Numerical Simulation of Incompressible Fluid Flow

Duncan L. Karnitz

Department of Mechanical Engineering: Michigan Technological University

Submitted as a final report for: MEEM 5240 – Computational Fluid Dynamics and Heat Transfer

Abstract

Simulation of the incompressible Navier-Stokes equations in C++ using first-order methods of explicit nature. Incompressibility is enforced implicitly and stability is maintained via adaptive time stepping according to maximum velocity, grid density, and Reynolds number.

Contiguous Equations

I will begin with a brief description of the Navier-Stokes equation with incompressible constraints:

$$\frac{\partial}{\partial t}\vec{u} + (\vec{u}\cdot\nabla)\vec{u} + \nabla p = \frac{1}{Re}(\nabla\cdot\nabla)\,\vec{u} + \vec{g}$$

$$\nabla\cdot\vec{u} = 0$$

The Navier-Stokes equation describes the evolution of a spatial laminar, viscous flow over time. With the left-hand side describing the convective, incompressible nature when combined with the velocity divergence criteria, and the right-hand side describing the diffusive nature of fluids and effects of external forces (here simplified with only gravitational forces acting as $\vec{g}$.

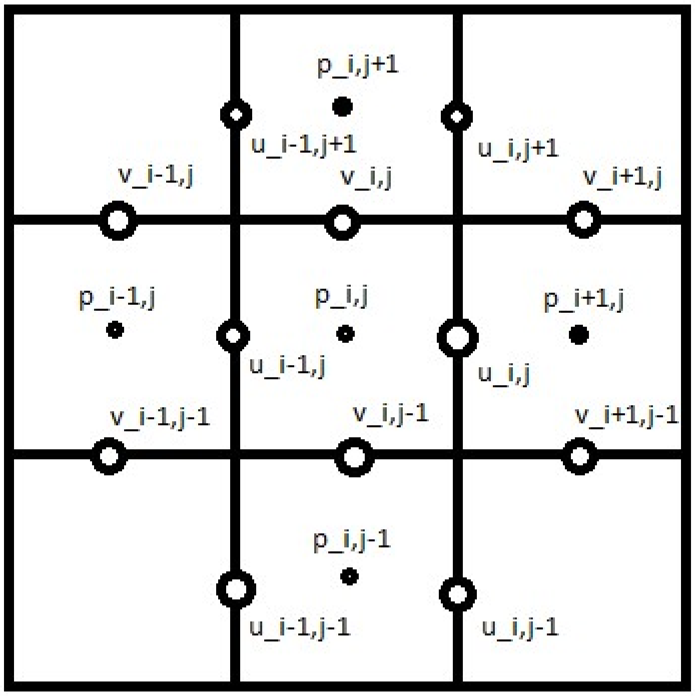
We make a final assumption that none of these characteristics depend on the density of the medium. In two dimensions, the equations can be split into components, $\vec{u} = [\,u\ v\,]^T, \nabla = \left[\frac{\partial}{\partial x}\ \frac{\partial}{\partial y}\right]^T$, and can be written out in component form:

$$\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} + g_x$$

$$\frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} + g_y$$

Discrete Domain

We will follow suggestions made by Bridson (2016) and Griebel (1998) to use a co-located grid for the discretizing the individual components (u, v, p) of the Navier-Stokes equation. We choose this method to avoid accidental trivially oscillating solutions to the differencing terms. The domain will as such be discretized and numbered in a cartesian manner (i, j) with pressure values sampled at uniform cartesian cell centers, and velocity components sampled uniformly at cartesian cell faces. The sampling of velocity components additionally makes evaluation of volume surface fluxes trivial. The following diagram illustrates the numbering relationship adapted by Griebel (1998):



Discrete Equations

The Navier-Stokes equations, like any ordinary differential equation of second order, can be discretized with any standard expansion of the contiguous terms via Taylor series expansion. For simplification, we will use only, at most, second order approximations to each term:

$$\frac{\partial u}{\partial t} = \frac{u^{n+1} - u^n}{dt}, \qquad \frac{\partial v}{\partial t} = \frac{v^{n+1} - v^n}{dt}$$

$$\frac{\partial p}{\partial x} = \frac{p_{i+1,j} - p_{i,j}}{\Delta x}, \qquad \frac{\partial p}{\partial y} = \frac{p_{i,j+1} - p_{i,j}}{\Delta y}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}, \qquad \frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}$$

$$\frac{\partial^2 v}{\partial x^2} = \frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{\Delta x^2}, \qquad \frac{\partial^2 v}{\partial y^2} = \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{\Delta y^2}$$

To handle the convective terms, Griebel (1998) refers to a stabilized form of the upwind differencing which involves mid-point approximations of the non-linear terms to the left and right of the discrete values of $\vec{u}$. The such named *donor-cell scheme*, "is applied mainly to the discretization of convection terms of the form $d(ku)/dx$." And can be written in a form which adaptively selects left and right-hand discretization based on the sign of the convective component. Thus, the scheme avoids domain of dependence errors that arise when using a two-point advection scheme which computes finite differences converse to the direction of flow. The advection terms written in conservative form thus become, via Griebel (1998):

$$\frac{\partial(u^2)}{\partial x} = \frac{1}{\Delta x}\left(\left(\frac{u_{i,j} + u_{i+1,j}}{2}\right)^2 - \left(\frac{u_{i-1,j} + u_{i,j}}{2}\right)^2\right)$$

$$+ \frac{\gamma}{\Delta x}\left(\frac{|u_{i,j} + u_{i+1,j}|}{2}\frac{u_{i,j} - u_{i+1,j}}{2} - \frac{|u_{i-1,j} + u_{i,j}|}{2}\frac{u_{i-1,j} - u_{i,j}}{2}\right)$$

$$\frac{\partial(uv)}{\partial y} = \frac{1}{\Delta y}\left(\frac{v_{i,j} + v_{i+1,j}}{2}\frac{u_{i,j} + u_{i,j+1}}{2} - \frac{v_{i,j-1} + v_{i+1,j-1}}{2}\frac{u_{i,j-1} + u_{i,j}}{2}\right)$$

$$+ \frac{\gamma}{\Delta y}\left(\frac{|v_{i,j} + v_{i+1,j}|}{2}\frac{u_{i,j} - u_{i,j+1}}{2} - \frac{|v_{i,j-1} + v_{i+1,j-1}|}{2}\frac{u_{i,j-1} - u_{i,j}}{2}\right)$$

$$\frac{\partial(v^2)}{\partial y} = \frac{1}{\Delta y}\left(\frac{(v_{i,j} + v_{i,j+1})^2}{2} - \frac{(v_{i,j-1} + v_{i,j})^2}{2}\right)$$

$$+ \frac{\gamma}{\Delta y}\left(\frac{|v_{i,j} + v_{i,j+1}|}{2}\frac{v_{i,j} - v_{i,j+1}}{2} - \frac{|v_{i,j-1} + v_{i,j}|}{2}\frac{v_{i,j-1} - v_{i,j}}{2}\right)$$

$$\frac{\partial(uv)}{\partial x} = \frac{1}{\Delta x}\left(\frac{u_{i,j} + u_{i,j+1}}{2}\frac{v_{i,j} + v_{i+1,j}}{2} - \frac{u_{i-1,j} + u_{i-1,j+1}}{2}\frac{v_{i-1,j} + v_{i,j}}{2}\right)$$

$$+ \frac{\gamma}{\Delta x}\left(\frac{|u_{i,j} + u(i,j+1)|}{2}\frac{v_{i,j} - v_{i+1,j}}{2} - \frac{|u_{i-1,j} + u_{i-1,j+1}|}{2}\frac{v_{i-1,j} - v_{i,j}}{2}\right)$$

The parameter γ in the above equations should be chosen to stabilize the computations to select, per Griebel (1998), the appropriate central-difference or pure donor-cell scheme. Contributions by Hirt (1975) derived that the parameter should be chosen to satisfy the relation:

$$\gamma \geq \max\left(\frac{\left|u_{i,j,max}\Delta t\right|}{\Delta x}, \frac{\left|v_{i,j,\max}\Delta t\right|}{\Delta y}\right)$$

We shall choose the equality from this relation for our implementation that follows.

Method

Taking the discrete approximations to the Navier-Stokes equation we can develop an explicit method for updating the approximations of (u, v, p) at each time step.

$$u^{n+1} = u^n + \Delta t\left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} + g_x - \frac{\partial p}{\partial x}\right]$$

$$v^{n+1} = v^n + \Delta t\left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} + g_y - \frac{\partial p}{\partial x}\right]$$

Griebel (1998) introduces an abbreviation to simplify the derivation of the implicit enforcement of incompressibility. Additionally, his abbreviations aid in the implementation of the method. We define:

$$F^n = u^n + \Delta t\left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} + g_x\right]$$

$$G^n = v^n + \Delta t\left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} + g_y\right]$$

Thus, the update to u^{n+1}, v^{n+1} can be formed implicitly based on pressure:

$$u^{n+1} = F^n - \Delta t\frac{\partial p^{n+1}}{\partial x}$$

$$v^{n+1} = G^n - \Delta t\frac{\partial p^{n+1}}{\partial y}$$

To enforce the continuity, incompressible constraints we evaluate the constraint at $t = n + 1$ to get an equation for pressure:

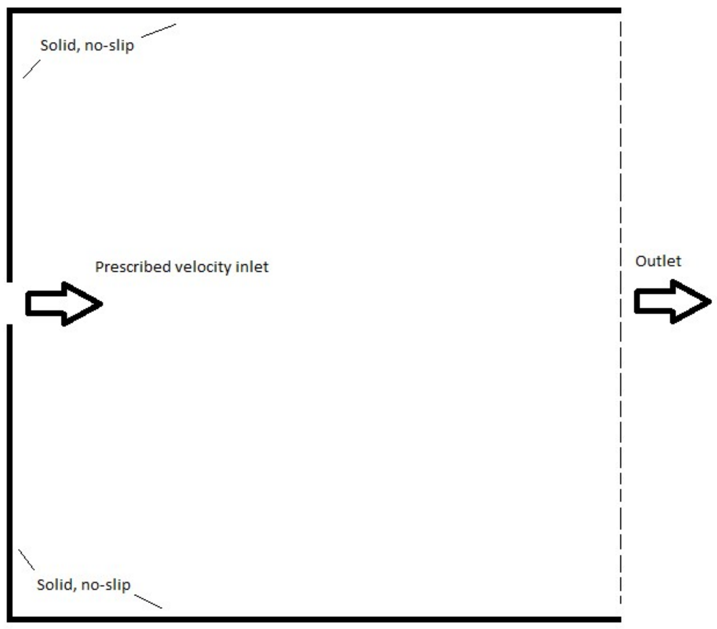
$$0 = \frac{\partial u^{n+1}}{\partial x} + \frac{\partial v^{n+1}}{\partial y} = \frac{\partial F^n}{\partial x} - \Delta t \frac{\partial^2 p^{n+1}}{\partial x^2} + \frac{\partial G^n}{\partial y} - \Delta t \frac{\partial^2 p^{n+1}}{\partial y^2}$$

$$\frac{\partial^2 p^{n+1}}{\partial x^2} + \frac{\partial^2 p^{n+1}}{\partial y^2} = \frac{1}{\Delta t}\left(\frac{\partial F^n}{\partial x} + \frac{\partial G^n}{\partial y}\right)$$

The Poisson equation for pressure can be easily solved by any iterative method. We gain a benefit here from our choice of a non-co-located grid. Since u, v are known at fluid cell faces F, G are also known directly at fluid cell faces. Since pressure is define at fluid cell centers the right-hand side derivatives of F, G can be evaluated easily by a backwards difference (according to our numbering of cell centers and cell faces in the discrete domain).

Implementation

The equations above were implemented in C++ to simulate the behavior of a prescribed velocity jet into a stationary fluid domain bounded by solid surfaces on three of the four sides of a two-dimensional cartesian domain. The setup is illustrated as such:



To handle boundary conditions, a thin layer of fluid cells are prescribed as boundary cells. For cells labeled as “Inlet”, at each step the prescribed velocity components were assigned. For cells labeled “Solid”, at each step the perpendicular velocity was set to zero to enforce that no fluid shall cross the fluid-solid interface. The tangential component sampled directly inside the solid was set to the inverse of the tangential component sampled directly outside the solid. This ensured that any derivative components calculated from the average of these two samples would evaluate to zero as expected for all velocity components immediately adjacent to a no-slip surface. For cells labeled as “Outlet”, at each step the velocity components were copied from the interior to the boundary to enforce a Zero-Neumann boundary condition for all components.

For the pressure Poisson equation, we used Zero-Neumann boundary conditions at the “Inlet” and “Solid” boundaries. This ensures that the prescribed velocities do not change on those boundaries. Inlets should stay at inlet speed and velocities perpendicular to solid faces should remain at zero. At the “Outlet” cells, we imposed an outlet pressure, typically zero for simplicity. A Neumann boundary condition was tested so that we would not artificially accelerate the flow at outlets but this proved to be unstable and failed to produce a convergent pressure solve. To simplify the setup of the left-hand side matrix at the boundaries, these conditions were re-assigned at each iteration of the SOR smoothing so that these conditions were iteratively approximated by the solve.

The discrete form of the Navier-Stokes equation was evaluated explicitly in time for all convection and diffusion terms. The pressure term was solved implicitly to enforce the incompressible constraint at each time step using an SOR smoother. The algorithm is described briefly as follows, and can be found explicitly in separate material as Appendix A:

1. Assign Dirichlet and Neuman velocity conditions at boundaries.
2. Determine a stable time step size and the parameter γ for the convective terms.
3. Evaluate explicit terms F^n, G^n at all interior points.
4. Generate the right-hand side of the system of equations for pressure.
5. Solve for p using an SOR smoother until converged in the L2 norm.
6. Evaluate $\vec{u}$ explicitly using the new pressures.

The algorithms were accelerated by using OpenMP to leverage multi-core processing. An export of the domain at fluid cell vertices was implemented for visualization in post-processing tools that support the VTK Ascii file format (e.g. VisIt, Lawrence Livermore National Labs).

Simulations

We started with a small domain (2.0m x 0.4m) and a relatively coarse discretization with fluid cells of size (0.025m x 0.025m). An inlet velocity of 100 m/s was assigned. An outlet relative pressure of zero was assigned for Outlet cells, and a Reynolds number of 100 was selected. The simulation progressed for 1.0 second, choosing a stable time step adaptively. Images of the flow simulation at 0.0s [Fig 1], 0.1s [Fig 2], and 1.0s [Fig 3] are found in the Figures. Additional experiments with varying Reynolds numbers were performed and can be found in figures [4-9]. Figures depict fluid flow speed, augmented with flow direction vectors in some cases. The final figure [Fig 9] shows flow direction vectors scaled with fluid speed to show low speed eddies in the flow domain.

Summary

Simulation of the Navier-Stokes equation in a direct fashion was accomplish with low-order approximations to each of the differential terms. A cartesian grid domain was used to simplify the implementation and numbering of discrete points in the C++ programming language. The domain was further implemented with non-co-located sampling of the fluid properties (u, v, p) which avoided issues of trivial oscillatory solutions to locally smooth functions and simplified the coupling between differential pressure terms and convective terms.

Improvements

The SOR smoother used to solve the pressure Poisson equation has poor convergence rates for large problems. Thus, direct simulation of fine detail fluid flow was not economical without immense computing capabilities since halving the spatial discretization caused an $O(n^5)$ increase in run-time due to halving the time-step size (double the number of steps), squaring the number of unknowns, and likewise squaring the number of SOR sweeps needed for convergence in the worst case. A better approach would be to implement a full-multigrid solver to reduce the complexity of the solve to scale linearly with the number of unknowns.

References

Bradie, B. (2006). A Friendly Introduction to Numerical Analysis. Pearson Prentice Hall.

Bridson, R. (2016). Fluid Simulation for Computer Graphics. CRC Press.

Griebel, M. (1998). Numerical Simulation in Fluid Dynamics. SIAM.

Hirt et al. (1975). *A Numerical Solution for Transient Fluid Flows*. Los Alamos National Lab.

Figures

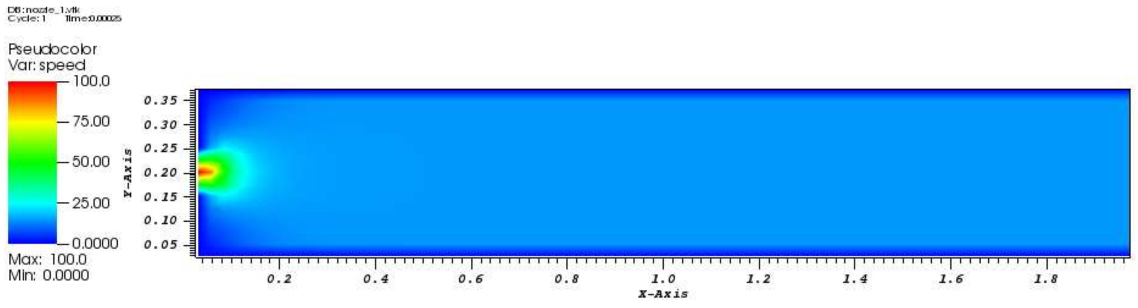


Figure 1. Inlet velocity 100 m/s. Re = 100, $\Delta x = 0.025$, Fluid domain state at t=0.0s.

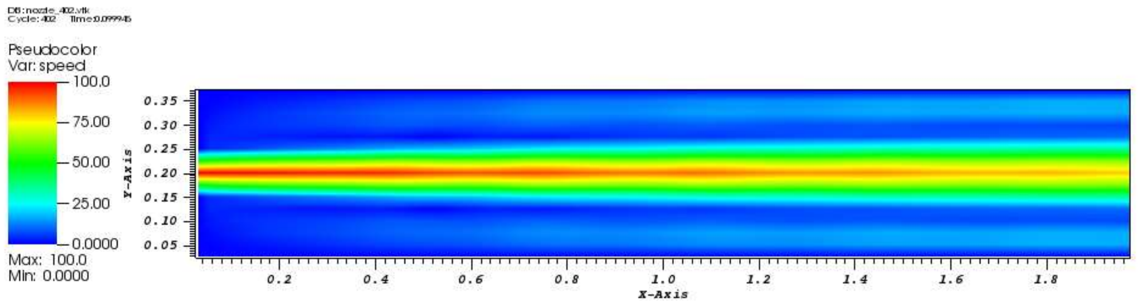


Figure 2. Inlet velocity 100 m/s. Re = 100, $\Delta x = 0.025$, Fluid domain state at t=0.1s.

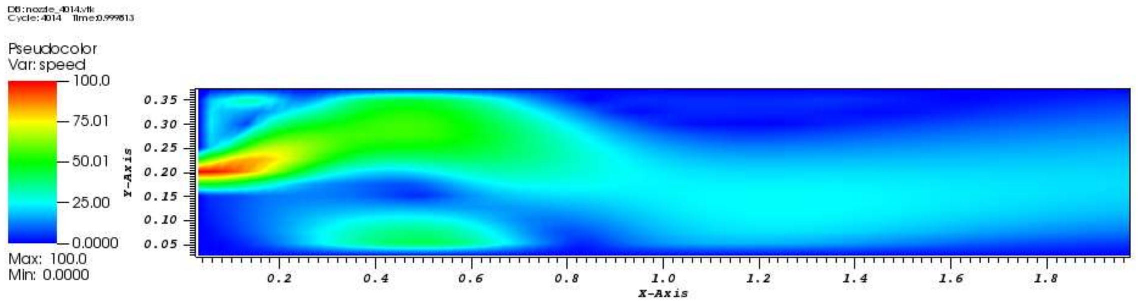


Figure 3. Inlet velocity 100 m/s. Re = 100, $\Delta x = 0.025$, Fluid domain state at t=1.0s.

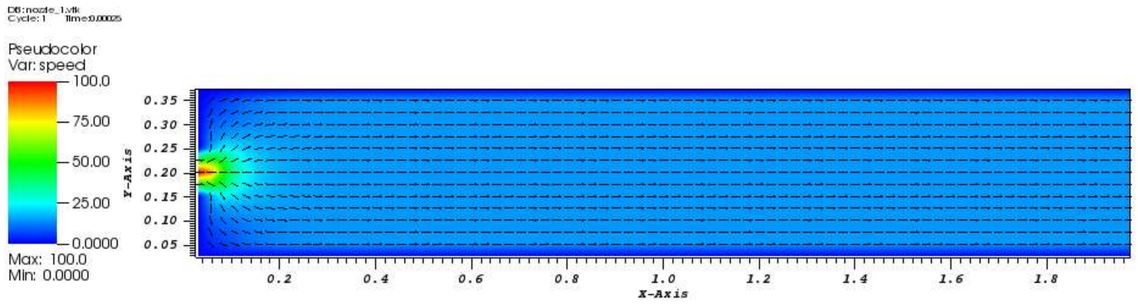


Figure 4. Inlet velocity 100 m/s. Re = 10000, $\Delta x = 0.025$, Fluid domain state at t=0.0s.

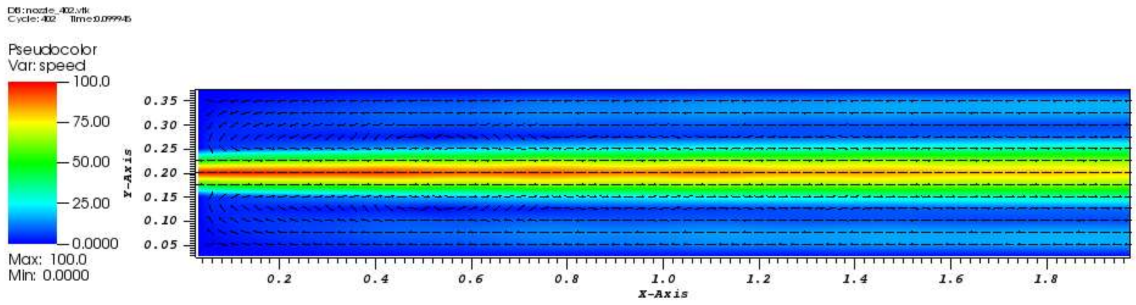


Figure 5. Inlet velocity 100 m/s. Re = 10000, $\Delta x = 0.025$, Fluid domain state at t=0.1s.

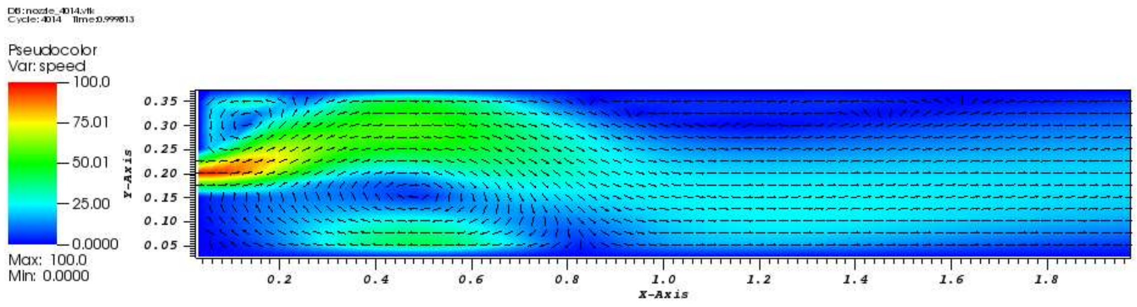


Figure 6. Inlet velocity 100 m/s. Re = 10000, $\Delta x = 0.025$, Fluid domain state at t=1.0s.

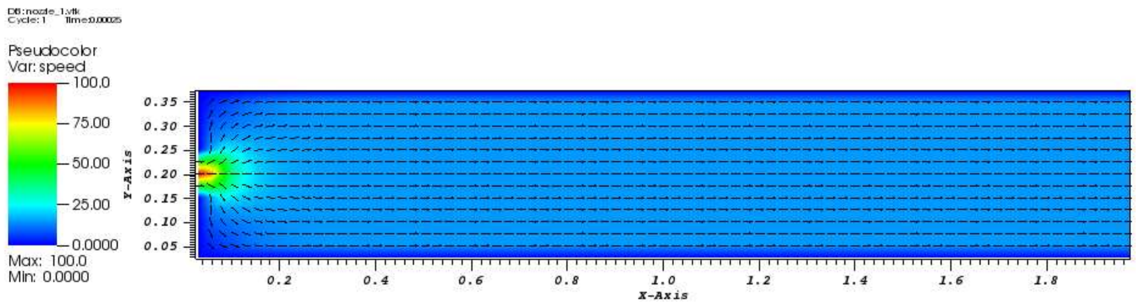


Figure 7. Inlet velocity 100 m/s. Re = 1e-3, $\Delta x = 0.025$, Fluid domain state at t=0.0s.

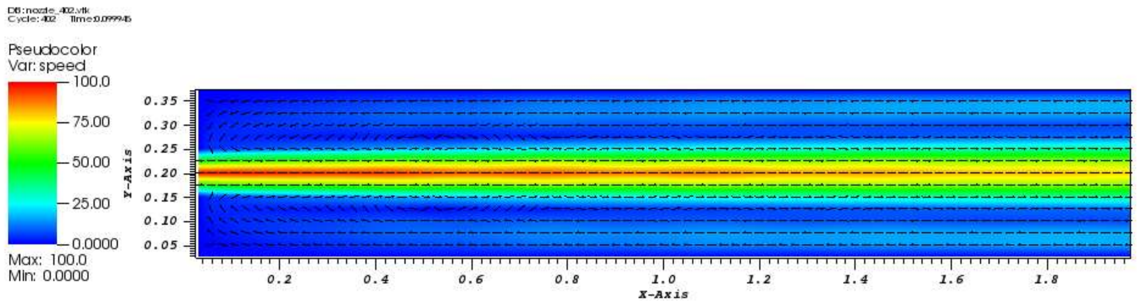


Figure 8. Inlet velocity 100 m/s. Re = 1e-3, $\Delta x = 0.025$, Fluid domain state at t=0.1s.

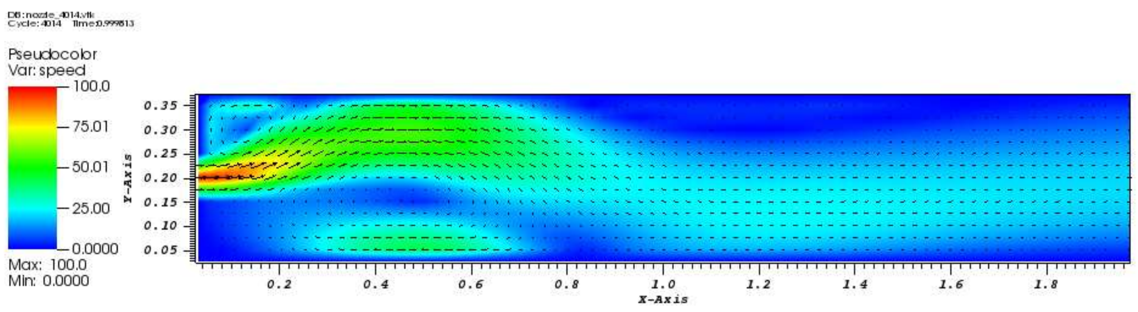


Figure 9. Inlet velocity 100 m/s. Re = 1e-3, $\Delta x = 0.025$, Fluid domain state at t=1.0s.