CE-580

COMPUTATIONAL TECHNIQUES

FOR

FLUID DYNAMICS

HOMEWORK #9

Solution of a Flow over a Backward Step using U-V-P Formulation

Taha Yaşar Demir

1881978

Problem Definition and Assumptions

Assumptions:

- 2-D dimensional flow
- Steady solution
- Laminar Flow
- Incompressible Fluid
- Uniform Incoming Flow

Problem is a flow over a backward-facing step. There is a sudden enlargement of cross section. Domain dimensions and parameter can be seen in figure 1.

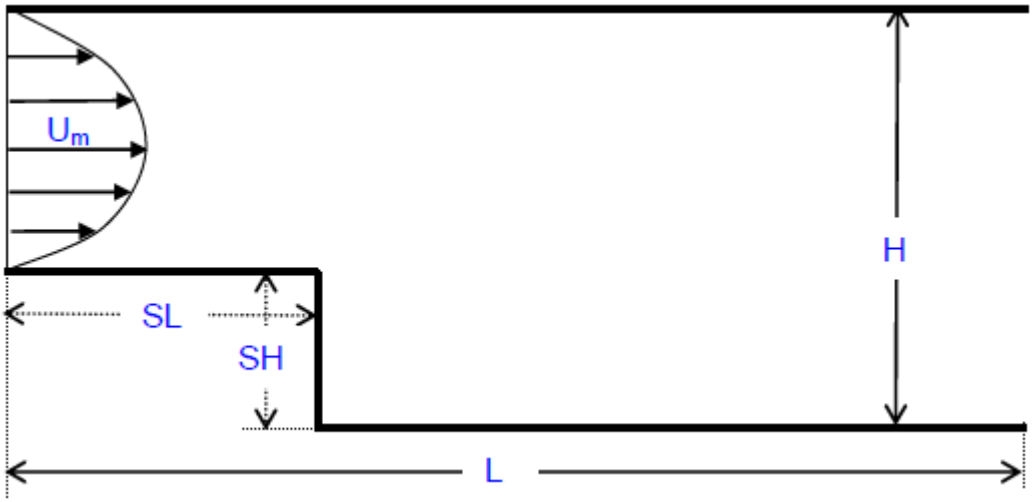


Figure 1: Problem Domain

Dimensions and parameters are

$$U_m = 0.2; 1; 2\ m/s\,, \qquad H = 0.1\ m\,, \qquad SL = 0.4\ m\,, \qquad SH = 0.05\ m\,, \qquad L = 2\ m$$

$$\nu = 10^{-4}\frac{m^2}{s}\,, \qquad \rho = 1000\ kg/m^3$$

Discretization

For two-dimensional laminar flow of an incompressible fluid is

x-momentum:

$$\frac{\partial}{\partial t}\int_{cv} u dV + \int_{cs} u\vec{V} dA = \int_{cs} \nu \nabla u \, dA - \int_{cv} \frac{1}{\rho}\frac{dp}{dx} dV$$

y-momentum:

$$\frac{\partial}{\partial t}\int_{cv} v dV + \int_{cs} v\vec{V} dA = \int_{cs} \nu \nabla v \, dA - \int_{cv} \frac{1}{\rho}\frac{dp}{dy} dV$$

Continuity:

$$\int_{cs} \vec{V} dA = 0$$

The domain is divided into finite volumes, CVs and the notation for a cartesian 2-D grid is given below in figure 2.

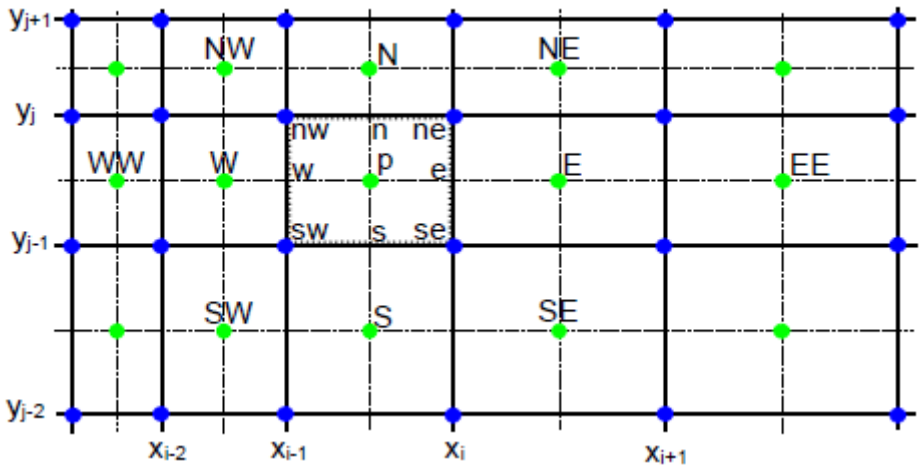


Figure 2: CVs and Notation

Discretization is done using staggered mesh system, cell centers are shifted so that it coincides with the dependent variables being solved. Figure 3 show the shifted control volumes for x-momentum.

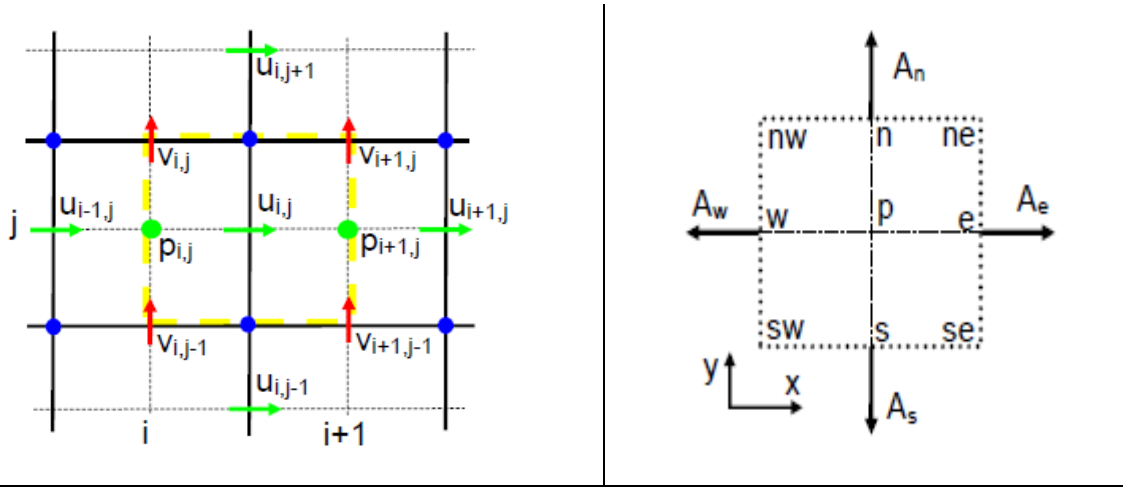


Figure 3: Shifted CV for X-Momentum(u-cell)

Now, x-momentum terms turned into finite volume formulation according to notation in figure 3.

The time rate of change term;

$$\frac{\partial}{\partial t}\int_{cv} udV = \frac{\partial}{\partial t}(u\Delta V) = \frac{\partial u}{\partial t}(\Delta V) = \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t}\Delta x\Delta y$$

The convective term;

$$\int_{cs} u\vec{V}dA = \int_{cs} u(udy) - u(vdx) = \sum_{enws} u(u\Delta y) - \sum_{enws} u(v\Delta x) = u_e q_e - u_w q_w + u_n q_n - u_s q_s$$
$$= ConU$$

Where, q_e, q_w, q_n, q_s are approximated for midpoints (2nd order accurate) such as,

$$q_e = (u_{i,j} + u_{i+1,j})\Delta y/2$$

$$q_w = (u_{i-1,j} + u_{i,j})\Delta y/2$$

$$q_n = (v_{i,j} + v_{i+1,j})\Delta x/2$$

$$q_s = (v_{i,j-1} + v_{i+1,j-1})\Delta x/2$$

Now we need u velocities

$$u_e = [u_{i,j} + u_{i+1,j} + \gamma\, sign(q_e)(u_{i,j} - u_{i+1,j})]/2$$

$$u_n = [u_{i,j} + u_{i,j+1} + \gamma\, sign(q_n)(u_{i,j} - u_{i,j+1})]/2$$

$$u_w = [u_{i-1,j} + u_{i,j} + \gamma\, sign(q_w)(u_{i-1,j} - u_{i,j})]/2$$

$$u_s = [u_{i,j-1} + u_{i,j} + \gamma\, sign(q_s)(u_{i,j-1} - u_{i,j})]/2$$

γ has values between 1 and 0, it is used for switching the calculation method of u values at walls. γ is assigned value of 1 for upwinding and 0 for central differences.

The diffusive term;

$$\int_{cs} \nu \nabla u \, dA = \int_{cs} \nu \left(\frac{du}{dx}dy - \frac{du}{dy}dx\right) = \sum_{ensw} \nu \left(\frac{du}{dx}\Delta y - \frac{du}{dy}\Delta x\right)$$
$$= \nu \left[\frac{\partial u}{\partial x}|_e \Delta y - \frac{\partial u}{\partial x}|_w \Delta y + \frac{\partial u}{\partial y}|_n \Delta x - \frac{\partial u}{\partial y}|_s \Delta x\right]$$
$$= \nu \left[\left(\frac{u_{i+1,j} - u_{i,j}}{\Delta x} - \frac{u_{i,j} - u_{i-1,j}}{\Delta x}\right)\Delta y + \left(\frac{u_{i,j+1} - u_{i,j}}{\Delta y} - \frac{u_{i,j} - u_{i,j-1}}{\Delta y}\right)\Delta x\right] = DifU$$

And Finally, Source term

$$-\int_{cv} \frac{1}{\rho}\frac{dp}{dx} dV = -\frac{1}{\rho}\int_{cv} \frac{dp}{dx} dxdy = -\frac{1}{\rho}\oint p dy = -\frac{p_e \Delta y - p_w \Delta y}{\rho} = \frac{1}{\rho}(p_{i,j} - p_{i+1,j})\Delta y$$

Now, Combining all discrete approximations in the x-momentum;

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}[DifU - ConU] + \frac{\Delta t}{\rho \Delta x}(p_{i,j} - p_{i+1,j})$$

$$u_{i,j}^{n+1} = F_{i,j}^{n} + \frac{\Delta t}{\rho \Delta x}(p_{i,j}^{n+1} - p_{i+1,j}^{n+1}), \qquad u_{i-1,j}^{n+1} = F_{i-1,j}^{n} + \frac{\Delta t}{\rho \Delta x}(p_{i-1,j}^{n+1} - p_{i,j}^{n+1})$$

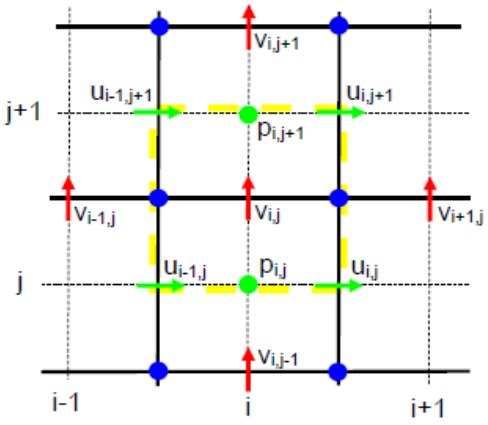
Where;

$$F_{i,j}^{n} = u_{i,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}[DifU - ConU]_{i,j}^{n}, \qquad F_{i-1,j}^{n} = u_{i-1,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}[DifU - ConU]_{i-1,j}^{n}$$

Procedure is similar for y-momentum;

$$\frac{\partial}{\partial t}\int_{cv} v dV + \int_{cs} v\vec{V} dA = \int_{cs} \nu \nabla v \, dA - \int_{cv} \frac{1}{\rho}\frac{dp}{dy} dV$$

The grid system is again shifted for v-cells as seen in figure 4



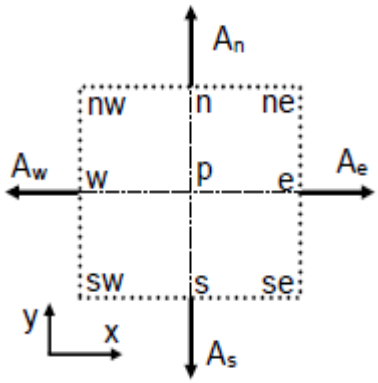


Figure 4: Shifted CV for Y-Momentum(u-cell)

The time rate of change term;

$$\frac{\partial}{\partial t}\int_{cv} v dV = \frac{\partial}{\partial t}(v\Delta V) = \frac{\partial v}{\partial t}\Delta V = \frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t}\Delta x \Delta y$$

The convective term;

$$\int_{cs} v\vec{V} dA = \oint v(udy) - v(vdx) = \sum_{enws} v(u\Delta y) - \sum_{enws} v(v\Delta x) = v_e q_e - v_w q_w + v_n q_n - v_s q_s$$
$$= ConV$$

Where

$$q_e = (u_{i,j} + u_{i,j+1})\Delta y/2$$

$$q_w = (u_{i-1,j} + u_{i-1,j+1})\Delta y/2$$

$$q_n = (v_{i,j} + v_{i,j-1})\Delta x/2$$

$$q_s = (v_{i,j} + v_{i,j-1})\Delta x/2$$

$$v_e = [v_{i,j} + v_{i+1,j} + \gamma\, sign(q_e)(v_{i,j} - v_{i+1,j})]/2$$

$$v_w = [v_{i-1,j} + v_{i,j} + \gamma\, sign(q_w)(v_{i-1,j} - v_{i,j})]/2$$

$$v_n = [v_{i,j} + v_{i,j+1} + \gamma\, sign(q_n)(v_{i,j} - v_{i,j+1})]/2$$

$$v_s = [v_{i,j-1} + v_{i,j} + \gamma\, sign(q_s)(v_{i,j-1} - v_{i,j})]/2$$

The diffusive term;

$$\int_{cs} v\nabla v\, dA = \oint v\left(\frac{dv}{dx}dy - \frac{dv}{dy}dx\right) = \sum_{ensw} v\left(\frac{dv}{dx}\Delta y - \frac{dv}{dy}\Delta x\right)$$
$$= v\left[\frac{\partial v}{\partial x}|_e \Delta y - \frac{\partial v}{\partial x}|_w \Delta y + \frac{\partial v}{\partial y}|_n \Delta x - \frac{\partial v}{\partial y}|_s \Delta x\right]$$
$$= v\left[\left(\frac{v_{i+1,j} - v_{i,j}}{\Delta x} - \frac{v_{i,j} - v_{i-1,j}}{\Delta x}\right)\Delta y + \left(\frac{v_{i,j+1} - v_{i,j}}{\Delta y} - \frac{v_{i,j} - v_{i,j-1}}{\Delta y}\right)\Delta x\right] = DifV$$

Finally, source term

$$-\int_{cv} \frac{1}{\rho}\frac{dp}{dy} dV = -\frac{1}{\rho}\int_{cv} \frac{dp}{dy} dx dy = -\frac{1}{\rho}\oint p dx = -\frac{p_e \Delta x - p_w \Delta x}{\rho} = \frac{1}{\rho}(p_{i,j} - p_{i,j+1})\Delta x$$

Now, Combining all discrete approximations in the x-momentum;

$$v_{i,j}^{n+1} = v_{i,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}[DifV - ConV] + \frac{\Delta t}{\rho \Delta x}\left(p_{i,j} - p_{i,j+1}\right)$$

$$v_{i,j}^{n+1} = G_{i,j}^{n} + \frac{\Delta t}{\rho \Delta x}\left(p_{i,j}^{n+1} - p_{i,j+1}^{n+1}\right), \qquad v_{i-1,j}^{n+1} = G_{i-1,j}^{n} + \frac{\Delta t}{\rho \Delta x}\left(p_{i,j-1}^{n+1} - p_{i,j}^{n+1}\right)$$

Where;

$$G_{i,j}^{n} = v_{i,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}[DifV - ConV]_{i,j}^{n}, \qquad G_{i-1,j}^{n} = v_{i-1,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}[DifV - ConV]_{i,j-1}^{n}$$

Third equation to apply finite volume formulation is the continuity equation, this time grid is not shifted. Pressures were already defined at the cell centers. For p-cells;

$$\int_{cs} \vec{V} dA = q_e - q_w + q_n - q_s = 0$$

$$\left(u_{i,j}^{n+1} - u_{i-1,j}^{n+1}\right)\Delta y + \left(v_{i,j}^{n+1} - v_{i,j-1}^{n+1}\right)\Delta x = 0$$

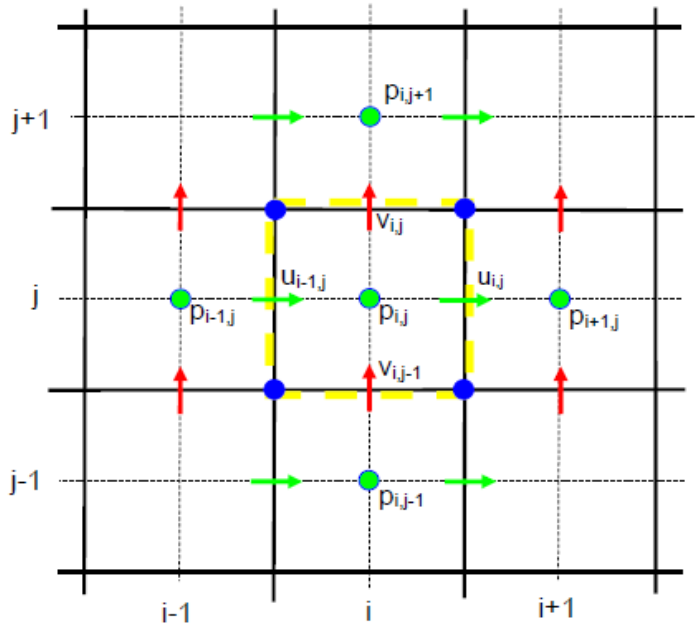


Figure 5: CV and notations for p-cells

Remember that u and v values at n+1 time level was formulated before, they are substituted in the formulated continuity equation that yields

$$\left[F_{i,j}^{n}+\frac{\Delta t}{\rho\Delta x}\left(p_{i,j}^{n+1}-p_{i+1,j}^{n+1}\right)-F_{i-1,j}^{n}-\frac{\Delta t}{\rho\Delta x}\left(p_{i-1,j}^{n+1}-p_{i,j}^{n+1}\right)\right]\Delta y$$
$$+\left[G_{i,j}^{n}+\frac{\Delta t}{\rho\Delta x}\left(p_{i,j}^{n+1}-p_{i,j+1}^{n+1}\right)-G_{i-1,j}^{n}+\frac{\Delta t}{\rho\Delta x}\left(p_{i,j-1}^{n+1}-p_{i,j}^{n+1}\right)\right]\Delta x=0$$

Rearranging above equation yields Poisson's equation for pressure;

$$\left[\frac{2}{\Delta x^{2}}+\frac{2}{\Delta y^{2}}\right]p_{i,j}^{n+1}=\frac{p_{i+1,j}^{n+1}+p_{i-1,j}^{n+1}}{\Delta x^{2}}+\frac{p_{i,j+1}^{n+1}+p_{i,j-1}^{n+1}}{\Delta y^{2}}-S_{i,j}$$

$$S_{i,j}=\left[\frac{F_{i,j}^{n}-F_{i-1,j}^{n}}{\Delta x}+\frac{G_{i,j}^{n}-G_{i,j-1}^{n}}{\Delta y}\right]\frac{\rho}{\Delta t}$$

Mesh System

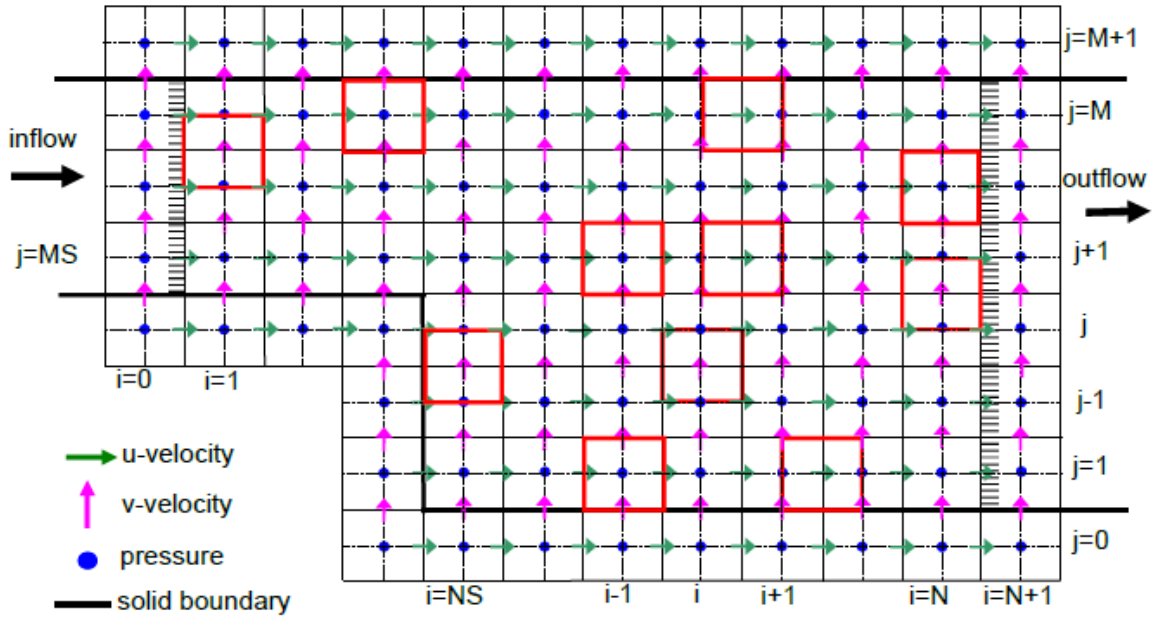


Figure 6: Mesh System

Mesh system is given in figure 6. Cell centers are shifted according to the variable that being solved. Also, there are ghost cells that are not being solved, these cells are used to implement boundary conditions. For Calculations 300x60 cells are used.

Boundary Conditions

Boundary conditions are applied with help of ghost cells. For simplicity these cells are assigned cell numbers too, this cause domain extent is different for each variable. Dimensions of each variable is found as

$$u(N+1, M+2)$$

$$v(N+2, M+1)$$

$$P(N+2, M+2)$$

With these indexes, boundary conditions are specified below. Note that indexes corresponding to step will be different

- Inflow;

$$v_{1,j} = 0, u_{1,j} = u_{uniform}(y)\,, F_{1,j} = u_{1,j}\,, P_{1,j} = P_{2,j}$$

- Outflow

$$v_{N+2,j} = 0, u_{N+1,j} = u_{N,j}\,, F_{N+1,j} = u_{N+1,j}\,, P_{N+2,j} = P_{N+1,j}$$

- Wall Boundaries
  - Horizontal part of the step (i=1 to NS-1)

$$v_{i,MS} = 0\,, u_{i,MS} = -u_{i,MS+1}\,, G_{i,MS} = v_{i,MS}\,, P_{i,MS} = P_{i,MS+1}$$

  - Vertical part of the step (j=1 to MS-1)

$$v_{NS,j} = -v_{NS+1,j}\,, u_{NS,j} = 0\,, F_{NS,j} = u_{NS,j}\,, P_{NS,j} = P_{NS+1,j}$$

  - Lower Wall (i=NS to N+2)

$$v_{i,1} = 0\,, u_{i,1} = -u_{i,2}\,, G_{i,1} = v_{i,1}\,, P_{i,1} = P_{i,2}$$

  - Upper Wall (i=1 to N+2)

$$v_{i,M+1} = 0\,, u_{i,M+2} = u_{i,M+1}\,, G_{i,M+1} = v_{i,M+1}\,, P_{i,M+2} = P_{i,M+1}$$

Now consider discretized x-momentum

$$u_{i,j}^{n+1} = F_{i,j}^{n} + \frac{\Delta t}{\rho \Delta x}\left(p_{i,j}^{n+1} - p_{i+1,j}^{n+1}\right)$$

At the inflow boundary(i=1);

$$u_{1,j}^{n+1} = F_{1,j}^{n} + \frac{\Delta t}{\rho \Delta x}\left(P_{1,j}^{n+1} - P_{2,j}^{n+1}\right)$$

Re writing pressure drop(i=1);

$$P_{1,j}^{n+1} - P_{2,j}^{n+1} = \frac{\rho \Delta t}{\Delta x}\left(u_{1,j}^{n+1} - F_{1,j}^{n}\right)$$

Pressure Poisson equation at i=2;

$$\left[\frac{P_{3,j}^{n+1} - 2P_{2,j}^{n+1} + P_{1,j}^{n+1}}{\Delta x^2} + \frac{P_{2,j+1}^{n+1} - 2P_{2,j}^{n+1} - P_{2,j-1}^{n+1}}{\Delta y^2}\right] = \left[\frac{F_{2,j}^{n} - F_{1,j}^{n}}{\Delta x} + \frac{G_{2,j}^{n} - G_{2,j-1}^{n}}{\Delta y}\right]\frac{\rho}{\Delta t}$$

Substituting the pressure drop at i=1 into the Poisson equation at i=2

$$\left[\frac{P_{3,j}^{n+1}-P_{2,j}^{n+1}}{\Delta x^2}+\frac{P_{2,j+1}^{n+1}-2P_{2,j}^{n+1}-P_{2,j-1}^{n+1}}{\Delta y^2}\right]=\left[\frac{F_{2,j}^{n}-u_{1,j}^{n+1}}{\Delta x}+\frac{G_{2,j}^{n}-G_{2,j-1}^{n}}{\Delta y}\right]\frac{\rho}{\Delta t}$$

Solution for pressure at node2 is independent of $F_{1,j}^{n}$. Therefore $F_{1,j}^{n}$ can be selected arbitrarily. A convenient choice is $F_{1,j}^{n}=u_{1,j}^{n+1}$ which leads to $P_{1,j}^{n+1}=P_{2,j}^{n+1}$. Note that this result of the normal derivative of the pressure be zero is artificial. If needed, after the final solution, the boundary value for pressure can be recalculated to get correct values.

Remember that inlet boundary condition is defined as uniform flow. Consider the x-momentum equation for 2-D, steady, laminar, incompressible, uniform flow between two parallel plates

$$\mu\frac{d^2u}{dy^2}=Cp$$

To get inlet velocity profile above equation is integrated twice with boundary conditions

$$@\ y=0\ ;u=0$$

$$@\ y=0.025\ (centerline);u=U_m\ ;\frac{du}{dy}=0$$

First integration yields

$$\mu\frac{du}{dy}=Cpy+C_1$$

Second integration yields

$$\mu u=\frac{Cpy^2}{2}+C_1y+C_2$$

Applying boundary conditions;

$$0.025Cp+C_1=0$$

$$\frac{0.025^2}{2}Cp+0.025C_1+C_2=U_m$$

$$C_2=0$$

Combining above equation gives velocity profile

$$\mu u(y)=-\left(\frac{0.1U_m}{0.025^2}\right)y^2+\left(\frac{0.2U_m}{0.025}\right)y$$

Numerical Solution Method

As previously mentioned, continuity equation yields Poisson equation for pressure.

$$P_{i,j}^{n+1} = P_{i,j}^{n} + \frac{1}{a}\left[\frac{P_{i+1,j}^{n+1} - P_{i-1,j}^{n+1}}{\Delta x^2} + \frac{P_{i,j+1}^{n+1} - P_{i,j-1}^{n}}{\Delta y^2} - S_{i,j}\right] - P_{i,j}^{n}$$

Where

$$a = \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}$$

Using Successive over relaxation method, above equation can be written as

$$P_{i,j}^{n+1} = P_{i,j}^{n} + \omega R_{i,j}$$

$$R_{i,j} = \frac{1}{a}\left[\frac{P_{i+1,j}^{n+1} - P_{i-1,j}^{n+1}}{\Delta x^2} + \frac{P_{i,j+1}^{n+1} - P_{i,j-1}^{n}}{\Delta y^2} - S_{i,j}\right] - P_{i,j}^{n}$$

Where ω is the over relaxation parameter which is between 1 and 2. Note that, while doing iterations, some P values will be at n+1 level and some will be at n level. As this is an iterative process, no need to keep an extra memory space for solution of this, just using most recent p values will work.

After Calculation of pressures, velocities are updated using equations;

$$u_{i,j}^{n+1} = F_{i,j}^{n} + \frac{\Delta t}{\rho \Delta x}\left(p_{i,j}^{n+1} - p_{i+1,j}^{n+1}\right) \; ; \; v_{i,j}^{n+1} = G_{i,j}^{n} + \frac{\Delta t}{\rho \Delta x}\left(p_{i,j}^{n+1} - p_{i,j+1}^{n+1}\right)$$

Note that terms F and G kept at time level n, since they consist nonlinear terms of u and v. With updated u and v values, source terms are calculated again, and pressures are updated.

Initial Conditions

Initial conditions are set as zero for Pressure and v-velocity component. For u-component, the inlet boundary condition is extended to the end of the step according to below formula.

$$\mu u(y) = -\left(\frac{0.1U_m}{0.025^2}\right)y^2 + \left(\frac{0.2U_m}{0.025}\right)y$$

After the step, again uniform flow Is assumed but this time equation constants are different since the parameters are different. In addition, velocity at centerline taken as $\frac{U_m}{2}$ in order to get same inlet and outlet discharge.

$$\mu u = -\left(\frac{0.05U_m}{0.05^2}\right)y^2 + \left(\frac{0.1U_m}{0.05}\right)$$

The u-velocity distribution is given in figure 7.

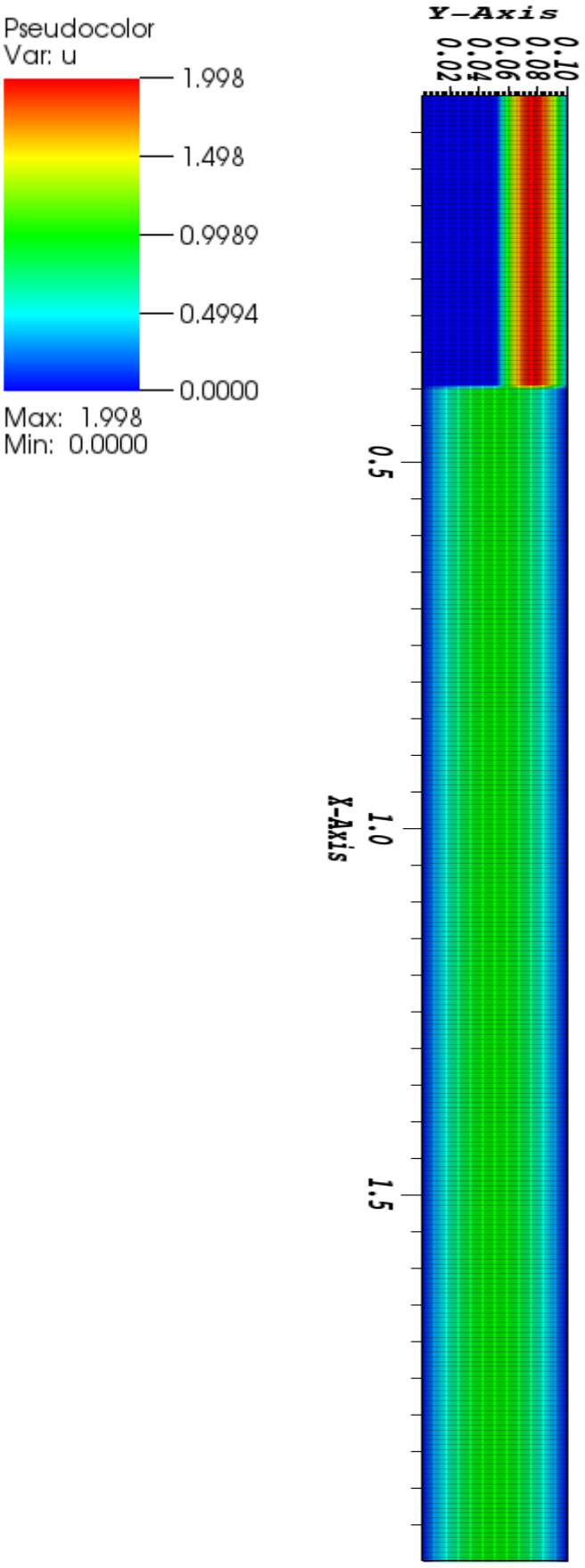


Figure 7: Initial Condition for u-velocity

Stability Conditions

Time step size is determined from CFL condition and Diffusion number. CFL condition and diffusion number definitions are given below

$$CFL = \alpha_x \frac{\Delta t}{\Delta x} + \alpha_y \frac{\Delta t}{\Delta y} < 1 \ ; \ d_f = \mu \left(\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} \right) \leq \frac{1}{2}$$

With 300x60 cell used;

$$\Delta x = \frac{2}{300} = 6.667 \times 10^{-3} \ ; \ \Delta y = \frac{0.1}{60} = 1.667 \times 10^{-3}$$

With $\alpha_x = 2, \alpha_y = 0, \mu = 0.1$,maximum time step allowed for CFL condition is approximately

$$\Delta t_{CFL_{max}} = 3.33 \times 10^{-3}$$

$$\Delta t_{d_{f_{max}}} = 1.307 \times 10^{-5}$$

Since there is 3 case with different centerline velocity, time step size for other cases are also calculated. Note that diffusion number won't be effected

For $U_m = 1 \ m/s$

$$\Delta t_{CFL_{max}} = 6.667 \times 10^{-3}$$

For $U_m = 0.2 \ m/s$

$$\Delta t_{CFL_{max}} = 3.33 \times 10^{-4}$$

So, the diffusion number sets the critical time step.

Condition for Convergence

Convergence condition is implemented in velocities. The difference between two consequent velocities are summed up and divided by the number of cells and U_m for normalization

$$Error = \frac{\sum_{i=1}^{N+1} \sum_{j=1}^{M+1} \left\{ \left| u_{old_{i,j}} - u_{i,j} \right| + \left| v_{old_{i,j}} - v_{i,j} \right| \right\}}{M * N * U_m}$$

Results and Discussion

Implementing the boundary conditions are very challenging in this problem. Since there is a step and the grid system is staggered. One must know the correct indexes corresponding to boundaries to implement boundary conditions. After carefully applying boundary conditions, rest is solving all the parameters iteratively and updating them.

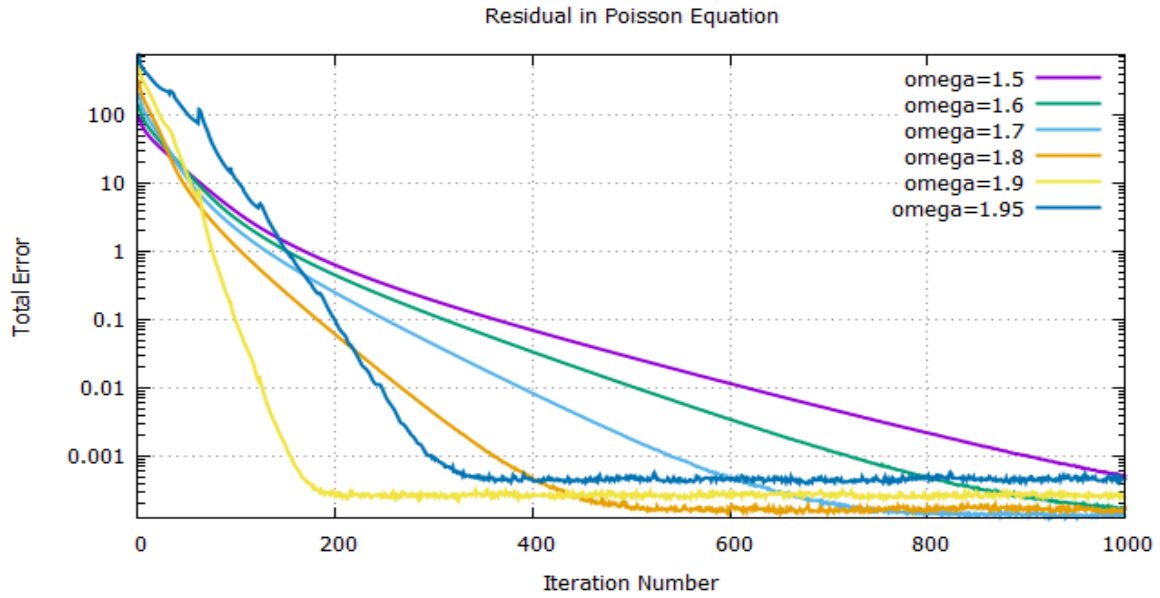


Figure 8: Effect of Omega on Residual

As mentioned before, continuity equation is represented as Poisson equation. Since, continuity must be satisfied at each iteration, Poisson equation itself must be iterated more than once. In this solution, Poisson equation iterated 30 times for each program loop. Also, over relaxation parameter is important in convergence of pressure Poisson equation. Effect of over-relaxation (omega) parameter on convergence is shown in figure 8 for 1000 iterations. Since, 30 iterations done in the code, best omega value is 1.8 for it has the steepest slope at the start.

For the general convergence, there is no strict convergence criteria. Error value of 10^{-6} seems like a good value but in the code, there are some points where total error gets below this value. But at those points solution is not converged. We can test the convergence by using another convergence parameter. A good choice of it is the discharge going in and going out of the domain. If discharge values are not perfectly met, the flow is not converged, or boundary conditions are not implemented correctly. So, constant 60 thousand iterations are done to match the incoming and outgoing flows. The results are tabulated in table 1. Note that, discharges are very close but not identical since there is numerical errors included. In addition, making 60 thousand iterations takes overall error to approximately $1.5\ 10^{-6}$ levels. Convergence history is presented in figure 9.

U_m	Re	$Discharge_{In}\ [m^2/s]$	$Discharge_{Out}\ [m^2/s]$	Percent Error
2.	2000	$6.67037\ 10^{-2}$	$6.67031\ 10^{-2}$	$8.153\ 10^{-4}$
1.	1000	$3.33518\ 10^{-2}$	$3.33521\ 10^{-2}$	$8.265\ 10^{-4}$
0.2	200	$6.67037\ 10^{-3}$	$6.67042\ 10^{-3}$	$7.679\ 10^{-4}$

Table 1: Incoming and Outgoing Discharges for all Cases

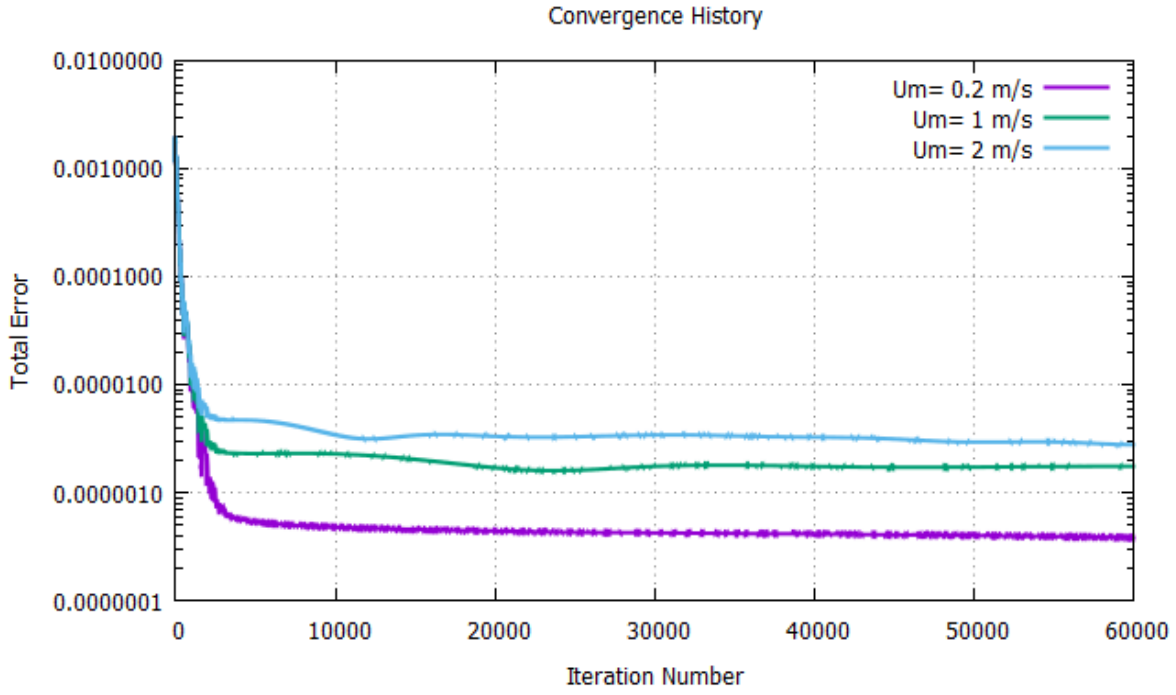


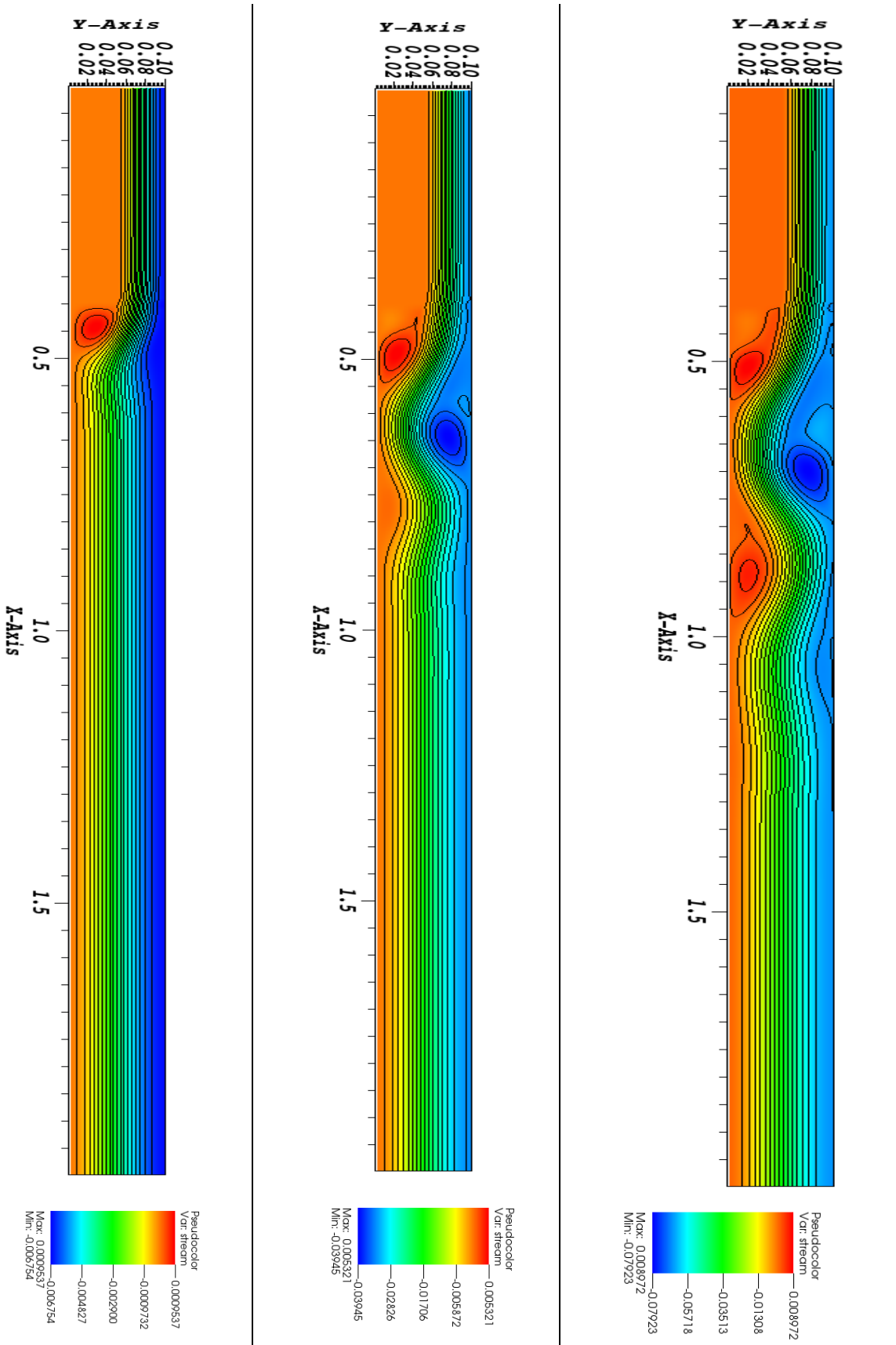
Figure 9: Convergence History

As Calculated before, maximum allowable time step is at around $1.3\ 10^{-5}$, but still this value causes oscillations in the solution when using upwinding schemes. In order to damp these oscillations, γ values can set to 0 and use central difference schemes in determining face values of velocities or smaller time steps can be used. Changing to central difference scheme perfectly eliminates the oscillations. However, there are vortexes caused by sudden expansion after the step, in these vortexes flow turns around. In this case, source of information should be adjusted by applying upwinding. Central difference results in wrong face values. So, it not a good way to handle the oscillations. Other method is to use smaller time steps. Reducing step size to 1×10^{-6} also eliminates the oscillations, but it takes much more computation time to converge and numerical error increases due to more operation done. To overcome this issue, combination of both methods is used, γ is assigned 0.9 and time step is decreased to 5×10^{-6}

Note that velocities are defined at the cell faces and their notation is shifted. To get a proper output, calculated velocities are averaged. By doing so, all the dependent variables u,c,P are represented at cell centers. This operation is just for visualizing the results does not included in the solution loop. In addition, stream function values are calculated from the lower wall. Stream function values are constant at walls, using this information lower wall stream function value is assigned as 0 and by forward differencing, all the stream function values integrated towards the upper wall.

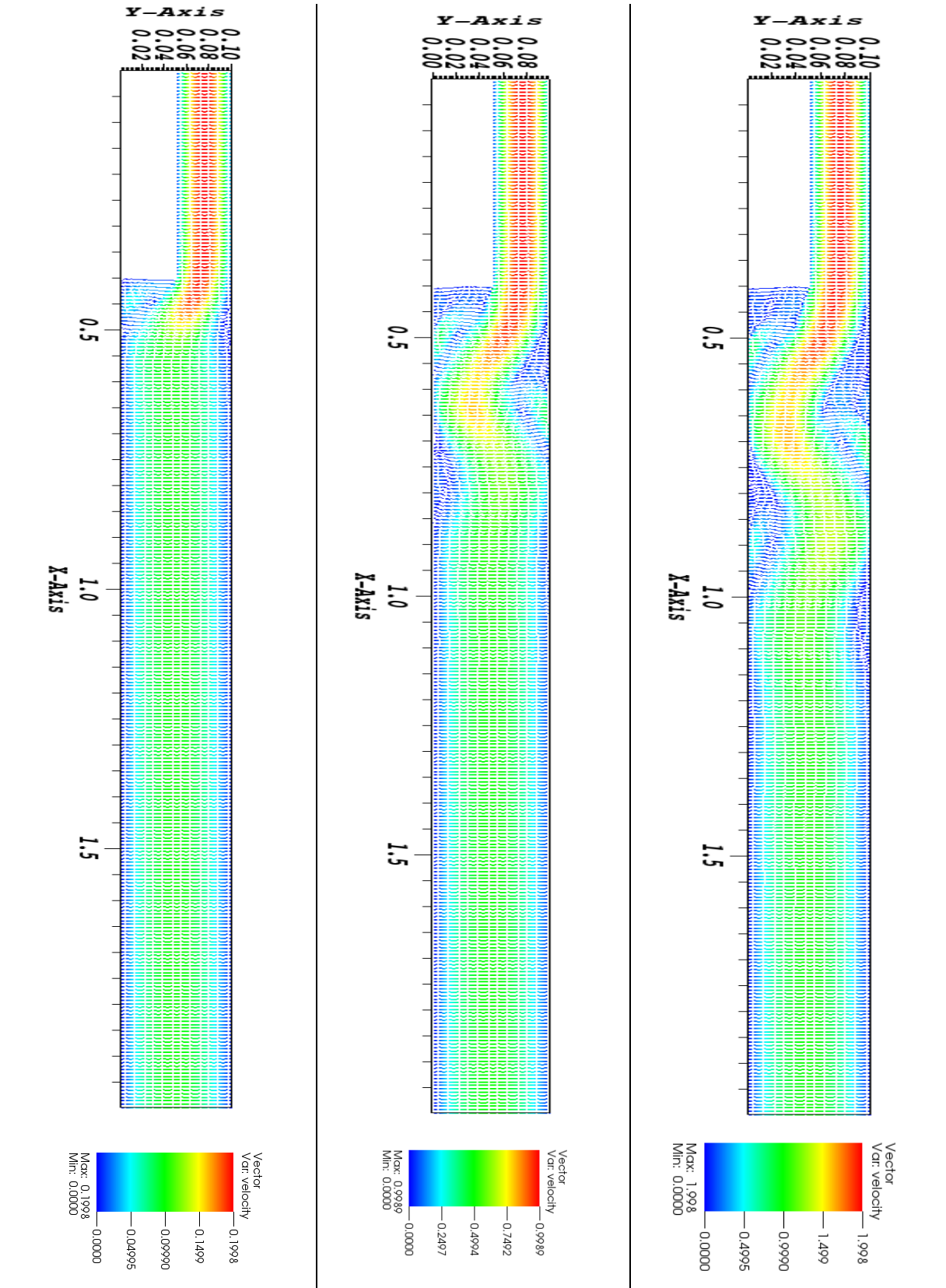
Results indicate that as Reynolds number increases the vortex formation becomes stronger and uniform flow formation is greatly disturbed. It takes more distance for flow to adjust itself and become uniform again. This is the reason why domain should be long enough to get correct solution.

For Re=2000 there are 3 vortexes, for Re=1000 there are 2 vortexes and for Re=200 there are one vortex just behind the step. All three results are represented in figure 10,11 and 12, as seen from the figures in all three cases outflow becomes uniform again.



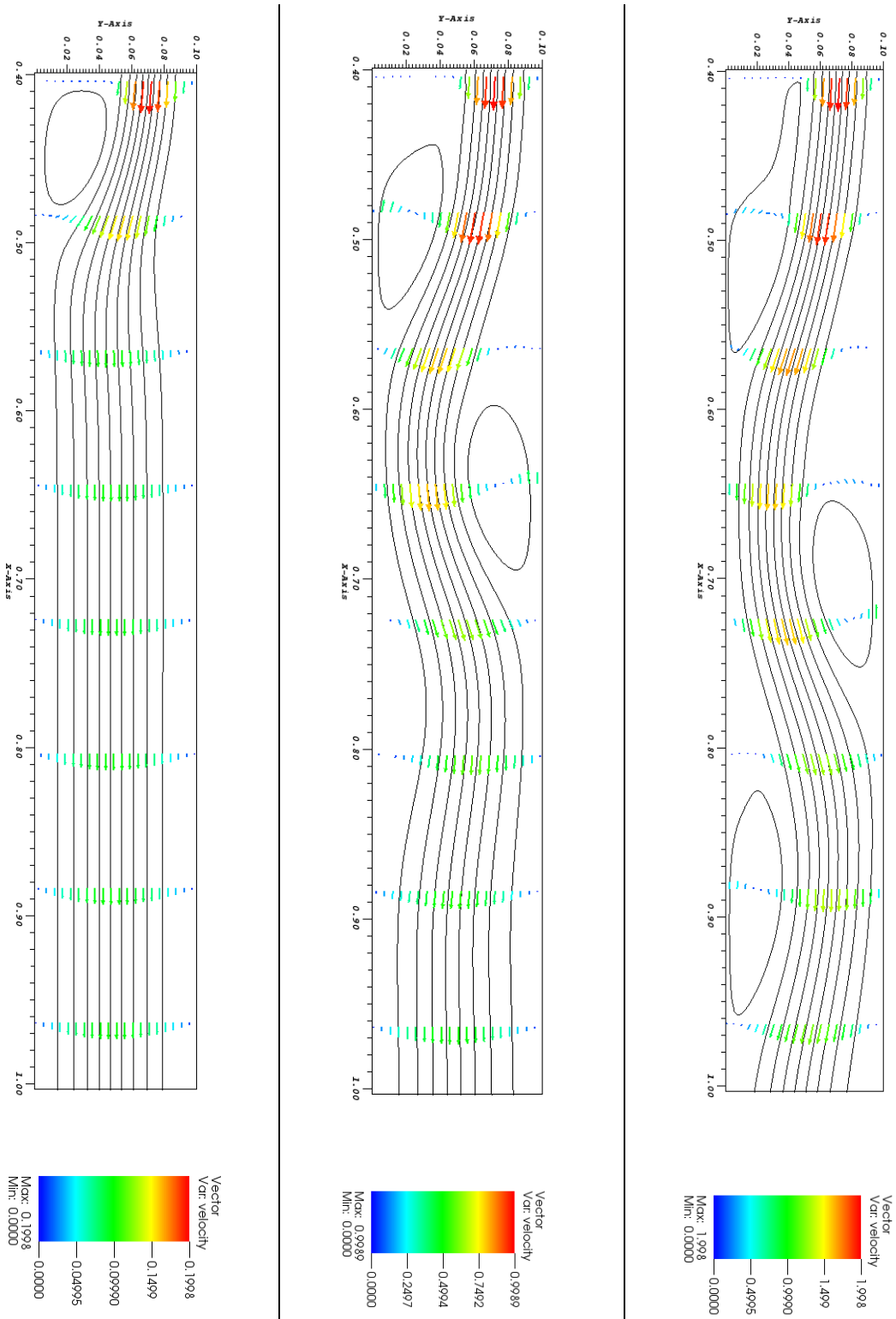
Re = 200	Re = 1000	Re = 2000

Figure 10: Streamline Contours and Stream Function Values



Re = 200	Re = 1000	Re = 2000

Figure 11: Velocity Vectors



Re = 200	Re=1000	Re=2000

Figure 12: Zoomed-In Vectors and Streamline Contours

```
      Program FlowOverBackwardFacingStep
c..Taha Yaşar Demir / 1881978
c..CE-580 / Homework #9
      parameter(mx=1001)
      common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
      common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >            NS,MS
      common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
      common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
     >      difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
     >      ,qxn(mx,mx),qxs(mx,mx)
      common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
     >            difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
     >      ,qyn(mx,mx),qys(mx,mx)
      common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
     >      Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
     >      Fn0(mx),Gm0(mx)
      common/Err/   E,res,tolerance

      open(11,file='mesh.tec',form="formatted")
      open(13,file="error.dat")
      open(14,file="Residual.dat")
      E = 1.
      tolerance = 1e-8
      call Initialize
      l = 1
c     do l=1,10000
      do while(E.gt.tolerance.and.l.lt.60000)
            call Boundary
            call Coefficients
            call Poisson
            call UpdateVelocities
            call Error(l)
            l = l+1
            if(mod(l,1000).eq.0) print*, l,E,res/(N*M)
      enddo
      print*, l
      call Output
      close(11)
      close(13)
      close(14)
      stop
      end
c-----------------------------------------------------------------------
      subroutine Initialize
      parameter(mx=1001)
      common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
      common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >            NS,MS
```

```
      rL  = 2.  !m /total length
      H   = 0.1 !m /total height
      SL  = 0.4 !m /step length
      SH  = 0.05!m /step height
      Um  = 2.  !m/s 0.2--1.0--2.0
      vis = 1e-4!m^2/s
      rho = 1e3 !km/m^3

      gamma = 1.8 ! Over-Relaxation Parameter

      call GenerateGrid

      dt = 5.e-6

      call InitialCondition

      return
      end
c-----------------------------------------------------------------------
      subroutine GenerateGrid
      parameter(mx=1001)
      common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
      common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >              NS,MS

      N  = 300 ! Cell count in x-direction
      M  = 60  ! Cell count in y-direction
      dx = 2.0/N
      dy = 0.1/M
      NS = ceiling(0.4/dx)  +1
      MS = ceiling(0.05/dy) +1

      do j = 1,M+1
            x(1,j) = 0.
            do i=1,N
                  x(i+1,j) = x(i,j) + dx
            enddo
            x(N+1,j) = rL
      enddo

      do i = 1,N+1
            y(i,1) = 0
            do j=1,M
                  y(i,j+1) = y(i,j) + dy
            enddo
            y(i,M+1) = H
      enddo

      do j = 1,M
            do i = 1,N
                  xc(i,j) = (x(i+1,j)+x(i,j))/2
                  yc(i,j) = (y(i,j+1)+y(i,j))/2
```

```
                enddo
        enddo

        a = (2/(dx**2)) + (2/(dy**2))


        return
        end

c-----------------------------------------------------------------------
        subroutine InitialCondition
        parameter(mx=1001)
        common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
        common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
   >             NS,MS
        common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
   >         Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
   >         Fn0(mx),Gm0(mx)
        common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
        real uniflow(mx)

        do i=1,N+2
                do j=1,M+2
                        u(i,j) = 0.
                        v(i,j) = 0.
                        p(i,j) = 0.
                enddo
        enddo

        do i=1,N+1
        if(i.lt.NS) then
        k=1
        do j = MS+1,M+1
        uniflow(k) = (-((0.1*Um/(0.025**2))*yc(1,k)**2)+
   +                ((0.2*Um/0.025)*yc(1,k)))/0.1
        u(i,j) = uniflow(k) ! wont updated in future iterations
        v(1,j) = 0. ! wont updated in future iterations
        k = k+1
    enddo
    else
    k=1
        do j = 1,M+1
        uniflow(k) = (-((0.05*Um/(0.05**2))*yc(1,k)**2)+
   +                ((0.1*Um/0.05)*yc(1,k)))/0.1
        u(i,j) = uniflow(k) ! wont updated in future iterations
        v(1,j) = 0. ! wont updated in future iterations
        k = k+1
    enddo
    endif
    enddo

    write(11,*) ' variables="x","y","u","v" '
```

```
          write(11,*) ' zone i=',N+1, 'j=',M+1
          do j = 1,M+1
          do i = 1,N+1
                    write(11,*) x(i,j),y(i,j),u(i,j),v(i,j)
          enddo
          enddo

          return
          end
c-----------------------------------------------------------------------
          subroutine Boundary
          parameter(mx=1001)
          common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >                  NS,MS
      common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
          common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
     >          difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
     >          ,qxn(mx,mx),qxs(mx,mx)
          common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
     >                  difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
     >          ,qyn(mx,mx),qys(mx,mx)
          common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
     >          Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
     >          Fn0(mx),Gm0(mx)

      ! Inflow Boundary Conditions
      do j = 1,M+2
          p(1,j) = p(2,j)
          F(1,j) = u(1,j)
      enddo
      ! Outflow Boundary Conditions
      do j = 1,M+1
          v(N+2,j) = 0.
c         Unp1(j) = u(N,j)
          u(N+1,j) = u(N,j)
          F(N+1,j) = u(N+1,j)
          p(N+2,j) = p(N+1,j)
      enddo
         u(N+1,M+2) = u(N,M+2)
          F(N+1,M+2) = u(N+1,M+2)
          p(N+2,M+2) = p(N+1,M+2)
      !! Wall Boundaries !!
      ! Horizaontal Part of the Step
      do i=1,NS-1
          v(i,MS)   = 0. ! on the wall
          u(i,MS)   =-u(i,MS+1) ! ghost cell
          G(i,MS)   = v(i,MS)
          p(i,MS)   = p(i,MS+1)
      enddo
          v(NS,MS)   = 0.
          G(NS,MS)   = v(NS,MS)
          p(NS,MS)   = p(NS,MS+1) !!!!!!!!!!!!!!!!!!!!!
```

```
    ! Vertical Part of the Step
    do j=1,MS-1
          v(NS,j)   =-v(NS+1,j) ! ghost cell
          u(NS,j)   = 0.  ! on the wall
          F(NS,j)   = u(NS,j)
          p(NS,j)   = p(NS+1,j)
    enddo
          p(NS,MS)   = p(NS+1,MS) !!!!!!!!!!!!!!!!!!!!
    ! Lower Wall
    do i=NS,N+2 ! maybe add v(i,1) = 0  and G(i,1) too
          v(i,1) = 0.
          u(i,1) =-u(i,2)
          G(i,1) = v(i,1)
          p(i,1) = p(i,2)
    enddo
    ! Upper Wall
    do i=1,N+2
          v(i,M+1) = 0.
          G(i,M+1) = v(i,M+1)
          p(i,M+2) = p(i,M+1)
    enddo
    do i=1,N+1
          u(i,M+2) =-u(i,M+1)
    enddo

          return
          end

c-----------------------------------------------------------------------
          subroutine Coefficients
          parameter(mx=1001)
          common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
          common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
    >                   NS,MS
          common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
          common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
    >           difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
    >           ,qxn(mx,mx),qxs(mx,mx)
          common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
    >                   difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
    >           ,qyn(mx,mx),qys(mx,mx)
     common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
    >           Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
    >           Fn0(mx),Gm0(mx)


          do i=1,N+2
                do j=1,M+2
                      gx(i,j) = 0.9
                      gy(i,j) = 0.9
                enddo
```

```
          enddo
! u-coefficients
     DO i =2,N
     IF(i.le.NS) THEN ! Before Step
          do j=MS+1,M+1
                    qxe(i,j) = 2*(u(i,j)+u(i+1,j))*dy
                    qxw(i,j) = 2*(u(i-1,j)+u(i,j))*dy
                    qxn(i,j) = 2*(v(i,j)+v(i+1,j))*dx
                    qxs(i,j) = 2*(v(i,j-1)+v(i+1,j-1))*dx
                    ue(i,j)  = (u(i,j)+u(i+1,j)+gx(i,j)*sign(1.,qxe(i,j))
    +                  *(u(i,j)-u(i+1,j)))/2.
                    un(i,j)  = (u(i,j)+u(i,j+1)+gx(i,j)*sign(1.,qxn(i,j))
    +                  *(u(i,j)-u(i,j+1)))/2.
          uw(i,j)  = (u(i-1,j)+u(i,j)+gx(i,j)*sign(1.,qxw(i,j))
    +            *(u(i-1,j)-u(i,j)))/2.
          us(i,j)  = (u(i,j-1)+u(i,j)+gx(i,j)*sign(1.,qxs(i,j))
    +            *(u(i,j-1)-u(i,j)))/2.
                    difu(i,j)= vis*( ( ((u(i+1,j)-u(i,j))/dx)
    +                        - ((u(i,j)-u(i-1,j))/dx) )*dy
    +                       +( ((u(i,j+1)-u(i,j))/dy)
    +                        - ((u(i,j)-u(i,j-1))/dy) )*dx )
                    conu(i,j)= ue(i,j)*qxe(i,j)-uw(i,j)*qxw(i,j)+
    +                   un(i,j)*qxn(i,j)-us(i,j)*qxs(i,j)
                    F(i,j)   = u(i,j) + (dt/(dx*dy))*(difu(i,j)-conu(i,j))
          enddo
     ELSE ! After Step
          do j=2,M+1
                    qxe(i,j) = 2*(u(i,j)+u(i+1,j))*dy
                    qxw(i,j) = 2*(u(i-1,j)+u(i,j))*dy
                    qxn(i,j) = 2*(v(i,j)+v(i+1,j))*dx
                    qxs(i,j) = 2*(v(i,j-1)+v(i+1,j-1))*dx
                    ue(i,j)  = (u(i,j)+u(i+1,j)+gx(i,j)*sign(1.,qxe(i,j))
    +                  *(u(i,j)-u(i+1,j)))/2.
                    un(i,j)  = (u(i,j)+u(i,j+1)+gx(i,j)*sign(1.,qxn(i,j))
    +                  *(u(i,j)-u(i,j+1)))/2.
                    uw(i,j)  = (u(i-1,j)+u(i,j)+gx(i,j)*sign(1.,qxw(i,j))
    +             *(u(i-1,j)-u(i,j)))/2.
                    us(i,j)  = (u(i,j-1)+u(i,j)+gx(i,j)*sign(1.,qxs(i,j))
    +             *(u(i,j-1)-u(i,j)))/2.
                    difu(i,j)= vis*( ( ((u(i+1,j)-u(i,j))/dx)
    +                        - ((u(i,j)-u(i-1,j))/dx) )*dy
    +                       +( ((u(i,j+1)-u(i,j))/dy)
    +                        - ((u(i,j)-u(i,j-1))/dy) )*dx )
                    conu(i,j)= ue(i,j)*qxe(i,j)-uw(i,j)*qxw(i,j)+
    +                   un(i,j)*qxn(i,j)-us(i,j)*qxs(i,j)
                    F(i,j)   = u(i,j) + (dt/(dx*dy))*(difu(i,j)-conu(i,j))
          enddo
     ENDIF
     ENDDO
! v-coefficients
          DO i=2,N+1
          IF (i.le.NS) THEN
```

```
                do j=MS+1,M ! Before Step
                qye(i,j) = 2*(u(i,j)+u(i,j+1))*dy
                qyw(i,j) = 2*(u(i-1,j)+u(i-1,j+1))*dy
                qyn(i,j) = 2*(v(i,j)+v(i,j+1))*dx
                qys(i,j) = 2*(v(i,j)+v(i,j-1))*dx
                ve(i,j)  = (v(i,j)+v(i+1,j)+gy(i,j)*sign(1.,qye(i,j))
     +             *(v(i,j)-v(i+1,j)))/2.
                vw(i,j)  = (v(i-1,j)+v(i,j)+gy(i,j)*sign(1.,qyw(i,j))
     +             *(v(i-1,j)-v(i,j)))/2.
                vn(i,j)  = (v(i,j)+v(i,j+1)+gy(i,j)*sign(1.,qyn(i,j))
     +             *(v(i,j)-v(i,j+1)))/2.
                vs(i,j)  = (v(i,j-1)+v(i,j)+gy(i,j)*sign(1.,qys(i,j))
     +             *(v(i,j-1)-v(i,j)))/2.
                difv(i,j)= vis*( ( ((v(i+1,j)-v(i,j))/dx)
     +                  - ((v(i,j)-v(i-1,j))/dx) )*dy
     +                 +( ((v(i,j+1)-v(i,j))/dy)
     +                  - ((v(i,j)-v(i,j-1))/dy) )*dx )
                conv(i,j)= ve(i,j)*qye(i,j)-vw(i,j)*qyw(i,j)+
     +              vn(i,j)*qyn(i,j)-vs(i,j)*qys(i,j)
                G(i,j)   = v(i,j) + (dt/(dx*dy))*(difv(i,j)-conv(i,j))
                enddo
        ELSE
                do j=2,M
                qye(i,j) = 2*(u(i,j)+u(i,j+1))*dy
                qyw(i,j) = 2*(u(i-1,j)+u(i-1,j+1))*dy
                qyn(i,j) = 2*(v(i,j)+v(i,j+1))*dx
                qys(i,j) = 2*(v(i,j)+v(i,j-1))*dx
                ve(i,j)  = (v(i,j)+v(i+1,j)+gy(i,j)*sign(1.,qye(i,j))
     +             *(v(i,j)-v(i+1,j)))/2.
                vw(i,j)  = (v(i-1,j)+v(i,j)+gy(i,j)*sign(1.,qyw(i,j))
     +             *(v(i-1,j)-v(i,j)))/2.
                vn(i,j)  = (v(i,j)+v(i,j+1)+gy(i,j)*sign(1.,qyn(i,j))
     +             *(v(i,j)-v(i,j+1)))/2.
                vs(i,j)  = (v(i,j-1)+v(i,j)+gy(i,j)*sign(1.,qys(i,j))
     +             *(v(i,j-1)-v(i,j)))/2.
                difv(i,j)= vis*( ( ((v(i+1,j)-v(i,j))/dx)
     +                  - ((v(i,j)-v(i-1,j))/dx) )*dy
     +                 +( ((v(i,j+1)-v(i,j))/dy)
     +                  - ((v(i,j)-v(i,j-1))/dy) )*dx )
                conv(i,j)= ve(i,j)*qye(i,j)-vw(i,j)*qyw(i,j)+
     +            vn(i,j)*qyn(i,j)-vs(i,j)*qys(i,j)
                G(i,j)   = v(i,j) + (dt/(dx*dy))*(difv(i,j)-conv(i,j))
                enddo
        ENDIF
        ENDDO

        return
        end

c-----------------------------------------------------------------------
        subroutine Poisson
        parameter(mx=1001)
```

```
      common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
      common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >              NS,MS
      common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
      common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
     >       difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
     >       ,qxn(mx,mx),qxs(mx,mx)
      common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
     >              difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
     >       ,qyn(mx,mx),qys(mx,mx)
      common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
     >       Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
     >       Fn0(mx),Gm0(mx)
      common/Err/   E,res,tolerance

      ! Source Term Calculation
      DO i=2,N+1
      IF (i.le.NS) THEN ! Before Step
            do j=MS,M+1
            S(i,j) = rho*((F(i,j)-F(i-1,j))/dx+(G(i,j)-G(i,j-1))/dy)/dt
            enddo
      ELSE ! After Step
            do j=2,M+1
            S(i,j) = rho*((F(i,j)-F(i-1,j))/dx+(G(i,j)-G(i,j-1))/dy)/dt
            enddo
      ENDIF
      ENDDO
      res = 0.
      ! Pressure Update
      do k=1,30

c           res = 0. ! for determination of gamma

      DO i=2,N+1
      IF (i.le.NS) THEN
            do j=MS+1,M+1
           p(i,j) = p(i,j)+ gamma*((((((p(i+1,j)+p(i-1,j)))/(dx**2) +
     +           (p(i,j+1)+p(i,j-1))/(dy**2))-S(i,j)))/a - p(i,j))
           res = res + abs((((((p(i+1,j)+p(i-1,j)))/(dx**2) +
     +           (p(i,j+1)+p(i,j-1))/(dy**2))-S(i,j)))/a - p(i,j)))
            enddo
      ELSE
            do j=2,M+1
           p(i,j) = p(i,j)+ gamma*((((((p(i+1,j)+p(i-1,j)))/dx**2 +
     +           (p(i,j+1)+p(i,j-1))/dy**2)-S(i,j)))/a - p(i,j))
           res = res + abs((((((p(i+1,j)+p(i-1,j)))/(dx**2) +
     +           (p(i,j+1)+p(i,j-1))/(dy**2))-S(i,j)))/a - p(i,j)))
            enddo
      ENDIF
      ENDDO
c     if(E.eq.1.) write(14,*) k,res/(N*M) ! for determination of gamma
      enddo
```

```
      return
      end


c-----------------------------------------------------------------------
            subroutine UpdateVelocities
            parameter(mx=1001)
            common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
            common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >                    NS,MS
            common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
            common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
     >           difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
     >           ,qxn(mx,mx),qxs(mx,mx)
            common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
     >                    difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
     >           ,qyn(mx,mx),qys(mx,mx)
      common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
     >           Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
     >           Fn0(mx),Gm0(mx)

! u-component
      DO i=2,N
      IF(i.le.NS) THEN
            do j=MS+1,M+1
            u(i,j) = F(i,j) + dt*(p(i,j)-p(i+1,j))/(rho*dx)
            enddo
      ELSE
            do j=2,M+1
            u(i,j) = F(i,j) + dt*(p(i,j)-p(i+1,j))/(rho*dx)
            enddo
      ENDIF
      ENDDO
! v-component
      DO i=2,N+1
      IF(i.lt.NS) THEN
            do j=MS+1,M
            v(i,j) = G(i,j) + dt*(p(i,j)-p(i,j+1))/(rho*dy)
            enddo
      ELSE
            do j=2,M
            v(i,j) = G(i,j) + dt*(p(i,j)-p(i,j+1))/(rho*dy)
            enddo
      ENDIF
      ENDDO

      return
      end
c-----------------------------------------------------------------------
            subroutine Output
```

```
      parameter(mx=1001)
      common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
      common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >              NS,MS
      common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
      common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
     >      difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
     >      ,qxn(mx,mx),qxs(mx,mx)
      common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
     >              difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
     >      ,qyn(mx,mx),qys(mx,mx)
      common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
     >      Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
     >      Fn0(mx),Gm0(mx)
      real disc_in,disc_out,u_out(mx,mx),v_out(mx,mx)
      real stream(mx,mx)


      do i=2,N+1
      do j=2,M+1
            u_out(i,j) = (u(i,j)+u(i-1,j))/2
            v_out(i,j) = (v(i,j)+v(i-1,j))/2
      enddo
      enddo

      do i=2,N+1
            stream(i,1) = 0.
            stream(i,2) = -u_out(i,j)*dy/2 + stream(i,1)
      enddo

      do i=2,N+1
      do j=2,M
            stream(i,j+1) = -u_out(i,j+1)*dy + stream(i,j)
      enddo
      enddo


      open(12,file="Velocities.tec",form='formatted')
      write(12,*) ' variables="x","y","u","v","p","stream" '
      write(12,*) 'zone i=',N, ' j=',M
      do j=2,M+1
      do i=2,N+1
      if((i.le.NS.and.j.le.MS).or.j.eq.1) then
      write(12,'(8E12.4)') xc(i-1,j-1),yc(i-1,j-1),0.,0.,p(i,j),0.
      else
      write(12,'(8E12.4)') xc(i-1,j-1),yc(i-1,j-1),
     +                     u_out(i,j),v_out(i,j),p(i,j),stream(i,j)
      endif
      enddo
      enddo
      close(12)
      disc_in=0.
```

```
      disc_out=0.

      do j=MS+1,M+1
            disc_in  = disc_in+ u(1,j)*dy
      enddo
      do j=2,M+1
            disc_out = disc_out + u(N,j)*dy
      enddo
      print*, "Discharge In", disc_in, "Discharge Out", disc_out
      print*, "Percent Difference" ,abs(100*(disc_out-disc_in)/disc_in)



            return
            end
c-----------------------------------------------------------------------
            subroutine Error(iter)
            parameter(mx=1001)
            common/init/  rL,H,SL,SH,Um,vis,rho,gamma,dt
            common/grid/  N,M,dx,dy,x(mx,mx),xc(mx,mx),y(mx,mx),yc(mx,mx),a,
     >                    NS,MS
            common/flow/  u(mx,mx),v(mx,mx),p(mx,mx),S(mx,mx)
            common/xcoef/ ue(mx,mx),uw(mx,mx),un(mx,mx),us(mx,mx),conu(mx,mx),
     >            difu(mx,mx),F(mx,mx),gx(mx,mx),qxe(mx,mx),qxw(mx,mx)
     >            ,qxn(mx,mx),qxs(mx,mx)
            common/ycoef/ ve(mx,mx),vw(mx,mx),vn(mx,mx),vs(mx,mx),conv(mx,mx),
     >                    difv(mx,mx),G(mx,mx),gy(mx,mx),qye(mx,mx),qyw(mx,mx)
     >            ,qyn(mx,mx),qys(mx,mx)
      common/bndry/ Un0(mx),Vn0(mx),Unp1(mx),Vnp1(mx),Pn0(mx),Pnp1(mx),
     >            Um0(mx),Vm0(mx),Ump1(mx),Vmp1(mx),Pm0(mx),Pmp1(mx),
     >            Fn0(mx),Gm0(mx)
            common/Err/   E,res,tolerance
      real u_old(mx,mx),v_old(mx,mx), sum


      if(iter.eq.1) then
            sum = N*M
            do i=2,N
            do j=2,M
                  u_old(i,j) = u(i,j)
                  v_old(i,j) = v(i,j)
            enddo
            enddo
      else
            sum = 0.
            do i=2,N
            do j=2,M
                  sum = sum + abs(u_old(i,j)-u(i,j))
                  sum = sum + abs(v_old(i,j)-v(i,j))
                  u_old(i,j) = u(i,j)
                  v_old(i,j) = v(i,j)
            enddo
```

```
            enddo
        endif
        E = sum/(N*M*Um)
        write(13,*) iter, E , res/(N*M)
c       print*, 'velocity Res', E



            return
            end

c-------------------------------------------------------------------
```

Fortran Source Code