H. Numerical and Coupling Improvements

a. Third-Order Upwind Algorithm

Third-order upwind schemes and variants of this scheme are discussed fully in Leonard (1979). A somewhat more complicated scheme has been put into the MOM code by the FRAM group, see Farrow and Stevens (1995). The third-order upwind scheme is switched on by setting the ifdef option *upwind3*.

The finite-difference expression for the advection of tracers is given by

$$\begin{aligned} ADV(i,j,k) = &- \frac{1}{dxt_i\ cst_j}(\overline{u}_E\ T_E^* - \overline{u}_W\ T_W^*) - \frac{1}{dyt_j}(\overline{v}_N\ T_N^* - \overline{v}_S\ T_S^*) \\ &- \frac{1}{dzt_k}(w_{k-1}\ T_T^* - w_k\ T_B^*) \end{aligned}$$

where $\overline{u}_W, \overline{u}_E, \overline{v}_S, \overline{v}_N$ are the T-grid cell face centered velocity components.

$$\begin{aligned} \overline{u}_E(i) &= \frac{u_{i,j} dyu_j + u_{i,j-1} dyu_{j-1}}{2dyt_j} \qquad\qquad \overline{u}_W(i) = \overline{u}_E(i-1) \\ \overline{v}_N(j) &= \frac{(v_{i,j} dxu_i + v_{i-1,j} dxu_{i-1}) csu_j}{2dxt_i cst_j} \\ \overline{v}_S(j) &= \frac{(v_{i,j-1} dxu_i + v_{i-1,j-1} dxu_{i-1}) csu_{j-1}}{2dxt_i cst_j} \end{aligned}$$

and the T^* quantities are the cell face centered tracer concentration at the east, west, north, south, top, and bottom faces. In the standard second order scheme of MOM, these are taken as the arithmetic average of the tracer concentrations in adjacent cells, e.g.,

$$T_E^* = \frac{1}{2}(T_{i+1,j} + T_{i,j})$$

In the third-order upwind formulation, the T^*s are replaced by an upwind direction biased three-point interpolation to the interface. In the zonal direction they are:

$$T_E^*(i) = \begin{cases} \gamma T_{i-1} + \beta^+ T_i + \alpha^+ T_{i+1} & \overline{u}_E > 0 \\ \beta^- T_i + \alpha^- T_{i+1} + \delta T_{i+2} & \overline{u}_E < 0 \end{cases}$$

$$T_W^*(i) = T_E^*(i-1)$$

where the coefficients $\alpha, \beta, \gamma, \delta$ are given by

$$\alpha^+ = \frac{dxt_i \ (2dxt_i + dxt_{i-1})}{(dxt_i + dxt_{i+1}) \ (dxt_{i+1} + dxt_{i-1} + 2dxt_i)}$$

$$\alpha^- = \frac{dxt_i(2dxt_{i+1} + dxt_{i+2})}{(dxt_i + dxt_{i+1}) \ (dxt_{i+1} + dxt_{i+2})}$$

$$\beta^+ = \frac{dxt_{i+1} \ (2dxt_i + dxt_{i-1})}{(dxt_i + dxt_{i+1}) \ (dxt_i + dxt_{i-1})}$$

$$\beta^- = \frac{dxt_{i+1}(2dxt_{i+1} + dxt_{i+2})}{(dxt_i + dxt_{i+1}) \ (dxt_i + dxt_{i+2} + 2dxt_{i+1})}$$

$$\gamma = \frac{-dxt_i * dxt_{i+1}}{(dxt_i + dxt_{i-1}) \ (dxt_{i+1} + dxt_{i-1} + 2dxt_i)}$$

$$\delta = \frac{-dxt_i * dxt_{i+1}}{(dxt_{i+1} + dxt_{i+2}) \ (dxt_i + dxt_{i+2} + 2dxt_{i+1})}$$

The interface tracer values $T_N^*, T_S^*, T_T^*,$ andT_B^* are obtained in an analogous manner.

Farrow, D.E., and D.P. Stevens, 1995: A new tracer advection scheme for Bryan and Cox type ocean general circulation models. *J. Phys. Oceanogr.,* **25**, 1731–1741.

Leonard, B.P., 1979: A stable and accurate convective modeling procedure based on quadratic upstream interpolation. *Comp. Meth. in Appl. Eng.*, **19**, 59–98.

b. Single Vertical Velocity

The second improvement concerns the vertical velocity computation. In the original scheme, two independent vertical velocity computations, one in the momentum and one in the tracer equations, are performed, resulting in two different vertical velocity fields, with the former typically much noisier than the latter. In the new method, the vertical velocity is computed only for the tracer equations. Consequently, this velocity is then area-averaged to obtain the vertical velocity for the momentum equations, eliminating the second computation. The related advection operator in the momentum equations is also modified accordingly to maintain the flux and kinetic energy conservation properties. This new treatment of the vertical velocity produces a unique and smooth vertical velocity field with rather modest effects on global distributions. Further details can be found in Webb (1995), where the new scheme is described in Section 7.

Webb, D.J., 1995: The vertical advection of momentum in Bryan-Cox-Semtner ocean general circulation models. *J. Phys. Oceanogr.,* **25**, 3186–3195.

c. North Pole Implementation

This section describes a consistent way to treat the north pole in B-grid models. The traditional finite difference approach breaks down because the zonal grid spacing goes to zero and there is no set of values to the north of the north pole. The approach described here treats the north pole as a circular grid point of radius dyt_{jmt} which carries values for all tracers and the transport streamfunction at its center. All grid points to the south of the pole are arranged in a standard B-grid, with internal velocity points offset one half grid spacing to the north and east from tracer and transport streamfunction points. There is no velocity point which corresponds to the tracer point at the pole. This option is activated by specifying the ifdef option *northpole*.

The momentum equations may be solved at the row just to the south of the north pole (jmt-1) using the standard solution procedure outlined in Section C. It is assumed here that a Laplacian frictional parameterization is used, some considerations may be necessary in order not to overflow arrays when using biharmonic or higher frictional parameterizations. The value of the internal mode velocities at j=jmt is not defined, but this poses no problem because the area of the interface between the jmt-1 and jmt velocity grid boxes is zero. All terms in the finite difference formulation which multiply $u_{i,jmt}$ or $v_{i,jmt}$ are also multiplied by cst_{jmt} (the cosine at the latitude of the the j=jmt T-point) which, for the north pole, is zero.

The solution of the tracer equation at the north pole requires some special treatment. The finite difference form of the tracer evolution equation at the circular grid may be written (neglecting friction) as

$$T^{n+1} = T^{n-1} + 2\Delta t\Big[\sum_{i=2}^{imt-1}(v_{i,jmt-1}dxu_i + v_{i-1,jmt-1}dxu_{i-1})$$

$$(T_{i,jmt} + T_{i,jmt-1})csu_{jmt-1}/(dyt_{jmt}dxt_i)\Big].$$

This is simply the sum of all meridional fluxes into the circular grid cell from the regular grid to the south. All terms on the right hand side are known. Once solved, the single new tracer value at the north pole can be filled into the j=jmt row of the tracer array so that the meridional fluxes through the interfaces with the interior grid can be calculated during the next time step.

The transport streamfunction is located at the same points as the tracers, so this will also require some special treatment at the pole. We take a similar approach by vertically averaging the internal mode zonal momentum equation and computing its line integral along the velocity points at row j=jmt-1. The contribution due to the surface pressure gradient identically vanishes, leaving

$$\int \overline{u}_t \mathrm{d}\lambda = \int \overline{\mathrm{u}'}_{\mathrm{t}} \mathrm{d}\lambda.$$

where $\overline{u'_t}$ is the vertical average of the internal mode tendency and λ is longitude. Using the definition of the transport streamfunction, this relation may be rewritten as

$$-\int \frac{1}{aH}\dot{\Psi}_\theta \mathrm{d}\lambda = \int \overline{\mathrm{u}'}_\mathrm{t} \mathrm{d}\lambda,$$

where a is the mean radius of the earth and H is the model depth at tracer points.

Hence, the barotropic circulation tendency around the pole determines the value of the streamfunction tendency ($\dot{\Psi}$) at the pole, relative to the average interior value. This equation may then be used to solve for the time rate of change of the transport streamfunction at the north pole ($\dot{\Psi}_{pole}$) as

$$\sum_{i=2}^{imt-1} \frac{2dxt_i}{H_{i,jmt-1} + H_{i-1,jmt-1}} \dot{\Psi}_{pole} =$$

$$\sum_{i=2}^{imt-1} \frac{2dxt_i}{H_{i,jmt-1} + H_{i-1,jmt-1}} \dot{\Psi}_{i,jmt-1} - \sum_{i=2}^{imt-1} dxt_i dyu_{jmt-1} \overline{u'_t}.$$

The forcing term on the right hand side is known and an estimate of the streamfunction tendency at j=jmt-1 is obtained during the standard iterative solution for the streamfunction in the interior. Once the estimate for the tendency at the pole is found, the usual iterative solution procedure is resumed. As for the tracer arrays, the single values for the streamfunction tendency and the streamfunction, once solved, must be stored in the appropriate array so the interior solution can proceed.

REMARK: The present implementation of the *northpole* ifdef option assumes equal grid intervals in the zonal direction when computing fluxes into the north pole gridbox.

d. Tapering of Horizontal and Isopycnal Mixing Coefficients

Our previous experience with global models showed that the diffusive computational stability at high latitudes, rather than advective stability, is usually the limiting factor for the model time step. Consequently, in the present model, we allow the horizontal eddy viscosity coefficients A_{MH}, A_I, and A_{ITD} to vary in the meridional direction. In particular, they are reduced at high latitudes. This treatment necessitates no changes in the scalar tracer equations. In contrast, the vector momentum equation requires additional horizontal friction terms so that pure solid body rotation does not produce any stress on the fluid. Consequently, we modified these viscous terms to follow Wajsowicz (1993). Because the model uses the leap-frog time integration scheme, the viscous/diffusive computational stability limits in the zonal direction in which the grid spacing becomes smaller at high latitudes are given by

$$A_{MH}\frac{\Delta t_U}{(\Delta x\ cos\phi)^2} \leq \frac{1}{4}, \qquad A_I\frac{\Delta t_T}{(\Delta x\ cos\phi)^2} \leq \frac{1}{4}.$$

In the present discussion, we consider only this direction for simplicity, because it is the most restrictive. In the above equations, Δt_U and Δt_T are the maximum momentum and tracer time steps, respectively, Δx is the longitudinal grid spacing, and ϕ is latitude. The model determines the allowable A_{MH}, A_I, and A_{ITD} from these stability equations, so that all these coefficients are kept at their full values as poleward as possible. In addition, to prevent excessively small A_{MH}, A_I, and A_{ITD} which can produce instabilities due to lack of enough diffusion, we restrict the minimum values to be 20% and 6% of their specified values in the 3×3 and 2×2 models, respectively. In the 3×3 model, A_{MH} tapering starts at 81.9°N, and it is kept constant at the 20% level starting at 87.3°N. A_I and A_{ITD} satisfy the above constraint at all latitudes. In the 2×2 model, A_{MH}, and A_I and A_{ITD} taperings start at 84.6°N and 85.2°N, respectively. They are kept constant at the 6% level north of 88.8°N. No tapering is applied in the Southern Hemisphere in both model configurations. This option is activated by setting the ifdef option *taperhrzvd*.

Wajsowicz, R. C., 1993: A consistent formulation of the anisotropic stress tensor for use in models of the large scale ocean circulation. *J. Comput. Phys.*, **105**, 333-338.

e. K. Bryan Acceleration Technique

The model time integration is performed in two stages. First, the acceleration technique of Bryan (1984) is applied in which the momentum and tracer equations have variable time steps, i. e.,

$$\alpha\left[\frac{\hat{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}^{n-1}}{2\Delta t}\right] = \mathbf{F}^{n-1,n},$$

$$\gamma(z)\left[\frac{[\theta, S]^{n+1} - [\theta, S]^{n-1}}{2\Delta t}\right] = G^{n-1,n}.$$

The temporal discretization is done using the leap-frog scheme, and $n-1$, n, and $n+1$ represent the previous, current, and next time levels, respectively. Also, $\hat{\mathbf{u}}$ is the horizontal baroclinic velocity vector, Δt is the fundamental (surface tracer) time step, θ is potential temperature, and S is salinity. $\mathbf{F}$ and G represent the rest of the terms in these equations, involving quantities from both the $n-1$ and n time levels. α and γ, which can vary in the vertical (z), are the time scale, or acceleration, factors. In the case of $\alpha > \gamma$, the momentum time step is smaller than the tracer time step by a factor of γ/α.

The model calender and all the forcing fields are based on the surface tracer time step ($\gamma^{-1}\Delta t$) by setting $\gamma = 1$ above 1058 m. The depth and timesteps given here and below apply to the 3x3 model version. With this choice, the annual cycles of heat and salt forcing are consistently applied. Here, $\Delta t = 440$ min was chosen, well within the advective computational stability limit. The smaller current speeds at depth are taken advantage of by setting $\gamma = 0.2$ between 1058 and 2502 m and $\gamma = 0.1$ between 2502 and 5000 m depth. Consequently, the deep ocean tracer time step is about 3 days; ten times larger than at the surface. To illustrate a disadvantage of this depth acceleration, consider a vertical heat flux, Q, between two adjacent model layers, k and $k+1$. Integration of the equation above leads to a change in the total heat content of the two layers by an amount

$$\Delta H \;=\; 2\Delta t\, Q\, (\gamma_k^{-1} \;-\; \gamma_{k+1}^{-1})\;.$$

Conservation of heat requires $\Delta H = 0$, which is violated whenever $\gamma_k \;\neq\; \gamma_{k+1}$. Therefore, vertical discontinuities in γ are restricted to depths where the fluxes are thought to be small enough that tracers are well conserved.

In the momentum equations, horizontal diffusion determines computational stability. At high latitudes this factor limits the baroclinic velocity time step to about 44 min, $\alpha = 10$, at all depths. To be consistent a time step of $\alpha^{-1}\Delta t$ is also used in the barotropic streamfunction equation. With this time step, the surface wind forcing is not consistently applied, because the annual cycle harmonic is applied 10 times per momentum year. As a result, some distortion of the seasonal cycle of ocean currents is expected. Analogous problems are not serious in the tracer fields, because their depth acceleration ($\gamma \neq 1$) is restricted to depths where the influence of the seasonal cycle on tracers is negligible. The acceleration technique is implemented by setting the values of *dtts, dtuv, dtsf,* and *dtxcel* in the model input data.

In the second stage of the time integration, the accelerated results were used as initial conditions for a synchronous time integration in which equal time steps and no depth acceleration were applied, i. e., Δt = 44 min and $\alpha = \gamma = 1$. This integration needs to be long enough to remedy any erroneous behavior spawned by violating conservation laws and by forcing mismatches in the acceleration stage, see Danabasoglu et al. (1996).

Bryan, K., 1984: Accelerating the convergence to equilibrium of ocean-climate models. *J. Phys. Oceanogr.,* **14**, 666–673.

Danabasoglu, G., J.C. McWilliams, and W.G. Large, 1996: Approach to equilibrium in accelerated global oceanic models. *J. Climate*, in press.

f. Ocean Surface Slope

The components of the ocean surface slope are computed exactly the same way as the diagnostic surface pressure gradient in the original MOM code. These slope components are accumulated at each timestep and only time-averaged fields over the *nysnc* steps in a coupling interval are passed to the flux coupler. The surface slope is nondimensionalized from the surface pressure gradient by dividing by the gravitational acceleration. This calculation is activated by setting the ifdef option *sflxpvmicetilt.*

g. Sea-Ice Formation and Melting

This improvement is needed when the ocean model is coupled to a sea-ice model, and is activated by the ifdef option *sflxiceflx.* At the end of each timestep, n, after surface fluxes and model transports have been applied, the upper *kmxice* model layers, where *kmxice* is a model input parameter, are scanned for temperatures below the freezing point,

T_f. At present T_f is set to the constant value of $-1.8°$C. However, the freezing point is a function of salinity and a more accurate equation for T_f, that will be implemented in the near future, is

$$\begin{aligned} T_f = &- .0575\ S_K + 0.001710523\ S_K^{3/2} \\ &- 0.0002154996\ S_K^2 \quad , \end{aligned}$$

at the local salinity, S_K. At each layer the potential mass per unit area of ice formation ($potice > 0$) or ice melt ($potice < 0$) is computed as

$$potice\ (k) = \mathrm{MAX}\left(\frac{(\varrho C_p)_o}{\Lambda_f}\ H_k\ (T_f - T_k) \quad , \quad qice\ (k-1) \right) \quad ,$$

where $(\varrho C_p)_o = 4.1 \times 10^6\ j/m^3/K$, is the product of ocean density and heat capacity, and $\Lambda_f = 334000\ j/Kg$ is the latent heat of fusion. Any ice that forms at depth is assumed to float towards the surface and this ice flux, $qice(k)$ (defined positive downwards), so $qice(k) \leq 0$ is accumulated bottom to top as

$$qice\ (k) = \sum_{kmxice}^{k} \ -potice\ (k) \quad ,$$

where $qice(kmxice-1) = 0$, assumes no ice formation below the $kmxice$ layer. As ice floats to the surface, it can either partially or completely melt in upper layers whose temperatures are above freezing.

At each layer the temperature and salinity are both adjusted in accord with the ice formed or melted in the layer:

$$\begin{aligned} T_k &= T_k + \frac{\Lambda_f}{H_k(\varrho C_p)_o} \quad potice(k) \\ S_k &= S_k + \frac{(S_o - S_i)}{\varrho_o\ H_k} \quad potice(k) \quad . \end{aligned}$$

The ice flux is accumulated at each location over the number of timesteps in a coupling interval, $nsync$, as

$$aqice(i,j) = \sum_{n=1}^{nsync} \ qice(k=1)_n .$$

If, at the end of a vertical scan, the upper layer temperature T_1 remains greater than freezing, ($qice(k = 1) = 0$), then the excess heat melts previously formed ice and $aqice(i,j)$, T_1 and S_1 are adjusted accordingly. Thus over the $nsync$ timesteps, ice is assumed to remain where it was formed. After $nsync$ timesteps, the accumulated ice flux $aqice(i,j)$ can be passed through the Flux Coupler to the sea-ice model as an equivalent downward heat flux (W/m^2).

$$qflux(i,j) = \frac{-\Lambda_f}{nsync\ \Delta t} \quad aqice(i,j)$$

where $aqice$ includes any adjustment due to melting previously formed ice in the last timestep, and Δt is the model timestep.

Thus, ocean temperature and salinity adjustments corresponding to the total ice formed ($qflux > 0$) are made prior to the ice being passed to the sea-ice model. However, warm surface model temperatures result in ($qflux < 0$), which is a potential to melt ice in the sea-ice model. Since the ocean does not know if sufficient ice is present at a given location, temperature and salinity adjustments are delayed until the appropriate heat and freshwater fluxes are received back from the sea-ice model through the Flux Coupler. Note, if sufficient ice is present in the absence of all other fluxes, the heat flux will be equal to the cooling needed to make $T_1 = T_f$, when applied over the next *nsync* timesteps.

h. Time Manager

The time-manager subroutine, *tmngr*, has evolved at NCAR over several years, and was eventually substantially rewritten in order to incorporate these changes in an orderly way. In a departure from the MOM 1.1 time manager, all time-dependent decisions in the NCAR CSM ocean model are based on an integer timestep counter, *itt*, and the corresponding elapsed model time in seconds, *ttsec*. Some MOM 1.1 variables which are based on the number of model days have integer timestep-number counterparts in the CSM ocean model, but others were eliminated during the restructuring. Many features of the MOM 1.1 time manager remain in the CSM version, although the underlying computations have changed. Choices of a simple or Julian calendar, allowing leap years or not, and an arbitrary year length are all available.

New features have been added. Because model I/O has been customized to the NCAR computing environment, new time-manager variables have been added to control the writing and storing of files. Restart and history files are independently controlled, and may be written at the end of each calendar month, or after a specified number of timesteps, or both. Other new time-manager variables include logical flags which signal the present timestep is the end of a day or year, and integer values of the present second, minute, and hour of the day and day, month, and year. All of these variables are available for use throughout the model upon the inclusion of the files *switch.h* and *ctmngr2.h*.

Controls for the model integration (starting and ending timesteps, the frequency of sampling for time-averaging, the frequency of writing diagnostics and history and restart files, and communication information for the Flux Coupler) are all set in the namelist *contrl*, which is located in the file *model.data* in the job script. Namelist *contrl* members include:

nb - (integer), the beginning timestep number. Set $nb = 0$ for initial run, $nb = itt$ for a continuation run, and $nb > 0$ for an autorestart, continuation run.

nlast - (integer), the last timestep number of the integration.

nrestrt - (integer), the number of timesteps between the creation of restart files (R files). Restart files are sent to the NCAR mass store immediately after creation.

ndgnstc - (integer), the number of timesteps between diagnostic calculations (global energetics and regional momentum and tracer term balances).

nfacdg - (integer), governs the frequency of printing the diagnostic computations in the subroutines *diag.F* and *diag2.F* (*i.e., nfacdg*ndgnstc* timesteps).

ntravg - (integer), the number of timesteps between the creation of regional tracer averages.

ntime_avg_freq - (integer), the number of timesteps between the accumulation of data used in time averaging.

ntime_avg_dump - (integer), the number of timesteps in the time-averaging interval. This parameter also controls the writing of the time-average history files, and should be an integer multiple of *ntime_avg_freq*. This is only valid when the logical switch *write_at_reg_int* is set true.

ntsi - (integer), the number of timesteps between the printing of on-line integral analyses and timestep information.

nprnstf - (integer), the number of timesteps between the printing of diagnostic streamfunction information (*mscan, resmax,* and island values).

nwrth1,2,3,4 - (integer), the number of time steps between the writing of various history files, if *write_at_reg_int* = .true.

ndmph1,2,3,4 - (integer), the number of time steps between the sending of various history files to the NCAR mass store. Selecting *ndmph* as a multiple of *nwrth* allows for multiple writes onto one history file.

write_at_reg_int - (logical), if true, causes history files to be written at regular intervals. Set *nwrth* and *ndmph* variables when using this option.

write_at_eom - (logical), if true, causes the writing of history files at the end of each month. Values of *nwrth* and *ldmph* are ignored, unless *write_at_reg_int* is also true.

The selection of the length of regular and leap years, the calendar type (Julian or equally-spaced months), and specification of the starting model time are all controlled in the namelist *tminfo*, which is also set in the file *model.data*. At the beginning of a model run, the starting time is specified as integer values of *day0:month0:year0* and *hour0:minut0:second0*. For example, a starting date of September 3, year 5 at 3:45:15pm would be initialized as *year0=5, month0=9, day0=3, hour0=15, minut0=45, sec0=15*. If the model is being restarted from restart files, these initial time values will be ignored, and the restart time will instead be read from the character variable *stamp*, which is read from the restart files. Variables in *tminfo* include:

yrnorm - (real), the length of a normal year, in days.

yrleap - (real), the length of a leap year, in days. Select *yrleap* = *yrnorm* if no leap year is desired.

julian - (logical), if true, selects the Julian calendar. If false, each month will contain *yrnorm*/12 days, and, even if specified, leap years will not be allowed.

sec0 - (integer) number of seconds; acceptable values: 0-59

minut0 - (integer) number of minutes; acceptable values: 0-59

hour0 - (integer) number of hours; acceptable values: 0-23

day0 - (integer) day number; acceptable values: 1-31

month0 - (integer) month number; acceptable values: 1-12

year0 - (integer) year number; acceptable values: 0,1, . . .