tracer variance

$$\partial_t \int d\vec{x}\, T^2 = 4\mathbf{S} \leq 0. \tag{B.12}$$

Hence, $4\mathbf{S}$ is the sink of tracer variance arising from the effects of downgradient diffusion. It is this reduction of tracer variance which is a fundamental property of dissipative tracer mixing; i.e., of diffusion. The discrete analog to this result says that by ensuring that the discretization of the functional $\mathbf{S}$ is negative semi-definite, the corresponding tracer variance will be reduced in source free regions of the lattice. By then taking the functional derivative of the discrete functional $\mathbf{S}$, it is ensured that the discrete diffusion operator $R(T)$ has the desired dissipative properties. This result provides the motivation for the formalism.

B.1.3 The functional for isopycnal diffusion

The diffusion tensor representing along and across isopycnal mixing is given in the projection operator form as

$$\begin{aligned} K^{mn} &= A_I(\delta^{mn} - \hat{\rho}^m \hat{\rho}^n) + A_D \hat{\rho}^m \hat{\rho}^n \\ &= (A_I - A_D)(\delta^{mn} - \hat{\rho}^m \hat{\rho}^n) + A_D \delta^{mn}. \end{aligned} \tag{B.13}$$

with δ^{mn} the Kronecker delta and $\hat{\rho} = \nabla\rho/|\nabla\rho|$ the unit vector normal to the isopycnal. The along (or adiabatic) and across (or diabatic) diffusion coefficients A_I and A_D are non-negative and can in general be functions of space-time. When acting on a vector, the adiabatic piece of the diffusion tensor projects the vector onto the local isopycnal direction, and the diabatic piece projects the vector into the diabatic direction.

Given this diffusion tensor, the functional for isopycnal diffusion is written

$$\mathbf{S} = -\frac{1}{2}\int d\vec{x}\, (A_I - A_D)\frac{|\nabla\rho \wedge \nabla T|^2}{|\nabla\rho|^2} - \frac{1}{2}\int d\vec{x}\, A_D |\nabla T|^2, \tag{B.14}$$

or more explicitly,

$$\begin{aligned} \mathbf{S} = \; &- \frac{1}{2}\int d\vec{x}\, (A_I - A_D)\frac{(T_x\rho_y - T_y\rho_x)^2 + (T_y\rho_z - T_z\rho_y)^2 + (T_z\rho_x - T_x\rho_z)^2}{\rho_x^2 + \rho_y^2 + \rho_z^2} \\ &- \frac{1}{2}\int d\vec{x}\, A_D(T_x^2 + T_y^2 + T_z^2). \end{aligned} \tag{B.15}$$

The small slope approximation amounts to taking the limit of $|\rho_x|, |\rho_y| << |\rho_z|$, and dropping terms of order $slope * (A_D/A_I)$, with *slope* the small isopycnal slope. The resulting functional is

$$\mathbf{S}^{small} = -\frac{1}{2}\int d\vec{x}\, A_I \frac{(T_y\rho_z - T_z\rho_y)^2 + (T_z\rho_x - T_x\rho_z)^2}{\rho_z^2} - \frac{1}{2}\int d\vec{x}\, A_D (T_z)^2. \tag{B.16}$$

B.1.4 Continuum diffusive fluxes

For comparison with the results in the discrete case, it is useful to obtain the continuum diffusion operator using the relation

$$R(T) = \nabla T \cdot \vec{F} = -\partial_m\left(\frac{\delta\mathbf{L}}{\delta T_m}\right) \tag{B.17}$$

With the quadratic form defined in equation (B.15), the diffusive fluxes are given by

$$-F^x = -\frac{\delta \mathbf{L}}{\delta T_x} = (A_I - A_D)\frac{\rho_y(T_x\rho_y - T_y\rho_x) + \rho_z(T_x\rho_z - T_z\rho_x)}{|\nabla\rho|^2} + A_D T_x, \quad (B.18)$$

$$-F^y = -\frac{\delta \mathbf{L}}{\delta T_y} = (A_I - A_D)\frac{\rho_z(T_y\rho_z - T_z\rho_y) + \rho_x(T_y\rho_x - T_x\rho_y)}{|\nabla\rho|^2} + A_D T_y, \quad (B.19)$$

$$-F^z = -\frac{\delta \mathbf{L}}{\delta T_z} = (A_I - A_D)\frac{\rho_x(T_z\rho_x - T_x\rho_z) + \rho_y(T_z\rho_y - T_y\rho_z)}{|\nabla\rho|^2} + A_D T_z. \quad (B.20)$$

The first term in these expressions vanishes when the tracer is parallel to the isopycnal. For most oceanographic cases, the difference $A_I - A_D \approx A_I$ to many orders of accuracy. This assumption is made in the model implementation of the full isopycnal diffusion tensor. Note the presence of the small diabatic pieces $A_D T_x$ and $A_D T_y$. In the case of very steep isopycnals, these terms become relevant.

The small angle result is

$$-F^x = A_I(T_x - T_z\frac{\rho_x}{\rho_z}) \quad (B.21)$$

$$-F^y = A_I(T_y - T_z\frac{\rho_y}{\rho_z}) \quad (B.22)$$

$$-F^z = A_I\left(-T_x\frac{\rho_x}{\rho_z} - T_y\frac{\rho_y}{\rho_z} + T_z\frac{\rho_x^2 + \rho_y^2}{\rho_z^2}\right) + A_D T_z, \quad (B.23)$$

These are the forms for the diffusive fluxes which will come out most naturally in the discretized case discussed in the next section.

B.2 Discretization of the diffusion operator

In various iterations with this derivation, it was found that the form of the functional given by equation (B.15) is most convenient to discretize as it allows for projection onto two-dimensional planar regions rather than having to consider a full three-dimensional discretization as might be necessary for discretizing the full tensor. Also, the derivation of the full tensor's discretizaton turned out to be easier than the small tensor because of the higher degree of symmetry with the full tensor. Therefore, the derivation of the full tensor is presented in these notes with the small slope results obtained from the small slope limits of the full tensor. An independent derivation starting from a discretization of the small slope version of the functional (equation (B.16)) verifies the validity of the small slope limit from the full tensor. Additionally, the derivation is presented for the MOM2 default grid in which for nonuniform grids, the T-point is not in the center of the T-cell. The form of the discretized operator is dependent on this choice of T-cell placement. As of this writing, only the MOM2 default grid discretization of the diffusion operator has been implemented. Therefore, it is recommended that one not use the option **t_center**.

As seen in the discussion from Section B.1, the discretization of the diffusion operator at a particular grid point is derived from the functional derivative of the discretized functional

$$R(T)_{i,k,j} = \frac{1}{V_{T_{i,k,j}}}\frac{\partial S[T_{i,k,j}]}{\partial T_{i,k,j}}. \quad (B.24)$$

It is necessary to divide out the volume of the T-cell in the discrete case since with finite cell volumes, the appropriate delta function which occurs in the derivative is the dimensionless

Kronecker delta rather than the dimensionful (dimensions 1/volume) Dirac delta which appears in the continuum case. The procedure, therefore, is to identify those pieces of the discretized functional which contain contributions from the discretized tracer value $T_{i,k,j}$, as these are the pieces which contribute to the diffusion operator for this T-cell. This enumeration depends on the particular discretization of the functional. One overriding principle used to guide this discretization, as emphasized by Dukowicz and Smith, is to recover the familiar discretization of the Laplacian (5-point in the two-dimensional case) in the case of flat isopycnals. Furthermore, all details about the discretization will be made at the level of the functional. These details include the particular grid choice (as motivated from Smith and Dukowicz), and the choice for reference points to be used in approximating the neutral directions.

In the following, no explicit reference will be made to the time step. As it is necessary to lag the time step by one step for numerical stability of the diffusion equation, a time step of $\tau - 1$ will be implicit throughout.

B.2.1 A one-dimensional warm-up

In order to illustrate the general framework provided by the functional approach, it is useful to consider the trivial case of one-dimensional diffusion. It should be noted that in one-dimension, there is no issue of isopycnal diffusion and so this example cannot illustrate any of the subtle issues related to the discretization of the neutral directions. Those issues will be discussed at the appropriate point in the derivation of the three-dimensional case.

Figure B.1 shows the grid, which corresponds to the x-axis in Figure B.2. Let $V(n)$ and $A(n)$ be the volume and diffusion coefficient corresponding to the subcell n. Consider the following discretization of the functional

$$S = -\frac{1}{2}\sum_i \sum_n A(n)V(n)(\delta_x T(n))^2, \tag{B.25}$$

where the n sum is over the subcells relevant for each T-cell. In particular, the four terms containing a contribution from T_i are given by

$$\begin{aligned} 2S[T_i] = &- A(7)V(7)(\delta_x T_{i-1})^2 - A(1)V(1)(\delta_x T_{i-1})^2 \\ &- A(2)V(2)(\delta_x T_i)^2 - A(5)V(5)(\delta_x T_i)^2, \end{aligned} \tag{B.26}$$

where the tracer derivative is assumed the same for the two subcells 7 and 1 and the two subcells 2 and 5. The volumes of the subcells are

$$V(1) = V(7) = \frac{dxu_{i-1}}{2} \tag{B.27}$$

$$V(2) = V(5) = \frac{dxu_i}{2}, \tag{B.28}$$

and the derivatives are

$$\delta_x T_{i-1} = \frac{T_i - T_{i-1}}{dxu_{i-1}} \tag{B.29}$$

$$\delta_x T_i = \frac{T_{i+1} - T_i}{dxu_i}. \tag{B.30}$$

The derivative of the functional is

$$\frac{\partial S}{\partial T_i} = -\frac{A(7)V(7)}{dxu_{i-1}}\,\delta_x T_{i-1} - \frac{A(1)V(1)}{dxu_{i-1}}\,\delta_x T_{i-1} + \frac{A(2)V(2)}{dxu_i}\,\delta_x T_i + \frac{A(5)V(5)}{dxu_i}\,\delta_x T_i, \tag{B.31}$$

which can be rearranged to

$$\begin{aligned}\frac{\partial S}{\partial T_i} &= \overline{A(2)}^x \delta_x T_i - \overline{A(7)}^x \delta_x T_{i-1} && \text{(B.32)}\\ &= dxt_i \, \delta_x \left(\overline{A(7)}^x \delta_x T_{i-1}\right), && \text{(B.33)}\end{aligned}$$

yielding the discretized diffusion operator acting on T_i

$$\frac{1}{dxt_i}\frac{\partial S}{\partial T_i} = R(T)_i = \delta_x \left(\overline{A(7)}^x \delta_x T_{i-1}\right). \tag{B.34}$$

The averaging operation for the diffusion coefficient is given by

$$\begin{aligned}\overline{A(2)}^x &= \frac{A(2)+A(5)}{2} && \text{(B.35)}\\ \overline{A(7)}^x &= \frac{A(7)+A(1)}{2}. && \text{(B.36)}\end{aligned}$$

It is this averaging which is the only difference from the discretization derived using more traditional approaches. In the isopycnal case, the freedom to define the diffusion coefficient differently for each of the subcells will be an important element in ensuring stability of the scheme for large isopycnal slopes.

The main points to be taken from this example are (A) The assumption that the tracer gradient is the same across the two adjacent subcells 1 & 7 and 2 & 5, respectively, (B) The presence of an average operator which arises from the recombination of terms, and (C) The freedom to prescribe a different diffusion coefficient to each of the sub-cells. The assumption (A) was necessary in order to derive the traditional 3-point Laplacian starting from the functional. These points, in somewhat more elaborate forms, will also arise in the discretization of the isopycnal diffusion operator.

B.2.2 Grid partitioning

Using the functional in the form given in equation (B.15) for the full tensor, or equation (B.16) for the small tensor, motivates a discretization which projects separately onto the three planar slices x-y, y-z, and z-x. The only derivative in the orthogonal direction is within the $|\nabla\rho|^{-2}$ piece in the full tensor. The discretization of the $A_D|\nabla T|^2$ term can be done using the traditional 5-point Laplacian and will not be discussed further.

Figures B.2 and B.3 show the projections for the x-y and z-x planes as defined on the MOM2 default grid. The y-z plane is similar. Within a plane, the central T-cell is partitioned into 4 generally non-equal squares, with one corner being the T-point and the other corners being the corners of the T-cell. The sides of these squares have dimensions 1/2 the relevant U-cell distance since the T-cell in the MOM2 default can generally be off centered. There are an additional 8 quarter cells which surround the 4 central cells, each of size determined by the 1/2 U-cell distances and each of which has one corner lying at a T-point. In the discretization considered here, these are the 12 cells for a particular x-y or z-x projection which contain contributions from the central T-point. They correspond to the 4 subcells discussed in the previous one-dimensional example. For the three planes, there are a total of 36 quarter cells to which $T_{i,k,j}$ contributes. The direction perpendicular to the plane has distance given by the T-distance, either dzt_k for the x-y plane, dxt_i for the y-z plane, or dyt_j for the z-x plane. Inside of these quarter cells, it is possible to prescribe a different diffusion coefficient. This added freedom in choosing the diffusion coefficients is essential to the method employed for maintaining the numerical stability when the slopes get large. These details for slope checking will be given in Section B.2.7

When discretizing the derivatives appearing in the functional, the derivative inside a triangle determined by adjacent T-points will be taken as constant. As seen in the one-dimensional example, this assumption is essential to reducing to the familiar discretization of the Laplacian in the flat isopycnal case (5-points in a two-dimensional example).

B.2.3 Subcell volumes

On a spherical earth, the areas in the x-y plane for a domain of longitudinal width $\Delta\lambda$ and latitudinal width $\phi_2 - \phi_1$ is given by $(a\Delta\lambda)(a \sin\phi_2 - a\sin\phi_1)$, where a is the radius of the earth. For box 1 in Figure B.2, for example, $a\Delta\lambda = (1/2)dxu_i$ and so the volume of box 1 is

$$V(1) = (1/2)dxu_i(a\sin\phi_j^U - a\sin\phi_j^T)dzt_k. \tag{B.37}$$

Taking $\phi_j^T = \phi_j^U - dyu_j/(2a)$ gives $\sin\phi_j^T = \sin\phi_j^U - \cos\phi_j^U dyu_j/(2a) + O(dyu_j/(2a))^2$. Hence, within this local β-plane approximation, the volume of box 1 is given by

$$V(1) = (1/4)dxu_i\ dyu_j\ \cos\phi_j^U\ dzt_k. \tag{B.38}$$

Since the variance reduction property desired is not sacrificed by making this approximation, the volumes of the 12 subcells in the x-y plane will be taken to be

$$\begin{aligned}
V(1) = V(7) = V(9) &= \frac{1}{4}dxu_{i-1}\cos\phi_j^U\ dyu_j\ dzt_k\\
V(2) = V(5) = V(10) &= \frac{1}{4}dxu_i\cos\phi_j^U\ dyu_j\ dzt_k\\
V(3) = V(8) = V(11) &= \frac{1}{4}dxu_{i-1}\cos\phi_{j-1}^U\ dyu_{j-1}\ dzt_k\\
V(4) = V(6) = V(12) &= \frac{1}{4}dxu_i\cos\phi_{j-1}^U\ dyu_{j-1}\ dzt_k.
\end{aligned} \tag{B.39}$$

For the z-x plane, there are no subtleties related to the spherical earth. Without approximation, the 12 subcells have the volumes

$$\begin{aligned}
V(13) = V(19) = V(21) &= \frac{1}{4}dxu_{i-1}\cos\phi_j^T\ dyt_j\ dzw_{k-1}\\
V(14) = V(17) = V(22) &= \frac{1}{4}dxu_i\cos\phi_j^T\ dyt_j\ dzw_{k-1}\\
V(15) = V(20) = V(23) &= \frac{1}{4}dxu_{i-1}\cos\phi_j^T\ dyt_j\ dzw_k\\
V(16) = V(18) = V(24) &= \frac{1}{4}dxu_i\cos\phi_j^T\ dyt_j\ dzw_k.
\end{aligned} \tag{B.40}$$

For the y-z plane, without approximation, the 12 subcells have the volumes

$$\begin{aligned}
V(25) = V(31) = V(33) &= \frac{1}{4}dxt_i\ \cos\phi_{j-1}^U dxy_{j-1}\ dzw_{k-1}\\
V(26) = V(29) = V(34) &= \frac{1}{4}dxt_i\ \cos\phi_j^U dyu_j\ dzw_{k-1}\\
V(27) = V(32) = V(35) &= \frac{1}{4}dxt_i\ \cos\phi_{j-1}^U dyu_{j-1} dzw_k\\
V(28) = V(30) = V(36) &= \frac{1}{4}dxt_i\ \cos\phi_j^U dyu_j\ dzw_k.
\end{aligned} \tag{B.41}$$

B.2.4 Tracer gradients within the subcells

Within the 36 cells, the gradient of the tracer and density are required. The form for the grid difference operators are given here. There are further details regarding the reference points to be used in determining the density gradient. The reference point issues are discussed in Section B.2.6. The difference operator approximating a derivative will be written

$$\delta_x T_{i,k,j} = \frac{T_{i+1,k,j} - T_{i,k,j}}{dxu_i \cos\phi_j^T} \tag{B.42}$$

$$\delta_y T_{i,k,j} = \frac{T_{i,k,j+1} - T_{i,k,j}}{dyu_j} \tag{B.43}$$

$$\delta_z T_{i,k,j} = \frac{T_{i,k,j} - T_{i,k+1,j}}{dzw_k}. \tag{B.44}$$

Notice how the $\cos\phi_j^T$ is absorbed into the δ_x operator.

x-y plane

For the x-y plane, suppressing the k-index, the tracer gradient in the 12 boxes is given by

$$\nabla T(1) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j} \tag{B.45}$$

$$\nabla T(2) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j} \tag{B.46}$$

$$\nabla T(3) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j} \tag{B.47}$$

$$\nabla T(4) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j} \tag{B.48}$$

$$\nabla T(5) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i+1,j} + \hat{k}\delta_z T_{i+1,j} \tag{B.49}$$

$$\nabla T(6) = \hat{\imath}\delta_x T_{i,j} + \hat{\jmath}\delta_y T_{i+1,j-1} + \hat{k}\delta_z T_{i+1,j} \tag{B.50}$$

$$\nabla T(7) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i-1,j} + \hat{k}\delta_z T_{i-1,j} \tag{B.51}$$

$$\nabla T(8) = \hat{\imath}\delta_x T_{i-1,j} + \hat{\jmath}\delta_y T_{i-1,j-1} + \hat{k}\delta_z T_{i-1,j} \tag{B.52}$$

$$\nabla T(9) = \hat{\imath}\delta_x T_{i-1,j+1} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j+1} \tag{B.53}$$

$$\nabla T(10) = \hat{\imath}\delta_x T_{i,j+1} + \hat{\jmath}\delta_y T_{i,j} + \hat{k}\delta_z T_{i,j+1} \tag{B.54}$$

$$\nabla T(11) = \hat{\imath}\delta_x T_{i-1,j-1} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j-1} \tag{B.55}$$

$$\nabla T(12) = \hat{\imath}\delta_x T_{i,j-1} + \hat{\jmath}\delta_y T_{i,j-1} + \hat{k}\delta_z T_{i,j-1}. \tag{B.56}$$

There is one thing to note regarding the z-derivative in these expressions. It is no longer centered on the i,j plane and so presents a problem. The only place this issue raises its head is in the $|\nabla\rho|^{-2}$ term for the full tensor, in which all three derivatives are required. A centering of the squared z-derivative will be prescribed, in which no computational modes are introduced and proper placement is given. This point will be addressed in the discussion of equation (B.208); for now, it will be ignored.

The functional derivative of these x-y plane gradients is given by

$$\frac{\partial\nabla T(1)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dzw_k} \tag{B.57}$$

$$\frac{\partial\nabla T(2)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dzw_k} \tag{B.58}$$

$$\frac{\partial\nabla T(3)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T dxu_{i-1}} + \frac{\hat{\jmath}}{dyu_{j-1}} + \frac{\hat{k}}{dzw_k} \tag{B.59}$$

$$\frac{\partial \nabla T(4)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} + \frac{\hat{\jmath}}{dyu_{j-1}} + \frac{\hat{k}}{dzw_k} \tag{B.60}$$
$$\frac{\partial \nabla T(5)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} \tag{B.61}$$
$$\frac{\partial \nabla T(6)}{\partial T_{i,j}} = -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} \tag{B.62}$$
$$\frac{\partial \nabla T(7)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T\, dxu_{i-1}} \tag{B.63}$$
$$\frac{\partial \nabla T(8)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T\, dxu_{i-1}} \tag{B.64}$$
$$\frac{\partial \nabla T(9)}{\partial T_{i,j}} = -\frac{\hat{\jmath}}{dyu_y} \tag{B.65}$$
$$\frac{\partial \nabla T(10)}{\partial T_{i,j}} = -\frac{\hat{\jmath}}{dyu_j} \tag{B.66}$$
$$\frac{\partial \nabla T(11)}{\partial T_{i,j}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{B.67}$$
$$\frac{\partial \nabla T(12)}{\partial T_{i,j}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{B.68}$$

z-x plane

For the z-x plane, suppressing the j-index, the tracer gradient in the 12 boxes is given by

$$\nabla T(13) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k-1} \tag{B.69}$$
$$\nabla T(14) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k-1} \tag{B.70}$$
$$\nabla T(15) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k} \tag{B.71}$$
$$\nabla T(16) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i,k} + \hat{k}\delta_z T_{i,k} \tag{B.72}$$
$$\nabla T(17) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i+1,k} + \hat{k}\delta_z T_{i+1,k-1} \tag{B.73}$$
$$\nabla T(18) = \hat{\imath}\delta_x T_{i,k} + \hat{\jmath}\delta_y T_{i+1,k} + \hat{k}\delta_z T_{i+1,k} \tag{B.74}$$
$$\nabla T(19) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i-1,k} + \hat{k}\delta_z T_{i-1,k-1} \tag{B.75}$$
$$\nabla T(20) = \hat{\imath}\delta_x T_{i-1,k} + \hat{\jmath}\delta_y T_{i-1,k} + \hat{k}\delta_z T_{i-1,k} \tag{B.76}$$
$$\nabla T(21) = \hat{\imath}\delta_x T_{i-1,k-1} + \hat{\jmath}\delta_y T_{i,k-1} + \hat{k}\delta_z T_{i,k-1} \tag{B.77}$$
$$\nabla T(22) = \hat{\imath}\delta_x T_{i,k-1} + \hat{\jmath}\delta_y T_{i,k-1} + \hat{k}\delta_z T_{i,k-1} \tag{B.78}$$
$$\nabla T(23) = \hat{\imath}\delta_x T_{i-1,k+1} + \hat{\jmath}\delta_y T_{i,k+1} + \hat{k}\delta_z T_{i,k} \tag{B.79}$$
$$\nabla T(24) = \hat{\imath}\delta_x T_{i,k+1} + \hat{\jmath}\delta_y T_{i,k+1} + \hat{k}\delta_z T_{i,k}. \tag{B.80}$$

The functional derivative of these z-x plane gradients is given by

$$\frac{\partial \nabla T(13)}{\partial T_{i,j}} = \frac{\hat{\imath}}{\cos\phi_j^T\, dxu_{i-1}} - \frac{\hat{\jmath}}{dyu_j} - \frac{\hat{k}}{dzw_{k-1}} \tag{B.81}$$
$$\frac{\partial \nabla T(14)}{\partial T_{i,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} - \frac{\hat{\jmath}}{dyu_j} - \frac{\hat{k}}{dzw_{k-1}} \tag{B.82}$$
$$\frac{\partial \nabla T(15)}{\partial T_{i,k}} = \frac{\hat{\imath}}{\cos\phi_j^T\, dxu_{i-1}} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dzw_k} \tag{B.83}$$

$$
\begin{aligned}
\frac{\partial \nabla T(16)}{\partial T_{i,k}} &= -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dzw_k} && \text{(B.84)}\\
\frac{\partial \nabla T(17)}{\partial T_{i,k}} &= -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} && \text{(B.85)}\\
\frac{\partial \nabla T(18)}{\partial T_{i,k}} &= -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} && \text{(B.86)}\\
\frac{\partial \nabla T(19)}{\partial T_{i,k}} &= \frac{\hat{\imath}}{\cos\phi_j^T\, dxu_{i-1}} && \text{(B.87)}\\
\frac{\partial \nabla T(20)}{\partial T_{i,k}} &= \frac{\hat{\imath}}{\cos\phi_j^T\, dxu_{i-1}} && \text{(B.88)}\\
\frac{\partial \nabla T(21)}{\partial T_{i,k}} &= -\frac{\hat{k}}{dzw_{k-1}} && \text{(B.89)}\\
\frac{\partial \nabla T(22)}{\partial T_{i,k}} &= -\frac{\hat{k}}{dzw_{k-1}} && \text{(B.90)}\\
\frac{\partial \nabla T(23)}{\partial T_{i,k}} &= \frac{\hat{k}}{dzw_k} && \text{(B.91)}\\
\frac{\partial \nabla T(24)}{\partial T_{i,k}} &= \frac{\hat{k}}{dzw_k} && \text{(B.92)}
\end{aligned}
$$

y-z plane

For the y-z plane, suppressing the i-index, the tracer gradient in the 12 boxes is given by

$$
\begin{aligned}
\nabla T(25) &= \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{y,k-1} && \text{(B.93)}\\
\nabla T(26) &= \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j,k-1} && \text{(B.94)}\\
\nabla T(27) &= \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j,k} && \text{(B.95)}\\
\nabla T(28) &= \hat{\imath}\delta_x T_{j,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j,k} && \text{(B.96)}\\
\nabla T(29) &= \hat{\imath}\delta_x T_{j+1,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j+1,k-1} && \text{(B.97)}\\
\nabla T(30) &= \hat{\imath}\delta_x T_{j+1,k} + \hat{\jmath}\delta_y T_{j,k} + \hat{k}\delta_z T_{j+1,k} && \text{(B.98)}\\
\nabla T(31) &= \hat{\imath}\delta_x T_{j-1,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j-1,k-1} && \text{(B.99)}\\
\nabla T(32) &= \hat{\imath}\delta_x T_{j-1,k} + \hat{\jmath}\delta_y T_{j-1,k} + \hat{k}\delta_z T_{j-1,k} && \text{(B.100)}\\
\nabla T(33) &= \hat{\imath}\delta_x T_{j,k-1} + \hat{\jmath}\delta_y T_{j-1,k-1} + \hat{k}\delta_z T_{j,k-1} && \text{(B.101)}\\
\nabla T(34) &= \hat{\imath}\delta_x T_{j,k-1} + \hat{\jmath}\delta_y T_{j,k-1} + \hat{k}\delta_z T_{j,k-1} && \text{(B.102)}\\
\nabla T(35) &= \hat{\imath}\delta_x T_{j,k+1} + \hat{\jmath}\delta_y T_{j-1,k+1} + \hat{k}\delta_z T_{j,k} && \text{(B.103)}\\
\nabla T(36) &= \hat{\imath}\delta_x T_{j,k+1} + \hat{\jmath}\delta_y T_{j,k+1} + \hat{k}\delta_z T_{j,k}. && \text{(B.104)}
\end{aligned}
$$

The functional derivative of these y-z plane gradients is given by

$$
\begin{aligned}
\frac{\partial \nabla T(25)}{\partial T_{j,k}} &= -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} + \frac{\hat{\jmath}}{dyu_{j-1}} - \frac{\hat{k}}{dzw_{k-1}} && \text{(B.105)}\\
\frac{\partial \nabla T(26)}{\partial T_{j,k}} &= -\frac{\hat{\imath}}{\cos\phi_j^T\, dxu_i} - \frac{\hat{\jmath}}{dyu_j} - \frac{\hat{k}}{dzw_{k-1}} && \text{(B.106)}
\end{aligned}
$$

$$\frac{\partial \nabla T(27)}{\partial T_{j,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} + \frac{\hat{\jmath}}{dyu_{j-1}} + \frac{\hat{k}}{dzw_k} \tag{B.107}$$

$$\frac{\partial \nabla T(28)}{\partial T_{j,k}} = -\frac{\hat{\imath}}{\cos\phi_j^T dxu_i} - \frac{\hat{\jmath}}{dyu_j} + \frac{\hat{k}}{dzw_k} \tag{B.108}$$

$$\frac{\partial \nabla T(29)}{\partial T_{j,k}} = -\frac{\hat{\jmath}}{dyu_j} \tag{B.109}$$

$$\frac{\partial \nabla T(30)}{\partial T_{j,k}} = -\frac{\hat{\jmath}}{dyu_j} \tag{B.110}$$

$$\frac{\partial \nabla T(31)}{\partial T_{j,k}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{B.111}$$

$$\frac{\partial \nabla T(32)}{\partial T_{j,k}} = \frac{\hat{\jmath}}{dyu_{j-1}} \tag{B.112}$$

$$\frac{\partial \nabla T(33)}{\partial T_{j,k}} = -\frac{\hat{k}}{dzw_{k-1}} \tag{B.113}$$

$$\frac{\partial \nabla T(34)}{\partial T_{j,k}} = -\frac{\hat{k}}{dzw_{k-1}} \tag{B.114}$$

$$\frac{\partial \nabla T(35)}{\partial T_{j,k}} = \frac{\hat{k}}{dzw_k} \tag{B.115}$$

$$\frac{\partial \nabla T(36)}{\partial T_{j,k}} = \frac{\hat{k}}{dzw_k} \tag{B.116}$$

B.2.5 Discretized functional

For each of the three planes, there are 12 components to the functional which contain contributions from the grid tracer value $T_{i,k,j}$. These 12 components correspond to the 12 subcells shown in Figures B.2 and B.3. For example, the x-y plane functional can be written

$$\begin{aligned} S^{(x-y)} &= -\frac{1}{2}\sum_{i,k}\sum_{n=1}^{12} A(n)V(n)\frac{|\nabla T(n)\wedge\nabla\rho(n)|^2}{|\nabla\rho(n)|^2} \\ &\equiv \sum_{i,k} L_{i,k}, \end{aligned} \tag{B.117}$$

where the discretized quadratic form is

$$\begin{aligned} L_{i,k} &= -\frac{1}{2}\sum_{n=1}^{12} A(n)V(n)\frac{|\nabla T(n)\wedge\nabla\rho(n)|^2}{|\nabla\rho(n)|^2} \\ &\equiv \sum_{n=1}^{12} L_{i,k}^{(n)}, \end{aligned} \tag{B.118}$$

and $A(n)$ is the non-negative diffusion coefficient for the subcell.

B.2.6 Reference points for computing the density gradients

The proper calculation of the density gradients is an essential element in the isopycnal diffusion discretization. There are two physical constraints that motivate a particular choice for the form of the discretization. The first involves the generally nontrivial dependence of the density on the pressure. Since neutral directions need to be approximated for defining the orientation of the diffusive fluxes (McDougall, 1987), the densities appearing in a particular discretized density gradient must be referenced to the *same* pressure level. In the z-coordinate model, it is simple to employ the same number of reference levels as there are depth levels. This allows for the most accurate approximation to the neutral directions as possible with the particular vertical resolution. Such a procedure was employed in the original Cox (1987) implementation of isopycnal diffusion. Additionally, the form for the discretization should reflect the need to have the adiabatic portion of the diffusion operator vanish for the case of a single active tracer. This constraint motivates the discretization of the density gradient explicitly in terms of the thermal and saline partial derivatives of the density, in which these derivatives are evaluated at the *same* pressure, temperature, and salinity for the density gradients appearing within a particular subcell component of the functional. If this prescription is followed, each term of the functional will identically vanish when density is a function of a single active tracer, which means that the adiabatic diffusion operator will vanish in this case as well.

The two physical constraints regarding reference points can be implemented numerically in a number of ways. For example, in an earlier derivation of the discretized diffusion operator, the issues of reference points were ignored until the very end of the derivation. At that point, reasonable choices were made for choosing the reference points, which consisted of referencing on the various sides of the T-cells consistent with the location of the diffusive fluxes. However, it was soon realized that the numerical constraint of defining a dissipative diffusion operator was not satisfied by this choice. Rather, in order to ensure a dissipative operator, *all* choices of discretization should be made in the framework of the discretized functional. As noted earlier, the power of this perspective is that whatever the discretization used for the functional, the resulting diffusion operator derived from taking the functional derivative will be dissipative.

To put some details behind these comments, consider the component of the x-y plane functional arising from the first subcell (see Figure B.2). Ignoring the issues of reference points, this term has the form

$$L^{(1)}_{i,k,j} = -\frac{A(1)V(1)}{2|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,k,j}\delta_y\rho_{i,k,j} - \delta_y T_{i,k,j}\delta_x\rho_{i-1,k,j}\right)^2. \tag{B.119}$$

The numerator has contributions from density at the three grid points: $\rho_{i,k,j+1}, \rho_{i,k,j}$, and $\rho_{i-1,k,j}$. These three points form a triangle, or triad, in the x-y plane (see Figure B.2). A self-consistent and simple choice of reference point is the corner of this triad, which for this case is the T-cell point i,k,j. With this choice, each of the three densities appearing in a triad will employ the *same* thermal and saline partial derivatives which are referenced to this single point. Therefore, the density gradients for this particular contribution to the functional will be discretized as

$$\begin{aligned}
\delta_x\rho_{i-1,k,j} &= \alpha_{i,k,j}\ \delta_x\theta_{i-1,k,j} + \beta_{i,k,j}\ \delta_x S_{i-1,k,j} \equiv \delta_x\rho^{(i,k,j)}_{i-1,k,j} && \text{(B.120)}\\
\delta_y\rho_{i,k,j} &= \alpha_{i,k,j}\ \delta_y\theta_{i,k,j} + \beta_{i,k,j}\ \delta_y S_{i,k,j} \equiv \delta_y\rho^{(i,k,j)}_{i,k,j}. && \text{(B.121)}
\end{aligned}$$

In this expression, the thermal and saline coefficients are written

$$\begin{aligned}
\alpha_{i,k,j} &= \rho_\theta\left(\theta_{i,k,j}, S_{i,k,j}, k\right) && \text{(B.122)}\\
\beta_{i,k,j} &= \rho_S\left(\theta_{i,k,j}, S_{i,k,j}, k\right), && \text{(B.123)}
\end{aligned}$$

where the arguments of the density partial derivatives indicate the value of the potential temperature, salinity, and pressure reference level k for use in their computation. MOM2 generally employs a cubic approximation to the UNESCO equation of state. Once this approximation is determined, the quadratic expressions for the partial derivatives ρ_θ and ρ_S are found and tabulated along with the cubic equation of state. The superscripts introduced on the density gradients allow for a compact notation which exposes the information about the referece point. Parentheses are included in this notation to help distinguish it from the subscripts referring to explicit grid points. Finally, the discretization of the denominator is prescribed to be consistent with that determined by the numerator. Note that only the x and y gradients are explicitly prescribed since the z gradient does not appear in the numerator. This ambiguity will be settled at a later stage of the derivation. For the present purposes, let

$$(\nabla\rho(1))^2 = (\delta_x\rho^{(i,k,j)}_{i-1,k,j})^2 + (\delta_y\rho^{(i,k,j)}_{i,k,j})^2 + (\delta_z\rho^{(i,k,j)}_{i,k,j})^2. \tag{B.124}$$

Functional in the x-y plane

The x-y plane functional is relevant only for the full tensor as it vanishes in the small angle limit. The quadratic form which contains contributions from the T-point $T_{i,k,j}$ in the x-y plane is given by the sum of the following 12 terms (k index suppressed)

$$L^{(1)}_{i,j} = -\frac{A(1)V(1)}{2|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i-1,j}\right)^2 \tag{B.125}$$

$$L^{(2)}_{i,j} = -\frac{A(2)V(2)}{2|\nabla\rho(2)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j)}_{i,j}\right)^2 \tag{B.126}$$

$$L^{(3)}_{i,j} = -\frac{A(3)V(3)}{2|\nabla\rho(3)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i-1,j}\right)^2 \tag{B.127}$$

$$L^{(4)}_{i,j} = -\frac{A(4)V(4)}{2|\nabla\rho(4)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i,j)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i,j}\right)^2 \tag{B.128}$$

$$L^{(5)}_{i,j} = -\frac{A(5)V(5)}{2|\nabla\rho(5)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i+1,j)}_{i+1,j} - \delta_y T_{i+1,j}\delta_x\rho^{(i+1,j)}_{i,j}\right)^2 \tag{B.129}$$

$$L^{(6)}_{i,j} = -\frac{A(6)V(6)}{2|\nabla\rho(6)|^2}\left(\delta_x T_{i,j}\delta_y\rho^{(i+1,j)}_{i+1,j-1} - \delta_y T_{i+1,j-1}\delta_x\rho^{(i+1,j)}_{i,j}\right)^2 \tag{B.130}$$

$$L^{(7)}_{i,j} = -\frac{A(7)V(7)}{2|\nabla\rho(7)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i-1,j)}_{i-1,j} - \delta_y T_{i-1,j}\delta_x\rho^{(i-1,j)}_{i-1,j}\right)^2 \tag{B.131}$$

$$L^{(8)}_{i,j} = -\frac{A(8)V(8)}{2|\nabla\rho(8)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho^{(i-1,j)}_{i-1,j-1} - \delta_y T_{i-1,j-1}\delta_x\rho^{(i-1,j)}_{i-1,j}\right)^2 \tag{B.132}$$

$$L^{(9)}_{i,j} = -\frac{A(9)V(9)}{2|\nabla\rho(9)|^2}\left(\delta_x T_{i-1,j+1}\delta_y\rho^{(i,j+1)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j+1)}_{i-1,j+1}\right)^2 \tag{B.133}$$

$$L^{(10)}_{i,j} = -\frac{A(10)V(10)}{2|\nabla\rho(10)|^2}\left(\delta_x T_{i,j+1}\delta_y\rho^{(i,j+1)}_{i,j} - \delta_y T_{i,j}\delta_x\rho^{(i,j+1)}_{i,j+1}\right)^2 \tag{B.134}$$

$$L^{(11)}_{i,j} = -\frac{A(11)V(11)}{2|\nabla\rho(11)|^2}\left(\delta_x T_{i-1,j-1}\delta_y\rho^{(i,j-1)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j-1)}_{i-1,j-1}\right)^2 \tag{B.135}$$

$$L^{(12)}_{i,j} = -\frac{A(12)V(12)}{2|\nabla\rho(12)|^2}\left(\delta_x T_{i,j-1}\delta_y\rho^{(i,j-1)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j-1)}_{i,j-1}\right)^2 \tag{B.136}$$

Functional in the z-x plane

The z-x plane quadratic form contains contributions from the T-point $T_{i,k,j}$ in the following 12 terms (j index suppressed)

$$L^{(13)}_{i,k} = -\frac{A(13)V(13)}{2|\nabla\rho(13)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho^{(i,k)}_{i-1,k} - \delta_x T_{i-1,k}\delta_z\rho^{(i,k)}_{i,k-1}\right)^2 \tag{B.137}$$

$$L_{i,k}^{(14)} = -\frac{A(14)V(14)}{2|\nabla\rho(14)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k}^{(i,k)} - \delta_x T_{i,k}\delta_z\rho_{i,k-1}^{(i,k)}\right)^2 \quad \text{(B.138)}$$

$$L_{i,k}^{(15)} = -\frac{A(15)V(15)}{2|\nabla\rho(15)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k}^{(i,k)}\right)^2 \quad \text{(B.139)}$$

$$L_{i,k}^{(16)} = -\frac{A(16)V(16)}{2|\nabla\rho(16)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k}^{(i,k)} - \delta_x T_{i,k}\delta_z\rho_{i,k}^{(i,k)}\right)^2 \quad \text{(B.140)}$$

$$L_{i,k}^{(17)} = -\frac{A(17)V(17)}{2|\nabla\rho(17)|^2}\left(\delta_z T_{i+1,k-1}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k-1}^{(i+1,k)}\right)^2 \quad \text{(B.141)}$$

$$L_{i,k}^{(18)} = -\frac{A(18)V(18)}{2|\nabla\rho(18)|^2}\left(\delta_z T_{i+1,k}\delta_x\rho_{i,k}^{(i+1,k)} - \delta_x T_{i,k}\delta_z\rho_{i+1,k}^{(i+1,k)}\right)^2 \quad \text{(B.142)}$$

$$L_{i,k}^{(19)} = -\frac{A(19)V(19)}{2|\nabla\rho(19)|^2}\left(\delta_z T_{i-1,k-1}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k-1}^{(i-1,k)}\right)^2 \quad \text{(B.143)}$$

$$L_{i,k}^{(20)} = -\frac{A(20)V(20)}{2|\nabla\rho(20)|^2}\left(\delta_z T_{i-1,k}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k}^{(i-1,k)}\right)^2 \quad \text{(B.144)}$$

$$L_{i,k}^{(21)} = -\frac{A(21)V(21)}{2|\nabla\rho(21)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right)^2 \quad \text{(B.145)}$$

$$L_{i,k}^{(22)} = -\frac{A(22)V(22)}{2|\nabla\rho(22)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k-1}^{(i,k-1)} - \delta_x T_{i,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right)^2 \quad \text{(B.146)}$$

$$L_{i,k}^{(23)} = -\frac{A(23)V(23)}{2|\nabla\rho(23)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k+1}^{(i,k+1)} - \delta_x T_{i-1,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right)^2 \quad \text{(B.147)}$$

$$L_{i,k}^{(24)} = -\frac{A(24)V(24)}{2|\nabla\rho(24)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k+1}^{(i,k+1)} - \delta_x T_{i,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right)^2 \quad \text{(B.148)}$$

Functional in the y-z plane

The y-z plane quadratic form contains contributions from the T-point $T_{i,k,j}$ in the following 12 terms (i index suppressed)

$$L_{j,k}^{(25)} = -\frac{A(25)V(25)}{2|\nabla\rho(25)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k-1}^{(j,k)}\right)^2 \quad \text{(B.149)}$$

$$L_{j,k}^{(26)} = -\frac{A(26)V(26)}{2|\nabla\rho(26)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k}^{(j,k)} - \delta_y T_{j,k}\delta_z\rho_{j,k-1}^{(j,k)}\right)^2 \quad \text{(B.150)}$$

$$L_{j,k}^{(27)} = -\frac{A(27)V(27)}{2|\nabla\rho(27)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k}^{(j,k)}\right)^2 \quad \text{(B.151)}$$

$$L_{j,k}^{(28)} = -\frac{A(28)V(28)}{2|\nabla\rho(28)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j,k}^{(j,k)} - \delta_y T_{j,k}\delta_z\rho_{j,k}^{(j,k)}\right)^2 \quad \text{(B.152)}$$

$$L_{j,k}^{(29)} = -\frac{A(29)V(29)}{2|\nabla\rho(29)|^2}\left(\delta_z T_{j+1,k-1}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k-1}^{(j+1,k)}\right)^2 \quad \text{(B.153)}$$

$$L_{j,k}^{(30)} = -\frac{A(30)V(30)}{2|\nabla\rho(30)|^2}\left(\delta_z T_{j+1,k}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k}^{(j+1,k)}\right)^2 \quad \text{(B.154)}$$

$$L_{j,k}^{(31)} = -\frac{A(31)V(31)}{2|\nabla\rho(31)|^2}\left(\delta_z T_{j-1,k-1}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k-1}^{(j-1,k)}\right)^2 \quad \text{(B.155)}$$

$$L_{j,k}^{(32)} = -\frac{A(32)V(32)}{2|\nabla\rho(32)|^2}\left(\delta_z T_{j-1,k}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k}^{(j-1,k)}\right)^2 \quad \text{(B.156)}$$

$$L_{j,k}^{(33)} = -\frac{A(33)V(33)}{2|\nabla\rho(33)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k-1}^{(j,k-1)} - \delta_y T_{j-1,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right)^2 \quad \text{(B.157)}$$

$$L_{j,k}^{(34)} = -\frac{A(34)V(34)}{2|\nabla\rho(34)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k-1}^{(j,k-1)} - \delta_y T_{j,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right)^2 \quad \text{(B.158)}$$

$$L_{j,k}^{(35)} = -\frac{A(35)V(35)}{2|\nabla\rho(35)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j-1,k+1}^{(j,k+1)} - \delta_y T_{j-1,k+1}\delta_z\rho_{j,k}^{(j,k+1)}\right)^2 \quad \text{(B.159)}$$

$$L_{j,k}^{(36)} = -\frac{A(36)V(36)}{2|\nabla\rho(36)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j,k+1}^{(j,k+1)} - \delta_y T_{j,k+1}\delta_z\rho_{j,k}^{(j,k+1)}\right)^2 \quad \text{(B.160)}$$

B.2.7 Slope constraint

In these expressions, the diffusion coefficient for a particular quarter cell is given by $A(n)$. This coefficient will be chosen to satisfy the numerical stability constraint on the diffusion equation. For the small angle tensor, the prescription of Gerdes et al (1991) is the MOM2 default, in which for large slopes, the diffusion coefficient is quadratically rescaled to a smaller value. Their prescription applied to the present formulation says for each quarter cell n, if the slope $|S(n)| = |\delta_{x_i}\rho/\delta_z\rho| > \delta$, then the corresponding diffusion coefficient $A(n)$ is rescaled as $A(n) \rightarrow A(n)(\delta/S(n))^2$. This prescription provides a unique definition of the diffusion coefficient and thus provides for a unique definition of the diffusive fluxes. An alternative used by Danabasoglu and McWilliams (1995) suggests a hyperobolic tangent rescaling. This second rescaling is implemented in MOM2 as an option. For the full tensor, the rescaling is $A(n) \rightarrow \delta\ A(n)(|S(n)| + |S(n)|^{-1})$ for $S_{(-1)} < |S(n)| < S_{(-1)}^{-1}$. For the special case of a grid in which $\delta > 1/2$, the full tensor requires no rescaling of the diffusion coefficients in order to maintain numerical stability. Slope constraints are discussed more completely in Griffies et al, (1997). Note that the original slope clipping scheme of Xoc (1987) is not supported anymore due to its potential to introduce unphysically large diabatic fluxes and to artificially cap of convection (see Griffies et al, 1997 for further discussion).

B.2.8 Derivative of the functional

x-y plane

Taking the derivative of the 12 contributions to the functional in the x-y plane yields

$$\begin{aligned}\frac{\partial L_{i,j}^{(1)}}{\partial T_{i,k,j}} &= -\frac{A(1)V(1)}{|\nabla\rho(1)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho_{i,j}^{(i,j)} - \delta_y T_{i,j}\delta_x\rho_{i-1,j}^{(i,j)}\right)\\ &\times\left(\frac{\delta_y\rho_{i,j}^{(i,j)}}{\cos\phi_j^T dxu_{i-1}} + \frac{\delta_x\rho_{i-1,j}^{(i,j)}}{dyu_j}\right)\end{aligned} \quad \text{(B.161)}$$

$$\begin{aligned}\frac{\partial L_{i,j}^{(2)}}{\partial T_{i,k,j}} &= -\frac{A(2)V(2)}{|\nabla\rho(2)|^2}\left(\delta_x T_{i,j}\delta_y\rho_{i,j}^{(i,j)} - \delta_y T_{i,j}\delta_x\rho_{i,j}^{(i,j)}\right)\\ &\times\left(-\frac{\delta_y\rho_{i,j}^{(i,j)}}{\cos\phi_j^T dxu_i} + \frac{\delta_x\rho_{i,j}^{(i,j)}}{dyu_j}\right)\end{aligned} \quad \text{(B.162)}$$

$$\begin{aligned}\frac{\partial L_{i,j}^{(3)}}{\partial T_{i,k,j}} &= -\frac{A(3)V(3)}{|\nabla\rho(3)|^2}\left(\delta_x T_{i-1,j}\delta_y\rho_{i,j-1}^{(i,j)} - \delta_y T_{i,j-1}\delta_x\rho_{i-1,j}^{(i,j)}\right)\\ &\times\left(\frac{\delta_y\rho_{i,j-1}^{(i,j)}}{\cos\phi_j^T dxu_{i-1}} - \frac{\delta_x\rho_{i-1,j}^{(i,j)}}{dyu_{j-1}}\right)\end{aligned} \quad \text{(B.163)}$$

$$\frac{\partial L_{i,j}^{(4)}}{\partial T_{i,k,j}} = -\frac{A(4)V(4)}{|\nabla\rho(4)|^2}\left(\delta_x T_{i,j}\delta_y\rho_{i,j-1}^{(i,j)} - \delta_y T_{i,j-1}\delta_x\rho_{i,j}^{(i,j)}\right)$$

$$\times \quad \left(-\frac{\delta_y \rho^{(i,j)}_{i,j-1}}{\cos\phi^T_j dxu_i} - \frac{\delta_x \rho^{(i,j)}_{i,j}}{dyu_{j-1}}\right) \tag{B.164}$$

$$\frac{\partial L^{(5)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(5)V(5)}{|\nabla\rho(5)|^2}\left(\delta_x T_{i,j}\delta_y \rho^{(i+1,j)}_{i+1,j} - \delta_y T_{i+1,j}\delta_x \rho^{(i+1,j)}_{i,j}\right)$$

$$\times \quad \left(-\frac{\delta_y \rho^{(i+1,j)}_{i+1,j}}{\cos\phi^T_j dxu_i}\right) \tag{B.165}$$

$$\frac{\partial L^{(6)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(6)V(6)}{|\nabla\rho(6)|^2}\left(\delta_x T_{i,j}\delta_y \rho^{(i+1,j)}_{i+1,j-1} - \delta_y T_{i+1,j-1}\delta_x \rho^{(i+1,j)}_{i,j}\right)$$

$$\times \quad \left(-\frac{\delta_y \rho^{(i+1,j)}_{i+1,j-1}}{\cos\phi^T_j dxu_i}\right) \tag{B.166}$$

$$\frac{\partial L^{(7)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(7)V(7)}{|\nabla\rho(7)|^2}\left(\delta_x T_{i-1,j}\delta_y \rho^{(i-1,j)}_{i-1,j} - \delta_y T_{i-1,j}\delta_x \rho^{(i-1,j)}_{i-1,j}\right)$$

$$\times \quad \left(\frac{\delta_y \rho^{(i-1,j)}_{i-1,j}}{\cos\phi^T_j dxu_{i-1}}\right) \tag{B.167}$$

$$\frac{\partial L^{(8)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(8)V(8)}{|\nabla\rho(8)|^2}\left(\delta_x T_{i-1,j}\delta_y \rho^{(i-1,j)}_{i-1,j-1} - \delta_y T_{i-1,j-1}\delta_x \rho^{(i-1,j)}_{i-1,j}\right)$$

$$\times \quad \left(\frac{\delta_y \rho^{(i-1,j)}_{i-1,j-1}}{\cos\phi^T_j dxu_{i-1}}\right) \tag{B.168}$$

$$\frac{\partial L^{(9)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(9)V(9)}{|\nabla\rho(9)|^2}\left(\delta_x T_{i-1,j+1}\delta_y \rho^{(i,j+1)}_{i,j} - \delta_y T_{i,j}\delta_x \rho^{(i,j+1)}_{i-1,j+1}\right)$$

$$\times \quad \left(\frac{\delta_x \rho^{(i,j+1)}_{i-1,j+1}}{dyu_j}\right) \tag{B.169}$$

$$\frac{\partial L^{(10)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(10)V(10)}{|\nabla\rho(10)|^2}\left(\delta_x T_{i,j+1}\delta_y \rho^{(i,j+1)}_{i,j} - \delta_y T_{i,j}\delta_x \rho^{(i,j+1)}_{i,j+1}\right)$$

$$\times \quad \left(\frac{\delta_x \rho^{(i,j+1)}_{i,j+1}}{dyu_j}\right) \tag{B.170}$$

$$\frac{\partial L^{(11)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(11)V(11)}{|\nabla\rho(11)|^2}\left(\delta_x T_{i-1,j-1}\delta_y \rho^{(i,j-1)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x \rho^{(i,j-1)}_{i-1,j-1}\right)$$

$$\times \quad \left(-\frac{\delta_x \rho^{(i,j-1)}_{i-1,j-1}}{dyu_{j-1}}\right) \tag{B.171}$$

$$\frac{\partial L^{(12)}_{i,j}}{\partial T_{i,k,j}} = -\frac{A(12)V(12)}{|\nabla\rho(12)|^2}\left(\delta_x T_{i,j-1}\delta_y \rho^{(i,j-1)}_{i,j-1} - \delta_y T_{i,j-1}\delta_x \rho^{(i,j-1)}_{i,j-1}\right)$$

$$\times \quad \left(-\frac{\delta_x \rho^{(i,j-1)}_{i,j-1}}{dyu_{j-1}}\right) \tag{B.172}$$

z-x plane

Taking the derivative of the 12 contributions to the functional in the z-x plane yields

$$\begin{aligned}
\frac{\partial L^{(13)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(13)V(13)}{|\nabla\rho(13)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho^{(i,k)}_{i-1,k} - \delta_x T_{i-1,k}\delta_z\rho^{(i,k)}_{i,k-1}\right)\\
&\times \left(-\frac{\delta_x\rho^{(i,k)}_{i-1,k}}{dzw_{k-1}} - \frac{\delta_z\rho^{(i,k)}_{i,k-1}}{\cos\phi^T_j\, dxu_{i-1}}\right) \qquad \text{(B.173)}\\
\frac{\partial L^{(14)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(14)V(14)}{|\nabla\rho(14)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho^{(i,k)}_{i,k} - \delta_x T_{i,k}\delta_z\rho^{(i,k)}_{i,k-1}\right)\\
&\times \left(-\frac{\delta_x\rho^{(i,k)}_{i,k}}{dzw_{k-1}} + \frac{\delta_z\rho^{(i,k)}_{i,k-1}}{\cos\phi^T_j\, dxu_{i}}\right) \qquad \text{(B.174)}\\
\frac{\partial L^{(15)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(15)V(15)}{|\nabla\rho(15)|^2}\left(\delta_z T_{i,k}\delta_x\rho^{(i,k)}_{i-1,k} - \delta_x T_{i-1,k}\delta_z\rho^{(i,k)}_{i,k}\right)\\
&\times \left(\frac{\delta_x\rho^{(i,k)}_{i-1,k}}{dzw_{k}} - \frac{\delta_z\rho^{(i,k)}_{i,k}}{\cos\phi^T_j\, dxu_{i-1}}\right) \qquad \text{(B.175)}\\
\frac{\partial L^{(16)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(16)V(16)}{|\nabla\rho(16)|^2}\left(\delta_z T_{i,k}\delta_x\rho^{(i,k)}_{i,k} - \delta_x T_{i,k}\delta_z\rho^{(i,k)}_{i,k}\right)\\
&\times \left(\frac{\delta_x\rho^{(i,k)}_{i,k}}{dzw_{k}} + \frac{\delta_z\rho^{(i,k)}_{i,k}}{\cos\phi^T_j\, dxu_{i}}\right) \qquad \text{(B.176)}\\
\frac{\partial L^{(17)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(17)V(17)}{|\nabla\rho(17)|^2}\left(\delta_z T_{i+1,k-1}\delta_x\rho^{(i+1,k)}_{i,k} - \delta_x T_{i,k}\delta_z\rho^{(i+1,k)}_{i+1,k-1}\right)\\
&\times \left(\frac{\delta_z\rho^{(i+1,k)}_{i+1,k-1}}{\cos\phi^T_j\, dxu_{i}}\right) \qquad \text{(B.177)}\\
\frac{\partial L^{(18)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(18)V(18)}{|\nabla\rho(18)|^2}\left(\delta_z T_{i+1,k}\delta_x\rho^{(i+1,k)}_{i,k} - \delta_x T_{i,k}\delta_z\rho^{(i+1,k)}_{i+1,k}\right)\\
&\times \left(\frac{\delta_z\rho^{(i+1,k)}_{i+1,k}}{\cos\phi^T_j\, dxu_{i}}\right) \qquad \text{(B.178)}\\
\frac{\partial L^{(19)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(19)V(19)}{|\nabla\rho(19)|^2}\left(\delta_z T_{i-1,k-1}\delta_x\rho^{(i-1,k)}_{i-1,k} - \delta_x T_{i-1,k}\delta_z\rho^{(i-1,k)}_{i-1,k-1}\right)\\
&\times \left(-\frac{\delta_z\rho^{(i-1,k)}_{i-1,k-1}}{\cos\phi^T_j\, dxu_{i-1}}\right) \qquad \text{(B.179)}\\
\frac{\partial L^{(20)}_{i,k}}{\partial T_{i,k,j}} &= -\frac{A(20)V(20)}{|\nabla\rho(20)|^2}\left(\delta_z T_{i-1,k}\delta_x\rho^{(i-1,k)}_{i-1,k} - \delta_x T_{i-1,k}\delta_z\rho^{(i-1,k)}_{i-1,k}\right)\\
&\times \left(-\frac{\delta_z\rho^{(i-1,k)}_{i-1,k}}{\cos\phi^T_j\, dxu_{i-1}}\right) \qquad \text{(B.180)}
\end{aligned}$$

$$\frac{\partial L_{i,k}^{(21)}}{\partial T_{i,k,j}} = -\frac{A(21)V(21)}{|\nabla\rho(21)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right) \times \left(-\frac{\delta_x\rho_{i-1,k-1}^{(i,k-1)}}{dzw_{k-1}}\right) \quad \text{(B.181)}$$

$$\frac{\partial L_{i,k}^{(22)}}{\partial T_{i,k,j}} = -\frac{A(22)V(22)}{|\nabla\rho(22)|^2}\left(\delta_z T_{i,k-1}\delta_x\rho_{i,k-1}^{(i,k-1)} - \delta_x T_{i,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}\right) \times \left(-\frac{\delta_x\rho_{i,k-1}^{(i,k-1)}}{dzw_{k-1}}\right) \quad \text{(B.182)}$$

$$\frac{\partial L_{i,k}^{(23)}}{\partial T_{i,k,j}} = -\frac{A(23)V(23)}{|\nabla\rho(23)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i-1,k+1}^{(i,k+1)} - \delta_x T_{i-1,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right) \times \left(\frac{\delta_x\rho_{i-1,k+1}^{(i,k+1)}}{dzw_k}\right) \quad \text{(B.183)}$$

$$\frac{\partial L_{i,k}^{(24)}}{\partial T_{i,k,j}} = -\frac{A(24)V(24)}{|\nabla\rho(24)|^2}\left(\delta_z T_{i,k}\delta_x\rho_{i,k+1}^{(i,k+1)} - \delta_x T_{i,k+1}\delta_z\rho_{i,k}^{(i,k+1)}\right) \times \left(\frac{\delta_x\rho_{i,k+1}^{(i,k+1)}}{dzw_k}\right) \quad \text{(B.184)}$$

$$\text{(B.185)}$$

y-z plane

Taking the derivative of the 12 contributions to the functional in the y-z plane yields

$$\frac{\partial L_{j,k}^{(25)}}{\partial T_{i,k,j}} = -\frac{A(25)V(25)}{|\nabla\rho(25)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k-1}^{(j,k)}\right) \times \left(-\frac{\delta_y\rho_{j-1,k}^{(j,k)}}{dzw_{k-1}} - \frac{\delta_z\rho_{j,k-1}^{(j,k)}}{dyu_{j-1}}\right) \quad \text{(B.186)}$$

$$\frac{\partial L_{j,k}^{(26)}}{\partial T_{i,k,j}} = -\frac{A(26)V(26)}{|\nabla\rho(26)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k}^{(j,k)} - \delta_y T_{j,k}\delta_z\rho_{j,k-1}^{(j,k)}\right) \times \left(-\frac{\delta_y\rho_{j,k}^{(j,k)}}{dzw_{k-1}} - \frac{\delta_z\rho_{j,k-1}^{(j,k)}}{dyu_j}\right) \quad \text{(B.187)}$$

$$\frac{\partial L_{j,k}^{(27)}}{\partial T_{i,k,j}} = -\frac{A(27)V(27)}{|\nabla\rho(27)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j-1,k}^{(j,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j,k}^{(j,k)}\right) \times \left(-\frac{\delta_y\rho_{j-1,k}^{(j,k)}}{dzw_k} - \frac{\delta_z\rho_{j,k}^{(j,k)}}{dyu_{j-1}}\right) \quad \text{(B.188)}$$

$$\frac{\partial L_{j,k}^{(28)}}{\partial T_{i,k,j}} = -\frac{A(28)V(28)}{|\nabla\rho(28)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j,k}^{(j,k)} - \delta_y T_{j,k}\delta_z\rho_{j,k}^{(j,k)}\right) \times \left(\frac{\delta_y\rho_{j,k}^{(j,k)}}{dzw_k} + \frac{\delta_z\rho_{j,k}^{(j,k)}}{dyu_j}\right) \quad \text{(B.189)}$$

$$
\begin{aligned}
\frac{\partial L_{j,k}^{(29)}}{\partial T_{i,k,j}} &= -\frac{A(29)V(29)}{|\nabla\rho(29)|^2}\left(\delta_z T_{j+1,k-1}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k-1}^{(j+1,k)}\right) \\
&\times \left(\frac{\delta_z\rho_{j+1,k-1}^{(j+1,k)}}{dyu_j}\right) && \text{(B.190)} \\
\frac{\partial L_{j,k}^{(30)}}{\partial T_{i,k,j}} &= -\frac{A(30)V(30)}{|\nabla\rho(30)|^2}\left(\delta_z T_{j+1,k}\delta_y\rho_{j,k}^{(j+1,k)} - \delta_y T_{j,k}\delta_z\rho_{j+1,k}^{(j+1,k)}\right) \\
&\times \left(\frac{\delta_z\rho_{j+1,k}^{(j+1,k)}}{dyu_j}\right) && \text{(B.191)} \\
\frac{\partial L_{j,k}^{(31)}}{\partial T_{i,k,j}} &= -\frac{A(31)V(31)}{|\nabla\rho(31)|^2}\left(\delta_z T_{j-1,k-1}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k-1}^{(j-1,k)}\right) \\
&\times \left(-\frac{\delta_z\rho_{j-1,k-1}^{(j-1,k)}}{dyu_{j-1}}\right) && \text{(B.192)} \\
\frac{\partial L_{j,k}^{(32)}}{\partial T_{i,k,j}} &= -\frac{A(32)V(32)}{|\nabla\rho(32)|^2}\left(\delta_z T_{j-1,k}\delta_y\rho_{j-1,k}^{(j-1,k)} - \delta_y T_{j-1,k}\delta_z\rho_{j-1,k}^{(j-1,k)}\right) \\
&\times \left(-\frac{\delta_z\rho_{j-1,k}^{(j-1,k)}}{dyu_{j-1}}\right) && \text{(B.193)} \\
\frac{\partial L_{j,k}^{(33)}}{\partial T_{i,k,j}} &= -\frac{A(33)V(33)}{|\nabla\rho(33)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j-1,k-1}^{(j,k-1)} - \delta_y T_{j-1,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right) \\
&\times \left(-\frac{\delta_y\rho_{j-1,k-1}^{(j,k-1)}}{dzw_{k-1}}\right) && \text{(B.194)} \\
\frac{\partial L_{j,k}^{(34)}}{\partial T_{i,k,j}} &= -\frac{A(34)V(34)}{|\nabla\rho(34)|^2}\left(\delta_z T_{j,k-1}\delta_y\rho_{j,k-1}^{(j,k-1)} - \delta_y T_{j,k-1}\delta_z\rho_{j,k-1}^{(j,k-1)}\right) \\
&\times \left(-\frac{\delta_y\rho_{j,k-1}^{(j,k-1)}}{dzw_{k-1}}\right) && \text{(B.195)} \\
\frac{\partial L_{j,k}^{(35)}}{\partial T_{i,k,j}} &= -\frac{A(35)V(35)}{|\nabla\rho(35)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j-1,k+1}^{(j,k+1)} - \delta_y T_{j-1,k+1}\delta_z\rho_{j,k}^{(j,k+1)}\right) \\
&\times \left(\frac{\delta_y\rho_{j-1,k+1}^{(j,k+1)}}{dzw_k}\right) && \text{(B.196)} \\
\frac{\partial L_{j,k}^{(36)}}{\partial T_{i,k,j}} &= -\frac{A(36)V(36)}{|\nabla\rho(36)|^2}\left(\delta_z T_{j,k}\delta_y\rho_{j,k+1}^{(j,k+1)} - \delta_y T_{j,k+1}\delta_z\rho_{j,k}^{(j,k+1)}\right) \\
&\times \left(\frac{\delta_y\rho_{j,k+1}^{(j,k+1)}}{dzw_k}\right) && \text{(B.197)} \\
& && \text{(B.198)}
\end{aligned}
$$

Recombination of terms in the x-y plane

At this stage, the functional derivatives have been computed and so the diffusion operator is available. In order to organize the results in a more compact form, it is useful to rearrange terms in order to identify diffusive fluxes. The contributions to the diffusion operator from the x-y plane portion of the functional yield the x-y cross terms and one of two pieces each for the x-x and y-y diagonal terms. These terms are of no concern for the numerical stability issues (see Griffies et al, 1997). Therefore, the diffusion coefficients in each sub-cell in the x-y plane will be set to their *a priori* value A_I^o. Finally, in this section all reference to depth levels will be omitted except for dzt_k.

The terms from subcells $1+7+9$ are

$$\begin{aligned}
& \left(\frac{A_I^o}{4} dzt_k dyu_j \frac{\cos\phi_j^U}{\cos\phi_j^T}\right) \times \\
& \left(\frac{(\delta_y \rho_{i,j}^{(i,j)})(\delta_x \rho_{i-1,j}^{(i,j)} \delta_y T_{i,j} - \delta_x T_{i-1,j} \delta_y \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i-1,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2}\right) \\
+ \; & \left(\frac{A_I^o}{4} dzt_k dyu_j \frac{\cos\phi_j^U}{\cos\phi_j^T}\right) \times \\
& \left(\frac{(\delta_y \rho_{i-1,j}^{(i-1,j)})(\delta_x \rho_{i-1,j}^{(i-1,j)} \delta_y T_{i-1,j} - \delta_x T_{i-1,j} \delta_y \rho_{i-1,j}^{(i-1,j)})}{(\delta_x \rho_{i-1,j}^{(i-1,j)})^2 + (\delta_y \rho_{i-1,j}^{(i-1,j)})^2 + (\delta_z \rho_{i-1,j}^{(i-1,j)})^2}\right) \\
+ \; & \left(\frac{A_I^o}{4} dxu_{i-1} dzt_k \cos\phi_j^U\right) \times \\
& \left(\frac{(\delta_x \rho_{i-1,j}^{(i,j)})(\delta_x \rho_{i-1,j}^{(i,j)} \delta_y T_{i,j} - \delta_x T_{i-1,j} \delta_y \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i-1,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2}\right) \\
+ \; & \left(\frac{A_I^o}{4} dxu_{i-1} dzt_k \cos\phi_j^U\right) \times \\
& \left(\frac{(\delta_x \rho_{i-1,j+1}^{(i,j+1)})(\delta_x \rho_{i-1,j+1}^{(i,j+1)} \delta_y T_{i,j} - \delta_x T_{i-1,j+1} \delta_y \rho_{i,j}^{(i,j+1)})}{(\delta_x \rho_{i-1,j+1}^{(i,j+1)})^2 + (\delta_y \rho_{i,j}^{(i,j+1)})^2 + (\delta_z \rho_{i,j+1}^{(i,j+1)})^2}\right).
\end{aligned} \tag{B.199}$$

Introducing a summation allows these four terms to be combined into two terms

$$\begin{aligned}
& \left(\frac{A_I^o}{4} dzt_k dyu_j \frac{\cos\phi_j^U}{\cos\phi_j^T}\right) \times \\
& \sum_{ip=-1}^{0} \delta_y \rho_{i+ip,j}^{(i+ip,j)} \, \frac{\delta_x \rho_{i-1,j}^{(i+ip,j)} \delta_y T_{i+ip,j} - \delta_x T_{i-1,j} \delta_y \rho_{i+ip,j}^{(i+ip,j)}}{(\delta_x \rho_{i-1,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2} \\
+ \; & \left(\frac{A_I^o}{4} dxu_{i-1} dzt_k \cos\phi_j^U\right) \times \\
& \sum_{jq=0}^{1} \delta_x \rho_{i-1,j+jq}^{(i,j+jq)} \, \frac{\delta_x \rho_{i-1,j+jq}^{(i,j+jq)} \delta_y T_{i,j} - \delta_x T_{i-1,j+jq} \delta_y \rho_{i,j}^{(i,j+jq)}}{(\delta_x \rho_{i-1,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2}.
\end{aligned} \tag{B.200}$$

The terms from subcells $2+5+10$ are

$$\left(\frac{A_I^o}{4} dzt_k dyu_j \frac{\cos\phi_j^U}{\cos\phi_j^T}\right) \times$$

$$
\begin{aligned}
& \left(\frac{(\delta_y \rho_{i,j}^{(i,j)})(\delta_y \rho_{i,j}^{(i,j)} \delta_x T_{i,j} - \delta_y T_{i,j} \delta_x \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4} dzt_k dyu_j \frac{\cos \phi_j^U}{\cos \phi_j^T} \right) \times \\
& \left(\frac{(\delta_y \rho_{i+1,j}^{(i+1,j)})(\delta_y \rho_{i+1,j}^{(i+1,j)} \delta_x T_{i,j} - \delta_y T_{i+1,j} \delta_x \rho_{i,j}^{(i+1,j)})}{(\delta_x \rho_{i,j}^{(i+1,j)})^2 + (\delta_y \rho_{i+1,j}^{(i+1,j)})^2 + (\delta_z \rho_{i+1,j}^{(i+1,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4} dxu_i dzt_k \cos \phi_j^U \right) \times \\
& \left(\frac{(\delta_x \rho_{i,j}^{(i,j)})(\delta_x \rho_{i,j}^{(i,j)} \delta_y T_{i,j} - \delta_x T_{i,j} \delta_y \rho_{i,j}^{(i,j)})}{(\delta_x \rho_{i,j}^{(i,j)})^2 + (\delta_y \rho_{i,j}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4} dxu_i dzt_k \cos \phi_j^U \right) \times \\
& \left(\frac{(\delta_x \rho_{i,j+1}^{(i,j+1)})(\delta_x \rho_{i,j+1}^{(i,j+1)} \delta_y T_{i,j} - \delta_x T_{i,j+1} \delta_y \rho_{i,j}^{(i,j+1)})}{(\delta_x \rho_{i,j+1}^{(i,j+1)})^2 + (\delta_y \rho_{i,j}^{(i,j+1)})^2 + (\delta_z \rho_{i,j+1}^{(i,j+1)})^2} \right).
\end{aligned}
\tag{B.201}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
& \left(\frac{A_I^o}{4} dzt_k dyu_j \frac{\cos \phi_j^U}{\cos \phi_j^T} \right) \times \\
& \sum_{ip=0}^{1} \delta_y \rho_{i+ip,j}^{(i+ip,j)} \; \frac{\delta_y \rho_{i+ip,j}^{(i+ip,j)} \delta_x T_{i,j} - \delta_y T_{i+ip,j} \delta_x \rho_{i,j}^{(i+ip,j)}}{(\delta_x \rho_{i,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2} \\
+ & \left(\frac{A_I^o}{4} dxu_i dzt_k \cos \phi_j^U \right) \times \\
& \sum_{jq=0}^{1} \delta_x \rho_{i,j+jq}^{(i,j+jq)} \; \frac{\delta_x \rho_{i,j+jq}^{(i,j+jq)} \delta_y T_{i,j} - \delta_x T_{i,j+jq} \delta_y \rho_{i,j}^{(i,j+jq)}}{(\delta_x \rho_{i,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2}.
\end{aligned}
\tag{B.202}
$$

The terms from subcells $3 + 8 + 11$ are

$$
\begin{aligned}
& \left(\frac{A_I^o}{4} dzt_k dyu_{j-1} \frac{\cos \phi_{j-1}^U}{\cos \phi_j^T} \right) \times \\
& \left(\frac{(\delta_y \rho_{i,j-1}^{(i,j)})(\delta_x \rho_{i-1,j}^{(i,j)} \delta_y T_{i,j-1} - \delta_x T_{i-1,j} \delta_y \rho_{i,j-1}^{(i,j)})}{(\delta_x \rho_{i-1,j}^{(i,j)})^2 + (\delta_y \rho_{i,j-1}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4} dzt_k dyu_{j-1} \frac{\cos \phi_{j-1}^U}{\cos \phi_j^T} \right) \times \\
& \left(\frac{(\delta_y \rho_{i-1,j-1}^{(i-1,j)})(\delta_x \rho_{i-1,j}^{(i-1,j)} \delta_y T_{i-1,j-1} - \delta_x T_{i-1,j} \delta_y \rho_{i-1,j-1}^{(i-1,j)})}{(\delta_x \rho_{i-1,j}^{(i-1,j)})^2 + (\delta_y \rho_{i-1,j-1}^{(i-1,j)})^2 + (\delta_z \rho_{i-1,j}^{(i-1,j)})^2} \right) \\
+ & \left(\frac{A_I^o}{4} dxu_{i-1} dzt_k \cos \phi_{j-1}^U \right) \times \\
& \left(\frac{(\delta_x \rho_{i-1,j}^{(i,j)})(\delta_y \rho_{i,j-1}^{(i,j)} \delta_x T_{i-1,j} - \delta_y T_{i,j-1} \delta_x \rho_{i-1,j}^{(i,j)})}{(\delta_x \rho_{i-1,j}^{(i,j)})^2 + (\delta_y \rho_{i,j-1}^{(i,j)})^2 + (\delta_z \rho_{i,j}^{(i,j)})^2} \right)
\end{aligned}
$$

$$
\begin{aligned}
+ &\left(\frac{A_I^\circ}{4} dxu_{i-1} dzt_k \cos\phi^U_{j-1}\right) \times \\
&\left(\frac{(\delta_x\rho^{(i,j-1)}_{i-1,j-1})(\delta_y\rho^{(i,j-1)}_{i,j-1}\delta_x T_{i-1,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j-1)}_{i-1,j-1})}{(\delta_x\rho^{(i,j-1)}_{i-1,j-1})^2 + (\delta_y\rho^{(i,j-1)}_{i,j-1})^2 + (\delta_z\rho^{(i,j-1)}_{i,j-1})^2}\right). \qquad \text{(B.203)}
\end{aligned}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
&\left(\frac{A_I^\circ}{4} dzt_k dyu_{j-1}\frac{\cos\phi^U_{j-1}}{\cos\phi^T_j}\right) \times \\
&\sum_{ip=-1}^{0} \delta_y\rho^{(i+ip,j)}_{i+ip,j-1}\; \frac{\delta_x\rho^{(i+ip,j)}_{i-1,j}\delta_y T_{i+ip,j-1} - \delta_x T_{i-1,j}\delta_y\rho^{(i+ip,j)}_{i+ip,j-1}}{(\delta_x\rho^{(i+ip,j)}_{i-1,j})^2 + (\delta_y\rho^{(i+ip,j)}_{i+ip,j-1})^2 + (\delta_z\rho^{(i+ip,j)}_{i+ip,j})^2} \\
+ &\left(\frac{A_I^\circ}{4} dxu_{i-1} dzt_k \cos\phi^U_{j-1}\right) \times \\
&\sum_{jq=-1}^{0} \delta_x\rho^{(i,j+jq)}_{i-1,j+jq}\; \frac{\delta_y\rho^{(i,j+jq)}_{i,j-1}\delta_x T_{i-1,j+jq} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j+jq)}_{i-1,j+jq}}{(\delta_x\rho^{(i,j+jq)}_{i-1,j+jq})^2 + (\delta_y\rho^{(i,j+jq)}_{i,j-1})^2 + (\delta_z\rho^{(i,j+jq)}_{i,j+jq})^2}. \qquad \text{(B.204)}
\end{aligned}
$$

The terms from subcells $4 + 6 + 12$ are

$$
\begin{aligned}
&\left(\frac{A_I^\circ}{4} dzt_k dyu_{j-1}\frac{\cos\phi^U_{j-1}}{\cos\phi^T_j}\right) \times \\
&\left(\frac{(\delta_y\rho^{(i,j)}_{i,j-1})(\delta_y\rho^{(i,j)}_{i,j-1}\delta_x T_{i,j} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i,j})}{(\delta_x\rho^{(i,j)}_{i,j})^2 + (\delta_y\rho^{(i,j)}_{i,j-1})^2 + (\delta_z\rho^{(i,j)}_{i,j})^2}\right) \\
+ &\left(\frac{A_I^\circ}{4} dzt_k dyu_{j-1}\frac{\cos\phi^U_{j-1}}{\cos\phi^T_j}\right) \times \\
&\left(\frac{(\delta_y\rho^{(i+1,j)}_{i+1,j-1})(\delta_y\rho^{(i+1,j)}_{i+1,j-1}\delta_x T_{i,j} - \delta_y T_{i+1,j-1}\delta_x\rho^{(i+1,j)}_{i,j})}{(\delta_x\rho^{(i+1,j)}_{i,j})^2 + (\delta_y\rho^{(i+1,j)}_{i+1,j-1})^2 + (\delta_z\rho^{(i+1,j)}_{i+1,j})^2}\right) \\
+ &\left(\frac{A_I^\circ}{4} dxu_i dzt_k \cos\phi^U_{j-1}\right) \times \\
&\left(\frac{(\delta_x\rho^{(i,j)}_{i,j})(\delta_y\rho^{(i,j)}_{i,j-1}\delta_x T_{i,j} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j)}_{i,j})}{(\delta_x\rho^{(i,j)}_{i,j})^2 + (\delta_y\rho^{(i,j)}_{i,j})^2 + (\delta_z\rho^{(i,j)}_{i,j})^2}\right) \\
+ &\left(\frac{A_I^\circ}{4} dxu_i dzt_k \cos\phi^U_{j-1}\right) \times \\
&\left(\frac{(\delta_x\rho^{(i,j-1)}_{i,j-1})(\delta_y\rho^{(i,j-1)}_{i,j-1}\delta_x T_{i,j-1} - \delta_y T_{i,j-1}\delta_x\rho^{(i,j-1)}_{i,j-1})}{(\delta_x\rho^{(i,j-1)}_{i,j-1})^2 + (\delta_y\rho^{(i,j-1)}_{i,j-1})^2 + (\delta_z\rho^{(i,j-1)}_{i,j-1})^2}\right). \qquad \text{(B.205)}
\end{aligned}
$$

Introducing a summation allows these four terms to be combined into two terms

$$
\begin{aligned}
&\left(\frac{A_I^\circ}{4} dzt_k dyu_{j-1}\frac{\cos\phi^U_{j-1}}{\cos\phi^T_j}\right) \times \\
&\sum_{ip=0}^{1} \delta_y\rho^{(i+ip,j)}_{i+ip,j-1}\; \frac{\delta_y\rho^{(i+ip,j)}_{i+ip,j-1}\delta_x T_{i,j} - \delta_y T_{i+ip,j-1}\delta_x\rho^{(i+ip,j)}_{i,j}}{(\delta_x\rho^{(i+ip,j)}_{i,j})^2 + (\delta_y\rho^{(i+ip,j)}_{i+ip,j-1})^2 + (\delta_z\rho^{(i+ip,j)}_{i+ip,j})^2} \\
+ &\left(\frac{A_I^\circ}{4} dxu_i dzt_k \cos\phi^U_{j-1}\right) \times
\end{aligned}
$$

$$\sum_{jq=-1}^{0} \delta_x \rho_{i,j+jq}^{(i,j+jq)} \frac{\delta_y \rho_{i,j-1}^{(i,j+jq)} \delta_x T_{i,j+jq} - \delta_y T_{i,j-1} \delta_x \rho_{i,j+jq}^{(i,j+jq)}}{(\delta_x \rho_{i,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j-1}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2}. \tag{B.206}$$

There are now a total of eight terms. It is possible to combine the pairs appearing with the same summation into a total of four terms, each with a double summation. These four terms are

$$\begin{aligned}
&\left(\frac{A_I^o dzt_k}{4\cos\phi_j^T}\right) \sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=-1}^{0} \delta_y \rho_{i+ip,j+jq}^{(i+ip,j)} \times \\
&\frac{\delta_x \rho_{i-1,j}^{(i+ip,j)} \delta_y T_{i+ip,j+jq} - \delta_x T_{i-1,j} \delta_y \rho_{i+ip,j+jq}^{(i+ip,j)}}{(\delta_x \rho_{i-1,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j+jq}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2} \\
+ &\left(\frac{dzt_k A_I^o}{4\cos\phi_j^T}\right) \sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=0}^{1} \delta_y \rho_{i+ip,j+jq}^{(i+ip,j)} \times \\
&\frac{\delta_y \rho_{i+ip,j+jq}^{(i+ip,j)} \delta_x T_{i,j} - \delta_y T_{i+ip,j+jq} \delta_x \rho_{i,j}^{(i+ip,j)}}{(\delta_x \rho_{i,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j+jq}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2} \\
+ &\left(\frac{A_I^o \cos\phi_{j-1}^U dzt_k}{4}\right) \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{jq=-1}^{0} \delta_x \rho_{i+ip,j+jq}^{(i,j+jq)} \times \\
&\frac{\delta_y \rho_{i,j-1}^{(i,j+jq)} \delta_x T_{i+ip,j+jq} - \delta_y T_{i,j-1} \delta_x \rho_{i+ip,j+jq}^{(i,j+jq)}}{(\delta_x \rho_{i+ip,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j-1}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2} \\
+ &\left(\frac{A_I^o dzt_k \cos\phi_j^U}{4}\right) \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{jq=0}^{1} \delta_x \rho_{i+ip,j+jq}^{(i,j+jq)} \times \\
&\frac{\delta_x \rho_{i+ip,j+jq}^{(i,j+jq)} \delta_y T_{i,j} - \delta_x T_{i+ip,j+jq} \delta_y \rho_{i,j}^{(i,j+jq)}}{(\delta_x \rho_{i+ip,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2}.
\end{aligned} \tag{B.207}$$

In order to identify a difference operator, in the first term of equation (B.207), let the label ip run from 0 to 1 and adjust the i label accordingly. With this shift, the first and second terms take the form

$$\begin{aligned}
- &\left(\frac{A_I^o dzt_k}{4\cos\phi_j^T}\right) \sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=0}^{1} \delta_y \rho_{i-1+ip,j+jq}^{(i-1+ip,j)} \times \\
&\frac{\delta_x T_{i-1,j} \delta_y \rho_{i-1+ip,j+jq}^{(i-1+ip,j)} - \delta_x \rho_{i-1,j}^{(i-1+ip,j)} \delta_y T_{i-1+ip,j+jq}}{(\delta_x \rho_{i-1,j}^{(i-1+ip,j)})^2 + (\delta_y \rho_{i-1+ip,j+jq}^{(i-1+ip,j)})^2 + (\delta_z \rho_{i-1+ip,j}^{(i-1+ip,j)})^2} \\
+ &\left(\frac{dzt_k A_I^o}{4\cos\phi_j^T}\right) \sum_{jq=-1}^{0} dyu_{j+jq}\cos\phi_{j+jq}^U \sum_{ip=0}^{1} \delta_y \rho_{i+ip,j+jq}^{(i+ip,j)} \times \\
&\frac{\delta_x T_{i,j} \delta_y \rho_{i+ip,j+jq}^{(i+ip,j)} - \delta_x \rho_{i,j}^{(i+ip,j)} \delta_y T_{i+ip,j+jq}}{(\delta_x \rho_{i,j}^{(i+ip,j)})^2 + (\delta_y \rho_{i+ip,j+jq}^{(i+ip,j)})^2 + (\delta_z \rho_{i+ip,j}^{(i+ip,j)})^2}.
\end{aligned}$$

In order to center the elements of the first term to the west side of the T-cell i,k,j, take an average over z of the z-derivative squared in the denominator. This prescription will also bring the second term to the east side, which allows for a zonal difference operator across the T-cell to be identified. This averaging in the denominator does not disturb any of the numerical or physical properties of the scheme and it allows for an unambiguous placement of these terms without introducing computational modes. With this prescription, these two terms become

$$\left(\frac{A_I^\circ dxt_i dzt_k}{4}\right) \quad \delta_x \left(\sum_{jq=-1}^{0} dyu_{j+jq} \cos\phi_{j+jq}^U \sum_{ip=0}^{1} \delta_y \rho_{i-1+ip,j+jq}^{(i-1+ip,j)} \times \frac{\delta_x T_{i-1,j} \delta_y \rho_{i-1+ip,j+jq}^{(i-1+ip,j)} - \delta_x \rho_{i-1,j}^{(i-1+ip,j)} \delta_y T_{i-1+ip,j+jq}}{(\delta_x \rho_{i-1,j}^{(i-1+ip,j)})^2 + (\delta_y \rho_{i-1+ip,j+jq}^{(i-1+ip,j)})^2 + .5\sum_{kr=0,1}(\delta_z \rho_{i-1+ip,k-1+kr,j}^{(i-1+ip,k,j)})^2} \right).$$

It is convenient to shift the jq sum and to introduce the volume of the T-cell $V_{T_{i,k,j}} = dxt_i dyt_j \cos\phi_j^T dxt_k$, which yields

$$\left(\frac{A_I^\circ V_{T_{i,k,j}}}{4dyt_j \cos\phi_j^T}\right) \quad \delta_x \left(\sum_{jq=0}^{1} dyu_{j-1+jq} \cos\phi_{j-1+jq}^U \sum_{ip=0}^{1} \delta_y \rho_{i-1+ip,j-1+jq}^{(i-1+ip,j)} \times \frac{\delta_x T_{i-1,j} \delta_y \rho_{i-1+ip,j-1+jq}^{(i-1+ip,j)} - \delta_x \rho_{i-1,j}^{(i-1+ip,j)} \delta_y T_{i-1+ip,j-1+jq}}{(\delta_x \rho_{i-1,j}^{(i-1+ip,j)})^2 + (\delta_y \rho_{i-1+ip,j-1+jq}^{(i-1+ip,j)})^2 + .5\sum_{kr=0,1}(\delta_z \rho_{i-1+ip,k-1+kr,j}^{(i-1+ip,k,j)})^2} \right) \tag{B.208}$$

Now let jq run from 0 to 1 in the third term of equation (B.207) and shift j accordingly, to bring the third and fourth terms to

$$- \left(\frac{A_I^\circ dzt_k \cos\phi_{j-1}^U}{4}\right) \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{jq=0}^{1} \delta_x \rho_{i+ip,j-1+jq}^{(i,j-1+jq)} \times \frac{\delta_y T_{i,j-1} \delta_x \rho_{i+ip,j-1+jq}^{(i,j-1+jq)} - \delta_y \rho_{i,j-1}^{(i,j-1+jq)} \delta_x T_{i+ip,j-1+jq}}{(\delta_x \rho_{i+ip,j-1+jq}^{(i,j-1+jq)})^2 + (\delta_y \rho_{i,j-1}^{(i,j-1+jq)})^2 + (\delta_z \rho_{i,j-1+jq}^{(i,j-1+jq)})^2}$$
$$+ \left(\frac{A_I^\circ dzt_k \cos\phi_{j}^U}{4}\right) \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{jq=0}^{1} \delta_x \rho_{i+ip,j+jq}^{(i,j+jq)} \times \frac{\delta_x \rho_{i+ip,j+jq}^{(i,j+jq)} \delta_y T_{i,j} - \delta_x T_{i+ip,j+jq} \delta_y \rho_{i,j}^{(i,j+jq)}}{(\delta_x \rho_{i+ip,j+jq}^{(i,j+jq)})^2 + (\delta_y \rho_{i,j}^{(i,j+jq)})^2 + (\delta_z \rho_{i,j+jq}^{(i,j+jq)})^2},$$

which is the meridional difference

$$\left(\frac{A_I^\circ dyt_j dzt_k}{4}\right) \quad \delta_y \left(\cos\phi_{j-1}^U \sum_{ip=-1}^{0} dxu_{i+ip} \sum_{jq=0}^{1} \delta_x \rho_{i+ip,j-1+jq}^{(i,j-1+jq)} \times \frac{\delta_y T_{i,j-1} \delta_x \rho_{i+ip,j-1+jq}^{(i,j-1+jq)} - \delta_y \rho_{i,j-1}^{(i,j-1+jq)} \delta_x T_{i+ip,j-1+jq}}{(\delta_x \rho_{i+ip,j-1+jq}^{(i,j-1+jq)})^2 + (\delta_y \rho_{i,j-1}^{(i,j-1+jq)})^2 + .5\sum_{kr=0,1}(\delta_z \rho_{i,k-1+kr,j-1+jq}^{(i,k,j-1+jq)})^2} \right),$$

where the z-derivative in the denominator was averaged in order to bring it to the appropriate meridional face of the T-cell. Shifting the ip sum and introducing the volume factor $V_{T_{i,k,j}}$ yields

$$\left(\frac{A_I^\circ V_{T_{i,k,j}}}{4dxt_i \cos\phi_j^t}\right) \quad \delta_y \left(\cos\phi_{j-1}^U \sum_{ip=0}^{1} dxu_{i-1+ip} \sum_{jq=0}^{1} \delta_x \rho_{i-1+ip,j-1+jq}^{(i,j-1+jq)} \times \frac{\delta_y T_{i,j-1} \delta_x \rho_{i-1+ip,j-1+jq}^{(i,j-1+jq)} - \delta_y \rho_{i,j-1}^{(i,j-1+jq)} \delta_x T_{i-1+ip,j-1+jq}}{(\delta_x \rho_{i-1+ip,j-1+jq}^{(i,j-1+jq)})^2 + (\delta_y \rho_{i,j-1}^{(i,j-1+jq)})^2 + .5\sum_{kr=0,1}(\delta_z \rho_{i,k-1+kr,j-1+jq}^{(i,k,j-1+jq)})^2} \right) \tag{B.209}$$

Recombination of terms in the z-x plane

For the x-z plane, it is important to be explicit about the particular value of the diffusion coefficient to be used in order to ensure that the numerical stability criteria discussed in Griffies et al, (1997) is satisfied. Explicit reference to the latitude will be omitted except for dyt_j.

The three terms $13 + 19 + 21$ combine to form

$$\begin{aligned}
& \frac{dxu_{i-1}\cos\phi_j^T dyt_j}{4} A(13)\delta_x\rho_{i-1,k}^{(i,k)} \times \\
& \left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k-1}^{(i,k)}}{(\delta_x\rho_{i-1,k}^{(i,k)})^2 + (\delta_y\rho_{i,k}^{(i,k)})^2 + (\delta_z\rho_{i,k-1}^{(i,k)})^2}\right) \\
+ \quad & \frac{dxu_{i-1}\cos\phi_j^T dyt_j}{4} A(21)\delta_x\rho_{i-1,k-1}^{(i,k-1)} \times \\
& \left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1,k-1}^{(i,k-1)} - \delta_x T_{i-1,k-1}\delta_z\rho_{i,k-1}^{(i,k-1)}}{(\delta_x\rho_{i-1,k-1}^{(i,k-1)})^2 + (\delta_y\rho_{i,k-1}^{(i,k-1)})^2 + (\delta_z\rho_{i,k-1}^{(i,k-1)})^2}\right) \\
+ \quad & \frac{dzw_{k-1}dyt_j}{4} A(13)\delta_z\rho_{i,k-1}^{(i,k)} \times \\
& \left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1,k}^{(i,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i,k-1}^{(i,k)}}{(\delta_x\rho_{i-1,k}^{(i,k)})^2 + (\delta_y\rho_{i,k}^{(i,k)})^2 + (\delta_z\rho_{i,k-1}^{(i,k)})^2}\right) \\
+ \quad & \frac{dzw_{k-1}dyt_j}{4} A(19)\delta_z\rho_{i-1,k-1}^{(i-1,k)} \times \\
& \left(\frac{\delta_z T_{i-1,k-1}\delta_x\rho_{i-1,k}^{(i-1,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i-1,k-1}^{(i-1,k)}}{(\delta_x\rho_{i-1,k}^{(i-1,k)})^2 + (\delta_y\rho_{i-1,k}^{(i-1,k)})^2 + (\delta_z\rho_{i-1,k-1}^{(i-1,k)})^2}\right).
\end{aligned} \tag{B.210}$$

The diffusion coefficient $A(13)$ is chosen to make the x-projection of the isopycnal slope

$$Sx^{(i,k)}(i-1,k|i,k-1) \equiv -\frac{\delta_x\rho_{i-1,k}^{(i,k)}}{\delta_z\rho_{i,k-1}^{(i,k)}} \tag{B.211}$$

satisfy

$$S_{(-)} \leq |Sx^{(i,k)}(i-1,k|i,k-1)| \leq S_{(-)}^{-1} \tag{B.212}$$

if the grid parameter

$$\delta = min(\frac{\Delta\, x_i \Delta\, z_k}{4A_I \Delta\, t}) \tag{B.213}$$

is $< 1/2$, where one and only one of the grid spacing is in the vertical. Otherwise, no rescaling of the diffusion coefficient is necessary to maintain numerical stability. To make explicit this association, introduce the notation

$$A(13) \equiv Ax^{(i,k)}(i-1,k|i,k-1). \tag{B.214}$$

Likewise, let

$$A(19) \equiv Ax^{(i-1,k)}(i-1,k|i-1,k-1), \tag{B.215}$$

$$A(21) \equiv Ax^{(i,k-1)}(i-1,k-1|i,k-1) \tag{B.216}$$

denote the other diffusion coefficients whose values are set according the value of their respective slopes. Introducing summation notation, the three terms $13 + 19 + 21$ can now be written

$$\begin{aligned}
&\frac{dxu_{i-1}\cos\phi_j^T dyt_j}{4}\sum_{kr=-1}^{0} && Ax^{(i,k+kr)}(i-1,k+kr|i,k-1)\delta_x\rho^{(i,k+kr)}_{i-1,k+kr}\times\\
& && \left(\frac{\delta_z T_{i,k-1}\delta_x\rho^{(i,k+kr)}_{i-1,k+kr}-\delta_x T_{i-1,k+kr}\delta_z\rho^{(i,k+kr)}_{i,k-1}}{(\delta_x\rho^{(i,k+kr)}_{i-1,k+kr})^2+(\delta_y\rho^{(i,k+kr)}_{i,k+kr})^2+(\delta_z\rho^{(i,k+kr)}_{i,k-1})^2}\right)\\
&+\frac{dzw_{k-1}dyt_j}{4}\sum_{ip=-1}^{0} && Ax^{(i+ip,k)}(i-1,k|i+ip,k-1)\delta_z\rho^{(i+ip,k)}_{i+ip,k-1}\times\\
& && \left(\frac{\delta_z T_{i+ip,k-1}\delta_x\rho^{(i+ip,k)}_{i-1,k}-\delta_x T_{i-1,k}\delta_z\rho^{(i+ip,k)}_{i+ip,k-1}}{(\delta_x\rho^{(i+ip,k)}_{i-1,k})^2+(\delta_y\rho^{(i+ip,k)}_{i+ip,k})^2+(\delta_z\rho^{(i+ip,k)}_{i+ip,k-1})^2}\right).
\end{aligned} \tag{B.217}$$

Similar considerations for the three other triads lead to the subcells $14 + 17 + 22$ becoming

$$\begin{aligned}
&\frac{dxu_{i}\cos\phi_j^T dyt_j}{4}\sum_{kr=-1}^{0} && Ax^{(i,k+kr)}(i,k+kr|i,k-1)\delta_x\rho^{(i,k+kr)}_{i,k+kr}\times\\
& && \left(\frac{\delta_z T_{i,k-1}\delta_x\rho^{(i,k+kr)}_{i,k+kr}-\delta_x T_{i,k+kr}\delta_z\rho^{(i,k+kr)}_{i,k-1}}{(\delta_x\rho^{(i,k+kr)}_{i,k+kr})^2+(\delta_y\rho^{(i,k+kr)}_{i,k+kr})^2+(\delta_z\rho^{(i,k+kr)}_{i,k-1})^2}\right)\\
&+\frac{dzw_{k-1}dyt_j}{4}\sum_{ip=0}^{1} && Ax^{(i+ip,k)}(i,k|i+ip,k-1)\delta_z\rho^{(i+ip,k)}_{i+ip,k-1}\times\\
& && \left(\frac{\delta_x T_{i,k}\delta_z\rho^{(i+ip,k)}_{i+ip,k-1}-\delta_z T_{i+ip,k-1}\delta_x\rho^{(i+ip,k)}_{i,k}}{(\delta_x\rho^{(i+ip,k)}_{i,k})^2+(\delta_y\rho^{(i+ip,k)}_{i+ip,k})^2+(\delta_z\rho^{(i+ip,k)}_{i+ip,k-1})^2}\right),
\end{aligned} \tag{B.218}$$

the subcells $15 + 20 + 23$ becoming

$$\begin{aligned}
&\frac{dxu_{i-1}\cos\phi_j^T dyt_j}{4}\sum_{kr=0}^{1} && Ax^{(i,k+kr)}(i-1,k+kr|i,k)\delta_x\rho^{(i,k+kr)}_{i-1,k+kr}\times\\
& && \left(\frac{\delta_x T_{i-1,k+kr}\delta_z\rho^{(i,k+kr)}_{i,k}-\delta_z T_{i,k}\delta_x\rho^{(i,k+kr)}_{i-1,k+kr}}{(\delta_x\rho^{(i,k+kr)}_{i-1,k+kr})^2+(\delta_y\rho^{(i,k+kr)}_{i,k+kr})^2+(\delta_z\rho^{(i,k+kr)}_{i,k})^2}\right)\\
&+\frac{dzw_{k}dyt_j}{4}\sum_{ip=-1}^{0} && Ax^{(i+ip,k)}(i-1,k|i+ip,k)\delta_z\rho^{(i+ip,k)}_{i+ip,k}\times\\
& && \left(\frac{\delta_z T_{i+ip,k}\delta_x\rho^{(i+ip,k)}_{i-1,k}-\delta_x T_{i-1,k}\delta_z\rho^{(i+ip,k)}_{i+ip,k}}{(\delta_x\rho^{(i+ip,k)}_{i-1,k})^2+(\delta_y\rho^{(i+ip,k)}_{i+ip,k})^2+(\delta_z\rho^{(i+ip,k)}_{i+ip,k})^2}\right),
\end{aligned} \tag{B.219}$$

and the subcells $16 + 18 + 24$ becoming

$$\begin{aligned}
&\frac{dxu_{i}\cos\phi_j^T dyt_j}{4}\sum_{kr=0}^{1} && Ax^{(i,k+kr)}(i,k+kr|i,k)\delta_x\rho^{(i,k+kr)}_{i,k+kr}\times\\
& && \left(\frac{\delta_x T_{i,k+kr}\delta_z\rho^{(i,k+kr)}_{i,k}-\delta_z T_{i,k}\delta_x\rho^{(i,k+kr)}_{i,k+kr}}{(\delta_x\rho^{(i,k+kr)}_{i,k+kr})^2+(\delta_y\rho^{(i,k+kr)}_{i,k+kr})^2+(\delta_z\rho^{(i,k+kr)}_{i,k})^2}\right)
\end{aligned}$$

$$+\frac{dzw_k dyt_j}{4}\sum_{ip=0}^{1} Ax^{(i+ip,k)}(i,k|i+ip,k)\delta_z\rho_{i+ip,k}^{(i+ip,k)} \times \left(\frac{\delta_x T_{i,k}\delta_z\rho_{i+ip,k}^{(i+ip,k)} - \delta_z T_{i+ip,k}\delta_x\rho_{i,k}^{(i+ip,k)}}{(\delta_x\rho_{i,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k}^{(i+ip,k)})^2}\right). \quad (B.220)$$

There are now a total of eight terms. It is possible to combine the pairs appearing with the same summation into a total of four terms, each with a double summation. These four terms are

$$\frac{\cos\phi_j^T dyt_j}{4}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=-1}^{0} Ax^{(i,k+kr)}(i+ip,k+kr|i,k-1)\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} \times \left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} - \delta_x T_{i+ip,k+kr}\delta_z\rho_{i,k-1}^{(i,k+kr)}}{(\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k-1}^{(i,k+kr)})^2}\right)$$

$$+\frac{\cos\phi_j^T dyt_j}{4}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=0}^{1} Ax^{(i,k+kr)}(i+ip,k+kr|i,k)\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} \times \left(\frac{\delta_x T_{i+ip,k+kr}\delta_z\rho_{i,k}^{(i,k+kr)} - \delta_z T_{i,k}\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)}}{(\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k}^{(i,k+kr)})^2}\right)$$

$$+\frac{dyt_j}{4}\sum_{kr=-1}^{0} dzw_{k+kr}\sum_{ip=0}^{1} Ax^{(i+ip,k)}(i,k|i+ip,k+kr)\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} \times \left(\frac{\delta_x T_{i,k}\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} - \delta_z T_{i+ip,k+kr}\delta_x\rho_{i,k}^{(i+ip,k)}}{(\delta_x\rho_{i,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)})^2}\right)$$

$$+\frac{dyt_j}{4}\sum_{kr=-1}^{0} dzw_{k+kr}\sum_{ip=-1}^{0} Ax^{(i+ip,k)}(i-1,k|i+ip,k+kr)\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} \times \left(\frac{\delta_z T_{i+ip,k+kr}\delta_x\rho_{i-1,k}^{(i+ip,k)} - \delta_x T_{i-1,k}\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)})^2}\right) \quad (B.221)$$

In the first term of equation (B.221), let the kr sum run from 0 to 1 and adjust the k labels appropriately to get the first and second terms into the form

$$\frac{\cos\phi_j^T dyt_j}{4}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=0}^{1} Ax^{(i,k-1+kr)}(i+ip,k-1+kr|i,k-1)\delta_x\rho_{i+ip,k-1+kr}^{(i,k-1+kr)} \times \left(\frac{\delta_z T_{i,k-1}\delta_x\rho_{i+ip,k-1+kr}^{(i,k-1+kr)} - \delta_x T_{i+ip,k-1+kr}\delta_z\rho_{i,k-1}^{(i,k-1+kr)}}{(\delta_x\rho_{i+ip,k-1+kr}^{(i,k-1+kr)})^2 + (\delta_y\rho_{i,k-1+kr}^{(i,k-1+kr)})^2 + (\delta_z\rho_{i,k-1}^{(i,k-1+kr)})^2}\right)$$

$$-\frac{\cos\phi_j^T dyt_j}{4}\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=0}^{1} Ax^{(i,k+kr)}(i+ip,k+kr|i,k)\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} \times \left(\frac{\delta_z T_{i,k}\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)} - \delta_x T_{i+ip,k+kr}\delta_z\rho_{i,k}^{(i,k+kr)}}{(\delta_x\rho_{i+ip,k+kr}^{(i,k+kr)})^2 + (\delta_y\rho_{i,k+kr}^{(i,k+kr)})^2 + (\delta_z\rho_{i,k}^{(i,k+kr)})^2}\right). \quad (B.222)$$

With an average on the y-derivative in the denominator, these two terms can be identified as the vertical difference

$$\frac{\cos\phi_j^T dyt_j dzt_k}{4}\delta_z\left(\sum_{ip=-1}^{0} dxu_{i+ip}\sum_{kr=0}^{1} Ax^{(i,k-1+kr)}(i+ip,k-1+kr|i,k-1)(\delta_x\rho_{i+ip,k-1+kr}^{(i,k-1+kr)}) \frac{\delta_z T_{i,k-1}\delta_x\rho_{i+ip,k-1+kr}^{(i,k-1+kr)} - \delta_x T_{i+ip,k-1+kr}\delta_z\rho_{i,k-1}^{(i,k-1+kr)}}{(\delta_x\rho_{i+ip,k-1+kr}^{(i,k-1+kr)})^2 + .5\sum_{jq=0,1}(\delta_y\rho_{i,k-1+kr,j-1+jq}^{(i,k-1+kr,j)})^2 + (\delta_z\rho_{i,k-1}^{(i,k-1+kr)})^2}\right).$$

Introducing the volume factor $V_{T_{i,k,j}}$ and shifting the ip sum, this difference becomes

$$\frac{V_{T_{i,k,j}}}{4dxt_i}\delta_z\left(\sum_{ip=0}^{1} dxu_{i-1+ip}\sum_{kr=0}^{1} Ax^{(i,k-1+kr)}(i-1+ip,k-1+kr|i,k-1)(\delta_x\rho_{i-1+ip,k-1+kr}^{(i,k-1+kr)}) \frac{\delta_z T_{i,k-1}\delta_x\rho_{i-1+ip,k-1+kr}^{(i,k-1+kr)} - \delta_x T_{i-1+ip,k-1+kr}\delta_z\rho_{i,k-1}^{(i,k-1+kr)}}{(\delta_x\rho_{i-1+ip,k-1+kr}^{(i,k-1+kr)})^2 + .5\sum_{jq=0,1}(\delta_y\rho_{i,k-1+kr,j-1+jq}^{(i,k-1+kr,j)})^2 + (\delta_z\rho_{i,k-1}^{(i,k-1+kr)})^2}\right). \tag{B.223}$$

In the fourth term of equation (B.221), shift the ip sum to 0,1 and adjust the i label accordingly to have the third and fourth terms take the form

$$\begin{aligned}
&\frac{dyt_j}{4}\sum_{kr=-1}^{0} dzw_{k+kr}\sum_{ip=0}^{1} Ax^{(i+ip,k)}(i,k|i+ip,k+kr)\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)}\times \\
&\qquad\left(\frac{\delta_x T_{i,k}\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)} - \delta_z T_{i+ip,k+kr}\delta_x\rho_{i,k}^{(i+ip,k)}}{(\delta_x\rho_{i,k}^{(i+ip,k)})^2 + (\delta_y\rho_{i+ip,k}^{(i+ip,k)})^2 + (\delta_z\rho_{i+ip,k+kr}^{(i+ip,k)})^2}\right) \\
-&\frac{dyt_j}{4}\sum_{kr=-1}^{0} dzw_{k+kr}\sum_{ip=0}^{1} Ax^{(i-1+ip,k)}(i-1,k|i-1+ip,k+kr)\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)}\times \\
&\qquad\left(\frac{\delta_x T_{i-1,k}\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)} - \delta_z T_{i-1+ip,k+kr}\delta_x\rho_{i-1,k}^{(i-1+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i-1+ip,k)})^2 + (\delta_y\rho_{i-1+ip,k}^{(i-1+ip,k)})^2 + (\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)})^2}\right),
\end{aligned}$$

which becomes the zonal difference upon averaging the y-difference in the denominator

$$\frac{dxt_i\cos\phi_j^T dyt_j}{4}\delta_x\left(\sum_{kr=-1}^{0} dzw_{k+kr}\sum_{ip=0}^{1} Ax^{(i-1+ip,k)}(i-1,k|i-1+ip,k+kr)(\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)}) \frac{\delta_x T_{i-1,k}\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)} - \delta_z T_{i-1+ip,k+kr}\delta_x\rho_{i-1,k}^{(i-1+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i-1+ip,k)})^2 + .5\sum_{jq=0,1}(\delta_y\rho_{i-1+ip,k,j-1+jq}^{(i-1+ip,k,j)})^2 + (\delta_z\rho_{i-1+ip,k+kr}^{(i-1+ip,k)})^2}\right). \tag{B.224}$$

Introducing the volume factor $V_{T_{i,k,j}}$ and shifting the kr sum, this difference becomes

$$\frac{V_{T_{i,k,j}}}{4dzt_k}\delta_x\left(\sum_{kr=0}^{1} dzw_{k-1+kr}\sum_{ip=0}^{1} Ax^{(i-1+ip,k)}(i-1,k|i-1+ip,k-1+kr)(\delta_z\rho_{i-1+ip,k-1+kr}^{(i-1+ip,k)}) \frac{\delta_x T_{i-1,k}\delta_z\rho_{i-1+ip,k-1+kr}^{(i-1+ip,k)} - \delta_z T_{i-1+ip,k-1+kr}\delta_x\rho_{i-1,k}^{(i-1+ip,k)}}{(\delta_x\rho_{i-1,k}^{(i-1+ip,k)})^2 + .5\sum_{jq=0,1}(\delta_y\rho_{i-1+ip,k,j-1+jq}^{(i-1+ip,k,j)})^2 + (\delta_z\rho_{i-1+ip,k-1+kr}^{(i-1+ip,k)})^2}\right). \tag{B.225}$$

Recombination of terms in the y-z plane

The y-z plane is done similarly to the z-x plane. Its solution can be read from the z-x solution with the appropriate index and $\cos\phi$ changes. The result is

$$\begin{aligned}
& \frac{V_{T_{i,k,j}}}{4\cos\phi_j^T\, dyt_j}\,\delta_z \left(\sum_{jq=0}^{1} \cos\phi^U_{j-1+jq} dyu_{j-1+jq} \sum_{kr=0}^{1} Ay^{(k-1+kr,j)}(k-1+kr, j-1+jq|k-1,j) \right. \\
& \left. \delta_y\rho^{(k-1+kr,j)}_{k-1+kr,j-1+jq} \frac{\delta_z T_{k-1,j}\delta_y\rho^{(k-1+kr,j)}_{k-1+kr,j-1+jq} - \delta_y T_{k-1+kr,j-1+jq}\delta_z\rho^{(k-1+kr,j)}_{k-1,j}}{.5\sum_{ip=0,1}(\delta_x\rho^{(i,k-1+kr,j)}_{i-1+ip,k-1+kr,j})^2 + (\delta_y\rho^{(k-1+kr,j)}_{k-1+kr,j-1+jq})^2 + (\delta_z\rho^{(k-1+kr,j)}_{k-1,j})^2} \right) \\
+ \quad & \frac{V_{T_{i,k,j}}}{4dzt_k\cos\phi_j^T}\,\delta_y \left(\sum_{kr=0}^{1} dzw_{k-1+kr} \sum_{jq=0}^{1} Ay^{(k,j-1+jq)}(k, j-1|k-1+kr, j-1+jq) \right. \\
& \left. \delta_z\rho^{(k,j-1+jq)}_{k-1+kr,j-1+jq} \frac{\delta_y T_{k,j-1}\delta_z\rho^{(k,j-1+jq)}_{k-1+kr,j-1+jq} - \delta_z T_{k-1+kr,j-1+jq}\delta_y\rho^{(k,j-1+jq)}_{k,j-1}}{.5\sum_{ip=0,1}(\delta_x\rho^{(k,j-1+jq)}_{i-1+ip,k,j-1+jq})^2 + (\delta_y\rho^{(k,j-1+jq)}_{k,j-1})^2 + (\delta_z\rho^{(k,j-1+jq)}_{k-1+kr,j-1+jq})^2} \right)
\end{aligned} \tag{B.226}$$

B.2.9 Diffusive fluxes

The above details provide the explicit form for the discretization of the diffusion operator. All that is needed is to divide out the volume factor $V_{T_{i,k,j}}$ according to equation (B.24). As an additional step, it is useful to identify diffusive fluxes since the MOM2 model is coded in terms of fluxes across cell faces. Identifying diffusive fluxes also allows for an easier implementation of the no-flux boundary conditions than working directly with the diffusion operator. These forms are provided in Section 15.16.3 in the main part of the manual.

B.3 General comments

This section presents some general comments and details regarding the implementation of the new isopycnal scheme.

B.3.1 Isopycnal diffusion operator

The isopycnal diffusion operator is given by the divergence of the diffusive fluxes

$$R[T]_{i,k,j} = -\left(\delta_x F^x_{i-1,k,j} + \frac{1}{\cos\phi_j^T}\,\delta_y F^y_{i,k,j-1} + \delta_z F^z_{i,k-1,j} \right) \tag{B.227}$$

The fluxes admit no computational modes since only nearest neighbor differences are employed. This operator is defined at the center of the T-grid cell $T_{i,k,jrow}$. Exposing the spherical coordinates, gives

$$\begin{aligned}
R[T]_{i,k,j} = \; &= DIFF_Tx_{i,k,j} + DIFF_Ty_{i,k,j} + DIFF_Tz_{i,k,j} \\
&= \frac{\delta_\lambda(\mathit{diff_fe}_{i-1,k,j})}{\cos\phi^T_{jrow}} + \frac{\delta_\phi(\mathit{diff_fn}_{i,k,j-1})}{\cos\phi^T_{jrow}} + \delta_z(\mathit{diff_fb}_{i,k-1,j}),
\end{aligned} \tag{B.228}$$

where the diffusive flux vector in the appendix is related to that of the model through

$$\vec{F}_{i,k,j} = -(\mathit{diff\_fe}_{i,k,j}\ \mathit{diff\_fn}_{i,k,j}\ \mathit{diff\_fb}_{i,k,j}). \tag{B.229}$$

The flux vector has components defined at the center of the east, north, and bottom of cell $T_{i,k,jrow}$respectively. The diffusive flux vector satisfies the appropriate flux conditions at the domain boundaries. At the walls, there is no normal flux; at the bottom, there is the option of specifying a bottom flux (for studying, say, geothermal processes; typically assumed zero), and at the top, surface tracer flux information is fed into the vertical flux component. In general, these flux conditions are enforced in the model using the mask array $tmask_{i,k,j}$.

The shifting of the labels on the respective diffusive flux components is necessary in order to bring the difference of the fluxes onto the center of the cell $T_{i,k,jrow}$. For example, the difference

$$\delta_\lambda(diff\_fe_{i-1,k,j}) = \frac{diff\_fe_{i,k,j} - diff\_fe_{i-1,k,j}}{dxt_i} \tag{B.230}$$

is defined at the center of $T_{i,k,jrow}$, whereas $\delta_\lambda(diff\_fe_{i,k,j})$ is defined at the center of $T_{i+1,k,jrow}$. The denominator dxt_i represents the grid distance between the east and west faces of $T_{i,k,jrow}$. The fluxes in the meridional and vertical directions follow similarly, which yields

$$\begin{aligned} R[T]_{i,k,j} &= \frac{diff\_fe_{i,k,j} - diff\_fe_{i-1,k,j}}{\cos\phi^T_{jrow}\cdot dxt_i} + \frac{diff\_fn_{i,k,j} - diff\_fn_{i,k,j-1}}{\cos\phi^T_{jrow}\cdot dyt_{jrow}} \\ &+ \frac{diff\_fb_{i,k-1,j} - diff\_fb_{i,k,j}}{dzt_k}. \end{aligned} \tag{B.231}$$

B.3.2 Reference points and grid stencil

With a pressure dependent equation of state for seawater, the particular choice made for the reference points needed to define the density gradients is very important. For example, not all choices provide for a dissipative diffusion operator. The choice for reference points was made at the stage of discretizing the functional. The crucial property that must be preserved in order to derive a dissipative diffusion operator is the sign definiteness of the functional since this object is the source for the tracer variance tendency (see equation (B.12)). Preserving the sign definiteness is trivial when choosing reference points at the stage of discretizing the functional, but is more difficult and generally not preserved if choosing these points at a later stage of the derivation.

The details for choosing these points are given in Section B.2.6 from the perspective of the functional. For the present purposes, it is sufficient to spell out the grid stencil used for the small angle flux F^x since the full tensor and other flux components have similar stencils. Figure B.4 shows this stencil as implied by the formula

$$\begin{aligned} -F^{x\ small}_{i,k,j} &= K^{11\ small}_{i,k,j}\delta_x T_{i,k,j} \\ &- \frac{1}{4dzt_k}\sum_{kr=0}^{1} dzw_{k-1+kr}\sum_{ip=0}^{1} A^{(i+ip,k,j)}_{ez}(i,k|i+ip,k-1+kr,j) \\ &\quad \frac{\delta_z T_{i+ip,k-1+kr,j}\ \delta_x\rho^{(i+ip,k,j)}_{i,k,j}}{\delta_z\rho^{(i+ip,k,j)}_{i+ip,k-1+kr,j}}. \end{aligned} \tag{B.232}$$

This flux is defined at the east face of the T-cell i,k,j, which is located in between the T-points i,k,j and $i+1,k,j$ shown in Figure B.4. The two reference points to be used in computing this flux are at the T-points i,k,j and $i+1,k,j$. The horizontal line represents the zonal density gradient, which is computed both with reference to the i,k point and to the $i+1,k$ point. The four vertical lines represent the four vertical density gradients which are computed with reference to the two corresponding points at level k.

B.3.3 Rescaling the along isopycnal diffusion coefficient

As discussed in Griffies et al, (1997), prescriptions for dealing with the isopycnal fluxes when the isopycnal slopes go outside the bounds of the numerically stable regime are *ad hoc*. The *ad hoc* nature of the rescaling is not surprising since the ability of the numerical grid to compute the diffusion operator with sufficient integrity breaks down for such isopycnal slopes. Three prescriptions have been implemented in MOM2 to handle these *limbo-regions*, two for the small angle tensor and one for the full tensor. Each prescription focuses on the x-z, y-z, z-x, and z-y off-diagonal components and rescales the diffusion coefficient for these terms independently and in a local manner, thus introducing the four diffusion coefficients A^{ez}, A^{nz}, A^{bx}, and A^{by}. Local rescaling provides for a unique definition of the fluxes, which is important in order to conserve the fluxes in the model.

To determine the value of each of the four diffusion coefficients, the new scheme requires the checking of slopes for four surrounding triads of grid points. Within a particular triad, the diffusion coefficient is rescaled according to the numerical stability constraints. Therefore, each of the four diffusion coefficients A^{ez}, A^{nz}, A^{bx}, and A^{by} contains four sub-components. Each particular triad can be involved in specifying more than one diffusion coefficient (e.g., see Figures B.2 and B.3). Hence, it is necessary to maintain the completely local specification of the sub-components of the diffusion coefficients. Otherwise, non-unique fluxes can arise. Additionally, without this local specification, the variance reducing property of the scheme is no longer guaranteed.

The need to rescale diffusion coefficients adds a bit of complexity to the algorithm as well as increased time and memory requirements. Additionally, the rescaling introduces an unphysical spatial dependence to the diffusion coefficient which adds a corresponding advective transport proportional to the divergence of the scaled diffusion coefficient. The introduction of the full tensor with a well resolved grid (i.e., a grid for which $\delta > 1/2$), provides a method which requires no rescaling and no unphysical advective transport. Additionally, there is a savings of memory due to the ability to collapse the four diffusion coefficients A^{ez}, A^{nz}, A^{bx}, and A^{by} into a single constant (e.g., for the simple case of A_I^o a constant, all diffusion coefficients are constant in space and time). The down-side of such a model is that maintaining $\delta > 1/2$ might involve relatively small time steps. Regardless, the ability to run with the full isopycnal diffusion tensor, without the *ad hoc* rescaling, provides for a self-consistent means of numerically investigating the issues of parameterizing tracer mixing for intermediate and large isopycnal slopes.

B.3.4 Vertical diffusion equation

The ability to identify the 3,3 component of the Redi tensor is very useful since it enables the vertical diffusion equation to be solved implicitly in the same manner as with the old scheme.

B.3.5 Diabatic piece

Full tensor

The diffusion coefficients $A(n)$ in the previous discussion of the full tensor corresponded to along isopycnal diffusion, minus any explicit diapycnal diffusion (e.g., see equations (B.18)-(B.20)). For modest slopes, these two coefficients are very different, with the along isopycnal diffusion roughly 10^7 larger. When the full tensor needs rescaling for the intermediate slopes in which it is not stable, the rescaled along isopycnal diffusivity is still roughly $10^3 - 10^4$ larger than the diapycnal diffusivity. Therefore, the diffusion coefficient is essentially the along isopycnal diffusivity. The discretization of diabatic diffusion therefore reduces to the discretization of

$\partial_m(A_D \partial_m T)$, with the vertical piece done implicitly along with the $K^{3,3}$ piece of the isopycnal diffusion tensor.

Small tensor

For the small tensor, the diabatic piece is simply $\partial_z(A_D \partial_z T)$, which should be done implicitly along with the $K^{3,3}$ piece of the isopycnal diffusion tensor.

B.3.6 Highlighting the different average operations

There are numerous differences between the new scheme and that implemented by Cox (1987). One difference is in the form of the averaging operations. In particular, for the small slope fluxes, taking a uniform grid in the respective directions and neglecting the specification of the reference points allows for the double sums in the new scheme to collapse to familiar averaging operators. For example, consider the x-z fluxes in the small angle limit. The old scheme used the discretization

$$-F^x_{i,k} = A_I \left(\delta_x T_{i,k} - \frac{\delta_z \overline{T}^{x,z}_{i,k-1}}{\delta_z \overline{\rho}^{x,z}_{i,k-1}} \, \delta_x \rho_{i,k} \right) \tag{B.233}$$

$$-F^z_{i,k} = -\frac{A_I}{\delta_z \rho_{i,k}} \left(\delta_x \overline{T}^{x,z}_{i-1,k} \, \delta_x \overline{\rho}^{x,z}_{i-1,k} \right) + A_I \frac{\delta_z T_{i,k}}{(\delta_z \rho_{i,k})^2} (\delta_x \overline{\rho}^{x,z}_{i-1,k})^2, \tag{B.234}$$

whereas the new fluxes (with uniform grid, neglecting the reference points, and assuming constant diffusion coefficients) are given by

$$-F^x_{i,k,j} = A_I \left(\delta_x T_{i,k,j} - \overline{\left(\frac{\delta_z T_{i,k-1,j}}{\delta_z \rho_{i,k-1,j}} \right)}^{x,z} \delta_x \rho_{i,k,j} \right) \tag{B.235}$$

$$-F^z_{i,k,j} = -\frac{A_I}{\delta_z \rho_{i,k,j}} \left(\overline{\delta_x T_{i-1,k,j} \, \delta_x \rho_{i-1,k,j}}^{x,z} \right) + A_I \frac{\delta_z T_{i,k,j}}{(\delta_z \rho_{i,k,j})^2} \, \overline{(\delta_x \rho_{i-1,k,j})^2}^{x,z}. \tag{B.236}$$

The difference between the fluxes is related to how the averages are applied to the z-derivative terms for the x-flux, and how the averages are applied to the x-derivative terms in the z-flux. Namely, the new scheme applies averages over a product or ratio of fields rather than individually as done in the original scheme. The new procedure provides for a scheme that has no computational modes. Indeed, this discretization might be an inspired guess by one motivated by the desire to eliminate computational modes. It is unclear how one could be inspired to guess the details of the grid weights and the reference points.

Appendix B contributed by
Stephen M. Griffies
smg@gfdl.gov
Last revised October 31, 1996

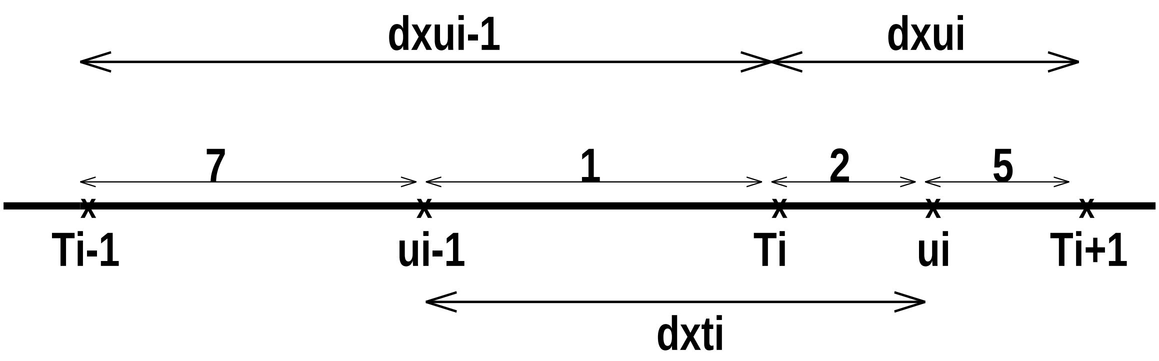


Figure B.1: One dimensional grid with subcells 7,1,2,5 corresponding to the x-axis cells in the x-y plane shown in the next figure.

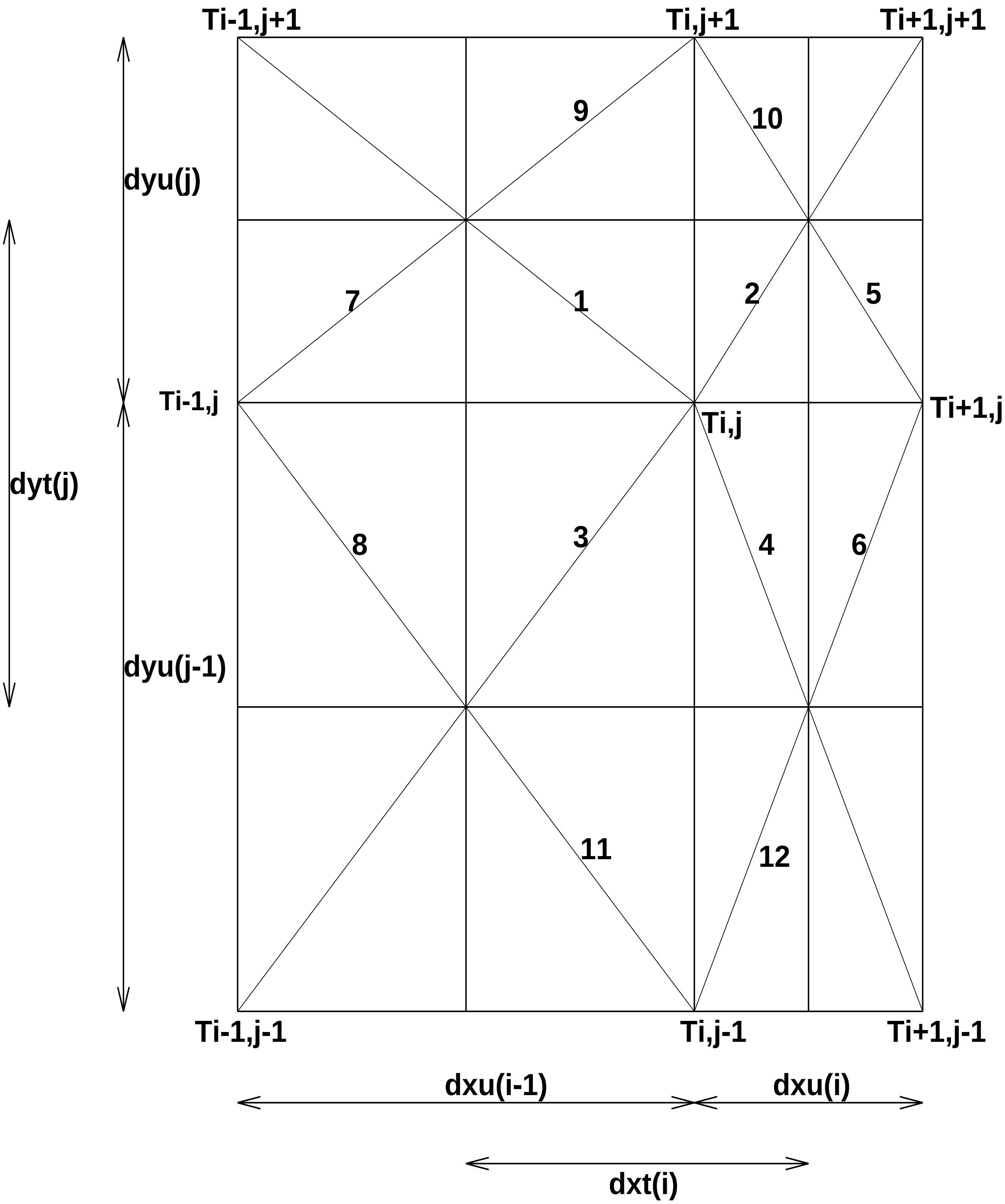


Figure B.2: MOM2 x-y plane and its partitioning into 12 quarter cells. The generally nonconstant grid spacing is indicated, which implies an offset T-point.

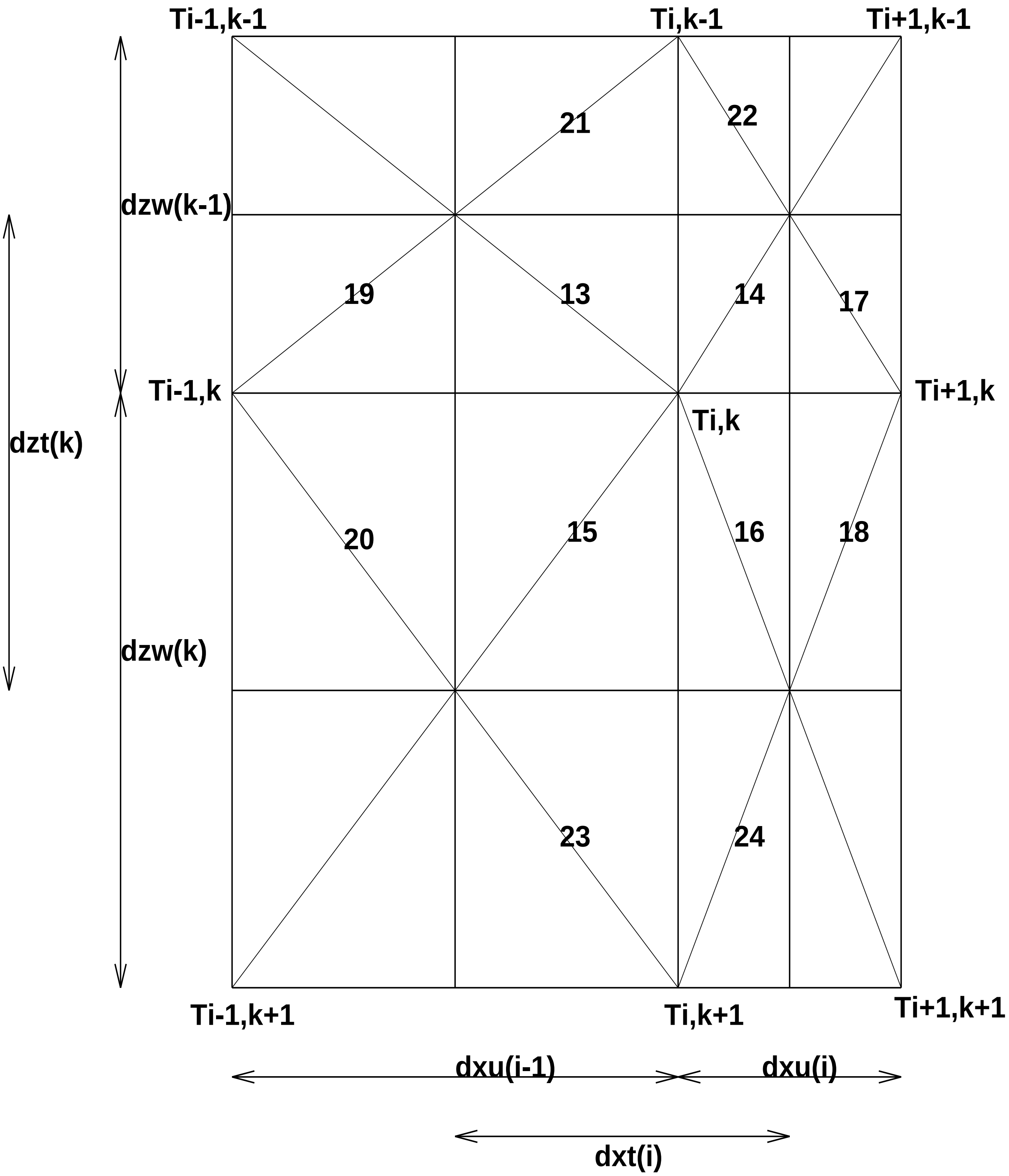


Figure B.3: MOM2 z-x plane and its partitioning into 12 quarter cells. The generally nonconstant grid spacing is indicated, which implies an offset T-point.

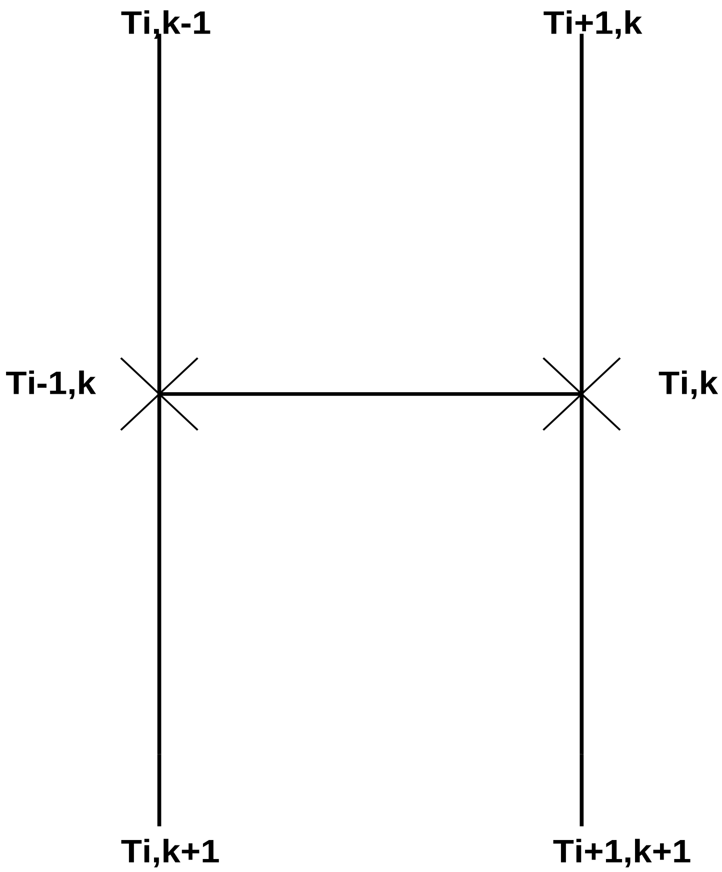


Figure B.4: Stencil in the x-z plane for the isopycnal x-flux. The x'ed points represent reference points used for computing the neutral directions.

Appendix C

A note about computational modes

Fundamental to the discretization of MOM is the discretization of fluxes: advective fluxes, diffusive fluxes, etc. Working with fluxes provides for a useful way to preserve the internal consistency of the transport of momentum and tracers, which means, for example, that we should have no false sources or sinks assuming we have a sound numerical scheme. Since we are fundamentally interested in the divergence of fluxes, the fluxes must be defined on the boundaries of the relevant grid cell: diffusive fluxes at the boundary of the T-cell and advective fluxes at the boundary of the U-cell. To achieve this placement of fluxes often requires some creative discretization in the form of averaging operations (see Section 11.2.1 for more details of finite difference operators). When introducing average operators, however, it is important to be aware of the potential to introduce computational modes. A computational mode is basically a configuration of the discretized field which is invisible to the object which is being discretized. With nearest neighbor discretization on the respective T and U grid, which is done in MOM for second order accurate expressions, computational modes take the form of "2-delta X" type waves; i.e., the smallest resolvable lattice wave. For higher order schemes, we are exposed to computational modes of longer wavelength. The presence of these waves often signal the ability for "grid noise" to manifest in the solution and so should be avoided.

As an example of the what is described above, we highlight one part of a recent study of isopycnal diffusion in the GFDL model. In Griffies et al., (1997) (see also Appendix B), it was found that one source of grid noise in MOM is the original Cox (1987) discretization of the isopycnal diffusive flux. For the small angle approximated diffusion tensor, the Cox (1987) discretization of the flux in a two-dimensional x-z model is

$$-F^x_{i,k} = A_I \left[\delta_x T_{i,k} - \left(\frac{\delta_x \rho_{i,k}}{\delta_z \overline{\rho}^{x,z}_{i,k-1}} \right) \delta_z \overline{T}^{x,z}_{i,k-1} \right], \tag{C.1}$$

$$-F^z_{i,k} = A_I \left(\frac{\delta_x \overline{\rho}^{x,z}_{i-1,k}}{\delta_z \rho_{i,k}} \right)^2 \left[\delta_z T_{i,k} - \left(\frac{\delta_z \rho_{i,k}}{\delta_x \overline{\rho}^{x,z}_{i-1,k}} \right) \delta_x \overline{T}^{x,z}_{i-1,k} \right]. \tag{C.2}$$

In order to define the diffusive fluxes consistently on the B-grid, the x-flux must be placed at the east face of the T-cell (i,k) and the z-flux at the bottom of this cell. With this placement, the divergence of these fluxes across the T-cell results in a diffusion operator $R(T)_{i,k} = -(\delta_x F^x_{i-1,k} + \delta_z F^z_{i,k-1})$ properly placed at the center of the cell at the location of the tracer $T_{i,k}$. For the off-diagonal terms (the second terms of the fluxes), a spatial averaging in the form of a double average $\overline{(\,)}^{x,z}$ brings the z-derivative terms appearing in the x-flux onto the east face of a T-cell and the x-derivative terms appearing in the z-flux onto the bottom face of the T-cell. This is a natural choice for averaging when working on the B-grid and provides an example of what we meant in the previous paragraph about "creative discretization".

There is a problem, however, with this discretization due to the presence of both the average and derivative operations acting in the same spatial direction on a single field. The problem is that this combination of operations introduces computational modes. For example, in the x-flux, the z-derivative of the tracer defined on the east face of T-cell (i,k) is

$$\delta_z \overline{T}^{x,z}_{i,k-1} = \frac{T_{i,k-1} - T_{i,k+1} + T_{i+1,k-1} - T_{i+1,k+1}}{4dzt_k}, \tag{C.3}$$

and likewise for the z-derivative of the density. A quick inspection of this forumla indicates that by taking both a z-average and a z-derivative allows for the presence of $2\Delta z$ computational modes $T_{i,k-1} = T_{i,k+1}$ and $\rho_{i,k-1} = \rho_{i,k+1}$. For fields containing this structure, the discretized z-derivative on the east face will vanish. For the z-flux, $2\Delta x$ computational modes exist due to the combination of the x-average and x-derivative. In general, when working on the B-grid and acting on a single field, such combinations of an average in one direction combined with a derivative in the same direction introduces computational modes in this field. The presence of the grid waves, and the ability to increase their amplitude, were two of the fundamental reasons that the Cox (1987) diffusion scheme was unstable and so required background horizontal diffusion. Consult Griffies et al., (1997) for complete details.

Section C contributed by
Stephen M. Griffies
smg@gfdl.gov
Last revised October 1996

Appendix D

References

This Appendix provides some references which might be of use to users MOM 2. Contained here are references which discuss issues ranging from numerical technicalities to coupled atmosphere-ocean climate modeling. This listing is not complete and the reader is encouraged to point out omissions or make suggestions for extending the list in order to make it more complete. The references are categorized for easier referencing, with some references crossing categories.

D.0.7 References used in the manual

Adcroft, A. ,C. Hill, and J. Marshall, 1996: Representation of Topography by Shaved Cells in a Height Coordinate Ocean Model. *Submitted to Monthly Weather Review*

Aris, R. , 1962: **Vectors, Tensors and the basic equations of Fluid Mechanics**, Dover publishing.

Boris, J. P. , D. L. Book, 1973: Flux-corrected transport, I. SHASTA: A fluid transport algorithm that works. *Journal of Computational Physics*, **11**, 38-69.

Brown, J. A. ,K. A. Campana, 1978: An economical time-differencing system for numerical weather prediction. *Monthly Weather Review*, **106**, 1125–1136.

Bryan, F. 1986: Maintenance and variability of the thermohaline circulation. Geophysical Fluid Dynamics Program Thesis, Princeton University.

Bryan, K., 1969: A numerical method for the study of the circulation of the world ocean. *Journal of Computational Physics*, **4**, 347–376.

Bryan, K. and M. D. Cox, 1972: An approximate equation of state for numerical models of the ocean circulation. *Journal of Physical Oceanography*, **2**, 510–514.

Bryan, K., S. Manabe, and R.C. Pacanowski, 1975: A global ocean-atmosphere climate model. Part II. The oceanic circulation. *Journal of Physical Oceanography*, **5**, 30–46.

Bryan, K., 1984: Accelerating the convergence to equilibrium of ocean-climate models. *Journal of Physical Oceanography*, **14**, 666–673.

Bryan, K., 1989: The Design of Numerical Models of the Ocean Circulation. *Oceanic Circulation Models: Combining Data and Dynamics, D.L.T. Anderson and J. Willebrand (eds.)*, Kluwer Academic Publishers, 465–500.

Bryan, K., 1991: Michael Cox (1941–1989): His pioneering contributions to ocean circulation modeling. *Journal of Physical Oceanography*, **21**, 1259–1270.

Courant and Hilbert, *Methods of Mathematical Physics*, 2 volumes. Wiley-Interscience.

Cox, M. D., 1984: A primitive equation, 3-dimensional model of the ocean. GFDL Ocean Group Technical Report No. 1.

Cox, M. D., 1987: Isopycnal diffusion in a z-coordinate ocean model. *Ocean modeling*, **74**, 1–5.

Deardorff, J. W., 1973: The use of subgrid scale transport equations in a three-dimensional model of atmospheric turbulence. *Journal of Fluid Eng.*, **Sep.**, 429-438.

Döscher, R. and Redler, R.: 1995, The relative influence of North Atlantic overflow and subpolar deep convection on the thermohaline circulation in an OGCM, submitted to *J. Phys. Oceanogr.*

DYNAMO groups at IMG Grenoble, JRC Southampton and IfM Kiel: 1994, 1994 scientific report, *EC MAST DYNAMO contract no. MAS2-CT93-0060*, .

Dukowicz, J. K. and R. D. Smith, 1994: Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *Journal of Geophysical Research*, **99**, 7991–8014.

Farrow, D. E. , D. P. Stevens, 1995: A new tracer advection scheme for Bryan and Cox type ocean general circulation models. *Journal of Physical Oceanography*, **25**, 1731-1741.

Fiadeiro M. E. , G. Veronis, 1977: On weighted-mean schemes for the finite-difference approximation to the advection-diffusion equation. *Tellus*, **29**, 512-522.

Gent, P. R., and J. C. McWilliams, 1990: Isopycnal mixing in ocean circulation models. *Journal of Physical Oceanography*, **20**, 150–155.

Gent, P. R., J. Willebrand, T. McDougall, and J. C. McWilliams, 1995: Parameterizing eddy-induced tracer transports in ocean circulation models. *Journal of Physical Oceanography*, **25**, 463–474.

Gerdes, R., C. Köberle, and J. Willebrand, 1991: The influence of numerical advection schemes on the results of ocean general circulation models. *Climate Dynamics*, **5**, 211–226.

Gill, A. E., 1982: **Atmosphere-Ocean Dynamics**. *Academic Press Inc.*

Goddard, L. 1995: The energetics of interannual variability in the tropical Pacific Ocean. Program in Atmospheric and Oceanic Sciences Thesis, Princeton University.

Goloviznin, V. M., Samarskii, A. A., and A. P. Favorskii, 1977: A variational approach to constructing finite-difference mathematical models in hydrodynamics. *Sov. Phys. Dokl.*, **22** 432–434.

Griffies, S. M., A. Gnanadesikan, R. C. Pacanowski, and V. Larichev, 1996: Towards an adiabatic realization of isopycnal diffusion in a z-Coordinate ocean model. *in preparation*

Haltiner, G. J. and R. T. Williams, 1980: **Numerical Prediction and Dynamic Meteorology**, Wiley.

Hamming, R. W., 1977: **Digital Filters**, Prentice-Hall.

Hankin, S. and M. Denham: 1994, **FERRET: An Analysis Tool for Gridded Data, Users Guide, Version 3.1**, NOAA/PMEL/TMAP.

Hanson, R. J. , and C. L. Lawson, 1969: Extensions and applications of the Householder algorithm for solving linear least squares problems. *Math. Comput.*, **23** 787-812.

Held, I. M. and V. D. Larichev, 1996: A scaling theory for horizontally homogeneous baroclinically unstable flow on a beta plane, Submitted to *Journal of Atmospheric Sciences*, **53**, 946–952.

Large, W. G. , J. C. McWilliams, and S. C. Doney, 1994: Oceanic vertical mixing: A review and a model with a nonlocal boundary layer parameterization. *Rev. of Geophys*, **32**, 363-403.

Larichev, V. D. and I. M. Held, 1995: Eddy amplitudes and fluxes in a homogeneous model of fully developed baroclinic instability. *Journal of Physical Oceanography*, **25**, 2285–2297.

Hellerman, S. and M. Rosenstein, 1983: Normal Monthly Stress over the World Ocean with Error Estimates. *Journal of Physical Oceanography*, **13**, 1093–1104.

Holland, W. R., 1975: Energetics of baroclinic oceans. In **Numerical Models of Ocean Circulation**, National Academy of Sciences, Washington, D.C.

Holloway, Greg, 1992: Representing Topographic Stress for Large-Scale Ocean Models. *Journal of Physical Oceanography*, **22**, 1033–1046.

Killworth, P. D., J. M. Smith, A. E. Gill, 1984: Speeding up ocean ciruclation models. *Ocean Modeling*, UNPUBLISHED MANUSCRIPT **56**, 1–5.

Leonard, B. P., 1979: A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer Methods in Applied Mechanics and Engineering*, **19**, 59–98

Levitus, S., 1982: Climatological atlas of the world ocean. *NOAA Prof. Pap. 13, 173 pp.* U. S. Government Printing Office, Washington, D. C.

Larichev, V. D. and I. M. Held, 1995: Eddy amplitudes and fluxes in a homogeneous model of fully developed baroclinic instability. *Journal of Physical Oceanography* in press.

McDougall, T. J., 1987: Neutral surfaces. *Journal of Physical Oceanography*, **17**, 1950–1964.

Middleton, J.F. and J.W. Loder, 1989: Skew fluxes in polarized wave fields. *Journal of Physical Oceanography*, **19**, 68–76.

NCAR, 1996: The NCAR CSM Ocean Model. *NCAR Technical Note NCAR/TN 423+STR* Climate and Clobal Dynamics Division, National Center for Atmospheric Research, Boulder, Colorado

Oort, A., 1983: Global atmospheric circulation statistics 1958-1973. *NOAA Prof. Pap. 14, 180 pp.* U. S. Government Printing Office, Washington, D. C.

Pacanowski, R. C., and G. Philander, 1981: Parametrization of vertical mixing in numerical models of the tropical ocean. *Journal of Physical Oceanography*, **11**, 1442–1451.

Pacanowski, R. C., 1987: Effect of Equatorial Currents on Surface Stress, *Journal of Physical Oceanography*, Vol 17, No. 6.

Pacanowski, R. C., K. Dixon, and A. Rosati, 1991: The GFDL modular ocean model user guide, The GFDL Ocean Group Technical Report No. 2., Geophysical Fluid Dynamics Laboratory, Princeton, USA.

Pierrehumbert, R. T. and H. Yang, 1993: Global chaotic mixing on isentropic surfaces. *Journal of Atmospheric Science*, **50**, No. 15, 2462-2480.

Philander, S. G. H. and R. C. Pacanowski, 1986: A model of the Seasonal Cycle in the Tropical Atlantic Ocean. *Journal of Geophysical Research*, **91**, 14192-14206.

Plumb, R. A. and J. D. Mahlman, 1987: The zonally averaged transport characteristics of the GFDL general circulation/transport model. *Journal of the Atmospheric Sciences*, **44**, 298–327.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, 1992: Numerical Recipes in Fortran, Second Edition, **Cambridge University Press**

Redi, M. H., 1982: Oceanic isopycnal mixing by coordinate rotation. *Journal of Physical Oceanography*, **12**, 1154–1158.

Redler, R. and Claus W. Böning, 1996: Effect of the overflows on the circulation of the North Atlantic: A regional model study, submitted to J. *Geophys. Res., Oceans*

Rhines, P. B., 1986: Lectures on ocean circulation dynamics. In **Large-scale transport processes in oceans and atmospheres**, edited by J. Willebrand and D. L. T. Anderson. D. Reidel Publishing.

Rood, R. B. , 1987: Numerical advection algorithms and their role in atmospheric transport and chemistry models. *Rev. Geophys*, **25**, 71-100.

Rosati, A. and K. Miyakoda, 1988: A general circulation model for upper ocean simulation. *Journal of Physical Oceanography*, **18**, 1601–1626.

Semtner, Jr., A. J., 1974: An oceanic general circulation model with bottom topography. In *Numerical Simulation of Weather and Climate*, Technical Report No. 9, UCLA Department of Meteorology.

Smagorinsky, J. 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, **91**, 99–164.

Smith, R. D., J. K. Dukowicz, and R. C. Malone, 1992: Parallel ocean general circulation modeling. *Physica D*, **60**, 38–61.

Stevens, D. P. 1991: The open boundary condition in the United Kingdom Fine– Resolution Antarctic Model, *J. Phys. Oceanogr.* **21**, 1494–1499.

Stevens, D. P. 1990: On open boundary conditions for three dimensional primitive equation ocean circulation models, *Geophys. Astrophys. Fluid Dynamics* **51**, 103–133.

Takacs, L. L., and R. Balgovind, 1983: High latitude filtering in global grid point models. *Monthly Weather Review*, **111**, 2005–2015.

Treguier, A. M., 1992: Kinetic energy analysis of an eddy resolving, primitive equation model of the North Atlantic. *Journal of Geophysical Research*, **97**, 687–701.

Treguier, A. M., J. K. Dukowicz, and K. Bryan: Properties of nonuniform grids used in ocean general circulation models. Submitted to *Journal of Geophysical Research* 1995.

Turner, J. 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, **91**, 99–164.

Turner, J. S., 1973: **Buoyancy effects in fluids**. *Cambridge University Press.*

Weaver, A. J. and E. S. Sarachik 1990: On the importance of vertical resolution in certain ocean general circulation models. *Journal of Physical Oceanography*, **20**, 600-609.

Wajsowicz, R. C. , 1993: A consistent formulation of the anisotropic stress tensor for use in models of the large-scale ocean circulation. *Journal of Computational Physics*, **105**, 333-338.

Webb, D. J., 1995: The Vertical Advection of Momentum in Bryan-Cox-Semtner OCean General Circulation Models. *Journal of Physical Oceanography*, **25**, 3186–3195.

Zalesak S. T. , 1979: Fully multidimensional flux-corrected transport algorithms for fluid. *Journal of Computational Physics*, **31**, 335-362.

D.0.8 Numerical

Arakawa, A., 1966: Computational design for long-term numerical integration of the equations of fluid flow: Two-dimensional incompressible flow. Part I. *Journal of Computational Physics*, **1**, 119–143.

Arakawa, A. and V. R. Lamb, 1977: Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics*, **17**, 174–265.

Bryan, K., 1969: A numerical method for the study of the circulation of the world ocean. *Journal of Computational Physics*, **4**, 347–376.

Boris, J. P. , D. L. Book, 1973: Flux-corrected transport, I. SHASTA: A fluid transport algorithm that works. *Journal of Computational Physics*, **11**, 38-69.

Brown, J. A. ,K. A. Campana, 1978: An economical time-differencing system for numerical weather prediction. *Monthly Weather Review*, **106**, 1125–1136.

Chervin, R. M. and A. J. Semtner, Jr., 1990: An ocean modeling system for supercomputer architectures of the 1990s. *Climate-Ocean Interactions*, M. E. Schlesinger, Ed., Kluwer Academic, 87–95.

Cox, M. D., 1984: A primitive equation, 3-dimensional model of the ocean. GFDL Ocean Group Technical Report No. 1.

Cox, M. D., 1987: Isopycnal diffusion in a z-coordinate ocean model. *Ocean modeling*, **74**, 1–5.

Dukowicz, J. K., R. D. Smith, and R. C. Malone, 1993: A reformulation and implementation of the Bryan-Cox-Semtner ocean model on the connection machine. *Journal of Atmospheric and Oceanic Technology*, **2**, 195–208.

Dukowicz, J. K. and R. D. Smith, 1994: Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *Journal of Geophysical Research*, **99**, 7991–8014.

Gerdes, R., C. Köberle, and J. Willebrand, 1991: The influence of numerical advection schemes on the results of ocean general circulation models. *Climate Dynamics*, **5**, 211–226.

Gerdes, R. 1993: A primitive equation ocean circulation model using a general vertical coordinate transformation 1. Description and testing of the model. *Journal of Geophysical Research*, **98**, 14683–14701.

Haltiner, G. J. and R. T. Williams, 1980: **Numerical Prediction and Dynamic Meteorology**, Wiley.

Killworth, P. D., J. M. Smith, A. E. Gill, 1984: Speeding up ocean ciruclation models. *Ocean Modeling*, UNPUBLISHED MANUSCRIPT **56**, 1–5.

Killworth, P. D., 1987: Topographic instabilities in level model OGCM's. *Ocean Modeling*, UNPUBLISHED MANUSCRIPT **75**, 9–12.

Killworth, P. D., D. Stainforth, D. J. Webb, and S. M. Paterson, 1991: The development of a free-surface Bryan-Cox-Semtner ocean model. *Journal of Physical Oceanography*, **21**, 1333–1348.

Korshiya, T. K., Tishkin, V. F., Favorskii, A. P., and M. Y. Shashkov, 1980: Flow-variational difference schemes for calculating the diffusion of a magnetic field. *Sov. Phys. Dokl.*, **25** 832–834.

Mesinger, F. and A. Arakawa, 1976: Numerical methods used in atmospheric problems, Vol. 1. WMO/ICSU Joint Organizing Committee, GARP Publications Series No. 17, 64 pages.

Pacanowski, R. C., K. Dixon, and A. Rosati, 1991: The GFDL modular ocean model user guide, The GFDL Ocean Group Technical Report No. 2., Geophysical Fluid Dynamics Laboratory, Princeton, USA.

Semtner, Jr., A. J., 1974: An oceanic general circulation model with bottom topography. In *Numerical Simulation of Weather and Climate*, Technical Report No. 9, UCLA Department of Meteorology.

Semtner, Jr., A. J., 1986a: Finite-difference formulation of a World Ocean model. In *Advanced Physical Oceanographic Numerical Modeling*, J. J. O'Brien, Ed., D. Reidel Publishing Company, 187–202.

Semtner, Jr., A. J., 1986a: History and methodology of modeling the circulation of the World Ocean. In *Advanced Physical Oceanographic Numerical Modeling*, J. J. O'Brien, Ed., D. Reidel Publishing Company, 23–32.

Smith, R. D., J. K. Dukowicz, and R. C. Malone, 1992: Parallel ocean general circulation modeling. *Physica D*, **60**, 38–61.

Send, U. and J. Marshall, 1995: Integral effects of deep convection. *Journal of Physical Oceanography*, **25,** 855–872.

Treguier, A. M., J. K. Dukowicz, and K. Bryan: Properties of nonuniform grids used in ocean general circulation models. Submitted to *Journal of Geophysical Research* 1995.

Webb, D. J., 1995: The Vertical Advection of Momentum in Bryan-Cox-Semtner OCean General Circulation Models. *Journal of Physical Oceanography*, **25**, 3186–3195.

Zalesak S. T. , 1979: Fully multidimensional flux-corrected transport algorithms for fluid. *Journal of Computational Physics*, **31**, 335-362.

D.0.9 General modeling issues

Bryan, K. and M. D. Cox, 1972: An approximate equation of state for numerical models of the ocean circulation. *Journal of Physical Oceanography*, **2**, 510–514.

Bryan, K., 1984: Accelerating the convergence to equilibrium of ocean-climate models. *Journal of Physical Oceanography*, **14**, 666–673.

Gerdes, R., C. Köberle, and J. Willebrand, 1991: The influence of numerical advection schemes on the results of ocean general circulation models. *Climate Dynamics*, **5**, 211–226.

Gerdes, R. 1993: A primitive equation ocean circulation model using a general vertical coordinate transformation 1. Description and testing of the model. *Journal of Geophysical Research*, **98**, 14683–14701.

Gerdes, R. 1993: A primitive equation ocean circulation model using a general vertical coordinate transformation 2. Application to an overflow problem. *Journal of Geophysical Research*, **98**, 14703–14726.

Hirst, A. C. and T. McDougall, 1996: Deep-water properties and surface buoyancy flux as simulated by a z-coordinate model including eddy-flux advection. *Journal of Physical Oceanography*, **26**, 1320–1343.

Haney, R. L., 1971: Surface thermal boundary condition for ocean circulation models. *Journal of Physical Oceanography*, **1**, 241–248.

Korshiya, T. K., Tishkin, V. F., Favorskii, A. P., and M. Y. Shashkov, 1980: Flow-variational difference schemes for calculating the diffusion of a magnetic field. *Sov. Phys. Dokl.*, **25** 832–834.

Power, S. B. 1995: Climate drift in a global ocean general circulation model. *Journal of Physical Oceanography*, **25**, 1025–1036.

Tishkin, V. F., Favorskii, A. P., and M. Y. Shashkov, 1979: Variational-difference schemes for the heat-conduction equation on nonregular grids. *Sov. Phys. Dokl.*, **24** 446–448.

Weaver, A. J. and E. S. Sarachik 1990: On the importance of vertical resolution in certain ocean general circulation models. *Journal of Physical Oceanography*, **20**, 600-609.

Zalesak S. T. , 1979: Fully multidimensional flux-corrected transport algorithms for fluid. *Journal of Computational Physics*, **31**, 335-362.

D.0.10 Sub-Grid Scale Parameterization

Bryan, F. O., 1987: Parameter sensitivity of primitive equation ocean general circulation models. *Journal of Physical Oceanography*, **17**, 970–985.

Cox, M. D., 1987: Isopycnal diffusion in a z-coordinate ocean model. *Ocean modeling*, **74**, 1–5.

Cummins, P. F., G. Holloway, and A. E. Gargett, 1990: Sensitivity of the GFDL ocean general circulation model to a parameterization of vertical diffusion. *Journal of Physical Oceanography*, **20**, 817–830.

Danabasoglu, G., J. C. McWilliams, and P. R. Gent, 1994: The role of mesoscale tracer transports in the global ocean circulation. *Science*, **264**, 1123–1126.

Gent, P. R., and J. C. McWilliams, 1990: Isopycnal mixing in ocean circulation models. *Journal of Physical Oceanography*, **20**, 150–155.

Gent, P. R., J. Willebrand, T. McDougall, and J. C. McWilliams, 1995: Parameterizing eddy-induced tracer transports in ocean circulation models. *Journal of Physical Oceanography*, **25**, 463–474.

Gough, W. A., and W. J. Welch, 1994: Parameter space exploration of an ocean general circulation model using isopycnal mixing parameterization. *Journal of Marine Research*, **52**, 773–796.

Gough, W. A., and C. A. Lin, 1995: Isopycnal mixing and the Veronis effect in an ocean general circulation model. *Journal of Marine Research*, **53**, 189–199.

Hirst, A. C. and T. McDougall, 1996: Deep-water properties and surface buoyancy flux as simulated by a z-coordinate model including eddy-flux advection. *Journal of Physical Oceanography*, **26**, 1320–1343.

Holland, W. R., 1989: Experiences with various parameterizations of sub-grid scale dissipation and diffusion in numerical models of ocean circulation. In *Parameterization of Small Scale Processes*, Proceedings of the Fourth 'Aha Huliko'a Hawaiian Winter Workshop, edited by P. Müller and G. Holloway, University of Hawaii at Manoa, 1–9.

Hirst, A. C., and W. Cai, 1994: Sensitivity of a World Ocean GCM to changes in subsurface mixing parameterization. *Journal of Physical Oceanography*, **24**, 1256–1279.

Jerlov 1968: **Optical oceanography**. *Elsevier.*

Larichev, V. D. and I. M. Held, 1995: Eddy amplitudes and fluxes in a homogeneous model of fully developed baroclinic instability. *Journal of Physical Oceanography* in press.

Marotzke, J., 1987: Influence of convective adjustment on the stability of the thermohaline circulation. *Journal of Physical Oceanography*, **21**, 903–907.

Marotzke, J. and J. Willebrand 1991: Multiple equilibria of the global thermohaline circulation. *Journal of Physical Oceanography*, **21**, 1372-1385.

McDougall, T.J., 1987: Neutral surfaces. *Journal of Physical Oceanography*, **17**, 1950–1964.

Pacanowski, R. C., and G. Philander, 1981: Parametrization of vertical mixing in numerical models of the tropical ocean. *Journal of Physical Oceanography*, **11**, 1442–1451.

Paulson and Simpson, 1977: Irradiance measurements in the upper ocean. *Journal of Physical Oceanography* **7**, 952-956.

Redi, M. H., 1982: Oceanic isopycnal mixing by coordinate rotation. *Journal of Physical Oceanography*, **12**, 1154–1158.

Rosati, A. and K. Miyakoda, 1988: A general circulation model for upper ocean simulation. *Journal of Physical Oceanography*, **18**, 1601–1626.

Smagorinsky, J. 1963: General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, **91**, 99–164.

D.0.11 General Numerical Oceanography: Eddy-Resolving

Beckmann, A., C. W. Böning, B. Brügge, and D. Stammer, 1994: On the generation and role of eddy variability in the central North Atlantic Ocean. *Journal of Geophysical Research*, **99**, 20,381–20,391.

Beckmann, A., C. W. Böning, C. Köeberle, and J. Willebrand, 1994: Effects of increased horizontal resolution in a simulation of the North Atlantic ocean. *Journal of Physical Oceanography*, **24**, 326–344.

Böning, C., 1988: Particle disperson and mixing of conservative properties in an eddy-resolving model. *Journal of Physical Oceanography*, **18**, 320–338.

Böning, C., 1989: Influence of a rough bottom topography on flow kinematics in an eddy-resolving circulation model. *Journal of Physical Oceanography*, **19**, 77–97.

Böning, C. and R. G. Budich, 1992: Eddy dynamics in a primitive equation model: sensitivity to horizontal resolution and friction. *Journal of Physical Oceanography*, **22**, 361–381.

Böning, C. W. and P. Herrmann, 1994: Annual cycle of poleward heat transport in the ocean: results from high-resolution modeling of the North and Equatorial Atlantic. *Journal of Physical Oceanography*, **24**, 91–107.

Bryan, F. O. and W. R. Holland, 1989: A high resolution simulation of the wind and thermohaline driven circulation in the North Atlantic Ocean. In *Parameterization of Small Scale Processes*, Proceedings of the Fourth 'Aha Huliko'a Hawaiian Winter Workshop, edited by P. Müller and G. Holloway, University of Hawaii at Manoa, 99–115.

Bryan, F. O., C. W. Böning, and W. R. Holland, 1995: On the midlatitude circulation in a high-resolution model of the North Atlantic. *Journal of Physical Oceanography*, **25**, 289–305.

Bryan, K. 1986: Poleward buoyancy transport in the ocean and mesoscale eddies. *Journal of Physical Oceanography*, **5**, 927–933.

Bryan, K. 1991: Poleward heat transport in the ocean. A review of a hierarchy of models of increasing resolution. *Tellus*, **43**, 104–115.

Cox, M. D., 1979: A numerical study of Somali Current eddies. *Journal of Physical Oceanography*, **9**, 311–326.

Cox, M. D., 1987: An eddy-resolving numerical model of the ventilated thermocline. *Journal of Physical Oceanography*, **15**, 1312–1324.

Cox, M. D., 1987: An eddy-resolving numerical model of the ventilated thermocline: time dependence. *Journal of Physical Oceanography*, **17**, 1044–1056.

Orlanski, I.: 1976, A simple boundary condition for unbounded hyperbolic flows, *J. Comp. Phys* **21**, 251–269.

Orlanski, I. and M. D. Cox, 1973: Baroclinic instability in ocean currents. *Geophysical Fluid Dynamics*, **4**, 297–332.

Semtner, Jr., A. J. and R. M. Chervin, 1988: A simulation of the Global Ocean circulation with resolved eddies. *Journal of Geophysical Research*, **93**, 15502–15522.

Treguier, A. M., 1992: Kinetic energy analysis of an eddy resolving, primitive equation model of the North Atlantic. *Journal of Geophysical Research*, **97**, 687–701.

D.0.12 General Numerical Oceanography: Non-eddy resolving

Bryan, K., 1962: A numerical investigation of a nonlinear model of a wind-driven ocean. *Journal of Atmospheric Sciences*, **20**, 594–606.

Cox, M. D., and K. Bryan, 1984: A numerical model of the ventilated thermocline. *Journal of Physical Oceanography*, **14**, 674–687.

Gerdes, R. 1993: A primitive equation ocean circulation model using a general vertical coordinate transformation 2. Application to an overflow problem. *Journal of Geophysical Research*, **98**, 14703–14726.

Holland, W. R., 1967: On the wind-driven circulation in an ocean with bottom topography. *Tellus*, **29**, 582–600.

Holland, W. R., 1973: Baroclinic and topographic influences on the transport in western boundary currents. *Geophysical Fluid Dynamics*, **4**, 187–210.

D.0.13 Tracers

Böning, C., 1988: Particle disperson and mixing of conservative properties in an eddy-resolving model. *Journal of Physical Oceanography*, **18**, 320–338.

Danabasoglu, G., J. C. McWilliams, and P. R. Gent, 1994: The role of mesoscale tracer transports in the global ocean circulation. *Science*, **264**, 1123–1126.

Duffy, P. B., P. Eltgroth, A. J. Bourgeois, and K. Caldeira, 1995: Effect of improved subgrid scale transport of tracers on uptake of bomb radiocarbon in the GFDL ocean general circulation model. *Geophysical Research Letters*, **22**, 1065–1068.

Gent, P. R., J. Willebrand, T. McDougall, and J. C. McWilliams, 1995: Parameterizing eddy-induced tracer transports in ocean circulation models. *Journal of Physical Oceanography*, **25**, 463–474.

Holland, W. R., 1971: ocean tracer distributions Part 1. A preliminary numerical experiment. *Tellus* **23**, 371–392.

Toggweiler, J. R., K. Dixon, and K. Bryan, 1989: Simulations of radiocarbon in a course-resolution world ocean model. 1. Steady state prebomb distribution. *Journal of Geophysical Research*, **94**, 8217–8242.

Toggweiler, J. R., K. Dixon, and K. Bryan, 1989: Simulations of radiocarbon in a course-resolution world ocean model. 2. Distributions of bomb-produced Carbon 14. *Journal of Geophysical Research*, **94**, 8243–8264.

Toggweiler, J. R. and B. Samuels, 1993: New radiocarbon constraints on the upwelling of abyssal water to the ocean's surface. In *The Global Carbon Cycle*, Edited by M. Heimann, NATO ASI Series Vol. I 15.

Veronis, G., 1975: The role of models in tracer studies. In **Numerical Models of Ocean Circulation**, National Academy of Sciences, Washington, D.C.

D.0.14 Atlantic and High-Latitude

Beckmann, A., C. W. Böning, B. Brügge, and D. Stammer, 1994: On the generation and role of eddy variability in the central North Atlantic Ocean. *Journal of Geophysical Research*, **99**, 20,381–20,391.

Beckmann, A., C. W. Böning, C. Köeberle, and J. Willebrand, 1994: Effects of increased horizontal resolution in a simulation of the North Atlantic ocean. *Journal of Physical Oceanography*, **24**, 326–344.

Böning, C. and R. G. Budich, 1992: Eddy dynamics in a primitive equation model: sensitivity to horizontal resolution and friction. *Journal of Physical Oceanography*, **22**, 361–381.

Böning, C., W. R. Holland, F. O. Bryan, G. Danabasoglu, and J. C. McWilliams, 1995: An overlooked problem in model simulations of the thermohaline circulation and heat transport in the Atlantic Ocean. *Journal of Climate*, **8**, 515–523.

Böning, C. W. and P. Herrmann, 1994: Annual cycle of poleward heat transport in the ocean: results from high-resolution modeling of the North and Equatorial Atlantic. *Journal of Physical Oceanography*, **24**, 91–107.

Bryan, F., 1986: High-latitude salinity effects and interhemispheric thermohaline circulations. *Nature*, **323**, 301–304.

Bryan, F. O., 1987: Parameter sensitivity of primitive equation ocean general circulation models. *Journal of Physical Oceanography*, **17**, 970–985.

Bryan, F. O. and W. R. Holland, 1989: A high resolution simulation of the wind and thermohaline driven circulation in the North Atlantic Ocean. In *Parameterization of Small Scale Processes*, Proceedings of the Fourth 'Aha Huliko'a Hawaiian Winter Workshop, edited by P. Müller and G. Holloway, University of Hawaii at Manoa, 99–115.

Bryan, F. O., C. W. Böning, and W. R. Holland, 1995: On the midlatitude circulation in a high-resolution model of the North Atlantic. *Journal of Physical Oceanography*, **25**, 289–305.

Döscher, R., C. Böning, and P. Herrmann, 1994: Response of circulation and heat transport in the North Atlantic to changes in thermohaline forcing in Northern Latitudes: A model study. *Journal of Physical Oceanography*, **24**, 2306–2320.

Hughes, T. M. C. and A. J. Weaver, 1994: Multiple equilibria of an asymmetric two-basin ocean model. *Journal of Physical Oceanography*, **24**, 619–637.

Moore, A. M. and C. J. C. Reason, 1993: The response of a global general circulation model to climatological surface boundary conditions for temperature and salinity. *Journal of Physical Oceanography*, **23**, 300–328.

Power, S. B. and R. Kleeman, 1993: Multiple equilibria in a global ocean general circulation model. *Journal of Physical Oceanography*, **23**, 1670–1681.

Power, S. B. A. M. Moore, D. A. Post, N. R. Smith, and R. Kleeman, 1994: Stability of North Atlantic Deep Water formation in a global ocean general circulation model. *Journal of Physical Oceanography*, **24**, 904–916.

Rahmstorf, S., 1994: Rapid climate transitions in a coupled ocean-atmosphere model. *Nature*, **372**, 82–85.

Sarmiento, J. L., and K. Bryan, 1982: An ocean transport model for the North Atlantic. *Journal of Geophysical Research*, **87**, 394–408.

Treguier, A. M., 1992: Kinetic energy analysis of an eddy resolving, primitive equation model of the North Atlantic. *Journal of Geophysical Research*, **97**, 687–701.

Tziperman, E., R. Toggweiler, Y. Feliks, and K. Bryan, 1994: Instability of the thermohaline circulation with respect to mixed boundary conditions: Is it really a problem for realistic models? *Journal of Physical Oceanography*, **24**, 217–232.

Weaver, A.J. and E.S. Sarachik, 1991: The role of mixed boundary conditions in numerical models of the ocean's climate. *Journal of Physical Oceanography*, **21**, 1470–1493.

Weaver, A.J., E.S. Sarachik, and J. Marotzke, 1991: Freshwater flux forcing of decadal and interdecadal oceanic variability. *Nature*, **353**, 836–838.

Weaver A. J., J. Marotzke, P. F. Cummins and E. S. Sarachik, 1993: Stability and variability of the thermohaline circulation. *Journal of Physical Oceanography*, **23**, 39-60.

D.0.15 Tropical

Hirst, A. C., and J. S. Godfrey, 1993: The role of Indonesian throughflow in a global ocean GCM. *Journal of Physical Oceanography*, **23**, 1057–1086.

Liu, Z., S. G. H. Philander, and R. C. Pacanowski, 1994: A GCM study of tropical-subtropical upper-ocean water exchange. *Journal of Physical Oceanography*, **24**, 2606–2623.

Pacanowski, R. C., and G. Philander, 1981: Parametrization of vertical mixing in numerical models of the tropical ocean. *Journal of Physical Oceanography*, **11**, 1442–1451.

Philander, S. G. H. and R. C. Pacanowski, 1980: The generation of equatorial currents. *Journal of Geophysical Research*, **85**, 1123–1136.

Philander, S. G. H. and R. C. Pacanowski, 1981: Response of equatorial oceans to periodic forcing. *Journal of Geophysical Research*, **86**, 1903–1916.

Philander, S. G. H. and R. C. Pacanowski, 1986: A model of the Seasonal Cycle in the Tropical Atlantic Ocean. *Journal of Geophysical Research*, **91**, 14192-14206.

Philander, S. G. H. and R. C. Pacanowski, 1986: Properties of Long Equatorial Waves in Models of the Seasonal Cycle in the Tropical Atlantic and Pacific Oceans. *Journal of Geophysical Research*, **91**, 14207-14211.

Philander, S. G. H. and R. C. Pacanowski, 1986: The Mass and Heat Budget in a Model of the Tropical Atlantic Ocean. *Journal of Geophysical Research*, **91**, 14212-14220.

Pacanowski, R. C., 1987: Effect of Equatorial Currents on Surface Stress, *Journal of Physical Oceanography*, Vol 17, No. 6.

Rosati, A. and K. Miyakoda, 1988: A general circulation model for upper ocean simulation. *Journal of Physical Oceanography*, **18**, 1601–1626.

D.0.16 Southern Ocean

Gill, A. E., and K. Bryan, 1971: Effects of geometry on the circulation of a three-dimensional Southern-Hemisphere ocean model. *Deep-Sea Research, 18*, 685–721.

England, M. H., 1992: On the formation of Antarctic Intermediate and Bottom Water in ocean general circulation models. J. Phys. Oceanogr., 22, 918-926.

England, M. H., J. S. Godfrey, A. C. Hirst and M. Tomczak, 1993: The mechanism for Antarctic Intermediate Water renewal in a World Ocean model. J. Phys. Oceanogr., 23, 1553-1560.

England, M. H., and V. C. Garcon, 1994: South Atlantic circulation in a World Ocean model, Annales Geophysicae, Special edition on the South Atlantic Ocean, 12, 812-825.

Toggweiler, J. R. and B. Samuels, 1993: Is the magnitude of the deep outflow from the Atlantic Ocean actually governed by the Southern Hemisphere winds? In *The Global Carbon Cycle*, Edited by M. Heimann, NATO ASI Series Vol. I 15.

Toggweiler, J. R. and B. Samuels, 1993: Effect of Drake Passage on the global thermohaline circulation. Preprint.

Toggweiler, J. R. and B. Samuels, 1993: Effect of sea ice on the salinity of Antarctic Bottom Waters. Preprint.

D.0.17 Global Ocean

Bryan, K. and M. Cox, 1967: A numerical investigation of the oceanic general circulation. *Tellus*, **19**, 54–80.

Bryan, K., 1969: Climate and the ocean circulation: The ocean model. *Monthly Weather Review*, **97**, 806–827.

Bryan, K. and M. D. Cox, 1972: The circulation of the world ocean: A numerical study. Part I, A homogeneous model. *Journal of Physical Oceanography*, **2**, 319–335.

Bryan, K., 1975: Three-dimensional numerical models of the ocean circulation. In **Numerical Models of Ocean Circulation**, National Academy of Sciences, Washington, D.C.

Bryan, K., 1979: Models of the world ocean. *Dynamics of Atmospheres and Oceans*, **3**, 327–338.

Bryan, K. and L. J. Lewis, 1979: A water mass model of the world ocean. *Journal of Geophysical Research, 84*, 2503–2517.

Bryan, K. and J. L. Sarmiento, 1985: Modeling ocean circulation. *Advances in Geophysics*, **28a**, 433–459.

Cox, M. D., 1975: A baroclinic numerical model of the world ocean: preliminary results. In **Numerical Models of Ocean Circulation**, National Academy of Sciences, Washington, D.C.

Cox, M. D., 1984: A primitive equation, 3-dimensional model of the ocean. GFDL Ocean Group Technical Report No. 1.

Cox, M. D., 1989: An idealized model of the World Ocean. Part I: The global-scale water masses. *Journal of Physical Oceanography*, **19**, 1730–1752.

Danabasoglu, G. and J. C. McWilliams, 1995: Sensitivity of the global ocean circulation to parameterizations of mesoscale tracer transports. Journal of Climate, **8**, 2967–2987.

England, M. H., 1993: Representing the global-scale water masses in ocean general circulation models. *Journal of Physical Oceanography*, **23**, 1523–1552.

England, M. H., V. C. Garcon, and J. F. Minster, 1994: Chlorofluorocarbon uptake in a World Ocean model, 1. Sensitivity to the surface gas forcing. J. Geophys. Res., 99, 25215-25233.

England, M. H., 1995: Using chlorofluorocarbons to assess ocean climate models, Geophys. Res. Letters (in press).

England, M. H., 1995: The age of water and ventilation time-scales in a global ocean model. J. Phys. Oceanogr., (in press).

Holland, W. R., 1979: The general circulation of the ocean and its modeling. *Dynamics of Atmospheres and Oceans*, **3**, 111–142.

McWilliams, J. C., 1996: Modeling the oceanic general circulation. Annual Review of Fluid Mechanics, **28**, 215–248.

Pond, S. and K. Bryan, 1976: Numerical models of the ocean circulation. *Reviews of Geophysics and Space Physics*, **14**, 243–263.

Semtner, Jr., A. J. and R. M. Chervin, 1988: A simulation of the Global Ocean circulation with resolved eddies. *Journal of Geophysical Research*, **93**, 15502–15522.

D.0.18 Coupled Atmosphere-Ocean

Bryan, K., F. G. Komro, S. Manabe, and M. C. Spelman, 1982: Transient climate response to increasing atmospheric Carbon Dioxide. *Science*, **215**, 56–58.

Bryan, K., S. Manabe, and R.C. Pacanowski, 1975: A global ocean-atmosphere climate model. Part II. The oceanic circulation. *Journal of Physical Oceanography*, **5**, 30–46.

Manabe, S., K. Bryan, and M. J. Spelman, 1979: A global ocean-atmosphere climate model with seasonal variation for future studies of climate sensitivity. *Dynamics of Atmospheres and Oceans*, **3**, 393–426.

Manabe, S., and R. J. Stouffer, 1988: Two stable equilibria of a coupled ocean-atmosphere model. *Journal of Climate*, **1**, 841–866.

Manabe, S., K. Bryan and M. J. Spelman, 1990: Transient response of a global coupled ocean-atmosphere model to doubling of atmospheric CO_2. *Journal of Physical Oceanography*, **20**, 722–749.

Manabe, S., R. J. Stouffer, M. J. Spelman, and K. Bryan, and M. J. Spelman, 1991: Transient responses of a coupled ocean-atmosphere model to gradual changes of atmospheric CO_2. Part I: annual mean response. *Journal of Climate*, **4**, 785–818.

Manabe, S., M. J. Spelman, and R. J. Stouffer 1992: Transient responses of a coupled ocean-atmosphere model to gradual changes of atmospheric CO_2. Part I. Seasonal response. *Journal of Climate*, **5**, 105–126.

Manabe, S. and R. J. Stouffer, 1994: Multiple-century response of a couple ocean-atmosphere model to an increase of atmospheric carbon dioxide. *Journal of Climate*, **7**, 5–23.

Philander, G. P., T. Yamagata, and R. C. Pacanowski,1984: Unstable Air-Sea Interactions in the Tropics. *Journal of Atmospheric Sciences*, **41**, 604–613.

Philander, G. P., R. C. Pacanowski, N. C. Lau, and M. J. Nath, 1992: Simulation of ENSO with a Global Atmosphere GCM Coupled to a High-Resolution Tropical Pacific Ocean GCM. *Journal of Climate*, **5**, 308–329.

Philander, G. P., N. C. Lau, R. C. Pacanowski, and M. J. Nath, 1989: Two different simulations of the Southern Oscillation and El Nino with coupled ocean-atmosphere general circulation models. *Phil Trans. R. Soc. Lond.*, **329**, 167–178.

Tziperman, E. and K. Bryan, 1993: Estimating global air-sea fluxes from surface properties and from climatological flux data using an oceanic general circulation model. *Journal of Geophysical Research*, **98**, 22629–22644.