MAE 6225 Computational Fluid Dynamics: Project 2

Shantanu Bailoor
GW ID: G44942912
shantanub@gwu.edu

Abstract

In this report, the development of a two-dimensional transient Navier-Stokes solver is described. A second-order central difference scheme over a staggered grid was used for spatial discretization. Time integration was achieved through a three stage Runge-Kutta scheme and a projection method, with predictor-corrector steps, was used for time advancement of solution. A previously developed SOR-based Poisson solver was incorporated to obtain solutions to the pressure equation. A ghost-cell based sharp interface, immersed boundary method was used to simulate the interaction between fluid and immersed, rigid bodies. The solver was tested for spatial and temporal accuracy against analytical solutions to the Taylor-Green vortex problem. The mean spatial and temporal orders of accuracy were found to be 1.82 and 2.725, respectively. Further, solution to shear driven cavity flow problem at Reynolds number 100 was obtained and results were compared with established data. The solver was able to sufficiently capture flow characteristics pertaining to primary and secondary vortices and match velocities to within 3% of published results. Finally, flow over a stationary, circular cylinder was simulated and periodic vortex shedding in the wake of the cylinder was observed.

1. Details of solver

1.1. Spatial Discretization:

In developing the current solver, a second-order central difference scheme (CDS) was employed for spatial discretization over a staggered grid. A staggered grid implies that different flow variables (u, v, p) were not located at the same grid points, but were, instead, distributed at different locations in every cell. This, effectively, led to three different grids – u-grid, v-grid and p-grid. Such a distribution of flow variables was implemented to avoid pressure-velocity decoupling, which is described in section 1.1.3. An extra layer of ghost cells is added around the grid for consistent and efficient boundary condition implementation. A sample, uniform grid of 5x5 interior cells, with the additional ghost layer is shown in Figure 1. A cell-centered approach was used, in that, the flow variables were computed at centers of their respective cells. Using a staggered grid also facilitates convenient positioning of flow variables with respect to each other for ease in computing fluxes across cell boundaries.

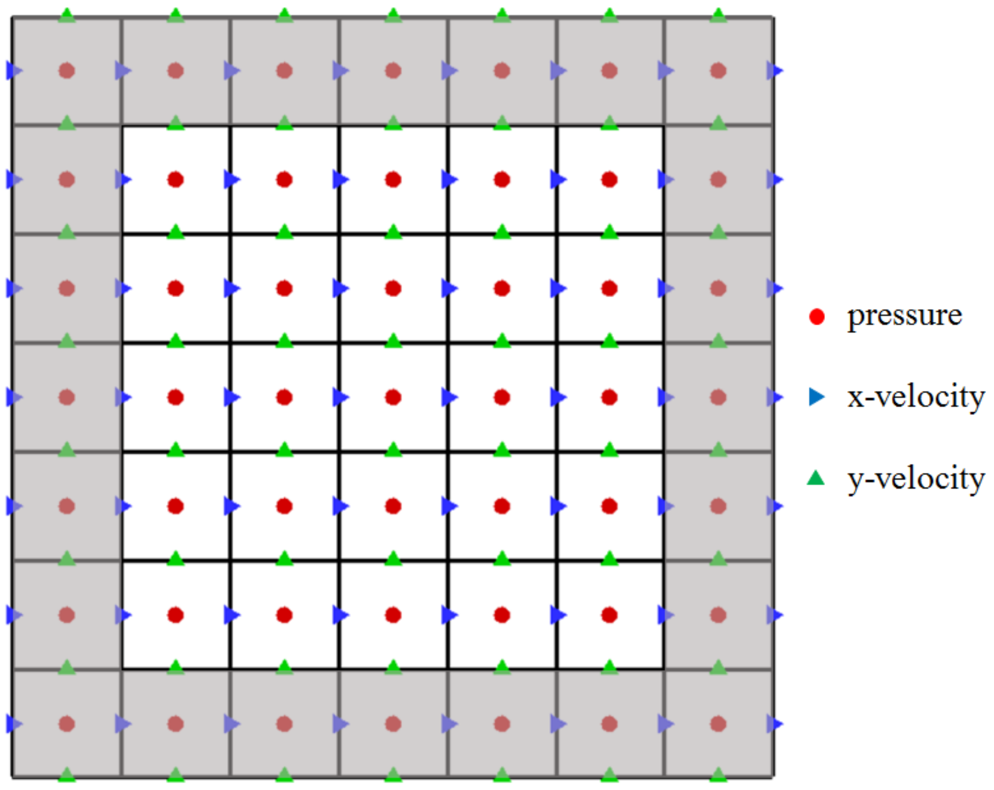

Figure 1: Sample uniform grid consisting of 5x5 cells, indicating the distribution of pressure and velocity points. An additional layer of ghost cells (shaded) is included for boundary treatment.

1.1.1. *Discretizing the x-momentum equation:*

Solutions to the *x*-momentum equation yield values for u, thus, the equation was solved on the *u*-grid, a part of which is shown in Figure 2. Consider the *x*-momentum equation for the incompressible Navier-Stokes equations:

$$u_t = F_c + F_v + F_p \tag{1}$$

Where u_t represents the time derivative of u, F_c, F_v and F_p denote the convective, diffusive and pressure fluxes, respectively, into a finite control volume of the *u*-cell. In discrete form, the fluxes are expressed as:

$$F_c = -(u_e.u_e - u_w.u_w + u_n.v_n - u_s.v_s) \tag{2}$$

$$F_v = \nu\left[\frac{1}{\Delta x}\left(\left.\frac{\partial u}{\partial x}\right|_e - \left.\frac{\partial u}{\partial x}\right|_w\right) + \frac{1}{\Delta y}\left(\left.\frac{\partial u}{\partial y}\right|_n - \left.\frac{\partial u}{\partial y}\right|_s\right)\right] \tag{3}$$

$$F_p = -\frac{1}{\rho}\left(\frac{p_e - p_w}{\Delta x}\right) \tag{4}$$

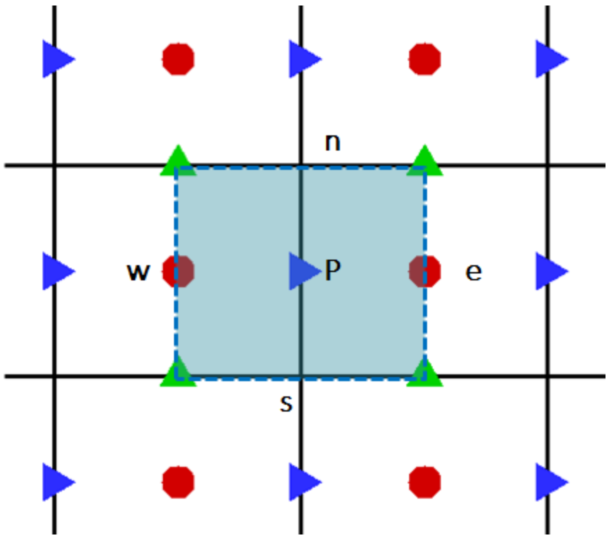

Figure 2: A typical *u*-grid cell, with *p*-nodes at face centers and *v*-nodes at corners.

The lowercase subscripts indicate the cell boundaries at which the quantities were computed. Thus, for a given cell, *P*, the values for *y*-velocity on its north and south faces and for pressure on the east and west faces were needed. The staggered arrangement of variables facilitated these computations by placing the *y*-velocity nodes at the corners of the *u*-cell and pressure nodes at the east and west face centers. The values of *x*-velocity at the face centers were computed through linear interpolation between neighboring cell-centers and values of *y*-velocity were obtained through interpolation between those at the cell corners.

1.1.2. *Discretizing the y-momentum equation:*

A similar analysis as done in section 1.1.1, was performed for the *y*-momentum equation, of the form:

$$v_t = F_c + F_v + F_p \tag{5}$$

The discrete form of the equation requires computing fluxes as follows:

$$F_c = -(v_e.u_e - v_w.u_w + v_n.v_n - v_s.v_s) \tag{6}$$

$$F_v = \nu\left[\frac{1}{\Delta x}\left(\left.\frac{\partial v}{\partial x}\right|_e - \left.\frac{\partial v}{\partial x}\right|_w\right) + \frac{1}{\Delta y}\left(\left.\frac{\partial v}{\partial y}\right|_n - \left.\frac{\partial v}{\partial y}\right|_s\right)\right] \tag{7}$$

$$F_p = -\frac{1}{\rho}\left(\frac{p_n - p_s}{\Delta y}\right) \tag{8}$$

As for the x-momentum equation, using a staggered grid (Figure 3) resulted in convenient positioning of the *u*-nodes and *p*-nodes for flux computations. The *y*-velocities at face centers were obtained through linear interpolation of *y*-velocity between respective cell-centers, while the *x*-velocities were obtained through interpolation of velocity at cell corners.

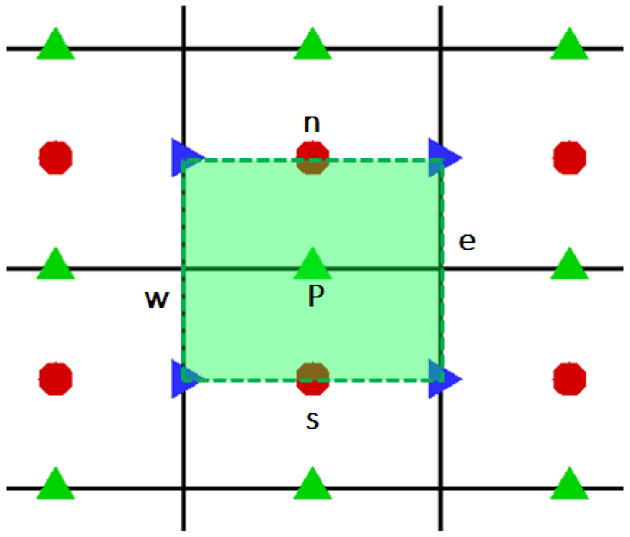

Figure 3: A typical *v*-grid cell, with *p*-nodes at face centers and *u*-nodes at corners.

In both the momentum equations, computing pressure fluxes using CDS was fairly straightforward. Since pressure and velocities were not defined at the same locations, every *p*-node was accounted for. Using the same discretization on a collocated grid could lead to the pressure at a point being ignored for the momentum flux computation at the same point. This causes a decoupling of pressure and velocity at that point, resulting in spurious, non-physical oscillations of flow variables throughout the domain.

1.1.3. *Discretizing the Poisson equation:*

Using a fractional time-stepping method led to a pressure Poisson equation, which was solved to obtain solution to pressure at the *(n+1)th* time-step. The Poisson equation was of the form:

$$\nabla^2 P^{n+1} = \frac{1}{\Delta t}\left(\frac{\partial u^n}{\partial x} + \frac{\partial v^n}{\partial y}\right) \tag{9}$$

The expression enclosed within parenthesis on the right side of the above equation represents the divergence of velocity at the pressure-cell center. The Laplacian of pressure was discretized leading to a linear system of equations, of the form:

$$a_P P_P + a_E P_E + a_W P_W + a_N P_N + a_S P_S = f_P \tag{10}$$

Where subscripts of the quantities represent the position of the point, relative to a particular *p*-cell-center, at which the quantity is computed and f_P is the divergence of velocity in the *p*-cell. The linear system was solved using a previously developed 2D-Poisson solver [1], which used a successive over-relaxation (SOR) scheme for accelerating convergence.

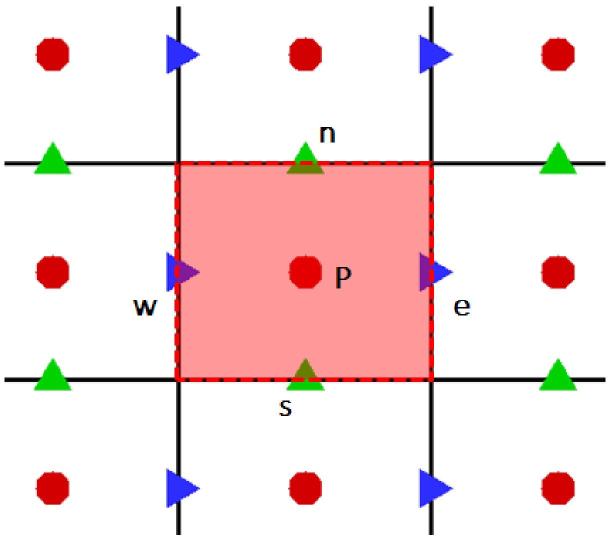

Figure 4: A typical p-grid cell, with u-nodes at vertical face centers and v-nodes at horizontal face-centers.

The staggered arrangement of flow variables over *p*-grid (Figure 4) facilitates computing gradients of *u* along the horizontal and of *v* along the vertical, by placing the *u*-nodes at the vertical face (east and west) centers and *v*-nodes at the horizontal face (north and south) centers. Such an arrangement of variables also permits a direct use of CDS for computing velocity divergence in the pressure equation and pressure fluxes in the momentum equations.

1.2. Time integration:

A fractional time-stepping method was used for predicting the transient response of the fluid. The time advancement was decomposed into two fractional steps. In the first pseudo-time step, for each momentum equation, a velocity prediction for the $n+1$*th* time step was made while taking into account only the convective and viscous fluxes. Thus,

$$u^{*n} = u^n + \Delta t(F_c^n + F_v^n) \tag{11}$$

This velocity prediction neither satisfied the corresponding momentum equation, due to the exclusion of the pressure flux, nor was divergence-free. Therefore, it required suitable correction, which was obtained by solving a pressure Poisson equation. The pressure Poisson equation was constructed by considering the divergence of the momentum equation and forcing the divergence-free constraint on the velocity. When this condition is satisfied, the pressure at the *(n+1)th* time-step can be calculated as:

$$\nabla^2 P^{n+1} = \frac{1}{\Delta t}\left(\frac{\partial u^{*n}}{\partial x} + \frac{\partial v^{*n}}{\partial y}\right) \tag{12}$$

Finally, corrected components of velocity were computed by adding the pressure flux, using the values at the *(n+1)th* time-step, to the respective components of the predicted velocities. Thus:

$$\begin{aligned} u^{n+1} &= u^{*n} + \Delta t\left(F_{px}^{n+1}\right) \\ v^{n+1} &= v^{*n} + \Delta t(F_{py}^{n+1}) \end{aligned} \tag{13}$$

The time advancement was used in conjunction with an explicit, three-stage Runge-Kutta time discretization which led to solving the above equations in three steps, each with a fraction of the chosen value of Δt. The three sub-iterations can be summarized as follows:

- Stage 1 (From $t \rightarrow t+(\Delta t/3)$):

$$G_i = F_{ci}^n + F_{vi}^n \tag{14}$$

$$v_i = u_i{}^n + \frac{\Delta t}{3} G_i \tag{15}$$

$$\nabla^2 P^* = \frac{1}{\Delta t/3}\left(\frac{\partial v_i}{\partial x_i}\right) \tag{16}$$

$$u_i{}^* = v_i + \frac{\Delta t}{3}\left(F_{pi}^*\right) \tag{17}$$

- Stage 2 (From *t+(*Δ*t /3)* → *t+(3*Δ*t /4)*):

$$G_i = \frac{-5}{9} G_i + F_{ci}^{*} + F_{vi}^{*} \tag{18}$$

$$v_i = {u_i}^* + \frac{15}{16} \Delta t G_i \tag{19}$$

$$\nabla^2 P^{**} = \frac{1}{5\Delta t/12} \left(\frac{\partial v_i}{\partial x_i} \right) \tag{20}$$

$$ {u_i}^{**} = v_i + \frac{5\Delta t}{12} \left(F_{pi}^{**} \right) \tag{21}$$

- Stage 3 (From *t+(3*Δ*t /4)* → *t*+Δ*t*):

$$G_i = \frac{-153}{128} G_i + F_{ci}^{**} + F_{vi}^{**} \tag{22}$$

$$v_i = {u_i}^{**} + \frac{8}{15} \Delta t G_i \tag{23}$$

$$\nabla^2 P^{n+1} = \frac{1}{\Delta t/4} \left(\frac{\partial v_i}{\partial x_i} \right) \tag{24}$$

$$ {u_i}^{n+1} = v_i + \frac{\Delta t}{4} \left(F_{pi}^{n+1} \right) \tag{25}$$

In equations (12)-(23), tensor notation is used to denote velocities, thus, u_i represents components of velocity and v_i the components of the predicted velocity. The details of this method can be found in [2].

1.3. Immersed boundary method

A ghost-cell based, sharp interface, immersed boundary method was used over a body non-conforming, Cartesian grid, to model fluid-solid interaction. Cartesian grid methods have recently gained popularity in fluid-structure interaction modeling due to the simplicity of grid generation, discretization of governing equations and boundary treatment. The fluid domain is represented by a set of Eulerian grid points and the solid using unstructured, Lagrangian marker points. Boundary conditions may be implemented in several ways. The easiest way would be to simply ignore the distance between the Lagrangian marker points and the closest Eulerian grid points. No interpolation is needed and boundary conditions may be directly applied at the Eulerian grid points. Doing so, results in a stepwise description of the immersed boundary and greater diffusion of the interface, especially, if a staggered grid is used.

An improvement to this method would be to use a volume-of-fluid (VOF) approach (Fadlun et al. [3]), in which velocities at Eulerian cells containing the immersed boundary are assumed to be a fraction of the resulting velocity, in the absence of an immersed boundary. This fraction is equal to the ratio of volume of the cell occupied by fluid to the total volume of the cell. Thus, for a cell that is 75% occupied by solid, the velocity assigned to that cell is 25% of the resulting velocity in the absence of any solid. VOF methods are easy to implement, stable and known to not result in spurious mass conservation errors. However, this, too, may not result in a sufficiently sharp characterization of the immersed boundary, as computing volume fractions may entail significant geometrical approximations, which become particularly significant on three-dimensional grids while describing complex geometries. Further, this method employs interpolation for flow variables but may not result in an accurate reconstruction of the immersed body.

A better delineation of the immersed surface can be achieved by interpolating values of velocities and pressure at near-boundary points. One way of doing so is linearly interpolating flow variables at the interface along a preferred direction (*x*- or *y*-), as described by Fadlun et al. [3]. This is a simple, second-order accurate approach and requires no complicated geometrical constructions. However, doing so not only ignores gradients along the other direction, but is also physically inaccurate as fluxes should be ideally computed normal to the boundary surface. This can be avoided by incorporating multidimensional interpolation, along normal to the boundary, as done by Balaras [4]. In this method, points adjacent to, and outside, the boundary are chosen for direct forcing such that boundary conditions at the interface are effectively satisfied. However, imposing Neumann boundary conditions using this approach becomes non-trivial and, indeed, complicated. Mittal et al. [5] proposed a similar sharp-interface method using a layer of ghost cells immediately adjacent to the immersed boundary and, instead, inside the solid body. Boundary conditions are applied at these ghost cells, such that requirements on velocities and pressure at the interface are met. By having the site for interpolation inside the body, describing normal gradients across the boundary is relatively easy. The method requires some additional bilinear interpolation, but affords trivial boundary condition formulation and a reasonably good boundary reconstruction.

The method used in the present study is closest to the one proposed by Mittal et al. [5] and is described as follows. The boundary, Γ, was described by a set of Lagrangian points. A surface can, thus, be described, in two-dimensions, as a series of piecewise linear segments. In three-dimensions, surfaces may be effectively tessellated using triangular elements. The cells of the Cartesian grid, referred to as Eulerian points, were tagged as solid or fluid based on their relative positions with respect to the solid boundary. A generalized search algorithm was employed such that cells with their centers enclosed by the boundary were treated as solid. This was achieved by considering the inner product of the normalized vector obtained by joining the Eulerian point under consideration with the center of the closest Lagrangian element with the unit normal vector of the Lagrangian element. The cell lies within the Lagrangian boundary if the inner product is positive and outside if the product is negative. The search algorithm is illustrated in Figure 5 (a).

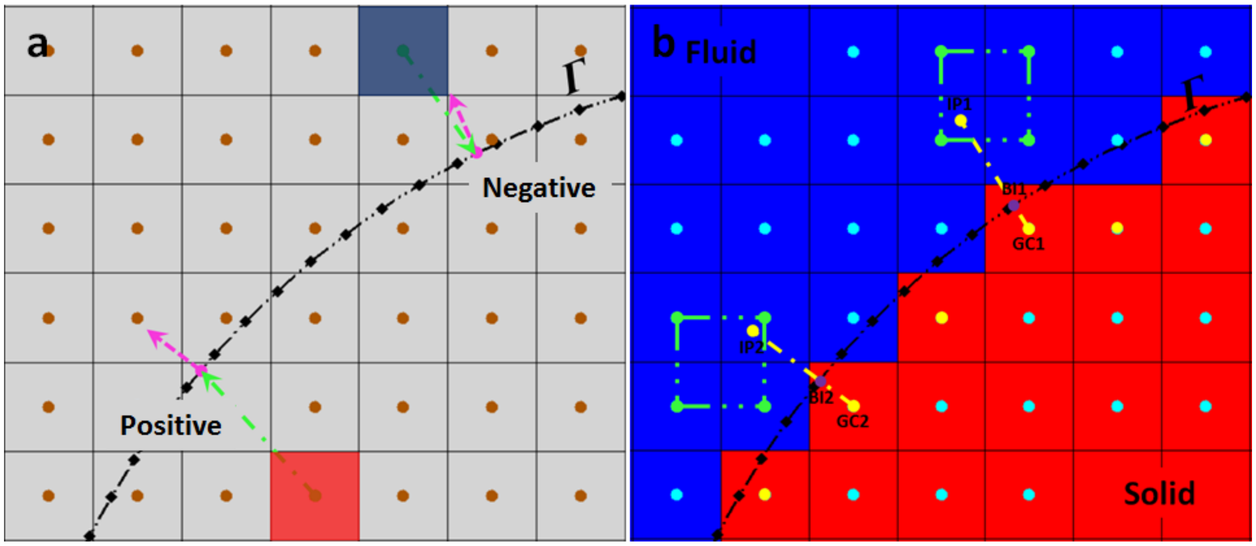

Figure 5: (a) Search algorithm to identify fluid and solid cells. The Eulerian cells are indicated with circles (●) and Lagrangian elements with diamonds (♦). The green arrows represent vectors drawn from Eulerian cell-centers to Lagrangian element-centers while the pink arrows represent unit normal vectors of the respective Lagrangian elements. Inner product is positive when the angle between the two vectors is less than 90 degrees and is otherwise negative. (b) Schematic of the ghost-cell based immersed boundary method, indicating the domain bifurcated into a fluid phase (shaded blue) and a solid phase (shaded red). The solid cells with yellow centers represent ghost cells. The group of four fluid cells with green centers enclose the projected image point (IP) and are used as basis for bilinear interpolation.

The upper highlighted cell projected a vector such that the inner product with the normal to the closest Lagrangian element was negative, and was therefore marked as a fluid cell (shaded blue). The lower highlighted cell projected a vector at an acute angle with the normal to the closest Lagrangian element. This cell was therefore marked as a solid cell (shaded red). Further, solid cells with at least one fluid neighbor were tagged as "ghost cells" (Figure 5 (b)). This method can be extended to non-analytic curves and is especially suited for convex curvatures.

Once the ghost cells were identified, their respective boundary intercepts (BI) and image points (IP) were computed. A normal was projected from each ghost cell to the boundary, *Γ*. The intersection of the normal and the boundary was called the boundary intercept. The probe was further extended by a suitable distance such that the resulting point was surrounded by four fluid nodes. This point was called the image point. Image points were constructed along normal from ghost cells for convenient boundary condition implementation. The values of flow variables (*u*,*v*,*p*) at the image point were interpolated using those at the four fluid nodes surrounding the image point. To do this, the positions of the image point and the four fluid nodes were mapped to a standard quadrilateral element (Figure 6) in (ξ, η) coordinates. The value of a variable, φ, at the corresponding position of image point is given by:

$$\varphi_{IP'} = N_1\varphi_1 + N_2\varphi_2 + N_3\varphi_3 + N_4\varphi_4 \tag{26}$$

Where $N_{1,2,3,4}$ represent the Lagrangian shape functions for the standard element and $\varphi_{1,2,3,4}$ represent the values of φ at the four corners of the standard quadrilateral. And

$$\begin{aligned} N_1 &= (1-\xi)(1-\eta) \\ N_2 &= \xi(1-\eta) \\ N_3 &= \xi\eta \\ N_4 &= (1-\xi)\eta \end{aligned} \tag{27}$$

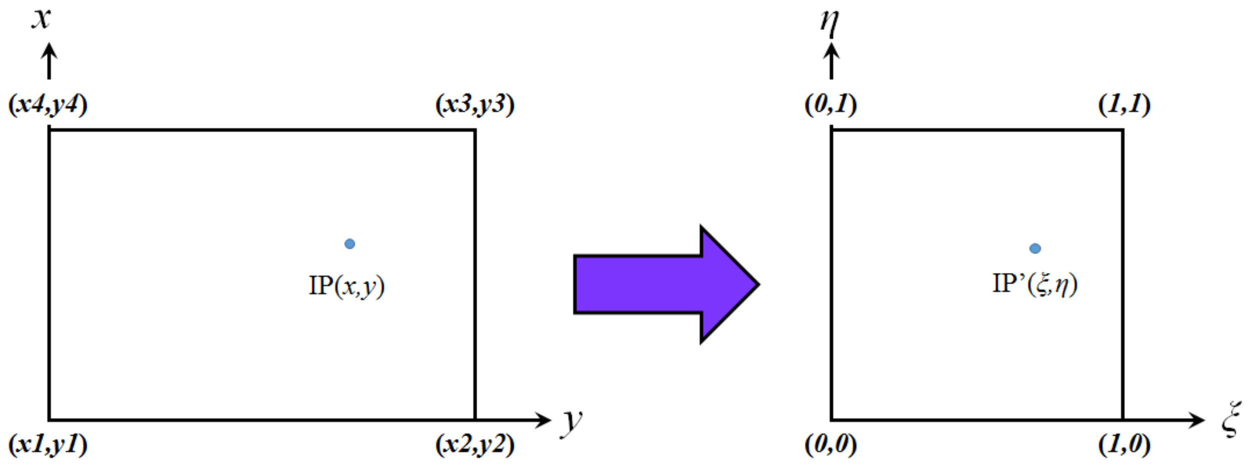

Figure 6: Isoparametric mapping to a standard quadrilateral element for bilinear interpolation.

Using the values at the image points, boundary conditions at the immersed boundary were enforced as:

1. Dirichlet:

$$\varphi_{GC} = \frac{(d_{BG} + d_{IB})\varphi_{BI} - d_{BG}\varphi_{IP}}{d_{IB}} \tag{28}$$

Where subscripts *GC*, *BI* and *IP* represent flow variable, φ, at the ghost cell, boundary intercept and image point, respectively and distances d_{BG} and d_{BI} represent the distance between the boundary intercept and ghost point and that between the image point and boundary intercept, respectively.

2. Neumann:

$$\varphi_{GC} = \varphi_{IP} - (d_{BG} + d_{IB})\varphi_{BI} \tag{29}$$

All symbols here have the same meaning as above, except φ_{BI} represents the derivative of φ along the normal. It is acknowledged that when the distances d_{BG} and d_{IP} are not equal, the above expression is not an accurate representation for the normal gradient at the boundary, and hence, obtaining a grid such that the two distances are as close to each other as possible must be striven towards.

These boundary conditions are implemented at every iteration (or sub-iteration, in case of multi-stage time integration), along with those at the domain bounds.

This method, coupled with non-uniform, Cartesian grids (Figure 7) can be a very effective and relatively inexpensive method, for modeling fluid-structure interaction. Using a non-uniform, stretched grid [6] in regions of lower interest (far-fields) lets one use fewer points where resolution is not critical, leading to faster computation. However, it is important to maintain a refined, uniform grid, with an aspect ratio close to 1, in the vicinity of the immersed boundary to accurately represent the boundary and to avoid preferential error propagation along a particular direction.

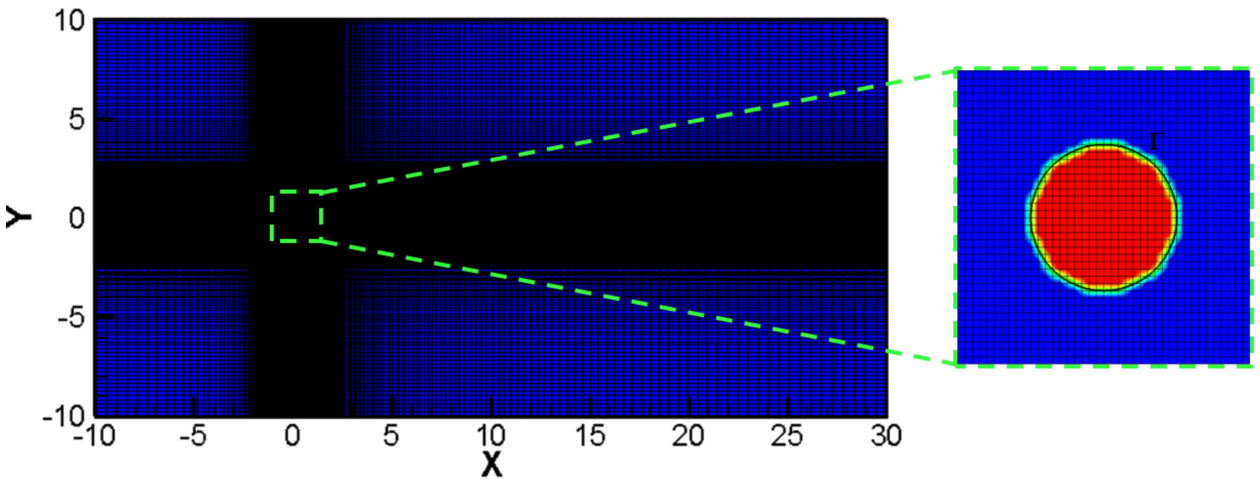

Figure 7: Immersed boundary method implementation on a sample stretched, non-uniform Cartesian grid showing a clustering of grid points in the vicinity of the immersed boundary and larger, skewed grids in the far field. (Inset) A uniform Cartesian grid is used around the immersed boundary, *Γ*, to accurately demarcate the fluid (blue) and solid (red) phases.

2. Approximations to the Taylor-Green vortex:

2.1. Problem Description:

The Taylor-Green vortex (TGV) is a classic flow problem with a closed-form exact solution to the unsteady, incompressible Navier-Stokes equations. Thus, the problem becomes a common benchmark case for CFD solvers. The Taylor-Green vortex describes spatially-periodic vortices with pressure and velocity intensities that decay exponentially with time. In two dimension, the problem may be described by equations (2)-(4):

$$u(x, y, t) = -e^{-2t}\cos x \sin y \tag{30}$$

$$v(x, y, t) = e^{-2t}\sin x \cos y \tag{31}$$

$$p(x, y, t) = \frac{-e^{-4t}}{4}(\cos 2x + \cos 2y) \tag{32}$$

Between the interval [0, 2π], the velocity and pressure fields, at *t* = 0, are as shown in Figure 8.

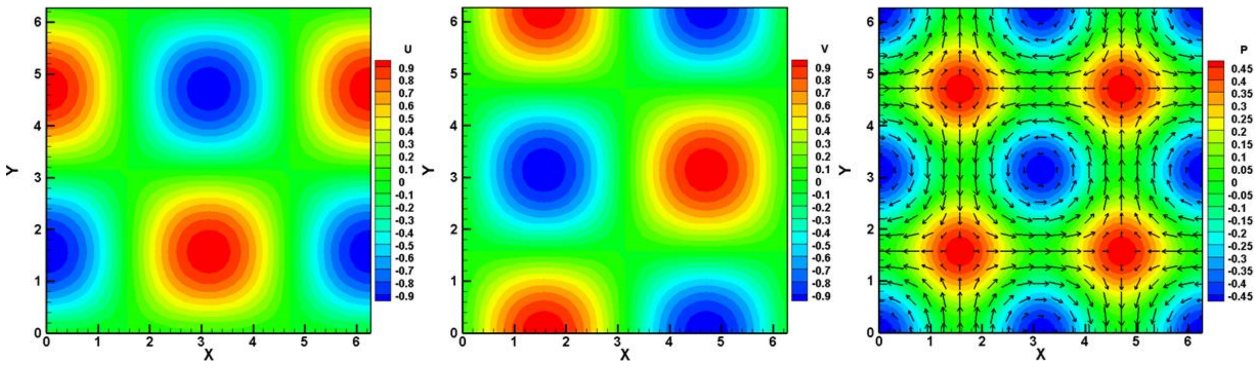

Figure 8: Initial conditions for (a) *x*-velocity, (b) *y*-velocity and (c) pressure, along with flow vectors, for the Taylor-Green vortex problem.

The velocities and pressure at every point in the domain decay exponentially with time and asymptotically approach zero. The decay of the velocity and pressure amplitudes is shown in Figure 9. The flow variables are symmetric about zero and thus, the signs of the amplitudes shown are not of any significance.

To model this problem, periodic boundary conditions for velocity and pressure were enforced at each domain face. Knowing that the size of the domain in each direction is equal to (or an integral multiple of) the period of variation of flow parameters, the periodic BC is enforced by equating the quantity at the ghost layer to that at the near-boundary layer at the opposite end of the domain.

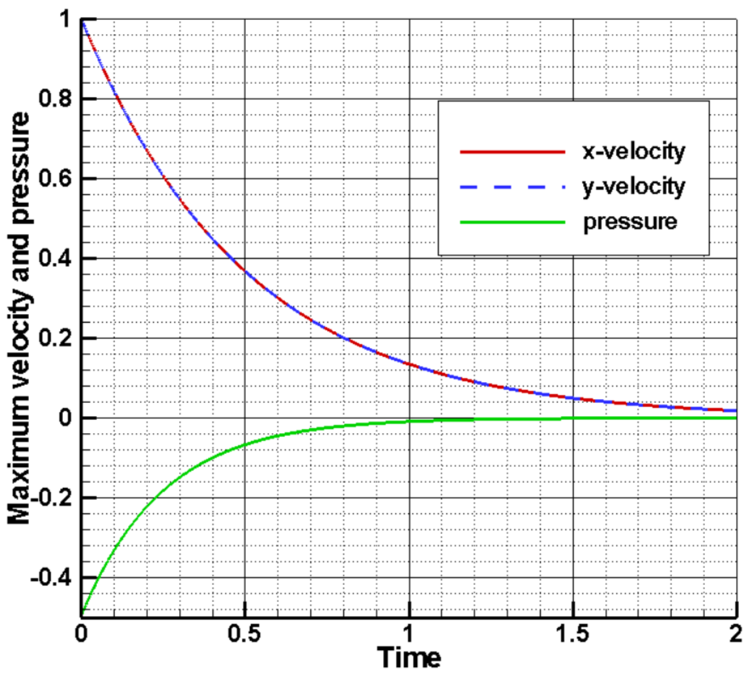

Figure 9: Solutions to the amplitudes of x-velocity, y-velocity and pressure in the Taylor-Green vortex problem. The flow quantities are symmetric and their amplitudes are equal on either side of zero.

2.2. Grid refinement study:

In order to test the spatial order of accuracy of the solver, the above TGV problem was simulated using grids of increasing spatial refinement, using the same time-step, Δt = 1*e*-3. Four uniform grids with 20^2, 30^2, 40^2 and 50^2 cells were used and the variation of global error norm (L_2) for the Poisson equation with cell count (N) was plotted, as shown in Figure 10. The errors were compared at time, t = 0.25. This instant was not selected arbitrarily but such that the flow variables were not very close to the initial conditions and, at the same time, had not dissipated too close to zero. Choosing an instant near the initial conditions could lead to naturally more accurate solutions, as the initial conditions were defined using analytic values and an instant closer to the end of simulation could result in very low values of flow variables, such that comparing their deviations from the analytic solution would be meaningless.

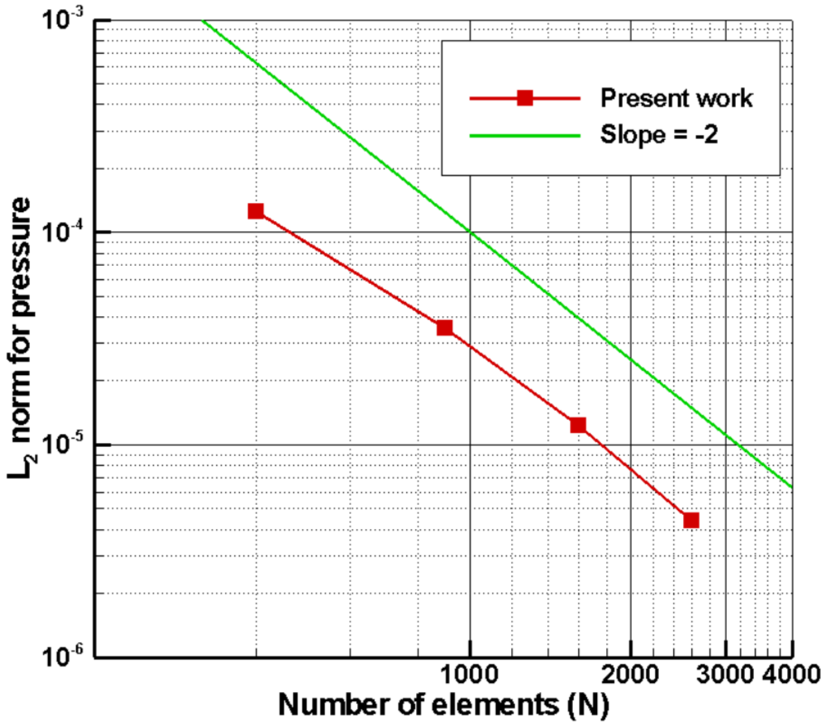

Figure 10: Variation of global pressure norm at *t* = 0.25 with element count.

The order of accuracy of the solver increased with the grid resolution and approached the formal order of 2 over the highly resolved grids of 40^2 and 50^2 elements. Over the range of grids tested, the average order of accuracy was 1.826.

2.3. Temporal accuracy:

A third-order accurate, three-stage Runge-Kutta (*RK3*) scheme was employed for time integration. In this set of simulations, the TGV problem was solved on a 21^2 uniform grid with three time-steps: Δt = 2*e*-3, 1*e*-3 and 1*e*-4. While modeling incompressible flows, the divergence-free constraint of the continuity equation is usually not explicitly solved, but is implicitly used to obtain an equation for pressure. Thus, the degree to which the continuity equation is satisfied – that is the divergence of the velocity – is commonly used as a metric to determine the accuracy of incompressible flow solvers. Thus, the total divergence at time, t = 0.25, was plotted against the time-step used, as shown in Figure 11. The order of accuracy increased from 2.429 when Δt was halved from 2*e*-3 to 1*e*-3, to 2.814 when Δt was decreased to 1*e*-4. The mean order of accuracy over the tested range of Δt was 2.725.

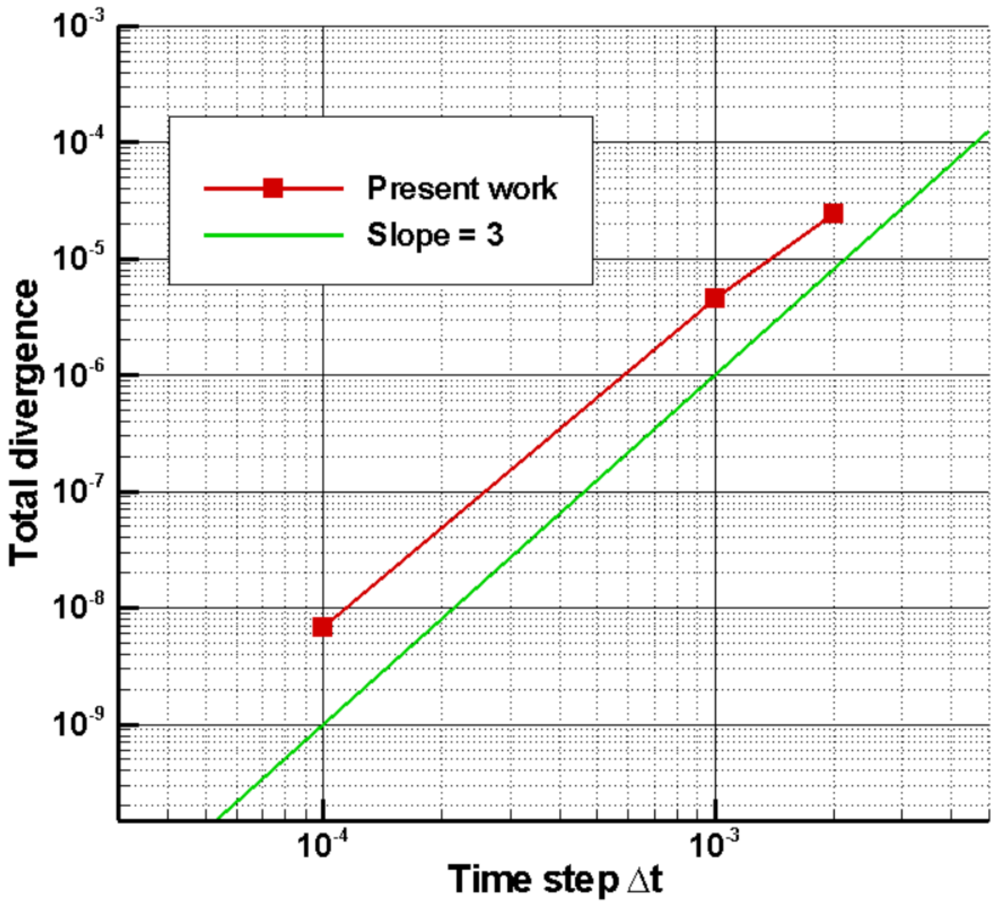

Figure 11: Variation of total divergence with time-step on a 21^2 gird.

3. Shear driven cavity flow:

3.1. Problem Description:

The shear driven cavity flow problem is a standard benchmark problem for steady-state as well as transient flow solvers. Although this problem does not have closed-form exact solutions, it has been widely studied numerically and results, which are widely agreed upon, have been previously established. The most popular work on this problem was conducted by Ghia et al. [7], with which the results from the present study are compared. The shear driven cavity flow problem involves a square domain, bounded by stationary walls at the left, right and bottom boundaries and a moving wall at the top face. Essentially, it describes the motion of a fluid in an open, square box, with its free surface moved using an infinitely-long belt. The schematic of this problem is shown in Figure 12.

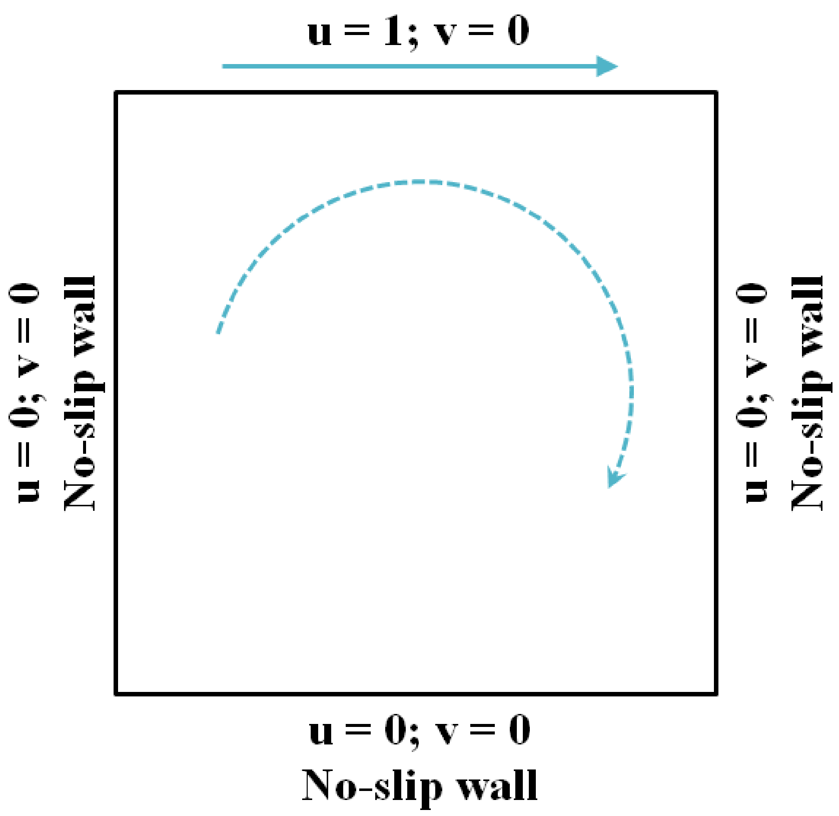

Figure 12: Schematic of the shear driven cavity flow problem.

The flow is driven by the shear forces between adjacent layers of fluid. The topmost layer is moved with the same speed as the cavity lid and the subsequent layers with decreasing speeds, depending on the fluid viscosity. The forward-moving fluid is obstructed and pushed downwards by the rigid wall at the right boundary and continues along the walls which it strikes. The bounded domain, thus, results in complex circulation of the fluid (Figure 13). Thus, the objective of this study was to accurately characterize the vortices resulting from such flow circulation.

Reynolds number for the cavity flow problem is defined as:

$$Re = \frac{U_{top} L}{\nu} \tag{33}$$

Where U_{top} represents the velocity of the top wall, L the cavity edge length and ν the fluid kinematic viscosity.

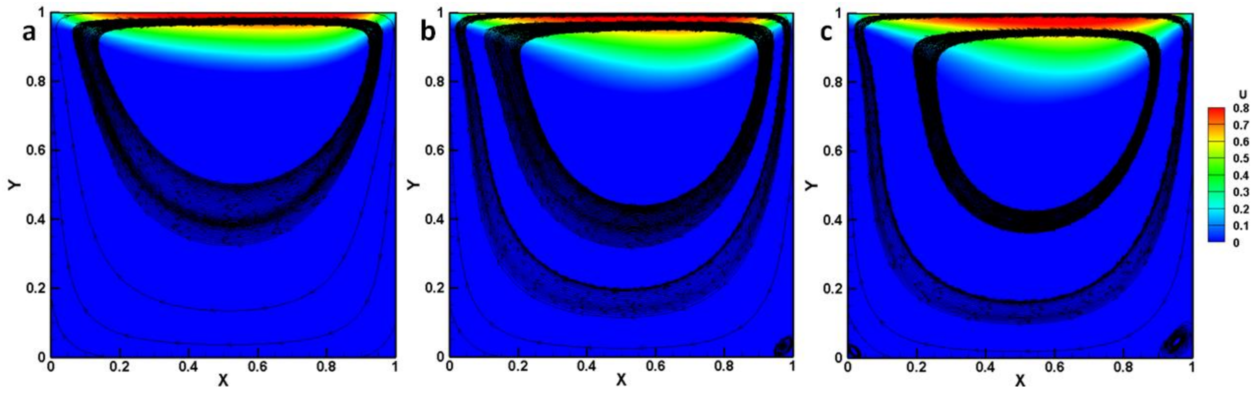

Figure 13: Development of shear-driven cavity flow at Re 100 at three different time-steps. First, (a) a large, central vortex was formed and as the simulation proceeded, (b) a small secondary vortex was formed at the bottom-right corner due to viscous stresses, and finally, (c) at steady state, another secondary vortex was formed at the bottom-left corner.

3.2. Comparison with Ghia et al. [7]:

A simulation at Reynolds number 100 was performed on a 41^2 uniform grid and the results obtained were compared with those of Ghia et al. [7]. Figure 14 (a) shows the flow circulation reported by Ghia et al. [7], indicating a large, central primary vortex and two smaller, secondary vortices at each bottom corner, Figure 14 (b) shows the corresponding flow circulation and contours of x-velocity from the present study. Although the grid resolution in the present simulations was much lower, with about one-third the number of points in each direction, as compared to the simulations of Ghia et al. [7], it sufficiently captured the flow details. The central vortex and the smaller, secondary vortices were distinctly resolved. Further, the sizes of each of these vortices were comparable with those reported by Ghia et al. [7].

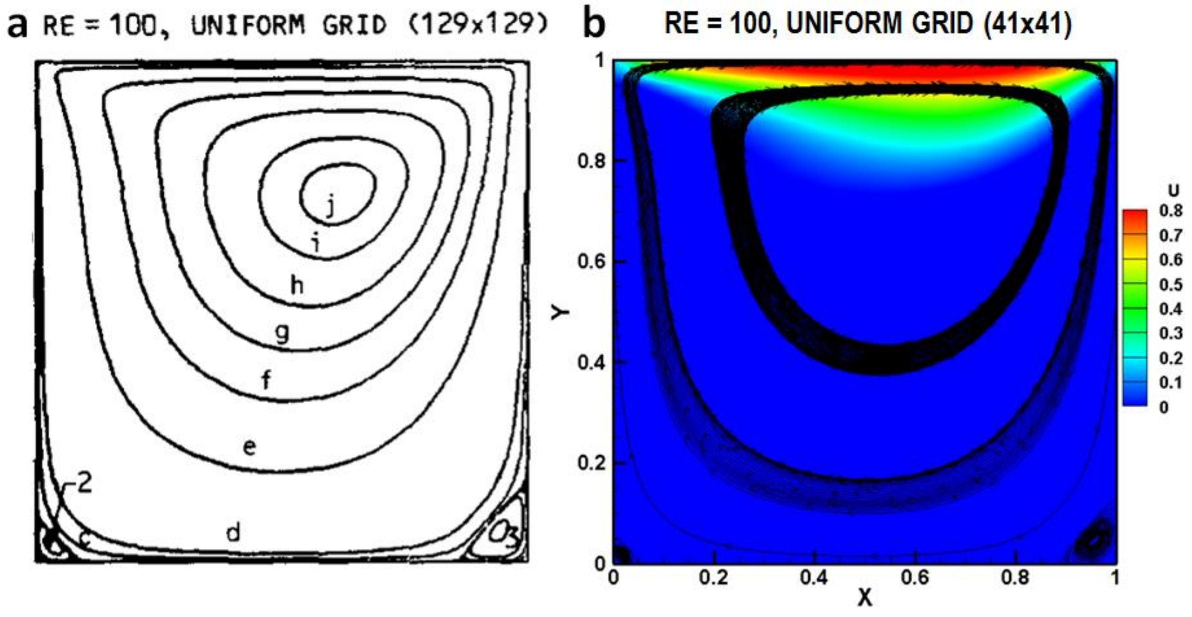

Figure 14: Simulation results at steady-state of the cavity flow problem at *Re* 100, reported by (a) Ghia et al. [7], indicating circulation of flow, and (b) in the present study showing *x*-velocity contours and streamlines, indicating flow circulation and vortical structures.

A comparison of velocity component variation in the present investigation with published data, along two sections, is shown in Figure 15. The results from the present investigation are in excellent agreement with those of Ghia et al. [7], with velocities at all interior points being within 3% of established data. Larger deviations were observed at the boundaries, which can be attributed to linear extrapolation errors by post-processing software, due to absence of solution points at the boundaries, since a cell-centered approach was adopted.

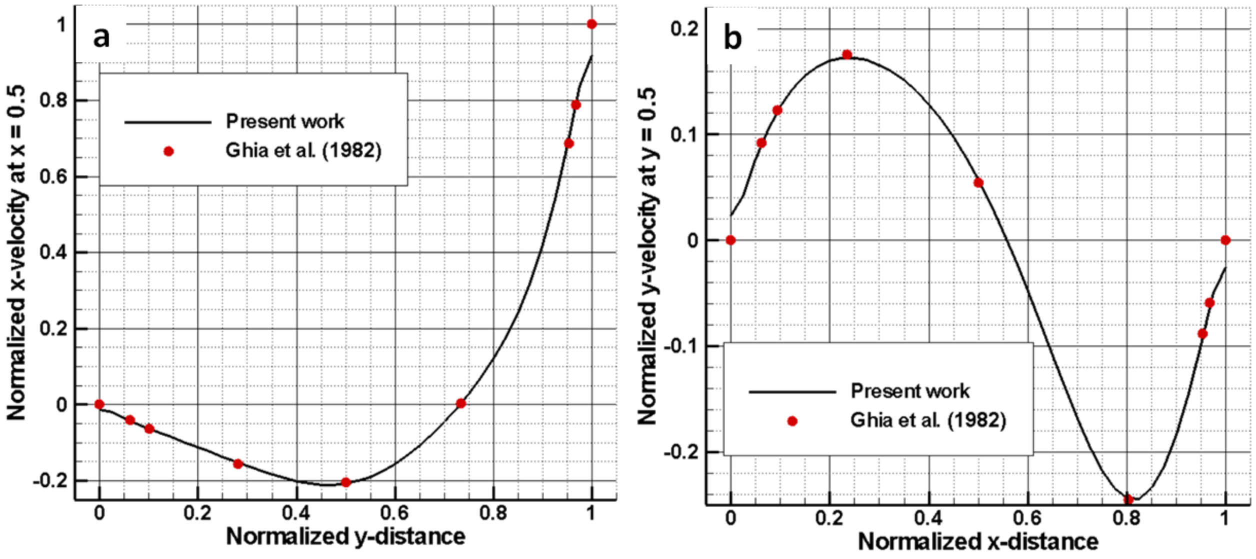

Figure 15: Comparison of (a) *x*-velocity against *y* at *x* = 0.5 and (b) *y*-velocity against *x* at *y* = 0.5. All quantities are normalized with respect to their respective maximum values.

4. Flow over Cylinder

4.1. Problem Description:

Flow over a circular cylinder is one of the most widely studied, fundamental benchmark test problems for fluid-solid interaction. The problem involves a stationary, circular cylinder enclosed in a domain. A uniform flow, inlet boundary condition is specified on one of the boundary faces and a convective, outflow boundary condition on the opposite face. As the domain represents a part of unbounded space, symmetric boundary conditions are imposed on faces parallel to the free stream velocity. A schematic of this test case is shown in Figure 16.

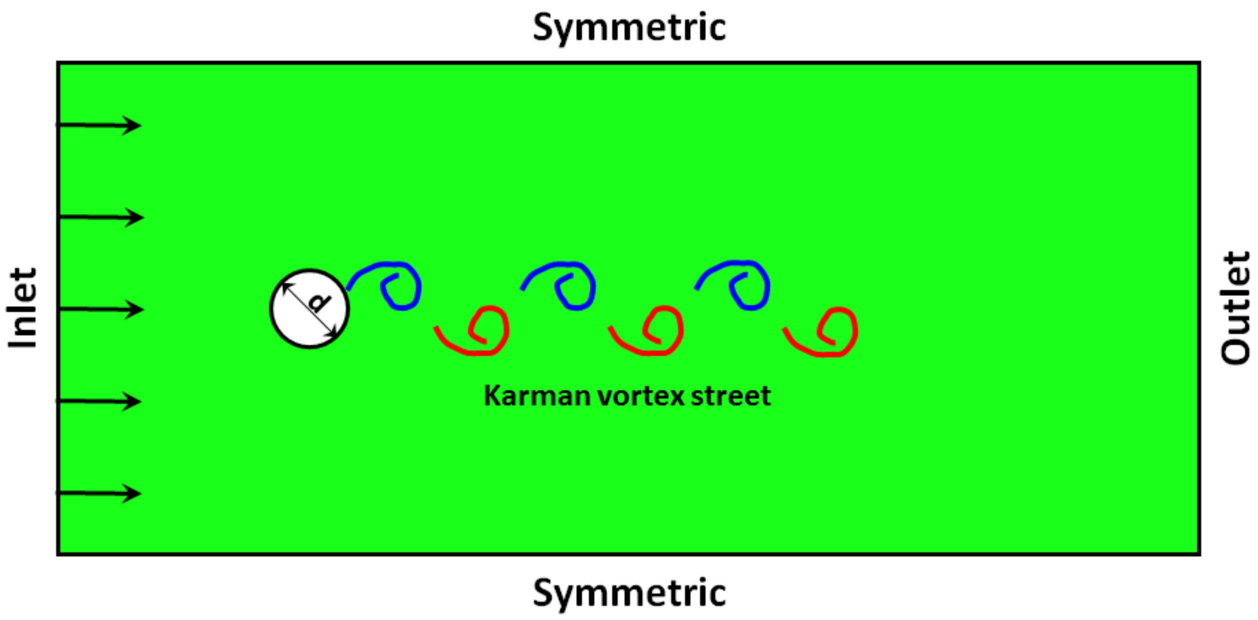

Figure 16: Schematic of flow over cylinder, indicating alternating vortex-shedding from the upper and lower halves of the cylinder, known as Karman Vortex Street.

Reynolds number for this problem is defined as:

$$Re = \frac{U_\infty d}{\nu} \tag{34}$$

Where U_∞ represents the free stream velocity, d the cylinder diameter and ν the fluid kinematic viscosity. At low Reynolds numbers (<48), a steady wake is formed downstream of the cylinder, due to viscous effects. As Re is increased, the larger inertial forces result in separation of boundary layer from the cylinder surface, and vortices are shed from wake of the cylinder. These vortices are not symmetrical on either side of the cylinder at a given instant of time, but are shed alternatingly from above and below the centerline (Figure 16). This phenomenon is called a Karman vortex street. For low to moderate Reynolds numbers ($50 < Re < 180$) vortex shedding is uniform along the length of the cylinder and can be accurately resolved using two-dimensional simulations. At higher Reynolds numbers (> 200), the phenomenon becomes three-dimensional, due to secondary instabilities and 2-D simulations may not be adequate. Thus, the ability to resolve the unsteady vortex street is a qualitative test for an FSI solver.

4.2. Simulation at *Re* 100:

A uniform Cartesian grid of 400x240 points was used over the limits $x \in (-2, 18)$ and $y \in (-6, 6)$. The cylinder was located at (0, 0) and had unity diameter. Results for the simulation are shown in Figure 17. Note that only a part of the domain is shown to accommodate results at various instants. As the flow impacted the cylinder, a stationary wake began forming immediately downstream of the cylinder (Figure 17 (a)). The wake consists of two opposing vortices, wherein flow circulation was observed. As the flow continued, the wake grew in size, spreading to a larger area downstream of the cylinder (Figure 17 (b)). At one instant, the size of the wake was too large to be sustained by the viscous forces. At this point, the boundary layer around the cylinder surface separated from the surface (Figure 17 (c)), leading to shedding of a vortex from the wake. As the simulation proceeded, more vortices were shed, alternatingly from the upper and lower sides of the cylinder, leading to a continuous stream of vortices or a Karman vortex street (Figure 17 (d)).

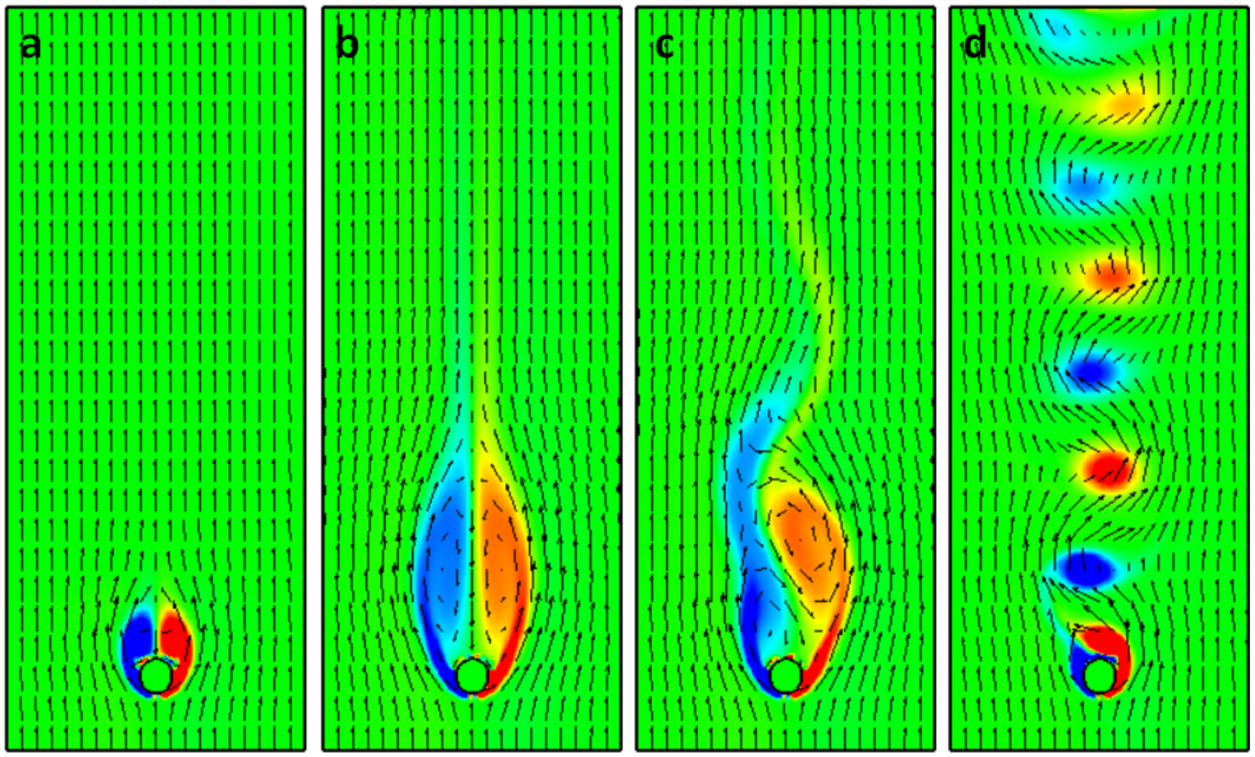

Figure 17: Vorticity contours and velocity vectors for flow over circular cylinder at Re 100 at different instants showing (a) onset of wake formation, (b) a fully developed wake and (c) instability in wake leading to (d) Karman vortex street.

5. Limitations

The present analysis demonstrates the influence of a body immersed in moving fluid, on the surrounding flow using a simple formulation of governing equations on Cartesian grid and a sharp-interface immersed boundary method. However, the study lacks rigorous detail for validating results of flow over a cylinder. A grid refinement study can be conducted to determine the spatial order of accuracy of the immersed boundary implementation. Flow over cylinder problems are validated using various parameters, like drag and lift coefficients, pressure coefficient, Strouhal number, etc. Including drag and lift calculations in computations and determining the Strouhal number for vortex shedding could make the results more complete. Finally, the FSI solver can be improved by incorporating non-uniform Cartesian grids for greater flexibility.

6. Conclusions

A two-dimensional, incompressible Navier-Stokes solver was developed with second-order central difference, spatial discretization and three-stage, third-order Runge-Kutta time integration schemes. A ghost-cell based sharp interface, immersed boundary method was implemented to model fluid-structure interaction. The solver was tested for accuracy in solving the Taylor-Green vortex problem and its spatial and temporal orders of accuracy were established as 1.82 and 2.725, respectively. Next, numerical approximations to the lid driven cavity problem at *Re* 100 were obtained and were validated against results of Ghia et al. [7]. Results from both studies were in excellent agreement and velocities from the present work were within 3% of published results. Finally, the flow over a stationary, circular cylinder at Reynolds number 100 was modeled. Periodic vortex shedding in the wake of the cylinder was observed, which is a known phenomenon called Karman vortex street. However, no quantitative validation was performed.

7. References

1. Bailoor, S., 2015, “MAE 6225 Project part 1”. The George Washington University

2. Balaras, E., 2015, “MAE 6225 Class notes”. The George Washington University

3. Fadlun, E.A., Verzicco, R., Orlandi, P. and Mohd-Yusof, J., 2000, “Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations”, Journal of Computational Physics 161, 35–60 (2000) doi:10.1006/jcph.2000.6484

4. Balaras, E., 2004, “Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations”, Computers & Fluids 33 (2004) 375–404

5. Mittal, R., Dong, H., Bozkurttas, M., Najjar, F.M., Vargas, A. and von Loebbecke, A., 2008, “A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries”, Journal of Computational Physics 227 (2008) 4825–4852

6. Seo, J.H., algridcar – a stretched Cartesian grid generating, non-commercial, software, obtained through personal communication. The software generates grid points using hyperbolic-tangent point distribution.

7. Ghia, U., Ghia, K.N. and Shin, C.T., 1982, "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method", Journal of Computational Physics, Vol. 48, pp. 387-411