DieCast Ocean General Circulation Model

Yu-Chiao Liang

08/29/2008

Outline

Primitive Equations in Diecast

The primitive equations of our ocean general circulation model are,

- Conservation of mass:

$$\nabla \cdot \mathbf{V} = \mathbf{0}, \tag{1}$$

- Horizontal momentun equations:

$$\frac{\partial u}{\partial t} = -\nabla \cdot u\mathbf{V} + \mathbf{fv} - \frac{\mathbf{1}}{\rho_{\mathbf{0}}}\frac{\partial \mathbf{p}}{\partial \mathbf{x}} + \nabla_{\mathbf{h}} \cdot \mathbf{A_h} \nabla_{\mathbf{h}} \mathbf{u} + \frac{\partial}{\partial \mathbf{z}}(\mathbf{A_v}\frac{\partial \mathbf{u}}{\partial \mathbf{z}}), \tag{2}$$

$$\frac{\partial v}{\partial t} = -\nabla \cdot v\mathbf{V} - \mathbf{fu} - \frac{\mathbf{1}}{\rho_{\mathbf{0}}}\frac{\partial \mathbf{p}}{\partial \mathbf{y}} + \nabla_{\mathbf{h}} \cdot \mathbf{A_h} \nabla_{\mathbf{h}} \mathbf{v} + \frac{\partial}{\partial \mathbf{z}}(\mathbf{A_v}\frac{\partial \mathbf{v}}{\partial \mathbf{z}}), \tag{3}$$

Primitive Equations in Diecast

- Conservation of scalar(salt or potential temperature)

$$\frac{\partial S}{\partial t} = -\nabla \cdot S\mathbf{V} + \nabla_{\mathbf{h}} \cdot \mathbf{K_h} \nabla_{\mathbf{h}} \mathbf{S} + \frac{\partial}{\partial \mathbf{z}}(\mathbf{K_v} \frac{\partial \mathbf{S}}{\partial \mathbf{z}}), \tag{4}$$

- Hydrostatic equation:

$$\frac{\partial p}{\partial z} = -(\rho - \bar{\rho})g, \tag{5}$$

- Equation of state:

$$\rho = \rho(S, T), \tag{6}$$

Primitive Equations in Diecast

where u and v are the velocity components in x and y directions, the velocity vector $\mathbf{V} = (\mathbf{x}, \mathbf{y}, \mathbf{w})$. f is Coriolis parameter, ρ_0 is the mean density, $\bar{\rho}$ is the horizontal average of density at depth z, p is the pressure, $A_h and A_v$ are the horizontal and vertical eddy viscosity, S is the salinity, $K_h and K_v$ are the horizontal and vertical eddy diffusivity, T is the potential temperature.

Note that equation of state (6) here is set to be function of salinity S and temperature T. If we are running a atmospheric circulation model, $p = \rho RT$ is replaced.

Hydrostatic approximation

Hydrostatic approximation gives us good computational strategy by integrating (5) from the depth z to the surface:

$$p = p_s + p_b, \qquad p_b = g \int_z^0 \rho dz, \tag{7}$$

where p_s is the surface pressure due to atmosphere or other phenomena.

Thus, we can get pressure field over whole domain that we considered.

Boundary Condition

The boundary conditions for the vertical velocity component w are based on rigid-lid approximation and the condition of no normal flow at the bottom. We set:

$$w_0 = 0, \tag{8}$$

$$w_{-h} = 0, \tag{9}$$

where the subscripts 0 and $-h$ denote at the sea surface and the bottom respectively. Integating (1) and use boundary conditions above,

$$\int_{-h}^{0} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} dz = -w_0 + w_{-h} = 0. \tag{10}$$

Note that we can set $w_{-h} = 0$ by using concept like σ-coordinate, which provide a uniform z-level spacing that is more easy for computation.

Definition of Operators

Before we go forward, for convinence, we rewrite our governing equations (2), (3), and (4) as the form:

$$
\begin{aligned}
\frac{\partial u}{\partial t} &+ \mathcal{L}u - \left(f + \frac{u \tan\phi}{R}\right) v = -\frac{1}{\rho_0}\frac{\partial p}{\partial x} + \mathcal{D}_m u + \frac{\partial}{\partial z}\left(K_m \frac{\partial u}{\partial z}\right), \qquad (11)\\
\frac{\partial v}{\partial t} &+ \mathcal{L}v + \left(f + \frac{u \tan\phi}{R}\right) u = -\frac{1}{\rho_0}\frac{\partial p}{\partial y} + \mathcal{D}_m v + \frac{\partial}{\partial z}\left(K_m \frac{\partial v}{\partial z}\right), \qquad (12)\\
\frac{\partial T}{\partial t} &+ \mathcal{L}T = \mathcal{D}_h T + \frac{\partial}{\partial z}\left(K_h \frac{\partial T}{\partial z}\right), \qquad (13)
\end{aligned}
$$

where $\mathcal{L}$ is advection operator, $\mathcal{D}_m$ is the momentum diffusion operator, and $\mathcal{D}_h$ is the temperature(or any scalar) diffusion operator.

Definition of Operators

Operators are defined as below:

$$\mathcal{L} = \vec{V}\cdot\nabla = u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}, \tag{14}$$

$$\mathcal{D}_m = \nabla_m \cdot A_m \nabla_m = \frac{\partial}{\partial x}A_m(x,y,z)\frac{\partial}{\partial x} + \frac{\partial}{\partial y}A_m(x,y,z)\frac{\partial}{\partial y}, \tag{15}$$

$$\mathcal{D}_h = \nabla_h \cdot A_h \nabla_h = \frac{\partial}{\partial x}A_h(x,y,z)\frac{\partial}{\partial x} + \frac{\partial}{\partial y}A_h(x,y,z)\frac{\partial}{\partial y}. \tag{16}$$

One of our goal is to treat these operators well, so that we need a 'good' numerical scheme for them.

Definition of Operators

Operators are defined as below:

$$\mathcal{L} = \vec{V} \cdot \nabla = u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}, \tag{14}$$

$$\mathcal{D}_m = \nabla_m \cdot A_m \nabla_m = \frac{\partial}{\partial x} A_m(x,y,z)\frac{\partial}{\partial x} + \frac{\partial}{\partial y} A_m(x,y,z)\frac{\partial}{\partial y}, \tag{15}$$

$$\mathcal{D}_h = \nabla_h \cdot A_h \nabla_h = \frac{\partial}{\partial x} A_h(x,y,z)\frac{\partial}{\partial x} + \frac{\partial}{\partial y} A_h(x,y,z)\frac{\partial}{\partial y}. \tag{16}$$

One of our goal is to treat these operators well, so that we need a 'good' numerical scheme for them.

Time Discretization

We adopt general 'Leap-frog' scheme in time advancement as a basis. Consider general ODE system model:

$$\frac{dU}{dt} = S(U(x,t), x, t), \tag{17}$$

where S is some source term that is known, U is variable that we compute. Discretized equation is written as:

$$\frac{f_i^{n+1} - f_i^{n-1}}{2\Delta t} = S(f(x_i, t_n), x_i, t_n), \tag{18}$$

where superscript $n-1$ and $n+1$ denote what time step it is, subscript i denotes at what grid point.

Time Discretization

Discretized form of equations (2), (3), and (4) can therefore be written as:

$$
\begin{aligned}
u^{n+1} &= u^{n-1} + 2\Delta t \\
&\times \left[\mathcal{D}_m u^{n-1} - \mathcal{L} u^n + f(\gamma v^{n+1} + (1-\gamma) v^n) - \frac{1}{\rho_0}\frac{\partial p^n}{\partial x} + \frac{\partial}{\partial z}\left(K_m \frac{\partial u^{n-1}}{\partial z}\right)\right], \\
v^{n+1} &= v^{n-1} + 2\Delta t \\
&\times \left[\mathcal{D}_m v^{n-1} - \mathcal{L} v^n + f(\gamma u^{n+1} + (1-\gamma) u^n) - \frac{1}{\rho_0}\frac{\partial p^n}{\partial y} + \frac{\partial}{\partial z}\left(K_m \frac{\partial v^{n-1}}{\partial z}\right)\right], \\
T^{n+1} &= T^{n-1} + 2\Delta t \times \left[\mathcal{D}_h T^{n-1} - \mathcal{L} T^n + \frac{\partial}{\partial z}\left(K_h \frac{\partial T^{n-1}}{\partial z}\right)\right], \qquad (19)
\end{aligned}
$$

Time Discretization

Now, we introduce concept of fractional step by setting:

$$u^{n+1} = \tilde{u}^{n+1} + (\hat{u}^{n+1} - \tilde{u}^{n+1}) + \Delta u^{n+1} \equiv \hat{u}^{n+1} + \Delta u^{n+1}, \tag{20}$$

$$v^{n+1} = \tilde{v}^{n+1} + (\hat{v}^{n+1} - \tilde{v}^{n+1}) + \Delta v^{n+1} \equiv \hat{v}^{n+1} + \Delta v^{n+1}, \tag{21}$$

$$T^{n+1} = \tilde{T}^{n+1} + (\hat{T}^{n+1} - \tilde{T}^{n+1}) \equiv \hat{T}^{n+1}, \tag{22}$$

where $\tilde{u}^{n+1}$, $\tilde{v}^{n+1}$,and $\tilde{T}^{n+1}$ are updated by momentun equations without Coriolis terms and temperature equation.

Note that the brackets '()' means updation including Coriolis term.

$\hat{T}^{n+1}$ is invalid since Coriolis force has no effect on temperature (or other scalar variables?). Δu^{n+1} and Δv^{n+1} are used for correction by pressure equation.

Time Discretization

Now, we introduce concept of fractional step by setting:

$$u^{n+1} = \tilde{u}^{n+1} + (\hat{u}^{n+1} - \tilde{u}^{n+1}) + \Delta u^{n+1} \equiv \hat{u}^{n+1} + \Delta u^{n+1}, \tag{20}$$

$$v^{n+1} = \tilde{v}^{n+1} + (\hat{v}^{n+1} - \tilde{v}^{n+1}) + \Delta v^{n+1} \equiv \hat{v}^{n+1} + \Delta v^{n+1}, \tag{21}$$

$$T^{n+1} = \tilde{T}^{n+1} + (\hat{T}^{n+1} - \tilde{T}^{n+1}) \equiv \hat{T}^{n+1}, \tag{22}$$

where $\tilde{u}^{n+1}$, $\tilde{v}^{n+1}$,and $\tilde{T}^{n+1}$ are updated by momentun equations without Coriolis terms and temperature equation.
Note that the brackets '()' means updation including Coriolis term.

$\hat{T}^{n+1}$ is invalid since Coriolis force has no effect on temperature (or other scalar variables?). Δu^{n+1} and Δv^{n+1} are used for correction by pressure equation.

Time discretization

So now we have discretized equations for variables $\tilde{u}^{n+1}$, $\tilde{v}^{n+1}$,and $\tilde{T}^{n+1}$:

$$\tilde{u}^{n+1} = u^{n-1} + 2\Delta t\left[\mathcal{D}_m u^{n-1} - \mathcal{L}u^n - \frac{1}{\rho_0}\frac{\partial p^n}{\partial x} + \frac{\partial}{\partial z}\left(K_m \frac{\partial u^{n-1}}{\partial z}\right)\right], \quad (23)$$

$$\tilde{v}^{n+1} = v^{n-1} + 2\Delta t\left[\mathcal{D}_m v^{n-1} - \mathcal{L}v^n - \frac{1}{\rho_0}\frac{\partial p^n}{\partial y} + \frac{\partial}{\partial z}\left(K_m \frac{\partial v^{n-1}}{\partial z}\right)\right], \quad (24)$$

$$\tilde{T}^{n+1} = T^{n-1} + 2\Delta t\left[\mathcal{D}_h T^{n-1} - \mathcal{L}T^n + \frac{\partial}{\partial z}\left(K_h \frac{\partial T^{n-1}}{\partial z}\right)\right]. \quad (25)$$

$\tilde{u}^{n+1}$, $\tilde{v}^{n+1}$,and $\tilde{T}^{n+1}$ can be updated since right hand sides of these equations are knownn variables.

Updation of Coriolis Term

Coriolis terms need to be treated serious, especially when solving fluid problems connected to geophysics.
Let variables $\hat{u}$ and $\hat{v}$ satisfies updating equations defined as below:

$$\hat{u}^{n+1} = \tilde{u}^{n+1} + 2\Delta t \times f v^n + correction, \tag{26}$$

$$\hat{v}^{n+1} = \tilde{v}^{n+1} - 2\Delta t \times f u^n + correction, \tag{27}$$

this is the basic ideal of updation. In our model, we accomplish updation by two steps, which includes 'implicit' updation.
Consider trapezoidal treatment for Coriolis terms, that is:

$$\begin{aligned} \hat{u}^{n+1} &= \tilde{u}^{n+1} + 2\Delta t f\left(\frac{\tilde{v}^{n+1} + v^{n-1}}{2}\right) \\ &= \tilde{u}^{n+1} + \Delta t f v^{n-1} + \Delta f \tilde{v}^{n+1} \end{aligned} \tag{28}$$

$$\begin{aligned} \hat{v}^{n+1} &= \tilde{v}^{n+1} - 2\Delta t f\left(\frac{\tilde{u}^{n+1} + u^{n-1}}{2}\right) \\ &= \tilde{v}^{n+1} + f\Delta t u^{n-1} + f\Delta t \tilde{u}^{n+1} \end{aligned} \tag{29}$$

Updation of Coriolis Term

Coriolis terms need to be treated serious, especially when solving fluid problems connected to geophysics.
Let variables $\hat{u}$ and $\hat{v}$ satisfies updating equations defined as below:

$$\hat{u}^{n+1} = \tilde{u}^{n+1} + 2\Delta t \times fv^n + correction, \tag{26}$$
$$\hat{v}^{n+1} = \tilde{v}^{n+1} - 2\Delta t \times fu^n + correction, \tag{27}$$

this is the basic ideal of updation. In our model, we accomplish updation by two steps, which includes 'implicit' updation.
Consider trapezoidal treatment for Coriolis terms, that is:

$$\begin{aligned} \hat{u}^{n+1} &= \tilde{u}^{n+1} + 2\Delta t f(\frac{\tilde{v}^{n+1} + v^{n-1}}{2}) \\ &= \tilde{u}^{n+1} + \Delta t f v^{n-1} + \Delta f \tilde{v}^{n+1} \end{aligned} \tag{28}$$
$$\begin{aligned} \hat{v}^{n+1} &= \tilde{v}^{n+1} - 2\Delta t f(\frac{\tilde{u}^{n+1} + u^{n-1}}{2}) \\ &= \tilde{v}^{n+1} + f\Delta t u^{n-1} + f\Delta t \tilde{u}^{n+1} \end{aligned} \tag{29}$$

Updation of Coriolis Term

First step is updating using first part of RHS of the equations (28) and (29) by the equations below:

$$QU \equiv \hat{u}_*^{n+1} = \tilde{u}^{n+1} + f\Delta t \times v^{n-1}, \tag{30}$$

$$QV \equiv \hat{v}_*^{n+1} = \tilde{v}^{n+1} - f\Delta t \times u^{n-1}, \tag{31}$$

where $*$ denotes intermediate variable. Second step is to update variables $\hat{u}^{n+1}, \hat{v}^{n+1}$. After first step, we need update second part of RHS of the equations (28) and (29). So, we can write down:

$$\hat{u}^{n+1} = \hat{u}_*^{n+1} + f\Delta t \times \hat{v}^{n+1}, \tag{32}$$

$$\hat{v}^{n+1} = \hat{v}_*^{n+1} - f\Delta t \times \hat{u}^{n+1}, \tag{33}$$

Updation of Coriolis Term

Substitiude (33) into (32) and (32) into (33), we can get equations after easy algebraic calculation:

$$\tilde{u}^{n+1} = \frac{1}{1+(f\Delta t)^2}[\hat{u}_*^{n+1} + f\Delta t \times \hat{v}_*^{n+1}] = \frac{1}{1+(f\Delta t)^2}(QU + f\Delta t \times QV), \quad (34)$$

$$\tilde{v}^{n+1} = \frac{1}{1+(f\Delta t)^2}[\hat{v}_*^{n+1} - f\Delta t \times \hat{u}_*^{n+1}] = \frac{1}{1+(f\Delta t)^2}(QV - f\Delta t \times QU). \quad (35)$$

It seems that we update Coriolis terms twice by similar numerical scheme, but why we do that?

- stability,
- non-oscillation,
- better performance,
- numerical experiments...

Updation of Coriolis Term

Substitiude (33) into (32) and (32) into (33), we can get equations after easy algebraic calculation:

$$\tilde{u}^{n+1} = \frac{1}{1+(f\Delta t)^2}[\hat{u}_*^{n+1} + f\Delta t \times \hat{v}_*^{n+1}] = \frac{1}{1+(f\Delta t)^2}(QU + f\Delta t \times QV), \quad (34)$$

$$\tilde{v}^{n+1} = \frac{1}{1+(f\Delta t)^2}[\hat{v}_*^{n+1} - f\Delta t \times \hat{u}_*^{n+1}] = \frac{1}{1+(f\Delta t)^2}(QV - f\Delta t \times QU). \quad (35)$$

It seems that we update Coriolis terms twice by similar numerical scheme, but why we do that?

- stability,
- non-oscillation,
- better performance,
- numerical experiments...

Updation of Correction terms

After updating Coriolis terms, $\hat{u}^{n+1}, \hat{v}^{n+1}$ and $\hat{T}^{n+1}$ are all be updated. Next step is to determine correction terms, Δu^{n+1} and Δv^{n+1}.
Suppose that $p^{n+1} = \hat{p}^{n+1} + \Delta p^{n+1}$. Substitute $u^{n+1} = \hat{u}^{n+1} + \Delta u^{n+1}$, and $v^{n+1} = \hat{v}^{n+1} + \Delta v^{n+1}$ into equations (19) and (19) respectively. Then using (23), (24), (**??**), and (**??**), we can get:

$$\Delta u^{n+1} = -2\Delta t \times \frac{\partial \Delta p^{n+1}}{\partial x}, \tag{36}$$

$$\Delta v^{n+1} = -2\Delta t \times \frac{\partial \Delta p^{n+1}}{\partial y}. \tag{37}$$

Updation of Correction terms

Consider equation (1) again, and use relationship between velocity correctios and pressure correction, we have:

$$\begin{aligned}\int_{-h}^{0}\left(\frac{\partial \hat{u}^{n+1}}{\partial x}+\frac{\partial \hat{v}^{n+1}}{\partial y}\right)dz &= -\int_{-h}^{0}\left(\frac{\partial \Delta u^{n+1}}{\partial x}+\frac{\partial \Delta v^{n+1}}{\partial y}\right)dz \\ &= 2\Delta t\times\int_{-h}^{0}\left(\frac{\partial^2 \Delta p^{n+1}}{\partial x^2}+\frac{\partial^2 \Delta p^{n+1}}{\partial y^2}\right)dz. \quad (38)\end{aligned}$$

Solving Δp^{n+1} is equivalent to solving a Poisson equation, which we apply EVP solver on it. After getting Δp^{n+1}, we can correct it back to pressure field and horizontal velocity field. Use (**??**) with any simple finite difference scheme, we can compute vertical velocity w^{n+1} from u^{n+1} and v^{n+1}. This velocity field therefore holds divergence-free property.

Time Filter

Time filter is used for controlling stability when we solve a time-dependent problem. Since we are using Leap-frog scheme as fundamental numerical time advancement scheme, filter scheme are only variation of the original scheme. Suppose now we already have previous time step varirables, say $n-1$, and we are in the n step. Our goal is to obtain variables which are at time step n after filtering.
Classical Leap-frog scheme is:

$$u^{n+1} - u^{n-1} = 2\Delta t \times F(u^n, t_n), \tag{39}$$

where F is source term.

Time Filter

Filtered scheme,

$$u_*^n = OFLTW \times u^n + FLTW \times (u^{n-1} + u^{n+1}), \tag{40}$$

where $*$ means variables that have been filtered at time step n, and $OFLTW = 0.9$, $FLTW = 0.05$ are set in the model. In DieCast ocean model, we accomplish this by assigning variables $U1$ and ULF cyclingly, where $U1$, ULF, $U2$ denote old, filtered (in the central time step), and new variables respectively.

DieCast

DieCast model is based on free surface boundary condition, rigid-lid general circulation model....... Here is the introduction of DieCast programming process. Structure of DieCast is:

1. Initialization
   - Determined derived scalars
   - Read data
     OPEN(14,file='TR') → run history data
     OPEN(19,file='SV') → restart data
   - CALL INITFS
2. Time Integration Loop **100**
   - Time step controled by ITF → DAYS
   - CALL FS → main computation SUBROUNTINE
3. Save Data
   - CALL XYPLOT
   - CALL XZPLOT

DieCast

There are six steps of numerical approach, which is a pressure-correction type numerical method for NSE. I describe them separately along with codes in DieCast.

In **SUBROUNTINE FS** and most SUBROUNTINE, COMMON command are used for distinquishing variables.

There are sveral velocity arrays use in DieCast, I clearify them here in order not to confused in latter computation algorithm.

- U is the velocity in the face,
- U2 is the velocity in the center of CV,
- ULF is the previous time's U2 value,
-

DieCast

There are six steps of numerical approach, which is a pressure-correction type numerical method for NSE. I describe them separately along with codes in DieCast.

In **SUBROUNTINE FS** and most SUBROUNTINE, COMMON command are used for distinquishing variables.

There are sveral velocity arrays use in DieCast, I clearify them here in order not to confused in latter computation algorithm.

- U is the velocity in the face,
- U2 is the velocity in the center of CV,
- ULF is the previous time's U2 value,
-

DieCast - Variables

Some important variables are described here, descriptions of all variables used can be seen in the *manual*.

- DT: time step size,
- ODT: time step,
- ODX,ODY,ODV,ODXV: inverse horizontal grid increments,
- ODZ: inverse layer thickness array,
- IN: mask array for scala quantities,
- IU,IV,IW: 3-d mask array for staggered ith-component advection velocity,
- U1,U2,ULF: old, filtered(central), leapfrog x-velocity field,
- U: staggered 'C' grid x-component non-divergent advection velocity.

First Step - Pressure

Guess the trial surface pressure $\bar{p}_s^n$ from the previous time step, that is $\bar{p}_s^n = p_s^{n-1}$ and integrate the hydrostatic equation (5) in order to get the intermediate pressure field $\bar{p}^n$ over the whole domain.
In **SUBROUNTINE FS**,

- RHO(I,J,K)=.0002*(20.-T2(I,J,K)),
- WFACE(I,J,1)=P0(I+1,J+1),
- REDG=G/(1.+100.*EXP(-0.5*DAYS)),
- TMP=G/ODZ(K)
- WFACE(I,J,L)= WFACE(I,J,K)+ TMP*RHO(I+1,J+1,K)

- from state equation, $\rho = \rho(S,T)$,
- '1' denotes surface layer,
- reduced gravity does not utlize here,
- set TMP variable, $g \times \Delta z$
- discrete hydrostatic equation, $p_f(i,j,k+1) = p_f(i,j,k) + g\rho(i+1,j+1,k)\Delta z$.

First Step - Pressure

Guess the trial surface pressure $\bar{p}_s^n$ from the previous time step, that is $\bar{p}_s^n = p_s^{n-1}$ and integrate the hydrostatic equation (5) in order to get the intermediate pressure field $\bar{p}^n$ over the whole domain.
In **SUBROUNTINE FS**,

- RHO(I,J,K)=.0002*(20.-T2(I,J,K)),
- WFACE(I,J,1)=P0(I+1,J+1),
- REDG=G/(1.+100.*EXP(-0.5*DAYS)),
- TMP=G/ODZ(K)
- WFACE(I,J,L)= WFACE(I,J,K)+ TMP*RHO(I+1,J+1,K)

- from state equation, $\rho = \rho(S,T)$,
- '1' denotes surface layer,
- reduced gravity does not utlize here,
- set TMP variable, $g \times \Delta z$
- discrete hydrostatic equation, $p_f(i,j,k+1) = p_f(i,j,k) + g\rho(i+1,j+1,k)\Delta z$.

First Step - Pressure

Guess the trial surface pressure $\bar{p}_s^n$ from the previous time step, that is $\bar{p}_s^n = p_s^{n-1}$ and integrate the hydrostatic equation (5) in order to get the intermediate pressure field $\bar{p}^n$ over the whole domain.
In **SUBROUNTINE FS**,

- RHO(I,J,K)=.0002*(20.-T2(I,J,K)),
- WFACE(I,J,1)=P0(I+1,J+1),
- REDG=G/(1.+100.*EXP(-0.5*DAYS)),
- TMP=G/ODZ(K)
- WFACE(I,J,L)= WFACE(I,J,K)+TMP*RHO(I+1,J+1,K)

- from state equation, $\rho = \rho(S, T)$,
- '1' denotes surface layer,
- reduced gravity does not utlize here,
- set TMP variable, $g \times \Delta z$
- discrete hydrostatic equation, $p_f(i,j,k+1) = p_f(i,j,k) + g\rho(i+1,j+1,k)\Delta z$.

First Step - Pressure

Guess the trial surface pressure $\bar{p}_s^n$ from the previous time step, that is $\bar{p}_s^n = p_s^{n-1}$ and integrate the hydrostatic equation (5) in order to get the intermediate pressure field $\bar{p}^n$ over the whole domain.
In **SUBROUNTINE FS**,

- RHO(I,J,K)=.0002*(20.-T2(I,J,K)),
- WFACE(I,J,1)=P0(I+1,J+1),
- REDG=G/(1.+100.*EXP(-0.5*DAYS)),
- TMP=G/ODZ(K)
- WFACE(I,J,L)= WFACE(I,J,K)+ TMP*RHO(I+1,J+1,K)

- from state equation, $\rho = \rho(S, T)$,
- '1' denotes surface layer,
- reduced gravity does not utlize here,
- set TMP variable, $g \times \Delta z$
- discrete hydrostatic equation, $p_f(i,j,k+1) = p_f(i,j,k) + g\rho(i+1,j+1,k)\Delta z$.

First Step - Pressure

Guess the trial surface pressure $\bar{p}_s^n$ from the previous time step, that is $\bar{p}_s^n = p_s^{n-1}$ and integrate the hydrostatic equation (5) in order to get the intermediate pressure field $\bar{p}^n$ over the whole domain.
In **SUBROUNTINE FS**,

- RHO(I,J,K)=.0002*(20.-T2(I,J,K)),
- WFACE(I,J,1)=P0(I+1,J+1),
- REDG=G/(1.+100.*EXP(-0.5*DAYS)),
- TMP=G/ODZ(K)
- WFACE(I,J,L)= WFACE(I,J,K)+ TMP*RHO(I+1,J+1,K)

- from state equation, $\rho = \rho(S, T)$,
- '1' denotes surface layer,
- reduced gravity does not utlize here,
- set TMP variable, $g \times \Delta z$
- discrete hydrostatic equation, $p_f(i,j,k+1) = p_f(i,j,k) + g\rho(i+1,j+1,k)\Delta z$.

First Step - Pressure

Guess the trial surface pressure $\bar{p}_s^n$ from the previous time step, that is $\bar{p}_s^n = p_s^{n-1}$ and integrate the hydrostatic equation (5) in order to get the intermediate pressure field $\bar{p}^n$ over the whole domain.
In **SUBROUNTINE FS**,

- RHO(I,J,K)=.0002*(20.-T2(I,J,K)),
- WFACE(I,J,1)=P0(I+1,J+1),
- REDG=G/(1.+100.*EXP(-0.5*DAYS)),
- TMP=G/ODZ(K)
- WFACE(I,J,L)= WFACE(I,J,K)+ TMP*RHO(I+1,J+1,K)

- from state equation, $\rho = \rho(S, T)$,
- '1' denotes surface layer,
- reduced gravity does not utlize here,
- set TMP variable, $g \times \Delta z$
- discrete hydrostatic equation, $p_f(i, j, k+1) = p_f(i, j, k) + g\rho(i+1, j+1, k)\Delta z$.

First Step - Pressure

Note that WFACE variable is the pressure on the control volume face, now we interpolate it into control volume averaged pressure.

- P(i+1,j+1,1)= .5*(WFACE(i,j,1)+WFACE(i,j,2)),
- P(i+1,j+1,K1)= .5*(WFACE(i,j,K1)+WFACE(i,j,K0)),
- P(i+1,j+1,k)= 12.*(WFACE(i,j,k)+WFACE(i,j,k+1)) +(WFACE(i,j,k)+WFACE(i,j,k+1) -WFACE(i,j,k-1)- WFACE(i,j,k+2)),
- P(i,j,k)=O24*P(i,j,k)

- 2nd-order accurate in top and bottom layler,
- 4th-order accurate in interior domain,
- $P(i+1,j+1,k) = 12*(p_f(i,j,k)+p_f(i,j,k+1))$ $p_f(i,j,k)+p_f(i,j,k+1)$ $-p_f(i,j,k-1)-p_f(i,j,k-2),$
- O24=$\frac{1.}{24.}$

First Step - Pressure

Note that WFACE variable is the pressure on the control volume face, now we interpolate it into control volume averaged pressure.

- P(i+1,j+1,1)= .5*(WFACE(i,j,1)+WFACE(i,j,2)),
- P(i+1,j+1,K1)= .5*(WFACE(i,j,K1)+WFACE(i,j,K0)),
- P(i+1,j+1,k)= 12.*(WFACE(i,j,k)+WFACE(i,j,k+1)) +(WFACE(i,j,k)+WFACE(i,j,k+1) -WFACE(i,j,k-1)- WFACE(i,j,k+2)),
- P(i,j,k)=O24*P(i,j,k)

- 2nd-order accurate in top and bottom layler,
- 4th-order accurate in interior domain,
- $P(i+1,j+1,k) = 12*(p_f(i,j,k)+p_f(i,j,k+1))$ $p_f(i,j,k)+p_f(i,j,k+1)$ $-p_f(i,j,k-1)-p_f(i,j,k-2)$,
- O24=$\frac{1.}{24.}$

First Step - Pressure

Note that WFACE variable is the pressure on the control volume face, now we interpolate it into control volume averaged pressure.

- P(i+1,j+1,1)= .5*(WFACE(i,j,1)+WFACE(i,j,2)),
- P(i+1,j+1,K1)= .5*(WFACE(i,j,K1)+WFACE(i,j,K0)),
- P(i+1,j+1,k)= 12.*(WFACE(i,j,k)+WFACE(i,j,k+1)) +(WFACE(i,j,k)+WFACE(i,j,k+1) -WFACE(i,j,k-1)- WFACE(i,j,k+2)),
- P(i,j,k)=O24*P(i,j,k)

- 2nd-order accurate in top and bottom layler,
- 4th-order accurate in interior domain,
- $P(i+1,j+1,k) = 12 * (p_f(i,j,k) + p_f(i,j,k+1))$ $p_f(i,j,k) + p_f(i,j,k+1)$ $-p_f(i,j,k-1) - p_f(i,j,k-2),$
- O24=$\frac{1.}{24.}$

First Step - Pressure

Note that WFACE variable is the pressure on the control volume face, now we interpolate it into control volume averaged pressure.

- P(i+1,j+1,1)= .5*(WFACE(i,j,1)+WFACE(i,j,2)),
- P(i+1,j+1,K1)= .5*(WFACE(i,j,K1)+WFACE(i,j,K0)),
- P(i+1,j+1,k)= 12.*(WFACE(i,j,k)+WFACE(i,j,k+1)) +(WFACE(i,j,k)+WFACE(i,j,k+1) -WFACE(i,j,k-1)- WFACE(i,j,k+2)),
- P(i,j,k)=O24*P(i,j,k)

- 2nd-order accurate in top and bottom layler,
- 4th-order accurate in interior domain,
- $P(i+1, j+1, k) = 12 * (p_f(i,j,k) + p_f(i,j,k+1))$ $p_f(i,j,k) + p_f(i,j,k+1)$ $-p_f(i,j,k-1) - p_f(i,j,k-2)$,
- O24$=\frac{1.}{24.}$

First Step - Pressure

Note that WFACE variable is the pressure on the control volume face, now we interpolate it into control volume averaged pressure.

- P(i+1,j+1,1)= .5*(WFACE(i,j,1)+WFACE(i,j,2)),
- P(i+1,j+1,K1)= .5*(WFACE(i,j,K1)+WFACE(i,j,K0)),
- P(i+1,j+1,k)= 12.*(WFACE(i,j,k)+WFACE(i,j,k+1)) +(WFACE(i,j,k)+WFACE(i,j,k+1) -WFACE(i,j,k-1)- WFACE(i,j,k+2)),
- P(i,j,k)=O24*P(i,j,k)

- 2nd-order accurate in top and bottom layler,
- 4th-order accurate in interior domain,
- $P(i+1, j+1, k) = 12 * (p_f(i,j,k) + p_f(i,j,k+1)) \; p_f(i,j,k) + p_f(i,j,k+1) \; - p_f(i,j,k-1) - p_f(i,j,k-2)$,
- O24=$\frac{1.}{24.}$

Second Step - Update Velocities

Update the trial integral average $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume using discretized momentum equations, e.g. u-component discretized equation:

$$\begin{aligned}
\frac{\bar{u}^{n+1}_{i,j,k} + u^{n-1}_{i,j,k}}{\Delta t} &= -(\left(\frac{\partial U u}{\partial x}\right)^n_{i,j,k} + \left(\frac{\partial V u}{\partial y}\right)^n_{i,j,k} \\
&+ \left(\frac{\partial W u}{\partial z}\right)^n_{i,j,k}) \\
&+ [2\Omega_e \sin(\phi_j) + u^n_{i,j,k} \tan(\frac{\phi_j}{r_e})] v^n_{i,j,k} \\
&- \left(\frac{\partial \bar{p}}{\partial x}\right)^n_{i,j,k} + dissipation, \qquad (41)
\end{aligned}$$

In **SUBROUNTIN FS, Loop** 500 is the main computation loop. In this loop it calculate horizontal velocity components on the control volume face.

Second Step - Update Velocities

Update the trial integral average $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume using discretized momentum equations, e.g. u-component discretized equation:

$$
\begin{aligned}
\frac{\bar{u}^{n+1}_{i,j,k} + u^{n-1}_{i,j,k}}{\Delta t} &= -(\left(\frac{\partial U u}{\partial x}\right)^n_{i,j,k} + \left(\frac{\partial V u}{\partial y}\right)^n_{i,j,k} \\
&+ \left(\frac{\partial W u}{\partial z}\right)^n_{i,j,k}) \\
&+ [2\Omega_e \sin(\phi_j) + u^n_{i,j,k} \tan(\frac{\phi_j}{r_e})] v^n_{i,j,k} \\
&- \left(\frac{\partial \bar{p}}{\partial x}\right)^n_{i,j,k} + dissipation,
\end{aligned} \tag{41}
$$

In **SUBROUNTIN FS, Loop** 500 is the main computation loop. In this loop it calculate horizontal velocity components on the control volume face.

Second Step - Pressure Gradient

In **Loop 500**, first it calculate pressure gradient, which is a forth-order accurate approximation.
Take $\frac{\partial p}{\partial x}$ term as example ($\frac{\partial p}{\partial y}$ term has similar algorithm).

- UFACE(i,j)= 6.*(P(i,j+1,k)+P(i+1,j+1,k)) +IU(i-1,j+1,k)*IU(i+1,j+1,k) *(P(i,j+1,k)+P(i+1,j+1,k) -P(i-1,j+1,k)-P(i+2,j+1,k)),
- UFACE(i,j)= IU(i,j+1,k)*UFACE(i,j) +(1.-IU(i,j+1,k)) *12.*(IN(i,j+1,k)*P(i,j+1,k) +IN(i+1,j+1,k)*P(i+1,j+1,k))
- PX(i,j)=IN(i,j,k)*O12 *(UFACE(i,j-1)-UFACE(i-1,j-1)).

- 4nd-order accurate interpolation to get pressure on the CV face,
- use nearest neighbor when data is not available. It may be improved at duo grid boundary by using coupled grid values,
- $O12=\frac{1.}{12.}$
- IN array is related to bathymetry, so do IU, IV, IW.

Second Step - Pressure Gradient

In **Loop 500**, first it calculate pressure gradient, which is a forth-order accurate approximation.
Take $\frac{\partial p}{\partial x}$ term as example ($\frac{\partial p}{\partial y}$ term has similar algorithm).

- UFACE(i,j)=
6.*(P(i,j+1,k)+P(i+1,j+1,k))
+IU(i-1,j+1,k)*IU(i+1,j+1,k)
*(P(i,j+1,k)+P(i+1,j+1,k)
-P(i-1,j+1,k)-P(i+2,j+1,k)),
- UFACE(i,j)=
IU(i,j+1,k)*UFACE(i,j)
+(1.-IU(i,j+1,k))
12.(IN(i,j+1,k)*P(i,j+1,k)
+IN(i+1,j+1,k)*P(i+1,j+1,k))
- PX(i,j)=IN(i,j,k)*O12
*(UFACE(i,j-1)-UFACE(i-1,j-1)).

- 4nd-order accurate interpolation to get pressure on the CV face,
- use nearest neighbor when data is not available. It may be improved at duo grid boundary by using coupled grid values,
- O12=$\frac{1.}{12.}$
- IN array is related to bathymetry, so do IU, IV, IW.

Second Step - Pressure Gradient

In **Loop** 500, first it calculate pressure gradient, which is a forth-order accurate approximation.
Take $\frac{\partial p}{\partial x}$ term as example ($\frac{\partial p}{\partial y}$ term has similar algorithm).

- UFACE(i,j)= 6.*(P(i,j+1,k)+P(i+1,j+1,k)) +IU(i-1,j+1,k)*IU(i+1,j+1,k) *(P(i,j+1,k)+P(i+1,j+1,k) -P(i-1,j+1,k)-P(i+2,j+1,k)),
- UFACE(i,j)= IU(i,j+1,k)*UFACE(i,j) +(1.-IU(i,j+1,k)) *12.*(IN(i,j+1,k)*P(i,j+1,k) +IN(i+1,j+1,k)*P(i+1,j+1,k))
- PX(i,j)=IN(i,j,k)*O12 *(UFACE(i,j-1)-UFACE(i-1,j-1)).

- 4nd-order accurate interpolation to get pressure on the CV face,
- use nearest neighbor when data is not available. It may be improved at duo grid boundary by using coupled grid values,
- O12=$\frac{1.}{12.}$
- IN array is related to bathymetry, so do IU, IV, IW.

Second Step - Pressure Gradient

In **Loop 500**, first it calculate pressure gradient, which is a forth-order accurate approximation.
Take $\frac{\partial p}{\partial x}$ term as example ($\frac{\partial p}{\partial y}$ term has similar algorithm).

- UFACE(i,j)=
6.*(P(i,j+1,k)+P(i+1,j+1,k))
+IU(i-1,j+1,k)*IU(i+1,j+1,k)
*(P(i,j+1,k)+P(i+1,j+1,k)
-P(i-1,j+1,k)-P(i+2,j+1,k)),
- UFACE(i,j)=
IU(i,j+1,k)*UFACE(i,j)
+(1.-IU(i,j+1,k))
12.(IN(i,j+1,k)*P(i,j+1,k)
+IN(i+1,j+1,k)*P(i+1,j+1,k))
- PX(i,j)=IN(i,j,k)*O12
*(UFACE(i,j-1)-UFACE(i-1,j-1)).

- 4nd-order accurate interpolation to get pressure on the CV face,
- use nearest neighbor when data is not available. It may be improved at duo grid boundary by using coupled grid values,
- O12=$\frac{1.}{12.}$
- IN array is related to bathymetry, so do IU, IV, IW.

Second Step - Pressure Gradient

In **Loop 500**, first it calculate pressure gradient, which is a forth-order accurate approximation.
Take $\frac{\partial p}{\partial x}$ term as example ($\frac{\partial p}{\partial y}$ term has similar algorithm).

- UFACE(i,j)=
  6.*(P(i,j+1,k)+P(i+1,j+1,k))
  +IU(i-1,j+1,k)*IU(i+1,j+1,k)
  *(P(i,j+1,k)+P(i+1,j+1,k)
  -P(i-1,j+1,k)-P(i+2,j+1,k)),
- UFACE(i,j)=
  IU(i,j+1,k)*UFACE(i,j)
  +(1.-IU(i,j+1,k))
  12.(IN(i,j+1,k)*P(i,j+1,k)
  +IN(i+1,j+1,k)*P(i+1,j+1,k))
- PX(i,j)=IN(i,j,k)*O12
  *(UFACE(i,j-1)-UFACE(i-1,j-1)).

- 4nd-order accurate interpolation to get pressure on the CV face,
- use nearest neighbor when data is not available. It may be improved at duo grid boundary by using coupled grid values,
- O12=$\frac{1.}{12.}$
- IN array is related to bathymetry, so do IU, IV, IW.

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-T1(I,J,K))*IW(I,J,L)

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

```
DO 350 J=2,J1
DO 350 I=2,I1
TMP=W(I,J,L)
UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
1 -EV(I-1,J-1,K)*(U1(I,J,L)-
U1(I,J,K))*IW(I,J,L)
VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
1 -EV(I-1,J-1,K)*(V1(I,J,L)-
V1(I,J,K))*IW(I,J,L)
350
TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
1 -HV(I-1,J-1,K)*(T1(I,J,L)-
T1(I,J,K))*IW(I,J,L)
```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Fluxes

Vertical, longitudinal, and latitudinal fluxes are calculated after pressure gradient is calculated. Fluxes are used to calculated $\bar{u}^{n+1}, \bar{v}^{n+1}$ in the control volume. Take Vertical fluxes for example (interpolation is done before calculating).

- ```
  DO 350 J=2,J1
  DO 350 I=2,I1
  TMP=W(I,J,L)
  UZ(I-1,J-1,LT)=TMP*SCR(I,J,1)
  1 -EV(I-1,J-1,K)*(U1(I,J,L)-
  U1(I,J,K))*IW(I,J,L)
  VZ(I-1,J-1,LT)=TMP*SCR(I,J,2)
  1 -EV(I-1,J-1,K)*(V1(I,J,L)-
  V1(I,J,K))*IW(I,J,L)
  350
  TZ(I-1,J-1,LT)=TMP*SCR(I,J,3)
  1 -HV(I-1,J-1,K)*(T1(I,J,L)-
  T1(I,J,K))*IW(I,J,L)
  ```

- SCR(i,j,1): U2 velocity (on the face),
- UZ=$W \times U$+turbulence terms,
- VZ=$W \times V$+turbulence terms,
- TZ=$W \times T$+turbulence terms,
- EV: vertical turbulent viscosity array,
- HV: vertical turbulent diffusivity array.
  ```

Second Step - Conservation Equations

After calculating fluxes and pressure gradient, we can use (41) to get horizontal momentum (and temperature) at the next time step on the center of the CV. Take Longitudinal momentum for example.

- U2(i,j,k)=U1(i,j,k)-DTIN
  *((UX(i,j-1)-UX(i-1,j-1)
  +PX(i,j))*ODX(j)
  +(UY(i-1,j)-UY(i-1,j-1)*ODYJ
  +(UZ(i-1,j-1,LT)=UZ(i-1,j-1,LB))
  *ODZ(k))

- DTIN=DT*IN(I,J,K)
- LT=2 ??
- LB=1 ??

Loop 500 is finished.
After calculating these variables, 'Open boundary conditions' are used. These are all determined by 'known' normal boundary velocity (NBV) i.e. boundary normal flux is UPWINDED for both inflow and outflow.
Loop 506 for SCALAR fluxes on boundaries, Loop 632 for MOMENTUM fluxes, Loop 644 for determining NBV.

Second Step - Conservation Equations

After calculating fluxes and pressure gradient, we can use (41) to get horizontal momentum (and temperature) at the next time step on the center of the CV. Take Longitudinal momentum for example.

- U2(i,j,k)=U1(i,j,k)-DTIN *((UX(i,j-1)-UX(i-1,j-1) +PX(i,j))*ODX(j) +(UY(i-1,j)-UY(i-1,j-1)*ODYJ +(UZ(i-1,j-1,LT)=UZ(i-1,j-1,LB)) *ODZ(k))

- DTIN=DT*IN(I,J,K)
- LT=2 ??
- LB=1 ??

Loop 500 is finished.
After calculating these variables, 'Open boundary conditions' are used. These are all determined by 'known' normal boundary velocity (NBV) i.e. boundary normal flux is UPWINDED for both inflow and outflow.
Loop 506 for SCALAR fluxes on boundaries, Loop 632 for MOMENTUM fluxes, Loop 644 for determining NBV.

Second Step - Conservation Equations

After calculating fluxes and pressure gradient, we can use (41) to get horizontal momentum (and temperature) at the next time step on the center of the CV. Take Longitudinal momentum for example.

- U2(i,j,k)=U1(i,j,k)-DTIN
  *((UX(i,j-1)-UX(i-1,j-1)
  +PX(i,j))*ODX(j)
  +(UY(i-1,j)-UY(i-1,j-1)*ODYJ
  +(UZ(i-1,j-1,LT)=UZ(i-1,j-1,LB))
  *ODZ(k))

- DTIN=DT*IN(I,J,K)
- LT=2 ??
- LB=1 ??

Loop 500 is finished.
After calculating these variables, 'Open boundary conditions' are used. These are all determined by 'known' normal boundary velocity (NBV) i.e. boundary normal flux is UPWINDED for both inflow and outflow.
Loop 506 for SCALAR fluxes on boundaries, Loop 632 for MOMENTUM fluxes, Loop 644 for determining NBV.

Second Step - Conservation Equations

After calculating fluxes and pressure gradient, we can use (41) to get horizontal momentum (and temperature) at the next time step on the center of the CV. Take Longitudinal momentum for example.

- U2(i,j,k)=U1(i,j,k)-DTIN *((UX(i,j-1)-UX(i-1,j-1) +PX(i,j))*ODX(j) +(UY(i-1,j)-UY(i-1,j-1)*ODYJ +(UZ(i-1,j-1,LT)=UZ(i-1,j-1,LB)) *ODZ(k))

- DTIN=DT*IN(I,J,K)
- LT=2 ??
- LB=1 ??

Loop 500 is finished.
After calculating these variables, 'Open boundary conditions' are used. These are all determined by 'known' normal boundary velocity (NBV) i.e. boundary normal flux is UPWINDED for both inflow and outflow.
Loop 506 for SCALAR fluxes on boundaries, Loop 632 for MOMENTUM fluxes, Loop 644 for determining NBV.

Second Step - Conservation Equations

After calculating fluxes and pressure gradient, we can use (41) to get horizontal momentum (and temperature) at the next time step on the center of the CV. Take Longitudinal momentum for example.

- U2(i,j,k)=U1(i,j,k)-DTIN *((UX(i,j-1)-UX(i-1,j-1) +PX(i,j))*ODX(j) +(UY(i-1,j)-UY(i-1,j-1)*ODYJ +(UZ(i-1,j-1,LT)=UZ(i-1,j-1,LB)) *ODZ(k))

- DTIN=DT*IN(I,J,K)
- LT=2 ??
- LB=1 ??

Loop 500 is finished.
After calculating these variables, ‘Open boundary conditions’ are used. These are all determined by ‘known’ normal boundary velocity (NBV) i.e. boundary normal flux is UPWINDED for both inflow and outflow.
Loop 506 for SCALAR fluxes on boundaries, Loop 632 for MOMENTUM fluxes, Loop 644 for determining NBV.

Third Step - Center to Face Transformation

In this step, interpolation of $\bar{u}^{n+1}, \bar{v}^{n+1}$ to $\bar{U}^{n+1}_{i,j,k}, \bar{V}^{n+1}_{i,j,k}$ at the cell face using 4th-order formula is operatied. Take u-momentum for example.

- SCR(i,j,1)= 6.*(U2(i,j,k)+U2(i+1,j,k))
- TMP=IN(i-1,j,k)*IN(i,j,k) *IN(i+1,j,k)*IN(i+2,j,k)
- SCR(I,J,1)=SCR(I,J,1)+TMP *(-U2(I-1,J,K)+U2(I,J,K) +U2(I+1,J,K)-U2(I+2,J,K))
- U(i,j,k)=O12*SCR(i,j,1)*IU(i,j,k)

- forth-order interpolation
-

$$\bar{q}_{i+1/2,j,k} = \frac{7}{12}(Q_{i,j,k} + Q_{i+1,j,k}) - \frac{1}{12}(Q_{i-1,j,k} + Q_{i+2,j,k})$$

Third Step - Center to Face Transformation

In this step, interpolation of $\bar{u}^{n+1}, \bar{v}^{n+1}$ to $\bar{U}^{n+1}_{i,j,k}, \bar{V}^{n+1}_{i,j,k}$ at the cell face using 4th-order formula is operatied. Take u-momentum for example.

- SCR(i,j,1)= 6.*(U2(i,j,k)+U2(i+1,j,k))
- TMP=IN(i-1,j,k)*IN(i,j,k) *IN(i+1,j,k)*IN(i+2,j,k)
- SCR(I,J,1)=SCR(I,J,1)+TMP *(-U2(I-1,J,K)+U2(I,J,K) +U2(I+1,J,K)-U2(I+2,J,K))
- U(i,j,k)=O12*SCR(i,j,1)*IU(i,j,k)

- forth-order interpolation
- $$\bar{q}_{i+1/2,j,k} = \frac{7}{12}(Q_{i,j,k} + Q_{i+1,j,k}) - \frac{1}{12}(Q_{i-1,j,k} + Q_{i+2,j,k})$$

Third Step - Center to Face Transformation

In this step, interpolation of $\bar{u}^{n+1}, \bar{v}^{n+1}$ to $\bar{U}^{n+1}_{i,j,k}, \bar{V}^{n+1}_{i,j,k}$ at the cell face using 4th-order formula is operatied. Take u-momentum for example.

- SCR(i,j,1)= 6.*(U2(i,j,k)+U2(i+1,j,k))
- TMP=IN(i-1,j,k)*IN(i,j,k) *IN(i+1,j,k)*IN(i+2,j,k)
- SCR(I,J,1)=SCR(I,J,1)+TMP *(-U2(I-1,J,K)+U2(I,J,K) +U2(I+1,J,K)-U2(I+2,J,K))
- U(i,j,k)=O12*SCR(i,j,1)*IU(i,j,k)

- forth-order interpolation
- $$\bar{q}_{i+1/2,j,k} = \frac{7}{12}(Q_{i,j,k} + Q_{i+1,j,k}) - \frac{1}{12}(Q_{i-1,j,k} + Q_{i+2,j,k})$$

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla\vec{v} = 0$ to get vertical velocity W, code is below:

```
W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)
```

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla \vec{v} = 0$ to get vertical velocity W, code is below:

W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla \vec{v} = 0$ to get vertical velocity W, code is below:

```
W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)
```

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla \vec{v} = 0$ to get vertical velocity W, code is below:

```
W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)
```

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla\vec{v} = 0$ to get vertical velocity W, code is below:

```
W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)
```

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla \vec{v} = 0$ to get vertical velocity W, code is below:

```
W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)
```

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Before we calculate correction for pressure term, DieCast model do outflow check.

- SUM=SUM+V(I,J1,K)*IN(I,J1,K) /(ODYV(J1)*ODZ(K))
- SUM=SUM/AROUT
- V(I,J1,K)=V(I,J1,K) +SUM*IN(I,J1,K)
- FORMAT('outflow vel correction = ',1PE9.2,' cm/sec')

- sum over CV in South and North boundary plus West and East,
- AROUT is the area of bpundary region to be adjusted to get zero net inflow,
- AROUT = 4.000607E13

After correction, use divergence free property $\nabla \vec{v} = 0$ to get vertical velocity W, code is below:

```
W(I,J,K+1)=W(I,J,K)-((U(I,J,K)-U(I-1,J,K))*ODX(J)
+(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*TEMP)*TEMP1
S(I-1,J-1)=-W(I,J,KB(I,J)+1)
```

Note that variables: CSV(J), TEMP, TEMP1.

Fourth Step - Pressure Correction

Idea of Pressure Correction here is that if we set final pressure has the form, $p^n = \bar{p}^n + \Delta\bar{p}$, where Δp is due to the change of rigid-lid pressure and thus independent of depth. Since $\bar{p}^n$ is derived form first step, we need to get Δp. Then final velocity can be written as:

$$U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U} \tag{42}$$

$$V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}, \tag{43}$$

where $\Delta\bar{U} = -\Delta t\frac{\partial\Delta\bar{p}}{\partial x}$ and $\Delta\bar{V} = -\Delta t\frac{\partial\Delta\bar{p}}{\partial y}$.

Integrating (1) we can get,

$$\int_0^D \left(\frac{\partial U^{n+1}}{\partial x} + \frac{\partial V^{n+1}}{\partial y}\right)dz = W^{n+1}(0) - W^{n+1}(D) = 0, \tag{44}$$

Fourth Step - Pressure Correction

Substitude (42) and (43) into (44), we can get:

$$\int_0^D \left(\frac{\partial \bar{U}^{n+1}}{\partial x} + \frac{\partial \bar{V}^{n+1}}{\partial y}\right)dz = \int_0^D -\left(\frac{\partial \Delta \bar{U}}{\partial x} + \frac{\partial \Delta \bar{V}}{\partial y}\right)dz. \tag{45}$$

This is a Poisson equation, if we furthermore subsitude $\Delta \bar{U} = -\Delta t \frac{\partial \Delta \bar{p}}{\partial x}$ and $\Delta \bar{V} = -\Delta t \frac{\partial \Delta \bar{p}}{\partial y}$ into it, we get:

$$\int_0^D -\left(\frac{\partial^2 \Delta p}{\partial x^2} + \frac{\partial^2 \Delta p}{\partial y^2}\right)dz = f(W), \tag{46}$$

where $f(W)$ can be viewed as the source term.

Fourth Step - Pressure Correction - EVP solver

Now, CALL REP SUBROUNTINE for EVP solver.
Input variable is S, output variable is X.
'S(I-1,J-1)=-W(I,J,KB(I,J)+1)'.
Littleeasy's work is mainly focus on here. Parallel EVE solver is promising due to its highly sufficient for Poisson equaiton.

Fourth Step - Pressure Correction - EVP solver

Now, CALL REP SUBROUNTINE for EVP solver.
Input variable is S, output variable is X.
'S(I-1,J-1)=-W(I,J,KB(I,J)+1)'.
Littleeasy's work is mainly focus on here. Parallel EVE solver is promising due to its highly sufficient for Poisson equaiton.

Fifth Step - Correction

Since we have variable X for pressure correction, we can therefor correct U^{n+1}, V^{n+1}.

- P0(I,J)=P0(I,J)+ODT*X(I,J)
- SCR(I,J,1)=(X(I+1,J)-X(I,J))*ODX(J)
- SCR(I,J,2)=(X(I,J+1)-X(I,J))*ODYV(J)
- U(I,J,K)=U(I,J,K)-SCR(I,J,1)*IU(I,J,K)
- V(I,J,K)=V(I,J,K)-SCR(I,J,2)*IV(I,J,K)

- $p^n = \bar{p}^n + \Delta\bar{p}$
- $\frac{\partial p}{\partial x}$
- $\frac{\partial p}{\partial y}$
- $U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U}$
- $V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}$

Fifth Step - Correction

Since we have variable X for pressure correction, we can therefor correct U^{n+1}, V^{n+1}.

- P0(I,J)=P0(I,J)+ODT*X(I,J)
- SCR(I,J,1)=(X(I+1,J)-X(I,J))*ODX(J)
- SCR(I,J,2)=(X(I,J+1)-X(I,J))*ODYV(J)
- U(I,J,K)=U(I,J,K)-SCR(I,J,1)*IU(I,J,K)
- V(I,J,K)=V(I,J,K)-SCR(I,J,2)*IV(I,J,K)

- $p^n = \bar{p}^n + \Delta\bar{p}$
- $\frac{\partial p}{\partial x}$
- $\frac{\partial p}{\partial y}$
- $U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U}$
- $V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}$

Fifth Step - Correction

Since we have variable X for pressure correction, we can therefor correct U^{n+1}, V^{n+1}.

- P0(I,J)=P0(I,J)+ODT*X(I,J)
- SCR(I,J,1)=(X(I+1,J)-X(I,J))*ODX(J)
- SCR(I,J,2)=(X(I,J+1)-X(I,J))*ODYV(J)
- U(I,J,K)=U(I,J,K)-SCR(I,J,1)*IU(I,J,K)
- V(I,J,K)=V(I,J,K)-SCR(I,J,2)*IV(I,J,K)

- $p^n = \bar{p}^n + \Delta\bar{p}$
- $\frac{\partial p}{\partial x}$
- $\frac{\partial p}{\partial y}$
- $U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U}$
- $V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}$

Fifth Step - Correction

Since we have variable X for pressure correction, we can therefor correct U^{n+1}, V^{n+1}.

- P0(I,J)=P0(I,J)+ODT*X(I,J)
- SCR(I,J,1)=(X(I+1,J)-X(I,J))*ODX(J)
- SCR(I,J,2)=(X(I,J+1)-X(I,J))*ODYV(J)
- U(I,J,K)=U(I,J,K)-SCR(I,J,1)*IU(I,J,K)
- V(I,J,K)=V(I,J,K)-SCR(I,J,2)*IV(I,J,K)

- $p^n = \bar{p}^n + \Delta\bar{p}$
- $\frac{\partial p}{\partial x}$
- $\frac{\partial p}{\partial y}$
- $U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U}$
- $V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}$

Fifth Step - Correction

Since we have variable X for pressure correction, we can therefor correct U^{n+1}, V^{n+1}.

- P0(I,J)=P0(I,J)+ODT*X(I,J)
- SCR(I,J,1)=(X(I+1,J)-X(I,J))*ODX(J)
- SCR(I,J,2)=(X(I,J+1)-X(I,J))*ODYV(J)
- U(I,J,K)=U(I,J,K)-SCR(I,J,1)*IU(I,J,K)
- V(I,J,K)=V(I,J,K)-SCR(I,J,2)*IV(I,J,K)

- $p^n = \bar{p}^n + \Delta\bar{p}$
- $\frac{\partial p}{\partial x}$
- $\frac{\partial p}{\partial y}$
- $U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U}$
- $V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}$

Fifth Step - Correction

Since we have variable X for pressure correction, we can therefor correct U^{n+1}, V^{n+1}.

- P0(I,J)=P0(I,J)+ODT*X(I,J)
- SCR(I,J,1)=(X(I+1,J)-X(I,J))*ODX(J)
- SCR(I,J,2)=(X(I,J+1)-X(I,J))*ODYV(J)
- U(I,J,K)=U(I,J,K)-SCR(I,J,1)*IU(I,J,K)
- V(I,J,K)=V(I,J,K)-SCR(I,J,2)*IV(I,J,K)

- $p^n = \bar{p}^n + \Delta\bar{p}$
- $\frac{\partial p}{\partial x}$
- $\frac{\partial p}{\partial y}$
- $U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U}$
- $V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}$

Sixth Step - Pressure Correction - EVP solver

It is the last step that interpolate cell average changes $\Delta\bar{u}$,$\Delta\bar{v}$ from $\Delta\bar{U}$, $\Delta\bar{V}$ by fourth-order method. Then get:

$$u^{n+1} = \bar{u}^{n+1} + \Delta\bar{u}, \tag{47}$$

$$v^{n+1} = \bar{v}^{n+1} + \Delta\bar{v}. \tag{48}$$

and then finalize computation of this time step.

```
DO 686 I=2,I1
686 SCR(I,J,1)=12.*(IU(I-1,J,K)*SCR(I-1,J,3)+IU(I,J,K)*SCR(I,J,3))
DO 687 I=3,I2
687 SCR(I,J,1)=SCR(I,J,1)
1 -IU(I-2,J,K)*SCR(I-2,J,3)+IU(I-1,J,K)*SCR(I-1,J,3)
2 +IU(I,J,K)*SCR(I,J,3)-IU(I+1,J,K)*SCR(I+1,J,3)
DO 688 I=2,I1
688 U2(I,J,K)=IN(I,J,K)*(U2(I,J,K)+O24*SCR(I,J,1))
689 CONTINUE
```

Sixth Step - Pressure Correction - EVP solver

It is the last step that interpolate cell average changes $\Delta\bar{u}$,$\Delta\bar{v}$ from $\Delta\bar{U}$, $\Delta\bar{V}$ by fourth-order method. Then get:

$$u^{n+1} = \bar{u}^{n+1} + \Delta\bar{u}, \tag{47}$$

$$v^{n+1} = \bar{v}^{n+1} + \Delta\bar{v}. \tag{48}$$

and then finalize computation of this time step.

```
DO 686 I=2,I1
686 SCR(I,J,1)=12.*(IU(I-1,J,K)*SCR(I-1,J,3)+IU(I,J,K)*SCR(I,J,3))
DO 687 I=3,I2
687 SCR(I,J,1)=SCR(I,J,1)
1 -IU(I-2,J,K)*SCR(I-2,J,3)+IU(I-1,J,K)*SCR(I-1,J,3)
2 +IU(I,J,K)*SCR(I,J,3)-IU(I+1,J,K)*SCR(I+1,J,3)
DO 688 I=2,I1
688 U2(I,J,K)=IN(I,J,K)*(U2(I,J,K)+O24*SCR(I,J,1))
689 CONTINUE
```

Sixth Step - Incompressible check

Now we check incompressible,

```
TMP=0.
ERR=0.
DO 710 K=1,K1
DO 710 J=2,J1
DO 710 I=2,I1
TMP1=(U(I,J,K)-U(I-1,J,K))*ODX(J)
TMP2=(CSV(J)*V(I,J,K)-CSV(J-1)*V(I,J-1,K))*OCS(J)*ODY(J)
TMP3=(W(I,J,K+1)-W(I,J,K))*ODZ(K)
TMP=TMP+MAX(ABS(TMP1),ABS(TMP2),ABS(TMP3))*IN(I,J,K)
710 ERR=ERR+ABS(TMP1+TMP2+TMP3)*IN(I,J,K)
ERR=ERR/TMP
WRITE(*,711) ERR
WRITE(14,711) ERR
711 FORMAT(' *** NORMALIZED mean incompressibility error = ',1PE9.2)
```

Sixth Step - FLTW method

Based on the paper *'Frequency Filter for Time Integrations'* by Richard Asselin, time filter is introduced into our DieCast model.
We use FLTW, 'Filtered Leap-frog-Trapezoidal Weighted' scheme, for time advance, which is a variety of FLT scheme. Basic time filter, for example,

$$\overline{F}(t) = F(t) + 0.5\nu[\overline{F}(t-1) - 2F(t) + F(t+1)]. \tag{49}$$

Well-known centered-filter is,

$$\overline{F}(t) = F(t) + 0.5\nu[F(t-1) - 2F(t) + F(t+1)]. \tag{50}$$

Sixth Step - FLTW method

Following Kurihara(1965), consider the differentail model,

$$\frac{\partial F}{\partial t} = i\omega F. \tag{51}$$

Write if as difference form with filter, we can get,

$$\frac{F(t+1) - F(t\bar{-}1)}{2\Delta t} = i\omega_A F(t) + i(\omega - \omega_A)\frac{F(t+1) + F(t\bar{-}1)}{2\Delta t}. \tag{52}$$

where ω and ω_A are two parameters.
Filter is used for two purposes:

1. Reducing damping
2. Stability

Sixth Step - FLTW method

Following Kurihara(1965), consider the differentail model,

$$\frac{\partial F}{\partial t} = i\omega F. \tag{51}$$

Write if as difference form with filter, we can get,

$$\frac{F(t+1) - F(t\bar{-}1)}{2\Delta t} = i\omega_A F(t) + i(\omega - \omega_A)\frac{F(t+1) + F(t\bar{-}1)}{2\Delta t}. \tag{52}$$

where ω and ω_A are two parameters.
Filter is used for two purposes:

1. Reducing damping
2. Stability

Sixth Step - FLTW method

By the concept of equation (51), we can update our variables using FLTW method. Difference equation is,

$$Q^{n+1} = \frac{w}{2}(Q^n + Q^{n-2}) + (1-w)Q^{n-1}, \tag{53}$$

where Q is arbitary variable. FLTW reduces to FLT for $w = 1$ and to leap-frog for $w = 0$.
Rewrite (53),

$$Q^{n+1} = Q^{n-1} + \frac{w}{2}(Q^n - 2Q^{n-1} + Q^{n-2}). \tag{54}$$

You can view last term as a **diffusion term** in time, that can 'smooth' solution that we got.

Sixth Step - FLTW method

By the concept of equation (51), we can update our variables using FLTW method. Difference equation is,

$$Q^{n+1} = \frac{w}{2}(Q^n + Q^{n-2}) + (1-w)Q^{n-1}, \tag{53}$$

where Q is arbitary variable. FLTW reduces to FLT for $w = 1$ and to leap-frog for $w = 0$.
Rewrite (53),

$$Q^{n+1} = Q^{n-1} + \frac{w}{2}(Q^n - 2Q^{n-1} + Q^{n-2}). \tag{54}$$

You can view last term as a **diffusion term** in time, that can 'smooth' solution that we got.

Sixth Step - FLTW method

Update using FLTW method,

```
712 DO 745 K=1,K1
DO 745 J=2,J1
DO 745 I=2,I1
T1(I,J,K)=OFLTW*TLF(I,J,K)+FLTW*(T1(I,J,K)+T2(I,J,K))
U1(I,J,K)=OFLTW*ULF(I,J,K)+FLTW*(U1(I,J,K)+U2(I,J,K))
V1(I,J,K)=OFLTW*VLF(I,J,K)+FLTW*(V1(I,J,K)+V2(I,J,K))
TLF(I,J,K)=T2(I,J,K)
ULF(I,J,K)=U2(I,J,K)
745 VLF(I,J,K)=V2(I,J,K),
```

where OFLTW=0.9, FLTW=$5.0000001E-02$.
U1 is the average values of ULF,U1,U2, so do V1 and T1. U1 is previous time step center velocity, U2 is what we want to get, ULF is the previous time step center velocity.

Sixth Step - FLTW method

After doing this, Biharmonic filter is applied by call **SUBROUNTINE BFLTXY** in order to reduce surface noise.
Now, the whole computation is complete, ready for next time step's computation.

We may think about why we need to 'correct' variables, now see the model output:

Sixth Step - FLTW method

After doing this, Biharmonic filter is applied by call **SUBROUNTINE BFLTXY** in order to reduce surface noise.
Now, the whole computation is complete, ready for next time step's computation.

We may think about why we need to 'correct' variables, now see the model output:

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the ‘cell’. or on the ‘face’. When mention to variables, I’ll note that variable is on the ‘face’ or in the ‘cell’ again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. ‘Space’ then ‘Time’.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

CCD in DieCast model

Before considering how to insert CCD scheme into DieCast, we have to clarify what variables are in the 'cell'. or on the 'face'. When mention to variables, I'll note that variable is on the 'face' or in the 'cell' again. Some notes are below:

1. Pressure is cell-quantity in general.
2. Pressure gradient is more important than pressure.
3. Velocity variables are cell-quantities when updating momentum equation,
4. but are face-quantities when checking incompressibility.
5. CCD applies to solve variables respect to face- or cell-quantity.
6. Boundary conditions needs to be inserted to the scheme.
7. Semi-discretized scheme. 'Space' then 'Time'.

Pressure and Pressure gradient

We have: P_s^{n-1} surface pressure.
Control equation: $\frac{\partial p}{\partial z} = -\rho g$.
Goal: get p field over the domain in the center.

$$\frac{7}{16}\left(\left(\frac{\delta p}{\delta z}\right)_{i+1} + \left(\frac{\delta p}{\delta z}\right)_{i-1}\right) + \left(\frac{\delta p}{\delta z}\right)_i - \frac{h}{16}\left(\left(\frac{\delta^2 p}{\delta z^2}\right)_{i+1} - \left(\frac{\delta^2 p}{\delta z^2}\right)_{i-1}\right) = \frac{5}{16h}(p_{i+1} - p_{i-1}) \quad (55)$$

$$\frac{9}{8h}\left(\left(\frac{\delta p}{\delta z}\right)_{i+1} - \left(\frac{\delta p}{\delta z}\right)_{i-1}\right) + \left(\frac{\delta p}{\delta z}\right)_i - \frac{1}{8}\left(\left(\frac{\delta^2 p}{\delta z^2}\right)_{i+1} + \left(\frac{\delta^2 p}{\delta z^2}\right)_{i-1}\right) = \frac{3}{h^2}(p_{i+1} - 2p_i + p_{i-1}) + B.C. \quad (56)$$

$$\frac{\partial p}{\partial z} = -\rho g, \Longrightarrow \int_0^h \frac{\partial p}{\partial z} dz = -\int_0^h \rho g dz, \Longrightarrow p(h) - p(0) = -(\rho g) \times (h - 0). \quad (57)$$

Pressure and Pressure gradient

We have: P_i.
Goal: get $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$ over the domain in the cell.

$$\frac{7}{16}\left(\left(\frac{\Delta p}{\Delta x}\right)_{i+1}+\left(\frac{\Delta p}{\Delta x}\right)_{i-1}\right)+\left(\frac{\Delta p}{\Delta x}\right)_{i}-\frac{h}{16}\left(\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i+1}-\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i-1}\right) = \frac{5}{16h}(p_{i+1}-p_{i-1}) \quad (58)$$

$$\frac{9}{8h}\left(\left(\frac{\Delta p}{\Delta x}\right)_{i+1}-\left(\frac{\Delta p}{\Delta x}\right)_{i-1}\right)+\left(\frac{\Delta p}{\Delta x}\right)_{i}-\frac{1}{8}\left(\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i+1}+\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i-1}\right) = \frac{3}{h^2}(p_{i+1}-2p_i+p_{i-1})+B.C. \quad (59)$$

$$p_i. \quad (60)$$