Numerical PDE - Final Project

Yu-Chiao Liang

08/29/2008

Outline

Intruduction

Solving Navier-Stokes equations is the very tedious problem in fluid dynamics.

Finite Difference scheme is intoduced to solve NSE.

We'll write Navier-Stokes equations as abbreviation form:

$$NSE$$

Mathematical Derivation

Solving Navier-Stokes equations is the very tedious problem in fluid dynamics.

Finite Difference scheme is intoduced to solve NSE.

We'll write Navier-Stokes equations as abbreviation form:

$$ajsdkl; f \tag{1}$$

Numerical Methods for NSE

When we discuss about numerical methods for NSE, important questions we should keep in our mind, that is:

- *'Are the existing solution algorithms for incompressible flow problems already optimal or os further, maybe even tremendous improvement necessary?'*
- *'Could our solution of numerical methods used for solving NSE be an 'efficient' and 'robust' one?'*
- *'What's the bottleneck of our numerical method? What are the advantages? What are the shortcomings?'*

Recall NSE

NSE has general form as below,

$$asdf \tag{2}$$

with appropriate boundary conditions and initial condition.
That's take a roughly look at discretization of terms in NSE.

Discretization of Terms in NSE - Convective term

Differential and integral forms of convective term are,

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} \qquad \text{and} \qquad \int_S \rho u_i \mathbf{v} \cdot \mathbf{ndS}. \tag{3}$$

Noted that,

1. Convective term is non-linear.
2. Convective term can be solved by FD, FV, FE ... etc.

Discretization of Terms in NSE - Convective term

Differential and integral forms of convective term are,

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} \qquad \text{and} \qquad \int_S \rho u_i \mathbf{v} \cdot \mathbf{ndS}. \tag{3}$$

Noted that,

1. Convective term is non-linear.
2. Convective term can be solved by FD, FV, FE ... etc.

Discretization of Terms in NSE - Viscous term

Differential and integral forms of viscous term are,

$$\frac{\partial \tau_{ij}}{\partial x_j} \qquad \text{and} \qquad \int_S (\tau_{ij}\mathbf{i_j}) \cdot \mathbf{ndS}. \tag{4}$$

Noted that,

1. For Newtonian and incompressible flow, $\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$.
2. Convective term can be solved by FD, FV, FE ... etc.
3. Diffusion terms and extra terms disappear for incompressible flows.
4. Other distributions of viscosity such as bulk viscosity can be identified.

Discretization of Terms in NSE - Viscous term

Differential and integral forms of viscous term are,

$$\frac{\partial \tau_{ij}}{\partial x_j} \qquad \text{and} \qquad \int_S (\tau_{ij}\mathbf{i_j}) \cdot \mathbf{ndS}. \tag{4}$$

Noted that,

1. For Newtonian and incompressible flow, $\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$.
2. Convective term can be solved by FD, FV, FE ... etc.
3. Diffusion terms and extra terms disappear for incompressible flows.
4. Other distributions of viscosity such as bulk viscosity can be identified.

Discretization of Terms in NSE - Pressure term

Differential forms of pressure term is,

$$-\frac{\partial p}{\partial x_i}. \tag{5}$$

Noted that,

1. This can be solved by FD, FV, FE ... etc.
2. Numerical methods for this derivative term might be differet from velocity terms since grid-setting of pressure and velocity may not coincide on the same grid.
3. Operator consistence must be paid attention all the time.

Discretization of Terms in NSE - Pressure term

Differential forms of pressure term is,

$$-\frac{\partial p}{\partial x_i}. \tag{5}$$

Noted that,

1. This can be solved by FD, FV, FE ... etc.
2. Numerical methods for this derivative term might be differet from velocity terms since grid-setting of pressure and velocity may not coincide on the same grid.
3. Operator consistence must be paid attention all the time.

Discretization of Terms in NSE - Pressure term

Integral form of pressure term has two approach concept. One is conservative approach,

$$-\int_S p\mathbf{i}_i \cdot \mathbf{n} dS. \tag{6}$$

That is treated pressre term as a surface force.

The other is non-conservative approach,

$$-\int_\Omega \nabla p \cdot \mathbf{i}_i d\Omega, \tag{7}$$

by viewing it in volumetric form.

Discretization of Terms in NSE - Pressure term

Integral form of pressure term has two approach concept. One is conservative approach,

$$-\int_S p\mathbf{i}_i \cdot \mathbf{n} dS. \tag{6}$$

That is treated pressre term as a surface force.

The other is non-conservative approach,

$$-\int_\Omega \nabla p \cdot \mathbf{i}_i d\Omega, \tag{7}$$

by viewing it in volumetric form.

Discretization of Terms in NSE - Pressure term

Noted that,

1. Difference of these integral forms appears for FV, doesn't for FD.
2. Conservative approach plays an important role for solving NSE, related to efficiency and accuracy of the numerical solution method.
3. Non-conservative approach introduces a global non-conservative error, which may be significant for finite grid size.

Discretization of Terms in NSE - Body-force terms

Body force terms can be conservative, such as gravitive force, or non-conservative. These terms are function of one or more variables, velocity, time for examples. If relating to velocity field, we might treat them by using central differece scheme in order to avoid destablization of the diagonal dominance of the matrix. Otherwise , the extra term is needed to be icluded.
In FV methods, these terms are integrated over the CV volume, usually using mean value approach, therefore they are calculated at the CV center.

Conservation

When we talk about conservation of something, we immediately think of something that doesn't change, more specific, in a specific region. Quite a few conservation laws are discribed in physical and mathematical problems, such as *First Thermodynamic Law*, *Energy conservation*.

A question arise when we mention conservative concept of a specific physical variable:

- How can we say that is conservative? By what reason and by what language?

We have to think about it first when we go on.

Conservation-Momentum

In fluid dynamics, *Energy conservation* and *Momentum conservation* are usually discussed. Discretized numerical schemes should, somehow, maintain these conservation.
Momentun conservation....

Conservation-Energy

Energy conservation is a more complex issue, in different physical conditions, significance of specific type of energy might be different, such as ocassion that heat transfer is important than kinetic energy. Energy equation can be derived by taking the scalar product of the momentum equation with the velociy.
We are primarily focus on kinetic energy conservation. In FV point of view, integral form of kinetic energy equation is,

$$\frac{\partial}{\partial t}\int_\Omega \rho\frac{v^2}{2}d\Omega = - \int_S \rho\frac{v^2}{2}\mathbf{v}\cdot\mathbf{n}dS - \int_S p\mathbf{v}\cdot\mathbf{n}dS + \int_S (\mathbf{S}\cdot\mathbf{v})\cdot\mathbf{n}dS - \int_\Omega (S : grad\mathbf{v} - pdiv(\mathbf{v}) + \rho\mathbf{b}\cdot\mathbf{v})d\Omega, \quad (8)$$

after applying Gauss' theorem.

Conservation-Energy

Energy conservation is a more complex issue, in different physical conditions, significance of specific type of energy might be different, such as ocassion that heat transfer is important than kinetic energy. Energy equation can be derived by taking the scalar product of the momentum equation with the velociy.
We are primarily focus on kinetic energy conservation. In FV point of view, integral form of kinetic energy equation is,

$$\begin{aligned}\frac{\partial}{\partial t}\int_\Omega \rho\frac{v^2}{2}d\Omega = &- \int_S \rho\frac{v^2}{2}\mathbf{v}\cdot\mathbf{n}dS - \int_S p\mathbf{v}\cdot\mathbf{n}dS + \int_S(\mathbf{S}\cdot\mathbf{v})\cdot\mathbf{n}dS \\ &- \int_\Omega (S : grad\mathbf{v} - pdiv(\mathbf{v}) + \rho\mathbf{b}\cdot\mathbf{v})d\Omega, \qquad (8)\end{aligned}$$

after applying Gauss' theorem.

Conservation-Energy

Several points of the energy equation are worth mentioning.

- If volumetric integral terms vanish, convection or pressure terms affect kinetic energy only on the surface of the CV.
- Kinetic energy in globally conserved.
- Kinetic energy equation shouldn't look as separate equation due to it is derived from momentun equations.
- Local Energy conservation can imply that velocity is bounded (in a CV).
- Energy method for proving stability of a numerical method.

Conservation-Energy

- Energy conservation says nothing about convergence or accuracy, an accurate convergent numerical scheme can be not consevative.
- Kinetic energy conservation is important in computing unsteady flows.
- We ask numerical method to hold some form of *Discretized Energy Conservation Equation*, but not easy to hold.
- Pressure terms are most important, since it can limit operators choosen for '*velocity*' and '*pressure*' terms. Derivation is shown in the next page.

Conservation-Energy

Consider equality:

$$\mathbf{v} \cdot grad(p) = div(p\mathbf{v}) - pdiv(\mathbf{v}), \tag{9}$$

general discretized scheme of this equaliy in integral form is:

$$\sum_{i=1}^{N} u_i G_i p \Delta\Omega = \sum_{S_b} p v_n \Delta S - \sum_{N} p D_i u_i \Delta\Omega, \tag{10}$$

where N is the number of CVs, S_b is the velocity that is normal to the surface, and $D_i u_i$ is the discretized velocity divergence in the continuity equation.

Conservation-Energy

Rewrite (10), we can get:

$$\sum_{i=1}^{N}(u_i G_i p + p D_i u_i)\Delta\Omega = \text{surface terms.} \tag{11}$$

If we use backward FD for pressure gradient terms, forward FD for divergence operator, we can get '*okay*' numerical 'consistent' scheme, that is:

$$\sum_{i=1}^{N}[(p_i - p_{i-1})u_i + (u_{i+1} - u_i)p_i] = u_{N+1}p_N - u_1 p_0. \tag{12}$$

Right hand side values are determined by only surface grid points.

Conservation-Energy

If we use .....

Conservation-Energy

- Energy equation gives constraints when solving Poisson equation. 'Divergence operator' *v.s* 'Gradient operator'.
- We should concern energy transfer, from kinetic to internal energy due to viscous dissipation. It might be significant in both compressible and impressible flows.
- The Crank-Nicolson is a good choice for time iteration scheme, due to its prperty of energy conservation.
- Momentum and energy conservation are governed by the same equations, thus make construction of numerical method difficult.

Conservation-Energy

- Discretize strong conservative form of momentum equations with FV metohd usually assures energy conservation globally.
- Adjustments are made if our numerical results are not energy conservative.
- For unsteady flows, such as simulation of global weather or simulation of turbulent, if energy conservation doesn't hold growth of kinetic energy or instabily might occur.
- For rotational flow, *Angular Momentum Conservation* is important that we should concern. Central difference schemes are much betther than upwind schemes w.r.t. angular momentum conservation.

Variable Arrangement on the Grid

There are at least two types of variable arrangement, one is **Colocated Arrangement**, the other is **Staggered Arrangements**. I think any numerical scheme should clarified the relationship between these two arrangement for specific variable we want to get, since FV or FD concept can therefore easily insert to help to sovle problems.

Colocated Arrangement

Advantages of colocated arrangement are:

- Useful in complicated solution domain.
- with boundaries that have slope discontinuities.
- with boundary conditions are discontinuous.
- avoid variables located at singularities.

Disadvantages are:

- for long time computation of incompressible flows, difficulties about pressure-velocity coupling arize.
- oscillations in the pressure term occur.

Staggered Arrangement

Harlow and Welsh (1965) first introduce staggered arrangement in Cartesian cordinates. The biggest advantage of the staggered arrangement are,

1. Strong coupling between the velocities and the pressure, helping avoid some problems of convergence and oscillations in pressure and velocity fields.
2. Conservative of kinetic energy.

Other staggered arrangements are to be introduce, such as partially staggered ALE method. But drawbacks of oscillation of pressure and velocity might occur.

Calculation of the Pressure

NSE is hard to solve because velocity and pressure are coupled in three momentum equations, therefore no independnet equation for the pressure term is valid.

- In compressible flows the continuity equaiton can be used to determine the density and the pressure can be calculated from state equations.
- In incompressible flows or low Mach number flows, continuity equation is not valid.

Our first goal is to derived pressure equation by divergence of momentum equation with continuity equation and then solve it by numerical method.

Calculation of the Pressure

NSE is hard to solve because velocity and pressure are coupled in three momentum equations, therefore no independnet equation for the pressure term is valid.

- In compressible flows the continuity equaiton can be used to determine the density and the pressure can be calculated from state equations.
- In incompressible flows or low Mach number flows, continuity equation is not valid.

Our first goal is to derived pressure equation by divergence of momentum equation with continuity equation and then solve it by numerical method.

Calculation of the Pressure

NSE is hard to solve because velocity and pressure are coupled in three momentum equations, therefore no independnet equation for the pressure term is valid.

- In compressible flows the continuity equaiton can be used to determine the density and the pressure can be calculated from state equations.
- In incompressible flows or low Mach number flows, continuity equation is not valid.

Our first goal is to derived pressure equation by divergence of momentum equation with continuity equation and then solve it by numerical method.

Calculation of the Pressure

NSE is hard to solve because velocity and pressure are coupled in three momentum equations, therefore no independnet equation for the pressure term is valid.

- In compressible flows the continuity equaiton can be used to determine the density and the pressure can be calculated from state equations.
- In incompressible flows or low Mach number flows, continuity equation is not valid.

Our first goal is to derived pressure equation by divergence of momentum equation with continuity equation and then solve it by numerical method.

Derivation of pressure equation

Take divergence to the momentum equation and then use continuity equation to simplify it, we can get:

$$div(gradp) = -div\left[div(\rho \mathbf{vv} - S) - \rho \mathbf{b} + \frac{\partial(\rho \mathbf{v})}{\partial t}\right], \tag{13}$$

component form:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial x_j}(\rho u_i u_j - \tau_{ij})\right] + \frac{\partial(\rho b_i)}{\partial x_i} + \frac{\partial^2 \rho}{\partial t^2}. \tag{14}$$

For the case of constant density and viscosity, the viscous and unsteady terms disappear by virtue of the continuity equation leaving:

Derivation of pressure equation

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial(\rho u_i u_j)}{\partial x_j}\right]. \tag{15}$$

This equation can be solved by numerical method for elliptic type equations, in this case, it can be viewed as a Poisson equation, which contains a *'Laplacian'* operator.

In numerical point of view, the *'Laplacian'* operator is products of divergence and gradint, so that two operators have to be choosen carefully and consistently. To maintain consistency, it is best to derive pressure equation from discretized momentum and continuity equations rather than discretize equation (15) directly. Our frist example for solving NSE can see what's going on.

Derivation of pressure equation

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial(\rho u_i u_j)}{\partial x_j}\right]. \tag{15}$$

This equation can be solved by numerical method for elliptic type equations, in this case, it can be viewed as a Poisson equation, which contains a *'Laplacian'* operator.
In numerical point of view, the *'Laplacian'* operator is products of divergence and gradint, so that two operators have to be choosen carefully and consistently. To maintain consistency, it is best to derive pressure equation from discretized momentum and continuity equations rather than discretize equation (15) directly. Our frist example for solving NSE can see what's going on.

First Example for Solving NSE - Explicit Time Advance

1. Semi-discretize (only in space $\frac{\delta}{\delta x_j}$ not in time) momentum equation:

$$\frac{\partial(\rho u_i)}{\partial t} = -\frac{\delta(\rho u_i u_j)}{\delta x_j} - \frac{\delta p}{\delta x_i} + \frac{\delta \tau_{ij}}{\delta x_j} = H_i - \frac{\delta p}{\delta x_i}, \tag{16}$$

where H_i is notation for the advective and viscous terms, which is not important here.

2. Discretize for time advancement by *Euler forward* method:

$$(\rho u_i)^{n+1} - (\rho u_i)^n = \Delta t \left(H_i^n - \frac{\delta p^n}{\delta x_i} \right), \tag{17}$$

where H_i is calculated by u_i at time step n.

First Example for Solving NSE - Explicit Time Advance

3. Suppose divegence of momentum is free at any time step, in general this doesn't hold, that is to satisfy continuity equation:

$$\frac{\delta(\rho u_i)^m}{\delta x_i} = 0, \tag{18}$$

where $m = n, n+1...$, different time steps. Now take numerical divergence on (17),

$$\frac{\delta(\rho u_i)^{n+1}}{\delta x_i} - \frac{\delta(\rho u_i)^n}{\delta x_i} = \Delta t \left[\frac{\delta}{\delta x_i} \left(H_i^n - \frac{\delta p^n}{\delta x_i} \right) \right]. \tag{19}$$

Force LHS of equation above to be zero maintaining divergence free.

First Example for Solving NSE - Explicit Time Advance

4. Poisson equation for the perssure p^n then is derived:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta p^n}{\delta x_i}\right) = \frac{\delta H_i^n}{\delta x_i}, \tag{20}$$

note that red operator is inherited from continuity equation, blue operator is from momentum equation. If equation (20) holds, then velocity field at time step $n+1$ is divergence free in discretized operator.

Let's go back to equation 18, this equation does not always hold with discretized divergence operator. In particulr, initial data can fit these equations? That's we must take care about and make modification if necessary.

First Example for Solving NSE - Explicit Time Advance

4. Poisson equation for the perssure p^n then is derived:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta p^n}{\delta x_i}\right) = \frac{\delta H_i^n}{\delta x_i}, \tag{20}$$

note that red operator is inherited from continuity equation, blue operator is from momentum equation. If equation (20) holds, then velocity field at time step $n+1$ is divergence free in discretized operator.

Let's go back to equation 18, this equation does not always hold with discretized divergence operator. In particulr, initial data can fit these equations? That's we must take care about and make modification if necessary.

First Example for Solving NSE - Explicit Time Advance

The following algorithm for this explicit time-advancing scheme summaried below:

1. Start with a velocity field u_i^n at time t_n which is assume to be divergence free, if not correct it.
2. Compute H_i^n and its divergence.
3. Solve the Poisson equation for the pressure p^n.
4. Compute the velocity field at $n+1$ time step, which is **intrisically** divergence free.

Drawbacks:

- Not accurate!
- Stability region is small

First Example for Solving NSE - Explicit Time Advance

The following algorithm for this explicit time-advancing scheme summaried below:

1. Start with a velocity field u_i^n at time t_n which is assume to be divergence free, if not correct it.
2. Compute H_i^n and its divergence.
3. Solve the Poisson equation for the pressure p^n.
4. Compute the velocity field at $n+1$ time step, which is **intrisically** divergence free.

Drawbacks:

- Not accurate!
- Stability region is small

First Example for Solving NSE - Explicit Time Advance

The following algorithm for this explicit time-advancing scheme summaried below:

1. Start with a velocity field u_i^n at time t_n which is assume to be divergence free, if not correct it.
2. Compute H_i^n and its divergence.
3. Solve the Poisson equation for the pressure p^n.
4. Compute the velocity field at $n+1$ time step, which is **intrisically** divergence free.

Drawbacks:

- Not accurate!
- Stability region is small

First Example for Solving NSE - Explicit Time Advance

The following algorithm for this explicit time-advancing scheme summaried below:

1. Start with a velocity field u_i^n at time t_n which is assume to be divergence free, if not correct it.
2. Compute H_i^n and its divergence.
3. Solve the Poisson equation for the pressure p^n.
4. Compute the velocity field at $n+1$ time step, which is **intrisically** divergence free.

Drawbacks:

- Not accurate!
- Stability region is small

First Example for Solving NSE - Explicit Time Advance

The following algorithm for this explicit time-advancing scheme summaried below:

1. Start with a velocity field u_i^n at time t_n which is assume to be divergence free, if not correct it.
2. Compute H_i^n and its divergence.
3. Solve the Poisson equation for the pressure p^n.
4. Compute the velocity field at $n+1$ time step, which is **intrisically** divergence free.

Drawbacks:

- Not accurate!
- Stability region is small

First Example for Solving NSE - Explicit Time Advance

The following algorithm for this explicit time-advancing scheme summaried below:

1. Start with a velocity field u_i^n at time t_n which is assume to be divergence free, if not correct it.
2. Compute H_i^n and its divergence.
3. Solve the Poisson equation for the pressure p^n.
4. Compute the velocity field at $n+1$ time step, which is **intrisically** divergence free.

Drawbacks:

- Not accurate!
- Stability region is small

Second Example for Solving NSE - implicit Time Advance

Similar as previous explicit method, we use classical backward Euler method to discretize momentum equation (46):

$$(\rho u_i)^{n+1} - (\rho u_i)^n = \Delta t \left(-\frac{\delta(\rho u_i u_j)^{n+1}}{\delta x_j} - \frac{\delta p^{n+1}}{\delta x_i} + \frac{\delta \tau_{ij}^{n+1}}{\delta x_j} \right). \quad (21)$$

Then derive pressure equation:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta p^{n+1}}{\delta x_i}\right) = \frac{\delta H_i^{n+1}}{\delta x_i}, \quad (22)$$

where H_i is notation for the advective and viscous terms, which is not important here.

Second Example for Solving NSE - implicit Time Advance

Similar as previous explicit method, we use classical backward Euler method to discretize momentum equation (46):

$$(\rho u_i)^{n+1} - (\rho u_i)^n = \Delta t \left(-\frac{\delta(\rho u_i u_j)^{n+1}}{\delta x_j} - \frac{\delta p^{n+1}}{\delta x_i} + \frac{\delta \tau_{ij}^{n+1}}{\delta x_j} \right). \quad (21)$$

Then derive pressure equation:

$$\frac{\delta}{\delta x_i} \left(\frac{\delta p^{n+1}}{\delta x_i} \right) = \frac{\delta H_i^{n+1}}{\delta x_i}, \quad (22)$$

where H_i is notation for the advective and viscous terms, which is not important here.

Second Example for Solving NSE - implicit Time Advance

Problem arises here that pressure cannot be calculated until velocity field at time $n+1$ is calculated. That is Poisson equation and the momentum equations have to be solved simultaneously. This can only be done by some iteration method.

Still another problem is that even if the pressure were known, equaion (22) are large system of non-linear equations which must be solved for the velocity field.

We can view this system as the matrix of FD Laplace equation, if we want to solve this accurately, Newton-Raphson iteration method or a secant method can be used.

Second Example for Solving NSE - implicit Time Advance

Problem arises here that pressure cannot be calculated until velocity field at time $n+1$ is calculated. That is Poisson equation and the momentum equations have to be solved simultaneously. This can only be done by some iteration method.

Still another problem is that even if the pressure were known, equaion (22) are large system of non-linear equations which must be solved for the velocity field.

We can view this system as the matrix of FD Laplace equation, if we want to solve this accurately, Newton-Raphson iteration method or a secant method can be used.

Second Example for Solving NSE - implicit Time Advance

An alternating way for solving non-linear matrix directly is by linearizing the non-linear terms in equation (22). Suppose that,

$$u_i^{n+1} = u_i^n + \Delta u_i \qquad p_i^{n+1} = p_i^n + \Delta p_i, \tag{23}$$

then non-linear term in H_i^n can be expressed as:

$$u_i^{n+1} u_j^{n+1} = u_i^n u_j^n + u_i^n \Delta u_j + u_j^n \Delta u_i + \Delta u_i \Delta u_j. \tag{24}$$

Assume that, for small Δt, $\Delta u_i \ \Delta t \frac{\partial u_i}{\partial t}$, so the last term in this (24) is second order in Δt and is smaller in magnitude than the error made in the time discretization. If we use second order Crank-Nicolson scheme, this term has error of the same order as the spatial discretization, thus can be neglected.

Second Example for Solving NSE - implicit Time Advance

An alternating way for solving non-linear matrix directly is by linearizing the non-linear terms in equation (22). Suppose that,

$$u_i^{n+1} = u_i^n + \Delta u_i \qquad p_i^{n+1} = p_i^n + \Delta p_i, \tag{23}$$

then non-linear term in H_i^n can be expressed as:

$$u_i^{n+1} u_j^{n+1} = u_i^n u_j^n + u_i^n \Delta u_j + u_j^n \Delta u_i + \Delta u_i \Delta u_j. \tag{24}$$

Assume that, for small Δt, Δu_i $\Delta t \frac{\partial u_i}{\partial t}$, so the last term in this (24) is second order in Δt and is smaller in magnitude than the error made in the time discretization. If we use second order Crank-Nicolson scheme, this term has error of the same order as the spatial discretization, thus can be neglected.

Second Example for Solving NSE - implicit Time Advance

Plug u_i^{n+1} and p_i^{n+1} into equation (21), we can get:

$$\rho\Delta u_i = \Delta t(-\frac{\delta(\rho u_i u_j)^n}{\delta x_j} - \frac{\delta(\rho u_i^n \Delta u_j)}{\delta x_j} - \frac{\delta(\rho \Delta u_i u_j^n)}{\delta x_j} - \frac{\delta p^n}{\delta x_i} - \frac{\delta \Delta p}{\delta x_i} + \frac{\delta \tau_{ij}^n}{\delta x_j} + \frac{\delta \Delta \tau_{ij}}{\delta x_j}) \quad (25)$$

in which we can see that non-linear term are removed. However we still need to solve a large system of linear equations.
Alternating direction implicit (ADI) method may be useful by spliting the equations into a series of one dimensional problems, each of which is block tridiagonal.

Second Example for Solving NSE - implicit Time Advance

Summarize discussions of ADI method, algorithm can be stated as:

- Calculate increment velocity from linearized momentum equation, get u_i^*.
- Solve Poisson equation for pressure correction based on continuity equation:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta \Delta p}{\delta x_i}\right) = \frac{1}{\Delta t}\frac{\delta(\rho u_i^*)}{\delta x_i}. \tag{26}$$

- Update the velocity:

$$u_i^{n+1} = u_i^* - \frac{\Delta t}{\rho}\frac{\delta \Delta p}{\delta x_i}, \tag{27}$$

which satisfy continuity equation.

Second Example for Solving NSE - implicit Time Advance

Summarize discussions of ADI method, algorithm can be stated as:

- Calculate increment velocity from linearized momentum equation, get u_i^*.
- Solve Poisson equation for pressure correction based on continuity equation:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta \Delta p}{\delta x_i}\right) = \frac{1}{\Delta t}\frac{\delta(\rho u_i^*)}{\delta x_i}. \tag{26}$$

- Update the velocity:

$$u_i^{n+1} = u_i^* - \frac{\Delta t}{\rho}\frac{\delta \Delta p}{\delta x_i}, \tag{27}$$

which satisfy continuity equation.

Second Example for Solving NSE - implicit Time Advance

But we have to pay twice expensive as the explicit method per time step.

Advantages of this implicit method are:

1. provide accurate solution for unsteady problem.
2. for steady problem without concerning about stability implicit method are useful.

Drawbacks are:

1. time step will be rather small.
2. error in linearized method no longer negligible for calculation of steady state problems, pressure-correction methond can be used avoiding this problem.

Second Example for Solving NSE - implicit Time Advance

But we have to pay twice expensive as the explicit method per time step.

Advantages of this implicit method are:

1. provide accurate solution for unsteady problem.
2. for steady problem without concerning about stability implicit method are useful.

Drawbacks are:

1. time step will be rather small.
2. error in linearized method no longer negligible for calculation of steady state problems, **pressure-correction methond can be used avoiding this problem.**

Pressure-Correction Methods

Some numerical methods talked before are:

- Newton-Raphson iteration method.
- Secent iteration method.
- Crank-Nicolson scheme.
- Alternating direction implicit (ADI) problem.

Pressure-Correction Methods

For steady problems, we can view them as unsteady ones that its solution is going to reach steady state. So, large time steps should be used in general, implicit methods thus happen to restrictions due to stability conditions, therefore are preferred for steady and slow-transient flows.

Some popular implicit methods use pressure equation to enforce mass conservation at each time step, we call them Pressure-Correction type methods.

Using pressure equation for mass conservation is the main concept of Pressure-Correction method.

Pressure-Correction Methods

For steady problems, we can view them as unsteady ones that its solution is going to reach steady state. So, large time steps should be used in general, implicit methods thus happen to restrictions due to stability conditions, therefore are preferred for steady and slow-transient flows.

Some popular implicit methods use pressure equation to enforce mass conservation at each time step, we call them Pressure-Correction type methods.

Using pressure equation for mass conservation is the main concept of Pressure-Correction method.

Pressure-Correction Methods

Discretized form of momentum equation without including pressure gradient term in the source term is writtem as:

$$A_P^{u_i} u_{i,P}^{n+1} + \sum_l A_l^{u_i} u_{i,l}^{n+1} = Q_{u_i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i} \right)_P , \tag{28}$$

where P is the index of the node, l denotes the neighbor points that appear in the discretized momentum equaiton. Q is the source term containing variables that can be calculated by u_i^n and body force or other lineaized terms at time steps $n+1$.

Noted that:

- Iteration methods seem the only choice for solving this equation.
- Steady flows and unsteady flows.

Pressure-Correction Methods

Discretized form of momentum equation without including pressure gradient term in the source term is writtem as:

$$A_P^{u_i} u_{i,P}^{n+1} + \sum_l A_l^{u_i} u_{i,l}^{n+1} = Q_{u_i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i} \right)_P , \tag{28}$$

where P is the index of the node, l denotes the neighbor points that appear in the discretized momentum equaiton. Q is the source term containing variables that can be calculated by u_i^n and body force or other lineaized terms at time steps $n+1$.
Noted that:

- Iteration methods seem the only choice for solving this equation.
- Steady flows and unsteady flows.

Pressure-Correction Methods

Iterations should be clearified as:

- *Outer iteration* is defined as iteration within one time step, in which coefficient and source matrices are updated.
- *Inner iteration* performed on linear systems with fixed coefficients.

On each outer iteration, we ,therefore, solve this:

$$A_P^{u_i} u_{i,P}^{m*} + \sum_l A_l^{u_i} u_{i,l}^{m*} = Q_{u_i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P . \tag{29}$$

RHS of this equation are calculated using results at the preceding outer iteration when beginning outer iteration.

Pressure-Correction Methods

Now let's begin Pressure-Correction processes.

1. Rewrite (29), velocity at node P can be stated as:

$$u_{i,P}^{m*} = \frac{Q_{u_i}^{m-1} - \sum_l A_l^{u_i} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P . \tag{30}$$

Generally speaking, $u_{i,P}^{m*}$ does not satisfy continuity equation.

Asterisk $*$ means that velocities calculated above are not final values.

For convenience, we let first term on RHS to be $\bar{u}_{i,p}^{m*}$.

2. Consider continuity equation:

$$\frac{\delta(\rho u_i^m)}{\delta x_i} = 0, \tag{31}$$

which can be achieved by correcting the pressure field.

Pressure-Correction Methods

Now let's begin Pressure-Correction processes.

1. Rewrite (29), velocity at node P can be stated as:

$$u_{i,P}^{m*} = \frac{Q_{u_i}^{m-1} - \sum_l A_l^{u_i} u_{i,l}^{m*}}{A_P^{u_i}} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P . \tag{30}$$

Generally speaking, $u_{i,P}^{m*}$ does not satisfy continuity equation.
Asterisk $*$ means that velocities calculated above are not final values.
For convenience, we let first term on RHS to be $\bar{u}_{i,p}^{m*}$.

2. Consider continuity equation:

$$\frac{\delta(\rho u_i^m)}{\delta x_i} = 0, \tag{31}$$

which can be achieved by correcting the pressure field.

Pressure-Correction Methods

Then we can get Poisson equation:

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_P^{u_i}}\left(\frac{\delta p^m}{\delta x_i}\right)\right]_P = \left[\frac{\delta(\rho \bar{u}_{i,p}^{m*})}{\delta x_i}\right]_P . \tag{32}$$

Noted that operators should be clearified as before.
After solving (32), we can get $u_{i,p}^m$. But now, although velocity field satisfies continuity equation, velocity and pressure fields do not satisfy momentun equation. We have to utilize another outer iteration until velocity and pressure fields satisfy both momentum and continuity equations.

Pressure-Correction Methods

Note again that methods of this kind, which first construct velocity field that does not satisfied continuity equation and then correct it by subtracting somthing (usually pressure gradient) are known as *projection methods*.

SIMPLE Concepts

After we get $u_{i,P}^{m*}$ and p^{m-1}, we can go forward by another algorithm of pressure-correction.

1. Suppose that:

$$u_i^m = u_i^{m*} + u' \quad \text{and} \quad p^m = p^{m-1} + p'. \tag{33}$$

Substitude them into momentum equation (29), then we obtain relation between the velocity and pressure corrections:

SIMPLE

$$u_{i,P}^{'} = \bar{u}_{i,P}^{'} - \frac{1}{A_P^{u_i}} \left(\frac{\delta p^{'}}{\delta x_i} \right)_P , \tag{34}$$

where $u_{i,P}^{'} = -\frac{\sum_l A_l^{u_i} u_{i,l}^{'}}{A_P^{u_i}}$.

2. Derive pressure-correction equation by using continuity equation and (34):

$$\frac{\delta}{\delta x_i} \left[\frac{\rho}{A_P^{u_i}} \left(\frac{\delta p^{'}}{\delta x_i} \right) \right]_P = \left[\frac{\delta (\rho u_{i,p}^{m*})}{\delta x_i} \right]_P + \left[\frac{\delta (\rho \bar{u}_i^{'})}{\delta x_i} \right]_P . \tag{35}$$

It is common practice to neglect $\bar{u}_i^{'}$ term in RHS.

SIMPLEC

Without negelecting term in equation (35), SIMPLEC method is introduced. 1. Approximate the velocity correction $u_i^{'}$ at any node,

$$u_{i,P}^{'} \approx \frac{\sum_l A_l^{u_i} u_{i,l}^{'}}{\sum_l A_l^{u_i}}. \tag{36}$$

and then approximate $\bar{u}_{i,P}^{'}$ as

$$\bar{u}_{i,P}^{'} \approx -u_{i,P}^{'} \frac{\sum_l A_l^{u_i}}{A_P^{u_i}}, \tag{37}$$

which , when inserted in (34) we can get:

SIMPLEC

$$u_{i,P}^{'} = -\frac{1}{A_P^{u_i} + \sum_l A_l^{u_i}} \left(\frac{\delta p^{'}}{\delta x_i} \right)_P . \tag{38}$$

Substitude this into equation (35), term that be neglected disappears immediately.

PISO

Still another method.....

Fractional Step Methods

Splitting velocity field

Streamfunction-Vorticity Methods

For impressible two-dimensional flows with constant flow properties, concept of streamfunction can be intriduced to solve NSE.

Artificial Compressibility Methods

Can a numerical method for the compressible flow also validate for incompressible flow? It is this question that generates methods of artificial compressibility method.
Compressible flow and incompressible flow are fundamentally difference in mathematical viewpoint,

- Compressible flow equations are hyperbolic-type, therefore, characteristic lines control signals travel at finite propagation speed.
- incompressible flow equations, however, are mixed parabolic-elliptic-type, lacking of time derivative terms.

Artificial Compressibility Methods

So, when we want to solve incompressible flow based on method of compressible one, adding time derivative of the pressure to continuity equation seems possible, that is why we call 'Artificial Compressibility'!!
Modified continuity can be adapted:

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial(\rho u_i)}{x_i} = 0, \tag{39}$$

where β is an artificial compressibility parameter whose value is key to the performance of this method (larger β means more 'incompressible').

Artificial Compressibility Methods

Addition of time derivative of pressure to the continuity equation means that we are no longer solving the true incompressible equations. For unsteady flow this method induce accurate and applicabiliy problem. At convergence, the time derivative is zero and the solution satisfied the incompressible equations.

There's big question that after modifying continuity equations, equations we solve are the consistence for original problem?

Whatever, that's move on!

Noted that intermediate velocity field $(u_i^*)^{n+1}$, does not satisfy the incompressible continuity equation,

$$\left[\frac{\delta(\rho u_i^*)^{n+1}}{\delta x_i}\right]_P = \Delta m_P. \tag{40}$$

Artificial Compressibility Methods

Addition of time derivative of pressure to the continuity equation means that we are no longer solving the true incompressible equations. For unsteady flow this method induce accurate and applicabiliy problem. At convergence, the time derivative is zero and the solution satisfied the incompressible equations.

There's big question that after modifying continuity equations, equations we solve are the consistence for original problem?

Whatever, that's move on!

Noted that intermediate velocity field $(u_i^*)^{n+1}$, does not satisfy the incompressible continuity equation,

$$\left[\frac{\delta(\rho u_i^*)^{n+1}}{\delta x_i}\right]_P = \Delta m_P. \tag{40}$$

Artificial Compressibility Methods

Several methods are available, but lots of problems will arize.
Here is an example.
1. Euler backward method used in time derivative due to reach steady solution as quickly as possible.

$$\frac{p_P^{n+1} - p_P^n}{\beta \Delta t} + \left[\frac{\delta(\rho u_i)}{\delta x_i}\right]_P^{n+1} = 0. \tag{41}$$

Problem is that velocity field is still unknow. 2. Derive Poisson equation for pressure or pressure correction by lineariation. The unknow quantity can be approximated:

$$(\rho u_i)^{n+1} \approx (\rho u_i^*)^{n+1} + \left[\frac{\partial(\rho u_i^*)}{\partial p}\right]^{n+1} (p^{n+1} - p^n) \tag{42}$$

Shallow Water Equation

Here is a question we should keep in our mind:
'Are the existing solution algorithms for incompressible flow problems already optimal or os further, maybe even tremendous improvement necessary?'

Outline

Outline:

- Purpose
- Mathematical Model
- Numerical Method
- Challenge
- Scientific Significance

Purpose

- Solve Riemann Problem with Shallow water equation.
- *ref: Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation*
- Try to be familiar with wave propagation algorithm.
- Dam-breaking testing problem.
- Two-phase Fluid problem.

Purpose

- Solve Riemann Problem with Shallow water equation.
- *ref: Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation*
- Try to be familiar with wave propagation algorithm.
- Dam-breaking testing problem.
- Two-phase Fluid problem.

Purpose

- Solve Riemann Problem with Shallow water equation.
- *ref: Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation*
- Try to be familiar with wave propagation algorithm.
- Dam-breaking testing problem.
- Two-phase Fluid problem.

Purpose

- Solve Riemann Problem with Shallow water equation.
- *ref: Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation*
- Try to be familiar with wave propagation algorithm.
- Dam-breaking testing problem.
- Two-phase Fluid problem.

Purpose

- Solve Riemann Problem with Shallow water equation.
- *ref: Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation*
- Try to be familiar with wave propagation algorithm.
- Dam-breaking testing problem.
- Two-phase Fluid problem.

Mathematical Model - Shallow Water Equation

Shallow water equation is based on some concepts:

- 'dimension approximation', it reduces the basic problem to one or two dimensions that are approximately horizontal.
- Integrated in the vertical from a lower boundary to the free surface, eliminating the free surface as a boundary of the solution region. So the domain is fixed.
- Two simplications:
  1. reducing the dimensions,
  2. fixing the solution domain.

Mathematical Model - Shallow Water Equation

Shallow water equation is based on some concepts:

- 'dimension approximation', it reduces the basic problem to one or two dimensions that are approximately horizontal.
- Integrated in the vertical from a lower boundary to the free surface, eliminating the free surface as a boundary of the solution region. So the domain is fixed.
- Two simplications:
  1. reducing the dimensions,
  2. fixing the solution domain.

Mathematical Model - Shallow Water Equation

Shallow water equation is based on some concepts:

- 'dimension approximation', it reduces the basic problem to one or two dimensions that are approximately horizontal.
- Integrated in the vertical from a lower boundary to the free surface, eliminating the free surface as a boundary of the solution region. So the domain is fixed.
- Two simplications:
  1. reducing the dimensions,
  2. fixing the solution domain.

Mathematical Model - Shallow Water Equation

Shallow water equation is based on some concepts:

- 'dimension approximation', it reduces the basic problem to one or two dimensions that are approximately horizontal.
- Integrated in the vertical from a lower boundary to the free surface, eliminating the free surface as a boundary of the solution region. So the domain is fixed.
- Two simplications:
  1. reducing the dimensions,
  2. fixing the solution domain.

Mathematical Model - Shallow Water Equation

In environmental fluid mechanics, shallow water model is valid. The most important assumption is that:

Theorem

the wave length must be long compared to the depth

Tide, tsunamis are examples by the fact that the ocean are at most a few kilometers deep but there breadth is hundreds of kilometers.
Some empirical information might be included to the shallow water equation for practical application.

Mathematical Model - Shalow Water Equaiton - First Step

Now let's derive the shallow water equation.

First, we state conservation equation and boundary conditions.

Consevation of Mass:

$$\int_{CS} (\vec{u} \cdot \vec{n}) dA = 0, \tag{43}$$

$$\frac{\partial u_i}{\partial x_i} = 0. \tag{44}$$

Conservation of Momentum:

$$\vec{F} = \int_{CV} \frac{\partial}{\partial t}(\rho \vec{u}) dV + \int_{CS} \rho \vec{u} (\vec{u} \cdot \vec{n}) dA, \tag{45}$$

$$\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} - \rho g \frac{\partial h}{\partial x_i} - \frac{\partial \tau + ji}{\partial x_j}. \tag{46}$$

Mathematical Model - Shalow Water Equaiton - First Step

The following boundary conditions are used:

$$p = 0 \qquad \text{on} \qquad z = \eta + H, \tag{47}$$

Defining free surface as:

$$S(x, y, z, t) = \eta(x, y, t) + H - z = 0. \tag{48}$$

Note that here we consider that particles on the surface remain on the surface. Take derivative on equation (48) at $z = \eta + H$, we can get:

$$\frac{DS}{Dt} = \frac{\partial \eta}{\partial t} + \tilde{u_x}\frac{\partial}{\partial x}(\eta + H) + \tilde{u_y}\frac{\partial}{\partial y}(\eta + H) - \tilde{u_z} = 0, \tag{49}$$

where $\tilde{u_x}$ means velocity evaluated at $z = \eta + H$.

Mathematical Model - Shalow Water Equaiton - First Step

Take derivative on equation (48) at $z = H(x, y)$, which means the solid bottom and is independent of time, the we can get:

$$\underline{u_x}\frac{\partial H}{\partial x} + \underline{u_y}\frac{\partial H}{\partial y} = \underline{u_z} \tag{50}$$

where $\underline{u_x}$ means velocity evaluated at the solid bottom.

Mathematical Model - Shalow Water Equaiton - Second Step

Integating (44) over the depth,

$$\int_H^{\eta+H} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) dz = 0. \tag{51}$$

Applied Leibnitz' theorem to it:

$$\frac{\partial}{\partial t} \int_{a(y,t)}^{b(y,t)} f(x,y,t)dx = \int_{a(y,t)}^{b(y,t)} \frac{\partial f}{\partial t} dx - f(a,y,t)\frac{\partial a}{\partial t} + f(b,y,t)\frac{\partial b}{\partial t}. \tag{52}$$

Then for x-direction term, we can get,

$$\int_H^{\eta+H} \frac{\partial u_x}{\partial x} dz = \frac{\partial}{\partial x} \int_H^{\eta+H} u_x dz - \tilde{u_x} \frac{\partial}{\partial x}(\eta + H) + \underline{u_x} \frac{\partial H}{\partial x}. \tag{53}$$

Similarly, y-component term can be derived as,

$$\int_H^{\eta+H} \frac{\partial u_y}{\partial y} dz = \frac{\partial}{\partial y} \int_H^{\eta+H} u_y dz - \tilde{u_y} \frac{\partial}{\partial y}(\eta + H) + \underline{u_y} \frac{\partial H}{\partial y}. \tag{54}$$

Mathematical Model - Shalow Water Equaiton - Second Step

Integating (44) over the depth,

$$\int_H^{\eta+H} \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \right) dz = 0. \tag{51}$$

Applied Leibnitz' theorem to it:

$$\frac{\partial}{\partial t} \int_{a(y,t)}^{b(y,t)} f(x,y,t)dx = \int_{a(y,t)}^{b(y,t)} \frac{\partial f}{\partial t} dx - f(a,y,t)\frac{\partial a}{\partial t} + f(b,y,t)\frac{\partial b}{\partial t}. \tag{52}$$

Then for x-direction term, we can get,

$$\int_H^{\eta+H} \frac{\partial u_x}{\partial x} dz = \frac{\partial}{\partial x} \int_H^{\eta+H} u_x dz - \tilde{u_x} \frac{\partial}{\partial x}(\eta + H) + \underline{u_x} \frac{\partial H}{\partial x}. \tag{53}$$

Similarly, y-component term can be derived as,

$$\int_H^{\eta+H} \frac{\partial u_y}{\partial y} dz = \frac{\partial}{\partial y} \int_H^{\eta+H} u_y dz - \tilde{u_y} \frac{\partial}{\partial y}(\eta + H) + \underline{u_y} \frac{\partial H}{\partial y}. \tag{54}$$

Mathematical Model - Shalow Water Equaiton - Second Step

Let

$$\bar{u_x} = \frac{1}{\eta}\int_H^{\eta+H} u_x dz \qquad \bar{u_y} = \frac{1}{\eta}\int_H^{\eta+H} u_y dz. \tag{55}$$

Pluging (55), (53), and (54) in (51)

$$\frac{\partial}{\partial x}(\bar{u_x}\eta) + \frac{\partial}{\partial y}(\bar{u_y}\eta) - \tilde{(u_x)}\frac{\partial}{\partial x}(\eta+H)$$
$$+\underline{u_x}\frac{\partial}{\partial x} - \tilde{u_y}\frac{\partial}{\partial y}(\eta+H) + \underline{u_y}\frac{\partial}{\partial y} + \tilde{u_z} - \underline{u_z} = 0. \tag{56}$$

Applying boundary condition (49) and (50)

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}(\bar{u_x}\eta) + \frac{\partial}{\partial y}(\bar{u_y}\eta) \tag{57}$$

Mathematical Model - Shalow Water Equaiton - Second Step

Let

$$\bar{u_x} = \frac{1}{\eta}\int_H^{\eta+H} u_x dz \qquad \bar{u_y} = \frac{1}{\eta}\int_H^{\eta+H} u_y dz. \tag{55}$$

Pluging (55), (53), and (54) in (51)

$$\begin{aligned} &\frac{\partial}{\partial x}(\bar{u_x}\eta) + \frac{\partial}{\partial y}(\bar{u_y}\eta) - \tilde{(u_x)}\frac{\partial}{\partial x}(\eta + H) \\ &+\underline{u_x}\frac{\partial}{\partial x} - \tilde{u_y}\frac{\partial}{\partial y}(\eta + H) + \underline{u_y}\frac{\partial}{\partial y} + \tilde{u_z} - \underline{u_z} = 0. \end{aligned} \tag{56}$$

Applying boundary condition (49) and (50)

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}(\bar{u_x}\eta) + \frac{\partial}{\partial y}(\bar{u_y}\eta) \tag{57}$$

Mathematical Model - Shalow Water Equaiton - Third Step

Integrating Momentum equation (46), by similar method, we can get for x-component,

$$\frac{\partial}{\partial t}(\eta\bar{u_x}) + \frac{\partial}{\partial x}(\beta_{xx}\eta\bar{u_x}^2) + \frac{\partial}{\partial y}(\beta_{xy}\eta\bar{u_x}\bar{u_y}) + g\eta\frac{\partial \eta}{\partial x} = g\eta S_{0x}. \tag{58}$$

For y-conponent,

$$\frac{\partial}{\partial t}(\eta\bar{u_y}) + \frac{\partial}{\partial y}(\beta_{yy}\eta\bar{u_y}^2) + \frac{\partial}{\partial x}(\beta_{yx}\eta\bar{u_x}\bar{u_y}) + g\eta\frac{\partial \eta}{\partial y} = g\eta S_{0y} \tag{59}$$

where S_{0x} and S_{0y} denote slopes of the bottom,

$$S_{0x} = \sin\theta_x = -\frac{\partial H}{\partial x} \qquad S_{0y} = \sin\theta_y = -\frac{\partial H}{\partial y}. \tag{60}$$

Mathematical Model - Shalow Water Equaiton - Third Step

β_{xy} and β_{xx} denote as:

$$\beta_{xy} = \frac{\int_H^{H+\eta} u_x u_y dz}{\bar{u_x}\bar{u_y}\eta} \qquad \beta_{xx} = \frac{\int_H^{H+\eta} u_x^2 dz}{\bar{u_x}^2\eta}. \tag{61}$$

Note that scale analysis is applied when deriving these equation.

Numerical Method

I take David L. Deorge's paper as reference for my numerical method, trying to solve shallow water system case.
Wave propagation algorithm is shown below:
For a system problem.

$$q_t + A(q,x)q_x = 0, \tag{62}$$

where $q_t \in \mathbb{R}$ and $A(q,x) \in \mathbb{R}^{m\times m}$. Note that $A(q,x) = A(q) = f^{'}(q)$. So, the first order scheme:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(A^+\Delta Q_{i-1/2}^n + A^-\Delta Q_{i+1/2}^n), \tag{63}$$

where $Q_i^{n+1} \approx \frac{1}{\Delta x}\int_{C_i} q(x,t^n)dx$, $C_i = [x_{i-1/2}, x_{i+1/2}]$, $\Delta x = (x_{i+1/2} - x_{i-1/2})$ and $\Delta t = (t^{n+1} - t^n)$.

Numerical Method

Note that $A^+\Delta Q^n_{i+1/2}$ are determined by solutions to Riemann problems ar the cell interfaces at $x_{i+1/2}$.
Second order scheme is shown below:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(A^+\Delta Q^n_{i-1/2} + A^-\Delta Q^n_{i+1/2}) - \frac{\Delta t}{\Delta x}(F^{n\tilde{}}_{i+1/2} - F^{n\tilde{}}_{i-1/2}), \qquad (64)$$

where $F^{n\tilde{}}_{i+1/2}$ can be determined by the waves in the Riemann problems at $x_{1/2}$.

Numerical Method

Furthermore, two-dimension problems can be interpreted as:

$$q_t + A(q,x,y)q_x + B(q,x,y)q_y = 0, \tag{65}$$

numerical scheme:

$$Q_{i,j}^{n+1} = ... \tag{66}$$

Semi-discretized scheme for one-dimension:

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}(A^+ \Delta Q_{i-1/2} + A^- \Delta Q_{i+1/2}) - \frac{1}{\Delta x}(\tilde{F}_{i+1/2} - \tilde{F}_{i-1/2}). \tag{67}$$

Now RK-method can be applied when we want higher-convergent order

Numerical Method

Furthermore, two-dimension problems can be interpreted as:

$$q_t + A(q,x,y)q_x + B(q,x,y)q_y = 0, \tag{65}$$

numerical scheme:

$$Q_{i,j}^{n+1} = ... \tag{66}$$

Semi-discretized scheme for one-dimension:

$$\frac{\partial Q_i}{\partial t} = -\frac{1}{\Delta x}(A^+ \Delta Q_{i-1/2} + A^- \Delta Q_{i+1/2}) - \frac{1}{\Delta x}(\tilde{F}_{i+1/2} - \tilde{F}_{i-1/2}). \tag{67}$$

Now RK-method can be applied when we want higher-convergent order

Numerical Method

Since this method is FV-based, high order FV method is tried to here. I think WENO and ENO type scheme might be help for this purpose. But need more study and discussion.

Challenge

I think my challenge is on transforming numerical scheme into efficient program. And how to create higher-order scheme. I already got some codes with two-phase model from teacher, and try to read them in order to help me design my program.

Scientific Significance

Not find one!

NSE with Free Surface

Free surface flow is the case where the size and shape of the solution region were part of the solution.

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

Primitive Equations in Diecast

The primitive equations of our ocean general circulation model are,

- Conservation of mass:

$$\nabla \cdot \mathbf{V} = \mathbf{0}, \tag{68}$$

- Horizontal momentun equations:

$$\frac{\partial u}{\partial t} = -\nabla \cdot u\mathbf{V} + \mathbf{fv} - \frac{\mathbf{1}}{\rho_\mathbf{0}} \frac{\partial \mathbf{p}}{\partial \mathbf{x}} + \nabla_\mathbf{h} \cdot \mathbf{A_h} \nabla_\mathbf{h} \mathbf{u} + \frac{\partial}{\partial \mathbf{z}} (\mathbf{A_v} \frac{\partial \mathbf{u}}{\partial \mathbf{z}}), \tag{69}$$

$$\frac{\partial v}{\partial t} = -\nabla \cdot v\mathbf{V} - \mathbf{fu} - \frac{\mathbf{1}}{\rho_\mathbf{0}} \frac{\partial \mathbf{p}}{\partial \mathbf{y}} + \nabla_\mathbf{h} \cdot \mathbf{A_h} \nabla_\mathbf{h} \mathbf{v} + \frac{\partial}{\partial \mathbf{z}} (\mathbf{A_v} \frac{\partial \mathbf{v}}{\partial \mathbf{z}}), \tag{70}$$

Primitive Equations in Diecast

- Conservation of scalar(salt or potential temperature)

$$\frac{\partial S}{\partial t} = -\nabla \cdot S\mathbf{V} + \nabla_{\mathbf{h}} \cdot \mathbf{K_h} \nabla_{\mathbf{h}} \mathbf{S} + \frac{\partial}{\partial \mathbf{z}} (\mathbf{K_v} \frac{\partial \mathbf{S}}{\partial \mathbf{z}}), \tag{71}$$

- Hydrostatic equation:

$$\frac{\partial p}{\partial z} = -(\rho - \bar{\rho})g, \tag{72}$$

- Equation of state:

$$\rho = \rho(S, T), \tag{73}$$

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.

Primitive Equations in Diecast

- Conservation of scalar(salt or potential temperature)

$$\frac{\partial S}{\partial t} = -\nabla \cdot S\mathbf{V} + \nabla_{\mathbf{h}} \cdot \mathbf{K_h} \nabla_{\mathbf{h}} \mathbf{S} + \frac{\partial}{\partial \mathbf{z}} (\mathbf{K_v} \frac{\partial \mathbf{S}}{\partial \mathbf{z}}), \tag{71}$$

- Hydrostatic equation:

$$\frac{\partial p}{\partial z} = -(\rho - \bar{\rho})g, \tag{72}$$

- Equation of state:

$$\rho = \rho(S, T), \tag{73}$$

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.

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 (72) 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 (72) 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 (72) 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 (72) 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 (72) 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 (72) 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:

$$\begin{aligned}\frac{\bar{u}^{n+1}_{i,j,k} + u^{n-1}_{i,j,k}}{\Delta t} &= -\Big(\left(\frac{\partial Uu}{\partial x}\right)^n_{i,j,k} + \left(\frac{\partial Vu}{\partial y}\right)^n_{i,j,k} \\ &+ \left(\frac{\partial Wu}{\partial z}\right)^n_{i,j,k}\Big) \\ &+ [2\Omega_e \sin(\phi_j) + u^n_{i,j,k} \tan(\frac{\phi_j}{r_e})]v^n_{i,j,k} \\ &- \left(\frac{\partial \bar{p}}{\partial x}\right)^n_{i,j,k} + dissipation, \end{aligned} \tag{74}$$

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:

$$\begin{aligned}\frac{\bar{u}^{n+1}_{i,j,k} + u^{n-1}_{i,j,k}}{\Delta t} &= -\Big(\left(\frac{\partial Uu}{\partial x}\right)^n_{i,j,k} + \left(\frac{\partial Vu}{\partial y}\right)^n_{i,j,k} \\ &+ \left(\frac{\partial Wu}{\partial z}\right)^n_{i,j,k}\Big) \\ &+ [2\Omega_e \sin(\phi_j) + u^n_{i,j,k} \tan(\frac{\phi_j}{r_e})]v^n_{i,j,k} \\ &- \left(\frac{\partial \bar{p}}{\partial x}\right)^n_{i,j,k} + dissipation, \end{aligned} \tag{74}$$

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 (74) 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 (74) 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 (74) 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 (74) 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 (74) 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:

$$U^{n+1} = \bar{U}^{n+1} + \Delta\bar{U} \tag{75}$$

$$V^{n+1} = \bar{V}^{n+1} + \Delta\bar{V}, \tag{76}$$

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 (68) we can get,

$$\int_0^D \left(\frac{\partial U^{n+1}}{\partial x} + \frac{\partial V^{n+1}}{\partial y}\right) dz = W^{n+1}(0) - W^{n+1}(D) = 0, \tag{77}$$

Fourth Step - Pressure Correction

Substitude (75) and (76) into (77), we can get:

$$\int_0^D (\frac{\partial \bar{U}^{n+1}}{\partial x} + \frac{\partial \bar{V}^{n+1}}{\partial y})dz = \int_0^D -(\frac{\partial \Delta\bar{U}}{\partial x} + \frac{\partial \Delta\bar{V}}{\partial y})dz. \tag{78}$$

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:

$$\int_0^D -(\frac{\partial^2 \Delta p}{\partial x^2} + \frac{\partial^2 \Delta p}{\partial y^2})dz = f(W), \tag{79}$$

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:

$$u^{n+1} = \bar{u}^{n+1} + \Delta\bar{u}, \tag{80}$$

$$v^{n+1} = \bar{v}^{n+1} + \Delta\bar{v}. \tag{81}$$

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:

$$u^{n+1} = \bar{u}^{n+1} + \Delta\bar{u}, \tag{80}$$

$$v^{n+1} = \bar{v}^{n+1} + \Delta\bar{v}. \tag{81}$$

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,

$$\bar{F(t)} = F(t) + 0.5\nu[\bar{F(t-1)} - 2F(t) + F(t+1)]. \tag{82}$$

Well-known centered-filter is,

$$\bar{F(t)} = F(t) + 0.5\nu[F(t-1) - 2F(t) + F(t+1)]. \tag{83}$$

Sixth Step - FLTW method

Following Kurihara(1965), consider the differentail model,

$$\frac{\partial F}{\partial t} = i\omega F. \tag{84}$$

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

$$\frac{F(t+1) - \bar{F(t-1)}}{2\Delta t} = i\omega_A F(t) + i(\omega - \omega_A)\frac{F(t+1) + \bar{F(t-1)}}{2\Delta t}. \tag{85}$$

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,

$$\frac{\partial F}{\partial t} = i\omega F. \tag{84}$$

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

$$\frac{F(t+1) - F(\bar{t-1})}{2\Delta t} = i\omega_A F(t) + i(\omega - \omega_A)\frac{F(t+1) + F(\bar{t-1})}{2\Delta t}. \tag{85}$$

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 (84), we can update our variables using FLTW method. Difference equation is,

$$Q^{n+1} = \frac{w}{2}(Q^n + Q^{n-2}) + (1-w)Q^{n-1}, \tag{86}$$

where Q is arbitary variable. FLTW reduces to FLT for $w = 1$ and to leap-frog for $w = 0$.
Rewrite (86),

$$Q^{n+1} = Q^{n-1} + \frac{w}{2}(Q^n - 2Q^{n-1} + Q^{n-2}). \tag{87}$$

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 (84), we can update our variables using FLTW method. Difference equation is,

$$Q^{n+1} = \frac{w}{2}(Q^n + Q^{n-2}) + (1-w)Q^{n-1}, \tag{86}$$

where Q is arbitary variable. FLTW reduces to FLT for $w = 1$ and to leap-frog for $w = 0$.
Rewrite (86),

$$Q^{n+1} = Q^{n-1} + \frac{w}{2}(Q^n - 2Q^{n-1} + Q^{n-2}). \tag{87}$$

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.

$$\frac{7}{16}\left(\left(\frac{\Delta p}{\Delta z}\right)_{i+1} + \left(\frac{\Delta p}{\Delta z}\right)_{i-1}\right) + \left(\frac{\Delta p}{\Delta z}\right)_i - \frac{h}{16}\left(\left(\frac{\Delta^2 p}{\Delta z^2}\right)_{i+1} - \left(\frac{\Delta^2 p}{\Delta z^2}\right)_{i-1}\right) = \frac{5}{16h}(p_{i+1} - p_{i-1}) \quad (88)$$

$$\frac{9}{8h}\left(\left(\frac{\Delta p}{\Delta z}\right)_{i+1} - \left(\frac{\Delta p}{\Delta z}\right)_{i-1}\right) + \left(\frac{\Delta p}{\Delta z}\right)_i - \frac{1}{8}\left(\left(\frac{\Delta^2 p}{\Delta z^2}\right)_{i+1} + \left(\frac{\Delta^2 p}{\Delta z^2}\right)_{i-1}\right) = \frac{3}{h^2}(p_{i+1} - 2p_i + p_{i-1}) + B.C. \quad (89)$$

$$\frac{\partial p}{\partial z} = -\rho g, \Longrightarrow \int_0^h \frac{\partial p}{\partial z} dz = -\int_0^h \rho g dz, \Longrightarrow p(h) - p(0) = -(\rho g) \times (h - 0). \quad (90)$$

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.

$$\frac{7}{16}\left(\left(\frac{\Delta p}{\Delta x}\right)_{i+1}+\left(\frac{\Delta p}{\Delta x}\right)_{i-1}\right)+\left(\frac{\Delta p}{\Delta x}\right)_{i}-\frac{h}{16}\left(\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i+1}-\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i-1}\right)$$
$$=\frac{5}{16h}(p_{i+1}-p_{i-1}) \quad (91)$$

$$\frac{9}{8h}\left(\left(\frac{\Delta p}{\Delta x}\right)_{i+1}-\left(\frac{\Delta p}{\Delta x}\right)_{i-1}\right)+\left(\frac{\Delta p}{\Delta x}\right)_{i}-\frac{1}{8}\left(\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i+1}+\left(\frac{\Delta^2 p}{\Delta x^2}\right)_{i-1}\right)$$
$$=\frac{3}{h^2}(p_{i+1}-2p_i+p_{i-1})+B.C. \quad (92)$$

$$p_i. \quad (93)$$

Shallow Water Equation

Here is a question we should keep in our mind:
'Are the existing solution algorithms for incompressible flow problems already optimal or os further, maybe even tremendous improvement necessary?'