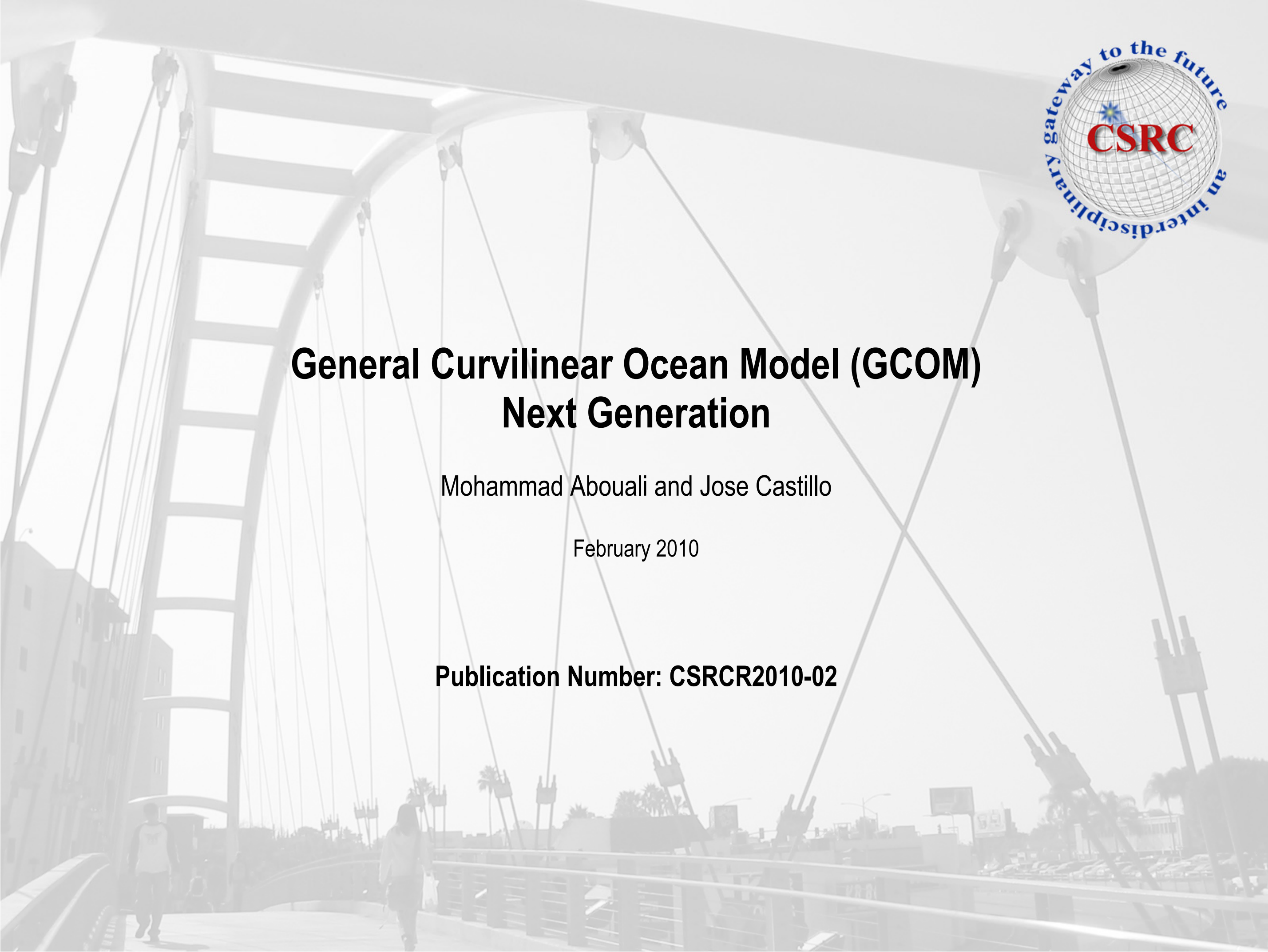

General Curvilinear Ocean Model (GCOM) Next Generation

Mohammad Abouali and Jose Castillo

February 2010

Publication Number: CSRCR2010-02

Computational Science &
Engineering Faculty and Students
Research Articles

Database Powered by the
Computational Science Research Center
Computing Group

COMPUTATIONAL SCIENCE
& ENGINEERING

SAN DIEGO STATE
UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

General Curvilinear Ocean Model (GCOM): Next Generation

Theory and Implementation, 2010

Abouali M., Castillo J.E.

gateway to the future
CSRC
an interdisciplinary

SAN DIEGO STATE
UNIVERSITY

Table of Contents:

List of Figures:

PART I: Theory and Governing Equations

1. Introduction

Nearly two thirds of the entire word population is living by a coast and their life is affected by their interactions with this coastal region. Fishing and other food supply, recreation and tourism, oil and Natural resources, or extracting renewable energy all happen in a coastal region; thus, the oceans have a huge effect on human life. The ocean is covering almost two thirds of the entire earth and by exchanging gas, energy, and aerosols with the atmosphere, affects our environment, climate, and weather. Hence, the ocean affects human activities even inland far from it. It is widely accepted that certain disturbances in the circulation of warm and cold water in the ocean have affected the bio-diversity of the earth previously, past ice age. Thus, studying the ocean and knowing how it inter-relates with human life is very important.

Using a numerical model is one of the methods used to study the oceans. Although human beings have been studying the oceans since they began trading overseas, advances in processing power, computer technology, and numerical methods have lead to more robust numerical models and more detail studies of the ocean both spatially and temporally.

Prof. Castillo and Prof. Torres started the General Curvilinear Ocean Model (GCOM) about 10 years ago at SDSU. The original model was written in FORTAN 77 and it included Momentum equations and a non-hydrostatic pressure equation (Torres, Mascarenhas and Castillo 2004) (Torres, Castillo and Mueller, et al. 2006) (Torres and Castillo, Stratified rotating flow over complex terrain 2003).

The “GCOM: Next Generation”, (GCOM-NG), a new code written in FORTRAN 90 standard, uses a staggered grid and includes the thermodynamic equations, i.e. conservation of salinity and temperature. The UNESCO Equation of State (EOS) was also added to calculate the density. Finally, the entire numerical scheme has been changed to make the code more accurate and faster. The following is the full description of the GCOM-NG and some test cases that demonstrate its capabilities.

2. Governing Equations

GCOM-NG is a non-dimensional model. Having the model in the non-dimensional form facilitates the comparison between different flow problems and helps the interpretation of the results (Geurts 2003). To non-dimensionalize the variables and parameters one has to choose a velocity scale, U^*, a length scale, L^*, salinity reference, S^*, a temperature reference, Θ^*, and a reference density, ρ^*. After that the non-dimensional variables are defined as:

Equation 1

$$x_i = \frac{x_i^*}{L^*},\quad u_i = \frac{u_i^*}{U^*},\quad t = \frac{t^*}{\left(L^*/U^*\right)},\quad p = \frac{p^*}{\rho^*\left(U^*\right)^2},\quad s = \frac{s^*}{S^*},\quad \theta = \frac{\theta^*}{\Theta^*}.$$

All the following equations are non-dimensional unless it is told otherwise.

2.1. Momentum Equation

GCOM-NG uses primitive Navier-Stokes equations with Boussinesq approximations. This means the changes in the density is neglected unless it is multiplied by the g-force (Stull 1988). The full non-dimensional momentum equation used in GCOM-NG is:

Equation 2

$$\underbrace{\frac{\partial u_i}{\partial t}}_{I} + \underbrace{u_j \frac{\partial u_i}{\partial x_j}}_{II} + \underbrace{\frac{\partial p}{\partial x_i}}_{III} - \underbrace{\frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j^2}}_{IV} - \underbrace{\frac{1}{Ro} \varepsilon_{ij3} u_j}_{V} + \underbrace{\frac{\rho'}{\rho^*} \frac{1}{Fr^2} \delta_{i3}}_{VI} = 0.$$

The description of each term is:

- **Term I:** Storage of the momentum,
- **Term II:** Advection,
- **Term III:** Pressure effect,
- **Term IV:** Diffusion,
- **Term V:** Coriolis force, due to the Earth rotation,
- **Term VI:** Buoyancy Effect.

There are a number of non-dimensional parameters used in the above equation. The first one is the Reynolds number, which is defined as:

Equation 3

$$\mathrm{Re} = \frac{U^* L^*}{\nu},$$

where ν is the kinematic viscosity defined as:

Equation 4

$$\nu = \frac{\mu^*}{\rho^*},$$

and μ^* is the viscosity. The next non-dimensional parameter, denoted as Ro, is called Rossby number. This Parameter controls the effect of the rotation of the Earth and is defined as:

Equation 5

$$Ro = \frac{U^*}{fL^*},$$

where we have:

Equation 6

$$f = 2\omega \sin(\varphi),$$

where $\omega = 0.725 \times 10^{-4} s^{-1}$ and φ is the latitude. It has to be noted that in the current version of GCOM-NG, the Rossby number is kept constant in the entire simulation domain; hence, one has to make sure that the entire simulation domain resides in a single f-plane. Also, note that the code is designed in such a way that if the user wants to ignore the rotation effect, this can be achieved by setting Rossby number to zero.

The Froud number is the last non-dimensional parameter used in the momentum equation. It is defined as:

Equation 7

$$Fr = \frac{U^*}{\sqrt{gL^*}},$$

where g is the g-force or the Earth acceleration constant. The remaining two symbols that are used in the momentum equation are defined as:

Equation 8

$$\varepsilon_{ijk} = \begin{cases} +1 & for \quad ijk = 123,231,312, \\ -1 & for \quad ijk = 321,213,132, \\ 0 & for \qquad other, \end{cases}$$

and

Equation 9

$$\delta_{i3} = \begin{cases} 1 & for \quad i = 3, \\ 0 & for \quad other. \end{cases}$$

In Equation 2, the buoyancy term is expanded as follow (Marshal and Plumb 2008) (Salmon 1998) (Stull 1988):

Equation 10

$$\rho' = \rho - \rho^*,$$

where ρ is the density of the parcel of the water, determined using the equation of state and ρ^* is the average of the density in a box surrounding the current cells with an envelope of two cells.

2.2. Conservation of Heat (Temperature Equation)

The first law of thermodynamics describes the conservation of enthalpy, from which we can derive the following equation for the temperature field or the conservation of heat equation:

Equation 11

$$\frac{\partial\theta}{\partial t} + u_j \frac{\partial\theta}{\partial x_j} = \frac{1}{\mathrm{Pr}_\theta \mathrm{Re}} \frac{\partial^2\theta}{\partial x_j^2} + Q_\theta,$$

where Q_θ is the non-dimensional source term. The relation between the dimensional and non-dimensional source term is:

Equation 12

$$Q_\theta = \frac{L^*}{U^*\Theta^*} Q_\theta^*.$$

The extra non-dimensional term in Equation 12 is called the Prandtl number, which is defined as the ratio of momentum viscosity to thermal viscosity:

Equation 13

$$\mathrm{Pr}_\theta = \frac{\nu}{\nu_\theta}.$$

The Prandtl number for 10°C seawater is usually about 10.

2.3. Conservation of Salt (Salinity Equation)

The equation for conservation of salt is the same as the equation for conservation of heat. The difference is the sink/source term, which is not discussed here. Using the same approach, as in conservation of heat, the non-dimensional equation for the conservation of salt can be written as:

Equation 14

$$\frac{\partial S}{\partial t}+u_j\frac{\partial S}{\partial x_j}=\frac{1}{Sc_S \operatorname{Re}}\frac{\partial^2 S}{\partial x_j^2}+Q_S.$$

The non-dimensional Schmidt Number is defined as:

Equation 15

$$Sc_S=\frac{\nu}{\nu_S}.$$

The Schmidt number is usually half the prandtl number in 10°C seawater.

2.4. Equation of State (EOS)

The equation of state (EOS) in the ocean is different than the equation of state for the atmosphere. The EOS for the ocean determines the density as a function of pressure, temperature, and salinity. The most commonly used one is (UNESCO 1980) (Millero, et al. 1980) (Millero and Poisson 1981) (McDougall and Jackett 2003) (Fofonoff and Millard 1983):

Equation 16

$$\rho^*\left(\theta^*,S^*,p^*\right)=\frac{\rho^*\left(\theta^*,S^*,0\right)}{1-\dfrac{p^*}{K\left(\theta^*,S^*,P^*\right)}},$$

where θ^* is the temperature in degree Celsius, S^* is the salinity in psu, and p^* is the pressure in bars. The error of Equation 16 is $3.5 \times 10^{-3} kgm^{-3}$ for a typical oceanic range as follow:

Equation 17

$$-2^\circ < \theta^* < 40\ ^\circ C,$$
$$0 < S^* < 40 \text{ psu},$$
$$0 < \text{p}^* < 1000 \text{ bars}.$$

The secant bulk modulus, and the density of seawater at one standard atmosphere pressure is defined as follow (Gill 1982) (Pond and Pickard 1989):

Equation 18

$$
\begin{array}{lll}
K\left(\theta^*,S^*,P^*\right) & = & \\
 & +19652.21 & \\
 & +148.4206\times\theta^* & -2.327105\times\theta^{*2} \\
 & +1.360477\times10^{-2}\theta^{*3} & -5.155288\times10^{-5}\theta^{*4} \\
 & +3.239908p^* & +1.43713\times10^{-3}\theta^*p^* \\
 & +1.16092\times10^{-4}\theta^{*2}p^* & -5.77905\times10^{-7}\theta^{*3}p^* \\
 & +8.50935\times10^{-5}p^{*2} & -6.12293\times10^{-6}\theta^*p^{*2} \\
 & +5.2787\times10^{-8}\theta^{*2}p^{*2} & \\
 & +54.6746\times S^* & -0.603459\times\theta^*S^* \\
 & +1.09987\times10^{-2}\theta^{*2}S^* & -6.1670\times10^{-5}\theta^{*3}S^* \\
 & +7.944\times10^{-2}S^{*1.5} & +1.6483\times10^{-2}\theta^*S^{*1.5} \\
 & -5.3009\times10^{-4}\theta^{*2}S^{*1.5} & +2.2838\times10^{-3}p^*S^* \\
 & -1.0981\times10^{-5}\theta^*p^*S^* & -1.6078\times10^{-6}\theta^{*2}p^*S^* \\
 & +1.91075\times10^{-4}p^*S^{*1.5} & -9.9348\times10^{-7}p^{*2}S^* \\
 & +2.0816\times10^{-8}\theta^*p^{*2}S^* & +9.1697\times10^{-10}\theta^{*2}p^{*2}S^*,
\end{array}
$$

Equation 19

$$
\begin{array}{lll}
\rho^*\left(\theta^*,S^*,0\right) & = & \\
 & +999.842594 & +6.793952\times10^{-2}\theta^* \\
 & -9.095290\times10^{-3}\theta^{*2} & +1.001685\times10^{-4}\theta^{*3} \\
 & -1.120083\times10^{-6}\theta^{*4} & +6.536332\times10^{-9}\theta^{*5} \\
 & +8.24493\times10^{-1}S^* & -4.0899\times10^{-3}\theta^*S^* \\
 & +7.64380\times10^{-5}\theta^{*2}S^* & -8.2467\times10^{-7}\theta^{*3}S^* \\
 & +5.38750\times10^{-9}\theta^{*4}S^* & -5.72466\times10^{-3}S^{*1.5} \\
 & +1.02270\times10^{-4}\theta^*S^{*1.5} & -1.6546\times10^{-6}\theta^{*2}S^{*1.5} \\
 & +4.83140\times10^{-4}S^{*2}. &
\end{array}
$$

By including the Equation of the State in the code, temperature and salinity are now active scalars. Changes in temperature and salinity will result in changes in density, and the changes in density affect the momentum equation via the buoyancy term; therefore, changes of temperature and salinity will affect the flow field.

2.5. Pressure Equation

There are different methods to calculate the pressure (Ferziger and Peric 2001) (Langtangen, Mardal and Winther 2002) (Harlow and Welch 1965). By applying the divergence on the momentum equation, the non-hydrostatic equation for the pressure can be achieved. Furthermore, by applying the continuity equation it can be simplified even more. The final result would be a Poisson equation for the pressure as follow:

Equation 20

$$\nabla^2 p = \frac{\nabla\cdot\,\vec{u}^{\,n}}{\delta t} - \nabla\cdot\left[\left(\vec{u}^{\,n}\cdot\nabla\right)\vec{u}^{\,n}\right] + \nabla\cdot\left[\frac{1}{Ro}\left(v^n\hat{i} - u^n\hat{j}\right)\right] - \frac{1}{\tilde{Fr}^{2}},$$

where the superscript *n* means the current time, and *δt* is the non-dimensional time step. In this equation we have:

Equation 21

$$\tilde{Fr} = \frac{U^*}{NL^*},$$

and N is Brunt-Väisälä

Equation 22

$$N^2 = -\frac{g}{\rho^*}\frac{\partial\rho}{\partial z^*}.$$

while later in the text, this method of solving pressure equation is changed to a more accurate and faster one.

3. Curvilinear Coordinates

The GCOM-NG is written in curvilinear coordinates, also known as boundary fitted coordinates. The governing equations, which are originally written in Cartesian coordinate, are transformed into generalized coordinates. Therefore, the physical space, (x,y,z), is mapped into a uniformly spaced cube, (ξ,η,ζ), Figure 1, which facilitates the implementation of the boundary conditions. Using the chain-rule of derivation we have (Hoffmann and Chiang 2000):

Equation 23

$$
\begin{aligned}
\frac{\partial(\;)}{\partial x} &= \xi_x \frac{\partial(\;)}{\partial \xi} + \eta_x \frac{\partial(\;)}{\partial \eta} + \zeta_x \frac{\partial(\;)}{\partial \zeta},\\
\frac{\partial(\;)}{\partial y} &= \xi_y \frac{\partial(\;)}{\partial \xi} + \eta_y \frac{\partial(\;)}{\partial \eta} + \zeta_y \frac{\partial(\;)}{\partial \zeta},\\
\frac{\partial(\;)}{\partial z} &= \xi_z \frac{\partial(\;)}{\partial \xi} + \eta_z \frac{\partial(\;)}{\partial \eta} + \zeta_z \frac{\partial(\;)}{\partial \zeta},
\end{aligned}
$$

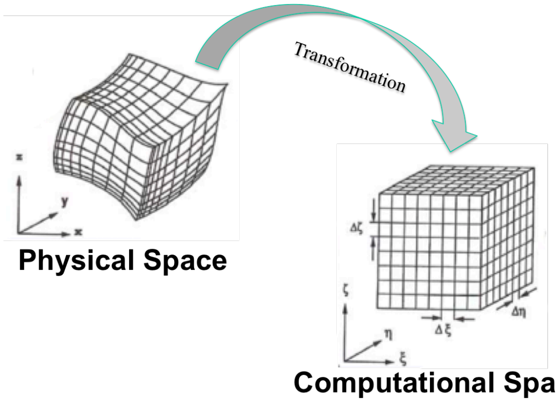

Figure 1: Transforming Physical Space to uniformly spaced Cube

where we have:

Equation 24

$$
\begin{aligned}
\xi_x &= J\left(y_\eta z_\zeta - y_\zeta z_\eta\right), & \eta_x &= J\left(y_\zeta z_\xi - y_\xi z_\zeta\right), & \zeta_x &= J\left(y_\xi z_\eta - y_\eta z_\xi\right),\\
\xi_y &= J\left(x_\zeta z_\eta - x_\eta z_\zeta\right), & \eta_y &= J\left(x_\xi z_\zeta - x_\zeta z_\xi\right), & \zeta_y &= J\left(x_\eta z_\xi - x_\xi z_\eta\right),\\
\xi_z &= J\left(x_\eta y_\zeta - x_\zeta y_\eta\right), & \eta_z &= J\left(x_\zeta y_\xi - x_\xi y_\zeta\right), & \zeta_z &= J\left(x_\xi y_\eta - x_\eta y_\xi\right),
\end{aligned}
$$

which are called metrics of the transformation, and *J* is the Jacobian of the transformation and is defined as:

Equation 25

$$
J = \frac{1}{x_\xi\left(y_\eta z_\zeta - y_\zeta z_\eta\right) - x_\eta\left(y_\xi z_\zeta - y_\zeta z_\xi\right) + x_\zeta\left(y_\xi z_\eta - y_\eta z_\xi\right)}.
$$

Using Equation 23, defining the transformed gradient and divergence operator is rather easy. However, the transformed Laplacian operator can be written as (Torres & Castillo, 2003):

Equation 26

$$
\begin{aligned}
\nabla^2(\ ) = \quad L(\ ) \quad & -L(x)\left[\xi_x \frac{\partial(\ )}{\partial \xi} + \eta_x \frac{\partial(\ )}{\partial \eta} + \zeta_x \frac{\partial(\ )}{\partial \zeta}\right] \\
& -L(y)\left[\xi_y \frac{\partial(\ )}{\partial \xi} + \eta_y \frac{\partial(\ )}{\partial \eta} + \zeta_y \frac{\partial(\ )}{\partial \zeta}\right] \\
& -L(z)\left[\xi_z \frac{\partial(\ )}{\partial \xi} + \eta_z \frac{\partial(\ )}{\partial \eta} + \zeta_z \frac{\partial(\ )}{\partial \zeta}\right],
\end{aligned}
$$

where

Equation 27

$$
\begin{aligned}
L(\ ) = \quad & a\frac{\partial^2(\ )}{\partial \xi^2} + b\frac{\partial^2(\ )}{\partial \eta^2} + c\frac{\partial^2(\ )}{\partial \zeta^2} \\
& +2\left[d\frac{\partial^2(\ )}{\partial \xi \partial \eta} + e\frac{\partial^2(\ )}{\partial \zeta \partial \eta} + q\frac{\partial^2(\ )}{\partial \xi \partial \zeta}\right],
\end{aligned}
$$

and

Equation 28

$$
\begin{aligned}
a &= \xi_x^2 + \xi_y^2 + \xi_z^2, \\
b &= \eta_x^2 + \eta_y^2 + \eta_z^2, \\
c &= \zeta_x^2 + \zeta_y^2 + \zeta_z^2, \\
d &= \xi_x \eta_x + \xi_y \eta_y + \xi_z \eta_z, \\
e &= \zeta_x \eta_x + \zeta_y \eta_y + \zeta_z \eta_z, \\
q &= \xi_x \zeta_x + \xi_y \zeta_y + \xi_z \zeta_z.
\end{aligned}
$$

It is worth to notice that just the coefficients of the equations are transformed and not the variables. Thus, the variables do not need to be transformed back.

4. Numerics

Finite difference was used to solve the above equations. The Forward explicit time stepping was used for the time discretization. Although the implicit methods give a better stability, unconditionally stable, the explicit method was used here for solving three components of the velocity. The reason for this choice is that implicit methods will end up in solving a system of linear equations, which is costly in huge problems; thus, not really suitable for practical problems. To solve the Poisson equation for pressure the Successive Over Relaxation (SOR) iterative method is used.

Central finite difference, second order, was used for spatial discretization. But the Navier-Stokes equation has a non-linear part, $\vec{u}\cdot\nabla\vec{u}$. It is well known that the sided discretization work better for this term, i.e. forward or backward depending on the sign of the velocity. For this non-linear term the forth-order scheme (Kawamura, Takami and Kuwahara 1986), defined as follow, was used:

Equation 29

$$\left(f_\xi \frac{\partial u}{\partial \xi}\right)_{i,j,k} = f_{\xi_{i,j,k}} \frac{-u_{i+2,j,k} + 8\left(u_{i+1,j,k} - u_{i-1,j,k}\right) + u_{i-2,j,k}}{12\delta\xi} + \left|f_{\xi_{i,j,k}}\right| \frac{u_{i+2,j,k} - 4u_{i+1,j,k} + 6u_{i,j,k} - 4u_{i-1,j,k} + u_{i-2,j,k}}{4\delta\xi},$$

where

Equation 30

$$f_\xi = u\xi_x + v\xi_y + w\xi_z.$$

At first glance, Equation 29 looks like a central finite difference. But it is actually one forward discretization combined with a backward forward discretization (remember this depends on the sign of the velocity) using the following relation:

Equation 31

$$\frac{f+|f|}{2} = \begin{cases} f & for \quad f \geq 0, \\ 0 & for \quad f \leq 0, \end{cases} \quad \frac{f-|f|}{2} = \begin{cases} 0 & for \quad f \geq 0, \\ f & for \quad f \leq 0. \end{cases}$$

by changing *i* to *j* or *k*, one can easily write the forth order expression for $f_\eta \frac{\partial u}{\partial \eta}$ and $f_\zeta \frac{\partial u}{\partial \zeta}$, where the coefficients are:

Equation 32

$$f_\eta = u\eta_x + v\eta_y + w\eta_z,$$
$$f_\zeta = u\zeta_x + v\zeta_y + w\zeta_z.$$

5. Test Cases:

5.1. Lid Driven Cavity:

In this case, a 3D lid driven cavity is studied. For the geometry of this case please refer to Figure 2. Except the top face (lid) all faces are no-slip walls.

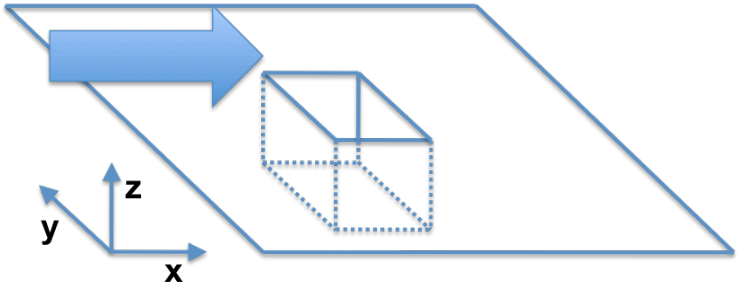

Figure 2: Geometry of Case I.

Here, the Reynolds number is equal to 1000 and the simulation was done for 10 non-dimensional time units. The iso-speed contours at the end of the simulation in a vertical plain passing from the middle of the box and close to the wall are shown in Figure 3 and Figure 4. The iso-speed contours at the end of the simulation in a horizontal plain passing close to the bottom of the box, middle of the box, and top of the box is shown in Figure 5, Figure 6, and Figure 7, respectively. In Figure 8 both iso-speed contours and velocity vectors can be seen.

5.2. Seamount Case I:

In this case, a seamount is generated based on the following formula:

Equation 33

$$z_{i,j,k} = L_z\left(1 - \frac{a}{1 + r^4 * L_0}\right) f_k,$$

where in this case $a=0.5$ and $L_0=40$ and f_k is a scaling function that causes more grid points to be distributed close to the bottom of the domain. In this example, the non-dimensional size of the domain is 4×2×1 and 96×32×32 grid points are used along the x, y, and z directions respectively, Figure 9, Figure 10. The boundary condition is shown in Figure 11. The result is shown in Figure 12.

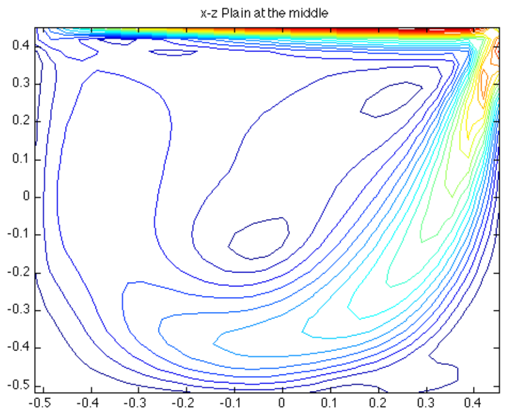

Figure 3: x-z plain (vertical) in the middle of the box.

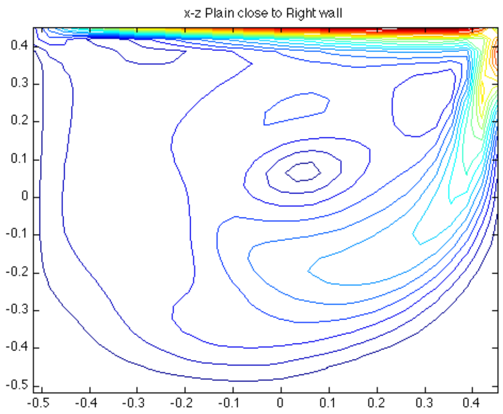

Figure 4: x-z plain (vertical) close to the wall.

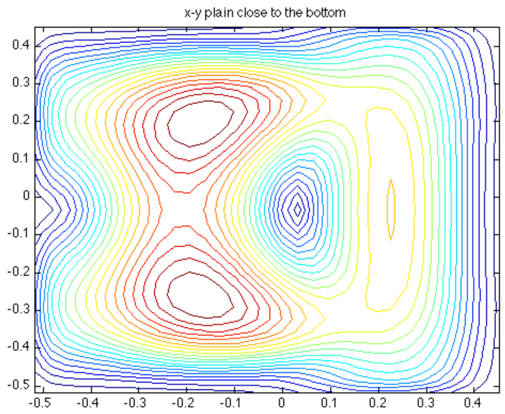

Figure 5: x-y plain (horizontal) close to the bottom of the box.

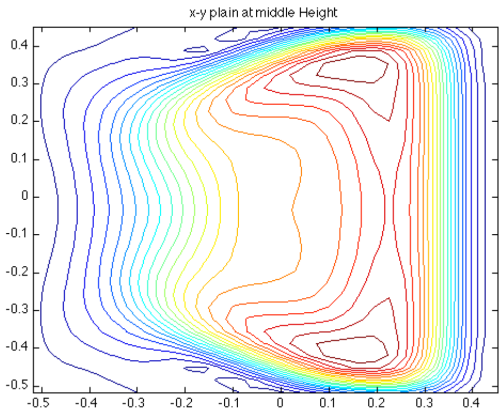

Figure 6: x-y plain (horizontal) in the middle of the box.

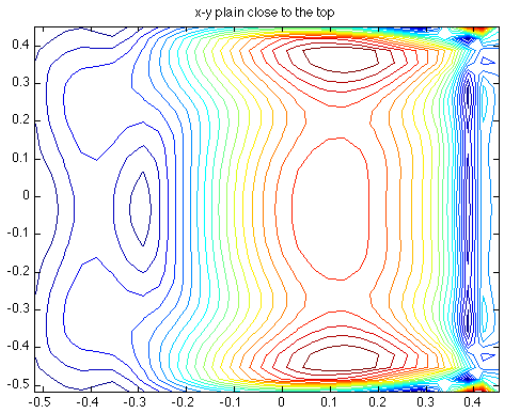

Figure 7: x-y plain (horizontal) close to the top of the box.

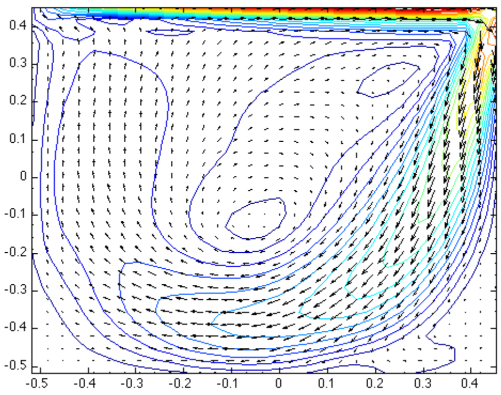

Figure 8: Iso speed contour lines and velocity vectors

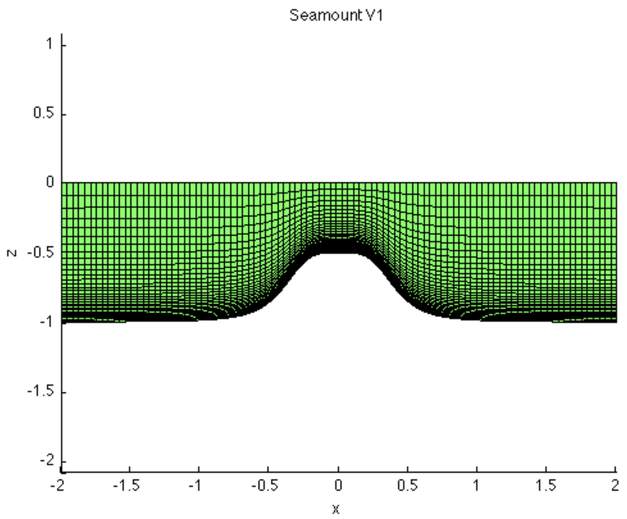

Figure 9: Seamount Case I

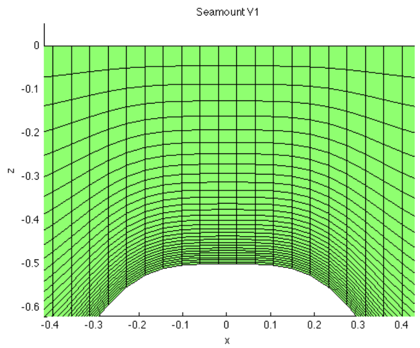

Figure 10: Seamount Case I Zoomed.

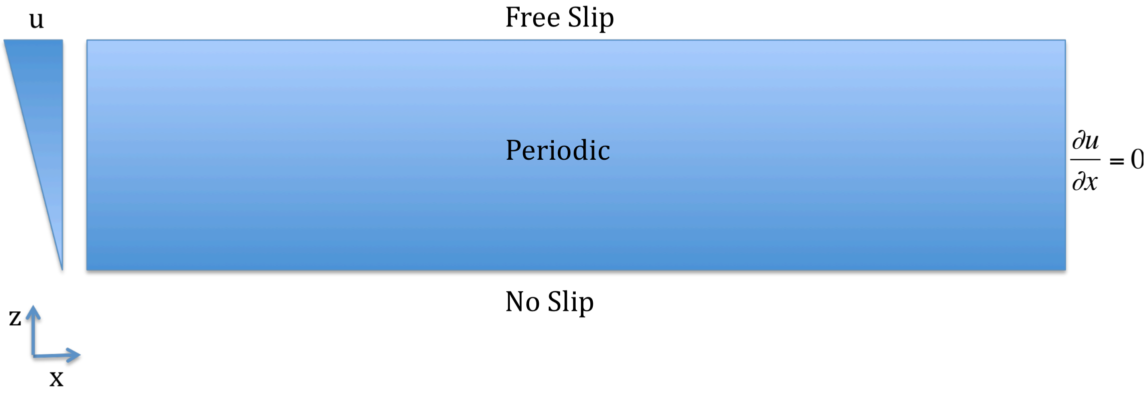

Figure 11: Seamount Boundary Condition.

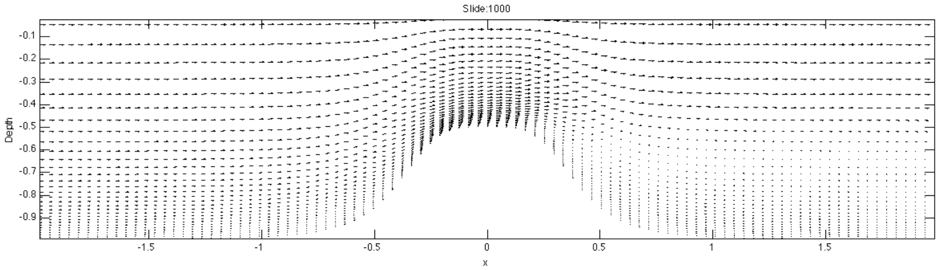

Figure 12: Result in Seamount Case I.

5.3. Seamount Case II:

In this example, everything is kept the same as in the Seamount Case I. The only difference is the function that generates the bathymetry of the seamount. The function for the seamount Case II is defined as:

Equation 34:

$$z_{i,j,k} = L_z\left(1 - ae^{-sL_0}\right)f_k .$$

Again *a=0.5* but *L₀=10*. Refer to Figure 13 and Figure 14 for the geometry of the seamount. The result is shown in Figure 15.

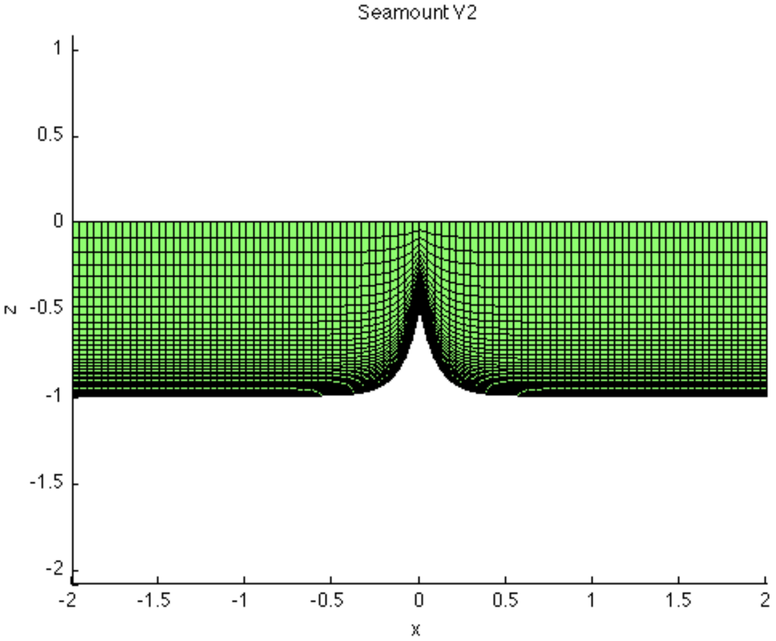

Figure 13: Seamount Case II.

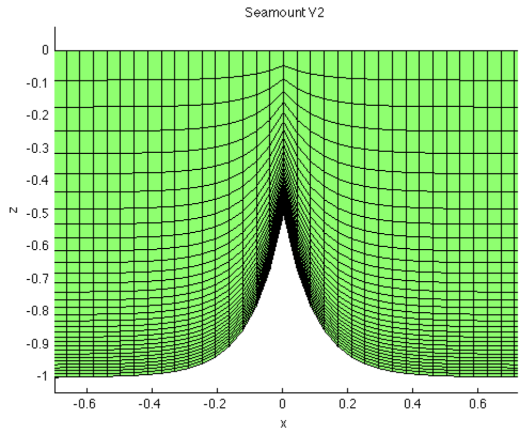

Figure 14: Seamount Case II Zoomed.

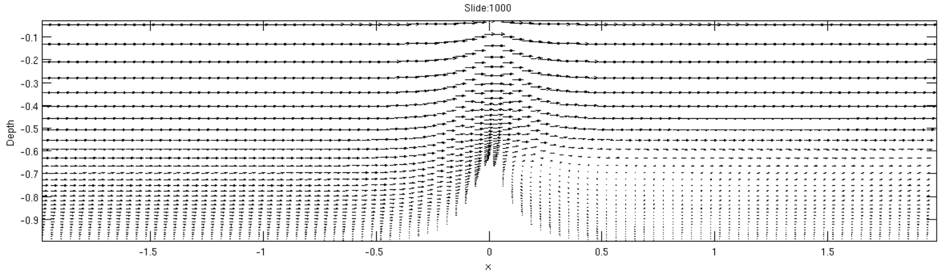

Figure 15: Result in Seamount Case II.

5.4. Energy of the system:

One of the ways to look at the results of a model is by plotting its energy and maximum speed. There are two things to notice in these graphs. (1) If there is any oscillations in the graph? (2) If the values are reaching a stable level?

The less oscillations in the graph the more desirable the results are. And if you are seeing any oscillations, those have to be bounded or even better decaying gradually. As the forcing is not changed during the entire simulation, at some point the system must reach a stable position. And this means that your graph should reflect this.

Here we have calculated the Volume averaged Kinetic Energy and Barotropic Energy as follow:

Equation 35

$$\varepsilon_{kin} = \frac{1}{2\upsilon}\sum\left(\Delta\upsilon_{ui,j,k}u_{i,j,k}^{2} + \Delta\upsilon_{vi,j,k}v_{i,j,k}^{2}\right),$$

Equation 36

$$\varepsilon_{bar} = \frac{1}{2\upsilon}\sum\left(\Delta\overline{\upsilon}_{ui,j}\overline{u}_{i,j}^{2} + \Delta\overline{\upsilon}_{vi,j}\overline{v}_{i,j}^{2}\right),$$

where we have:

Equation 37

$$\overline{u}_{i,j,k} = \frac{1}{\Delta\overline{\upsilon}_{ui,j}}\sum_{k}\Delta\upsilon_{ui,j,k}u_{i,j,k},$$

$$\overline{v}_{i,j,k} = \frac{1}{\Delta\overline{\upsilon}_{vi,j}}\sum_{k}\Delta\upsilon_{vi,j,k}v_{i,j,k},$$

Equation 38

$$\overline{\upsilon}_{ui,j} = \sum_{k}\Delta\upsilon_{ui,j,k},$$

$$\overline{\upsilon}_{vi,j} = \sum_{k}\Delta\upsilon_{vi,j,k}.$$

The graph for “Seamount Case I” is shown in Figure 16, Figure 17, Figure 18, and Figure 19.

5.5. Buoyancy Driven Flow:

In this case, we have assumed that the flow is in a box with no-slip wall at the sides except the top lid, Figure 2. The entire fluid is set at rest at time zero and no forcing is done on the fluid motion. The temperature of the fluid is set to zero (non-dimensional) all over, except for a smaller box in the middle of the domain, where the temperature is set to one (non-dimensional). The Temperature is allowed to escape from the top lid.

This hot bucket of water has the lower density according to the equation of state. Thus, this bucket experiences some buoyancy forces and rises to the surface and finally escapes from there. For the simulation results refer to Figure 20 and Figure 21.

6. Fractional Step:

Previously, the divergence operator was applied on the momentum equation to derive the Equation 20 for the pressure. It can be seen that the dilation term, i.e. $\nabla^{2}(\nabla\cdot\vec{u})$, is not present there. This is due to the continuity equation, which says that this term should be zero.

Although the physics says that this term should be zero, it is wise to keep this term in the pressure equation, as it gives better numerical stability and more accurate results. Implementing this term is difficult and implementing it in the curvilinear coordinates is even much more difficult. On the other hand, including this term will result in a more accurate and more stable model.

The alternate solution to this approach is to make use of a fractional step method (Kim and Moin 1985). In this method the velocity is calculated first by:

Equation 39

$$\frac{u_i' - u_i^n}{\delta t} = -u_j \frac{\partial u_i}{\partial x_j} + \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j^2} + \frac{1}{Ro} \varepsilon_{ij3} u_j - \frac{\rho'}{\rho} \frac{1}{Fr^2} \delta_{i3}.$$

It can be noted that this equation has all the terms as the momentum equation except the pressure effect. Thus, after evaluating this equation an intermediate value for the velocity is achieved. Still, the velocity field is needed to be updated by adding the pressure effect to obtain the final value for the velocity, or:

Equation 40

$$\frac{u_i^{n+1} - u_i'}{\delta t} = -\frac{\partial p}{\partial x_i}.$$

Still an equation for the pressure is needed. This equation can be achieved by applying the divergence to the Equation 41 and using the continuity equation, which gives:

Equation 41

$$\nabla^2 p = \frac{1}{\delta t} \nabla \cdot u'.$$

This equation for pressure is much easier to evaluate and the effect of the dilation term is also indirectly included. Thus, it is faster to evaluate, it is more accurate and it conserves better the continuity of mass. These equations were implemented in the code and the previous case was tested, Figure 22. It can be seen that the stable condition is achieved much faster.

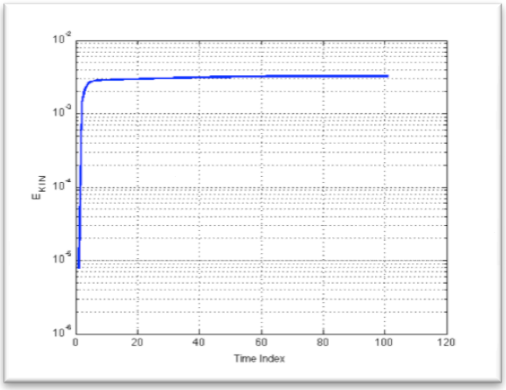

Figure 16: Volume Averaged Kinetic Energy.

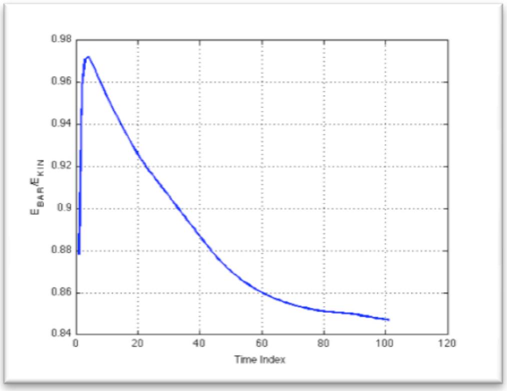

Figure 17: Fraction of Barotropic to Kinetic Energy.

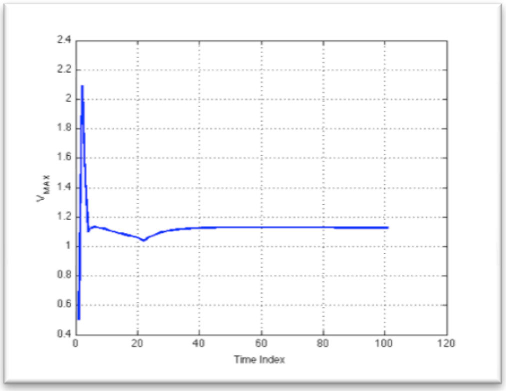

Figure 18: Maximum Speed.

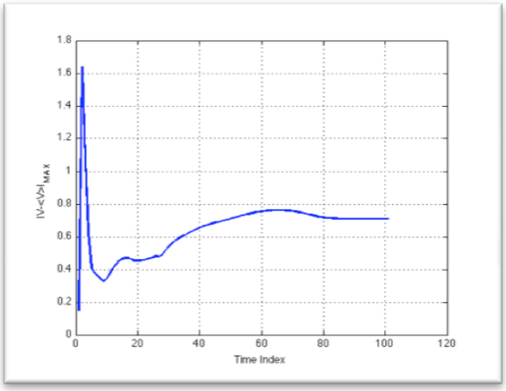

Figure 19: Maximum Differences in Speed.

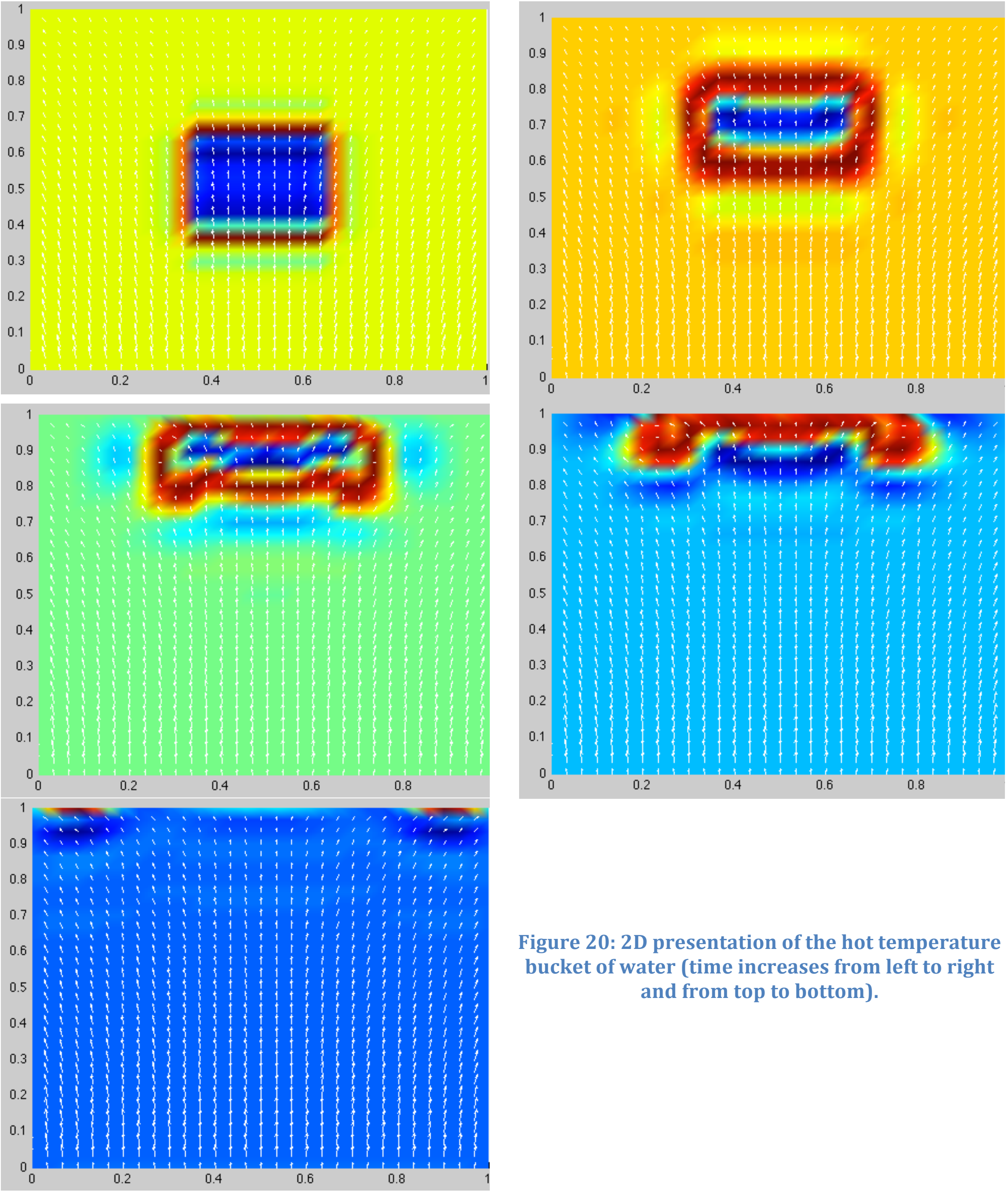

Figure 20: 2D presentation of the hot temperature bucket of water (time increases from left to right and from top to bottom).

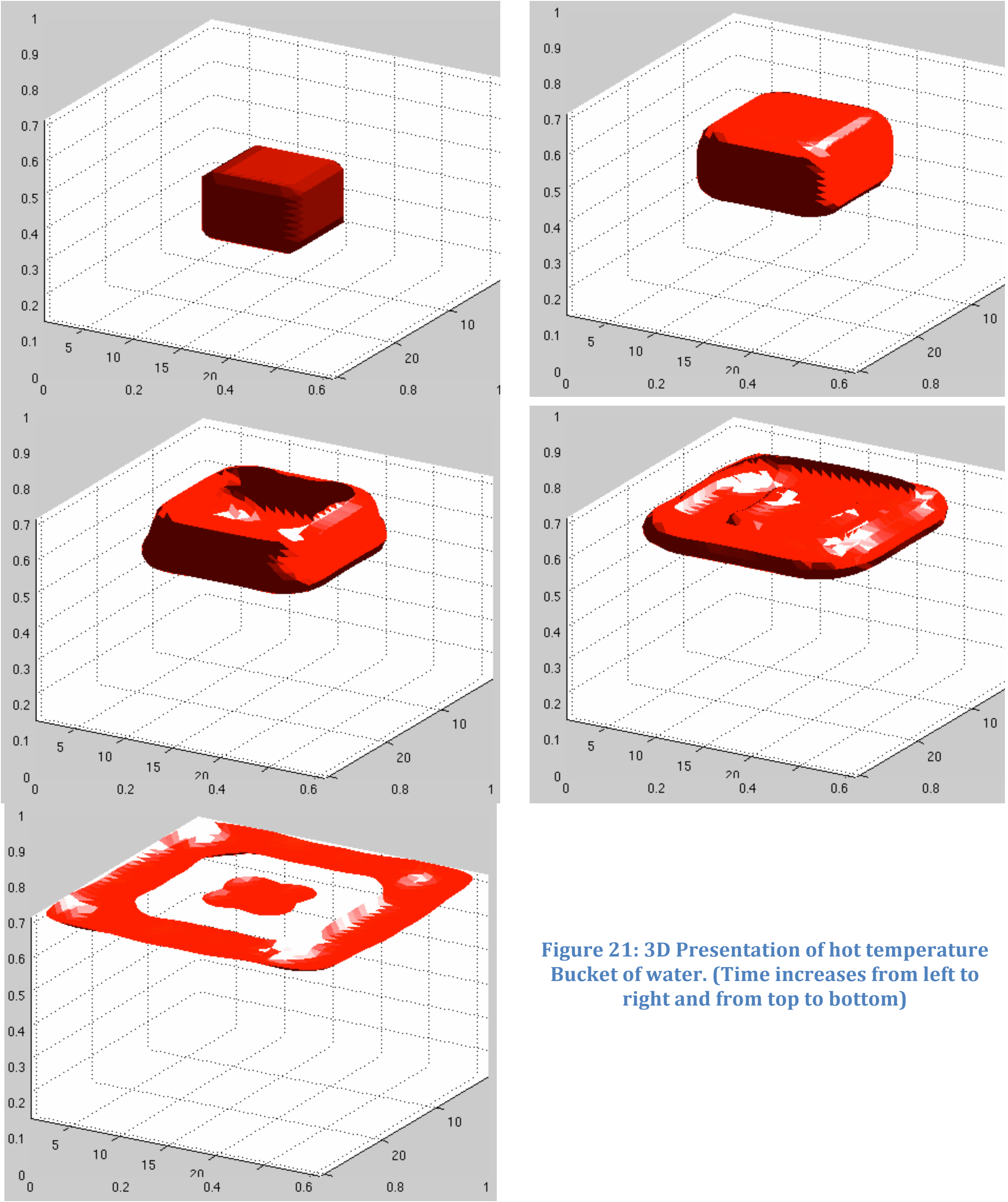

Figure 21: 3D Presentation of hot temperature Bucket of water. (Time increases from left to right and from top to bottom)

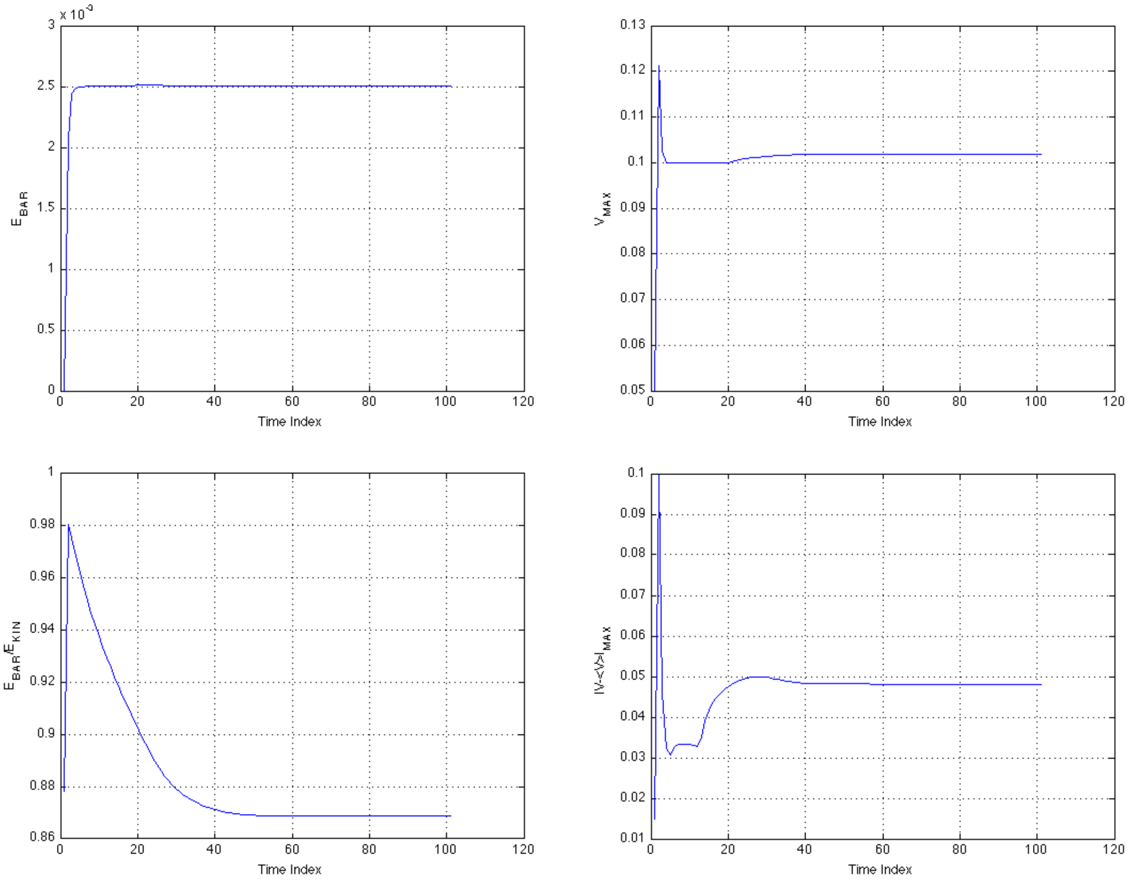

Figure 22: Results of Fractional Step.

PART II: Program Details and Operating the Code:

7. List of Files:

GCOM-NG includes the following main files:

- **Main.F90:** This is the driver of the entire code. Different parts of the code are called from here.
- **ReadParam.F90:** Reads the parameters from ./PARAM/param.dat.
- **ReadGrid.F90:** Reads the grid information from ./GRID/Grid.dat
- **CalcMetrics.F90 & CalcDerivatice.F90:** These two files calculate the required metrics for coordinate transformation to general curvilinear coordinate system.
- **WriteToFile.F90:** This file writes out all the fields, such as velocity, pressure, and density.
- **SolveUExplicit_S1.F90:** This file solves for the U component of the velocity.
- **SolveVExplicit_S1.F90:** This file solves for the V component of the velocity.
- **SolveWExplicit_S1.F90:** This file solves for the W component of the velocity.
- **SolveP_Fractional.F90:** This file solves for the pressure, using the fractional step method.
- **CorrectU4Pressure.F90:** This file updates the U component of the velocity for the updated pressure.
- **CorrectV4Pressure.F90:** This file updates the V component of the velocity for the updated pressure.
- **CorrectW4Pressure.F90:** This file updates the W component of the velocity for the updated pressure.
- **SolveTExplicit.F90:** This file solves for Temperature field.
- **SolveSExplicit.F90:** This file solves for Salinity field.
- **EOS.F90:** This file calculates the density, using UNESCO equation of state.

The above files are the main files of the GCOM-NG model and they do not need to be changed. Each file contains a FORTRAN 90 subroutine of the same name as the filename. The Main.F90 calls these subroutines. In addition to the above-mentioned files, the user has to provide the following files and subroutines:

- **InitCond.F90:** This subroutine initializes all the fields.
- **bcondDens.F90:** A subroutine to set or calculate the boundary conditions for density.
- **bcondP.F90:** A subroutine to set or calculate the boundary conditions for pressure.
- **bcondU.F90:** A subroutine to set or calculate the boundary conditions for U component of the velocity.

- **bcondV.F90:** A subroutine to set or calculate the boundary conditions for V component of the velocity.
- **bcondW.F90:** A subroutine to set or calculate the boundary conditions for W component of the velocity.
- **bcondS.F90:** A subroutine to set or calculate the boundary conditions for Salinity.
- **bcondT.F90:** A subroutine to set or calculate the boundary conditions for Temperature.

A sample file is provided in the source directory of the GCOM-NG so that the user can learn how these files are needed to be prepared. The user can edit the sample files and tune the code for the particular boundary condition and the initial values of his/her choice.

8. Compiling:

A make file is provided in the main directory of the GCOM-NG. Under Linux or Mac OSX system, the user need to type “make” to compile the entire code. It has been assumed that the GFORTRAN compiler is already installed. By default, GCOM-NG is compiled by GFORTRAN. But the user can try the compiler of his/her choice. GCOM-NG has been tested with iFort compiler.

Under Microsoft Windows machine, the user is advised to refer to the manual of the compiler that is installed on the system.

9. Parameters:

There are certain parameters required by the model, before it can successfully run. Of course, changing these parameters does not require recompiling the code. They are written in ‘./PARAM/param.dat’ and read by ReadParam.F90. These parameters are:

- **dt:** Time step.
- **Max File Number:** The maximum number of snap-shots of all of the variables that the user is deciding to store in a file for post processing.
- **Write out Frequency:** The number of time steps between each snap-shots; thus, the total time of simulation is:

Equation 42

$$\text{Write out Frequency} \times \text{Max File Number} \times \text{dt}$$

- **Omega:** This parameter is required for SOR solver for the pressure. Refer to any description on SOR method to learn how to adjust this parameter. Default value is 1.
- **Epsilon:** The maximum error desired when solving the Poisson's equation for pressure. Default value is 10-7.
- **SOR maximum number of iterations:** The maximum number of iterations allowed in SOR solver. The SOR solver will exit if this number is reached, even if the desired maximum error is not achieved.
- **Reynolds Number**
- **PrT:** Prandtl number for Temperature.
- **PrS:** Schmidt number for salinity.
- **Rossby Number**
- **Froud Number**
- **Velocity Scale:** The velocity scale used in non-dimensionalizing the velocity field.
- **Length Scale:** The length scale used in non-dimensionalizing the grid.
- **Temperature Scale:** The temperature scale used in non-dimensionalizing the temperature field.
- **Salinity Scale:** The salinity scale used in non-dimensionalizing the salinity field.

By default a param.dat file, which must be modified by the user, is provided in the model. The parameters must be given in the same order as in the provided file. Numbers must start from character 17 of each line and they must be given in a floating-point format, so the file is a fixed format. So even 1 must be typed 1.0, otherwise the program will fail to read the parameters properly.

10. Variables and Include Files:

There are a number of include files in './src/includes'. These include files are:

- **Grid.inc:** defines the grid variables.
- **Metrics.inc:** degines the metrics of the transformation.
- **ModelParam.inc:** defines the parameters described in above section.
- **ProbSize:** defines the dimension of the problem.
- **Scalars.inc:** defines all the scalar variables, such as pressure.
- **Velocity.inc:** defines all components of the velocity, i.e. u, v, w.

None of the above include files are needed to be changed except the ProbSize.inc. There are six variables that can be changed in this include file, as follow:

- **IMax:** Number of grid points in "I" direction.
- **JMax:** Number of grid points in "J" direction.
- **KMax:** number of grid points in "K" direction.

- **Idi:** The inverse of the spacing along the "I" direction in curvilinear coordinate. Default value is 1.
- **Idj:** The inverse of the spacing along the "J" direction in curvilinear coordinate. Default value is 1.
- **Idk:** The inverse of the spacing along the "K" direction in curvilinear coordinate. Default value is 1.

If "Idi", Idj", and "Idk" are set to 1, then the grid is transformed into a uniformly spaced cube, where all the spacing are equal to 1. It is recommended not to change this number. But sometimes changing this number may lead to lower numerical errors due to finite precision available in computers. The full list of variables and their sizes is given in Table 1. For the location of the variables and their indices refer to Figure 23.

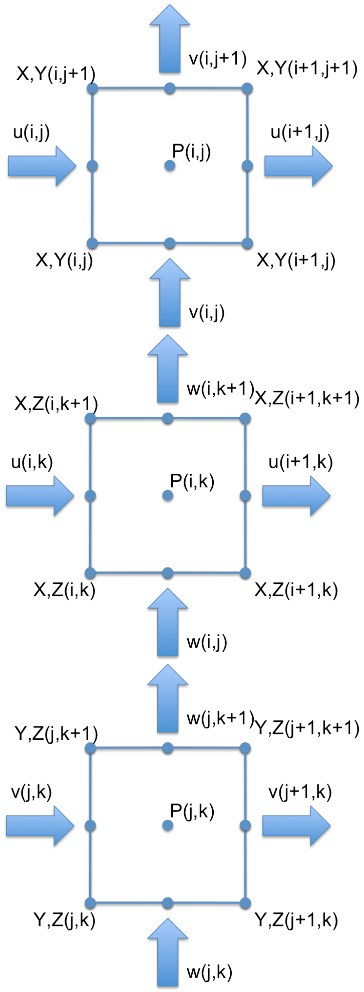

Figure 23: Location of the variables and their indices.

11. Algorithm and Flow chart:

The detail of how each different module is executed in GCOM is shown in Figure 24. It has to be noted that the procedure shown in Figure 24 is based on the fractional step method.

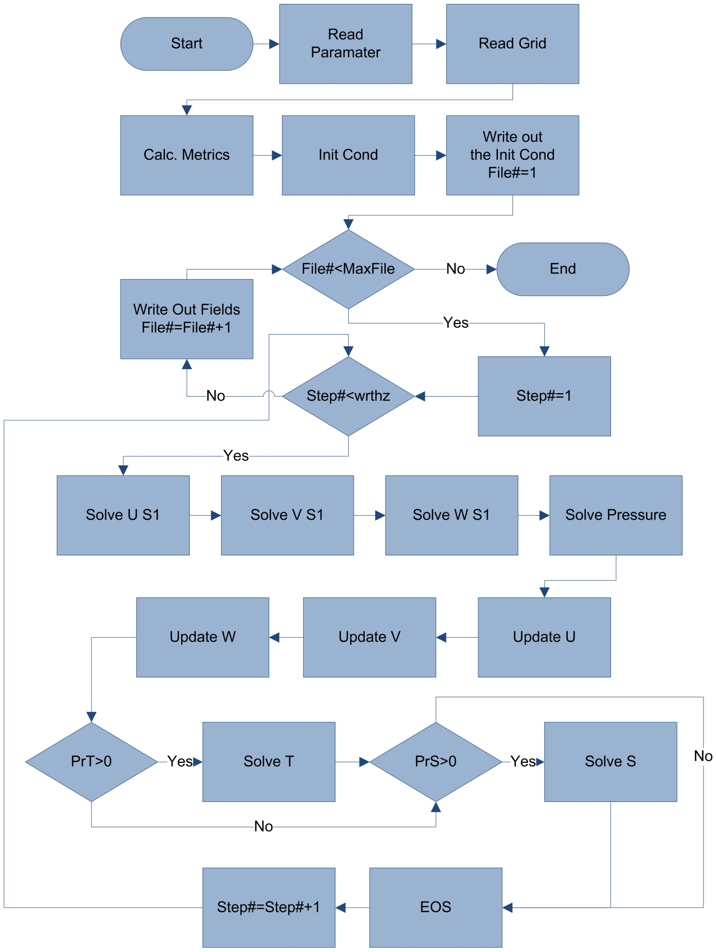

Figure 24: Flow Chart of GCOM-NG based on Fractional Step.

Table 1: Full list of variables.

Name	Size	Description
IMax	1	Number of nodes in I direction
JMax	1	Number of nodes in J direction
KMax	1	Number of Nodes in K direction
Idi	1	Inverse of the spacing of the cube in I direction
Idj	1	Inverse of the spacing of the cube in J direction
Idk	1	Inverse of the spacing of the cube in K direction
Idi2	1	Idi ^ 2
Idj2	1	Idj ^ 2
Idk2	1	Idk ^ 2
dt	1	Time Step
MaxFileNo	1	Maximum number of File
wrthz	1	Number of time steps between each file
omp	1	Omega for SOR solver
epsp	1	Maximum Error in SOR solver
itemp	1	Maximum number of iterations in SOR Solver
Re	1	Reynolds Number
PrT	1	Prandtl Number for temperature
PrS	1	Schmidt Number for Salinity
Ros	1	Rossby Number
Fr	1	Froud Number
UStar	1	Velocity Scale
LStar	1	Length Scale
TStar	1	Temperatur Scale
SStar	1	Salinity Scale
rhoStar	1	Reference Density
gforce	1	acceleration due to gravity
IRe	1	Inverse of Reynolds number
IPrT	1	Inverse of Prandtl number
IPrS	1	Inverse of Schmidt number

IRos	1	Inverse of Rossby number
IFr2	1	Inverse of Froud number to the second power
x	(1:IMax,1:Jmax,1:Kmax)	x coordinate of the grid nodes
y	(1:IMax,1:Jmax,1:Kmax)	y coordinate of the grid nodes
z	(1:IMax,1:Jmax,1:Kmax)	z coordinate of the grid nodes
P	(-1:IMax+1,-1:JMax+1,-1:KMax+1)	Pressure
T, T0	(-1:IMax+1,-1:JMax+1,-1:KMax+1)	Temperature
S, S0	(-1:IMax+1,-1:JMax+1,-1:KMax+1)	Salinity
dens	(-1:IMax+1,-1:JMax+1,-1:KMax+1)	Density
u, u0	(-1:IMax+2,-1:JMax+1,-1:KMax+1)	u component of the velocity
v, v0	(-1:IMax+1,-1:JMax+2,-1:KMax+1)	v component of the velocity
w, w0	(-1:IMax+1,-1:JMax+1,-1:KMax+2)	w component of the velocity
ixc	(1:IMax-1,1:JMax-1,1:KMax-1)	ix=di/dx at the center of the cell
iyc	(1:IMax-1,1:JMax-1,1:KMax-1)	iy=di/dy at the center of the cell
izc	(1:IMax-1,1:JMax-1,1:KMax-1)	iz=di/dz at the center of the cell
jxc	(1:IMax-1,1:JMax-1,1:KMax-1)	jx=dj/dx at the center of the cell
jyc	(1:IMax-1,1:JMax-1,1:KMax-1)	jy=dj/dy at the center of the cell
jzc	(1:IMax-1,1:JMax-1,1:KMax-1)	jz=dj/dz at the center of the cell
kxc	(1:IMax-1,1:JMax-1,1:KMax-1)	kx=dk/dx at the center of the cell
kyc	(1:IMax-1,1:JMax-1,1:KMax-1)	ky=dk/dy at the center of the cell
kzc	(1:IMax-1,1:JMax-1,1:KMax-1)	kz=dk/dz at the center of the cell
Jacc	(1:IMax-1,1:JMax-1,1:KMax-1)	The Jacobian of transformation at the center of the cell
ac	(1:IMax-1,1:JMax-1,1:KMax-1)	a=ix^2 + iy^2 + iz^2 at the center of the cell
bc	(1:IMax-1,1:JMax-1,1:KMax-1)	b=jx^2 + jy^2 + jz^2 at the center of the cell
cc	(1:IMax-1,1:JMax-1,1:KMax-1)	c=kx^2 + ky^2 + kz^2 at the center of the cell
dc	(1:IMax-1,1:JMax-1,1:KMax-1)	d=jx*ix+jy*iy+jz*iz at the center of the cell
ec	(1:IMax-1,1:JMax-1,1:KMax-1)	e=kx*jx+ky*jy+kz*jz at the center of the cell
qc	(1:IMax-1,1:JMax-1,1:KMax-1)	q=kx*ix+ky*iy+kz*iz at the center of the cell
Lxc	(1:IMax-1,1:JMax-1,1:KMax-1)	Lx=a*xii+b*xjj+c*xkk+2.0D0*(d*xij+e*xjk+q*xik) at the center of the cell
Lyc	(1:IMax-1,1:JMax-1,1:KMax-1)	Ly=a*yii+b*yjj+c*ykk+2.0D0*(d*yij+e*yjk+q*yik) at the center of the cell
Lzc	(1:IMax-1,1:JMax-1,1:KMax-1)	Lz=a*zii+b*zjj+c*zkk+2.0D0*(d*zij+e*zjk+q*zik) at the center of the cell
fnpc	(1:IMax-1,1:JMax-1,1:KMax-1)	fnp=-2*(a+b+c) at the center of the cell
ixu	(1:IMax,1:JMax-1,1:KMax-1)	ix=di/dxat the same position as U-Component of the velocity

iyu	(1:IMax,1:JMax-1,1:KMax-1)	iy=di/dyat the same position as U-Component of the velocity
izu	(1:IMax,1:JMax-1,1:KMax-1)	iz=di/dzat the same position as U-Component of the velocity
jxu	(1:IMax,1:JMax-1,1:KMax-1)	jx=dj/dxat the same position as U-Component of the velocity
jyu	(1:IMax,1:JMax-1,1:KMax-1)	jy=dj/dyat the same position as U-Component of the velocity
jzu	(1:IMax,1:JMax-1,1:KMax-1)	jz=dj/dzat the same position as U-Component of the velocity
kxu	(1:IMax,1:JMax-1,1:KMax-1)	kx=dk/dxat the same position as U-Component of the velocity
kyu	(1:IMax,1:JMax-1,1:KMax-1)	ky=dk/dyat the same position as U-Component of the velocity
kzu	(1:IMax,1:JMax-1,1:KMax-1)	kz=dk/dzat the same position as U-Component of the velocity
Jauu	(1:IMax,1:JMax-1,1:KMax-1)	The Jacobian of transformationat the same position as U-Component of the velocity
au	(1:IMax,1:JMax-1,1:KMax-1)	a=ix^2 + iy^2 + iz^2at the same position as U-Component of the velocity
bu	(1:IMax,1:JMax-1,1:KMax-1)	b=jx^2 + jy^2 + jz^2at the same position as U-Component of the velocity
cu	(1:IMax,1:JMax-1,1:KMax-1)	c=kx^2 + ky^2 + kz^2at the same position as U-Component of the velocity
du	(1:IMax,1:JMax-1,1:KMax-1)	d=jx*ix+jy*iy+jz*izat the same position as U-Component of the velocity
eu	(1:IMax,1:JMax-1,1:KMax-1)	e=kx*jx+ky*jy+kz*jzat the same position as U-Component of the velocity
qu	(1:IMax,1:JMax-1,1:KMax-1)	q=kx*ix+ky*iy+kz*izat the same position as U-Component of the velocity
Lxu	(1:IMax,1:JMax-1,1:KMax-1)	Lx=a*xii+b*xjj+c*xkk+2.0D0*(d*xij+e*xjk+q*xik)at the same position as U-Component of the velocity
Lyu	(1:IMax,1:JMax-1,1:KMax-1)	Ly=a*yii+b*yjj+c*ykk+2.0D0*(d*yij+e*yjk+q*yik)at the same position as U-Component of the velocity
Lzu	(1:IMax,1:JMax-1,1:KMax-1)	Lz=a*zii+b*zjj+c*zkk+2.0D0*(d*zij+e*zjk+q*zik)at the same position as U-Component of the velocity
ixv	(1:Imax-1,1:JMax,1:KMax-1)	ix=di/dxat the same position as V-Component of the velocity
iyv	(1:Imax-1,1:JMax,1:KMax-1)	iy=di/dyat the same position as V-Component of the velocity
izv	(1:Imax-1,1:JMax,1:KMax-1)	iz=di/dzat the same position as V-Component of the velocity
jxv	(1:Imax-1,1:JMax,1:KMax-1)	jx=dj/dxat the same position as V-Component of the velocity
jyv	(1:Imax-1,1:JMax,1:KMax-1)	jy=dj/dyat the same position as V-Component of the velocity
jzv	(1:Imax-1,1:JMax,1:KMax-1)	jz=dj/dzat the same position as V-Component of the velocity
kxv	(1:Imax-1,1:JMax,1:KMax-1)	kx=dk/dxat the same position as V-Component of the velocity
kyv	(1:Imax-1,1:JMax,1:KMax-1)	ky=dk/dyat the same position as V-Component of the velocity
kzv	(1:Imax-1,1:JMax,1:KMax-1)	kz=dk/dzat the same position as V-Component of the velocity
Javv	(1:Imax-1,1:JMax,1:KMax-1)	The Jacobian of transformationat the same position as V-Component of the velocity
av	(1:Imax-1,1:JMax,1:KMax-1)	a=ix^2 + iy^2 + iz^2at the same position as V-Component of the velocity
bv	(1:Imax-1,1:JMax,1:KMax-1)	b=jx^2 + jy^2 + jz^2at the same position as V-Component of the velocity

cv	(1:Imax-1,1:JMax,1:KMax-1)	c=kx^2 + ky^2 + kz^2at the same position as V-Component of the velocity
dv	(1:Imax-1,1:JMax,1:KMax-1)	d=jx*ix+jy*iy+jz*izat the same position as V-Component of the velocity
ev	(1:Imax-1,1:JMax,1:KMax-1)	e=kx*jx+ky*jy+kz*jzat the same position as V-Component of the velocity
qv	(1:Imax-1,1:JMax,1:KMax-1)	q=kx*ix+ky*iy+kz*izat the same position as V-Component of the velocity
Lxv	(1:Imax-1,1:JMax,1:KMax-1)	Lx=a*xii+b*xjj+c*xkk+2.0D0*(d*xij+e*xjk+q*xik)at the same position as V-Component of the velocity
Lyv	(1:Imax-1,1:JMax,1:KMax-1)	Ly=a*yii+b*yjj+c*ykk+2.0D0*(d*yij+e*yjk+q*yik)at the same position as V-Component of the velocity
Lzv	(1:Imax-1,1:JMax,1:KMax-1)	Lz=a*zii+b*zjj+c*zkk+2.0D0*(d*zij+e*zjk+q*zik)at the same position as V-Component of the velocity
ixw	(1:Imax-1,1:Jmax-1,1:KMax)	ix=di/dxat the same position as W-Component of the velocity
iyw	(1:Imax-1,1:Jmax-1,1:KMax)	iy=di/dyat the same position as W-Component of the velocity
izw	(1:Imax-1,1:Jmax-1,1:KMax)	iz=di/dzat the same position as W-Component of the velocity
jxw	(1:Imax-1,1:Jmax-1,1:KMax)	jx=dj/dxat the same position as W-Component of the velocity
jyw	(1:Imax-1,1:Jmax-1,1:KMax)	jy=dj/dyat the same position as W-Component of the velocity
jzw	(1:Imax-1,1:Jmax-1,1:KMax)	jz=dj/dzat the same position as W-Component of the velocity
kxw	(1:Imax-1,1:Jmax-1,1:KMax)	kx=dk/dxat the same position as W-Component of the velocity
kyw	(1:Imax-1,1:Jmax-1,1:KMax)	ky=dk/dyat the same position as W-Component of the velocity
kzw	(1:Imax-1,1:Jmax-1,1:KMax)	kz=dk/dzat the same position as W-Component of the velocity
Jaww	(1:Imax-1,1:Jmax-1,1:KMax)	The Jacobian of transformationat the same position as W-Component of the velocity
aw	(1:Imax-1,1:Jmax-1,1:KMax)	a=ix^2 + iy^2 + iz^2at the same position as W-Component of the velocity
bw	(1:Imax-1,1:Jmax-1,1:KMax)	b=jx^2 + jy^2 + jz^2at the same position as W-Component of the velocity
cw	(1:Imax-1,1:Jmax-1,1:KMax)	c=kx^2 + ky^2 + kz^2at the same position as W-Component of the velocity
dw	(1:Imax-1,1:Jmax-1,1:KMax)	d=jx*ix+jy*iy+jz*izat the same position as W-Component of the velocity
ew	(1:Imax-1,1:Jmax-1,1:KMax)	e=kx*jx+ky*jy+kz*jzat the same position as W-Component of the velocity
qw	(1:Imax-1,1:Jmax-1,1:KMax)	q=kx*ix+ky*iy+kz*izat the same position as W-Component of the velocity
Lxw	(1:Imax-1,1:Jmax-1,1:KMax)	Lx=a*xii+b*xjj+c*xkk+2.0D0*(d*xij+e*xjk+q*xik)at the same position as W-Component of the velocity
Lyw	(1:Imax-1,1:Jmax-1,1:KMax)	Ly=a*yii+b*yjj+c*ykk+2.0D0*(d*yij+e*yjk+q*yik)at the same position as W-Component of the velocity
Lzw	(1:Imax-1,1:Jmax-1,1:KMax)	Lz=a*zii+b*zjj+c*zkk+2.0D0*(d*zij+e*zjk+q*zik)at the same position as W-Component of the velocity

Bibliography

- Ferziger, Joel H., and Milovan Peric. *Computational Methods for Fluid Dynamics.* 3rd Edition. Vol. 1. 1 vols. Springer, 2001.
- Fofonoff, N.P., and R.C. Millard. *Algorithms for computation of fundamental properties of seawater.* UNESCO, 1983.
- Geurts, Bernard. *Elements of direct and large-eddy simulation.* R.T. Edwards, Inc, 2003.
- Gill, A. E. *Atmosphere-Ocean dynamics.* New York: Acadmeic Press, 1982.
- Harlow, Francis H., and Eddie Welch. "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface." *The Physics of Fluids* 8, no. 12 (1965): 2182-2189.
- Hoffmann, Klaus A., and Steve T. Chiang. *Computational Fluid Dynamics.* Vol. 2. 3 vols. Engineering Education System Book, 2000.
- Kawamura, Tetuya, Hideo Takami, and Kunio Kuwahara. "Computation of High Reynolds Number flow around a circular cylinder with surface roughness." *Fluid Dynamics Research* 1 (1986): 145.
- Kim, J., and P. Moin. "Application of a fractional-step method to incompressible Navier-Stokes equations." *Journal of Computational Physics* 59 (1985): 208-232.
- Langtangen, Hans P., Kent-Andre Mardal, and Ragnar Winther. "Numerical methods for incompressible viscous flow." *Advances in Water Resources* 25 (2002): 1125-1146.
- Marshal, John, and R. Alan Plumb. *Atmosphere, ocean, and climate dynamics : an introductory text.* Amsterdam: Elsevier Academic Press, 2008.
- McDougall, Trevor J., and David R. Jackett. "Accurate and computationally efficient algorithms for potential temperature and density of seawater." *Journal of Atmospheric and Oceanic Technology* 20 (2003): 730.
- Millero, Frank J., Chen-Tung Chen, Alvin Bradshaw, and Karl Schleicher. "A new high pressure equation of state for seawater." *Deep-Sea Research* 27A (1980): 255-264.
- Millero, Frank, and Alain Poisson. "International one-atmosphere equation of state of seawater." *Deep-Sea Research* 28A (1981): 625-629.
- Nievergelt, Y. *Wavelets Made Easy.* Birkhaeser.
- Pond, S., and G. L. Pickard. *Introductory Dynamical Oceanography, 2nd ed.* New York: Pergamon, 1989.
- Salmon, Rick. *Lectures on geophysical fluid dynamics.* New York: Oxford University Press, 1998.
- Stull, Roland. *An Introduction to Boundary Layer Meteorology.* Kluwer Academic Pub, 1988.
- Talukder, K.H., and K. Harada. "Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image." *IAENG International Journal of Applied Mathematics* 36: 1-9.

- Torres, C.R., A.S. Mascarenhas, and J.E. Castillo. “Three-dimensinal stratified flow over Alarcon Seamount Gulf of California entrance.” *Seep-Sea Researcg Part II* 51 (2004): 647-657.
- Torres, Carlos, and Jose Castillo. “Stratified rotating flow over complex terrain.” *Applied Numerical Mathematics* 47 (2003): 531-541.
- Torres, Carlos, Jose Castillo, Jim Mueller, Charles Trees, and German Larrazabal. “Modeling 3D Coastal Circulation in Boundary-fitted Coordinates: The GCOM Project.” *California and the World Ocean 06 Conference.* Long Beach, 2006.
- UNESCO. *Tenth report of the joint panel on oceanographic tables and standards.* UNCESCO, 1980.