Part VI

Miscellaneous

Chapter 19

Basics of the finite difference method

This chapter describes the basics of finite difference methods for solving differential equations. The general principles of the finite differencing methods are introduced using the diffusion equation as an example in Section 19.1. Sections 19.2 and 19.3 describe applying finite difference methods of time and space derivatives in differential equations. Considerations in finite-difference methods for advection-diffusion equations are discussed in Section 19.4. An implicit method for solving the diffusion equation is described in Section 19.5. Section 19.6 summarizes discretization schemes used by MRI.COM.

Durran (1999) treats the basics of finite difference methods for solving differential equations of advection and diffusion in geophysical fluid dynamics.

19.1 Diffusion equation

As an example, consider an initial-boundary value problem expressed by a one-dimensional diffusion equation (heat conductive equation),

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}. \tag{19.1}$$

Given $T(x,0) = f(x)$ as the initial distribution and $T(0,t) = T(L,t) = 0$ as the boundary condition, the analytical solution is

$$T(x,t) = \sum_{m=0}^{\infty} f_m e^{-\kappa k_m^2 t} \sin(k_m x), \tag{19.2}$$

where

$$f_m = \frac{2}{L}\int_0^L f(x)\sin(k_m x)dx, \quad k_m = \frac{m\pi}{L}. \tag{19.3}$$

Next, consider the finite difference method to get the solution numerically. In the finite difference method, grids are set with a finite increment in space and time, and each term in the equation is evaluated at each grid using $T_j^n = T(x_j, t_n)$. For example,

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \kappa \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{\Delta x^2}, \tag{19.4}$$

where $\Delta t = t_{n+1} - t_n$ and $\Delta x = x_{j+1} - x_j$.

Distribution at the new time level T^{n+1} can then be calculated if T^n is known. This finite difference equation is identical to the original differential equation (19.1) in the limit $\Delta t \to 0, \Delta x \to 0$ (**consistency**).

If the initial distribution is assumed to be $f(x) = T_0 \sin k_1 x$, the solution of the finite difference equation (19.4) for $t = t_n$ is

$$T_j^n = \lambda^n T_0 \sin k_1 x_j, \tag{19.5}$$

where

$$\lambda = 1 - \frac{2\kappa\Delta t}{\Delta x^2}(1 - \cos k_1 \Delta x). \tag{19.6}$$

In order to suppress oscillation and divergence of the solution (**stability**), $0 < \lambda < 1$ is necessary and Δx and Δt must be set to satisfy this condition. This solution is identical to the analytical solution in the limit of $\Delta t \to 0, \Delta x \to 0$ (**convergence**).

To summarize, the finite difference method that satisfies consistency, stability, and convergence is the necessary condition for an accurate solution.

19.2 Finite difference expressions for time derivatives

The following four finite difference expressions are employed for the time derivatives in MRI.COM:

$$\text{forward}: \frac{T^{n+1}-T^n}{\Delta t} = F(T^n) \tag{19.7}$$

$$\text{backward}: \frac{T^{n+1}-T^n}{\Delta t} = F(T^{n+1}) \tag{19.8}$$

$$\text{Matsuno}: \frac{T^{*n+1}-T^n}{\Delta t} = F(T^n), \quad \frac{T^{n+1}-T^n}{\Delta t} = F(T^{*n+1}) \tag{19.9}$$

$$\text{leap-frog}: \frac{T^{n+1}-T^{n-1}}{2\Delta t} = F(T^n). \tag{19.10}$$

The scheme used in the previous section is the forward scheme. The forward, backward, and Matsuno schemes use the values at two time levels and are accurate to $O(\Delta t)$, while the leap-frog scheme uses three time levels and is accurate to $O(\Delta t^2)$. Basically, the leap-frog scheme is employed in MRI.COM because of its higher order accuracy.

However, the leap-frog scheme cannot be applied to the diffusion equation. A solution by the finite difference method using the leap-frog scheme is given by

$$T_j^n = (T_a \lambda_a^n + T_b \lambda_b^n) \sin k_1 x_j, \tag{19.11}$$

where

$$\lambda_a = -\frac{-\alpha + \sqrt{\alpha^2+4}}{2}, \quad \lambda_b = -\frac{-\alpha - \sqrt{\alpha^2+4}}{2} \quad \left(\alpha \equiv \frac{4\kappa\Delta t}{\Delta x^2}(1-\cos k_1 \Delta x)\right). \tag{19.12}$$

Because $\lambda_b < -1$ for arbitrary values of α, the divergent mode with oscillation is always included (**computational mode**). In order to avoid this computational mode, the forward scheme is employed for diffusion and viscosity terms in MRI.COM.

When the diffusion and viscosity coefficients are very large as in the surface mixed layer, the time step has to be unusually small for the stability of the forward scheme according to (19.6). In such a case, the backward scheme is used for vertical diffusion and viscosity (implicit method; see Section 19.5). Though the time integration at each point can proceed without referring to the result of other points by the forward, leap-frog, and Matsuno schemes, it must be done by solving combined linear equations in the backward scheme (see Section 19.5).

The Matsuno scheme is useful for suppressing the computational mode in the leap-frog scheme. By defaults, the Matsuno scheme is used once per twelve steps of the leap-frog scheme in MRI.COM. This interval can be changed at run time using a namelist parameter (`nstep_matsuno_interval`) of namelist `nml_time_step` (Table 21.6). It should be noted that the Matsuno scheme needs twice as many numerical operations as the forward and leap-frog schemes.

19.3 Finite difference expression for space derivatives

Let us consider a one-dimensional advection equation,

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x}, \tag{19.13}$$

where u is a constant velocity. The solution is

$$T(x,t) = T(x-ut, 0). \tag{19.14}$$

Using the leap-frog scheme for time differencing, the finite difference equation can be written as follows:

$$\frac{T_j^{n+1}-T_j^{n-1}}{2\Delta t} = -u\frac{T_{j+\frac{1}{2}}^n - T_{j-\frac{1}{2}}^n}{\Delta x}, \tag{19.15}$$

where $T_{j-\frac{1}{2}}^n$ and $T_{j+\frac{1}{2}}^n$ are the values at the left and right faces of the grid cell for x_j, i.e., values at $x_{j-\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$. The point $x_{j-\frac{1}{2}}$ is defined as the central point between x_j and x_{j-1}. Because the transport of T at the boundary that enters a grid cell is identical to that leaving the adjacent grid cell, the total T in the whole system is conserved in this finite difference equation.

Chapter 19 Basics of the finite difference method

There are several methods to decide $T^n_{j-\frac{1}{2}}$ using a value at a single or multiple grid points. The following are two simple and frequently used formulations,

$$\text{upstream finite difference}: T^n_{j-\frac{1}{2}} = T^n_{j-1}\ (u>0), \quad T^n_{j-\frac{1}{2}} = T^n_j\ (u<0), \tag{19.16}$$

$$\text{central finite difference}: T^n_{j-\frac{1}{2}} = \frac{T^n_{j-1}+T^n_j}{2}. \tag{19.17}$$

The former is accurate to $O(\Delta x)$, and the latter is accurate to $O(\Delta x^2)$.

In central finite differencing, the expression for (19.15) is

$$\frac{T^{n+1}_j - T^{n-1}_j}{2\Delta t} = -u\frac{T^n_{j+1} - T^n_{j-1}}{2\Delta x}. \tag{19.18}$$

Assuming the solution to be $T(x,t) = \tau(t)e^{-ikx}$,

$$\tau^{n+1} = \tau^{n-1} + 2i\alpha\tau^n, \quad \text{where } \alpha \equiv \frac{u\Delta t}{\Delta x}\sin k\Delta x. \tag{19.19}$$

It is stable (neutral) if $|\alpha| \le 1$. To be stable for any wave number,

$$\left|\frac{u\Delta t}{\Delta x}\right| \le 1 \tag{19.20}$$

must be satisfied (**CFL condition**). However, if $\tau^n = \tau^0 e^{-in\Delta\theta}$,

$$\Delta\theta = -\sin^{-1}[\mu\sin k\Delta x] (\text{where } \mu \equiv \frac{u\Delta t}{\Delta x}). \tag{19.21}$$

Expanding the r.h.s. by a Taylor expansion we obtain

$$\begin{aligned}\Delta\theta &\simeq -\mu\sin k\Delta x - \frac{1}{6}(\mu\sin k\Delta x)^3 \\ &\simeq -\mu k\Delta x + \frac{\mu(k\Delta x)^3}{6} - \frac{\mu^3(k\Delta x)^3}{6} \\ &= -\mu k\Delta x\left\{1 - \frac{(k\Delta x)^2}{6}(1-\mu^2)\right\}.\end{aligned} \tag{19.22}$$

This means that the phase of the solution from this finite difference scheme is delayed relative to that of analytical solution, depending on its wavenumber (**numerical dispersion**). Therefore, a distribution with maxima and minima that do not exist in the initial distribution arises. However, this method is popularly used since the kinetic energy is conserved by employing the central difference in the advection term in the equation of motion. Moreover, the "Arakawa method," which can nearly conserve the enstrophy (squared vorticity) for horizontally non-divergent flows, is adopted in MRI.COM by using the central difference. This topic is treated in Chapter 7.

Using the upstream finite difference, the finite difference equation (19.15) is

$$\frac{T^{n+1}_j - T^{n-1}_j}{2\Delta t} = -u\frac{T^n_j - T^n_{j-1}}{\Delta x}. \tag{19.23}$$

Expanding the r.h.s. by a Taylor expansion we obtain

$$-u\frac{\partial T}{\partial x} + \frac{u\Delta x}{2}\frac{\partial^2 T}{\partial x^2} + O(\Delta x^2). \tag{19.24}$$

The second term has the diffusion (heat conductive) form (which disappears in the central finite differencing). Actually, the initial distribution diffuses when the advection equation is solved by the upstream finite difference (**numerical diffusion**).

The third order schemes (QUICK, QUICKEST, and UTOPIA) can be used in MRI.COM to suppress the numerical dispersion and diffusion somewhat in the advection calculation for tracers, but not completely. The grid boundary value is set in QUICK as

$$T^n_{j-\frac{1}{2}} = \frac{-T^n_{j-2} + 6T^n_{j-1} + 3T^n_j}{8}(u>0), \quad T^n_{j-\frac{1}{2}} = \frac{3T^n_{j-1} + 6T^n_j - T^n_{j+1}}{8}(u<0). \tag{19.25}$$

The QUICKEST method uses the time averaged value at the grid boundary as the tracer value to be transported, considering the change of the value there by advection during one time step. UTOPIA is a multi-dimensional extension of QUICKEST. The details of these schemes are described in Chapter 8.

19.4 Finite differencing of advection-diffusion equation

According to the above restriction, when the advection-diffusion equation (Eqs. (8.1) and (9.1)) is expressed in finite difference form using the leap-frog scheme, it is necessary to use present (previous) time level for advection (diffusion) term. The following finite difference equation is then employed:

$$\frac{T^{n+1} - T^{n-1}}{2\Delta t} = -\mathcal{A}(T^n) + \mathcal{D}(T^{n-1}), \tag{19.26}$$

where $\mathcal{A}$ and $\mathcal{D}$ are advection and diffusion operators, respectively.

When the vertical diffusion term is very large, $\mathcal{D}(T^{n+1})$ is used instead of $\mathcal{D}(T^{n-1})$. This formula is an implicit scheme and is described in the next section.

19.5 Implicit method for vertical diffusion equation

Turbulent mixing is parameterized by using high vertical diffusivity and viscosity determined by boundary layer models, which was treated in Chapter 15. The time step must be set very small to keep the calculation stable when viscosity and diffusivity are very high, since the time tendency becomes very large due to the rapid mixing. To avoid this problem, the implicit method uses the advanced (mixed) state for evaluating viscosity and diffusivity, unlike the normal explicit method where previous or present values are used.

Expressing the present time step as n and the time step before and after as $n \pm 1$, the finite-difference method is applied to the advection-diffusion equation using the leap-frog scheme. The diffusion term is written separately using $(n-1)$ step for the horizontal direction and $(n+1)$ step for the vertical direction,

$$\frac{(T^{n+1} - T^{n-1})}{2\Delta t} = -\mathcal{A}(T^n) + \mathcal{D}_H(T^{n-1}) + \mathcal{D}_V(T^{n+1}). \tag{19.27}$$

Putting all the terms involving T^{n+1} on the l.h.s.,

$$T^{n+1} - 2\Delta t \mathcal{D}_V(T^{n+1}) = T^{n-1} + 2\Delta t(-\mathcal{A}(T^n) + \mathcal{D}_H(T^{n-1})) \tag{19.28}$$

is obtained, which is an algebraic equation for T^{n+1}. The finite difference form is rewritten specifically as

$$T_k^{n+1} - 2\Delta t \frac{1}{\Delta z_k}\left(\kappa_{k-\frac{1}{2}}(T_{k-1}^{n+1} - T_k^{n+1})/\Delta z_{k-\frac{1}{2}} - \kappa_{k+\frac{1}{2}}(T_k^{n+1} - T_{k+1}^{n+1})/\Delta z_{k+\frac{1}{2}}\right) \tag{19.29}$$
$$= T_k^{n-1} + 2\Delta t(-\mathcal{A}(T_k^n) + \mathcal{D}_H(T_k^{n-1})).$$

By putting

$$a = \frac{2\Delta t \kappa_{k-\frac{1}{2}}}{\Delta z_k \Delta z_{k-\frac{1}{2}}}, \quad b = 1 + a + c, \quad c = \frac{2\Delta t \kappa_{k+\frac{1}{2}}}{\Delta z_k \Delta z_{k+\frac{1}{2}}}, \tag{19.30}$$

we get,

$$-aT_{k-1}^{n+1} + bT_k^{n+1} - cT_{k+1}^{n+1} = T_k^{n-1} + 2\Delta t(-\mathcal{A}(T_k^n) + \mathcal{D}_H(T_k^{n-1})). \tag{19.31}$$

Setting $\mathcal{F}_k \equiv -\mathcal{A}(T_k^n) + \mathcal{D}_H(T_k^{n-1})$, this is expressed in the matrix form as

$$\begin{pmatrix} b & -c & & & & & \\ -a & b & -c & & & & \\ & -a & b & -c & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -a & b & -c & \\ & & & & -a & b & -c \\ & & & & & -a & b \end{pmatrix} \begin{pmatrix} T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \\ \vdots \\ T_{KM-2}^{n+1} \\ T_{KM-1}^{n+1} \\ T_{KM}^{n+1} \end{pmatrix} = \begin{pmatrix} T_1^{n-1} + 2\Delta t\mathcal{F}_1 \\ T_2^{n-1} + 2\Delta t\mathcal{F}_2 \\ T_3^{n-1} + 2\Delta t\mathcal{F}_3 \\ \vdots \\ T_{KM-2}^{n-1} + 2\Delta t\mathcal{F}_{KM-2} \\ T_{KM-1}^{n-1} + 2\Delta t\mathcal{F}_{KM-1} \\ T_{KM}^{n-1} + 2\Delta t\mathcal{F}_{KM} \end{pmatrix}. \tag{19.32}$$

The l.h.s. has the form of the tri-diagonal matrix.

19.5.1 A solution of tri-diagonal matrix

In general, simultaneous linear equations for n variables with tri-diagonal matrix coefficients

$$\begin{pmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & A_{n-1} & B_{n-1} & C_{n-1} \\ & & & A_n & B_n \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_{n-1} \\ X_n \end{pmatrix} = \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_{n-1} \\ D_n \end{pmatrix} \tag{19.33}$$

are solved using the Thomas method, which is modified from LU decomposition,

$$P_1 = C_1/B_1 \tag{19.34}$$
$$Q_1 = D_1/B_1 \tag{19.35}$$
$$P_k = \frac{C_k}{B_k - A_k P_{k-1}} \qquad (2 \le k \le n-1) \tag{19.36}$$
$$Q_k = \frac{D_k - A_k Q_{k-1}}{B_k - A_k P_{k-1}} \qquad (2 \le k \le n) \tag{19.37}$$
$$X_n = Q_n \tag{19.38}$$
$$X_k = Q_k - P_k X_{k+1} \qquad (1 \le k \le n-1). \tag{19.39}$$

19.6 Summary of schemes used by MRI.COM

MRI.COM employs centered difference schemes for spatial discretization except for the tracer advection terms.

Figure 19.1 shows the schematic of the relation between components of the governing equations in terms of time integration. The situation depicted here is the advancement in time from n to $n+1$, represented by the thick red arrows. The main components employ the leap-frog scheme, inserting Matsuno (Euler-backward) scheme with a specified interval. Sea ice and turbulence closure employs forward scheme.

Colored arrows represent data transfer among the components in advancing in time using the leap-frog time ($n \rightarrow n+1$). (a) State at the starting time ($n-1$) is used for computing surface fluxes. (b) Surface current at the starting time and SST and SSS of the present time is given to the sea ice component. (c) Flux from the sea ice component is averaged with that of the previous time step for use in the ocean component (d) for conservation. (e) Horizontal velocity field of the present time is used to compute vertical velocity and tracer transport. (f) Diagnosed vertical velocity is used immediately in the advection schemes of momentum and tracers. (g) Temperature and salinity at the present time is used to compute the pressure gradient term. (h) Vertically integrated forcing term computed by the baroclinic component is used by the barotropic component. (i) The barotropic transport is returned to the baroclinic component and used immediately as the vertically averaged velocity for computing the total velocity. For turbulent closure scheme, states at both present (j) and new (k) time level are used to estimate shear production and buoyancy loss of turbulent kinetic energy. (l) Turbulent kinetic energy at the present time is used to compute vertical mixing coefficient. Blue arrows represent the transfer of extant data obtained by the previous time steps. Green arrows represent the immediate use of the new data computed in this time step. These relations constrain the order of operations within a time step. The source side of the green arrows must be calculated earlier than the recipient side.

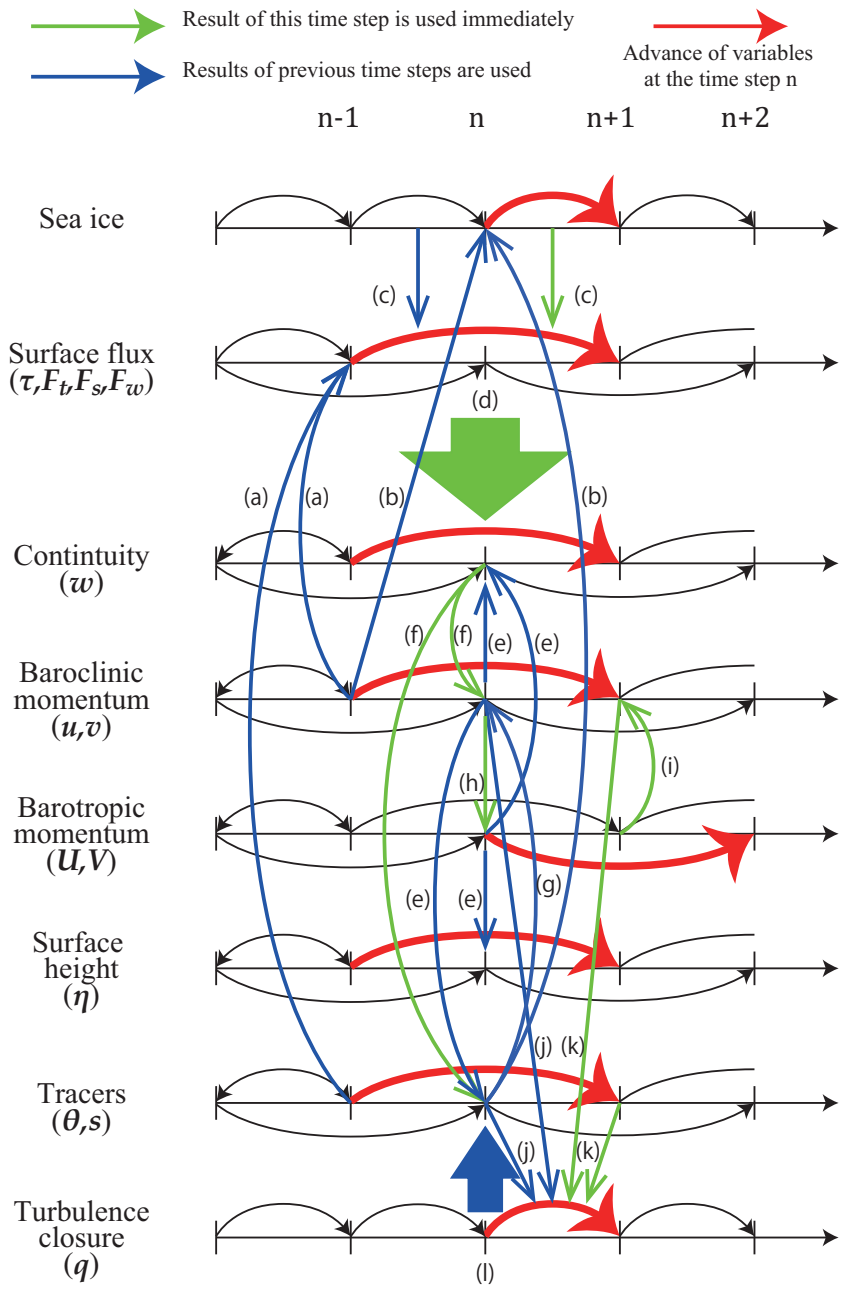

Figure19.1 Schematic of the relation between components of the governing equations in terms of time integration. The situation depicted here is the advancement in time from n to $n + 1$, which is represented by the thick red arrows. Blue arrows represent the transfer of extant data obtained by the previous time steps. Green arrows represent the immediate use of data computed at this time step. These relations constrain the order of operations within a time step. See text for the detailed explanation.

Chapter 20

Generalized orthogonal curvilinear coordinate grids

This chapter introduces generalized orthogonal curvilinear coordinates and presents related calculus.

20.1 Outline

A finite volume ocean model on geographic coordinates does not have any problem concerning the South Pole because it does not calculate around the South Pole. However, serious problems arise around the North Pole where the meridian concentrates to one point in the ocean. First, it is necessary to calculate the temporal evolution of the physical quantity in a special way only there, because the relations between U-cells that surround the North Pole and the northernmost T-cell are topologically peculiar. Next, even if a cell doesn't touch the North Pole, its zonal lattice interval is extremely small near the North Pole. Therefore, a short time step for integration is required owing to the limitation of the CFL condition. This limitation is reflected directly in the increased calculation time required. Moreover, when the zonal grid intervals in low latitudes and the Arctic region are extremely different, the arguments about accuracies of numerical schemes and the parameters for diffusion and viscosity operators generally cannot be applied uniformly to a model domain.

The following can be considered to avoid such problems concerning the North Pole. 1) Creating a huge island including the North Pole. The finite-difference calculation in the island is abandoned, and the lateral boundary values are restored to the climatology. 2) Shifting the singular points of the model to a continent or a huge island by changing the model's horizontal grid system. The MRI.COM scheme adopts the latter approach, which is outlined in this section.

Because the MRI.COM code is based on generalized orthogonal coordinates, the geographical latitude (ϕ) and longitude (λ) are not of a great concern for the calculus in the model. However, it is necessary to know the land and sea distribution, sea depth, scale factor, and the Coriolis parameter given as a function of λ and ϕ at every grid point of the model prior to the calculation. We describe the method of generating the orthogonal coordinate grid system in Section 20.2. Using a conformal transformation in the general sense, the functions that describe the relation between model coordinates (μ, ψ) and geographic coordinates (λ, ϕ), $\lambda(\mu, \psi)$, $\phi(\mu, \psi)$, $\mu(\lambda, \phi)$, and $\psi(\lambda, \phi)$, are obtained.

Because an atmospheric boundary condition is given in many cases at grid points in geographic coordinates, it is necessary to prepare tables for converting the surface atmospheric temperature, the wind stress, and so on. To convert a vector quantity, we must remember that the direction of the μ contour differs from that of the λ contour (meridian). The difference is described in Section 20.3. We can use the functions, $(\lambda, \phi) \Longleftrightarrow (\mu, \psi)$, to convert a scalar quantity as well as sea depth and the Coriolis parameter. The total flux that the ocean receives from the atmosphere should be equal to the total flux that the atmosphere gives to the ocean. The method for conserving the total flux is explored in Section 20.4. The vector operation in generalized orthogonal coordinates is concisely described in Section 20.5.

20.2 Generation of orthogonal coordinate system using conformal mapping

We designate the plane that touches the sphere at the North Pole as $\mathbf{S}_\mathrm{N}$. A polar stereographic projection is a conformal transformation in the general sense, so that an orthogonal coordinate system on the sphere is mapped onto an orthogonal coordinate system on $\mathbf{S}_\mathrm{N}$ and the orthogonality is preserved on the reverse transformation (Figure 20.1). Moreover, if $\mathbf{S}_\mathrm{N}$ is assumed to be a complex plane, various conformal transformations can be defined on it. Therefore, applying (i) the polar stereographic projection, (ii) a conformal transformation on the complex plane $\mathbf{S}_\mathrm{N}$, and (iii) the reverse polar stereographic projection to a geographic coordinate grid point (λ, ϕ) on the sphere, an orthogonal coordinate grid point on the sphere can be obtained (Bentzen et al., 1999).

The functions $\mu(\lambda, \phi)$ and $\psi(\lambda, \phi)$ are obtained by the following procedure:

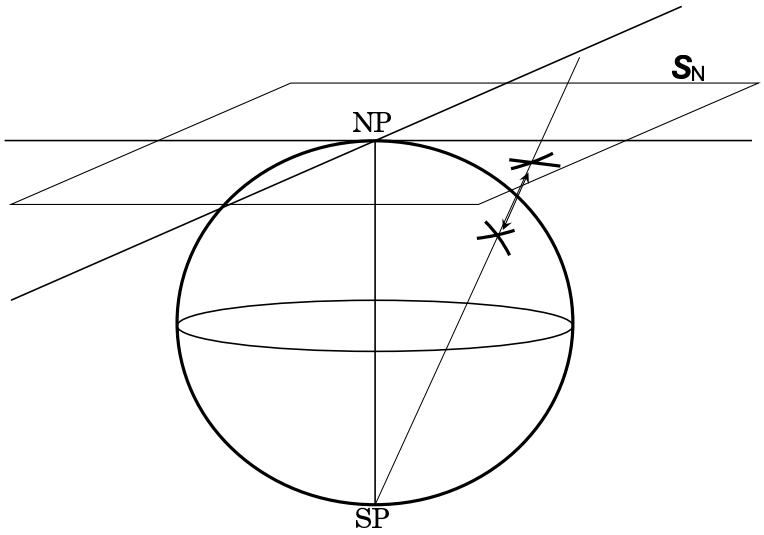

Figure20.1 Schematic illustration of a Polar stereographic projection (a conformal transformation in the general sense between the sphere and $\mathbf{S}_N$).

1. From a point (λ, ϕ) on the sphere to a point z on $\mathbf{S}_N$ (polar stereographic projection). Defining colatitude $\phi' = \pi/2 - \phi$,

$$z = \tan\left(\frac{\phi'}{2}\right) e^{i\lambda}, \tag{20.1}$$

   where the origin of $\mathbf{S}_N$ corresponds to $\phi' = 0$ ($\phi = \pi/2$), and the positive part of the real axis corresponds to $\lambda = 0$.
2. Conformal transformation M_C on $\mathbf{S}_N$:

$$\zeta = M_C(z). \tag{20.2}$$

3. From a point ζ on $\mathbf{S}_N$ to a point (μ, ψ) on the sphere (reverse polar stereographic projection).

$$\mu = \arg(\zeta), \tag{20.3}$$
$$\psi' = 2\arctan|\zeta|, \tag{20.4}$$
$$\psi = \pi/2 - \psi'. \tag{20.5}$$

Functions $\lambda(\mu, \psi)$ and $\phi(\mu, \psi)$ are obtained by reversing the above procedure.
Defining $\psi' = \pi/2 - \psi$,

$$\zeta = \tan\left(\frac{\psi'}{2}\right) e^{i\mu}, \tag{20.6}$$

$$z = M_C^{-1}(\zeta), \tag{20.7}$$

and

$$\lambda = \arg(z), \tag{20.8}$$
$$\phi' = 2\arctan|z|, \tag{20.9}$$
$$\phi = \pi/2 - \phi'. \tag{20.10}$$

Thus, when a model coordinate grid point, $(\mu_0 + \Delta\mu \times (i-1),\ \psi_0 + \Delta\psi \times (j-1))$ is given, we know the geographic position of the point, Coriolis parameter, etc., at once.

Bentzen et al. (1999) used the linear fraction conversion as a conformal transformation on $\mathbf{S}_N$. That is,

$$\zeta \quad = \quad M_C(z) \quad = \quad \frac{(z-a)(b-c)}{(c-a)(b-z)}, \tag{20.11}$$

where the three complex numbers a, b, and c expressed by

$$a = \tan\left(\frac{\phi'_a}{2}\right) e^{i\lambda_a}, \quad b = \tan\left(\frac{\phi'_b}{2}\right) e^{i\lambda_b}, \quad c = \tan\left(\frac{\phi'_c}{2}\right) e^{i\lambda_c}, \tag{20.12}$$

correspond to the three geographic coordinate grid points (λ_a, ϕ_a), (λ_b, ϕ_b), and (λ_c, ϕ_c), which are mapped to the model coordinate grid points, $(\mu, \psi) = (0, \pi/2)$, $(0, -\pi/2)$, $(0, 0)$, respectively. Therefore, the singular point $(\mu, \psi) = (0, \pi/2)$ in the model calculation can be put on Greenland, by setting (λ_a, ϕ_a) at 75°N and 40°W. If `TRIPOLAR` or `JOT` option is specified instead of `SPHERICAL`, then two singular points: $(\mu, \psi) = (0, \pi/2)$ and $(0, -\pi/2)$ can be put on arbitrary land locations by suitably setting (λ_a, ϕ_a) and (λ_b, ϕ_b), which are model parameters (`north_pole_lon`, `north_pole_lat`, `south_pole_lon`, and `south_pole_lat` in degree) that should be specified in namelist `nml_poles` (Table 3.2).

When `TRIPOLAR` option is specified, the parameters are set to $\phi_a = \phi_b = 64$°N, $\lambda_a = 80$°E, and $\lambda_b = 100$°W. The transformed grids are used for the region north of 64°N, and geographic coordinates are used for the region south of 64°N. This tripolar coordinate system can express the Arctic Sea with a higher resolution than the Southern Ocean. The adoption of geographic coordinates south of 64°N enables us to do the assimilation and analysis with relative ease (Figure 20.2).

When `JOT` option is specified, the Joukowski conversion is used as a conformal transformation on $\mathbf{S}_\mathrm{N}$. That is,

$$z = M_C^{-1}(\zeta) = \left(\zeta + \frac{\psi_0'^2}{\zeta}\right) e^{i\mu_0}. \tag{20.13}$$

This Joukowski conversion maps the area outside the circle with a radius of ψ'_0 centered at the origin of the ζ-plane to the whole domain of the z-plane, and rotates it by μ_0. The left panel of Figure 20.2 presents an example where the coordinate system is created by setting ψ'_0 to 20° and μ_0 to 80°. Because there is no discontinuity of grid spacing in this coordinate system, the singular points can be put at various positions. For instance, the singular point on the North American side can be put on the Labrador Peninsula or in Greenland.

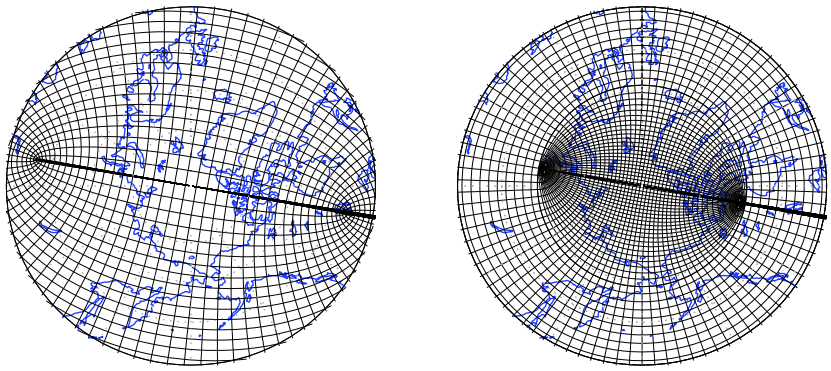

Figure20.2 Model coordinate grid arrangement in the Arctic sea. Left: Grid system made through the Joukowski conversion (`JOT`). Right: Combination of the coordinate systems made through the linear fraction conversion and conventional geographic coordinates (`TRIPOLAR`).

Functions $\lambda(\mu, \psi)$ and $\phi(\mu, \psi)$ are defined as subroutine `mp2lp`, and functions $\mu(\lambda, \phi)$ and $\psi(\lambda, \phi)$ are defined as subroutine `lp2mp`. Module programs `trnsfrm.{spherical, moebius, tripolar, jot}.F90` contain these internal subroutines. These functions, especially `mp2lp`, are frequently used when the topography and the surface boundary condition are made before starting the main integration of model.

20.3 Rotation of vector

A vector expressed in geographic coordinates (λ, ϕ) should be rotated when observed from model coordinates (μ, ψ). Taking advantage of the local orthogonality of coordinate axes, the angle (α) is obtained as an angle at which the meridian

($\lambda = \lambda_0$) of geographic coordinates intersects that of the model coordinates ($\mu = \mu_0$) at a certain point. First, we set

$$z = f(\zeta), \quad z = x + iy, \quad \zeta = u + iv, \tag{20.14}$$

$$f'(\zeta) = \frac{\partial x}{\partial u} + i\frac{\partial y}{\partial u} = \frac{\partial y}{\partial v} - i\frac{\partial x}{\partial v}. \tag{20.15}$$

At a certain point $z_0 = f(\zeta_0)$ on geographic coordinates, the angle θ at which a curve $v = v_0$ meets a straight line $y = y_0$ is given by

$$\tan\theta = \left[\frac{\partial y}{\partial x}\right]_{v_0} = \left[\frac{\partial y}{\partial u}\Big/\frac{\partial x}{\partial u}\right]_{v_0},$$

then (see Figure 20.3),

$$\theta = \arg(f'(\zeta_0)). \tag{20.16}$$

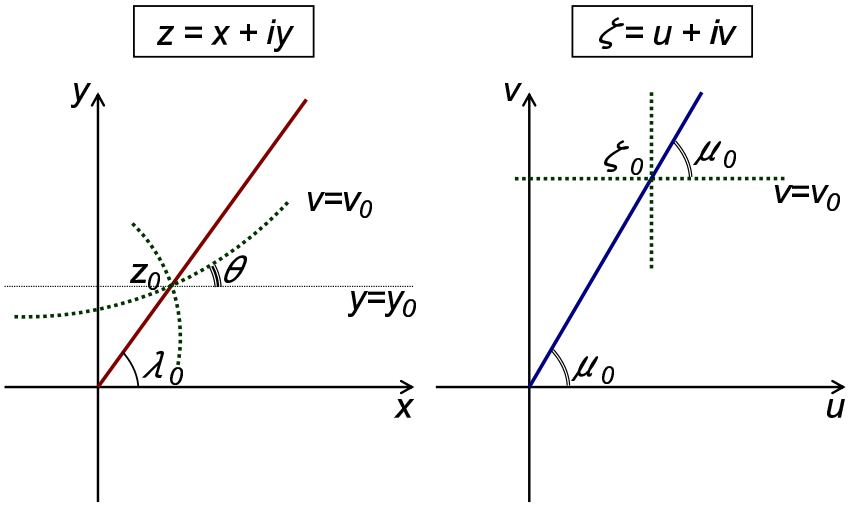

Figure20.3 A meridian (red) in geographic coordinates (λ, ϕ) (left) and a meridian (blue) in model coordinates (μ, ψ) (right).

Assuming $\lambda = \arg(z)$ and $\mu = \arg(\zeta)$, at point ζ_0 the straight line ($v = v_0$) meets the straight line ($\mu = \mu_0$) at the angle $-\mu_0$ ($\mu_0 \to v_0$), and at point z_0 the meridian ($\lambda = \lambda_0$) meets the curve ($v = v_0$) at the angle $\lambda_0 - \theta$ ($v_0 \to \lambda_0$). The meridian ($\lambda = \lambda_0$) in λ-ϕ coordinates meets the line ($\mu = \mu_0$) in μ-ψ coordinates at angle α given as follows:

$$\begin{aligned}\alpha &= -\mu_0 + \lambda_0 - \theta \\ &= \lambda_0 - \mu_0 - \arg(f'(\zeta_0)).\end{aligned} \tag{20.17}$$

Subroutine `rot_mp2lp` defined in `trnsfrm.{spherical,moebius,tripolar,jot}.F90` returns $(\cos\alpha, \sin\alpha)$ at a specified grid point of the model. A wind stress vector (τ_x, τ_y) in geographic coordinates should appear in the model ocean described in the μ-ψ coordinate system as $(\tau_x \cos\alpha - \tau_y \sin\alpha, \tau_x \sin\alpha + \tau_y \cos\alpha)$.

20.4 Mapping a quantity from geographic coordinates to transformed coordinates

We consider a method to receive a quantity $G_{I,J}$ given at the geographic coordinate grids (I, J) as the quantity $H_{M,N}$ at the model coordinate grids (M, N) (Figure 20.4). The quantities are wind stress components after the vector rotation, precipitation per unit area, sea surface atmospheric temperature, and so on. In addition, the average depth at a model grid point can also be calculated by the following method because bottom topography (depths of sea floor) is usually given in geographic coordinates.

Chapter 20 Generalized orthogonal curvilinear coordinate grids

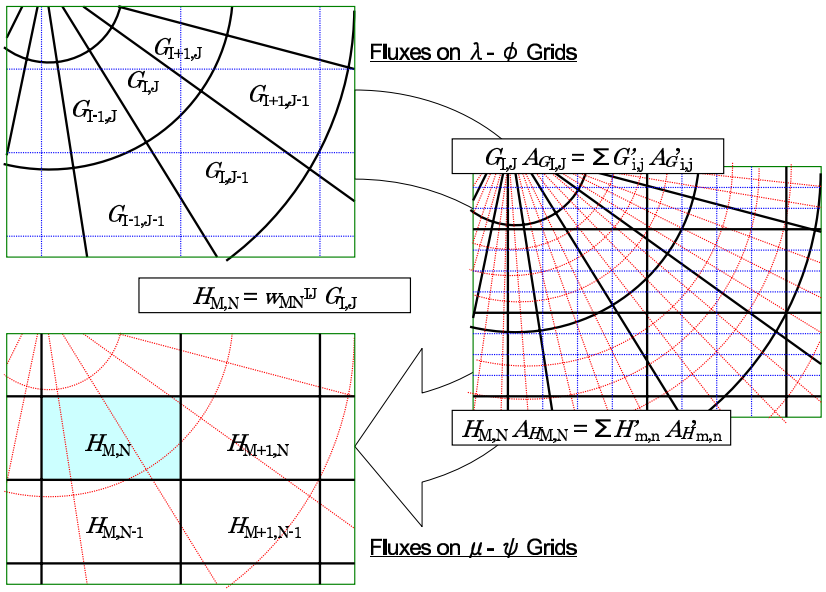

Figure20.4 Grids (I, J) and (M, N) subdivided into finer grids (i, j) and (m, n).

Grids (I, J) and (M, N) are suitably subdivided into finer grids (i, j) and (m, n). We call these filter grids. A quantity $G'_{i,j}$ is assumed to be homogeneously distributed in the geographic filter grids (i, j) covered by grid (I, J),

$$G'_{i,j} = G_{I,J}.$$

Assume the quantity at a model filter grid $H'_{m,n}$ is equal to that at the nearest geographic filter grid,

$$H'_{m,n} = G'_{i(m,n),j(m,n)}.$$

The quantity at model grid (M, N) is obtained as the area-weighted average:

$$H_{M,N} = \frac{1}{A_{HM,N}} \sum_{m,n} A_{H'm,n} H'_{m,n}, \tag{20.18}$$

where $A_{HM,N}$ is the area of model grid and $A_{H'm,n}$ is the area of model filter grid.

When the grid intervals of geographic filter grid (i, j) and model filter grid (m, n) are extremely small, the total quantity (flux) received on the model grids (M, N) is equal to the total quantity (flux) given by the geographic grids (I, J). The relation between the quantity in the geographic grids and that in the model grids is defined by weight w,

$$H_{M,N} = \sum_{I,J} w(M, N, I, J) G_{I,J}. \tag{20.19}$$

How is the quantity converted in an actual calculation in the model?

1. When the strict conservation of quantity (flux) is necessary:
   Fresh water is not permitted to be generated or vanish at the surface boundary in a run using an atmosphere-ocean coupled model, for instance. In this case, $w(M, N, I, J)$ is prepared beforehand, and the flux is passed from the atmosphere through equation (20.19) to the ocean. The resolution of the filter grid need not be extremely fine, provided that every geographic filter grid is linked to more-than-zero model filter grids and

$$\sum_{I,J} A_{GI,J} = \sum_{i,j} A'_{G'i,j} = \sum_{m,n} A'_{H'm,n} = \sum_{M,N} A_{HM,N}.$$

2. When conservation need not be guaranteed:
   When the ocean model is driven by the surface boundary condition based on atmospheric re-analysis data, the amount of fresh water entering the sea as precipitation and river discharge is not equal to that drawn from the ocean

through evaporation and sublimation. Therefore, the global sea surface height rises or descends during years of integration. It is not very important to pursue complete conservation of fresh-water under this condition. In such a case, the flux at a model grid point can be prepared beforehand using equation (20.19), to avoid the time-consuming flux conversions in the model calculation.

20.5 Vector operation and differentiation in generalized orthogonal coordinates

To formulate the model equations, we have to know the vector operation and differentiation in generalized orthogonal coordinates. Some basic formulae used in formulating primitive equations are presented here.

The line element vector $\delta\mathbf{x}$ at a certain point (μ, ψ, r) in an arbitrary general orthogonal coordinate system is expressed as

$$\delta\mathbf{x} = h_\mu \delta\mu \mathbf{e}_\mu + h_\psi \delta\psi \mathbf{e}_\psi + h_r \delta r \mathbf{e}_r, \tag{20.20}$$

where basis vectors $\mathbf{e}_\mu$, $\mathbf{e}_\psi$, and $\mathbf{e}_r$ are mutually orthogonal unit vectors, and h_μ, h_ψ, and h_r are scale factors.

Defining

$$\nabla = \frac{\mathbf{e}_\mu}{h_\mu}\frac{\partial}{\partial\mu} + \frac{\mathbf{e}_\psi}{h_\psi}\frac{\partial}{\partial\psi} + \frac{\mathbf{e}_r}{h_r}\frac{\partial}{\partial r}, \tag{20.21}$$

the gradient of scalar $A(\mu, \psi, r)$ is

$$\nabla A = \frac{\mathbf{e}_\mu}{h_\mu}\frac{\partial A}{\partial\mu} + \frac{\mathbf{e}_\psi}{h_\psi}\frac{\partial A}{\partial\psi} + \frac{\mathbf{e}_r}{h_r}\frac{\partial A}{\partial r}, \tag{20.22}$$

and the divergence of vector $\mathbf{A} = A_\mu\mathbf{e}_\mu + A_\psi\mathbf{e}_\psi + A_r\mathbf{e}_r$ is

$$\nabla\cdot\mathbf{A} = \frac{1}{h_\mu h_\psi h_r}\left[\frac{\partial(h_\psi h_r A_\mu)}{\partial\mu} + \frac{\partial(h_r h_\mu A_\psi)}{\partial\psi} + \frac{\partial(h_\mu h_\psi A_r)}{\partial r}\right]. \tag{20.23}$$

The r component of curl$\mathbf{A}$ is

$$\frac{1}{h_\mu h_\psi}\left[\frac{\partial(h_\psi A_\psi)}{\partial\mu} - \frac{\partial(h_\mu A_\mu)}{\partial\psi}\right]. \tag{20.24}$$

The calculation of velocity advection includes $(\mathbf{a}\cdot\nabla)\mathbf{A}$, where $\mathbf{a}$ is an arbitrary vector ($\mathbf{a} = a_\mu\mathbf{e}_\mu + a_\psi\mathbf{e}_\psi + a_r\mathbf{e}_r$).

The μ component of $(\mathbf{a}\cdot\nabla)\mathbf{A}$ is

$$\mathbf{a}\cdot\nabla A_\mu + \frac{A_\psi}{h_\mu h_\psi}\left(a_\mu\frac{\partial h_\mu}{\partial\psi} - a_\psi\frac{\partial h_\psi}{\partial\mu}\right) + \frac{A_r}{h_r h_\mu}\left(a_\mu\frac{\partial h_\mu}{\partial r} - a_r\frac{\partial h_r}{\partial\mu}\right). \tag{20.25}$$

The second and third terms are so-called "metric" terms in the equation of motion in spherical coordinates.

These expressions in spherical coordinates (λ, ϕ, r) are shown next. Defining longitude λ, latitude ϕ, and radius of the earth r, scale factors are $h_\lambda = r\cos\phi$, $h_\phi = r$, and $h_r = 1$.

Velocity vector $\mathbf{v}$ is

$$\mathbf{v} = u\mathbf{e}_\lambda + v\mathbf{e}_\phi + w\mathbf{e}_r, \tag{20.26}$$

where $\mathbf{e}_\lambda$, $\mathbf{e}_\phi$, and $\mathbf{e}_r$ are the eastward, northward, and upward unit vectors, respectively, and $(u, v, w) = (r\cos\phi\dot{\lambda}, r\dot{\phi}, \dot{r})$.

The gradient of scalar function $A(\lambda, \phi, r)$ is,

$$\nabla A = \frac{\mathbf{e}_\lambda}{r\cos\phi}\frac{\partial A}{\partial\lambda} + \frac{\mathbf{e}_\phi}{r}\frac{\partial A}{\partial\phi} + \mathbf{e}_r\frac{\partial A}{\partial r}, \tag{20.27}$$

where

$$\nabla = \frac{\mathbf{e}_\lambda}{r\cos\phi}\frac{\partial}{\partial\lambda} + \frac{\mathbf{e}_\phi}{r}\frac{\partial}{\partial\phi} + \mathbf{e}_r\frac{\partial}{\partial r}. \tag{20.28}$$

For vector $\mathbf{A} = A_\lambda\mathbf{e}_\lambda + A_\phi\mathbf{e}_\phi + A_r\mathbf{e}_r$, the divergence is

$$\nabla\cdot\mathbf{A} = \frac{1}{r\cos\phi}\left[\frac{\partial A_\lambda}{\partial\lambda} + \frac{\partial(\cos\phi A_\phi)}{\partial\phi}\right] + \frac{\partial(r^2 A_r)}{r^2\partial r}. \tag{20.29}$$

and the r component of curl$\mathbf{A}$ is

$$[\mathrm{curl}\mathbf{A}]_r = \frac{1}{r\cos\phi}\left[\frac{\partial A_\phi}{\partial\lambda} - \frac{\partial(\cos\phi A_\lambda)}{\partial\phi}\right]. \tag{20.30}$$

The λ component of $(\mathbf{a} \cdot \nabla)\mathbf{A}$ is

$$[(\mathbf{a} \cdot \nabla)\mathbf{A}]_\lambda = \mathbf{a} \cdot \nabla A_\lambda - \frac{A_\phi a_\lambda \tan\phi}{r} + \frac{A_r a_\lambda}{r}. \tag{20.31}$$

The Coriolis force in generalized orthogonal coordinates (μ, ψ, r) is given as

$$2\Omega \times \mathbf{v} = (2\Omega_\psi w - 2\Omega_r v)\mathbf{e}_\mu + (2\Omega_r u - 2\Omega_\mu w)\mathbf{e}_\psi + (2\Omega_\mu v - 2\Omega_\psi u)\mathbf{e}_r, \tag{20.32}$$

where $\Omega = \Omega_\mu \mathbf{e}_\mu + \Omega_\psi \mathbf{e}_\psi + \Omega_r \mathbf{e}_r$ is the rotation vector of the Earth, and $\mathbf{v} = u\mathbf{e}_\mu + v\mathbf{e}_\psi + w\mathbf{e}_r$ is the velocity vector. We designate $f_\mu = 2\Omega_\mu$, $f_\psi = 2\Omega_\psi$, and $f = f_r = 2\Omega_r$ in Chapter 2. The rotation vector of the Earth is $(\Omega_\lambda, \Omega_\phi, \Omega_r) = (0, \Omega\cos\phi, \Omega\sin\phi)$ in geographic coordinates (λ, ϕ, r).

Chapter 21

User's Guide

This chapter briefly explains the procedures needed to run MRI.COM. The description in this chapter is based on MRI.COM version 4.4 (MRICOM-4_4_beta01-20170118) and some contents presented in this chapter may not be used for the latest version. It is recommended that users refer to `README.First`, `README.Options`, `README.Namelist`, `README.Monitor`, and `README.Restart` in the `docs` directory when setting up a model.

The minimal information to prepare, run, and post-process is presented in this chapter in the following order:

- **Model setup:** User defined parameter files and compilation (Section 21.1 and Table 21.1).
- **Input data:** Grid spacing, topography, and surface forcing etc., to be read at run time (Section 21.2 and Table 21.2).
- **Restart file:** Explanation of restart files (Section 21.3).
- **Execution:** Explanation of the runtime parameters that control time-integration (Section 21.4).
- **Post process:** A description of monitor files (Section 21.5).

Note that cgs units are employed to express physical values in the model.

We are developing a comprehensive package of tools "MRI.COM eXecution Environment (MXE)" that aggregates programs for preprocessing, execution, postprocessing, and analysis of MRI.COM experiments. This is briefly explained in Section 21.6

21.1 Model setup

21.1.1 Model configuration file (configure.in)

This section describes the procedures necessary for setting up the model and compiling its programs. First, prepare `configure.in` that contains the information about model options and grid size. This is needed for compilation. Following is an example of `configure.in`.

An example `configure.in` for Global tripolar $1^\circ \times 0.5^\circ$ grid model

```
DEFAULT_OPTIONS="IDEALAGE ICE SIDYN CALALBSI SMAGOR VIS9P DIFAJS GLS VVDIMP
 SOMADVEC ISOPYCNAL HFLUX TAUBULK WFLUX RUNOFF SFLUXR LWDOWN BULKNCAR BULKITER
 CYCLIC ZSTAR BBL TRIPOLAR PARALLEL"
#
NAME_MODEL='GLOBAL'
IMUT=364
JMUT=368
KM=51
KSGM=51
KBBL=1
NPARTX=8
NPARTY=4
NUM_ICECAT=5
NUMTRC_P=1
```

Additional parameters are required for some particular model options. Those are listed in Table 21.1 (see also README.Options).

Table21.1 Model parameters to be set in configure.in

option name	variable name	description
always required	IMUT, JMUT, KM	zonal/meridional/vertical grid number
	NAME_MODEL	name of the model (default = "tmOGCM")
	KSGM	The number of variable layers near the sea surface. KSGM must equal KM for ZSTAR option. See Chapter 6.
	NSFMRGN	the number of side-boundary ghost cells to reduce the communication cost in parallel computation (see Ishizaki and Ishikawa, 2006)
BBL	KBBL	the number layers of bottom boundary layer model; must be 1
PARALLEL	NPARTX, NPARTY	the number of zonally/meridionally partitioned region for a computation using parallel processors: the number of parallel processes should be NPARTX × NPARTY
passive tracers	NUMTRC_P	only when any passive tracer is calculated (NUMTRC_P ≥ 1)
ICE	NUM_ICECAT	the number of thickness categories of sea ice

21.1.2 Compilation of the model

A standard compiling script is prepared as compile.sh in the src directory. The part depending on the system (OS, Fortran compiler, and compiler option) are found in machines directory.

To compile the programs, execute compile.sh. The function of compile.sh is to create param.F90, icecat/ice_param.F90, and Makefile from configure.in, param.F90.in, icecat/ice_param.F90.in, and Makefile.in respectively, by running configure and to execute the command make to create the executable file ogcm. The environment variables for compilation are set in configure using the options prescribed in configure.in. If configure.in is newer than param.F90, the parameter values defined in configure.in replace those in param.F90.in to create new param.F90.

The program files that should be compiled are automatically selected according to the descriptions of the relationships in Makefile, but users should be careful since it might not be perfect. The compilation should be carried out after executing ./compile.sh clean when any compile option in configure.in is rewritten.

21.2 Preparation of input data files for execution

According to user's specification of model options and job parameters, the data files listed on Table 21.2 should be prepared along with the files that must be always prepared. See Section 21.3 for how to handle restart files.

Table21.2 Main input data files and their related program files. Here, *name_model* represents the specific name given as NAME_MODEL in configure.in.

subject	file name specified in (NAMELIST.*name_model*)	read from
runtime job parameters	NAMELIST.*name_model*	each package
specification about monitoring	NAMELIST.*name_model*.MONITOR	history.F90 etc.
variable horizontal grid spacing	file_dxdy_tbox_deg (nml_horz_grid)	gridm.F90
variable vertical grid spacing	file_dz_cm (nml_vert_grid)	gridm.F90
grid cell area and line elements	file_scale (nml_grid_scale)	gridm.F90
topography	file_topo (nml_topo)	topo.F90
reference data for tracers	trcref(_surf)_conf%file_data	restore_cond.F90
restoring coefficient for tracers	rstcoef(_surf)_conf%file_data	restore_cond.F90
surface forcing	file_data (nml_force_data)	force.F90
surface forcing grid	file_data_grid (nml_force_data)	force.F90
restart files	nmlrs_*element*, *element* being elements	history.F90

21.2.1 Grid spacing and cell area

Details on how to specify grid information and how to prepare the necessary data is given in Section 3.6. The grid spacing data file should be prepared for each of the horizontal (`file_dxdy_tbox_deg`) or vertical (`file_dz_cm`) directions when variable grid spacing is used for that direction. The units are in degrees for the horizontal and in cm for the vertical.

When the model grid points are defined on the basis of general orthogonal coordinates, the quarter cell area and line elements should be prepared. The units are in cgs. It is read from the file `file_scale`.

When spherical coordinates are used (`SPHERICAL` option), e.g., the grids are defined on geographical latitude and longitude, the grid information is analytically calculated in `gridm.F90`, and the file `file_scale` is not necessary.

21.2.2 Topography

Land-sea distribution and sea-floor topography should be given by the topography data file `file_topo`. The topographic data consist of the 4-byte integer array `HO4(imut, jmut)` that contains the sea floor depths of the velocity grid points (in cm) and the 4-byte integer array `EXNN(imut, jmut)` that contains its corresponding vertical level. They should be written unformatted and sequentially as follows:

Format of topographic data (`file_topo`)

```
integer(4) :: ho4(imut,jmut),exnn(imut,jmut)
open(unit=nu,file=file_topo)
write(unit=nu) ho4, exnn
```

An example of the topography for global $1^\circ \times 0.5^\circ$ model is shown in Figure 21.1. In creating a model topography, especially for a low-resolution model, the user should be careful that the important gateways for the ocean circulation be kept open and that the land blocking the ocean circulation be kept closed.

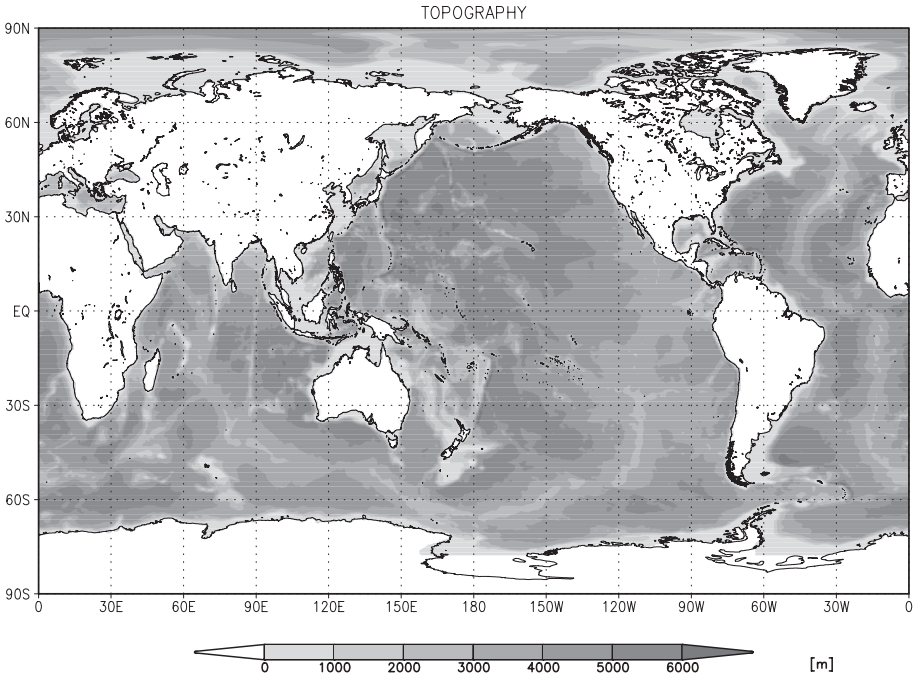

Figure21.1 Example of ocean model topography (global $1^\circ \times 0.5^\circ$ grid model).

21.2.3 Reference data and restoring coefficient for tracers

For each tracer, the integration may be started from an initial state based on the reference data and the tracer values may be restored to the reference data during the integration. To do this, tracer reference values and restoring coefficients should be prepared. See Chapter 13 for how to prepare these data.

21.2.4 Surface forcing data

See Chapter 14 for how to prepare surface forcing data. The surface forcing data are read at a uniform time interval. A leap year is set according to the calendar subroutine. Climatological data may be used repeatedly.

The following data files should be prepared according to the chosen model options. Each file is opened only once at the beginning of run time and thus should contain all the data needed for that run.

Table21.3 Surface data to be read from namelist `nml_force_data`.

	name	units	usage
X-ward wind stress	U-wind	$\mathrm{dyn\,cm^{-2}}$	if not `TAUBULK`
Y-ward wind stress	V-wind	$\mathrm{dyn\,cm^{-2}}$	if not `TAUBULK`
X-ward wind speed	U-wind	$\mathrm{cm\,s^{-1}}$	if `TAUBULK`
Y-ward wind speed	V-wind	$\mathrm{cm\,s^{-1}}$	if `TAUBULK`
Downward shortwave radiation	ShortWave	$\mathrm{erg\,s^{-1}\,cm^{-2}} = 10^{-3}\,\mathrm{W\,m^{-2}}$	
Downward longwave radiation	LongWave	$\mathrm{erg\,s^{-1}\,cm^{-2}}$	Net longwave by default (SST is required). Downward if `LWDOWN`
Sea surface temperature	SST	°C	unnecessary if `LWDOWN`
Surface air temperature	TempAir	°C	
Surface air specific humidity	SphAir	1	
Scalar wind speed	ScalarWind	$\mathrm{cm\,s^{-1}}$	unnecessary if `TAUBULK`
Sea level pressure	SeaLevelPressure	hPa	also available for `SLP`
precipitation	Precipitation	$\mathrm{g\,s^{-1}\,cm^{-2}}$	`WFLUX`
river discharge	RiverDischargeRate	$\mathrm{g\,s^{-1}\,cm^{-2}}$	`RUNOFF`
Sea ice area fraction	IceConcentrationClimatology	1	`ICECLIM`

21.3 Restart file

A set of restart files provides an instantaneous state necessary to resume a model integration. This section explains how to handle restart files. Table 21.4 lists restart variables of MRI.COM as well as their namelist that specifies their restart file attributes.

Table21.4 Restart variables and their namelist block that specifies their restart file attributes.

variable name	namelist block	model options
temperature	`nmlrs_t`	
salinity	`nmlrs_s`	
x velocity	`nmlrs_u`	
y velocity	`nmlrs_v`	
sea surface height	`nmlrs_ssh`	
x barotropic transport	`nmlrs_uml`	
y barotropic transport	`nmlrs_vml`	
x transport due to SSH diffusion	`nmlrs_ssh_dflx_x`	
y transport due to SSH diffusion	`nmlrs_ssh_dflx_y`	
horizontally averaged density and pressure used for the equation of state	`nmlrs_density`	required if `CALPP`

Continued on next page

Table 21.4 – continued from previous page

variable name	namelist block	model options
vertical diffusivity coefficient	nmlrs_vmix_avd	VVDIMP
vertical viscosity coefficient	nmlrs_vmix_avm	VVDIMP
passive tracers	nmlrs_ptrc	NUMTRC_P > 0
x tidal transport	nmlrs_tide_um	TIDE
y tidal transport	nmlrs_tide_vm	TIDE
tidal sea surface height	nmlrs_tide_ssh	TIDE
vertical diffusion coefficient of turbulence kinetic energy	nmlrs_my_avq	MELYAM
turbulence velocity scale	nmlrs_my_q	MELYAM
turbulence length scale	nmlrs_my_alo	MELYAM
vertical diffusion coefficient of turbulence kinetic energy	nmlrs_nk_avq	NOHKIM
turbulence kinetic energy	nmlrs_nk_eb	NOHKIM
vertical diffusion coefficient of turbulence kinetic energy	nmlrs_gls_avk	GLS
vertical diffusion coefficient of generic variable	nmlrs_gls_avp	GLS
turbulence kinetic energy	nmlrs_gls_eke	GLS
generic variable	nmlrs_gls_psi	GLS
turbulent length scale	nmlrs_gls_alo	GLS
moments of second order moment advection scheme	nml_somadv	SOMADVEC and adv_scheme%name = "som" in nml_tracer_data for any tracer
sea ice initial state	nml_seaice_rst_in	ICE
sea ice final state	nml_seaice_rst_out	ICE

21.3.1 Restart for the ocean model

For restart variables of the main part of ocean model, the attributes of their restart files are given by the namelist blocks that start with "nmlrs_". The element names that follow are t, s, u, v, ssh, etc. as listed on Table 21.4. Required elements depend on model options. See docs/README.Restart for more information. Namelists must be written in NAMELIST.*name_model* (default) as follows:

An example namelist for temperature restart files

```
&nmlrs_t
   fname = 'result/rs_t',
/
```

where fname specifies the basename of the restart files. This information is shared by both input and output files. In the above example, the name of a restart file for temperature is result/rs_t.YYYYMMDDHHMMSS. A suffix indicating the date and time of data is always added to the basename.

For passive tracers, namelist nmlrs_ptrc should be repeated as many times as the number of passive tracers (=numtrc_p) with a right order. The order is determined at tracer_vars.F90. An example for a set of biogeochemical tracers is shown in Section 11.6.

Restart files are saved in a Fortran direct-access format, which can be read by a following program.

Program to read a restart file of the ocean model

```
character(14)        :: date = '20010101000000'  !- for 0:00z1JAN2001
real(8)              :: d(imut,jmut,km)
integer(4),parameter :: nu = 10 ! device number

open(nu, file='result/rs_t.'//date, form='unformatted', &
      & access='direct',recl=imut*jmut*km*8)
read(nu,rec=1) d
close(nu)
```

The file endian is that of the computer where the model runs. There are support tools to visualize restart files in the directory, `tools/MK_GRADS`. Please follow the instructions given by `tools/MK_GRADS/00README.txt`. The MXE package (Section 21.6) also has tools for visualization in the directory, `postp/`.

21.3.2 Restart for the SOM advection scheme

When the SOM tracer advection scheme is used (`SOMADVEC`), the model may read and write additional 9 restart files of the 2nd order moments for each tracer. Their names are `BASENAME_SS_NN.YYYYMMDDHHMMSS`, where `BASENAME` indicates a character string specified separately for input and out files by `file_som_in` and `file_som_out` in namelist `nml_somadv`, `SS` elements of the moments (9 types), `NN` a tracer number (`01` for temperature, `02` for salinity, numbers after `03` for other tracers), and the suffix for the date and time. The format of each file is the same as the tracer restart files. See Section 8.6 for the SOM scheme, and `docs/README.Namelist` for other items of `nml_somadv`.

21.3.3 Restart for the sea ice model

To handle restart files of the sea ice model, dedicated namelist blocks must be specified separately for input and output as follows:

An example namelist for sea ice restart input

```
&nml_seaice_rst_in
   file_icecat_restart_in = 'result/rs_ice',
/
```

An example namelist for sea ice restart output

```
&nml_seaice_rst_out
   file_icecat_restart_out_temp = 'result/rs_ice',
   nwrt_rst_ic                  = 480,  !- set same as nstep_seaice
/
```

where `file_icecat_restart_in` and `file_icecat_restart_out_temp` specify the basenames of input and output files, and `nwrt_rst_ic` specifies the time step interval to write instantaneous states (usually, set it the same as the time step number of the integration). See `docs/README.Namelist` for other items. (Before MRI.COM ver.4.2, one must set `l_file_name_calendar = .true.` in `nml_seaice_time_conf` to add a suffix of the date and time to the file name in the same manner as the ocean model. After ver.4.4, the suffix is always added.)

The restart file can be read by the following program.

Program to read restart for the sea ice model

```
integer(4), parameter :: ncat = 5 ! number of thickness categories
! ice concentration
real(8) :: aicen (1:imut,1:jmut,0:ncat), a0iceo(imut,jmut)
! ice thickness
real(8) :: hicen (1:imut,1:jmut,0:ncat), hiceo (imut,jmut)
! averaged sea-ice thickness
real(8) :: hin   (1:imut,1:jmut,0:ncat), hi    (imut,jmut)
! snow depth
real(8) :: hsnwn (1:imut,1:jmut,0:ncat), hsnwo (imut,jmut)
! averaged snow thickness
real(8) :: hsn   (1:imut,1:jmut,0:ncat), hsnw  (imut,jmut)
! ice surface temperature
real(8) :: tsfcin(1:imut,1:jmut,0:ncat), tsfci (imut,jmut)
! ice temperature
real(8) :: t1icen(1:imut,1:jmut,0:ncat)
! sea surface skin temperature
real(8) :: t0icen(1:imut,1:jmut,0:ncat)
! sea surface skin salinity
real(8) :: s0n   (1:imut,1:jmut,0:ncat)
! skin temperature beneath the sea ice
real(8) :: t0iceo(imut,jmut)
! skin temperature in the open leads
real(8) :: t0icel(imut,jmut)
! stress tensor
real(8) :: sigma1(imut,jmut), sigma2(imut,jmut), sigma3(imut,jmut)

integer(4),parameter :: nu = 10 ! device number
integer(4)           :: i = 0, m

open(nu, file='result/rs_ice', form='unformatted', &
  & access='direct', recl=imut*jmut*8)

do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) aicen(1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) hin  (1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) hsn  (1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) hicen(1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) hsnwn(1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) tsfcin(1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) t1icen(1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) t0icen(1:imut,1:jmut,m) ; enddo
do m = 0, ncat ; i = i + 1 ; read(nu,rec=i) s0n(1:imut,1:jmut,m) ; end do

i = i + 1 ; read(nu,rec=i) a0iceo(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) hi(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) hsnw(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) hiceo(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) hsnwo(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) uice(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) vice(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) sigma1(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) sigma2(1:imut,1:jmut)
i = i + 1 ; read(nu,rec=i) sigma3(1:imut,1:jmut)

close(nu)
```

21.3.4 Control of input and output at run time

a. Input

Input of restart files at run time is controlled by namelist listed on Table 21.5. By setting `l_rst_in` = `.true.` in `nml_run_ini_state`, all necessary elements are read from restart files. However, the specification by `l_rst_in` may be overridden for some elements by specifying namelists dedicated to those elements. Note that model tries to read restart files for all passive tracers, thus namelist `nmlrs_ptrc` must be always specified appropriately.

Table21.5 Namelist blocks that control input of restart files.

namelist block	variable name	Usage
`nml_run_ini_state`	`l_rst_in`	`.true.` : read restart files and resume integration `.false.` (default): initial state is set internally No motion (u,v,ssh=0). Initial condition for temperature and salinity are deteremined by `nml_tracer_run`
`nml_barotropic_run`	`l_rst_barotropic_in`	default = `l_rst_in`
`nml_baroclinic_run`	`l_rst_baroclinic_in`	default = `l_rst_in`
`nml_tracer_run`	`l_rst_tracer_in`	default = `l_rst_in` applicable to only temperature and salinity
`nml_vmix_run`	`l_rst_vmix_in`	default = `l_rst_in`
`nml_tide_run`	`l_rst_tide_in`	default = `l_rst_in`
`nml_melyam_run`	`l_rst_melyam_in`	default = `l_rst_in`
`nml_nohkim_run`	`l_rst_nohkim_in`	default = `l_rst_in`
`nml_gls_run`	`l_rst_gls_in`	default = `l_rst_in`
`nml_density_run`	`l_rst_density_in`	default = `l_rst_in`
`nml_seaice_run`	`l_rst_seaice_in`	default = `.false.`
`nml_tracer_data`	`adv_scheme%lrstin_som`	= `.true.` to read restart of SOM scheme
`nml_tracer_data`	`adv_scheme%lrstout_som`	= `.true.` to write restart of SOM scheme

b. Output

Model always try to write restart files of the oceanic part according to the specification by the namelists listed on Table 21.4. Users have to specify those namelists appropriately to obtain restart files and terminate the model normally. Output from the SOM scheme may be suppressed if `adv_scheme%lrstout_som` = `.false.` in `nml_tracer_data`.

21.4 Execution

To run a model, a shell script that handles input/output files, executes the compiled binary `ogcm`, and post-processes is usually prepared. The namelist parameters that control the time integration and the namelist file that specify sampling have to be given.

- `NAMELIST`.*name_model* controls the job and gives parameters to sub-packages. See `docs/README.Namelist` for details.
- `NAMELIST`.*name_model*`.MONITOR` specifies sampling. See `docs/README.Monitor` for details.

Parameters used by sub-packages are explained in corresponding chapters. Runtime parameters relevant to time-integration are listed on Tables 21.6 through 21.11.

Table21.6 Namelist `nml_time_step`.

variable name	units	description	usage
`dt_sec`	sec	unit time interval for equation of motion and tracer	required
`nstep_matsuno_interval`	integer	time step interval with which Matsuno (Euler-backward) time-integration scheme is used	default = 12 steps
`alpha_bryan_1984`	real(8)	acceleration parameter α of Bryan (1984). See Section 2.3.1.	default = 1
`l_monitor_time`	logical	time step monitor is written to standard output ()	default = `.true.`

Table21.7 Namelist `nml_run_period`.

variable name	units	description	usage
`nstep_total`	integer	total number of time step of this run	required

Table21.8 Namelist `nml_exp_start` that specifies start time of the whole experiment.

variable name	units	description	usage
`year`	integer(4)	year	default = −999
`month`	integer(4)	month	default = 1
`day`	integer(4)	day	default = 1
`hour`	integer(4)	hour	default = 0
`minute`	integer(4)	minute	default = 0
`second`	integer(4)	second	default = 0

Table21.9 Namelist `nml_run_ini` that specifies start time of this run.

variable name	units	description	usage
`year`	integer(4)	year	default = 1
`month`	integer(4)	month	default = 1
`day`	integer(4)	day	default = 1
`hour`	integer(4)	hour	default = 0
`minute`	integer(4)	minute	default = 0
`second`	integer(4)	second	default = 0

Table21.10 Namelist `nml_calendar` that specifies treatment of leap year.

variable name	units	description	usage
`l_force_leap`	logical	Use `l_leap` to decide whether the current year is leap or not.	default = `.false.`, the realistic calendar is followed
`l_leap`	logical	the current year is a leap year or not.	default = `.false.`

Table21.11 Namelist `nml_stdout` that specifies standard output (program log).

variable name	units	description	usage
`l_stdout2file`	logical	Standard output is written to separate files for MPI processes	default = `.false.`
`file_base_stdout`		file name is *name_model*-`file_base_stdout`-stdout.XXXX (XXXX: mpi process number)	
`l_debug`	logical	debug mode	default = `.false.`

21.5 Monitoring

This section summarizes outputs of the mean state (history) data used for monitoring and analyses.

21.5.1 History of the ocean and ice models

In MRI.COM version 4, specification of history outputs has been significantly changed and commonized in the ocean and sea ice models. Users prepare a namelist file named `NAMELIST.`*name_model*`.MONITOR` dedicated to history outputs (*name_model* can be specified by `NAME_MODEL` in `configure.in`, default=`OGCM`), and write the following namelist blocks, `nml_history`, in it as needed,

An example namelist for mean temperature output

```
&nml_history
  name          = 'temperature'
  file_base     = 'result/hs_t'
  interval_step = 48
  [suffix       = 'day']
/
```

where `name` specifies the variable name to be output、`file_base` the basename of the history files, `interval_step` the output interval in terms of the integration time step. An option item of `suffix` specifies the depth of calendar date and time used in the file suffix. In the above example, the time-mean temperature is written to the file, `result/hs_t.YYYYMMDD`, at every 48-th time step.

History files are saved in a Fortran direct-access format, which can be read by a following program.

Program to read ocean model history data

```
character(8)         :: date = '20010101'  !- for 1JAN2001
real(4)              :: d(imut,jmut,km)
integer(4),parameter :: nu = 10 ! device number

open(nu, file='result/hs_t.'//date, form='unformatted', &
  & access='direct', recl=imut*jmut*km*4)
read(nu,rec=1) d
close(nu)
```

A GrADS control file to visualize data is also made by default (`result/hs_t.ctl` in the above example).

The state variables that can be monitored by namelist `nml_history` are listed in `docs/README.Monitor`. There are many options in `nml_history`, such as netCDF output, snapshot output, averaging in the model region, output in a specified sub region, addition of an offset value, multiplication by a factor, and so on. See `docs/README.Monitor` for the available options.

History outputs of the sea ice model are also specified by `nml_history` in the same manner as the ocean model. State variables that can be monitored are listed in the sea ice section of `docs/README.Monitor`. The format of output files is also common, except that a missing value for grids of ocean without sea ice should be different from the one for land grids. By default, the value of 0.e0 is used for the former, while -9.99e33 for the latter. (These values can be changed by `nml_seaice_hst` written in `NAMELIST.`*name_model*.)

21.6 MRI.COM eXecution Environment (MXE)

We have been developing a package "MRI.COM eXecution Environment (MXE)" that aggregates programs for preprocessing, execution, postprocessing, and analysis of MRI.COM experiments. By using prepared tools, users can relatively easily perform complex work of model building and various analyses. In fact, this package is also used as a common basis for developing various ocean models at the Meteorological Research Institute.

The MXE package includes the following tools.

- Preprocessing tools for creating experiment setup files (directory `prep`)
- Directory template for running MRI.COM (`exp`)
- Postprocessing tools for visualizing output files (`postp`)
- Various analysis tools (`anl`)

- A Fortran library that provides basic subroutines such as grid and topography information (`lib`)

The main programs are written in Fortran and sample shell scripts are also included for execution. Several tools in the package have been confirmed by unit testing. This package is managed independently from MRI.COM, but it is provided to users as an accompanying material. Since it is often updated like MRI.COM, it is recommended to use the latest version. For details on how to use MXE, see the file `README-MXE.md` in the top directory (in Japanese).

21.7 Appendix

21.7.1 Model options

The model options are listed on Table 21.12. Only major options are listed here. Description about those related to bio-geochemical models can be found in Chapter 11. The description of all options for the latest version can be found in `src/README.Options`. Though an expression like `OGCM_PARALLEL` is used in the source program, `OGCM_` is omitted when users specify options in `configure.in`.

Table21.12 Description of Model Options

Model option	Description
`BBL`	uses the bottom boundary layer model
`BIHARMONIC`	uses biharmonic operator for both horizontal viscosity and diffusion (*) If `ISOPYCNAL` is also selected, the biharmonic form is used only for viscosity and not for diffusion.
`BULKKONDO2`	Kondo (1975) is used for the surface flux bulk formula.
`BULKNCAR`	Large and Yeager (2004; 2009) is used for the surface flux bulk formula. This option corresponds to the COREs. (*) `BULKKONDO2`, `BULKNCAR` is available only for `HFLUX` case.
`BULKITER`	Bulk transfer coefficient is calculated using iterative method if the observed wind speed is not at 10 m. (*) use with `BULKKONDO2` and `BULKNCAR`
`CALALBSI`	Sea-ice albedo is calculated using sea-ice conditions according to Los-Alamos model instead of using a constant value
`CALPP`	considers the time variation of pressure for the equation of state
`CARBON`	bio-geochemical process is included (*) `numtrc_p`= 4 for Obata-Kitamura model; `numtrc_p`= 8 for NPZD model
`CBNHSTRUN`	atmospheric pCO_2 is given from file (*) use with `CARBON`
`CHLMA94`	shortwave penetration scheme with chlorophyll concentration by Morel and Antoine (1994) (*) use with `NPZD` and `SOLARANGLE`
`CYCLIC`	uses zonally cyclic condition
`DIFAJS`	sets large vertical diffusion coefficient ($1.0\,m^2\,s^{-1}$) between unstable points instead of convective adjustment
`F2003`	Program uses Fortran 2003 features
`FSVISC`	calculates viscosity explicitly in the barotropic momentum equation
`GMANISOTROP`	Anisotropic horizontal variation of thickness diffusion is used (*) use with `ISOPYCNAL`
`GMTAPER`	Taper GM vector stream function near the sea surface (*) cannot be used with `SLIMIT`, `GMANISOTROP`, `AFC`

Continued on next page

Table 21.12 – continued from previous page

Model option	Description
GMVAR	Horizontal thickness diffusion is allowed to vary in horizontal
GLS	Generic length scale model of Umlauf and Burchard (2003)
HFLUX	calculates sea surface heat flux using bulk formula
ICE	sea ice is included
ICECLIM	reading climatological sea-ice fractional area from file
ISOPYCNAL	uses isopycnal diffusion and Gent-McWilliams' parameterization for eddy induced tracer transport velocity (thickness diffusion)
ISOTAPER	Taper isopycnal diffusion coefficient (ISOPYCNAL)
LWDOWN	Longwave radiation data include only downward component instead of default net radiation
MELYAM	uses Mellor and Yamada Level 2.5 for mixed layer model
MPDATAADVEC	uses MPDATA for tracer advection
NOHKIM	uses Noh's mixed layer model
NOMATSUNO	uses forward finite difference instead of Matsuno scheme
NPZD	NPZD process is included (*) use with CARBON
OFFNESTPAR	Used as the lower-resolution part of an off-line one-way nesting calculation
OFFNESTSUB	Used as the higher-resolution part of an off-line one-way nesting calculation
PARALLEL	parallel calculation using MPI. The number of zonally and meridionally partitioned regions should be specified as NPARTX and NPARTY, respectively.
PARENT	executed as low resolution model of the nesting calculation
RUNOFF	uses river runoff data (*) available only for WFLUX
SFLUXR	SSS is restored to the climatological sea surface salinity as the salinity flux
SFLUXW	calculates salinity flux converting from the surface freshwater flux (*) available only for WFLUX
SIDYN	sea-ice dynamics model with EVP rheology (*) available only for ICE
SLIMIT	Tapers thickness diffusion near the sea surface
SLP	Sea surface is elevated/depressed according to surface atmospheric pressure
SMAGHD	uses the Smagorinsky viscosity coefficient multiplied by a constant ratio as the horizontal diffusion coefficient (*) available only for SMAGOR
SMAGOR	uses the Smagorinsky parameterization for horizontal viscosity
SOLARANGLE	solar insolation angle is considered in calculating shortwave penetration
SOMADVEC	uses second order moment advection by Prather (1986)
SPHERICAL	calculates scale factor semi-analytically for spherical coordinates
SUB	executed as a high resolution model of the nesting calculation
TAUBULK	calculates the wind stress using bulk formulae by reading wind speed over the ocean
TDEW	reads dew-point temperature and converts to specific humidity (*) available only for HFLUX
TIDE	Tide producing forcing is activated
TRCBIHARM	uses biharmonic operator for horizontal diffusion (*) Should not be used with ISOPYCNAL

Continued on next page

Table 21.12 – continued from previous page

Model option	Description
TRIPOLAR	Tripolar system is used to construct model grids of a global model (*) Cannot be used with SPHERICAL
UTZQADVEC	"UTOPIA" and "QUICKEST" scheme can be used for horizontal and vertical tracer advection with ultimate limiter
VISANISO	Anisotropic viscosity coefficients are used (*) use with VIS9P
VIS9P	calculates the viscosity using adjacent 9 grid points
VISBIHARM	uses biharmonic operator for both horizontal viscosity
VMBG3D	reads 3-D vertical viscosity and diffusion coefficients from a file
VVDIMP	calculates the vertical diffusion/viscosity by implicit method (*) it is automatically loaded if any mixed layer model is used or ISOPYCNAL option is selected
WADJ	adjusts sea surface freshwater flux every time step to keep its global sum to be zero (*) available only for WFLUX
WFLUX	uses the sea surface freshwater flux to force the model
ZSTAR	zstar (z*) vertical coordinate (*) KSGM = KM
MOVE	used as ocean module for data assimilation (MOVE) system
SCUP	use simple coupler (SCUP) library
SCUPCGCM	used as an ocean module for a coupled model using scup for communication
SCUPNEST	on-line nesting (*) use with SCUP