PCOM_en Manual

Niansen Ou
LASG, Institute of Atmospheric Physics, Chinese Academy of Sciences

Apr, 2014

Contents

1 Introduction

Manual of PCOM
This document is generated using Protex [1] (modified by Ou).
The purpose of this manual is to document the code used by Zhang (2013).

[1] http://gmao.gsfc.nasa.gov/software/protex

2 Model Formulation

2.1 The Basic Equations

The horizontal momentum equations in spherical coordinates may be expressed in the form (Note 2014)

$$\frac{du}{dt} = f^* v - \frac{1}{a\rho\cos\varphi}\frac{\partial p}{\partial \lambda} + \mathcal{D}_u, \tag{2.1a}$$

$$\frac{dv}{dt} = -f^* u - \frac{1}{a\rho}\frac{\partial p}{\partial \varphi} + \mathcal{D}_v, \tag{2.1b}$$

where a is the radius of the Earth, φ latitude, λ longitude, u and v are the zonal and meridional velocity components, $f^* = 2\Omega\sin\varphi + u\tan\varphi/a$ the apparent Coriolis parameter, ρ density, p pressure, $\mathcal{D}_u$ and $\mathcal{D}_v$ frictional force per unit mass.

d/dt is defines as (Note 2014)

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial}{\partial \lambda} + \frac{v}{a}\frac{\partial}{\partial \varphi} + w\frac{\partial}{\partial z}, \tag{2.2}$$

where $w \equiv dz/dt$ is the vertical velocity.

For the large-scale motions, the vertical momentum equation is reduced to the hydrostatic relation (Note 2014)

$$\frac{\partial p}{\partial z} = -\rho g. \tag{2.3}$$

The mass conservation equation is (Note 2014)

$$\frac{1}{\rho}\frac{d\rho}{dt} + \nabla\cdot\mathbf{v} = 0, \tag{2.4}$$

where the velocity divergence $\nabla\cdot\mathbf{v}$ is (Note 2014)

$$\nabla\cdot\mathbf{v} = \frac{1}{a\cos\varphi}\left[\frac{\partial u}{\partial \lambda} + \frac{\partial(v\cos\varphi)}{\partial\varphi}\right] + \frac{\partial w}{\partial z}. \tag{2.5}$$

The equation of state of sea water can be expressed as

$$\rho = \rho(T, S, p), \tag{2.6}$$

where T is the potential temperature, S is salinity.

The calculation of the density is based on the formulas recommended by Millero and Poisson (1981) by modifying the coefficients for inputting the potential temperature (Jackett and Mcdougall 1995).

Note that we use the potential temperature as the tracer; thus, during the invertible adiabatic processes, both the potential temperature and salinity remain constant, The tracers equations are

$$\frac{dT}{dt} = \mathcal{Q}_T \tag{2.7}$$

$$\frac{dS}{dt} = \mathcal{Q}_S \tag{2.8}$$

where $\mathcal{Q}_T$ and $\mathcal{Q}_S$ represent the turbulent diffusion for potential temperature and salinity, respectively.

2.2 Equations in p-σ Coordinate

In order to model the general circulation in a compressible ocean, PCOM will be formulated in a pressure coordinate system. Since the surface pressure can vary with time and space, the normalized p-σ coordinate has therefore been adopted, which is defined as

$$\sigma = \frac{p - p_a}{p_h} \tag{2.9}$$

where p is the pressure of the water grid at current time t, $p_a = p_a(\lambda, \varphi, t)$ is atmospheric pressure at the sea surface (SLP), $p_h = p_h(\lambda, \varphi, t)$ is the sea bottom pressure (already remove the pressure at the sea level).
Also, we define a normalized bottom pressure σ_h

$$\sigma_h = \frac{p_h}{p_{h0}} \tag{2.10}$$

where $p_{h0} = p_h(\lambda, \varphi, 0)$ is sea bottom pressure a initial state ($t = 0$), which is the bottom pressure of initial state $p_b(\lambda, \varphi, 0)$ minus the SLP $p_a(\lambda, \varphi, 0)$ of initial state $t = 0$.
Combine the above two equations, we have

$$\sigma = \frac{p - p_a}{\sigma_h p_{h0}} \tag{2.11}$$

σ will always be 0 at the surface at any time t, and 1 at the sea bottom. But σ_h wil be slightly greater or less than 1, and it will not be a uniform value at the bottom too. Only at the initial state, σ_h will be 1 everywhere at the sea bottom. The following derivation use σ_h extensively.
The vetical integration in σ will be transformed to p as

$$\int_0^\sigma A d\sigma = \frac{1}{\sigma_h p_{h0}} \int_{pa}^{p} A dp \tag{2.12}$$

where $d\sigma = dp/p_h = dp/(\sigma_h p_{h0})$.
The horizontal momentum equations in the p-σ coordinate can be rewritten as (see appendix A.1 for derivation)

$$\frac{du}{dt} = f^* v - \frac{1}{a\cos\varphi}\left(\frac{\partial \phi}{\partial \lambda} + \frac{1}{\rho}\frac{\partial p}{\partial \lambda}\right) + \mathcal{D}_u, \tag{2.13a}$$

$$\frac{dv}{dt} = -f^* u - \frac{1}{a}\left(\frac{\partial \phi}{\partial \varphi} + \frac{1}{\rho}\frac{\partial p}{\partial \varphi}\right) + \mathcal{D}_v, \tag{2.13b}$$

The substantial time difference operator (see appendix A.7 for derivation)

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \frac{u}{a\cos\varphi}\frac{\partial}{\partial \lambda} + \frac{v}{a}\frac{\partial}{\partial \varphi} + \dot{\sigma}\frac{\partial}{\partial \sigma} \tag{2.14}$$

The viscous stress terms are parameterized as (see appendix C for more information)

$$\mathcal{D}_u = \frac{1}{p_h}\nabla_\sigma \cdot (p_h A_m \nabla_\sigma u) + \frac{1}{p_h}\frac{\partial}{\partial \sigma}\left(\frac{\rho^2 g^2}{\sigma_h}\kappa_m \frac{\partial u}{\partial \sigma}\right) + A_m\left(\frac{1 - \tan^2\varphi}{a^2}u - \frac{2\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial \lambda}\right), \tag{2.15a}$$

$$\mathcal{D}_v = \frac{1}{p_h}\nabla_\sigma \cdot (p_h A_m \nabla_\sigma v) + \frac{1}{p_h}\frac{\partial}{\partial \sigma}\left(\frac{\rho^2 g^2}{\sigma_h}\kappa_m \frac{\partial v}{\partial \sigma}\right) + A_m\left(\frac{1 - \tan^2\varphi}{a^2}v + \frac{2\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial \lambda}\right), \tag{2.15b}$$

in which A_m and κ_m represent lateral and vertical viscosity, respectively. ∇_σ is the horizontal gradient operator in pressure-σ coordinate. And

$$\nabla_\sigma \cdot (p_h A_m \nabla_\sigma u) = \frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial \lambda}\left(p_h A_m \frac{\partial u}{\partial \lambda}\right) + \frac{1}{a^2\cos\varphi}\frac{\partial}{\partial \varphi}\left(p_h A_m \cos\varphi \frac{\partial u}{\partial \varphi}\right) \tag{2.16}$$

Here we assume that lateral mixing of tracer is along the pressure-σ surface. $\nabla_\sigma \cdot (p_h A_m \nabla_\sigma v)$ has a similar form as the above.

In order to form a set of finite difference schemes which may guarantee the total energy be conserved, similar to 曾庆存, 张学洪 (1987), we introduce a new horizontal "velocity" $\mathbf{V}$

$$\mathbf{V} = (U, V) = (\sqrt{p_h}u, \sqrt{p_h}v) \tag{2.17}$$

Note that $\frac{1}{2}\mathbf{V} \cdot \mathbf{V}$ just represents the kinetic energy per unit volume in the pressure-σ coordinates. Multiplying by $\sqrt{p_{bt}}$ and with the aid of the mass conservation equation, the momentum equations may be rewritten as (see appendix A.2)

$$\frac{\partial U}{\partial t} + \mathcal{M}(U) = f^* V - \frac{\sqrt{p_h}}{a\cos\varphi}\left(\frac{\partial \phi}{\partial \lambda} + \frac{1}{\rho}\frac{\partial p}{\partial \lambda}\right) + \sqrt{p_h}\mathcal{D}_u \tag{2.18a}$$

$$\frac{\partial V}{\partial t} + \mathcal{M}(V) = -f^* U - \frac{\sqrt{p_h}}{a}\left(\frac{\partial \phi}{\partial \varphi} + \frac{1}{\rho}\frac{\partial p}{\partial \varphi}\right) + \sqrt{p_h}\mathcal{D}_v \tag{2.18b}$$

where $\mathcal{M}$ is an advection operator

$$\mathcal{M}(\mu) = \frac{1}{a\cos\varphi}\left[\frac{\partial(u\mu)}{\partial\lambda} - \frac{\mu}{2}\frac{\partial u}{\partial\lambda} + \frac{\partial(v\mu\cos\varphi)}{\partial\varphi} - \frac{\mu}{2}\frac{\partial(v\cos\varphi)}{\partial\varphi}\right] + \frac{\partial(\dot\sigma\mu)}{\partial\sigma} - \frac{\mu}{2}\frac{\partial\dot\sigma}{\partial\sigma} \tag{2.19}$$

The hydrostatic equation is

$$\frac{\partial\sigma}{\partial z} = \frac{\partial}{\partial z}\left(\frac{p - p_t}{p_h}\right) = \frac{1}{p_h}\frac{\partial p}{\partial z} = -\frac{\rho g}{p_h} \tag{2.20}$$

The mass conservation equation is (see appendix A.3 for derivation)

$$\frac{\partial p_h}{\partial t} + \nabla \cdot (p_h \mathbf{v}) = 0, \tag{2.21}$$

where $\mathbf{v} = u\,\mathbf{e}_\lambda + v\,\mathbf{e}_\varphi + \dot\sigma\,\mathbf{e}_\sigma$ is velocity vector, $\dot\sigma \equiv \frac{d\sigma}{dt}$. and the transport term

$$\nabla \cdot (p_h \mathbf{v}) = \frac{1}{a\cos\varphi}\left[\frac{\partial(p_h u)}{\partial\lambda} + \frac{\partial(p_h v\cos\varphi)}{\partial\varphi}\right] + \frac{\partial(p_h\dot\sigma)}{\partial\sigma} \tag{2.22}$$

Equation 2.21 also serves as the surface pressure tendency equation.

The transport equations for tracers (potential temperature T and salinity S) are (see appendix A.4 for derivation)

$$\frac{\partial(p_h T)}{\partial t} + \nabla \cdot (p_h T\mathbf{v}) = p_h \mathcal{Q}_T, \tag{2.23a}$$

$$\frac{\partial(p_h T)}{\partial t} + \nabla \cdot (p_h S\mathbf{v}) = p_h \mathcal{Q}_S, \tag{2.23b}$$

where $\nabla \cdot (p_h T\mathbf{v})$ and $\nabla \cdot (p_h S\mathbf{v})$ have the similar form as Eq. 2.22, and the diffusion terms are parameterized as

$$\mathcal{Q}_T = \frac{1}{p_h}\nabla_\sigma \cdot (p_h A_h \nabla_\sigma T) + \frac{1}{p_h}\frac{\partial}{\partial\sigma}\left(\frac{\rho^2 g^2}{p_h}\kappa_h\frac{\partial T}{\partial\sigma}\right), \tag{2.24a}$$

$$\mathcal{Q}_S = \frac{1}{p_h}\nabla_\sigma \cdot (p_h A_h \nabla_\sigma S) + \frac{1}{p_h}\frac{\partial}{\partial\sigma}\left(\frac{\rho^2 g^2}{p_h}\kappa_h\frac{\partial S}{\partial\sigma}\right), \tag{2.24b}$$

in which A_h and κ_h represent lateral and vertical diffusivity, respectively. Here we assume that lateral mixing of tracer is along the pressure-σ surface.

Note that the tracers equations have be rewritten in the flux form with the aid of the continuity equation so that the prediction variables are $p_{bt}T$ and $p_{bt}S$ rather than T and S in order to keep the total tracers be conserved in the numerical model.
Equations (2.13), (2.21), and (2.23a) are prediction equations.
The state equation of seawater is

$$\rho = \rho(T, S, p) \tag{2.25}$$

The diagnostic equation of geopotential height is (see appendix A.5 for derivation)

$$\phi = gz_b + \int_\sigma^1 \frac{p_h}{\rho} d\sigma \tag{2.26}$$

The vertical velocity $\dot{\sigma}$ is diagnosed by (see appendix A.6 for derivation)

$$p_h\dot{\sigma} = (p_h\dot{\sigma})_{\sigma=0} - \sigma\frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\int_0^\sigma \left[\frac{\partial (p_h u)}{\partial \lambda} + \frac{\partial (p_h v\cos\varphi)}{\partial \varphi}\right] d\sigma \tag{2.27}$$

2.3 The Boundary Conditions

At all side walls and bottom, the velocity is zero and there is no heat and salt flux cross bottom and lateral boundary. At the free surface ($\sigma = 0$), the kinematic boundary conditions are

$$p_h\dot{\sigma} = -\rho_0 g F_{E-P}, \tag{2.28}$$

$$\frac{\rho^2 g}{p_h} A_v \frac{\partial u}{\partial \sigma} = -\tau_\lambda, \tag{2.29}$$

$$\frac{\rho^2 g}{p_h} A_v \frac{\partial v}{\partial \sigma} = -\tau_\varphi, \tag{2.30}$$

where F_{E-P} is the rate of evaporation minus precipitation; ρ_0 is the density of fresh water; τ_λ and τ_φ represent the zonal and meridional component wind stress, respectively. Note that the river runoff may be taken into account by combined with evaporation minus precipitation.
The turbulent mixing for temperature at sea surface is

$$\frac{\rho^2 g}{p_h} \kappa_v \frac{\partial T}{\partial \sigma} = -\frac{F_h}{C_p}, \tag{2.31}$$

where F_h is the air-sea heat flux, C_p is the heat capacity of sea water.
The restoring boundary condition may be written as

$$\frac{\rho^2 g}{p_h} \kappa_v \frac{\partial S}{\partial \sigma} = -\frac{p_h \delta\sigma_1}{g\tau_s}(S^* - S_1), \tag{2.32}$$

where τ_s is the restoring time scale, $\delta\sigma_1$ is the upper layer resolution in the pressure-σ coordinates, S^* is the observed sea surface salinity, and S_1 is the salinity in the upper layer.
The natural boundary condition (Huang 1993) is

$$\frac{\rho^2 g}{p_h} \frac{\partial S}{\partial \sigma} = -S_1 \rho_0 F_{E-P}. \tag{2.33}$$

Note the turbulent salt flux at the sea surface exactly balances the vertical advection flux induced by the imbalance of evaporation minus precipitation, so that the total salt flux cross the air-sea interface is zero.

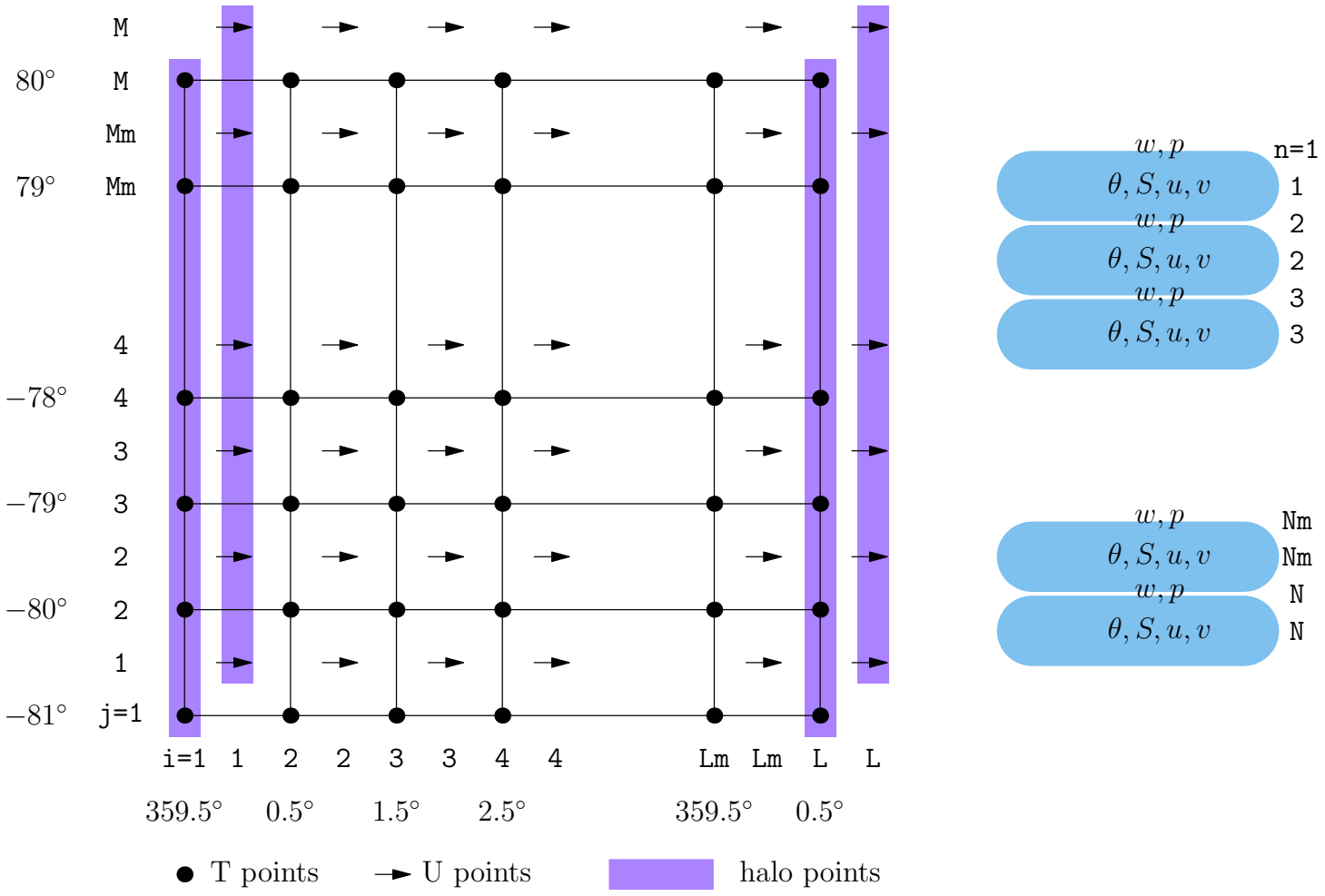

Figure 1: PCOM horizontal stagger B-grid and vertical layer

3 The Finite-Difference Scheme

The grid system is a rectangular Arakawa staggered B-grid (Bryan 1969) containing T cells and U cells (Fig.1). The T cell is arranged at the integer grid which defines the location of tracer quantities, and the U cell is arranged at the half-integer grid which defines the location of the zonal and meridional velocity components.

In the vertical direction, there are also two types of grid. One is the grid lies on the middle of the layers, the other is on the boundary of each layers. The location of the vertical velocity component $\dot{\sigma}$ is defined on the surface of T cell (i.e., at the level $\sigma_{k-1/2}$), and the geopotential ϕ is defined at the center of T cell (i.e., at the level σ_k).

In horizontal, we define T grid as integer grid, and in vertical, the middle of layers as integer grid. We use i, j, k to identify integer grid, use ih, jh, kh to identify the corresponding half-integer grid.

For example, we can classify the variables by grids:

`ijk-grid` - T (potential temperature), S (salinity), p (pressure), w (vertical velocity), ρ (water density), ϕ (geopotential height).

`ihjhk-grid` - U (zonal velocity), V (meridional velocity).

`ijkh-grid` - p (pressure), w (vertical velocity).

`ij-grid` - (sea surface height).

`ih-grid` - *lat*

`i-grid` - latt

The finite-difference schemes are based on 曾庆存, 张学洪 (1987). Here we introduce the following operator for average and difference at the half-integer grids and integer grids

1. $\overline{(\ )}^{\mathtt{A}}$ Average operator, linear interpolate variables between integer grid and half integer grid. e.g., assuming F is a 3d variable at `ijk` grid (T, S are such variables), then
   - $\overline{F}^{\mathtt{Ai}} = (F_i + F_{i+1})/2$, and will be at `ihjk` grid.
   - $\overline{F}^{\mathtt{Ak}}$ will be at `ijkh` grid by linear interpolate F_k and F_{k+1} (note that the grid is un-uniform in vertical direction).
   - $\overline{F}^{\mathtt{Aij}} = (F_{ij} + F_{i+1,j} + F_{i,j+1} + F_{i+1,j+1})/4$, and will be at `ihjhk` grid.

   If F is at half-integer grid, the analogue conclusion stand, except that the result change to integer grid.
2. $\overline{(\ )}^{\mathtt{D}}$ Difference operator, it will also switch the variable between integer grid and half-integer grid. e.g., assuming F is a 3d variable at `ijk` grid (T, S are such variables), then
   - $\overline{F}^{\mathtt{Di}} = F_{i+1} - F_i$, and will be at `ihjk` grid.
   - $\overline{F}^{\mathtt{Dk}} = F_{k+1} - F_k$, and will be at `ijkh` grid. (note that the grid is un-uniform in vertical direction).

   If F is at half-integer grid, the analogue conclusion stand, except that the result change to integer grid.
3. $\overline{(\ )}^{\mathtt{P}}$ Partial operator, it will calculate the partial difference at the indicate direction. e.g., assuming F is a 3d variable at `ijk` grid (T, S are such variables), then
   - $\overline{F}^{\mathtt{Px}} = (F_{i+1} - F_i)/(a\cos\varphi\Delta\lambda)$, and will be at `ihjk` grid.
   - $\overline{F}^{\mathtt{Py}} = (F_{j+1} - F_j)/(a\Delta\varphi)$, and will be at `ijhk` grid.

Thus, the surface pressure tendency equation, by integrating (2.27) in vertical direction, may be written as (see appendix B.1 for derivation)

$$\left[\frac{\partial p_h}{\partial t}\right]_{i,j} = -\frac{1}{a\cos\varphi_j\Delta\lambda}\overline{\overline{PU_B}^{Aj}}^{Di} - \frac{1}{a\cos\varphi_j\Delta\varphi}\overline{\overline{PV_B\cos\varphi}^{Ai}}^{Dj} - \rho_0 g F_{E-P}, \tag{3.1}$$

where $P = \sqrt{\overline{p_h}^{i,j}}$, $U_B = \int_0^1 U d\sigma$, and $V_B = \int_0^1 V d\sigma$. This scheme guarantees total mass conservation if

$$\sum_i\sum_j F_{E-P} a^2 \cos\varphi_j \Delta\lambda\Delta\varphi = 0 \tag{3.2}$$

i.e., no net global or domain averaged freshwater flux cross the air-sea interface.
Similarly, the diagnostic vertical velocity can also be obtained by integrating (2.27) from surface to the k layer of σ, that is

$$[p_h\dot{\sigma}]_{i,j,k} = -\left[\rho_0 g F_{E-P} + \sigma_k\frac{\partial p_h}{\partial t} + \frac{1}{a\cos\varphi_j}\sum_{k=0}^{k}\left(\overline{\overline{PU}^{Aj}}^{Di} + \overline{\overline{PV\cos\varphi}^{Ai}}^{Dj}\right)\Delta\sigma_k\right] \tag{3.3}$$

The momentum equations may be expressed by

$$\left[\frac{\partial U}{\partial t}\right]_{i+\frac{1}{2},j+\frac{1}{2},k} = \left[-\mathcal{M}(U) + f^* V - \frac{P}{a\cos\varphi_{j+\frac{1}{2}}\Delta\lambda}\left(\overline{\overline{\phi}^{Aj}}^{Di} + \overline{\alpha}^{i,j}\overline{\overline{P}^{Aj}}^{Di}\right) + P\mathcal{D}_u\right], \tag{3.4a}$$

$$\left[\frac{\partial V}{\partial t}\right]_{i+\frac{1}{2},j+\frac{1}{2},k} = \left[-\mathcal{M}(V) - f^* U - \frac{P}{a\Delta\varphi}\left(\overline{\overline{\phi}^{Ai}}^{Dj} + \overline{\alpha}^{i,j}\overline{\overline{P}^{Ai}}^{Dj}\right) + P\mathcal{D}_v\right], \tag{3.4b}$$

in which $\alpha = \rho^{-1},\ u = U/P,\ v = V/P$. Note in PCOM $U,\ V$ are the prognostic variables but the real velocity u and v are the diagnostic variables. The finite-difference scheme of the advection operator is (see B.2 for derivation)

$$\mathcal{M}(\mu) = \frac{1}{2a\cos\varphi_{j+\frac{1}{2}}\Delta\lambda}\left[2\widetilde{\overline{u}^i\overline{\mu}^i}^i - \mu\overline{\overline{u}^{Ai}}^{Di}\right] + \frac{1}{2a\cos\varphi_{j+\frac{1}{2}}\Delta\varphi}\left[2\overline{\mu}^j\widetilde{\overline{v\cos\varphi}^j}^j - \mu\overline{\overline{v\cos\varphi}^{Aj}}^{Dj}\right] + \frac{1}{\Delta\sigma_k}\left(\widetilde{\dot{\overline{\sigma}}^{i,j}\overline{\mu}^k}^k - \frac{\mu}{2}\overline{\dot{\overline{\sigma}}^{Ai,j}}^{Dk}\right. \tag{3.5}$$

where

$$\Delta\sigma_k = \sigma_{k+1/2} - \sigma_{k-1/2}, \tag{3.6a}$$

$$\Delta\sigma_{k+\frac{1}{2}} = \sigma_{k+1} - \sigma_k. \tag{3.6b}$$

The viscous terms are

$$\begin{aligned} P\mathcal{D}_u = {} & \frac{1}{Pa^2\cos^2\varphi_{j+\frac{1}{2}}\Delta^2\lambda}\overline{\overline{\sigma_h}^{\mathrm{Aj}}A_m\overline{u}^{\mathrm{Dj}}}^{\mathrm{Di}} \\ & + \frac{\cos\varphi_{j+1}}{Pa^2\cos\varphi_{jh}\Delta^2\varphi}\left(\overline{\sigma_h}^{\mathrm{Ai}}A_m\overline{u}^{\mathrm{Dj}}\right)_{j+1} - \frac{\cos\varphi_j}{Pa^2\cos\varphi_{jh}\Delta^2\varphi}\left(\overline{\sigma_h}^{\mathrm{Ai}}A_m\overline{u}^{\mathrm{Dj}}\right)_j \\ & + \frac{1}{P}\underline{\frac{\overline{\rho}^{k^2}g^2}{\overline{p_h}^k}\kappa_m\overline{u}^k}_k + A_m\left(\frac{1-\tan^2\varphi_{j+\frac{1}{2}}}{a^2}U - \frac{2P\tan\varphi_{j+\frac{1}{2}}}{a^2\cos\varphi_{jh}\Delta\lambda}\overline{\overline{v}^{\mathrm{Ai}}}^{\mathrm{Di}}\right) \end{aligned} \tag{3.7a}$$

$$\begin{aligned} P\mathcal{D}_v = {} & \frac{1}{Pa^2\cos^2\varphi_{j+\frac{1}{2}}\Delta^2\lambda}\overline{\overline{\sigma_h}^{\mathrm{Aj}}A_m\overline{v}^{\mathrm{Dj}}}^{\mathrm{Di}} \\ & + \frac{\cos\varphi_{j+1}}{Pa^2\cos\varphi_{jh}\Delta^2\varphi}\left(\overline{\sigma_h}^{\mathrm{Ai}}A_m\overline{v}^{\mathrm{Dj}}\right)_{j+1} - \frac{\cos\varphi_j}{Pa^2\cos\varphi_{jh}\Delta^2\varphi}\left(\overline{\sigma_h}^{\mathrm{Ai}}A_m\overline{v}^{\mathrm{Dj}}\right)_j \\ & + \frac{1}{P}\underline{\frac{\overline{\rho}^{k^2}g^2}{\overline{p_h}^k}\kappa_m\overline{v}^k}_k + A_m\left(\frac{1-\tan^2\varphi_{j+\frac{1}{2}}}{a^2}V + \frac{2P\tan\varphi_{j+\frac{1}{2}}}{a^2\cos\varphi_{jh}\Delta\lambda}\overline{\overline{u}^{\mathrm{Ai}}}^{\mathrm{Di}}\right) \end{aligned} \tag{3.7b}$$

The tracers equations are

$$\left[\frac{\partial(p_h T)}{\partial t}\right]_{i,j,k} = -ADV(T) + DIF(T) \tag{3.8a}$$

$$\left[\frac{\partial(p_h S)}{\partial t}\right]_{i,j,k} = -ADV(S) + DIF(S) \tag{3.8b}$$

where

$$ADV(T) = \frac{1}{a\cos\varphi_j}\left(\underline{\overline{PU}^j\overline{T}^i}_i + \underline{\overline{PV\cos\varphi_{jh}}^i\overline{T}^j}_j\right) + \underline{p_h\dot{\sigma}\overline{T}^k}_k \tag{3.9}$$

$$\begin{aligned} DIF(T) = {} & \frac{1}{a^2\cos^2\varphi_j\Delta^2\lambda}\overline{\overline{\sigma_h}^{\mathrm{Ai}}A_h\overline{T}^{\mathrm{Di}}}^{\mathrm{Di}} + \frac{\cos\varphi_{jh}}{a^2\cos\varphi_j\Delta^2\varphi}\left(\overline{\sigma_h}^{\mathrm{Aj}}A_h\overline{T}^{\mathrm{Dj}}\right)_{j+\frac{1}{2}} - \frac{\cos\varphi_{j-\frac{1}{2}}}{a^2\cos\varphi_j\Delta^2\varphi}\left(\overline{\sigma_h}^{\mathrm{Aj}}A_h\overline{T}^{\mathrm{Dj}}\right)_{j-\frac{1}{2}} \\ & + \underline{\frac{\overline{\rho}^{k^2}g^2}{p_h}\kappa_v\underline{T}_k}_k \end{aligned} \tag{3.10}$$

It can be proven that the total salt is conserved if the natural boundary conditions is applied. On the other hand, the total potential temperature may be conserved if both the heat flux and fresh water flux at sea surface are equal to zero, or, for the more general case is, if

$$\sum_i \sum_j (F_h - C_p \rho_0 F_{E-P} T_1) a^2 \cos\varphi_j \Delta\lambda \Delta\varphi = 0 \tag{3.11}$$

Furthermore, it can prove that if all the variables are defined at identical time levels and the frictional terms are neglected, the above-given finite-difference equations can guarantee an accurate conversion between the mechanical energy and internal energy.

4 Time Splitting Integration Method

Since the model includes an explicit free surface, we therefore choose to solve the full equations by means of a time-splitting integration method (Zeng) to overcome the time step limit imposed by the gravity wave processes. In other words, we may use a short time step to integrate the geostrophic adjustment processes (mode-I) which involves inertia external gravity wave, while apply a longer time step to the quasi-geostrophic process (mode-II) which involves advection and dissipation. Furthermore, a more longer time step is used to solve the thermodynamic processes (mode-III).

(1) Mode-I: geostrophic adjustment process. It allows for the momentum derivative associated with the pressure gradient and Coriolis forces, as well as the surface pressure derivative associated with the convergence or divergence of the mass flux, while the potential temperature and the salinity are unchanged.

Assume there is no water flux during the fast adjustment process, according to (3.1), the prediction equation for surface pressure of mode-I is

$$[p_h]^{n_b+1} = \overline{[p_h]}^{n_b-1} + 2\Delta t_b \left[-\frac{1}{a\cos\varphi_j} \left(\underline{\overline{PU_B}}^{j}_{i} + \overline{\underline{\overline{PV_B\cos\varphi}}^{Dj}}^{Ai} \right) \right]^{n_b} \tag{4.1}$$

Neglect the advective terms $\mathcal{M}(\mu)$ and the viscous terms $\mathcal{D}$ in the horizontal momentum equations (3.4), we get the prediction equations for horizontal velocity in mode-I

$$[U]^{n_b+1} = \overline{[U]}^{n_b-1} + 2\Delta t_b \left\{ f^* \frac{[V]^{n_b+1} + \overline{[V]}^{nb-1}}{2} - \left[\frac{P}{a\cos\varphi_{jh}} \left(\underline{\overline{\phi}^{j}}_{i} + \overline{\alpha}^{i,j} \underline{\overline{P}^{j}}_{i} \right) \right]^{n_b} \right\}, \tag{4.2a}$$

$$[V]^{n_b+1} = \overline{[V]}^{n_b-1} + 2\Delta t_b \left\{ -f^* \frac{[U]^{n_b+1} + \overline{[U]}^{nb-1}}{2} - \left[\frac{P}{a} \left(\underline{\overline{\phi}^{i}}_{j} + \overline{\alpha}^{i,j} \underline{\overline{P}^{i}}_{j} \right) \right]^{n_b} \right\}, \tag{4.2b}$$

where the superscript n_b denotes the n-th time level of mode-I; Δt_b is the time step; the bar symbol represents temporal filter (Asselin 1972) which would be used to remove the computational mode associated with the standard leapfrog scheme. Here a semi-implicit scheme is adopted to deal with the Coriolis terms.

From the definition of σ in (2.9), we get the diagnostic equation for pressure

$$[p]^{n_b} = \sigma [p_{bt}]^{n_b} + [p_t]^{n_t}, \tag{4.3}$$

where n_t represents the current time levels of mode-III.

Since the calculation of the density is prohibitively expensive, on the other hand, both the potential temperature and salinity are actually held unchanged during the integration of mode-I, therefore, instead of using the standard

density formula, here the density is taken the one calculated in n_t time level by adding a perturbation term due to change of the pressure. Thus the diagnostic equation of density is

$$[\rho]^{n_b} = [\rho(T,S,p)]^{n_t} + \left[\frac{\partial\rho}{\partial p}\right]^{n_t} ([p]^{n_b} - [p]^{n_t}) \tag{4.4}$$

From (2.26), we have the diagnostic equation for geopotential height

$$[\phi]^{n_b} = gz_b + \int_\sigma^1 \left[\frac{p_{bt}}{\rho}\right]^{n_b} d\sigma \tag{4.5}$$

(2) mode-II: advection and dissipation process. Only account the advective terms $\mathcal{M}(\mu)$ and the viscous terms $\mathcal{D}$ in the horizontal momentum equations (3.4), we get the prediction equations for horizontal velocity in mode-II

$$[U]^{n_c+1} = U' + 2\Delta t_c \left([-\mathcal{M}(U)]^{n_c} + P[\mathcal{D}_u]^{n_c-1}\right) \tag{4.6a}$$
$$[V]^{n_c+1} = V' + 2\Delta t_c \left([-\mathcal{M}(V)]^{n_c} + P[\mathcal{D}_v]^{n_c-1}\right) \tag{4.6b}$$

where the superscript n_t denotes the n-th time level of mode-II ; Δt_c is the time step; U' and V' are the intermediate results of U and V produced by integration of mode-I at $(n_c+1)\Delta t_c$ time levels. The velocities used in the advection and dissipation operator are given by

$$[u]^{n_c} = \left[\frac{U}{\sqrt{p_{bt}}}\right]^{n_c} \tag{4.7a}$$
$$[v]^{n_c} = \left[\frac{V}{\sqrt{p_{bt}}}\right]^{n_c} \tag{4.7b}$$

The diagnostic equation for vertical velocity is (see appendix A.6 for derivation)

$$[p_h\dot{\sigma}]^{n_c} = \frac{1}{a\cos\varphi}\int_0^\sigma \left\{\frac{\partial}{\partial\lambda}\left[\sqrt{p_{bt}}(U_b - U)\right] + \frac{\partial}{\partial\varphi}\left[\sqrt{p_{bt}}(V_b - V)\cos\varphi\right]\right\}^{n_c} d\sigma - (1-\sigma)\rho_0 g F_{E-P} \tag{4.8}$$

In addition, the Asselin temporal filter is applied to both U and V at $n_c\Delta t_c$ time levels, i.e.,

$$[\mu]^{n_c} = (1-\alpha_t)[\mu]^{n_c} + \frac{\alpha_t}{2}\left([\mu]^{n_c+1} + [\mu]^{n_c-1}\right) \tag{4.9}$$

where μ represents U and V , α_t is the temporal filter coefficient.

The integrations of mode-I and mode-II are synchronous. For example, when mode-II is integrated from $(n_c-1)\Delta t_c$ time levels to $(n_c+1)\Delta t_c$ time levels on a normal leapfrog time step, mode-I will be therefore integrated $2\Delta t_c/\Delta t_b$ steps, of which the Euler backward scheme is applied to the first time step and the leapfrog scheme applied to the rest.

(3)mode-III: thermodynamic process. The prediction equations for the potential temperature and salinity may be written as

$$[p_{bt}T]^{n_t+1} = [p_{bt}T]^{n_t-1} + 2\Delta t_t \left(-[\nabla\cdot(p_h T)]^{n_t} + [p_{bt}\mathcal{Q}_T]^{n_t-1}\right) \tag{4.10a}$$
$$[p_{bt}S]^{n_t+1} = [p_{bt}S]^{n_t-1} + 2\Delta t_t \left(-[\nabla\cdot(p_h S)]^{n_t} + [p_{bt}\mathcal{Q}_S]^{n_t-1}\right) \tag{4.10b}$$

where Δt_t represents the time step for tracers. The integrations of mode-III may be asynchronous with mode-II if accelerated time steps are applied (Bryan, 1984).

Note that the prognostic variables of mode-III are $p_{bt}T$ and $p_{bt}S$ rather than T and S themselves. Correspondingly, the advection terms in tracers equations are taken the flux form so it is easy to keep the total tracers conserved.

The horizontal advection velocities for tracers may take the time averaged velocities of mode-I, which can satisfy the continuity equation (2.21)

$$-2\Delta t_c[\nabla\cdot(p_h\mathbf{v})]^{n_t}=[p_{bt}]^{n_t+1}-[p_{bt}]^{n_t-1} \tag{4.11}$$

In fact, if we set tracers be constant and discard the mixing terms, the tracers equation will be just the continuity equation.

If the accelerated time steps are applied to the integration of tracers equations, however, it will introduce a minor computational error because $\Delta t_t \neq \Delta t_c$ thereby the continuity equation (2.21) is no longer satisfied.

T and S may be calculated by

$$[T]^{n_t+1}=\frac{[p_{bt}T]^{n_t+1}}{[p_{bt}]^{n_t+1}} \tag{4.12a}$$

$$[S]^{n_t+1}=\frac{[p_{bt}S]^{n_t+1}}{[p_{bt}]^{n_t+1}} \tag{4.12b}$$

And the density will be calculated according to the updated potential temperature, salinity, as well as the pressure

$$[\rho]^{n_t+1}=\rho\left([T]^{n_t+1},[S]^{n_t+1},[p]^{n_t+1}\right) \tag{4.13}$$

5 Some Global Variables in the Code

`adv_vbtiso` - Isopycnal advective velocity on the bottom face of the T cell. `(imt,km,jmt)`, `ihj` grid.

1. Set to 0 in by `call isopyi` in `main.f90`
2. In `isopyc.f90` (by `call isoadv`, in which by `call isow`)

$$\begin{aligned}\texttt{adv\_vbtiso} &= \texttt{w} \\ &= \int_{p_t}^{p}\left(\frac{\partial \mathrm{u}}{a\cos\varphi\partial\lambda}+\frac{\partial \mathrm{v}}{a\cos\varphi\partial\varphi}\right)dp+\frac{1}{p_h}\int_{p_t}^{p}\left[-\int_{p_t}^{p_b}\left(\frac{\partial \mathrm{u}}{a\cos\varphi\partial\lambda}+\frac{\partial \mathrm{v}}{a\cos\varphi\partial\varphi}\right)dp\right]dp\end{aligned} \tag{5.1}$$

`adv_vetiso` - Isopycnal advective velocity on the eastern face of the T cell. `(imt,km,jmt)`, `ihj` grid.

1. Set to 0 in by `call isopyi` in `main.f90`
2. In `isopyc.f90` (by `call isoadv`)

$$\begin{aligned}\texttt{adv\_vetiso} &= -A_I\sigma_h\cos\varphi\frac{\partial K_{13}}{\partial z} \\ &= -\texttt{athkdf}*\texttt{fy}*\texttt{cosu}*\texttt{rdz0}*\overline{\texttt{K1}(:,:,:,3)}^{Dk}\end{aligned} \tag{5.2}$$

`adv_vntiso` - Isopycnal advective velocity on the northern face of the T cell. `(imt,km,jmt)`, `ihj` grid.

1. Set to 0 in by `call isopyi` in `main.f90`
2. In `isopyc.f90` (by `call isoadv`)

$$\begin{aligned}\texttt{adv\_vntiso} &= -A_I\sigma_h\cos\varphi\frac{\partial K_{23}}{\partial z} \\ &= -\texttt{athkdf}*\texttt{fy}*\texttt{cosu}*\texttt{rdz0}*\overline{\texttt{K2}(:,:,:,3)}^{Dk}\end{aligned} \tag{5.3}$$

`am` - A_m, horizontal viscosity for momentum equation, `(imt,jmt,km)`, `ihjhk` grid.

1. Set to `am_c = 1.0e7` in `setcon.f90`.
2. Calculate new `am` by Smagorinsky Scheme in `tracer.f90`

`afb1` - Asselin filter coef. (Asselin 1972). Equal to $\alpha/2$ in 刘海龙, 俞永强, 李 薇, 张学洪 (2003)[P.25].

`afb2` - Asselin filter coef. (Asselin 1972). Equal to $1-\alpha$ in 刘海龙, 俞永强, 李 薇, 张学洪 (2003)[P.25].

`ah` - Lateral diffusivity for tracers, `(imt,jmt,km)`, `ijk` grid.

1. In `setcon.f90`, set to `ah_c` (10^7) from `namelist`.

`ahisop` - Lateral diffusivity for isopycnal mixing. `(imt,jmt,km)`, `ijk` grid.

1. Set to `ah` in `main.f90` (by `call isopyi`).

`athkdf` - Isopycnal thickness diffusivity (刘海龙, 俞永强, 李 薇, 张学洪 2003, P.12), `(imt,jmt,km)`, `ihjhk` grid,

1. Set to 0 in `main.f90` (by `call setcon`).
   NOTE: in `setcon.f90`, just define, not set actually.

`bct` - Observed sea surface temperature, `(imt,jmt)`

1. In `interp.f90`, linearly interped from monthly value of `bcf(:,:,:,3)`

`diffu` - $P\mathcal{D}_u$, viscous term for u-momentum equation, `(imt,jmt,km)`, `ihjhk` grid.

1. Set to 0 in `inirun.f90`
2. In `readyc.f90`:

$$\texttt{diffu} = \frac{1}{\sqrt{\sigma_h}}\nabla_h\cdot(\sigma_h A_m \nabla_h u) + \frac{\partial}{\partial p}\left(\kappa_m \frac{g\rho_0}{p_h}\frac{\partial U}{\partial p}\right) + A_m\left(\frac{1-\tan^2\varphi}{a^2}U - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial\lambda}\right) \tag{5.4a}$$

$$= A_m\left\{\left[\frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(\sigma_h\frac{\partial u}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi\partial\varphi}\left(\sigma_h\cos\varphi\frac{\partial u}{\partial\varphi}\right)\right]\frac{1}{\sqrt{p_h}}\right.$$
$$\left. + \frac{1-\tan^2\varphi}{a^2}U - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial\lambda}\right\} + \frac{\partial}{\partial p}\left(\kappa_m\frac{g\rho_0}{\sigma_h}\frac{\partial U}{\partial z}\right) \tag{5.4b}$$

$$= \texttt{am}\left\{\left[\texttt{sdxu}*\sigma_h\overline{\frac{\partial u}{\partial\lambda}}^{Di} + \texttt{r1c}*2\sigma_h\overline{u}^{Dj+1} - \texttt{r1d}*2\sigma_h\overline{u}^{Dj}\right]/\sqrt{p_h}\right. \tag{5.4c}$$

$$\left. + \texttt{cv1}*\texttt{up} - \texttt{cv2}*\overline{v}^{Di}*\texttt{spbt}\right\} + \texttt{rdz}*\overline{\kappa_m\frac{g\rho_0}{\sigma_h}\frac{\partial U}{\partial z}}^{Dk} \tag{5.4d}$$

Note: (4.3) is differenct that (2.15a) in the second term ($\frac{g\rho}{\sigma_h}$ vs $\frac{g^2\rho^2}{\sigma_h^2}$) may be this is a bug. Also note that $\overline{v}^{Di}$ adopt the leapfrog scheme, so `cv2` do not have to multiply by 2

`diffv` - $P\mathcal{D}_v$, viscous term for v-momentum equation, `(imt,jmt,km)`, `ihjhk` grid.

1. Set to 0 in `inirun.f90`
2. In `readyc.f90`:

$$\texttt{diffv} = \frac{1}{\sqrt{p_h}}\nabla_h\cdot(\sigma_h A_m \nabla_h v) + \frac{\partial}{\partial p}\left(\kappa_m \frac{g\rho_0}{p_h}\frac{\partial V}{\partial p}\right) + A_m\left(\frac{1-\tan^2\varphi}{a^2}V + \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial\lambda}\right) \tag{5.5a}$$

$$= A_m\left\{\left[\frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(\sigma_h\frac{\partial v}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi\partial\varphi}\left(\sigma_h\cos\varphi\frac{\partial v}{\partial\varphi}\right)\right]\frac{1}{\sqrt{p_h}}\right.$$
$$\left. + \frac{1-\tan^2\varphi}{a^2}V - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial\lambda}\right\} + \frac{\partial}{\partial p}\left(\kappa_m\frac{g\rho_0}{\sigma_h}\frac{\partial V}{\partial z}\right) \tag{5.5b}$$

$$= \texttt{am}\left\{\left[\texttt{sdxu}*\sigma_h\overline{\frac{\partial v}{\partial\lambda}}^{Di} + \texttt{r1c}*2\sigma_h\overline{v}^{Dj+1} - \texttt{r1d}*2\sigma_h\overline{v}^{Dj}\right]/\sqrt{p_h}\right. \tag{5.5c}$$

$$\left. + \texttt{cv1}*\texttt{vp} - \texttt{cv2}*\overline{u}^{Di}*\texttt{spbt}\right\} + \texttt{rdz}*\overline{\kappa_m\frac{g\rho_0}{\sigma_h}\frac{\partial V}{\partial z}}^{Dk} \tag{5.5d}$$

`emp` - Fresh water flux, F_{E-P}, evaporation minus precipitation, `(imt,jmt)`, `ij` grid.

1. Set by `bcf(:,:,:,6)` in `interp.f90`

`kref` - Depth of the middle of each leayer, in m, `0.01z0`, `(km)`, `k` grid. In `isopyc.f90`:

```
kref(k) = z0(k)*0.01
```

`kappa_h` - Vertical mixing coefficient for tracers. `(imt,jmt,km)`, `ijk` grid.

1. In `setcon.f90`, read from `vmix.nc` (Zhang, Lin, and Huang 2014).

`leapfrog_b` - (false,true) on a (eular backward, normal leapfrog) time step in mode-I . The "b" indicates the barotropic mode.

1. Set to `.false.` at the beginning of every month cycle in `main.f90`.
2. Set to `.true.` on the first day of the current month in `barotr.f90`.

`leapfrog_c` - (false,true) on a (eular backward, normal leapfrog) time step in mode-II . The "b" indicates the baroclinic mode.

1. Set to `.false.` at the beginning of every month cycle in `main.f90`.
2. Set to `.true.` on the first day of the current month in `bclinc.f90`.

`leapfrog_t` - (false,true) on a (eular backward, normal leapfrog) time step in tracer. The "t" indicates the tracer mode.

1. Set to `.false.` at the beginning of every month cycle in `main.f90`.
2. Set to `.true.` on the first day of the current month in `tracer.f90`.

`euler_back` - (false,true) on a (eular forward, backward) time step in mode-I .

1. Set to `.true.` at the beginning of every month cycle.
2. Set to `.false.` after star-up integration on the first day of the current month in `barotr.f90`.

`ncname` - initial file: input/pcom_ini.nc

`nbb` - "1" baroclinic step = "nbb" barotropic step

`ncc` - "1" tracerstep = "ncc" baroclinic step

`nss` - 1 day = "nss" tracer step

`onbb` = 1/(nbb+1)

`oncc` = 1/(ncc+1)

`onbc` = 1/(nbb*ncc+1)

`rhoi` - Water density ρ for isopycnal mixing. `(imt,km,jmt,1:3)`, `i,j,k` grid.

1. In `main.f90`, set to 1 by `call isopyi`.
2. Calc. in `tracer.f90` (by `call isopyc`):

`up` - U component of horizontal "velocity" $\mathbf{V} = (U, V) = (\sqrt{p_h}u, \sqrt{p_h}v)$, `(imt,jmt,km,2)`, `ihjhk` grid.

1. Set to 0 in `inirun.f90`

2. Prognose in `bclinc.f90` with Euler forward scheme in the first baroclinic time step in every month:

```
up(i,j,k,tau) = up(i,j,k,taum) + du(i,j,k)*dtuv
```

`vp` - V component of horizontal "velocity" $\mathbf{V} = (U, V) = (\sqrt{p_h}u, \sqrt{p_h}v)$, `(imt,jmt,km,2)`, `ihjhk` grid.

1. Set to 0 in `inirun.f90`
2. Prognose in `bclinc.f90` with Euler forward scheme in the first baroclinic time step in every month:

```
vp(i,j,k,tau) =vp(i,j,k,taum) + dv(i,j,k)*dtuv
```

`upb` - Barotropic component of `up`, U_B, `(imt,jmt,2)`, `ihjh` grid.

1. Set to zero in `inirun.f90`
2. Prognose in `barotr.f90`:
   `upb(i,j,tau) = upb(i,j,taum) + du2(i,j)*dtsf`,
   then filter by Asselin filter.

`pt` - $\frac{\partial \rho}{\partial T}$, calc. at `density.f90` of `rho_ref` subroutine

`ps` - $\frac{\partial \rho}{\partial S}$

`secday` - $\frac{1}{24*60*60}$, how many days in 1 second.

`deltap` - 1.0e-2

`deltat` - 1.0e-2

`deltas` - 1.0e-3

`rdeltap` - $\frac{1}{2\texttt{deltap}}$

`rdeltat` - $\frac{1}{2\texttt{deltat}}$

`rdeltas` - $\frac{1}{2\texttt{deltas}}$

`decibar` - change the unit of pressure from $dynes/cm^2$ to decibar 1 dynes/cm**2 = 0.1 N/m**2 = 1.0e-3 mbar = 1.0e-5 decibar, so decibar = 1.0e-5

`gamma_s` - γ_s, parameter for restoring salinity boundary condition.

1. Defined in `param.h`.
2. In `setcon.f90`, set to

$$\gamma_s = \frac{1}{120 \times 24 \times 60 \times 60} \qquad \texttt{gamma\_t} = \texttt{r120} * \texttt{secday} \tag{5.6}$$

`gamma_t` - γ_t, parameter for calculation of surface heat flux, defined in `param.h`.

1. In `setcon.f90`

$$\gamma_t = \frac{g * \mathtt{ddd}}{C_p}$$
$$\mathtt{gamma\_t} = \frac{\mathtt{grav} * 40.0}{3901.0} \times 0.1 \tag{5.7}$$

where $\mathtt{ddd} = 40 W/(m^2 \cdot K)$, is the surface heat flux; $C_p = 3901 J/(kg \cdot K)$, is the specific heat capacity of sea water; and 0.1 is the factor for unit change:

$$\frac{/m^2}{/Kg} = \frac{Kg}{m^2} = 0.1 \frac{g}{cm^2} \tag{5.8}$$

`gravr` - `grav*rho_0` = $g\rho_0$

`rho_0` - reference density of the water, $1.029 g/cm^3$

`grav` - earth's gravitational acceleration, $g = 980 cm/s^2$

`mat_myid` - mat_myid(ncpux+2, ncpuy), 243-251, main.f90. Say, ncpux=8, ncpuy=4, then

```
    7       15       23       31

    0       8        16       24
    1       9        17       25
    2       10       18       26
    3       11       19       27
    4       12       20       28
    5       13       21       29
    6       14       22       30
    7       15       23       31

    0       8        16       24
```

Note that the first and last row is for wrap up

`runtime` - calc. running time using MPI

`sa_first` - ?

`t_stepu` - A countor of baroclinic time steps. The value indicates the NEXT baroclinc time step.

`umn` - Horizontal velocity, u, `(imt,jmt,km)`, `ihjhk` grid.

1. diagnose in `tracer.f90`:

$$\mathtt{umn} = u \tag{5.9a}$$
$$= \frac{\sqrt{p_h} U}{\sigma_h} \tag{5.9b}$$
$$= \frac{\mathtt{ump}}{\mathtt{pbt}} \tag{5.9c}$$

ump - Pressure horizontal velocity, $\sqrt{p_h}U$, $\sqrt{p_h}\sqrt{p_h}u$, (imt,jmt,km), ihjhk grid.

1. Set to 0 in inirun.f90
2. In readyt.f90:

$$\begin{aligned} \texttt{ump} &= U\sqrt{p_h} && (5.10a)\\ &= \sqrt{p_h}u \cdot \sqrt{p_h} && (5.10b)\\ &= \texttt{up} * \texttt{spbt} && (5.10c) \end{aligned}$$

3. Accumulate $U\sqrt{p_h}$ in bclinc.f90:

```
ump(i,j,k) = ump(i,j,k) + up(i,j,k,tau)*spbt(i,j)
```

4. Calculate time-average value of baroclinic values for the current tracer time step in tracer.f90:

```
ump(i,j,k) = ump(i,j,k)*oncc
```

vmn - Horizontal velocity, v, (imt,jmt,km), ihjhk grid.

1. Diagnose in tracer.f90:

$$\begin{aligned} \texttt{vmn} &= v && (5.11a)\\ &= \frac{\sqrt{p_h}V\cos\varphi}{\sigma_h\cos\varphi} && (5.11b)\\ &= \frac{\texttt{vmp}}{\texttt{pbt} * \texttt{cosu}} && (5.11c) \end{aligned}$$

vmp - Pressure horizontal velocity, $\sqrt{p_h}V\cos\varphi$, $\sqrt{p_h}\sqrt{p_h}v\cos\varphi$, (imt,jmt,km), ihjhk grid.

1. Set to 0 in inirun.f90
2. In readyt.f90:

$$\begin{aligned} \texttt{vmp} &= V\sqrt{p_h}\cos\varphi && (5.12a)\\ &= \sqrt{p_h}v \cdot \sqrt{p_h}\cos\varphi && (5.12b)\\ &= \texttt{vp} * \texttt{spbt} * \texttt{cosu} && (5.12c) \end{aligned}$$

3. Accumulate $V\sqrt{p_h}$ in bclinc.f90:

```
vmp(i,j,k) = vmp(i,j,k) + vp(i,j,k,tau)*spbt(i,j)*cosu(j)
```

4. Calculate time-average value of baroclinic values for the current tracer time step in tracer.f90:

```
vmp(i,j,k) = vmp(i,j,k)*oncc
```

umm - Pressure horizontal velocity, $\sqrt{p_h}U$, $\sqrt{p_h}\sqrt{p_h}u$, (imt,jmt,km), ihjhk grid. Storage of ump.

1. Set to 0 in inirun.f90

2. In `readyt.f90`:

$$\texttt{umm} = U\sqrt{p_h} \tag{5.13a}$$
$$= \texttt{ump} \tag{5.13b}$$

`vmm` - Pressure horizontal velocity, $\sqrt{p_h}V\cos\varphi$, $\sqrt{p_h}\sqrt{p_h}v\cos\varphi$, `(imt,jmt,km)`, `ihjhk` grid. Storage of `vmp`.

1. Set to 0 in `inirun.f90`
2. In `readyt.f90`:

$$\texttt{vmm} = V\sqrt{p_h}\cos\varphi \tag{5.14a}$$
$$= \texttt{vmp} \tag{5.14b}$$

`tau` - 1, indicate current time step

`taum` - 2, indicate previous time step

`pmup` - Sea bottom pressure σ_h for solving baroclinic equations, `(imt,jmt)`, `ij` grid.

1. set to `pbt(:,:,tau)` (i.e. 1) in `inirun.f90`
2. set to `pbt(:,:,tau)` in `readyc.f90`
3. Accumulate `pbt(:,:,tau)` in `barotr.f90`:

```
pmup(i,j) = pmup(i,j) + pbt(i,j,tau)
```

4. Calculate time-average value for the current baroclinic time step in `bclinc.f90`:

```
pmup(i,j) = pmup(i,j)*onbb
```

`pmum` - Sea bottom pressure σ_h for solving baroclinic equations, `(imt,jmt)`, `ij` grid.

1. set to `pbt(:,:,tau)` (i.e. 1) in `inirun.f90`
2. set to baroclinic-average σ_h in `readyc.f90`

```
pmum(i,j) = pmup(i,j)
```

`pmn` - Height of sea surface. `(imt,jmt)`, `ij` grid.

1. In `tracer.f90`:

$$\texttt{pmn} = -h + \sigma_h g \int_{p_t}^{p_b} \frac{1}{\rho} dp \tag{5.15a}$$
$$= \texttt{phib/grav} + \texttt{pbar/grav} \sum_{k=1}^{n} \texttt{dz} * \texttt{rho} \tag{5.15b}$$
$$= \texttt{phi} \tag{5.15c}$$

`pmtp` - Normalized sea bottom pressure σ_h for solving tracer equations, `(imt,jmt)`, `ij` grid.

1. Assigned from `pbt(:,:,tau)` (i.e., 1) in `inirun.f90`
2. Assigned from `pbt(:,:,tau)` in `readyt.f90`
3. Accumulate `pbt(:,:,tau)` in `barotr.f90`:

```
pmtp(i,j) = pmtp(i,j) + pbt(i,j,tau)
```

4. Calculate time-average value for the current tracer time step in `tracer.f90`:

```
pmtp(i,j) = pmtp(i,j)*onbc
```

`pmtm` - `(imt,jmt)`, `ij` grid.

1. Assigned from `pbt(:,:,tau)` (i.e., 1) in `inirun.f90`
2. Assigned with average σ_h for tracer step (i.e., `pmtp`) in `readyt.f90`.

`spbt` - $\sqrt{p_h}$, square root of sea bottom pressure (minus sea surface atmospheric pressure), `(imt,jmt)`, `ihjh` grid.

1. set to 1.0 in `inirun.f90`.
2. In `readyt.f90`:

$$\mathtt{spbt} = \sqrt{p_{bt}} \tag{5.16a}$$

$$= \mathtt{p5} * \sqrt{\overline{\mathtt{pbt}}^{Aij}} \tag{5.16b}$$

`pbt` - Normalized sea bottom pressure, σ_h, `(imt,jmt,2)`, last dimension is for time integration, `ijkh` grid.

$$\sigma_h = \frac{p - p_a}{p_0 - p_{a0}} \tag{5.17a}$$

$$= \frac{p - \mathtt{bcp}}{\mathtt{z}} \tag{5.17b}$$

where p is the pressure at the middle of the layer at current time, p_a is atmosphere pressure at current time , `z(k)` is the initial pressure at the middle of the layer (already minus initial atmospheric pressure). So `pbt` is 1 for every layers in the initial state, and fluctuate with time. And we have

$$p(t) = \mathtt{pbt(t)z(k)} + p_a \tag{5.18}$$

1. `pbt(:,:,tau)` set to 1 in `inirun.f90` (`call setpbt`)
2. `pbt(:,:,taum)` set to `pbt(:,:,tau)` (`c1`) in `inirun.f90`
3. Prognose in `barotr.f90`:
   `pbt(i,j,tau) = pbt(i,j,taum) + dp(i,j) *dtsf`,
   then filter by Asselin filter, and switch `tau` and `taum` time levels.

`pbt_st` - Set to 1 in `inirun.f90`

`pbxn` - Barotropic compoent of the zonal gradient of sea bottom geopotential height (see Appendix E)

1. Set to 0 in `inirun.f90`
2. Calculate in `readyt.f90` (`call prsgrd`):
   (a) set to the vertical integration of $-\phi_b$ (by `call vinteg_ns(...)`)

$$\texttt{pbxn} = \int_{\sigma}^{1}(-\phi_b)d\sigma \tag{5.19a}$$
$$= \int_{\sigma}^{1}\left(\int_{p}^{p_b}\frac{1}{\rho}dp + \frac{p_0}{\rho}\right)d\sigma \tag{5.19b}$$
$$= \frac{1}{p_{h0}}\int_{p_t}^{p_b}\left(\int_{p}^{p_b}\frac{1}{\rho}dp + \frac{p_0}{\rho}\right)dp \tag{5.19c}$$
$$= \frac{1}{p_{h0}}\int_{p_t}^{p_b}\left(\overline{\texttt{rhodp}}^{Ai} + \overline{\texttt{rho}}^{Ai}p\right)dp \tag{5.19d}$$
$$= \texttt{rzu}*\sum_{k=1}^{n}\left(\overline{\texttt{rhodp(i,j,k)}}^{Ai} + \overline{\texttt{rho(i,j,k)}}^{Ai}p\right)\texttt{dz(k)} \tag{5.19e}$$
$$\tag{5.19f}$$

   (b) Multiplied by a x-direction differential factor $\frac{1}{a\cos\varphi\Delta\lambda}$

$$\texttt{pbxn} = \frac{1}{a\cos\varphi\Delta\lambda}\frac{1}{p_{h0}}\int_{p_t}^{p_b}(-\phi_b)dp \tag{5.20a}$$
$$= \texttt{rdxt}*\texttt{pbxn} \tag{5.20b}$$

`pbxs` - The same as `pbxn`, except vertically integrated at the at `j-1` grid. (i.e., the south neighbor) when `call vinteg_ns`

`pbye` - Barotropic compoent of the meridional gradient of sea bottom geopotential height, $\frac{1}{a\Delta\varphi}\frac{1}{p_{h0}}\int_{p_t}^{p_b}(-\phi_b)dp$, `(imt,jmt),ihj` grid.

1. Set to 0 in `inirun.f90`
2. Calculate in `readyt.f90` (`call prsgrd`):
   (a) set to the vertical integration of $-\phi_b$ by (`call vinteg_ew(...)`):

$$\texttt{pbye} = \frac{1}{p_{h0}}\int_{p_t}^{p_b}(-\phi_b)dp \tag{5.21a}$$
$$= \int_{p_t}^{p_b}\left(\overline{\texttt{rhodp}}^{Aj} + \overline{\texttt{rho}}^{Aj}p\right)dp \tag{5.21b}$$
$$= \frac{1}{p_{h0}}\sum_{k=1}^{n}\left(\overline{\texttt{rhodp(i,j,k)}}^{Aj} + \overline{\texttt{rho(i,j,k)}}^{Aj}*\texttt{z(k)}\right)\texttt{dz(k)} \tag{5.21c}$$
$$\tag{5.21d}$$

   (b) Multiplied by a factor of $\frac{1}{a\Delta\varphi}$

$$\texttt{pbye} = \frac{1}{a\Delta\varphi}\frac{1}{p_{h0}}\int_{p_t}^{p_b}(-\phi_b)dp \tag{5.22a}$$
$$= \texttt{rdy}*\texttt{pbye} \tag{5.22b}$$

pbyw - The same as pbye, except vertically integrated at the at i-1 grid. (i.e., the west neighbor) when call vinteg_ew

pcxn - Vertical integrated of zonal gradient of potential height difference, (imt,jmt),ihj grid. $\int_\sigma^1 \frac{\partial}{\partial a\cos\varphi\partial\lambda}(\phi-\phi_b)d\sigma$. From the definition of pbxn, we have $\phi-\phi_b=\int_p^{p_b}\frac{1}{\rho}dp$

1. Set to 0 in inirun.f90
2. Calculate in readyt.f90 (call prsgrd):
   (a) set to the vertical integration of the gradient of $\int_p^{p_b}\frac{1}{\rho}dp$ (rhodp) (call vinteg_ns(...)):

$$\texttt{pcxn}=\int_\sigma^1\left[\frac{\partial}{a\cos\varphi\partial\lambda}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)\right]d\sigma \tag{5.23a}$$

$$=\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left[\frac{\partial}{a\cos\varphi\partial\lambda}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)\right]dp \tag{5.23b}$$

$$=\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left(\frac{1}{a\cos\varphi\Delta\lambda}*\overline{\int_p^{p_b}\frac{1}{\rho}dp}^{Di}\right)dp \tag{5.23c}$$

$$=\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left(\texttt{rdxt}*\overline{rhodp}^{Di}\right)dp \tag{5.23d}$$

$$=\texttt{rzu}\sum_{k=1}^{n}\left(\texttt{rdxt}*\overline{rhodp}^{Di}\right)\texttt{dz(k)} \tag{5.23e}$$

   (b) Multiplied by a factor of 0.5:

$$\texttt{pcxn}=\frac{1}{2}\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left[\frac{\partial}{a\cos\varphi\partial\lambda}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)\right]dp \tag{5.24a}$$

$$=\texttt{p5}*\texttt{pcxn} \tag{5.24b}$$

pcxs - The same as pcxn, except vertically integrated at the at j-1 grid. (i.e., the south neighbor) when call vinteg_ns

pcye - Vertical integration of meridional gradient of $\phi-\phi_b$, i.e., $\int_\sigma^1\frac{\partial}{a\partial\varphi}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)d\sigma$. (imt,jmt),ihj grid

1. Set to 0 in inirun.f90
2. Calculate in readyt.f90 (call prsgrd):
   (a) set to the vertical integration of the gradient of $\int_p^{p_b}\frac{1}{\rho}dp$ (rhodp) (call vinteg_ew(...)):

$$\texttt{pcye}=\int_\sigma^1\left[\frac{\partial}{a\partial\varphi}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)\right]d\sigma \tag{5.25a}$$

$$=\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left[\frac{\partial}{a\partial\varphi}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)\right]dp \tag{5.25b}$$

$$=\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left(\frac{1}{a\Delta\varphi}*\overline{\int_p^{p_b}\frac{1}{\rho}dp}^{Dj}\right)dp \tag{5.25c}$$

$$=\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left(\texttt{rdy}*\overline{rhodp}^{Dj}\right)dp \tag{5.25d}$$

$$=\texttt{rzu}\sum_{k=1}^{n}\left(\texttt{rdy}*\overline{rhodp}^{Dj}\right)\texttt{dz(k)} \tag{5.25e}$$

(b) Multiplied by a factor of 0.5:

$$\texttt{pcye} = \frac{1}{2}\frac{1}{p_{h0}}\int_{p_t}^{p_b}\left[\frac{\partial}{a\partial\varphi}\left(\int_p^{p_b}\frac{1}{\rho}dp\right)\right]dp \tag{5.26a}$$

$$= \texttt{p5} * \texttt{pcye} \tag{5.26b}$$

`pcyw` - The same as `pcye`, except vertically integrated at the at `i-1` grid. (i.e., the west neighbor) when `call vinteg_ew`

`rho` - `(imt,jmt,km)`, `ijk` grid

1. ρ, initial density calculated from initial field in `inirun.f90` (`call setpbt`), g/cm^3
2. $1/\rho$, set to reciprocal of density in `readyt.f90` (`call density`).
3. Update to $1/rho$ again in `tracer.f90` (`unrdens` function)

`rhodp` - $(\phi - \phi_b)/\sigma_h$, indefinite integration of $(1/\rho)dp$, `(imt,jmt,km)`, `ijkh` grid.

1. Set to 0 in `inirun.f90`
2. Calculate in `readyt.f90` (`call prsgrd`):

$$\texttt{rhodp} = \frac{\phi - \phi_b}{\sigma_h} \tag{5.27a}$$

$$= \frac{1}{\sigma_h}\int_p^{p_b}\frac{1}{\rho}dp \tag{5.27b}$$

$$= \frac{1}{\sigma_h}\int_{p_t}^{p_b}\frac{1}{\rho}dp - \frac{1}{\sigma_h}\int_{p_t}^{p}\frac{1}{\rho}dp \tag{5.27c}$$

$$= \sum_{k=1}^{n}\texttt{rho(i,j,k)}dz(k) - \overline{\sum_{k=1}^{n}\texttt{rho(i,j,k)}dz(k)}^{Ak} \tag{5.27d}$$

where n is the number of water layers in `ij` grid (`itn(i,j)`)

`t` - tracers(potential temperature, salinity), in two time steps. 5D, the last 2 dimension is both 2.

- When read in the initial field, t(:,:,:,1,1) = t(:,:,:,1,2) store temperature, t(:,:,:,2,1) = t(:,:,:,2,2) store salinity.

`area` - global sea area, for "T" grid

$$area = \iint a^2\cos\varphi\underline{\textbf{tmask}(\lambda,\varphi)}d\lambda d\varphi \tag{5.28}$$

`acfl` - Maximum viscosity for momentum equation, used in the Smagrorinsky scheme in `trcer.f90`, `(jmt)`, `jh` grid.

1. In `grdvar.f90`

$$\texttt{acfl} = \frac{(a\cos\varphi\Delta\lambda)^2 + (a\Delta\varphi)^2}{8\Delta t_{ts}} \tag{5.29a}$$

$$= \frac{(1/\texttt{rdxu(j)})**2 + (1/\texttt{rdy})**2}{8 * \texttt{dtts}} \tag{5.29b}$$

where Δt_{ts} is time step for tracers.

`bcp` - Surface pressure, boundary condition forcing, `(imt,jmt)`, `ij` grid. dyn/cm^2

1. In `interp.f90`, linearly interpolate from monthly data to daily value.

`bcf` - Surface forcing fields, `(imt,jmt,12,7)`, `ihjhk` grid.

```
!     bcf(i,j,12,1) :   sea surface zonal wind stress      (dynes/cm**2)
!    bcf(i,j,12,2) :  sea surface meridional wind stress  (dynes/cm**2)
 !      bcf(i,j,12,3) :   sea surface air temperature        (Celsius)
!     bcf(i,j,12,4) :  sea surface air pressure           (dynes/cm**2)
 !      bcf(i,j,12,5) :   sea surface salinity               (psu)
!      bcf(i,j,12,6) :  rate of evaporation minus precipitation (cm/s)
!    bcf(i,j,12,7) :  coefficient for calculation of Heat Flux (w/m2/c)
!    bcf(i,j,12,7) :  heat flux, positive is into ocean (w/m^2) (overwritten)
```

1. Read from `input/pcom_bcf.nc` in `inirun.f90`

`bottom_h` - sea bottom height relative to the deepest sea in the model, positive

`cost` - $\cos\varphi$, for "T" grid

`cosu` - $\cos\varphi$, for "U" grid

`cv1` - metric term, for "U" grid

$$cv1(\varphi) = \frac{1-\tan^2\varphi}{a^2} \tag{5.30}$$

`cv2` - Distance factor, `(imt)`, `jh` grid. In `grdvar.f90`:

$$\texttt{cv2} = \frac{\tan\varphi}{a^2\cos\varphi\Delta\lambda} \tag{5.31a}$$

$$= \frac{\tan\varphi}{*}\texttt{rdxu} * \texttt{radius} \tag{5.31b}$$

`ddd`

$$ddd(\lambda,\varphi) = g\frac{H(\lambda,\varphi)}{C_p}10^{-1} \tag{5.32}$$

where $g = 980cm/s^2$, $H(\lambda,\varphi)$ is heat flux into the ocean, read from forcing file(in W/m^2), and C_p is specific heat capacity of sea water, $3901J/kg/K$, 10^{-1} is unit change (change H to W/cm^2 and C_p to $J/g/K$)

`du` - Horizontal advection + surface pressure gradient + viscosity terms, `(imt,jmt,km)`, `ihjhk` grid.

1. Set to zero in `inirun.f90`
2. In `readyc.f90`:

$$\texttt{du} = -\texttt{ux} - \sqrt{p_h}\frac{\partial p_a}{\rho_0 a\cos\varphi\partial\lambda} + \texttt{diffu}$$

$$= -\texttt{ux} - \texttt{pax} * \texttt{spbt} + \texttt{diffu} \tag{5.33a}$$

where:

$$\texttt{ux} = \frac{\partial(Uu)}{a\cos\varphi\partial\lambda} - \frac{U}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Uv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{U}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} + \frac{\partial(Uw)}{\partial p} - \frac{U}{2}\frac{\partial w}{\partial p} \tag{5.34}$$

$$\texttt{diffu} = \frac{1}{\sqrt{\sigma_h}}\nabla_h\cdot(\sigma_h A_m\nabla_h u) + \frac{\partial}{\partial p}\left(\kappa_m\frac{g\rho_0}{p_h}\frac{\partial U}{\partial p}\right) + A_m\left(\frac{1-\tan^2\varphi}{a^2}U - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial\lambda}\right) \tag{5.35}$$

3. In `bclinc.f90`, represent the whole velocity tendency, and use in Coriolis adjust process. After that, it is used a storage for other variables, like temporally store velocity U, V.

`dv` - `(imt,jmt,km)`, `ihjhk` grid.

1. Set to zero in `inirun.f90`
2. In `readyc.f90`:

$$\begin{aligned}\texttt{dv} &= -\texttt{vx} - \sqrt{p_h}\frac{\partial p_a}{\rho_0 a\partial\varphi} + \texttt{diffv}\\ &= -\texttt{vx} - \texttt{pay}*\texttt{spbt} + \texttt{diffv}\end{aligned} \tag{5.36a}$$

where:

$$\texttt{vx} = \frac{\partial(Vu)}{a\cos\varphi\partial\lambda} - \frac{V}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Vv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{V}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} + \frac{\partial(Vw)}{\partial p} - \frac{V}{2}\frac{\partial w}{\partial p} \tag{5.37}$$

$$\texttt{diffv} = \frac{1}{\sqrt{p_h}}\nabla_h\cdot(\sigma_h A_m\nabla_h v) + \frac{\partial}{\partial p}\left(\kappa_m\frac{g\rho_0}{p_h}\frac{\partial V}{\partial p}\right) + A_m\left(\frac{1-\tan^2\varphi}{a^2}V + \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial\lambda}\right) \tag{5.38}$$

3. In `bclinc.f90`, represent the whole velocity tendency, and use in Coriolis adjust process.

`dub` - Barotropic component of `du`, `(imt,jmt)`, `ihjh` grid.

1. Set to 0 in `inirun.f90`
2. In `readyc.f90` (`call vinteg`):

$$\texttt{dub} = \frac{1}{p_{bt}}\int_{p_t}^{p_b}\texttt{du}dp \tag{5.39a}$$

where

$$\texttt{du} = -P\mathcal{M}(u) - P\frac{\partial p_a}{\rho_0 a\cos\varphi\partial\lambda} + P\mathcal{D}_u \tag{5.40a}$$

$$\begin{aligned}&= -\left[\frac{\partial(Uu)}{a\cos\varphi\partial\lambda} - \frac{U}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Uv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{U}{2}\frac{\partial v\cos\varphi}{a\cos\varphi\partial\varphi} + \frac{\partial(Uw)}{\partial p} - \frac{U}{2}\frac{\partial w}{\partial p}\right]\\ &- \texttt{pax}*\texttt{spbt} + \texttt{diffu}\end{aligned} \tag{5.40b}$$

3. In `barotr.f90`, at the first barotropic time step:

$$\texttt{dub} = \texttt{dub} - \texttt{adu} \tag{5.41a}$$

$$= \frac{1}{p_h}\int_{p_t}^{p_b}\left[-P\mathcal{M}(u) - P\frac{\partial p_a}{\rho_0 a\cos\varphi\partial\lambda} + P\mathcal{D}_u\right]dp - \nabla_h\cdot(A_m\nabla_h U_B) \tag{5.41b}$$

`dxdyt` - for "T" grid

$$dxdyt(\varphi) = a^2 \cos\varphi d\lambda d\varphi \tag{5.42}$$

`dxdyu` - analogous to **dxdyt**, but for "U" grid

`dz` - Initial pressure of increment for each layer, $\Delta p_0, \Delta p/\sigma_h, \rho g \Delta z, pre(k+1) - pre(k)$, positive, `(km)`, `k` grid.

1. Set to 0 at `main.f90`
2. Set to $\overline{p}^{Dk}$, `pre(k+1)-pre(k)` in `grdvar.f90` (`call setpn`)

Note that $\Delta p = \sigma_h \Delta p_0$, so vertical integration by Δp can be convert to Δp_0. And this is how PCOM calculate vertical integration.

`dz0` - thickness of each layer, `(km)`, `k` grid. Δz.

1. Set to 0 in `main.f90`.
2. Set to Δz, i.e., thicknesses of each layers, in `grdvar.f90` (by `call setpn`).

`e` - For isopycnal mixing, `(imt,kmp1,jmt,3)`, `ijkh` grid.

1. Set to 0 in `main.f90` (by `call isopyi`)

`ebea` - Coriolis adjustment coefficient when using Euler-back time scheme, `(jmt)`, `jh` grid. In `inirun.f90`:

$$\texttt{ebea} = \frac{1}{1 + (\Omega \sin\varphi \Delta t_b)^2} \tag{5.43a}$$

$$= \frac{\texttt{c1}}{\texttt{c1} + (\texttt{p5} * \texttt{ff} * \texttt{dtsf})^2} \tag{5.43b}$$

`ebeb` - Coriolis adjustment coefficient when using Euler-back time scheme, `(jmt)`, `jh` grid. In `inirun.f90`:

$$\texttt{ebeb} = \frac{\Omega \sin\varphi \Delta t_b}{1 + (\Omega \sin\varphi \Delta t_b)^2} \tag{5.44a}$$

$$= \frac{\texttt{p5} * \texttt{ff} * \texttt{dtsf}}{\texttt{c1} + (\texttt{p5} * \texttt{ff} * \texttt{dtsf})^2} \tag{5.44b}$$

`ebla` - Coriolis adjustment coefficient when using leapfrog time scheme, `(jmt)`, `jh` grid. In `inirun.f90`:

$$\texttt{ebla} = \frac{1}{1 + (2\Omega \sin\varphi \Delta t_b)^2} \tag{5.45a}$$

$$= \frac{1}{1 + (\texttt{ff(j)} * \texttt{p5} * \texttt{d2dtsf})^2} \tag{5.45b}$$

`eblb` - Coriolis adjustment coefficient when using leapfrog time scheme, `(jmt)`, `jh` grid. In `inirun.f90`: In `inirun.f90`:

$$\texttt{ebla} = \frac{2\Omega \sin\varphi \Delta t_b}{1 + (2\Omega \sin\varphi \Delta t_b)^2} \tag{5.46a}$$

$$= \frac{\texttt{ff(j)} * \texttt{p5} * \texttt{d2dtsf}}{1 + (\texttt{ff(j)} * \texttt{p5} * \texttt{d2dtsf})^2} \tag{5.46b}$$

`epea` - parameter used for semi-implicitly handle of Coriolis term, at jh-grid. And epeb, ..., eblb are all at jh-grid.

$$\texttt{epea}_{j+\frac{1}{2}} = \frac{1}{1 + (\Omega \sin \varphi_{j+\frac{1}{2}} \Delta t_c)^2} \tag{5.47}$$

`epeb`

$$\texttt{epeb}_{j+\frac{1}{2}} = \frac{\Omega \sin \varphi_{j+\frac{1}{2}} \Delta t_c}{1 + (\Omega \sin \varphi_{j+\frac{1}{2}} \Delta t_c)^2} \tag{5.48}$$

`epla`

$$\texttt{epla}_{j+\frac{1}{2}} = \frac{1}{1 + (2\Omega \sin \varphi_{j+\frac{1}{2}} \Delta t_c)^2} \tag{5.49}$$

`eplb`

$$\texttt{eplb}_{j+\frac{1}{2}} = \frac{2\Omega \sin \varphi_{j+\frac{1}{2}} \Delta t_c}{1 + (2\Omega \sin \varphi_{j+\frac{1}{2}} \Delta t_c)^2} \tag{5.50}$$

`ff` - Coriolis term, `(jmt)`, `jh` grid. In `grdvar.f90`:

$$\texttt{ff} = 2\Omega \sin \varphi \tag{5.51a}$$

$$= 2 * \texttt{omega} * \sin(\texttt{lat(j)} * \texttt{torad}) \tag{5.51b}$$

`gr_mass` - Grid-mass, `(imt,jmt,km)`, `ijk` grid. $\Delta p/g, \rho \Delta z$

1. Set to zero in `main.f90`.
2. In `tracer.f90`:

$$\texttt{gr}_{\texttt{m}}\texttt{ass} = \frac{\sigma_h \Delta p}{g} \tag{5.52a}$$

$$= \frac{\texttt{dz} * \texttt{pbar}}{\texttt{grav}} \tag{5.52b}$$

`h` - depth of the sea bottom on ijk cell, positive, in "cm"

`itn` - get number of layers for each ijk grid

`lat` - latitude, jh-grid

$$\texttt{lat}_{j+\frac{1}{2}} = \text{read in from NetCDF file} \tag{5.53}$$

`lon` - longitude, j-grid

$$\texttt{lon}_i = \text{read in from NetCDF file} \tag{5.54}$$

`p` - Sea water pressure, p. `(imt,jmt,km)`, `ijkh` grid.

Set to zero in `main.f90`.

Diagnose in `tracer.f90`:

$$\texttt{p} = \sigma_h(p_0 - p_{a0}) + p_a \tag{5.55a}$$

$$= \texttt{par} * \texttt{z} + \texttt{bcp} \tag{5.55b}$$

See σ_h for more detailed information.

`pax` - Zonal gradient of sea surface air pressure, `(imt,jmt)`, `ihjh` grid.

1. Set to 0 in `inirun.f90`
2. In `readyt.f90`(`call prsgrd`):

$$\texttt{pax} = \frac{1}{\rho_0} \cdot \frac{\partial p_a}{a \cos\varphi \partial\lambda} \tag{5.56a}$$

$$= \texttt{rrho\_0} * \overline{\overline{\texttt{bcp}}^{Di} * \texttt{rdxt}}^{Aj} \tag{5.56b}$$

`pay` - Meridional gradient of air pressure, `(imt,jmt)`, `ihjh` grid.

1. Set to 0 in `inirun.f90`
2. In `readyt.f90` (`call prsgrd`):

$$\texttt{pay} = \frac{1}{\rho_0} \cdot \frac{\partial p_a}{a \partial\varphi} \tag{5.57a}$$

$$= \texttt{rrho\_0} * \overline{\overline{\texttt{bcp}}^{Dj} * \texttt{rdxt}}^{Ai} \tag{5.57b}$$

`phib` - ϕ_{b0}, Initial geopotential height at sea bottom (when the sea surface height is zero), `(imt,jmt)`, `ij` grid

1. In `grdvar.f90`:

$$\texttt{phib}_{i,j} = -gh \tag{5.58}$$

where h is the depth of the sea bottom, positive, in cm, and $h = 0$ at land.

`phibx` - Partial difference of sea bottom geopotential height ϕ_b (`phib`) at east-west direction, `(imt,jmt)`, `ihjh` grid. In `inirun.f90`:

$$\texttt{phibx} = 2\frac{\partial \phi_b}{a \cos\varphi \partial\lambda} \tag{5.59a}$$

$$= 2\overline{\overline{\texttt{phib} * \texttt{rdxt}}^{Di}}^{Aj} \tag{5.59b}$$

`phiby` - Partial difference of `phib` at north-south direction, `(imt,jmt)`, `ihjh` grid. In `inirun.f90`:

$$\texttt{phiby} = 2\frac{\partial \phi_b}{a \partial\varphi} \tag{5.60a}$$

$$= 2\overline{\overline{\texttt{phib} * \texttt{rdy}}^{Dj}}^{Ai} \tag{5.60b}$$

`pn` - p_{h0}, sea bottom pressure of the initial T-S field, `(imt,jmt)`, `ijkh` grid, in $dyne/cm^2$, for "T" grid

$$\texttt{pn}_{i,j} = \rho g h \tag{5.61}$$

h is the sea bottom depth, positive. Notice that the above formula has excluded the atmospheric pressure already. (Since it only acount contribution from the warter column)

1. In `grdvar.f90`, set to sea bottom pressure $\rho g h$ of the initial field at sea grid and 1 at land (`call setpn`)

r1a

$$r1a(j) = \frac{1}{(a\Delta\varphi)^2} \cdot \frac{\cos\varphi_{jh}}{\cos\varphi_j} \quad \text{for (j>1),} \tag{5.62a}$$

$$r1a(1) = r1a(2) \tag{5.62b}$$

r1b

$$r1b(j) = \frac{1}{(a\Delta\varphi)^2} \cdot \frac{\cos\varphi_{j-\frac{1}{2}}}{\cos\varphi_j} \quad \text{for (j>1),} \tag{5.63a}$$

$$r1b(1) = r1a(2) \tag{5.63b}$$

r1c Distance factor, (jmt), jh grid.

1. In grdvar.f90

$$\texttt{r1c} = \frac{1}{2}\frac{1}{(a\Delta\varphi)^2} \tag{5.64a}$$

$$= \frac{1}{2}\texttt{rdy}^2 \tag{5.64b}$$

$$= \frac{1}{2}\frac{\cos\varphi_{(j+1)}}{\cos\varphi_{jh}}\texttt{rdy}^2 \tag{5.64c}$$

$$= \frac{1}{2}\frac{\texttt{cost(j+1)}}{\texttt{cosuj}}\texttt{rdy}^2 \tag{5.64d}$$

r1d Distance factor, the same as r1c, but for the south neighbor grid, i.e., r1c is at j+1 grid, and r1d is at j grid, (jmt).

1. In grdvar.f90

$$\texttt{r1d} = \frac{1}{2}\frac{1}{(a\Delta\varphi)^2} \tag{5.65a}$$

$$= \frac{1}{2}\texttt{rdy}^2 \tag{5.65b}$$

$$= \frac{1}{2}\frac{\cos\varphi_j}{\cos\varphi_{jh}}\texttt{rdy}^2 \tag{5.65c}$$

$$= \frac{1}{2}\frac{\texttt{cost(j)}}{\texttt{cosuj}}\texttt{rdy}^2 \tag{5.65d}$$

rdx

$$rdx(\lambda) = \frac{1}{a\Delta\lambda} \tag{5.66}$$

rdxt - in Eq.3.1 of

$$\texttt{rdxt}_j = \frac{1}{a\cos\varphi_j\Delta\lambda} \tag{5.67}$$

rdxu - in Eq.3.4a of

$$\texttt{rdxu}_{j+\frac{1}{2}} = \frac{1}{a\cos\varphi_{j+\frac{1}{2}}\Delta\lambda} \tag{5.68}$$

rdy

$$\texttt{rdy}_j = \frac{1}{a\Delta\varphi} \tag{5.69}$$

`rdyt` - in Eq.3.1 of

$$\texttt{rdyt}_j = \frac{1}{a\cos\varphi_j \Delta\varphi} \tag{5.70}$$

`rdyu` - in Eq.3.5 of

$$\texttt{rdyu}_{j+\frac{1}{2}} = \frac{1}{a\cos\varphi_{j+\frac{1}{2}} \Delta\varphi} \tag{5.71}$$

`rdz` - Reciprocal of pressure increament, $1/(dz), 1/\Delta p, 1/\rho g \Delta z, 1/[pre(k+1) - pre(k)]$, `(km)`, `k` grid.

1. Set to 0 in `main.f90`
2. Set to $1/dz$ in `grdvar.f90`

`rdz0` - Reciprocal of depths increament, `(km)`, `k` grid. $1/(dz0), 1/\Delta z$.

1. Set to $1/$`dz0` in `main.f90` (by `call isopyi`)

`rdzw` - 1/(distance of the neighbor middle layers), `(km)`, `k` grid

1. Set to 0 in `main.f90`
2. In `grdvar.f90`:

$$\texttt{rdzw} = \frac{1}{\texttt{z\_0(k+1)} - \texttt{z\_0k}} \tag{5.72}$$

`rzu` - Reciprocal of initial sea bottom pressure, `(imt,jmt)`, `ihjh` grid

$$\texttt{rzu} = \frac{1}{p_{h0}} = \frac{1}{\texttt{zu}} = \frac{1}{\rho g h} \tag{5.73}$$

h is possitive. Set to $1/$`zu` in `grdvar.f90` (0 at land)

`slmxr` - Maximum slope in GM scheme.

1. Defined in `param.h`
2. Set to 100 in `main.f90` (by `call isopyi`)

`sdxt` - in Eq.3.9 of

$$\texttt{sdxt}_j = \frac{1}{a^2\cos^2\varphi_j \Delta^2\lambda} \tag{5.74}$$

`sdxu` - Distance factor, `(jmt)`, `jh` grid,

$$\texttt{sdxu}_{j+\frac{1}{2}} = \frac{1}{2a^2\cos^2\varphi_{j+\frac{1}{2}} \Delta^2\lambda} \tag{5.75}$$

1. In `grdvar.f90`:

$$\texttt{sdxu} = \frac{1}{2}\frac{1}{a^2\cos^2\varphi_{j+\frac{1}{2}} \Delta^2\lambda} \tag{5.76a}$$

$$= \texttt{rdxu} * \texttt{rdxu} * \texttt{p5} \tag{5.76b}$$

`tmask`

$$tmask(\lambda,\varphi) = \begin{cases} 1 & \text{, if "T" grid is water} \\ 0 & \text{, if "T" grid is land} \end{cases} \tag{5.77}$$

`umask` - analogous to **tmask**, but for "U" grid

`w` - Vertical velocity w. `(imt,jmt,kmp1)`, `ihjhkh` grid.

1. Set to 0 in `inirun.f90`
2. In `readyc.f90`:
   (a) Diagnose by `call upwelling`:
   $$\mathtt{w} = \int_{p_t}^{p} \left[\frac{1}{p_b} \int_{p_b}^{p_t} \left(\frac{\partial(\sqrt{\sigma_h} U)}{a\cos\varphi \partial\lambda} + \frac{\partial(\sqrt{\sigma_h} V\cos\varphi)}{a\cos\varphi\partial\varphi} \right) dp - F_{E-P} + \frac{\partial(\sqrt{\sigma_h} U)}{a\cos\varphi\partial\lambda} + \frac{\partial(\sqrt{\sigma_h} V\cos\varphi)}{a\cos\varphi\partial\varphi} \right] dp \tag{5.78}$$
   (b) Interpolate from `ij` in last step grid to `ihjh` grid:
   $$\mathtt{w} = \overline{w/\sigma_h}^{Aij} \tag{5.79}$$
3. In `tracer.f90`, Diagnose by `call upwelling` (the same form as in `readyc.f90`):
   $$\mathtt{w} = \int_{p_t}^{p} \left[\frac{1}{p_b} \int_{p_b}^{p_t} \left(\frac{\partial(\sqrt{\sigma_h} U)}{a\cos\varphi \partial\lambda} + \frac{\partial(\sqrt{\sigma_h} V\cos\varphi)}{a\cos\varphi\partial\varphi} \right) dp - F_{E-P} + \frac{\partial(\sqrt{\sigma_h} U)}{a\cos\varphi\partial\lambda} + \frac{\partial(\sqrt{\sigma_h} V\cos\varphi)}{a\cos\varphi\partial\varphi} \right] dp \tag{5.80}$$

`wmn` - Vertical velocity, w, `(imt,jmt,km)`, `ihjhk` grid. (Note that `wmn` is mainly for output, so it lies in the `k` grid, not `kh` grid)

1. Diagnose in `tracer.f90`:
   $$\mathtt{wmn} = \frac{\int_{p_t}^{p} \left[\frac{1}{p_b} \int_{p_b}^{p_t} \left(\frac{\partial(\sqrt{\sigma_h} U)}{a\cos\varphi \partial\lambda} + \frac{\partial(\sqrt{\sigma_h} V\cos\varphi)}{a\cos\varphi\partial\varphi} \right) dp - F_{E-P} + \frac{\partial(\sqrt{\sigma_h} U)}{a\cos\varphi\partial\lambda} + \frac{\partial(\sqrt{\sigma_h} V\cos\varphi)}{a\cos\varphi\partial\varphi} \right] dp}{\rho\sigma_h g} \tag{5.81a}$$
   $$= \frac{\mathtt{w} * \mathtt{rho}}{\mathtt{pbar} * \mathtt{grav}} \tag{5.81b}$$

`z` - Initial pressure p_0 at the middle of each layer, dyn/cm^2, positive, `km`, `k` grid.

1. Set to 0 in `main.f90`
2. Set to the pressure of the center layer in `grdvar.f90` (`call setpn`)
   $$\mathtt{z} = \rho g h \tag{5.82a}$$
   $$= \sum_{k=1}^{n} \mathtt{denz(k)} * \mathtt{grav} * \mathtt{dz0(k)} \tag{5.82b}$$
   where `denz(k)` is calculated by recursive method.

Note that p_0 `z` is only calculated once, and has removed the atmopheric pressure at the sea level.

`z0` - depth of the middle of each layer, in 'cm', `(km)`, `k` grid.

1. Read in from `pcom_ini.nc` in `grdvar.f90` by `call netcdf_read_cor`, then multiply by 100 to convert from m to cm.

`zb` - depth of the bottom of each layer, determined by **z0**

`zu` - p_{h0}, sea bottom pressure, `(imt,jmt)`, `ihjh` grid, similar to `pn`, but at U grid. Set $\sum \Delta p(k)$ in `grdvar.f90` for U-grid.

6 Variables in the Output Files

6.1 Sea Surface Height

Variable name in the monthly output file: `ssh`
Variable definition in `pconst.h`, `pname, p_units, p_longname`, note that `p_longname` is not correct in `pconst.h`

1. `ssh` is written into the output in `diag.f90` through the variable `pzhy`.
2. `pzhy` is assigned by `spsi` with a factor of $1/100$, to get rid of the halo longitude, and change back to "m" units.
3. `spsi` is actually the anomaly to the global mean (area weighted) average value.
4. `spsi` is gathered from `pmn_mo`.
5. `pmn_mo` is the time mean of `pmn`, which do not change in `diag.f90`.
6. `pmn` is assigned from local variable `psi` in `tracer.f90`.
7. `psi` = $h + \sum \rho^2 \texttt{pbar} \Delta z$
8. `pbar` is assigned with the average of `pmtp` and `pmtm` in `tracer.f90` before.
9. `pmtp` is multiply by a factor of `onbc` in `tracer.f90` before, while `pmtm` do not change in `tracer.f90` before assigned to `pbar`

7 Potential Bugs

1. In setcon.f90, Line 65-66, it cut vmix.nc value to fit `kh_max`, but didn't cut vmix to fit `kh_min`, i.e., `kh_c`

   Also, the code want to set missing value to 0.1, but it seems it doesn't work. The missing value are still unchanged. See the test on 101, case `pcom_1.0\exp02_check`

2. In the old code of Huang and Jin, A_m and A_h is different by 10 times, but now they are the same. see Line 208,209 of `setcon.f90`. Can change them back in the namelist of `am_c` and `ah_c`, to see the effects.

3. In main.f90, it defined allocatable arrays before MPI initialization, may be MPI init. should put in the very first?

4. Set `pbt_st` to 1 where it is zero at the beginning of `ready_st.f90`

5. `cv2` should be 2 times of the current value in `grdvar.f90`, it will affects the viscous terms.

6. The Coriolis adjust seems wrong in `barotr.f90`, the lines

```
du2(i,j) = ebla(j)*a(i,j) + eblb(j)*b(i,j)
dv2(i,j) = ebla(j)*b(i,j) - eblb(j)*a(i,j)
```

   in Line 217, 218 and Line 224, 225 should be

```
du2(i,j) = ebla(j)*a(i,j) - eblb(j)*b(i,j)
dv2(i,j) = ebla(j)*b(i,j) + eblb(j)*a(i,j)
```

   i,e, the "-" and "+" signs should be changed. See Appendix D for derivation.

7. Line 210-213 of `tracer.f90`:

```
dt_am(i,j,k)=2*rdxu(j)*(umn(i+1,j,k)-umn(i-1,j,k))-  &
     2*rdy*(vmn(i,j+1,k)-vmn(i,j-1,k))+vmn(i,j,k)*tan(lat(j)*torad)/radius
ds_am(i,j,k)=2*rdxu(j)*(vmn(i+1,j,k)-vmn(i-1,j,k))+  &
     2*rdy*(umn(i,j+1,k)-umn(i,j-1,k))-umn(i,j,k)*tan(lat(j)*torad)/radius
```

   I think the 2 in `dt_am` and `ds_am` is a mistake, it should be divided, not multiplied by 2. These may be explain the PCOM simulates a warmer temperature and fresher salinity, because the diffusion is over large.

8. The variable `athkdf` not defined before used. It will be set to 0 implicitly. This may cause GM scheme not work at all. (All thought it divides `sah` to `athkdf` in `setcon.f90`, but `sah` itself has not been initialized before the division.)

9. In `upwelling.f90`, it is weird that `w` seems to be just ω in p-coordinate, not $\dot{\sigma}$ in p-σ coordinate. And the `pn` in Line 78 of

```
w(i,j,k)=(w(i,j,k-1)+dz(k-1)*(dp(i,j)/pn(i,j)+c(i,j,k-1)))  &
```

should be `pn*pbt`, i.e., the original code may be use p_{h0} instead of p_h incorrectly.

10. In `readyc.f90`, Line 340-341:

```
diffu(i,j,k) = diffu(i,j,k) + rdz(k)*(wua(i,j)-wub(i,j))
diffv(i,j,k) = diffv(i,j,k) + rdz(k)*(wva(i,j)-wvb(i,j))
```

    i.e., the vertical viscosities is

    (7.1)

    But according to Eq. 2.15

$$\begin{aligned}\frac{1}{\sigma_h}\frac{\partial}{\partial\sigma}\left(\frac{\rho^2 g^2}{\sigma_h}\kappa_m\frac{\partial U}{\partial\sigma}\right) &= \frac{1}{\sigma_h}p_h\frac{\partial}{\partial p}\left(\frac{\rho^2 g^2}{\sigma_h}\kappa_m\frac{p_h}{-\rho g}\frac{\partial U}{\partial z}\right)\\ &= \frac{1}{\sigma_h}p_h\frac{\partial}{\partial p}\left(-\frac{\rho g}{\sigma_h}\kappa_m p_h\frac{\partial U}{\partial z}\right)\\ &= -\frac{1}{\sigma_h}p_h^2\frac{\partial}{\partial p}\left(\frac{\rho g}{\sigma_h}\kappa_m\frac{\partial U}{\partial z}\right)\end{aligned} \tag{7.2}$$

    So this may be cause vertical mixing incorrectly in the model.

11. See Section 8.21

12. See the global variable `pbxn`

8 Routine/Function Algorithm

8.1 Module: barotr.f90

Algorithm:

1. For every barotropic step:

   (a) compute the "artificial" horizontal viscosity $\nabla_h \cdot (A_m \nabla_h U_B)$ (adu) and $\nabla_h \cdot (A_m \nabla_h V_B)$ (adv).

$$\texttt{adu} = \nabla_h \cdot (A_m \nabla_h U_B) \tag{8.1a}$$

$$= \frac{1}{a^2 \cos\varphi} \frac{\partial}{\partial \varphi} \left(A_m \cos\varphi \frac{\partial U_B}{\partial \varphi} \right) + \frac{1}{a^2 \cos^2\varphi} \frac{\partial}{\partial \lambda} \left(A_m \frac{\partial U_B}{\partial \lambda} \right) \tag{8.1b}$$

$$= \frac{A_m}{a^2 \cos\varphi_{jh} \Delta\varphi} \left(\cos\varphi_{(j+1)} \frac{U_B(j+1) - U_B(j)}{\Delta\varphi} - \cos\varphi_j \frac{U_B(j) - U_B(j-1)}{\Delta\varphi} \right) + \frac{A_m}{a^2 \cos^2\varphi_{jh} \Delta\lambda} \left(\frac{U_B(i+1) - U_B(i)}{\Delta\lambda} - \frac{U_B(i) - U_B(i-1)}{\Delta\lambda} \right) \tag{8.1c}$$

$$= \frac{A_m \cos\varphi_{(j+1)}}{a^2 \cos\varphi_{jh} \Delta^2\varphi} \left(U_B(j+1) - U_B(j) \right) - \frac{A_m \cos\varphi_{(j+1)}}{a^2 \cos\varphi_{jh} \Delta^2\varphi} \left(U_B(j) - U_B(j-1) \right) + \frac{A_m}{a^2 \cos^2\varphi_{jh} \Delta^2\lambda} \left[(U_B(i+1) - U_B(i)) - (U_B(i) - U_B(i-1)) \right] \tag{8.1d}$$

$$= \texttt{am} * \left(2\texttt{r1c} * \overline{U_B}^{Dj+1} - 2\texttt{r1d} * \overline{U_B}^{Dj} \right) + \texttt{am} * 2\texttt{sdxu} * \overline{\overline{U_B}^{Di}}^{Di} \tag{8.1e}$$

   (b) Calculate spbt($\sqrt{\sigma_h}$)

   (c) Check whether spbt blow up, if so, stop the model run.

   (d) Calculate the pressure gradients etax and etay due to change of free surface

$$\texttt{etax} = \sqrt{\sigma_h} \left\{ \frac{\partial \phi_b}{a \cos\varphi \partial\lambda} + \frac{\partial \sigma_h}{a \cos\varphi \partial\lambda} \frac{1}{p_{h0}} \int_{p_t}^{p_b} (-\phi_b) dp + 2\sigma_h \frac{1}{p_{h0}} \int_{p_t}^{p_b} \frac{\partial(\phi - \phi_b)}{a \cos\varphi \partial\lambda} dp \right\} \tag{8.2a}$$

$$= \sqrt{\sigma_h} \left\{ \frac{\partial \phi_b}{a \cos\varphi \partial\lambda} + \frac{\partial \sigma_h}{a \cos\varphi \partial\lambda} \frac{1}{p_{h0}} \int_{p_t}^{p_b} (-\phi_b) dp + 2\sigma_h \frac{1}{p_{h0}} \int_{p_t}^{p_b} \left[\frac{\partial}{a \cos\varphi \partial\lambda} \left(\int_p^{p_b} \frac{1}{\rho} dp \right) \right] dp \right\} \tag{8.2b}$$

$$= \texttt{spbt} * \texttt{p5} \left(\texttt{phibx+ } 2\overline{\texttt{pbt} * \texttt{pbxx}}^{Di}{}^{Aj} + 4\overline{\texttt{pbt} * \texttt{pcxx}}^{Ai}{}^{Aj} \right) \tag{8.2c}$$

$$\texttt{etay} = \sqrt{\sigma_h} \left\{ \frac{\partial \phi_b}{a \partial\varphi} + \frac{\partial \sigma_h}{a \partial\varphi} \frac{1}{p_{h0}} \int_{p_t}^{p_b} (-\phi_b) dp + 2\sigma_h \frac{1}{p_{h0}} \int_{p_t}^{p_b} \frac{\partial(\phi - \phi_b)}{a \partial\varphi} dp \right\} \tag{8.3a}$$

$$= \sqrt{\sigma_h} \left\{ \frac{\partial \phi_b}{a \partial\varphi} + \frac{\partial \sigma_h}{a \partial\varphi} \frac{1}{p_{h0}} \int_{p_t}^{p_b} (-\phi_b) dp + 2\sigma_h \frac{1}{p_{h0}} \int_{p_t}^{p_b} \left[\frac{\partial}{a \partial\varphi} \left(\int_p^{p_b} \frac{1}{\rho} dp \right) \right] dp \right\} \tag{8.3b}$$

$$= \texttt{spbt} * \texttt{p5} \left(\texttt{phiby+ } 2\overline{\texttt{pbt} * \texttt{pbyx}}^{Dj}{}^{Ai} + 4\overline{\texttt{pbt} * \texttt{pcyx}}^{Aj}{}^{Ai} \right) \tag{8.3c}$$

(e) Calculate `advection + viscosiy + coriolis + pressure gradient`, and assign it to the variable `a`, `b`

$$\texttt{a} = \frac{1}{p_{h0}} \int_{p_t}^{p_b} \left[-P\mathcal{M}(u) - P\frac{\partial p_a}{\rho_0 a \cos\varphi \partial\lambda} + P\mathcal{D}_u \right] dp + 2\Omega \sin\varphi V_B - \texttt{etax} \tag{8.4a}$$

$$= \texttt{dub} + \texttt{ff} * \texttt{vpb} - \texttt{etax} \tag{8.4b}$$

$$\texttt{b} = \frac{1}{p_{h0}} \int_{p_t}^{p_b} \left[-P\mathcal{M}(v) - P\frac{\partial p_a}{\rho_0 a \partial\varphi} + P\mathcal{D}_v \right] dp - 2\Omega \sin\varphi U_B - \texttt{etay} \tag{8.4c}$$

$$= \texttt{dvb} - \texttt{ff} * \texttt{upb} - \texttt{etay} \tag{8.4d}$$

(f) Calculate horizontal velocity tendency `du2`, `dv2` by Coriolis adjustment (Liu, Yu, Li, and Zhang 2003)[P.26, 2.95], also see Appendix D for derivation.

(g) Calculate the bottom pressure tendency `dp`, i.e., the right side of Eq.3.1

$$\texttt{dp} = -\frac{1}{p_{h0}} \left[\frac{\partial(PU_B)}{a\cos\varphi\partial\lambda}(PU_B) + \frac{\partial(PV_B\cos\varphi)}{a\cos\varphi\partial\varphi}(PV_B\cos\varphi) \right] \tag{8.5a}$$

$$= -\frac{1}{p_{h0}} \left(\frac{1}{a\cos\varphi\Delta\lambda} \overline{\overline{PU_B}^{Aj}}^{Di} + \frac{1}{a\cos\varphi\Delta\varphi} \overline{\overline{PV_B\cos\varphi}^{Ai}}^{Dj} \right) \tag{8.5b}$$

$$= -\frac{1}{\texttt{pn}} \left(\texttt{rdxt}*\overline{\overline{\texttt{zu}*\texttt{spbt}*\texttt{upb}}^{Aj}}^{Di} + \texttt{rdyt}*\overline{\overline{\texttt{zu}*\texttt{spbt}*\texttt{vpb}*\texttt{cosu}}^{Ai}}^{Dj} \right) \tag{8.5c}$$

(h) Time integration of `upb, vpb, pbt` for the current integration time step using:

i. Euler-backward time scheme on every beginning of the current month:

```
pbt(i,j,tau) = pbt(i,j,taum) + dp(i,j)*dtsf
```

Then Recalculate the tendency of `upb, vpb, pbt` for the current time step and set `leapfrog_b` to `.true.` (i.e., using normal leapfrog time scheme for the rest of time step in the current month)

ii. Leapfrog time scheme on other time step of the current month:

```
t1             = pbt(i,j,taum) + dp (i,j)*c2dtsf
pbt(i,j,taum)  = pbt(i,j,tau)
pbt(i,j,tau)   = t1
```

Applied Asselin filer if needed (Liu et al. 2003)[P.25]

(i) Accumulate normalized sea bottom pressure σ_h to `pmup` and `pmtp` for baroclinic and tracer mode.

2. END

INTERFACE:

```
    =================
  subroutine barotr(ivn,itn,upb,vpb,r1c,r1d,sdxu,am,dub,dvb,spbt,pbt,phibx,   &
                phiby,pbxn,pbxs,pbye,pbyw,pcxn,pcxs,pcye,pcyw,ff,ebla,    &
               eblb,ebea,ebeb,pn,zu,cosu,rdxt,rdyt,leapfrog_b,euler_back, &
                dtsf,c2dtsf,afb1,afb2,pmup,pmtp,nbb,imt,jmt,km,imm,jmm,   &
               myid,west,east,north,south,asselin_b,snbc,emp,jstn,jedn,jsts,  &
                       jeds,smtha,fcof,umask,phib,pdxn,pdxs,pdye,pdyw,  &
                       lat,lon)
```

8.2 Module: bclinc.f90

Algorithm:

1. Calc. time-average σ_h for baroclinic time step.

```
pmup(i,j) = pmup(i,j)*onbb
```

2. Calc. $\sqrt{\sigma_h}$

$$\sqrt{\sigma_h} = \sqrt{\overline{\sigma_h}^{Aij}} \tag{8.6a}$$

$$= \sqrt{\overline{\texttt{pbar}}^{Aij}} \tag{8.6b}$$

3. Calc. pressure gradients terms

$$\texttt{px} = \frac{\partial(PU_B)}{a\cos\varphi\partial\lambda}\left(\frac{p}{\rho} + \phi_{b0} + \phi - \phi_{b0}\right) \tag{8.7a}$$

$$= \frac{\partial(PU_B)}{a\cos\varphi\partial\lambda}\left(\frac{p}{\rho} + \phi\right) \tag{8.7b}$$

$$= \frac{1}{a\cos\varphi\Delta\lambda}\left(\frac{1}{\rho}\overline{\sigma_h p_0}^{Di} + \overline{\phi_{b0} + \sigma_h\int_p^{p_b}\frac{1}{\rho}dp}^{Di}\right) \tag{8.7c}$$

$$= \texttt{rdxt}\left(\overline{\texttt{rho}}^{Dk}\overline{\texttt{pbar}*\texttt{z}}^{Di} + \overline{\texttt{phib}+\texttt{pbar}*\texttt{rhodp}}^{Di}\right) \tag{8.7d}$$

$$\texttt{py} = \frac{\partial}{a\partial\varphi}\left(\frac{p}{\rho} + \phi\right) \tag{8.7e}$$

$$= \frac{1}{a\Delta\varphi}\left(\frac{1}{\rho}\overline{\sigma_h p_0}^{Dj} + \overline{\phi_{b0} + \sigma_h\int_p^{p_b}\frac{1}{\rho}dp}^{Dj}\right) \tag{8.7f}$$

$$= \texttt{rdy}\left(\overline{\texttt{rho}}^{Dk}\overline{\texttt{pbar}*\texttt{z}}^{Dj} + \overline{\texttt{phib}+\texttt{pbar}*\texttt{rhodp}}^{Dj}\right) \tag{8.7g}$$

$$\tag{8.7h}$$

4. In every layer:

   (a) Calc. pressure gradient terms

$$\texttt{etax} = \frac{\partial(PU_B)}{a\cos\varphi\partial\lambda}\left(\frac{p}{\rho} + \phi\right)\sqrt{\sigma_h} \tag{8.8a}$$

$$= \overline{\texttt{px}}^{Aj} * \texttt{spbt} \tag{8.8b}$$

$$\texttt{etay} = \frac{\partial}{a\partial\varphi}\left(\frac{p}{\rho} + \phi\right)\sqrt{\sigma_h} \tag{8.8c}$$

$$= \overline{\texttt{py}}^{Ai} * \texttt{spbt} \tag{8.8d}$$

   (b) Calculate tendency terms of horizontal velocities, i.e., advection + viscosity + pressure gradient + Coriolis

$$\texttt{a} = \texttt{du} + 2\Omega\sin\varphi V - \texttt{etax} \tag{8.9a}$$

$$\texttt{b} = \texttt{dv} - 2\Omega\sin\varphi U - \texttt{etay} \tag{8.9b}$$

where

$$\begin{aligned}\texttt{du} &= -\texttt{ux} - \sqrt{\sigma_h}\frac{\partial p_a}{\rho_0 a\cos\varphi\partial\lambda} + \texttt{diffu} \\ &= -\texttt{ux} - \texttt{pax} * \texttt{spbt} + \texttt{diffu} \end{aligned} \tag{8.10a}$$

$$\texttt{ux} = \frac{\partial(Uu)}{a\cos\varphi\partial\lambda} - \frac{U}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Uv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{U}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} + \frac{\partial(Uw)}{\partial p} - \frac{U}{2}\frac{\partial w}{\partial p} \tag{8.10b}$$

$$\texttt{diffu} = \frac{1}{\sqrt{\sigma_h}}\nabla_h\cdot(\sigma_h A_m\nabla_h u) + \frac{\partial}{\partial p}\left(\kappa_m\frac{g\rho_0}{p_h}\frac{\partial U}{\partial p}\right) + A_m\left(\frac{1-\tan^2\varphi}{a^2}U - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial\lambda}\right) \tag{8.10c}$$

$$\begin{aligned}\texttt{dv} &= -\texttt{vx} - \sqrt{\sigma_h}\frac{\partial p_a}{\rho_0 a\partial\varphi} + \texttt{diffv} \\ &= -\texttt{vx} - \texttt{pay} * \texttt{spbt} + \texttt{diffv} \end{aligned} \tag{8.11a}$$

$$\texttt{vx} = \frac{\partial(Vu)}{a\cos\varphi\partial\lambda} - \frac{V}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Vv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{V}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} + \frac{\partial(Vw)}{\partial p} - \frac{V}{2}\frac{\partial w}{\partial p} \tag{8.11b}$$

$$\texttt{diffv} = \frac{1}{\sqrt{p_h}}\nabla_h\cdot(\sigma_h A_m\nabla_h v) + \frac{\partial}{\partial p}\left(\kappa_m\frac{g\rho_0}{p_h}\frac{\partial V}{\partial p}\right) + A_m\left(\frac{1-\tan^2\varphi}{a^2}V + \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial\lambda}\right) \tag{8.11c}$$

(c) Coriolis adjustment of velocity tendency `a, b` (see Appendix D), and assign it to `du, dv`.

5. If it is the first baroclinic time step of the month:

   (a) Time integration of $(U,V) = (\sqrt{\sigma_h}u, \sqrt{\sigma_h}v)$ with Euler forward scheme:

   ```
   up(i,j,k,tau) = up(i,j,k,taum) + du(i,j,k)*dtuv
   ```

   (b) Calc. vertical integration of U, V (`call vinteg`) to `a, b`

   (c) Baroclinic and barotropic interactions:

   ```
   up(i,j,k,tau) = up(i,j,k,tau) - a(i,j) + upb(i,j,tau)
   ```

   (d) Set `leapfrog_c = True`

6. If it isn't the first baroclinic time step of the month:

   (a) Time integration of $(U,V) = (\sqrt{\sigma_h}u, \sqrt{\sigma_h}v)$ with leapfrog scheme, and store the result to `du, dv`:

   ```
   du(i,j,k,tau) = up(i,j,k,taum) + du(i,j,k)*c2dtuv
   ```

   (b) Calc. vertical integration of U, V (`call vinteg`) to `a, b` (note that now $(U,V) =$ `(du,dv)`)

   (c) Baroclinic and barotropic interactions:

   ```
   up(i,j,k,tau) = up(i,j,k,tau) - a(i,j) + upb(i,j,tau)
   ```

   And do Asselin filter if needed (Liu et al. 2003, P.25)

7. Calc. $\sqrt{\sigma_h}$:

$$\sqrt{\sigma_h} = \sqrt{\overline{\sigma_h}^{Aij}} \tag{8.12a}$$

$$= \sqrt{\overline{\texttt{pbar}}^{Aij}} \tag{8.12b}$$

8. Accumulate `ump, vmp`.

INTERFACE:

```
   =================
subroutine bclinc(pmup,pmum,phib,spbt,pbt,rhodp,rho,rdxt,rdy,ump,vmp,upb,    &
            vpb,up,vp,du,dv,epla,eplb,epea,epeb,ff,umask,ivn,z,dz,rzu, &
            onbb,leapfrog_c,dtuv,c2dtuv,afc1,afc2,cosu,imt,jmt,km,imm, &
                    jmm,west,east,north,south,asselin_c,itn)
   =================
```

8.3 Module: convect.f90

Convective adjustment if unstable stratification ocurs. A full convective adjustment scheme, based on GFDL's version
Algorithm:

1. Done.

I don't understand this procedure yet **INTERFACE:**

```
subroutine convect(t,pt,ps,itn,dz,imt,jmt,km,nt,imm,jmm,myid,west,east,north,sout
             dpo_con,din_con,p,pbar,mass_up,bottom_h,total_in,total_po)
```

8.4 Module: density.f90

Calculate reciprocal of water density at the middle of each water grid.

1. T = t(i,j,k,1,tau)
2. S = t(i,j,k,2,tau)
3. pressure at the middle of the layer is

$$p(t) = \texttt{pbt(t)z(k)} + p_a(t) \tag{8.13}$$

Then calculate the reciprocal of density ρ.

```
rho(i,j,k) = rdens(t0,s0,p0)
```

INTERFACE:

```
subroutine density(tmask,z,t,pbt,bcp,rho,decibar,imt,jmt,km,nt,imm,jmm,fixp,  &
                   west,east,north,south)
```

8.4.1 rho_ref

Calculate $\frac{\partial \rho}{\partial T}$, $\frac{\partial \rho}{\partial S}$ by giving a perturbation to the state equation. The perturbation is (set at `setcon.f90`)

$$\Delta T = 10^{-2}, \Delta S = 10^{-3} \tag{8.14}$$

Then

$$\frac{\partial \rho}{\partial T} = \frac{\rho(T + \Delta T, S, p) - \rho(T - \Delta T, S, p)}{2\Delta T} \tag{8.15a}$$

$$\frac{\partial \rho}{\partial S} = \frac{\rho(T, S + \Delta S, p) - \rho(T, S - \Delta S, p)}{2\Delta S} \tag{8.15b}$$

$$\tag{8.15c}$$

INTERFACE:

```
subroutine rho_ref(tmask,t,z,pbt,pt,ps,deltat,deltas,rdeltat,rdeltas,decibar,fixp
                   imt,jmt,km,nt,imm,jmm,west,east,north,south)
```

8.4.2 undens

This function calculates the density of seawater using the standard equation of state recommended by unesco(1981).

1. Convert input pressure from $decibar$ to bar
   ($1decibar = 0.1bar, 1bar = 10^5 Pa$)

   ```
   p=p0*1.0d-1 + 1.013 !! standard Pa
   ```

2. Calc. density `rw` for 0 bar pressure, i.e., ρ^0 at Table 1 of Millero and Poisson (1981).

$$\begin{aligned}
\texttt{rw} =& \rho^0 \\
=& 9.99842594 \times 10^2 + 6.793952 \times 10^{-2} t - 9.095290 \times 10^{-3} t^2 + \\
& 1.001685 \times 10^{-4} t^3 - 1.120083 \times 10^{-6} t^4 + 6.536336 \times 10^{-9} t^5 + \\
& (8.24493 \times 10^{-1} - 4.0899 \times 10^{-3} t + 7.6438 \times 10^{-5} t^2 - 8.2467 \times 10^{-7} t^3 + 5.3875 \times 10^{-9} t^4) S + \\
& (-5.72466 \times 10^{-3} + 1.0227 \times 10^{-4} t - 1.6546 \times 10^{-6} t^2) S^{3/2} + 4.8314 \times 10^{-4} S^2
\end{aligned} \tag{8.16}$$

3. Calc. density `rk` for p bar pressure, i.e., K at Table 1 of Jackett and Mcdougall (1995).

$$\begin{aligned}
\texttt{rk} =& K \\
=& 1.965933 \times 10^4 + 1.444304 \times 10^2\theta - 1.706103 \times 10^0\theta^2 \\
&+ 9.648704 \times 10^{-3}\theta^3 - 4.190253 \times 10^{-5}\theta^4 \\
&+ (5.284855 \times 10^1 - 3.101089 \times 10^{-1}\theta + 6.283263 \times 10^{-3}\theta^2 - 5.084188 \times 10^{-5}\theta^3)S \\
&+ (3.886640 \times 10^{-1} + 9.085835 \times 10^{-3}\theta - 4.619924 \times 10^{-4}\theta^2)S^{3/2} \\
&+ (3.186519 \times 10^0 + 2.212276 \times 10^{-2}\theta - 2.984642 \times 10^{-4}\theta^2 + 1.956415 \times 10^{-6}\theta^3)p \\
&+ [(6.704388 \times 10^{-3} - 1.847318 \times 10^{-4}\theta + 2.059331 \times 10^{-7}\theta^2)S + 1.480266 \times 10^{-4}S^{3/2}]p \\
&+ (2.102898 \times 10^{-4} - 1.202016 \times 10^{-5}\theta + 1.394680 \times 10^{-7}\theta^2 \\
&- 2.040237 \times 10^{-6}S + 6.128773 \times 10^{-8}S\theta + 6.207323 \times 10^{-10}S\theta^2)p^2
\end{aligned} \tag{8.17}$$

4. Calc. ρ(`undens`) by Eq. A1 of Jackett and Mcdougall (1995)

$$\begin{aligned}
\texttt{undens} &= \frac{\rho(S,\theta,0)}{1 - p/K(S,\theta,p)} \times 10^{-3} \\
&= \frac{\texttt{rw}}{1 - p/\texttt{rk}} \times 10^{-3}
\end{aligned} \tag{8.18}$$

   Note that $\times 10^{-3}$ is for convert unit from kg/m^3 to g/cm^3.

5. Done.

INTERFACE:

```
function undens(t,s,p0)
```

8.5 Module: diag.f90

Model output. Algorithm:

1. Gather `tmask, umask, dxdyt` to `stmask, sumask, sdxdyt`
2. Accumulate `runtime` by `dtts`
3. If the current tracer time step is suitable for output, turn on output switches `diag_out` and restart switches `restr_out`. Set output parameters such as file names(`normalname, fname`) and `factor` for average.
4. Accumulate `t(:,:,:,1,tau), t(:,:,:,2,tau)` to `tmn_mo, smn_mo`, accumulate `umn, vmn, wmn` to `umn_mo, vmn_mo, wmn_mo`, accumulate `pmn` to `pmn_mo`.
5. Calculate instantaneous Kinetic energy in the three directioins: $0.5mu^2 = 0.5\rho\Delta S\Delta z u^2$, the same for `v, w`.
6. If the current tracer time step is for output:
   (a) Gather `umn_mo, vmn_mo, wmn_mo, tmn_mo, smn_mo, pmn_mo` to `sumn, svmn, swmn, stmn, ssm`
   respectively.

(b) Calculate area-weighted `spsi` to `ev`. (`spsi` is the height of water column).

(c) Calculate global mean of `spsi` and assign to `ev`, use `spsi` as sea surface height (SSH) further.

(d) Assign `sumn, svmn, swmn, stmn, ssmn, spsi` to `uzhy, vzhy, wzhy, tzhy, szhy, pzhy`(which do not have halo longitude grid), respectively.

(e) Read coordinates(`lat, lon, z`) from `pcom_int.nc` (`ncname`) and write it to the current output file.

(f) Write `uzhy, vzhy, wzhy, tzhy, szhy, pzhy` to the current output file.

(g) Reset `tmn_mo, smn_mo, umn_mo, vmn_mo, wmn_mo, pmn_mo` to zero.

7. If `runtime` is divided by `infotime`, then it is time for output instantaneous infomation (note that the minimum increasement of `runtime` is `dtts`, so generally, `infotime` should not be less than `dtts`, otherwise, the frequency of this output step is not want you expected):

   (a) Set `eku, ekv, ekw` to zero.

   (b) Accumulate grid-point kinetic engergy to `eku, ekv, ekw`, and sum up these three to `ke_tol`.

   (c) Calculate global mean sea surface height to `ev`.

   (d) Output `month, day, runtime, ke_tol, ev` for diagnose.

8. When the current tracer time step is for restart output, write out `month, t, pbt, up, vp, upb, vpb, pmtp, ump,` to the restart file.

9. END

INTERFACE:

```
subroutine diag(imt,jmt,simt,sjmt,km,nt,imm,jmm,kmp1,nss,dtts,mode_t,year,month,
       mth,day,daypm,dxdyt,dxdyu,tmask,umask,itn,area,runtime,io_time,io_month, &
       restr_time,restr_mon,infotime,myid,ncpux,ncpuy,west,east,north,south,  &
       mat_myid,stager_t,t_stepu,pbt,pbt_st,up,vp,upb,vpb,pmtp,ump,vmp,adv_u,  &
       adv_v,uvtshdiag,pmn,umn,vmn,wmn,t,tmn_mo,smn_mo,pmn_mo,umn_mo,vmn_mo,  &
         wmn_mo,am,bcu,bcv,sa_ini,ma_ini,ke_ini,in_ini,po_ini,gr_mass, &
        dpo_adv  ,dpo_hdif  ,dpo_vdif  ,dpo_imp  ,dpo_con  ,dpo_bc   , &
       dpo_adv_mo,dpo_hdif_mo,dpo_vdif_mo,dpo_imp_mo,dpo_con_mo,dpo_bc_mo ,  &
        din_adv  ,din_hdif  ,din_vdif  ,din_imp  ,din_con  ,din_bc   , &
       din_adv_mo,din_hdif_mo,din_vdif_mo,din_imp_mo,din_con_mo,din_bc_mo ,  &
        dpo_ice  ,dpo_ast   ,dpo_bar   ,dke_bcf  ,dke_fri  ,dke_adv  , &
       dpo_ice_mo,dsa_ast   ,dpo_bar_mo ,dke_bcf_mo,dke_fri_mo,dke_adv_mo,  &
        din_ice  ,din_ast   ,din_bar   ,dke_ape  ,dke_pre  ,dke_bar  , &
       din_ice_mo,din_ast_mo ,din_bar_mo ,dke_ape_mo,dke_pre_mo,dke_bar_mo,  &
             dke_cor,dke_cor_mo,dpo_bcp,  &
         total_in,total_po,wind_en_mo,gr_mass2,gr_mass_mo,sa_first,dtuv)
```

8.6 Module: grdvar.f90

1. Read in model coordinate, including resolution and topography.
2. Set up T mask and U mask.
3. Set up geopotential height at sea bottom Φ_b.
4. Set up sea bottom pressure p_h (pn) (by `call setpn`).
5. Calc. the latitude-dependent coefficients in the viscous term (3.7) and the diffusion term (3.9)

INTERFACE:

```
subroutine grdvar(z,z0,dz0,pn,itn,ivn,tmask,umask,phib,dz,rdz,rdzw,zu,rzu,  &
            jstn,jedn,jeds,jsts,lat,lon,cost,cosu,ff,rdxt,rdxu,rdyt,rdyu,  &
             sdxt,sdxu,r1a,r1b,r1c,r1d,cv1,cv2,dxdyt,dxdyu,area,rdy,   &
                    decibar,imt,jmt,km,dlam,dphi,imm,jmm,kmp1,kmm1,  &
            myid,ncpux,ncpuy,west,east,north,south,mat_myid,simt,sjmt,  &
                    smth_start_nlat,smth_start_slat,bottom_h,acfl,dtts)
```

8.7 Module: inirun.f90

1. Set up Asselin temporal filter parameter
2. Calc. time steps in the mode-splitting technique
3. Set up parameters for semi-implicit handle Coriolis terms. I don't understand this part yet.
4. Read in atmospheric forcing
5. Read in initial T,S field
6. Set p_h in (2.9) to 1
7. initialize velocities, tracers, and other prognostic variables to zero
8. initialize some p_h related items to 1
9. read in restart field for the prognostic variables if it is a restart run

INTERFACE:

```
subroutine inirun(afb1,afc1,aft1,afb2,afc2,aft2,dtts,dtuv,dtsf,nss,ncc,nbb,  &
            onbb,oncc,onbc,c2dtsf,c2dtuv,c2dtts,epea,epeb,epla,eplb,   &
            ebea,ebeb,ebla,eblb,bcf,pt,ps,t,pbt,pbt_st,up,vp,upb,vpb,  &
             spbt,w,dub,dvb,du,dv,diffu,diffv,pmup,pmtp,pmum,pmtm,ump,  &
              vmp,umm,vmm,pax,pay,pbxn,pcxn,pdxn,pbxs,pcxs,pdxs,pbye,   &
           pcye,pdye,pbyw,pcyw,pdyw,rhodp,phibx,phiby,phib,rdxt,rdy,   &
              ff,month,restrt,rho,fixp,itn,imt,jmt,km,nt,imm,jmm,kmp1,  &
           dz,decibar,myid,ncpux,ncpuy,west,east,north,south,mat_myid,  &
               simt,sjmt,monloop,yearloop,t_stepu,stager_t,   &
               adv_u,adv_v,am)
```

8.8 Module: interp.f90

Interpolate monthly forcing field to the current day of the month, and calculate Newtonian restoring coefficient for heat flux. **INTERFACE:**

```
subroutine interp(day,daymd,daypm,mth,bcf,bcu,bcv,bct,bcp,bcs,emp,ddd,imt,jmt,  &
            simt,sjmt,myid,ncpux,ncpuy,mat_myid,monloop,yearloop,monlong)
```

8.9 Module: isopyc

initilization of isopycnal mixing (GM90) **INTERFACE:**

```
subroutine isopyi(imt,jmt,km,kmp1,slmxr,ahisop,ah,fzisop,kref,rdz0,   &
              dz0,z0,K1,K2,K3,adv_vetiso,adv_vntiso,rhoi,e,adv_vbtiso)
```

8.9.1 isopyc

Isopycnal mixing.
Algorithm:

1. Calc. densities `rhoi`for the surrounding three layers.

   NOTE: pressure passed to `undens` function here is actually depth of the sea in meters. For appromixate, 1 meter of head = 0.9804 decibar. The reason why not use the actual pressure may be: in the GM scheme, what we concern is the density gradient, not the density itself. What is more, the density should refer to the same pressure level when calculate gradient. So as long as we calculate the gradient at the same reference level, it doesn't matter what the exactly pressure level we choose. That's why we can use a approximate pressure to calculate the density.

2. Calc. `K2(,,,3)` centered on the northern face of "T" cells (`call K2_3`)
3. Calc. `K1(,,,3)` centered on the eastern face of "T" cells (`call K1_3`)
4. Calc. `K3` centered on bottom face of "T" cells (`call K3_123`)
5. Calc. isopycnal advective velocities for tracers (`call isoadv`)
6. Done.

INTERFACE:

```
subroutine isopyc(imt,jmt,km,kmp1,imm,jmm,nt,itn,tmask,kref,fzisop,rdz0,dz0,z0,
            rdxt,rdyt,rdy,rdzw,dz,pn,cosu,t,slmxr,athkdf,K1,K2,K3,  &
          adv_vetiso,adv_vntiso,adv_vbtiso,rhoi,e,fx,fy,west,east,north, &
                 south)
```

8.9.2 K1_3

Compute `K1(:,:,:,3)` at the center of the eastern face of "T" cells.
Algorithm:

1. Calc. vertical gradient of density ρ (`rhoi`)

$$\begin{aligned} \mathtt{e}(:,:,:,3) &= 10^{10} \times \frac{\partial \rho}{a\cos\varphi \partial\lambda} \\ &= 10^{10} \times \frac{\Delta\overline{\rho}^{Dk}}{\Delta z} \\ &= \mathtt{c1e10} * \mathtt{rdz0} * \overline{\mathtt{rhoi}}^{Dk} \end{aligned} \tag{8.19}$$

2. Linearly extrapolate densities to ocean surface for calculation of $\Delta\rho/\Delta z$ involving level 1.

   Suppose a line pass (x_1, y_1) and (x_2, y_2), then the intercept of the y axis is

$$y_b = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1} \tag{8.20}$$

   In this case, the line pass $(0.5\Delta z_1, \rho_1)$ and $(\Delta z_1 + 0.5\Delta z_2, \rho_2)$, so

$$\begin{aligned} y_b =& \frac{\rho_1(\Delta z_1 + 0.5\Delta z_2) - \rho_2 0.5\Delta z_1}{0.5 z_1 + 0.5 z_2} \\ =& \mathtt{rdzw(1)} * [\rho_1(\Delta z_1 + 0.5\Delta z_2) - \rho_2 0.5\Delta z_1] \end{aligned} \tag{8.21}$$

   Then

$$\begin{aligned} \mathtt{e} =& 10^{10} \times \frac{\partial \rho}{\partial z} \\ =& \frac{10^{10}}{\Delta z_1} \times (y_b - 0.5(\rho_1 + \rho_2)) \\ =& \mathtt{fxd} * [\mathtt{fxc} - \mathtt{p5} * (\mathtt{fxa} + \mathtt{fxb})] \end{aligned} \tag{8.22}$$

3. Linearly extrapolate densities to ocean bottom for calculation of $\Delta\rho/\Delta z$ involving sea bottom.

   Assuming the origin point is at the sea bottom, then in this case, the line pass $(0.5\Delta z_k, \rho_k)$ and $(\Delta z_k + 0.5\Delta z_{k-1}, \rho_{k-1})$, so

$$\begin{aligned} y_b =& \frac{\rho_k(\Delta z_k + 0.5\Delta z_{k-1}) - \rho_{k-1} 0.5\Delta z_k}{0.5 z_k + 0.5 z_{k-1}} \\ =& \mathtt{rdzw(k-1)} * [\rho_k(\Delta z_k + 0.5\Delta z_{k-1}) - \rho_{k-1} 0.5\Delta z_k] \end{aligned} \tag{8.23}$$

   Then

$$\begin{aligned} \mathtt{e} =& 10^{10} \times \frac{\partial \rho}{\partial z} \\ =& \frac{10^{10}}{\Delta z_k} \times (0.5(\rho_k + \rho_{k-1}) - y_b) \\ =& \mathtt{rdz0k} * \mathtt{c1e10} * [\mathtt{p5} * (\mathtt{fxa} + \mathtt{fxb}) - \mathtt{fxc}] \end{aligned} \tag{8.24}$$

4. Calc. the zonal of density ρ (rhoi)

$$\begin{aligned} \mathtt{e}(:,:,:,1) &= 10^{10} \times \frac{\partial \rho}{a \partial \varphi} \\ &= 10^{10} \times \frac{\Delta \overline{\rho}^{Di}}{a \cos\varphi \Delta z} \\ &= \mathtt{c1e10} * \mathtt{rdxt} * \overline{\mathtt{rhoi}}^{Di} \end{aligned} \tag{8.25}$$

5. Calc. the meridional of density ρ (rhoi)

$$\begin{aligned} \mathtt{e}(:,:,:,2) &= 10^{10} \times \frac{\partial \rho}{\partial z} \\ &= 10^{10} \times \frac{\Delta \overline{\rho}^{Dj}}{a \Delta z} \\ &= \mathtt{c1e10} * \mathtt{rdy} * \overline{\mathtt{rhoi}}^{Dj} \end{aligned} \tag{8.26}$$

6. Check `e(:,:,:,3)`, if it is greater than

$$-10^{10} \sqrt{\left(\frac{\partial \rho}{a \partial \varphi}\right)^2 + \left(\frac{\partial \rho}{\partial z}\right)^2} \times 100 \tag{8.27}$$

   then set to the above value.

7. Calc. rotate tensor compoent `K23`

$$\begin{aligned} \mathtt{K2(i,k,j,3)} &= -10^{10} \times \frac{\frac{\partial \rho}{a \partial \varphi} * \frac{\partial \rho}{a \cos\varphi \partial \lambda} * 1.0}{\left(\frac{\partial \rho}{a \cos\varphi \partial \lambda}\right)^2 + 10^{-25}} \\ &= -10^{10} \times \frac{\frac{\partial \rho}{a \partial \varphi} * \frac{\partial \rho}{a \cos\varphi \partial \lambda} * \mathtt{fzisop}}{\left(\frac{\partial \rho}{a \cos\varphi \partial \lambda}\right)^2 + eps} \end{aligned} \tag{8.28}$$

8. Done

INTERFACE:

```
subroutine k1_3(imt,jmt,imm,jmm,km,kmp1,slmxr,itn,e,rhoi,rdz0,dz0,tmask,  &
                rdxt,rdy,rdzw,fzisop,K1,west,east,north,south)
```

8.9.3 K2_3

Compute `K2(:,:,:,3)` at the center of the northern face of "T" cells.
Algorithm:

1. Calc. vertical gradient of density ρ (rhoi)

$$\begin{aligned} \mathtt{e}(:,:,:,3) &= 10^{10} \times \frac{\partial \rho}{a \cos\varphi \partial \lambda} \\ &= 10^{10} \times \frac{\Delta \overline{\rho}^{Dk}}{\Delta z} \\ &= \mathtt{c1e10} * \mathtt{rdz0} * \overline{\mathtt{rhoi}}^{Dk} \end{aligned} \tag{8.29}$$

2. Linearly extrapolate densities to ocean surface for calculation of $\Delta\rho/\Delta z$ involving level 1.

   Suppose a line pass (x_1, y_1) and (x_2, y_2), then the intercept of the y axis is

$$y_b = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1} \tag{8.30}$$

   In this case, the line pass $(0.5\Delta z_1, \rho_1)$ and $(\Delta z_1 + 0.5\Delta z_2, \rho_2)$, so

$$\begin{aligned} y_b =& \frac{\rho_1(\Delta z_1 + 0.5\Delta z_2) - \rho_2 0.5\Delta z_1}{0.5 z_1 + 0.5 z_2} \\ =& \texttt{rdzw(1)} * [\rho_1(\Delta z_1 + 0.5\Delta z_2) - \rho_2 0.5\Delta z_1] \end{aligned} \tag{8.31}$$

   Then

$$\begin{aligned} \texttt{e} =& 10^{10} \times \frac{\partial \rho}{\partial z} \\ =& \frac{10^{10}}{\Delta z_1} \times (y_b - 0.5(\rho_1 + \rho_2)) \\ =& \texttt{fxd} * [\texttt{fxc} - \texttt{p5} * (\texttt{fxa} + \texttt{fxb})] \end{aligned} \tag{8.32}$$

3. Linearly extrapolate densities to ocean bottom for calculation of $\Delta\rho/\Delta z$ involving sea bottom.

   Assuming the origin point is at the sea bottom, then in this case, the line pass $(0.5\Delta z_k, \rho_k)$ and $(\Delta z_k + 0.5\Delta z_{k-1}, \rho_{k-1})$, so

$$\begin{aligned} y_b =& \frac{\rho_k(\Delta z_k + 0.5\Delta z_{k-1}) - \rho_{k-1} 0.5\Delta z_k}{0.5 z_k + 0.5 z_{k-1}} \\ =& \texttt{rdzw(k-1)} * [\rho_k(\Delta z_k + 0.5\Delta z_{k-1}) - \rho_{k-1} 0.5\Delta z_k] \end{aligned} \tag{8.33}$$

   Then

$$\begin{aligned} \texttt{e} =& 10^{10} \times \frac{\partial \rho}{\partial z} \\ =& \frac{10^{10}}{\Delta z_k} \times (0.5(\rho_k + \rho_{k-1}) - y_b) \\ =& \texttt{rdz0k} * \texttt{c1e10} * [\texttt{p5} * (\texttt{fxa} + \texttt{fxb}) - \texttt{fxc}] \end{aligned} \tag{8.34}$$

4. Calc. the zonal of density ρ (`rhoi`)

$$\begin{aligned} \texttt{e(:,:,:,1)} &= 10^{10} \times \frac{\partial \rho}{a \partial \varphi} \\ &= 10^{10} \times \frac{\Delta \overline{\rho}^{Di}}{a \cos\varphi \Delta z} \\ &= \texttt{c1e10} * \texttt{rdxt} * \overline{\texttt{rhoi}}^{Di} \end{aligned} \tag{8.35}$$

5. Calc. the meridional of density ρ (`rhoi`)

$$\begin{aligned} \texttt{e(:,:,:,2)} &= 10^{10} \times \frac{\partial \rho}{\partial z} \\ &= 10^{10} \times \frac{\Delta \overline{\rho}^{Dj}}{a \Delta z} \\ &= \texttt{c1e10} * \texttt{rdy} * \overline{\texttt{rhoi}}^{Dj} \end{aligned} \tag{8.36}$$

6. Check `e(:,:,:,3)`, if it is greater than

$$-10^{10}\sqrt{\left(\frac{\partial\rho}{a\partial\varphi}\right)^2+\left(\frac{\partial\rho}{\partial z}\right)^2}\times 100 \tag{8.37}$$

then set to the above value.

7. Calc. rotate tensor compoent `K23`

$$\begin{aligned}\texttt{K2(i,k,j,3)} &= -10^{10}\times\frac{\frac{\partial\rho}{\partial z}*\frac{\partial\rho}{a\cos\varphi\partial\lambda}*\texttt{1.0}}{\left(\frac{\partial\rho}{a\cos\varphi\partial\lambda}\right)^2+10^{-25}}\\ &= -10^{10}\times\frac{\frac{\partial\rho}{\partial z}*\frac{\partial\rho}{a\cos\varphi\partial\lambda}*\texttt{fzisop}}{\left(\frac{\partial\rho}{a\cos\varphi\partial\lambda}\right)^2+eps}\end{aligned} \tag{8.38}$$

8. Done

INTERFACE:

```
subroutine k2_3(imt,jmt,imm,jmm,km,kmp1,slmxr,itn,e,rhoi,rdz0,dz0,tmask,  &
                rdxt,rdy,rdzw,fzisop,K2,west,east,north,south)
```

8.9.4 K3_123

Calc. isopycnal diffusive tracer fluxes.
Algorithm:

1. Calc. the zonal gradient of density ρ (`rhoi`)

$$\begin{aligned}\texttt{e(:,:,:,1)} &= 10^{10}\times\frac{\partial\rho}{a\partial\varphi}\\ &= 10^{10}\times\frac{\Delta\overline{\rho}^{Di}}{a\cos\varphi\Delta\lambda}\\ &= \texttt{c1e10}*\texttt{rdxt}*\overline{\texttt{rhoi}}^{Di}\end{aligned} \tag{8.39}$$

2. Calc. the meridional gradient of density ρ (`rhoi`)

$$\begin{aligned}\texttt{e(:,:,:,2)} &= 10^{10}\times\frac{\partial\rho}{\partial z}\\ &= 10^{10}\times\frac{\Delta\overline{\rho}^{Dj}}{a\Delta\varphi}\\ &= \texttt{c1e10}*\texttt{rdy}*\overline{\texttt{rhoi}}^{Dj}\end{aligned} \tag{8.40}$$

3. Calc. vertical gradient of density ρ (`rhoi`)

$$\begin{aligned}\texttt{e(:,:,:,3)} &= 10^{10}\times\frac{\partial\rho}{a\cos\varphi\partial\lambda}\\ &= 10^{10}\times\frac{\Delta\overline{\rho}^{Dk}}{\Delta z}\\ &= \texttt{c1e10}*\texttt{rdzw}*\overline{\texttt{rhoi}}^{Dk}\end{aligned} \tag{8.41}$$

4. Check `e(:,:,:,3)`, if it is greater than

$$-10^{10}\sqrt{\left(\frac{\partial\rho}{a\partial\varphi}\right)^2+\left(\frac{\partial\rho}{\partial z}\right)^2}\times 100 \tag{8.42}$$

then set to the above value.

5. Calc. rotate tensor compoent `K23`

$$\texttt{K3(i,k,j,1)} = 10^{10}\times\frac{\partial\rho}{a\cos\varphi\partial\lambda}\frac{\partial\rho}{\partial z}\times\frac{\texttt{1.0}}{\left(\frac{\partial\rho}{\partial z}\right)^2+10^{-25}} \tag{8.43a}$$

$$= 10^{10}\times\frac{\partial\rho}{a\cos\varphi\partial\lambda}\frac{\partial\rho}{\partial z}\times\frac{\texttt{fzisop}}{\left(\frac{\partial\rho}{\partial z}\right)^2+\texttt{eps}} \tag{8.43b}$$

$$\texttt{K3(i,k,j,2)} = 10^{10}\times\frac{\partial\rho}{a\partial\varphi}\frac{\partial\rho}{\partial z}\times\frac{\texttt{1.0}}{\left(\frac{\partial\rho}{\partial z}\right)^2+10^{-25}} \tag{8.43c}$$

$$= 10^{10}\times\frac{\partial\rho}{a\partial\varphi}\frac{\partial\rho}{\partial z}\times\frac{\texttt{fzisop}}{\left(\frac{\partial\rho}{\partial z}\right)^2+\texttt{eps}} \tag{8.43d}$$

$$\texttt{K3(i,k,j,3)} = 10^{10}\times\sqrt{\left(\frac{\partial\rho}{a\cos\varphi\partial\lambda}\right)^2+\square\left(\frac{\partial\rho}{a\partial\varphi}\right)^2}\times\frac{\texttt{1.0}}{\left(\frac{\partial\rho}{\partial z}\right)^2+10^{-25}} \tag{8.43e}$$

$$= 10^{10}\times\sqrt{\left(\frac{\partial\rho}{a\cos\varphi\partial\lambda}\right)^2+\square\left(\frac{\partial\rho}{a\partial\varphi}\right)^2}\times\frac{\texttt{fzisop}}{\left(\frac{\partial\rho}{\partial z}\right)^2+\texttt{eps}} \tag{8.43f}$$

6. Done

INTERFACE:

```
subroutine k3_123(imt,jmt,imm,jmm,km,kmp1,slmxr,e,rhoi,rdz0,dz0,tmask,  &
                  rdxt,rdy,rdzw,fzisop,K3)
```

8.9.5 isoadv

Compute isopycnal transport velocities.
Algorithm:
NOTE: The variable `athkdf` is unintialized here (and usually default to 0), so it may be a bug. The model will not include eddy-induced velocity at all. This may be a problem for tracer mixing along isopycals.

1. Compute the meridional component of the isopycnal mixing velocity at the center of the northern face of the "T" cells (`adv_vntiso`):

$$\begin{aligned}\texttt{adv\_vntiso} &= -A_I p_h\cos\varphi\frac{\partial K_{23}}{\partial z}\\ &= -\texttt{athkdf}*\texttt{fy}*\texttt{cosu}*\texttt{rdz0}*\overline{\texttt{K2(:,:,:,3}}^{Dk}\end{aligned} \tag{8.44}$$

2. Compute the zonal component of the isopycnal mixing velocity at the center of the eastern face of the "T" cells (adv_vetiso):

$$\begin{aligned} \texttt{adv\_vetiso} &= -A_I p_h \cos\varphi \frac{\partial K_{13}}{\partial z} \\ &= -\texttt{athkdf} * \texttt{fy} * \texttt{cosu} * \texttt{rdz0} * \overline{\texttt{K1}(:,:,:,3)}^{Dk} \end{aligned} \tag{8.45}$$

3. compute vertical mass advection: Ps*dz/dt (adv_vbtiso), (by call isow):

   (a) Calc.:

$$\frac{\partial \texttt{u}}{a\cos\varphi\partial\lambda} + \frac{\partial \texttt{v}}{a\cos\varphi\partial\varphi} = \texttt{dz} * (\overline{\texttt{u}}^{Di} * \texttt{rdxt} + \overline{\texttt{v}}^{Dj} * \texttt{rdyt}) \tag{8.46}$$

   (b) Diagnose w by

$$\begin{aligned} \texttt{w} &= \int_{p_t}^{p} \left(\frac{\partial \texttt{u}}{a\cos\varphi\partial\lambda} + \frac{\partial \texttt{v}}{a\cos\varphi\partial\varphi} \right) dp + \frac{1}{p_h} \int_{p_t}^{p} \left[- \int_{p_t}^{p_b} \left(\frac{\partial \texttt{u}}{a\cos\varphi\partial\lambda} + \frac{\partial \texttt{v}}{a\cos\varphi\partial\varphi} \right) dp \right] dp \\ &= \texttt{w(k-1)} + \texttt{w(k)} + \texttt{dz(k)} * \texttt{dp/pn} \end{aligned} \tag{8.47}$$

4. Done.

INTERFACE:

```
subroutine isoadv(imt,jmt,imm,jmm,km,athkdf,itn,rdxt,rdyt,dz,pn,fx,fy,rdz0,  &
                  cosu,tmask,adv_vetiso,adv_vntiso,adv_vbtiso,K1,K2,west,    &
                         east,north,south)
```

8.9.6 isoflux

Calc. K3 centered on bottom face of "T" cells
Algorithm:

1. Compute the vertical tracer flux temp at the northern face of T cells.

$$\begin{aligned} \texttt{temp} &= \int_{p_t}^{p} \frac{\partial \texttt{t}}{\partial p} \\ &= \texttt{rdz0} * * \overline{\texttt{t}}^{Dk} \end{aligned} \tag{8.48}$$

2. Calc. temp(:,:,1)

$$\begin{aligned} \texttt{temp}(:,:,1) &= \frac{1}{2} \int_{p_t}^{p} \frac{\partial \texttt{t}}{\partial p} \\ &= \texttt{p5} * \texttt{rdz0} * * \overline{\texttt{t}}^{Dk} \end{aligned} \tag{8.49}$$

3. Linearly extrapolate tracer to ocean bottom.

Assuming the origin point is at the sea bottom, then in this case, the line pass $(0.5\Delta z_k, \mathtt{t}_k)$ and $(\Delta z_k + 0.5\Delta z_{k-1}, \mathtt{t}_{k-1})$, so

$$\begin{aligned} y_b =& \frac{T_k(\Delta z_k + 0.5\Delta z_{k-1}) - T_{k-1}0.5\Delta z_k}{0.5z_k + 0.5z_{k-1}} \\ =& \mathtt{rdzw(k-1)} * [T_k(\Delta z_k + 0.5\Delta z_{k-1}) - T_{k-1}0.5\Delta z_k] \end{aligned} \tag{8.50}$$

Then

$$\begin{aligned} \mathtt{temp(:,:,km)} =& \frac{\partial T}{\partial z} \\ =& \Delta z_k \times (0.5(T_k + T_{k-1}) - y_b) \\ =& \mathtt{rdz0k} * [\mathtt{p5} * (\mathtt{fxa} + \mathtt{fxb}) - \mathtt{fxc}] \end{aligned} \tag{8.51}$$

4. Calc. meridional tracer flux:

$$\begin{aligned} \mathtt{tf} =& \mathtt{tf} + A_I \frac{\partial(PV_B \cos\varphi)}{a\cos\varphi\partial\varphi} \left[\left(\frac{\partial T}{a\partial\varphi} + K_{23} \int_{p_t}^{p} \frac{\partial \mathtt{t}}{\partial p} \right) 0.5 p_h \cos\varphi \right] \\ =& \mathtt{tf} + A_I \frac{\partial(PV_B \cos\varphi)}{a\cos\varphi\partial\varphi} \left[\left(\mathtt{rdy} * \overline{t}^{Dj} + \mathtt{K2(:,:,:,3)} * \mathtt{temp} \right) * \mathtt{p5} * \mathtt{fy} * \mathtt{cosu} \right] \\ =& \mathtt{tf} + A_I \frac{\partial(PV_B \cos\varphi)}{a\cos\varphi\partial\varphi} [\mathtt{wkj}] \end{aligned} \tag{8.52}$$

where `tf` is the tracer flux input through parameters:

$$\begin{aligned} \mathtt{tf} =& -\sqrt{p_h} U \frac{\partial \mathtt{t}}{a\cos\varphi\partial\lambda} - \sqrt{p_h} V \cos\varphi \frac{\partial \mathtt{t}}{a\cos\varphi\partial\varphi} - W \int_{p_t}^{p} \frac{\partial \mathtt{t}}{\partial p} \\ =& -\mathtt{rdxt} * \mathtt{du} * \overline{\mathtt{t}}^{Di} - \mathtt{rdyt} * \mathtt{dv} * \overline{\mathtt{t}}^{j} - \mathtt{rdz} * \mathtt{w} * \overline{\mathtt{t}}^{Dk} \end{aligned} \tag{8.53}$$

where W is the value after `call upwelling`.

5. Compute the vertical tracer flux `temp` at the eastern face of T cells.

$$\begin{aligned} \mathtt{temp} =& \int_{p_t}^{p} \frac{\partial \mathtt{t}}{\partial p} \\ =& \mathtt{rdz0} * * \overline{T}^{Dk} \end{aligned} \tag{8.54}$$

6. Compute the above term on the top and bottom value by linear extrapolation.

7. Compute of zonal tracer flux

$$\begin{aligned} \mathtt{tf} =& \mathtt{tf} + A_I \frac{\partial(PU_B)}{a\cos\varphi\partial\lambda} \left[\left(\frac{\partial T}{a\cos\varphi\partial\lambda} + K_{13} \int_{p_t}^{p} \frac{\partial \mathtt{t}}{\partial p} \right) 0.5 p_h \right] \\ =& \mathtt{tf} + A_I \frac{\partial(PU_B)}{a\cos\varphi\partial\lambda} \left[\left(\mathtt{rdxt} * \overline{T}^{Di} + \mathtt{K1(:,:,:,3)} * \mathtt{temp} \right) * \mathtt{p5} * \mathtt{fx} \right] \\ =& \mathtt{tf} + A_I \frac{\partial(PV_B \cos\varphi)}{a\cos\varphi\partial\varphi} [\mathtt{wki}] \end{aligned} \tag{8.55}$$

8. Compute the vertical tracer flux containing the `K31` and `K32` components which are to be solved explicitly. The K33 component will be treated semi-implicitly

$$\begin{aligned} \texttt{tf} =& \texttt{tf} + g\rho_0 \left(\frac{\partial K_{31}}{\partial p} + \frac{\partial K_{32}}{\partial p} \right) \\ =& \texttt{tf} + g\rho_0 \Delta p \left(A_I K_{31} \frac{\partial T}{a \cos\varphi \partial \lambda} + A_I K_{32} \frac{\partial T}{a \partial \varphi} \right) \\ =& \texttt{tf} + \texttt{gravr} * \texttt{rdz} \left(\texttt{ahisop} * K_{31} \frac{\partial T}{a \cos\varphi \partial \lambda} + \texttt{ahisop} * K_{32} \frac{\partial T}{a \partial \varphi} \right) \end{aligned} \tag{8.56}$$

9. Compute the meridional component of the isopycnal velocity mixing

$$\begin{aligned} \texttt{tf} =& \texttt{tf} - \texttt{adv\_vntiso} \frac{\partial T}{a \cos\varphi \partial \varphi} \\ =& \texttt{tf} - \texttt{adv\_vntiso} * \texttt{rdyt} * \overline{T}^{Dj} \end{aligned} \tag{8.57}$$

10. Compute the zonal component of the isopycnal velocity mixing

$$\begin{aligned} \texttt{tf} =& \texttt{tf} - \texttt{adv\_vetiso} \frac{\partial T}{a \cos\varphi \partial \lambda} \\ =& \texttt{tf} - \texttt{adv\_vetiso} * \texttt{rdxt} * \overline{T}^{Di} \end{aligned} \tag{8.58}$$

11. Compute the vertical component of the isopycnal velocity mixing

$$\begin{aligned} \texttt{tf} =& \texttt{tf} - \texttt{adv\_vbtiso} \frac{\partial T}{\partial p} \\ =& \texttt{tf} - \texttt{adv\_vbtiso} * \texttt{rdz} * \overline{T}^{Dk} \end{aligned} \tag{8.59}$$

12. Done.

INTERFACE:

```
subroutine isoflux(tf,mtrace,imt,jmt,imm,jmm,nt,km,kmp1,itn,gravr,rdz0,dz0,rdz,
                   rdzw,rdxt,rdyt,rdy,cosu,tmask,t,ahisop,K1,K2,K3,adv_vetiso, &
                            adv_vntiso,adv_vbtiso,fx,fy)
```

8.10 Module: main.f90

PCOM main program. It reads an input namelist, then output model variables and restart files after integration.

1. Define allocatable arrays.
2. Initialize MPI environments.
3. Read in `namelist`.
4. Set basic MPI parameters and distribute them among all processes.
5. Compute other basic parameters in all processes.

6. Initialize all MPI arrays in every process.
7. Initialize basic arrays for integration:
   (a) Set scalar quantities (`call setcon`).
   (b) Set resolution, `t/u` mask and the parameters depended on latitude (`call grdvar`).
   (c) Initialization (`call inirun`).
8. integration cycle, for every month:
   (a) Set Euler backward scheme
   (b) If monlong, distribute real forcing to bcf for the past, now and future months
   (c) In every day of the month:
      i. record start time of MPI
      ii. compute coefficients for calculation of potential density anomaly (`call rho_ref`)
      iii. interpolates monthly mean forcing fields (`call interp`)
      iv. in every thermal time step:
         A. calc. baroclinic pressure and the relevant variables (`call readyt`)
         B. In every baroclinic time step of one thermal time step:
            a. Calculate momentum advection, diffusion and their vertical integrals (`call readyc`)
            b. Compute `pbt, upb, vpb` at "tau+1" time level (`call barotr`)
            c. Prediction of baroclinic mode (`call bclinc`)
            d. Increase the time step countor `t_stepu` by 1.
         C. Prediction of temperature and salinity (`call tracer`)
         D. Convective adjustment if unstable stratification occurs (`call convect`)
         E. Output model data (`call diag`)
      v. Record stop time of MPI, and print out the escaped time for one day integration.
   (d) Increase `month` by 1, and decrease `runlen_mon` by 1.
9. END (`call mpi_finalize`)

INTERFACE:

```
program main
```

8.11 Module: prsgrd.f90

1. Calc. horizontal gradient of atmospheric pressure

$$\texttt{pax} = \frac{1}{\rho_0} \cdot \frac{\partial p_a}{a \cos\varphi \partial\lambda} \tag{8.60a}$$

$$= \texttt{rrho\_0} * \overline{\overline{\texttt{bcp}}^{Di}} * \texttt{rdxt}^{Aj} \tag{8.60b}$$

$$\texttt{pay} = \frac{1}{\rho_0} \cdot \frac{\partial p_a}{a \partial \varphi} \tag{8.61a}$$

$$= \texttt{rrho\_0} * \overline{\overline{\texttt{bcp}}^{Dj}}^{Ai} * \texttt{rdy} \tag{8.61b}$$

where P_a is atmospheric pressure at the sea surface.

2. Calc. $\texttt{rhodp} = \int_p^{p_b} \frac{1}{\rho} dp$
3. Calc. vertically integrated pressure-associated terms `pbxn` and `pbxs`
4. Calc. vertically integrated of zonal gradient of $\texttt{rhodp} = \int_p^{p_b} \frac{1}{\rho} dp$ (`pcxn, pcxs`)
5. Calc. vertically integrated pressure-associated terms `pbye` and `pbyw`
6. Calc. vertically integrated of meridional gradient of $\texttt{rhodp} = \int_p^{p_b} \frac{1}{\rho} dp$ (`pcye, pcyw`)

INTERFACE:

```
subroutine prsgrd(bcp,rdxt,rdy,pax,pay,itn,ivn,rho,rhodp,z,dz,rzu,pbxn,pbxs,  &
                  pbye,pbyw,pcxn,pcxs,pcye,pcyw,pdxn,pdxs,pdye,pdyw,imt,jmt,  &
                          km,imm,jmm,kmp1,west,east,north,south)
```

8.12 Module: readyc.f90

Momentum advections, viscosities and atmpospheric pressure terms

Algorithm:

1. Store σ_h to `pmum`, prepare to accumulate new time-averaged σ_h in `pmup`.
2. Calc. $\sqrt{\sigma_h}$

```
spbt(i,j) = p5*sqrt(pbt(i,j  ,tau) + pbt(i+1,j  ,tau) + &
                    pbt(i,j+1,tau) + pbt(i+1,j+1,tau))
```

3. "Prognose" vertical velocity w by `call upwelling`:

$$\frac{\partial \mathrm{w}}{\partial p} = \frac{1}{p_{h0}} \int_{p_t}^{p_b} \left(\frac{\partial \mathrm{u}}{a \cos\varphi \partial\lambda} + \frac{\partial \mathrm{v}}{a \cos\varphi \partial\varphi} \right) dp + \frac{\partial \mathrm{u}}{a \cos\varphi \partial\lambda} + \frac{\partial \mathrm{v}}{a \cos\varphi \partial\varphi} \tag{8.62}$$

4. Calculate vertical velocity on the surface of U cell:

$$p_h \dot{\sigma} = \frac{p_h \sigma_h \dot{\sigma}}{\sigma_h} = \frac{\mathrm{w}}{\mathrm{pbt}} \tag{8.63}$$

5. Calc. advections, horizontal velocities and viscosities in every layer:

(a) Dianose u, v:

$$u = \frac{U}{\sqrt{\sigma_h}} \tag{8.64a}$$

$$v = \frac{V}{\sqrt{\sigma_h}} \tag{8.64b}$$

(b) Calc. advection by u:

$$\begin{aligned} \mathtt{ux} &= \frac{\partial(Uu)}{a\cos\varphi\partial\lambda} - \frac{U}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} \\ &= \frac{1}{a\cos\varphi\Delta\lambda}\left(\overline{\overline{U}^{Ai}\overline{u}^{Ai}}^{Di} - \frac{U}{2}\overline{\overline{u}^{Ai}}^{Di}\right) \end{aligned} \tag{8.65a}$$

$$\begin{aligned} \mathtt{vx} &= \frac{\partial(Vu)}{a\cos\varphi\partial\lambda} - \frac{V}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} \\ &= \frac{1}{a\cos\varphi\Delta\lambda}\left(\overline{\overline{V}^{Ai}\overline{u}^{Ai}}^{Di} - \frac{V}{2}\overline{\overline{u}^{Ai}}^{Di}\right) \end{aligned} \tag{8.65b}$$

(c) Calc. advection by v:

$$\begin{aligned} \mathtt{ux} &= \mathtt{ux} + \frac{\partial(Uv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{U}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} \\ &= \mathtt{ux} + \frac{1}{a\cos\varphi\Delta\lambda}\left(\overline{\overline{U}^{Aj}\overline{v\cos\varphi}^{Aj}}^{Dj} - \frac{U}{2}\overline{\overline{v\cos\varphi}^{Aj}}^{Dj}\right) \end{aligned} \tag{8.66a}$$

$$\begin{aligned} \mathtt{vx} &= \mathtt{vx} + \frac{\partial(Vv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{V}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} \\ &= \mathtt{vx} + \frac{1}{a\cos\varphi\Delta\lambda}\left(\overline{\overline{V}^{Aj}\overline{v\cos\varphi}^{Aj}}^{Dj} - \frac{V}{2}\overline{\overline{v\cos\varphi}^{Aj}}^{Dj}\right) \end{aligned} \tag{8.66b}$$

(d) Calc. advection by w:

$$\begin{aligned} \mathtt{ux} &= \mathtt{ux} + \frac{\partial(U\dot\sigma)}{\partial\sigma} - \frac{U}{2}\frac{\partial\dot\sigma}{\partial\sigma} \\ &= \mathtt{ux} + \frac{\partial(Up_h\dot\sigma)}{\partial p} - \frac{U}{2}\frac{\partial(p_h\dot\sigma)}{\partial p} \\ &= \mathtt{ux} + \frac{1}{\Delta p}\left(\overline{p_h\dot\sigma\overline{U}^{Ak}}^{Dk} - \frac{U}{2}\overline{p_h\dot\sigma}^{Dk}\right) \end{aligned} \tag{8.67a}$$

$$\begin{aligned} \mathtt{vx} &= \mathtt{vx} + \frac{\partial(V\dot\sigma)}{\partial\sigma} - \frac{V}{2}\frac{\partial\dot\sigma}{\partial\sigma} \\ &= \mathtt{vx} + \frac{\partial(Vp_h\dot\sigma)}{\partial p} - \frac{V}{2}\frac{\partial(p_h\dot\sigma)}{\partial p} \\ &= \mathtt{vx} + \frac{1}{\Delta p}\left(\overline{p_h\dot\sigma\overline{V}^{Ak}}^{Dk} - \frac{V}{2}\overline{p_h\dot\sigma}^{Dk}\right) \end{aligned} \tag{8.67b}$$

(e) Plus surface pressure gradient and viscosities to the tendency du,dv

$$\begin{aligned} \mathtt{du} &= -\mathtt{ux} - \sqrt{\sigma_h}\frac{\partial p_a}{\rho_0 a\cos\varphi\partial\lambda} + \mathtt{diffu} \\ &= -\mathtt{ux} - \mathtt{pax} * \mathtt{spbt} + \mathtt{diffu} \end{aligned} \tag{8.68a}$$

$$\begin{aligned} \mathtt{dv} &= -\mathtt{vx} - \sqrt{\sigma_h}\frac{\partial p_a}{\rho_0 a\partial\varphi} + \mathtt{diffv} \\ &= -\mathtt{vx} - \mathtt{pay} * \mathtt{spbt} + \mathtt{diffv} \end{aligned} \tag{8.68b}$$

where the advection terms:

$$\texttt{ux} = \frac{\partial(Uu)}{a\cos\varphi\partial\lambda} - \frac{U}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Uv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{U}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} + \frac{\partial(Up_h\dot{\sigma})}{\partial p} - \frac{U}{2}\frac{\partial(p_h\dot{\sigma})}{\partial p} \tag{8.69a}$$

$$\texttt{vx} = \frac{\partial(Vu)}{a\cos\varphi\partial\lambda} - \frac{V}{2}\frac{\partial u}{a\cos\varphi\partial\lambda} + \frac{\partial(Vv\cos\varphi)}{a\cos\varphi\partial\varphi} - \frac{V}{2}\frac{\partial(v\cos\varphi)}{a\cos\varphi\partial\varphi} + \frac{\partial(Vp_h\dot{\sigma})}{\partial p} - \frac{V}{2}\frac{\partial(p_h\dot{\sigma})}{\partial p} \tag{8.69b}$$

and the viscosity terms `diffu,diffv` are calculated as the following step.

Note: in the baroclinic time step, if we assume the density ρ is a constant, then

$$\begin{aligned}\frac{\partial\phi}{\partial\lambda} + \frac{1}{\rho}\frac{\partial p}{\partial\lambda} &= \frac{1}{\rho}\frac{\partial(\rho\phi)}{\partial\lambda} + \frac{\partial p}{\partial\lambda} \\ &= \frac{1}{\rho}\frac{\partial(-\rho gz + \rho gz + p_a)}{\partial\lambda} \\ &= \frac{1}{\rho}\frac{\partial p_a}{\partial\lambda}\end{aligned} \tag{8.70}$$

(f) Calc. horizontal viscosities in tau time level

$$\texttt{diffu} = \frac{1}{\sqrt{\sigma_h}}\nabla_h\cdot(\sigma_h A_m\nabla_h u) + A_m\left(\frac{1-\tan^2\varphi}{a^2}U - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial\lambda}\right) \tag{8.71a}$$

$$= A_m\left\{\left[\frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(\sigma_h\frac{\partial u}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi\partial\varphi}\left(\sigma_h\cos\varphi\frac{\partial u}{\partial\varphi}\right)\right]\frac{1}{\sqrt{\sigma_h}}\right. \tag{8.71b}$$

$$\left. + \frac{1-\tan^2\varphi}{a^2}U - \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial v}{\partial\lambda}\right\}$$

$$= \texttt{am}\left\{\left[\texttt{sdxu}*\sigma_h\overline{\frac{\partial u}{\partial\lambda}}^{Di} + \texttt{r1c}*2\sigma_h\overline{u}^{Dj+1} - \texttt{r1d}*2\sigma_h\overline{u}^{Dj}\right]/\sqrt{\sigma_h}\right. \tag{8.71c}$$

$$\left. + \texttt{cv1}*\texttt{up} - \texttt{cv2}*\overline{v}^{Di}*\texttt{spbt}\right\} \tag{8.71d}$$

$$\texttt{diffv} = \frac{1}{\sqrt{p_h}}\nabla_h\cdot(\sigma_h A_m\nabla_h v) + A_m\left(\frac{1-\tan^2\varphi}{a^2}V + \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial\lambda}\right) \tag{8.71e}$$

$$= A_m\left\{\left[\frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(\sigma_h\frac{\partial v}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi\partial\varphi}\left(\sigma_h\cos\varphi\frac{\partial v}{\partial\varphi}\right)\right]\frac{1}{\sqrt{\sigma_h}}\right. \tag{8.71f}$$

$$\left. + \frac{1-\tan^2\varphi}{a^2}V + \sqrt{\sigma_h}\frac{\tan\varphi}{a^2\cos\varphi}\frac{\partial u}{\partial\lambda}\right\}$$

$$= \texttt{am}\left\{\left[\texttt{sdxu}*\sigma_h\overline{\frac{\partial v}{\partial\lambda}}^{Di} + \texttt{r1c}*2\sigma_h\overline{v}^{Dj+1} - \texttt{r1d}*2\sigma_h\overline{v}^{Dj}\right]/\sqrt{\sigma_h}\right. \tag{8.71g}$$

$$\left. + \texttt{cv1}*\texttt{vp} + \texttt{cv2}*\overline{u}^{Di}*\texttt{spbt}\right\} \tag{8.71h}$$

6. Calc. vertical viscosity in every layer:

$$\texttt{diffu} = \texttt{diffu} + \frac{\partial}{\partial p}\left(\kappa_m \frac{g\rho_0}{\sigma_h}\frac{\partial U}{\partial z}\right) \tag{8.72a}$$

$$= \texttt{diffu} + \texttt{rdz}*\overline{\kappa_m \frac{g\rho_0}{\sigma_h}\frac{\partial U}{\partial z}}^{Dk} \tag{8.72b}$$

$$\texttt{diffv} = \texttt{diffv} + \frac{\partial}{\partial p}\left(\kappa_m \frac{g\rho_0}{\sigma_h}\frac{\partial V}{\partial z}\right) \tag{8.72c}$$

$$= \texttt{diffv} + \texttt{rdz}*\overline{\kappa_m \frac{g\rho_0}{\sigma_h}\frac{\partial V}{\partial z}}^{Dk} \tag{8.72d}$$

$$\tag{8.72e}$$

7. vertical integration of `du,dv` to `dub,dvb` (`call vinteg`)

INTERFACE:

```
subroutine readyc(umask,tmask,ivn,pmum,pmup,pbt,spbt,du,dv,dub,dvb,up,vp,cosu,  &
                  rdxt,rdxu,rdyu,rdyt,sdxu,r1c,r1d,cv1,cv2,dz,rdz,rdzw,rzu,pn,  &
                  w,pax,pay,diffu,diffv,am,kappa_m,gravr,cdbot,leapfrog_c,bcu,  &
                  bcv,imt,jmt,km,imm,jmm,kmp1,west,east,north,south,snbc,emp)
```

8.13 Module: readyc_st.f90

1. Stored `pbt_st(:,:,1)` in `pbt(:,:,tau)`
2. Calc. P in (4.6)
3. (`call upwelling`)I don't understand this part yet I guess it diagnose the vertical velocity w, but in a p-coordinate.
4. calculate horizontal advection velocities u and v in (4.7).

INTERFACE:

```
subroutine readyc_st(umask,tmask,ivn,pbt_st,du,dv,adv_u,adv_v,dub,   &
        dvb,up,vp,cosu,rdxt,rdxu,rdyu,rdyt,sdxu,r1c,r1d,cv1,cv2,dz,  &
       rdz,rdzw,rzu,pn,w,pax,pay,diffu,diffv,am,kappa_m,gravr,cdbot,  &
              bcu,bcv,imt,jmt,km,imm,jmm,kmp1,west,east,north,   &
              south,snbc,emp,t_stepu,dke_bcf,dke_fri)
```

8.14 Module: readyt.f90

Calculate the variables depended on stratification.

Algorithm:

1. Calc. reciprocal of density ρ (`call density`)
2. Store p_h, prepare to accumulate new time-averaged p_h
3. Store horizontal velocity $\sqrt{p_h}U$, $\sqrt{p_h}V\cos\varphi$
4. Calculate pressure gradients related to stratification (`call prsgrd`)

INTERFACE:

```
subroutine readyt(tmask,z,dz,rzu,t,pbt,spbt,pmtm,pmtp,bcp,rho,rhodp,umm,vmm,   &
                  ump,vmp,up,vp,cosu,rdxt,rdy,pax,pay,itn,ivn,pbxn,pbxs,pbye,  &
                  pbyw,pcxn,pcxs,pcye,pcyw,pdxn,pdxs,pdye,pdyw,imt,jmt,km,nt,  &
                          imm,jmm,kmp1,decibar,west,east,north,south,  &
                          fixp)
```

8.15 Module: readyt_st.f90

1. Set `pbt_st` to 1 if where it is zero.
2. Store `pbt_st(:,:,1)` in `pbt(:,:,1)`
3. Calc. density of the water grid. (`call density`)
4. calculate pressure gradients related to stratification (`call prsgrd`) I can't understand this part yet.

INTERFACE:

```
subroutine readyt_st(tmask,z,dz,rzu,t,pbt_st,bcp,rho,rhodp,    &
                     up,vp,cosu,rdxt,rdy,pax,pay,itn,ivn,pbxn,pbxs,pbye,   &
               pbyw,pcxn,pcxs,pcye,pcyw,pdxn,pdxs,pdye,pdyw,imt,jmt,km,nt,  &
                     imm,jmm,kmp1,decibar,west,east,north,south,  &
                     fixp)
```

8.16 Module: setcon.f90

Set A_m, A_h, κ_m and κ_h in equation 2.15 and equation 2.24.
A_m is determined by `fam` in `namelist`.

- If `fam` = 1, read from file
- Otherwise, set to `am_c` (10^7) in the `namelist`

A_h is determined by `fah` in `namelist`.

- If `fah` = 1, read from file
- Otherwise, set to `ah_c` (10^7) in the `namelist`

κ_h is determined by `fkh` in `namelist`.

- If `fkh` = 1, read κ_h from file
- If `fkh` = 2, set κ_h as a function of layers.
- Otherwise, set κ_h to `kh_c` (0.1) in the `namelist`

κ_m is determined by `fkm` in `namelist`.

- If `fkm` = 1, read κ_m from file
- Otherwise, set κ_m as a function of layers

Set restoring coefficient of γ_t and γ_s, but I can't completely understand yet.

INTERFACE:

```
subroutine setcon(fam,fah,fkm,fkh,kh_max,am_c,ah_c,km_c,kh_c,am,ah,kappa_m,kappa_
            athkdf_c,athkdf,gravr,decibar,deltap,deltat,deltas,rdeltap,  &
                    rdeltat,rdeltas,gamma_t,gamma_s,imt,jmt,km,myid,   &
                    ncpux,ncpuy,simt,sjmt,mat_myid)
```

8.17 Module: setpbt.f90

1. Calc. the initial water density ρ
2. Set bottom pressure, i.e., p_h in (2.9), to 1

INTERFACE:

```
subroutine setpbt(rho,pbt,itn,pt,ps,dz,decibar,imt,jmt,km,kmp1,nt,   &
                  fixp)
```

8.18 Module: setpn.f90

Algorithm:

1. Read in global averaged T,S stratification, 60 layers, from `pcom_ini.nc`.
2. Calc. the thickness of each layer, Δz_0 (`dz0`).
3. Calc. the pressure using an iterative method.
4. Get pressure p (`z`) and pressure increment Δp (`dz`) for each layer.
5. Get sea bottom pressure p_h (`pn`), set to 1 at land.
6. Done.

INTERFACE:

```
subroutine setpn(dz0,z0,dz,z,pn,imt,jmt,km,kmp1,itn,decibar,    &
                 phib,myid,ncpux,ncpuy,mat_myid,west,east,north,south)
```

8.19 Module: tracer.f90

compute potential temperature and salinity at `tau+1` time level
Algorithm:

1. Calc. time-average value of `pmtp, ump, vmp` for the current tracer time step.
2. Calc. vertical mass advection (`call upwelling(ump,vmp,w,...)`)
3. Diagnose velocity field (u, v, w) (`umn,vmn,wmn`) for output.

$$\texttt{umn} = \frac{\sqrt{p_h}U}{p_h} \tag{8.73a}$$

$$= \frac{\texttt{wka}}{\texttt{pbar}} \tag{8.73b}$$

$$\texttt{vmn} = \frac{\sqrt{p_h}V\cos\varphi}{p_h\cos\varphi} \tag{8.73c}$$

$$= \frac{\texttt{wkb}}{\texttt{pbar}*\texttt{cosu}} \tag{8.73d}$$

$$\texttt{wmn} = \frac{\texttt{w}}{\rho p_h g} \tag{8.73e}$$

$$= \frac{\texttt{w}*\texttt{rho}}{\texttt{pbar}*\texttt{grav}} \tag{8.73f}$$

4. Diagnose p, T, S of each grid, and calc. $1/\rho$, i.e., `rho`.
5. Calc. `phi`:

$$\texttt{phi} = -h + p_h g\int_{p_t}^{p_b}\frac{1}{\rho}dp \tag{8.74a}$$

$$= \texttt{phib/grav} + \texttt{pbar/grav}\sum_{k=1}^{n}\texttt{dz}*\texttt{rho} \tag{8.74b}$$

and `gr_mass`:

$$\texttt{gr}_{\texttt{m}}\texttt{ass} = \frac{p_h\Delta p}{g} \tag{8.75a}$$

$$= \frac{\texttt{dz}*\texttt{pbar}}{\texttt{grav}} \tag{8.75b}$$

6. Calc. new `am` by Smagorinsky Scheme (see 刘海龙, 俞永强, 李 薇, 张学洪 (2003)[P.10])

$$\texttt{am} = \texttt{cofe} * \sqrt{\texttt{dt}_\texttt{a}\texttt{m}^2 + \texttt{ds}_\texttt{a}\texttt{m}^2} \tag{8.76a}$$

$$\texttt{cofe} = \left[\frac{0.56}{\pi}\frac{1}{2}\left(a\Delta\varphi + a\cos\varphi\Delta\lambda\right)\right]^2 \tag{8.76b}$$

$$\texttt{dt}_\texttt{a}\texttt{m} = 4\frac{\partial u}{a\cos\varphi\partial\lambda} - 4\frac{\partial v}{a\partial\varphi} + \frac{v\tan\varphi}{a} \tag{8.76c}$$

$$= 2\texttt{rdxu} * 2\overline{\texttt{umn}}^{Di} - 2\texttt{rdy} * 2\overline{\texttt{vmn}}^{Dj} + \frac{\texttt{vmn}*\tan(\texttt{lat}*\texttt{torad})}{\texttt{radius}} \tag{8.76d}$$

$$\texttt{ds}_\texttt{a}\texttt{m} = \frac{\partial v}{a\cos\varphi\partial\lambda} + \frac{\partial u}{a\partial\varphi} - \frac{u\tan\varphi}{a} \tag{8.76e}$$

$$= 2\texttt{rdxu} * 2\overline{\texttt{vmn}}^{Di} + 2\texttt{rdy} * 2\overline{\texttt{umn}}^{Dj} - \frac{\texttt{umn}*\tan(\texttt{lat}*\texttt{torad})}{\texttt{radius}} \tag{8.76f}$$

$$\tag{8.76g}$$

where a is the radius of the Earth.

Note: I think the 4 in `dt_am` and `ds_am` is a mistake, it should be divided, not multiplied by 2. These may be explain the PCOM simulates a warmer temperature and fresher salinity, because the diffusion is over large.

7. Calc. temporary variables `du, dv`

$$\texttt{du} = \frac{\sqrt{p_h}U}{2} \tag{8.77a}$$

$$= \frac{\overline{\texttt{ump}}^{Aj}}{2} \tag{8.77b}$$

$$\texttt{dv} = \frac{\sqrt{p_h}V\cos\varphi}{2} \tag{8.77c}$$

$$= \frac{\overline{\texttt{vmp}}^{Ai}}{2} \tag{8.77d}$$

8. For each tracer `t`:

   (a) Calc. `wka`:

$$\begin{aligned}\texttt{wka} &= -\sqrt{p_h}U\frac{\partial \texttt{t}}{a\cos\varphi\partial\lambda} - \sqrt{p_h}V\cos\varphi\frac{\partial \texttt{t}}{a\cos\varphi\partial\varphi} - W\int_{p_t}^{p}\frac{\partial \texttt{t}}{\partial p} \\ &= -\texttt{rdxt} * \texttt{du}*\overline{\texttt{t}}^{Di} - \texttt{rdyt} * \texttt{dv}*\overline{\texttt{t}}^{j} - \texttt{rdz} * \texttt{w}*\overline{\texttt{t}}^{Dk}\end{aligned} \tag{8.78}$$

   where W is the value after `call upwelling`.

   (b) Compute the isopycnal/dipycnal mixing. (by `call isoflux`) xz and yz isopycnal diffusive flux are solved explicitly; while zz component will be solved implicitly.

   (c) Compute vertical diffusion:

$$\begin{aligned}\texttt{wka} &= \texttt{wka} + \frac{1}{2}\frac{\partial}{\partial p}\left[g\rho_0(\kappa_h + A_I K_{33})\frac{\partial T}{\partial z}\right] \\ &= \texttt{wka} + \texttt{aidifrdz}\left[\texttt{gravr}(\texttt{kappa\_h} + \texttt{ahisop}K_{33})\texttt{rdzw}\overline{T}^{Dk}\right] \\ &= \texttt{wka} + \texttt{aidifrdz}\overline{\texttt{flxb}}^{Dk}\end{aligned} \tag{8.79}$$

   where `wka` is the one `tf` in `call isoflux`.

(d) Apply sea surface heat flux boundary condition for temperature tracer.

$$\begin{aligned}
\texttt{wka}(:,:,1) &= \texttt{wka}(:,:,1) + \frac{1}{2}\frac{\partial}{\partial p}[\gamma_t(T_o - T(:,:,1))] \\
&= \texttt{wka}(:,:,1) + \frac{1}{2}\frac{\partial}{\partial p}[\gamma_t(T_o - T(:,:,1))] \\
&= \texttt{wka}(:,:,1) + \frac{1}{2}\frac{\partial}{\partial p}[\texttt{gamma_t}(T_o - T(:,:,1))] \\
&= \texttt{wka}(:,:,1) + \frac{1}{2}\frac{\partial}{\partial p}[\frac{g * \texttt{ddd}}{C_p}(T_o - T(:,:,1))] \\
&= \texttt{wka}(:,:,1) + \texttt{aidifrdz}(1) * \texttt{stf}
\end{aligned} \tag{8.80}$$

(e) Apply restoring b.c. for salinity

$$\begin{aligned}
\texttt{wka}(:,:,1) &= \texttt{wka}(:,:,1) + \frac{1}{2}\frac{\partial}{\partial p}[\gamma_s(S_o - S(:,:,1))p_h p] \\
&= \texttt{wka}(:,:,1) + \frac{1}{2}\frac{\partial}{\partial p}[\texttt{gamma_s}(S_o - S(:,:,1))\texttt{pbar} * \texttt{dz}(1)] \\
&= \texttt{wka}(:,:,1) + \texttt{aidifrdz}(1) * \texttt{stf}
\end{aligned} \tag{8.81}$$

(f) Compute T, S at tau+1 time level. Doesn't include implicit diffusion

$$\begin{aligned}
\texttt{wka} &= T + \frac{\texttt{wka}}{p}\Delta t \\
&= \texttt{t} + \frac{\texttt{wka}}{\texttt{pbar}} * \texttt{dtts2}
\end{aligned} \tag{8.82}$$

(g) Add the component due to implicit diffusion (by `call invtri`)
I don't understand this part yet.

(h) If temperature less that $-1.5°C$(`tbice`), then set to `tbice`

```
wka(i,j,k) = dmax1(tbice, wka(i,j,k))
```

where `dmax1` is a archaic form of intrinsic function `max`.

(i) Apply Asselin filter (Asselin 1972) to tracers

$$\begin{aligned}
[\mu]^{n_c} &= (1-\alpha_t)[\mu]^{n_c} + \frac{\alpha_t}{2}\left([\mu]^{n_c+1} + [\mu]^{n_c-1}\right) \\
&= \texttt{aft2}[\mu]^{n_c} + \texttt{aft1}\left([\mu]^{n_c+1} + [\mu]^{n_c-1}\right) \\
\texttt{t(taum)} &= \texttt{aft2} * \texttt{t(tau)} + \texttt{aft1} * (\texttt{wka} + \texttt{t(taum)})
\end{aligned} \tag{8.83}$$

(j) Shift time step

```
t(i,j,k,n,tau) = wka(i,j,k)
```

9. Set `leapfrog_t` to `.True.` if it is `.False.`

10. Done.

INTERFACE:

```
   =================
subroutine tracer(t,w,pmtp,pmtm,umm,vmm,ump,vmp,onbc,oncc,sdxt,rdxt,rdyt,   &
        rdz,rdzw,r1a,r1b,tmask,itn,dz,pn,bct,bcs,ddd,gamma_t,gamma_s,  &
         du,dv,ah,am,sma_c,acfl,kappa_h,gravr,leapfrog_t,c2dtts,dtts,  &
        imt,jmt,km,nt,imm,jmm,kmp1,west,east,north,south,myid,asselin_t,  &
            implicitvmix,snbc,tnbc,aft1,aft2,emp,dz0,z0,rdy,cosu,K1,   &
         K2,K3,adv_vetiso,adv_vntiso,adv_vbtiso,rhoi,e,slmxr,ahisop,   &
             athkdf,fzisop,kref,rdz0,umn,vmn,wmn,pmn,rho,phib,gr_mass, &
               bottom_h,ncpux,ncpuy,mat_myid,z,nss,rdxu,lat,bcp,  &
               dpo_adv,dpo_hdif,dpo_vdif,dpo_bc,din_adv,din_hdif,  &
        din_vdif,din_bc,dpo_imp,din_imp,dpo_ice,din_ice,dpo_ast,din_ast, &
               dsa_ast,p,pbar,mass_up)
   =================
```

8.20 Module: upwelling.f90

Compute vertical mass advection $\sigma_h\dot{\sigma}$, upward is positive, i.e., the following

$$\frac{\partial \mathtt{w}}{\partial p} = \frac{1}{p_{h0}} \int_{p_t}^{p_b} \left(\frac{\partial \mathtt{u}}{a\cos\varphi \partial\lambda} + \frac{\partial \mathtt{v}}{a\cos\varphi \partial\varphi} \right) dp + \frac{\partial \mathtt{u}}{a\cos\varphi \partial\lambda} + \frac{\partial \mathtt{v}}{a\cos\varphi \partial\varphi} \tag{8.84}$$

where u and v are input parameters.
Note that

$$\frac{\partial(\sigma_h\dot{\sigma})}{\partial\sigma} = p_h \frac{\partial(\sigma_h\dot{\sigma})}{\partial p} \tag{8.85}$$

When $\mathtt{u} = \sigma_h u = \sqrt{\sigma_h}U$, $\mathtt{v} = \sigma_h v\cos\varphi = \sqrt{\sigma_h}V\cos\varphi$, according to mass conservation equations of Eq. 2.21, w in Eq. 8.84 should be

$$\mathtt{w} = p_h\sigma_h\dot{\sigma} \tag{8.86}$$

Algorithm:

1. Calc. vertical integration of

   $$\mathtt{dp} = \int_{p_t}^{p_b} \left(\frac{\partial \mathtt{u}}{a\cos\varphi \partial\lambda} + \frac{\partial \mathtt{v}}{a\cos\varphi \partial\varphi} \right) dp \tag{8.87}$$

   and

   $$\mathtt{c} = \frac{\partial \mathtt{u}}{a\cos\varphi \partial\lambda} + \frac{\partial \mathtt{v}}{a\cos\varphi \partial\varphi} \tag{8.88}$$

   for every layer.

2. "Prognose" vertical velocity by vertically integration.

   $$\begin{aligned} \mathtt{w(k)} &= \mathtt{w(k-1)} + \left[\frac{1}{p_{h0}} \int_{p_t}^{p_b} \left(\frac{\partial \mathtt{u}}{a\cos\varphi \partial\lambda} + \frac{\partial \mathtt{v}}{a\cos\varphi \partial\varphi} \right) dp + \frac{\partial \mathtt{u}}{a\cos\varphi \partial\lambda} + \frac{\partial \mathtt{v}}{a\cos\varphi \partial\varphi} \right] dp \\ &= \mathtt{w(k-1)} + (\mathtt{dp/pn} + \mathtt{c}) * \mathtt{dz} \end{aligned} \tag{8.89}$$

INTERFACE:

```
subroutine upwelling(u,v,w,rdxt,rdyt,tmask,dz,pn,imt,jmt,km,imm,jmm,kmp1,  &
                     west,east,north,south,snbc,emp)
```

8.21 Module: vinteg

This subroutine perform the vertical integration of a 3d input parameter `wk3`, and assign the result to the 2D input parameter `wk2`. The independent difference variable is `dz(k)`, i.e., pressure difference dp. After vertical integration, it multiply `wk2` with a factor `rzu`, i.e., pressure factor $1/p_{h0}$.
Note: According to the definition of σ_h

$$\begin{aligned}\int_0^1 F(\sigma)d\sigma &= \frac{1}{p_h}\int_{p_t}^{p_b} F(\sigma(p))dp \\ &= \frac{1}{\sigma_h p_{h0}}\int_{p_t}^{p_b} F(\sigma(p))dp\end{aligned} \tag{8.90}$$

But the code use the equation

$$\int_0^1 F(\sigma)d\sigma = \frac{1}{p_{h0}}\int_{p_t}^{p_b} F(\sigma(p))dp \tag{8.91}$$

INTERFACE:

```
subroutine vinteg(wk3,wk2,ivn,dz,rzu,imt,jmt,km)
```

8.21.1 vinteg_ns

This subroutine perform the vertical integration of a 3d input parameter `wk`. The independent difference variable is `dz(k)`. It calculates both the vertical integration in two neighbor latitude grid `j` and `j-1`, and assigns the result to two 2d variables `an` and `as`, respectively. After vertical integration, it multiply `an` and `as` with a factor `rzu`.
INTERFACE:

```
subroutine vinteg_ns(wk,an,as,ivn,dz,rzu,imt,jmt,km,imm,jmm)
```

Appendices

Appendix A Derivations of Equations in Pressure-σ Coordinate

A.1 Horizontal Momentum Equations

We already have the horizontal momentum equations 2.1 in the (λ, φ, z, t) coordinate. If we change coordinate to $(\lambda, \varphi, \sigma, t)$, only vertical coordinate change, the scalar terms, like $f^*, u, v, a, \rho, \mathcal{D}_u, \mathcal{D}_v$, are independent of vertical coordinate. From equations 2.1 we have

$$\left(\frac{du}{dt}\right)_z = f^* v - \frac{1}{a\rho\cos\varphi}\left(\frac{\partial p}{\partial \lambda}\right)_z + \mathcal{D}_u, \tag{A.1a}$$

$$\left(\frac{dv}{dt}\right)_z = -f^* u - \frac{1}{a\rho}\left(\frac{\partial p}{\partial \varphi}\right)_z + \mathcal{D}_v, \tag{A.1b}$$

So we should express these four $(...)_z$ terms in the above in our new σ coordinate.
First, let's deal with $(\partial p/\partial\lambda)_z, (\partial p/\partial\varphi)_z$. Since $P(\lambda, \varphi, z, t) = P(\lambda, \varphi, \sigma, t)$, and $\sigma = \sigma(\lambda, \varphi, z, t)$, according to the chain rule of composite functions (see 谢盛刚, 李 娟, 陈秋桂 (2004)[P.68 定理 8.4.1])

$$\left(\frac{\partial p}{\partial \lambda}\right)_z = \left(\frac{\partial p}{\partial \lambda}\right)_\sigma + \frac{\partial p}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \lambda}\right)_z \tag{A.2}$$

From equation 2.9, we get $\partial p/\partial\lambda = p_h$, so

$$\left(\frac{\partial p}{\partial \lambda}\right)_z = \left(\frac{\partial p}{\partial \lambda}\right)_\sigma + p_h\left(\frac{\partial \sigma}{\partial \lambda}\right)_z \tag{A.3}$$

Now we need to know the expression of $(\partial\sigma/\partial\lambda)_z$ in $(\lambda, \varphi, \sigma, t)$ coordinate. Since

$$\sigma(\lambda, \varphi, z, t) = 0, \tag{A.4}$$

partial difference λ from both sides, we get (see 谢盛刚, 李 娟, 陈秋桂 (2004)[P.78])

$$\left(\frac{\partial \sigma}{\partial \lambda}\right)_z + \frac{\partial \sigma}{\partial z}\left(\frac{\partial z}{\partial \lambda}\right)_\sigma = 0. \tag{A.5}$$

While

$$\frac{\partial \sigma}{\partial z} = \frac{\partial}{\partial z}\left(\frac{p - p_t}{p_h}\right) = \frac{1}{p_h}\frac{\partial p}{\partial z} = -\frac{\rho g}{p_h} \tag{A.6}$$

Combine equation A.6 with equation A.5, we have

$$\left(\frac{\partial \sigma}{\partial \lambda}\right)_z = \frac{\rho}{p_h}\left(\frac{\partial \phi}{\partial \lambda}\right)_\sigma, \tag{A.7}$$

where $\phi = gz$ is the geo-potential height. Combine equation A.7 with equation A.3, we get

$$\left(\frac{\partial p}{\partial \lambda}\right)_z = \left(\frac{\partial p}{\partial \lambda}\right)_\sigma + \rho\left(\frac{\partial \phi}{\partial \lambda}\right)_\sigma. \tag{A.8}$$

Similarly, we have

$$\left(\frac{\partial p}{\partial \varphi}\right)_z = \left(\frac{\partial p}{\partial \varphi}\right)_\sigma + \rho\left(\frac{\partial \phi}{\partial \varphi}\right)_\sigma. \tag{A.9}$$

Also, we can prove the substantial difference operator in the z and σ coordinates have the same form (see appendix A.7). That is

$$\left(\frac{d}{dt}\right)_z = \left(\frac{d}{dt}\right)_\sigma \tag{A.10}$$

Substitute equations A.10, A.8, and A.9 into equation A.1, we finally have the horizontal momentum equations 2.13 in the pressure-σ coordinate

$$\left(\frac{du}{dt}\right)_\sigma = f^* v - \frac{1}{a\rho\cos\varphi}\left[\left(\frac{\partial p}{\partial\lambda}\right)_\sigma + \rho\left(\frac{\partial\phi}{\partial\lambda}\right)_\sigma\right] + \mathcal{D}_u, \tag{A.11a}$$

$$\left(\frac{dv}{dt}\right)_\sigma = -f^* u - \frac{1}{a\rho}\left[\left(\frac{\partial p}{\partial\varphi}\right)_\sigma + \rho\left(\frac{\partial\phi}{\partial\varphi}\right)_\sigma\right] + \mathcal{D}_v. \tag{A.11b}$$

A.2 Transformation of the Horizontal Momentum Equations

According to difference rule

$$\begin{aligned}\sqrt{p_h}\frac{du}{dt} &= \frac{d(\sqrt{p_h}u)}{dt} - u\frac{d\sqrt{p_h}}{dt} \\ &= \frac{dU}{dt} - u\frac{d\sqrt{p_h}}{dt}\end{aligned} \tag{A.12}$$

From the mass conservation equation (2.21), we have

$$\begin{aligned}\frac{dp_h}{dt} + p_h\nabla\cdot\mathbf{v} &= 0 \\ 2\sqrt{p_h}\frac{d\sqrt{p_h}}{dt} + p_h\nabla\cdot\mathbf{v} &= 0 \\ \frac{d\sqrt{p_h}}{dt} &= -\frac{\sqrt{p_h}}{2}\nabla\cdot\mathbf{v}\end{aligned} \tag{A.13}$$

Substitute (A.13) into (A.12), get

$$\begin{aligned}\sqrt{p_h}\frac{du}{dt} &= \frac{\partial U}{\partial t} + \mathbf{v}\cdot\nabla U + \frac{U}{2}\nabla\cdot\mathbf{v} \\ &= \frac{\partial U}{\partial t} + \nabla\cdot(U\mathbf{v}) - \frac{U}{2}\nabla\cdot\mathbf{v} \\ &= \frac{\partial U}{\partial t} + \mathcal{M}(U)\end{aligned} \tag{A.14}$$

where the advection operator $\mathcal{M}$ is defined as

$$\mathcal{M}(U) = \nabla\cdot(U\mathbf{v}) - \frac{U}{2}\nabla\cdot\mathbf{v} \tag{A.15}$$

Similar to (2.22), we express the divergence operator in pressure-σ coordinate, get

$$\begin{aligned}\mathcal{M}(U) = &\frac{1}{a\cos\varphi}\frac{\partial(uU)}{\partial\lambda} + \frac{1}{a\cos\varphi}\frac{\partial(vU\cos\varphi)}{\partial\varphi} + \frac{\partial(\dot\sigma U)}{\partial\sigma} \\ &- \frac{U}{2}\left[\frac{1}{a\cos\varphi}\frac{\partial u}{\partial\lambda} + \frac{1}{a\cos\varphi}\frac{\partial(v\cos\varphi)}{\partial\varphi} + \frac{\partial\dot\sigma}{\partial\sigma}\right] \\ = &\frac{1}{a\cos\varphi}\left[\frac{\partial(uU)}{\partial\lambda} - \frac{U}{2}\frac{\partial u}{\partial\lambda} + \frac{\partial(vU\cos\varphi)}{\partial\varphi} - \frac{U}{2}\frac{\partial(v\cos\varphi)}{\partial\varphi}\right] \\ &+ \left[\frac{\partial(\dot\sigma U)}{\partial\sigma} - \frac{U}{2}\frac{\partial\dot\sigma}{\partial\sigma}\right]\end{aligned} \tag{A.16}$$

A.3 Mass Conservation Equation

The vector form of mass conservation equation is (Yang, Liu, and Liu 1982, P.92 5.51)

$$\underbrace{\frac{\partial \rho}{\partial t}}_{\text{local change}} + \overbrace{\nabla \cdot (\rho \mathbf{v})}^{\text{divergence of mass}} = 0. \tag{A.17}$$

Express in z coordinate

$$\frac{\partial \rho}{\partial t} + \frac{1}{a\cos\varphi}\frac{\partial(\rho u)}{\partial \lambda} + \frac{1}{a\cos\varphi}\frac{\partial(\rho v\cos\varphi)}{\partial \varphi} + \frac{\partial(\rho w)}{\partial z} = 0 \tag{A.18}$$

Notice that

$$\rho = -\frac{1}{g}\frac{\partial p}{\partial z} = -\frac{1}{g}\frac{\partial p}{\partial \sigma}\frac{\partial \sigma}{\partial z} = -\frac{p_h}{g}\frac{\partial \sigma}{\partial z} \tag{A.19}$$

Substitute (A.19) into (A.18), we have

$$\begin{aligned}&\left(\frac{\partial}{\partial t}\right)_z\left(p_h\frac{\partial \sigma}{\partial z}\right) + \frac{1}{a\cos\varphi}\left[\left(\frac{\partial}{\partial \lambda}\right)_z\left(up_h\frac{\partial \sigma}{\partial z}\right) + \left(\frac{\partial}{\partial \varphi}\right)_z\left(v\cos\varphi p_h\frac{\partial \sigma}{\partial z}\right)\right]\\ &+ \frac{\partial}{\partial z}\left(wp_h\frac{\partial \sigma}{\partial z}\right) = 0\end{aligned} \tag{A.20}$$

$$\begin{aligned}&\frac{\partial \sigma}{\partial z}\left(\frac{\partial p_h}{\partial t}\right)_z + p_h\left(\frac{\partial}{\partial t}\right)_z\left(\frac{\partial \sigma}{\partial z}\right) + \frac{1}{a\cos\varphi}\left[\frac{\partial \sigma}{\partial z}\left(\frac{\partial}{\partial \lambda}\right)_z(up_h) + up_h\left(\frac{\partial}{\partial \lambda}\right)_z\left(\frac{\partial \sigma}{\partial z}\right) + \right.\\ &\left.\frac{\partial \sigma}{\partial z}\left(\frac{\partial}{\partial \varphi}\right)_z(vp_h\cos\varphi) + vp_h\cos\varphi\left(\frac{\partial}{\partial \varphi}\right)_z\left(\frac{\partial \sigma}{\partial z}\right)\right] + \frac{\partial}{\partial z}\left(wp_h\frac{\partial \sigma}{\partial z}\right) = 0\end{aligned} \tag{A.21}$$

Substitute (A.56a) into the above equation, get

$$\begin{aligned}&\frac{\partial \sigma}{\partial z}\left[\left(\frac{\partial p_h}{\partial t}\right)_\sigma + \frac{\partial p_h}{\partial \sigma}\left(\frac{\partial \sigma}{\partial t}\right)_z\right] + p_h\frac{\partial}{\partial z}\left(\frac{\partial \sigma}{\partial t}\right)_z +\\ &\frac{1}{a\cos\varphi}\left\{\frac{\partial \sigma}{\partial z}\left[\left(\frac{\partial}{\partial \lambda}\right)_\sigma(up_h) + \frac{\partial(up_h)}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \lambda}\right)_z\right] + up_h\frac{\partial}{\partial z}\left(\frac{\partial \sigma}{\partial \lambda}\right)_z\right.\\ &\left.\frac{\partial \sigma}{\partial z}\left[\left(\frac{\partial}{\partial \varphi}\right)_\sigma(vp_h\cos\varphi) + \frac{\partial(vp_h\cos\varphi)}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \varphi}\right)_z\right] + vp_h\cos\varphi\frac{\partial}{\partial z}\left(\frac{\partial \sigma}{\partial \varphi}\right)_z\right\} +\\ &\frac{\partial}{\partial z}\left(wp_h\frac{\partial \sigma}{\partial z}\right) = 0\end{aligned} \tag{A.22}$$

Multiply $\frac{\partial z}{\partial \sigma}$ to both sides of the above equation, and notice that $\frac{\partial}{\partial z}\frac{\partial z}{\partial \sigma} = \frac{\partial}{\partial \sigma}$, we get

$$\begin{aligned}&\left(\frac{\partial p_h}{\partial t}\right)_\sigma + \left(\frac{\partial \sigma}{\partial t}\right)_z\frac{\partial p_h}{\partial \sigma} + p_h\frac{\partial}{\partial \sigma}\left(\frac{\partial \sigma}{\partial t}\right)_z + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial \lambda}\right)_\sigma(up_h)\\ &+ \frac{1}{a\cos\varphi}\left[\left(\frac{\partial \sigma}{\partial \lambda}\right)_z\frac{\partial(up_h)}{\partial \sigma} + up_h\frac{\partial}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \lambda}\right)_z\right] + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial \varphi}\right)_\sigma(vp_h\cos\varphi)\\ &+ \frac{1}{a\cos\varphi}\left[\frac{\partial(vp_h\cos\varphi)}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \varphi}\right)_z + vp_h\cos\varphi\frac{\partial}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \varphi}\right)_z\right] + \frac{\partial}{\partial \sigma}\left(wp_h\frac{\partial \sigma}{\partial z}\right) = 0\end{aligned} \tag{A.23}$$

Combine pairs of terms, get

$$\left(\frac{\partial p_h}{\partial t}\right)_\sigma + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma (up_h) + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\varphi}\right)_\sigma (vp_h\cos\varphi) + \frac{\partial}{\partial\sigma}\left[p_h\left(\frac{\partial\sigma}{\partial t}\right)_z\right] + \frac{1}{a\cos\varphi}\frac{\partial}{\partial\sigma}\left[up_h\left(\frac{\partial\sigma}{\partial\lambda}\right)_z\right] + \frac{1}{a\cos\varphi}\frac{\partial}{\partial\sigma}\left[vp_h\cos\varphi\left(\frac{\partial\sigma}{\partial\varphi}\right)_z\right] + \frac{\partial}{\partial\sigma}\left(wp_h\frac{\partial\sigma}{\partial z}\right) = 0 \tag{A.24}$$

$$\left(\frac{\partial p_h}{\partial t}\right)_\sigma + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma (up_h) + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\varphi}\right)_\sigma (vp_h\cos\varphi) + \frac{\partial}{\partial\sigma}\left\{p_h\left[\left(\frac{\partial\sigma}{\partial t}\right)_z + \frac{u}{a\cos\varphi}\left(\frac{\partial\sigma}{\partial\lambda}\right)_z + \frac{v}{a}\left(\frac{\partial\sigma}{\partial\varphi}\right)_z + w\frac{\partial\sigma}{\partial z}\right]\right\} = 0 \tag{A.25}$$

According to (2.2), the terms inside [...] of the above is actually

$$\left(\frac{d\sigma}{dt}\right)_z = \dot\sigma \tag{A.26}$$

Combine the above two equations, we get

$$\left(\frac{\partial p_h}{\partial t}\right)_\sigma + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma (up_h) + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\varphi}\right)_\sigma (vp_h\cos\varphi) + \frac{\partial}{\partial\sigma}(\dot\sigma p_h) = 0 \tag{A.27}$$

Written in the vector form

$$\left(\frac{\partial p_h}{\partial t}\right)_\sigma + \nabla\cdot(p_h\mathbf{v}) = 0, \tag{A.28}$$

where

$$\nabla = \mathbf{e}_\lambda\frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma + \mathbf{e}_\varphi\frac{1}{a}\left(\frac{\partial}{\partial\varphi}\right)_\sigma + \mathbf{e}_\sigma\frac{\partial}{\partial\sigma} \tag{A.29}$$

$$\mathbf{v} = u\,\mathbf{e}_\lambda + v\,\mathbf{e}_\varphi + \dot\sigma\,\mathbf{e}_\sigma \tag{A.30}$$

$$\nabla\cdot(p_h\mathbf{v}) = \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma (up_h) + \frac{1}{a\cos\varphi}\left(\frac{\partial}{\partial\varphi}\right)_\sigma (vp_h\cos\varphi) + \frac{\partial}{\partial\sigma}(\dot\sigma p_h) \tag{A.31}$$

Eq. A.28 yields

$$\begin{aligned}
\frac{\partial p_h}{\partial t} + \mathbf{v}\cdot\nabla p_h + p_h\nabla\cdot\mathbf{v} &= 0\\
\frac{dp_h}{dt} + p_h\nabla\cdot\mathbf{v} &= 0\\
\frac{d\sigma_h p_{h0}}{dt} + \sigma_h p_{h0}\nabla\cdot\mathbf{v} &= 0\\
p_{h0}\frac{d\sigma_h}{dt} + \sigma_h\frac{dp_{h0}}{dt} + \sigma_h p_{h0}\nabla\cdot\mathbf{v} &= 0
\end{aligned} \tag{A.32}$$

Since p_{h0} is the intial sea bottom pressure, it will not change with time, nor will it be transported by advection (the initial state is static), so $dp_{h0}/dt = 0$, and we have

$$\frac{d\sigma_h}{dt} + \sigma_h\nabla\cdot\mathbf{v} = 0 \tag{A.33}$$

$$\frac{\partial\sigma_h}{\partial t} + \nabla\cdot(\sigma_h\mathbf{v}) = 0 \tag{A.34}$$

A.4 Tracer Equations

We take potential temperature T as an example.
By rule of difference

$$\frac{d(p_h T)}{dt} = p_h \frac{dT}{dt} + T\frac{dp_h}{dt} \tag{A.35}$$

From the tracer equation in z-coordinate of (2.7) the the relation ship of substantial time difference operator in z and σ coordinate of (A.10), we have

$$\left(\frac{dT}{dt}\right)_\sigma = \mathcal{Q}_T \tag{A.36}$$

And from the mass conservation equation (2.21) we have

$$\frac{\partial p_h}{\partial t} + \mathbf{v}\cdot\nabla p_h + p_h \nabla\cdot\mathbf{v} = 0 \tag{A.37}$$

$$\frac{dp_h}{dt} = -p_h \nabla\cdot\mathbf{v} \tag{A.38}$$

Substitute (A.38) and (A.36) into (A.35), we have

$$\frac{d(p_h T)}{dt} = p_h \mathcal{Q}_T - T p_h \nabla\cdot\mathbf{v} \tag{A.39}$$

And substantial change equal local change plus advective term

$$\frac{d(p_h T)}{dt} = \frac{\partial(p_h T)}{\partial t} + \mathbf{v}\cdot\nabla(p_h T) \tag{A.40}$$

Combine the above two equations, we get

$$\frac{\partial(p_h T)}{\partial t} + \nabla\cdot(p_h T\mathbf{v}) = p_h \mathcal{Q}_T \tag{A.41}$$

A.5 Geopotential Height

From the hydrostatic relation

$$\frac{\partial z}{\partial \sigma} = \frac{\partial z}{\partial p}\frac{\partial p}{\partial \sigma} = -\frac{p_h}{\rho g} \tag{A.42}$$

In vertical direction, $\frac{\partial z}{\partial \sigma} = \frac{dz}{d\sigma}$, ($z$ not related to λ, φ, t in a fixed time in vertical direction), so

$$dz = -\frac{p_h}{\rho g} d\sigma \tag{A.43}$$

$$\int_\sigma^1 dz = -\frac{1}{g}\int_\sigma^1 \frac{p_h}{\rho} d\sigma \tag{A.44}$$

$$z_b - z = -\frac{1}{g}\int_\sigma^1 \frac{p_h}{\rho} d\sigma \tag{A.45}$$

$$gz = gz_b + \int_\sigma^1 \frac{p_h}{\rho} d\sigma \tag{A.46}$$

$$\phi = gz_b + \int_\sigma^1 \frac{p_h}{\rho} d\sigma \tag{A.47}$$

A.6 Vertical Velocity Equation

Integrate the mass conservation equation (2.21) from the surface to an arbitary vertical depth

$$\int_0^\sigma \left[\frac{\partial p_h}{\partial t} + \frac{1}{a\cos\varphi}\frac{\partial (p_h u)}{\partial \lambda} + \frac{1}{a\cos\varphi}\frac{\partial (p_h v\cos\varphi)}{\partial \varphi} + \frac{\partial (p_h \dot{\sigma})}{\partial \sigma}\right] d\sigma = 0 \tag{A.48}$$

$$\frac{\partial p_h}{\partial t}\sigma + \frac{1}{a\cos\varphi}\int_0^\sigma \left[\frac{\partial (p_h u)}{\partial \lambda} + \frac{\partial (p_h v\cos\varphi)}{\partial \varphi}\right] d\sigma + p_h\dot{\sigma} - (p_h\dot{\sigma})_{\sigma=0} = 0 \tag{A.49}$$

So

$$p_h\dot{\sigma} = (p_h\dot{\sigma})_{\sigma=0} - \sigma\frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\int_0^\sigma \left[\frac{\partial (p_h u)}{\partial \lambda} + \frac{\partial (p_h v\cos\varphi)}{\partial \varphi}\right] d\sigma \tag{A.50}$$

From (B.5), we get

$$\frac{\partial p_h}{\partial t} = -\rho_0 g F_{E-P} - \frac{1}{a\cos\varphi}\left[\frac{\partial (PU_b)}{\partial \lambda} + \frac{\partial (PV_b\cos\varphi)}{\partial \varphi}\right]. \tag{A.51}$$

Substitute (A.51) into (A.50), we have

$$\begin{aligned} p_h\dot{\sigma} &= -\rho_0 g F_{E-P} + \sigma\rho_0 g F_{E-P} + \frac{1}{a\cos\varphi}\int_0^\sigma d\sigma\left[\frac{\partial (\sqrt{p_h}U_b)}{\partial \lambda} + \frac{\partial (\sqrt{p_h}V_b\cos\varphi)}{\partial \varphi}\right] \\ &\quad - \frac{1}{a\cos\varphi}\int_0^\sigma \left[\frac{\partial (\sqrt{p_h}U)}{\partial \lambda} + \frac{\partial (\sqrt{p_h}V\cos\varphi)}{\partial \varphi}\right] \\ &= \frac{1}{a\cos\varphi}\int_0^\sigma \left\{\frac{\partial [\sqrt{p_h}(U_b - U)]}{\partial \lambda} + \frac{\partial [\sqrt{p_h}(V_b - V)\cos\varphi]}{\partial \varphi}\right\} - (1-\sigma)\rho_0 g F_{E-P} \end{aligned} \tag{A.52}$$

A.7 Substantial Difference Operator

Assume a scalar filed $F(\lambda, \varphi, z, t)$, according to equation 2.2, we have

$$\left(\frac{dF}{dt}\right)_z = \left(\frac{\partial F}{\partial t}\right)_z + \frac{u}{a\cos\varphi}\left(\frac{\partial F}{\partial \lambda}\right)_z + \frac{v}{a}\left(\frac{\partial F}{\partial \varphi}\right)_z + w\frac{\partial F}{\partial z}. \tag{A.53}$$

So we need to know the expressions of the following terms in Pressure-σ coordinate

$$\left(\frac{\partial F}{\partial t}\right)_z, \left(\frac{\partial F}{\partial \lambda}\right)_z, \left(\frac{\partial F}{\partial \varphi}\right)_z \tag{A.54}$$

Since

$$F(\lambda, \varphi, z, t) = F(\lambda, \varphi, \sigma, t) \tag{A.55}$$

According to the chain rule of composite functions, difference the above equation, we get

$$\left(\frac{\partial F}{\partial t}\right)_z = \left(\frac{\partial F}{\partial t}\right)_\sigma + \frac{\partial F}{\partial \sigma}\left(\frac{\partial \sigma}{\partial t}\right)_z \tag{A.56a}$$

$$\left(\frac{\partial F}{\partial \lambda}\right)_z = \left(\frac{\partial F}{\partial \lambda}\right)_\sigma + \frac{\partial F}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \lambda}\right)_z \tag{A.56b}$$

$$\left(\frac{\partial F}{\partial \varphi}\right)_z = \left(\frac{\partial F}{\partial \varphi}\right)_\sigma + \frac{\partial F}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \varphi}\right)_z \tag{A.56c}$$

$$\frac{\partial F}{\partial z} = \frac{\partial F}{\partial \sigma}\frac{\partial \sigma}{\partial z} \tag{A.56d}$$

Substitute equations A.56a into equation A.53, after some term-arrangement, we get

$$\begin{aligned}\left(\frac{dF}{dt}\right)_z &= \left(\frac{\partial F}{\partial t}\right)_\sigma + \frac{u}{a\cos\varphi}\left(\frac{\partial F}{\partial \lambda}\right)_\sigma + \frac{v}{a}\left(\frac{\partial F}{\partial \varphi}\right)_\sigma \\ &+ \frac{\partial F}{\partial \sigma}\left[\left(\frac{\partial \sigma}{\partial t}\right)_z + \frac{u}{a\cos\varphi}\left(\frac{\partial \sigma}{\partial \lambda}\right)_z + \frac{v}{a}\left(\frac{\partial \sigma}{\partial \varphi}\right)_z + w\frac{\partial \sigma}{\partial z}\right] \\ &= \left(\frac{\partial F}{\partial t}\right)_\sigma + \frac{u}{a\cos\varphi}\left(\frac{\partial F}{\partial \lambda}\right)_\sigma + \frac{v}{a}\left(\frac{\partial F}{\partial \varphi}\right)_\sigma + \frac{\partial F}{\partial \sigma}\left(\frac{d\sigma}{dt}\right)_z \\ &= \left(\frac{\partial F}{\partial t}\right)_\sigma + \frac{u}{a\cos\varphi}\left(\frac{\partial F}{\partial \lambda}\right)_\sigma + \frac{v}{a}\left(\frac{\partial F}{\partial \varphi}\right)_\sigma + \dot{\sigma}\frac{\partial F}{\partial \sigma} \\ &= \left(\frac{dF}{dt}\right)_\sigma, \end{aligned} \tag{A.57}$$

where $\dot{\sigma}$ is defined as the vertical velocity in $(\lambda, \varphi, \sigma, t)$ coordinate.

Appendix B Derivation of Finite Difference Scheme

B.1 Finite Difference Form of Pressure Tendency

Let $\sigma = 1$ in the diagnostic equation (2.27), and combine the boundary conditions of (2.28), we get

$$(p_h\dot{\sigma})_{\sigma=1} = (p_h\dot{\sigma})_{\sigma=0} - 1 \cdot \frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\int_0^1 \left[\frac{\partial (p_h u)}{\partial \lambda} + \frac{\partial (p_h v\cos\varphi)}{\partial \varphi}\right] d\sigma \tag{B.1}$$

$$0 = -\rho_0 g F_{E-P} - \frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\int_0^1 \left[\frac{\partial (p_h u)}{\partial \lambda} + \frac{\partial (p_h v\cos\varphi)}{\partial \varphi}\right] d\sigma \tag{B.2}$$

Notice that $\frac{\partial p_h}{\partial \sigma} = 0$ (in vertical direction, p_h located at a fixed point of σ value), so

$$0 = -\rho_0 g F_{E-P} - \frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\left[\frac{\partial}{\partial \lambda}\left(\sqrt{p_h}\int_0^1 \sqrt{p_h} u d\sigma\right) + \frac{\partial}{\partial \varphi}\left(\sqrt{p_h}\cos\varphi\int_0^1 \sqrt{p_h} v d\sigma\right)\right] \tag{B.3}$$

$$0 = -\rho_0 g F_{E-P} - \frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\left[\frac{\partial}{\partial \lambda}\left(\sqrt{p_h}U_b\right) + \frac{\partial}{\partial \varphi}\left(\sqrt{p_h}V_b\cos\varphi\right)\right] \tag{B.4}$$

$$0 = -\rho_0 g F_{E-P} - \frac{\partial p_h}{\partial t} - \frac{1}{a\cos\varphi}\left[\frac{\partial (PU_b)}{\partial \lambda} + \frac{\partial (PV_b\cos\varphi)}{\partial \varphi}\right] \tag{B.5}$$

Since p_h is at T grid of Fig.1, and u, v are at U grid of Fig.1, we need to interpolate p_h from T grid to U grid by $\overline{p_h}^{i,j}$ before calculating barotropic integration U_b and V_b. That is

$$U_b = \int_0^1 \sqrt{\overline{p_h}^{i,j}} u d\sigma,\ V_b = \int_0^1 \sqrt{\overline{p_h}^{i,j}} v d\sigma. \tag{B.6}$$

So U_b is at U grid now. Then $\overline{U_b}^{j}$ will interpolate it to $j+1/2$ grid, and an "$^{\sim i}$" operation will further move it to T grid, that's what $\frac{\partial p_h}{\partial t}$ lies on. And (B.5) transform to

$$0 = -\rho_0 g F_{E-P} - \left[\frac{\partial p_h}{\partial t}\right]_{i,j} - \frac{1}{a\cos\varphi_j}\left[\overline{\overline{PU_b}^{Aj}}^{Di} + \overline{\overline{P\cos\varphi V_b}^{Ai}}^{Dj}\right], \tag{B.7}$$

where $P = \sqrt{\overline{p_h}^{i,j}}$.

B.2 Advective Operator in Horizontal Momentum Equations

From (A.15), $\mathcal{M}(U)$ is the combination of two divergence operator in pressure-σ coordinate.
Similar to (2.22), the velocity divergence in pressure-σ coordinate can be expressed as

$$\nabla\cdot\mathbf{v} = \frac{1}{a\cos\varphi}\left[\frac{\partial u}{\partial \lambda} + \frac{\partial (v\cos\varphi)}{\partial \varphi}\right] + \frac{\partial \dot{\sigma}}{\partial \sigma} \tag{B.8}$$

Since the tendency of zonal velocity is defined at U-grid (see Fig.1), we need to express $\nabla\cdot\mathbf{v}$ on U-grid, that is

$$[\nabla\cdot\mathbf{v}]_{i+1/2,j+1/2,k} = \frac{1}{a\cos\varphi_{j+\frac{1}{2}}}\left[\overline{\overline{u}^{Ai}}^{Di} + \overline{\overline{(v\cos\varphi_{jh})}^{Aj}}^{Dj}\right] + \overline{\overline{\dot{\sigma}}^{Ai,j}}^{Dk}, \tag{B.9}$$

in which

$$\underbrace{u}_{i+\frac{1}{2},j+\frac{1}{2},k} \rightarrow \underbrace{\overline{u}^{i}}_{i,j+\frac{1}{2},k} \rightarrow \underbrace{\overline{\overline{u}^{Ai}}^{Di}}_{i+\frac{1}{2},j+\frac{1}{2},k},$$

$$\underbrace{v\cos\varphi_{jh}}_{i+\frac{1}{2},j+\frac{1}{2},k} \rightarrow \underbrace{\overline{v\cos\varphi_{jh}}^{j}}_{i+\frac{1}{2},j,k} \rightarrow \underbrace{\overline{\overline{v\cos\varphi_{jh}}^{Aj}}^{Dj}}_{i+\frac{1}{2},j+\frac{1}{2},k},$$

$$\underbrace{w}_{i,j,k+\frac{1}{2}} \rightarrow \underbrace{\overline{w}^{i,j}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} \rightarrow \underbrace{\overline{\overline{w}^{Ai,j}}^{Dk}}_{i+\frac{1}{2},j+\frac{1}{2},k}.$$

Change $(\overline{u}^{i}, \overline{v\cos\varphi_{jh}}^{j}, \overline{\dot{\sigma}}^{i,j})$ to $(\overline{U}^{i}\overline{u}^{i}, \overline{U}^{j}\overline{v\cos\varphi_{jh}}^{j}, \overline{U}^{k}\overline{\dot{\sigma}}^{i,j})$ in Eq.B.9 (interpolating before multiplying), we get

$$[\nabla\cdot(U\mathbf{v})]_{i+\frac{1}{2},j+\frac{1}{2},k} = \frac{1}{a\cos\varphi_{j+\frac{1}{2}}}\left[\widetilde{\overline{U}^{i}\overline{u}^{i}}^{\,i} + \overline{U}^{j}\widetilde{\overline{(v\cos\varphi_{jh})}^{j}}^{\,j}\right] + \widetilde{\overline{U}^{k}\overline{\dot{\sigma}}^{i,j}}^{\,k}, \tag{B.10}$$

Substitute (B.8) and (B.10) into (A.15), we get the discrete form of advective operator $\mathcal{M}(U)$ on U-grid in pressure-σ coordinate

$$\begin{aligned}[\mathcal{M}(U)]_{i+\frac{1}{2},j+\frac{1}{2},k} = \frac{1}{a\cos\varphi_{j+\frac{1}{2}}}\Bigg(&\underline{\overline{U}^{i}\overline{u}^{i}}_{i} - \frac{U}{2}\underline{\overline{u}^{i}}_{i} + \underline{\overline{U}^{j}\overline{v\cos\varphi_{j+\frac{1}{2}}}^{j}}_{j} - \frac{U}{2}\underline{\overline{v\cos\varphi_{j+\frac{1}{2}}}^{j}}_{j}\Bigg)\\ &+ \underline{\overline{U}^{k}\overline{\dot{\sigma}}^{i,j}}_{k} - \frac{U}{2}\underline{\overline{\dot{\sigma}}^{i,j}}_{k}\end{aligned} \tag{B.11}$$

Appendix C Parameterization of Viscous Stress Terms in Momentum Equation in Pressure-σ Coordinate

We take $\mathcal{D}_u$ for an example.
From Versteeg and Malalasekera (2007)[P.23] we know that viscous terms can be expressed as

$$\mathcal{D}_u = \nabla \cdot (\mu \nabla u) + s_u \tag{C.1}$$

Now we would like to know the expression of $\nabla \cdot (\mu \nabla u)$. From (2.14) we have

$$\nabla = \mathbf{e}_\lambda \frac{1}{a\cos\varphi}\frac{\partial}{\partial\lambda} + \mathbf{e}_\varphi \frac{1}{a}\frac{\partial}{\partial\varphi} + \mathbf{e}_\sigma \frac{\partial}{\partial\sigma} \tag{C.2}$$

So

$$\mu\nabla u = \mathbf{e}_\lambda \frac{\mu}{a\cos\varphi}\frac{\partial u}{\partial\lambda} + \mathbf{e}_\varphi \frac{\mu}{a}\frac{\partial u}{\partial\varphi} + \mathbf{e}_\sigma \mu\frac{\partial u}{\partial\sigma} \tag{C.3}$$

Similar to (2.22), we have

$$\begin{aligned}\nabla \cdot (\mu\nabla u) &= \frac{1}{a\cos\varphi}\left[\frac{\partial}{\partial\lambda}\left(\frac{\mu}{a\cos\varphi}\frac{\partial u}{\partial\lambda}\right) + \frac{\partial}{\partial\varphi}\left(\frac{\mu\cos\varphi}{a}\frac{\partial u}{\partial\varphi}\right)\right] + \frac{\partial}{\partial\sigma}\left(\mu\frac{\partial u}{\partial\sigma}\right) \\ &= \frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(\mu\frac{\partial u}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi}\frac{\partial}{\partial\varphi}\left(\mu\cos\varphi\frac{\partial u}{\partial\varphi}\right) + \frac{\partial}{\partial\sigma}\left(\mu\frac{\partial u}{\partial\sigma}\right)\end{aligned} \tag{C.4}$$

If we let $\mu = p_h A_h$, we get

$$\nabla \cdot (p_h A_h \nabla u) = \frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(p_h A_h\frac{\partial u}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi}\frac{\partial}{\partial\varphi}\left(p_h A_h\cos\varphi\frac{\partial u}{\partial\varphi}\right) + \frac{\partial}{\partial\sigma}\left(p_h A_h\frac{\partial u}{\partial\sigma}\right) \tag{C.5}$$

If in vertical direction, we use a different parameter A_v, and assume it satisfy

$$\frac{A_h}{A_v} = \left(\frac{\partial\sigma}{\partial z}\right)^2 \tag{C.6}$$

From the hydrostatic relationship in pressure-σ coordinate of (2.20), the above equation change to

$$A_h = \frac{\rho^2 g^2}{p_h^2} A_v \tag{C.7}$$

Substitute the above to the last term of (C.5), we get

$$\nabla \cdot (p_h A_h \nabla u) = \frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(p_h A_h\frac{\partial u}{\partial\lambda}\right) + \frac{1}{a^2\cos\varphi}\frac{\partial}{\partial\varphi}\left(p_h A_h\cos\varphi\frac{\partial u}{\partial\varphi}\right) + \frac{\partial}{\partial\sigma}\left(\frac{\rho^2 g^2}{p_h}A_v\frac{\partial u}{\partial\sigma}\right) \tag{C.8}$$

Finally, multiply both sides with $1/p_h$, we get

$$\begin{aligned}\frac{1}{p_h}\nabla \cdot (p_h A_h \nabla u) &= \frac{1}{p_h}\frac{1}{a^2\cos^2\varphi}\frac{\partial}{\partial\lambda}\left(p_h A_h\frac{\partial u}{\partial\lambda}\right) + \frac{1}{p_h}\frac{1}{a^2\cos\varphi}\frac{\partial}{\partial\varphi}\left(p_h A_h\cos\varphi\frac{\partial u}{\partial\varphi}\right) \\ &\quad + \frac{1}{p_h}\frac{\partial}{\partial\sigma}\left(\frac{\rho^2 g^2}{p_h}A_v\frac{\partial u}{\partial\sigma}\right)\end{aligned} \tag{C.9}$$

The right side correspond to the first two terms of the right side of (2.15a).

Appendix D Coriolis Adjustment

The derivation of Coriolis adustment mentioned in 刘海龙, 俞永强, 李 薇, 张学洪 (2003)[P.26].
The horizontal momentum equation can be expressed as

$$\frac{\partial u}{\partial t} = -fv + r_u \tag{D.1a}$$

$$\frac{\partial v}{\partial t} = fu + r_v \tag{D.1b}$$

where r_u and r_v are the combination of the terms excluding Coriolis-associated terms. Write the above equations in a semi-implicit time scheme (i.e., express Coriolis associated terms in a implicit scheme, and other terms in a explicit scheme), we get the following two types:
(1) Euler-back time scheme:

$$\frac{u^{n+1} - u^n}{\Delta t} = -f\frac{v^{n+1} + v^n}{2} + r_u^n \tag{D.2a}$$

$$\frac{v^{n+1} - v^n}{\Delta t} = f\frac{u^{n+1} + u^n}{2} + r_v^n \tag{D.2b}$$

We use a substitution method to solve the above algebraic equation

$$u^{n+1} = u^n - \frac{f\Delta t}{2}(v^n + u^{n+1}) + \Delta t r_u^n \tag{D.3a}$$

$$= u^n - \frac{f\Delta t}{2}v^n - \frac{f\Delta t}{2}\left(v^n + \frac{f\Delta t}{2}(u^n + u^{n+1}) + \Delta t r_v^n\right) + \Delta t r_u^n \tag{D.3b}$$

$$= u^n - f\Delta t v^n - \left(\frac{f\Delta t}{2}\right)^2 u^n - \left(\frac{f\Delta t}{2}\right)^2 u^{n+1} - \frac{f\Delta^2 t}{2}r_v^n + \Delta t r_u^n \tag{D.3c}$$

$$= \left(\frac{f\Delta t}{2}\right)^2 u^{n+1} + \Delta t(-fv^n + r_u^n) + \left[1 + \left(\frac{f\Delta t}{2}\right)^2\right] u^n - 2\left(\frac{f\Delta t}{2}\right)^2 u^n - \frac{f\Delta^2 t}{2}r_v^n \tag{D.3d}$$

$$= \left(\frac{f\Delta t}{2}\right)^2 u^{n+1} + \Delta t(-fv^n + r_u^n) + \left[1 + \left(\frac{f\Delta t}{2}\right)^2\right] u^n - \frac{f\Delta^2 t}{2}(fu^n + r_v^n) \tag{D.3e}$$

$$= -\varepsilon^2 u^{n+1} + \Delta t R_u^n + (1+\varepsilon^2)u^n - \varepsilon\Delta t R_v^n \tag{D.3f}$$

$$= -\varepsilon^2 u^{n+1} + (1+\varepsilon^2)u^n + \Delta t(R_u^n - \varepsilon R_v^n) \tag{D.3g}$$

where $\varepsilon = f\Delta t/2$, $R_u^n = -fv^n + r_u^n$, $R_v^n = fu^n + r_v^n$.
So we get

$$(1+\varepsilon^2)\frac{u^{n+1} - u^n}{\Delta t} = R_u^n - \varepsilon R_v^n \tag{D.4}$$

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{R_u^n - \varepsilon R_v^n}{1+\varepsilon^2} \tag{D.5}$$

In a similar way, we can get

$$\frac{v^{n+1} - v^n}{\Delta t} = \frac{R_v^n + \varepsilon R_u^n}{1+\varepsilon^2} \tag{D.6}$$

(2) Leapfrog time scheme:

$$\frac{u^{n+1}-u^{n-1}}{2\Delta t} = -f\frac{u^{n+1}+u^{n-1}}{2} + r_u^{n-1} \tag{D.7a}$$

$$\frac{v^{n+1}-v^{n-1}}{2\Delta t} = f\frac{v^{n+1}+v^{n-1}}{2} + r_v^{n-1} \tag{D.7b}$$

Again, use the substitution method to solve the above algebraic equation:

$$u^{n+1} = u^{n-1} - f\Delta t(u^{n-1}+u^{n+1}) + 2\Delta t r_u^{n-1} \tag{D.8a}$$

$$= u^{n-1} - f\Delta t u^{n-1} - f\Delta t\left(u^{n-1} + f\Delta t(u^{n-1}+u^{n+1}) + 2\Delta t r_v^{n-1}\right) + 2\Delta t r_u^{n-1} \tag{D.8b}$$

$$= u^{n-1} - 2f\Delta t u^{n-1} - (f\Delta t)^2 u^{n-1} - (f\Delta t)^2 u^{n+1} - 2f\Delta^2 t r_v^{n-1} + 2\Delta t r_u^{n-1} \tag{D.8c}$$

$$= -\varepsilon^2 u^{n+1} + 2\Delta t(-f u^{n-1} + r_u^{n-1}) + (1+\varepsilon^2)u^{n-1} - 2(f\Delta t)^2 u^{n-1} - 2f\Delta^2 t r_v^{n-1} \tag{D.8d}$$

$$= -\varepsilon^2 u^{n+1} + 2\Delta t(-f u^{n-1} + r_u^{n-1}) + (1+\varepsilon^2)u^{n-1} - 2f\Delta^2 t(f u^{n-1} + r_v^{n-1}) \tag{D.8e}$$

$$= -\varepsilon^2 u^{n+1} + 2\Delta t R_u^{n-1} + (1+\varepsilon^2)u^{n-1} - 2f\Delta^2 t R_v^{n-1} \tag{D.8f}$$

$$= -\varepsilon^2 u^{n+1} + (1+\varepsilon^2)u^{n-1} + 2\Delta t(R_u^{n-1} - \varepsilon R_v^{n-1}) \tag{D.8g}$$

where $\varepsilon = f\Delta t$(differenct from (1)), $R_u^{n-1} = -f u^{n-1} + r_u^{n-1}$, $R_v^{n-1} = f u^{n-1} + r_v^{n-1}$.
So we get

$$(1+\varepsilon^2)\frac{u^{n+1}-u^{n-1}}{2\Delta t} = R_u^{n-1} - \varepsilon R_v^{n-1} \tag{D.9}$$

$$\frac{u^{n+1}-u^{n-1}}{2\Delta t} = \frac{R_u^{n-1} - \varepsilon R_v^{n-1}}{1+\varepsilon^2} \tag{D.10}$$

In a similar way, we can get

$$\frac{v^{n+1}-v^{n-1}}{2\Delta t} = \frac{R_v^{n-1} + \varepsilon R_u^{n-1}}{1+\varepsilon^2} \tag{D.11}$$

Appendix E Geopotential Height

In a p-σ coordinate, the pressure gradient will convert to the gradient of geopotential height.
Since

$$\frac{\partial \phi}{\partial p} = -\frac{1}{\rho} \tag{E.1}$$

Integrated from a specific depth to sea bottom

$$\begin{aligned} \int_p^{p_b} \frac{\partial \phi}{\partial p} dp &= -\int_p^{p_b} \frac{1}{\rho} dp \\ \phi_b - \phi &= -\int_p^{p_b} \frac{1}{\rho} dp \\ \phi_b &= \phi - \int_p^{p_b} \frac{1}{\rho} dp \\ \phi_b &= -\frac{p}{\rho} - \int_p^{p_b} \frac{1}{\rho} dp \end{aligned} \tag{E.2}$$

Note: In the code of `prsgrd.f90`,

$$\phi_b = -\frac{p_0}{\rho} - \int_p^{p_b} \frac{1}{\rho} dp \tag{E.3}$$

which use the initial pressure p_0 instead of p, and this is because the sea surface height is usually very small that the sea bottom, and $gz \approx g(z + z')$, so $\phi_b \approx \phi_{b0}$

References

Richard Asselin. Frequency filter for time integrations. *Monthy Weather Review*, 100:487--490, 1972.

Kirk Bryan. Climate and the ocean circulation. *Monthly Weather Review*, 97:806--827, 1969. doi: 10.1175/1520-0493(1969)097<0806:CATOC>2.3.CO;2. URL http://dx.doi.org/10.1175/1520-0493(1969)097<0806:CATOC>2.3.CO;2.

Rui Xin Huang. Real freshwater flux as a natural boundary condition for the salinity balance and thermohaline circulation forced by evaporation and precipitation. *J. Phys. Oceanogr.*, 23(11):2428--2446, November 1993. ISSN 0022-3670. URL http://dx.doi.org/10.1175/1520-0485(1993)023<2428:RFFAAN>2.0.CO;2.

David R. Jackett and Trevor J. Mcdougall. Minimal adjustment of hydrographic profiles to achieve static stability. *J. Atmos. Oceanic Technol.*, 12(2):381--389, April 1995. ISSN 0739-0572. URL http://dx.doi.org/10.1175/1520-0426(1995)012<0381:MAOHPT>2.0.CO;2.

Hailong Liu, Yongqiang Yu, Wei Li, and Xuehong Zhang. Lasg/iap climate system ocean model (licom1.0) (in chinese), 2003.

Frank J. Millero and Alain Poisson. International one-atmosphere equation of state of seawater. *Deep Sea Research Part A. Oceanographic Research Papers*, 28(6):625 -- 629, 1981. ISSN 0198-0149. doi: http://dx.doi.org/10.1016/0198-0149(81)90122-9. URL http://www.sciencedirect.com/science/article/pii/0198014981901229.

Note. *Frequently Used Mathematics*. ASUS laptop, 2014.

Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

Dasheng Yang, Yubin Liu, and Shigua Liu. *Dynamic Meteorology (revised version, in Chinese)*. China Meteorological Press, 1982.

Yu Zhang. *Oceanic General Circulation Model Based on Mass and Energy Conservation and the Balance of Energy in the Oceanic General Circulation (in Chinese)*. PhD thesis, University of Chinese Academy of Sciences, 151pp, 2013.

Yu Zhang, YiHua Lin, and RuiXin Huang. A climatic dataset of ocean vertical turbulent mixing coefficient based on real energy sources. *Science China Earth Sciences*, 57(10):2435--2446, 2014.

刘海龙, 俞永强, 李 薇, 张学洪. Lasg/iap 气候系统海洋模式 (licom1.0), 2003.

曾庆存, 张学洪. 球面上斜压原始方程组保持总有效能量守恒的差分格式. 大气科学, 11(2):113--127, 1987.

谢盛刚, 李 娟, 陈秋桂. 微积分（下）. 科学出版社, 2004.

Index