Numerical solution of Navier stokes in 1D, 2D and turbulence modelling

Shubham P. Raghuvanshi

May 9, 2021

1 Particle vs field approach

- There are two main approaches to fluid dynamics namely the particle approach a.k.a. Lagrangian description and the field approach a.k.a Eulerian description. In the Lagrangian description the flow is described by following a very large number of interacting fluid particles as they move. In the Eulerian description, a flow field is defined over the domain whose properties over space domain and time describe the fluid flow. In our analysis we will take the latter approach since the former is computationally more expensive.
- To this end we describe the fluid by a set of fields permeating over the domain of analysis. These fields at least include the mass density field $\rho(\vec{x}, t)$, velocity field $\vec{u}(\vec{x}, t)$ and a hydrodynamic pressure field $P(\vec{x}, t)$, for our current analysis we will limit our attention to these three fields. However if we want to study the heat transfer or transfer of any other fluid property across with the fluid then we can include their corresponding fields. If the fluid is to be studied under the effect of any external force then we can include an external force field.

2 Conservation of mass

We define material volume $V_m(t)$ as the volume occupied by a specific collection of fluid particles, such a volume moves and deforms within a fluid flow such a way that it always contains the same mass elements, none enter the volume and none leave it, like the volume enclosed by surface of a baloon. Therefore by definition

$$\frac{d}{dt}\int_{V_m(t)} \rho\left(\vec{x}, t\right) dV = 0 \tag{1}$$

Which implies

$$\int_{V_m(t)} \left(\frac{\partial \rho}{\partial t} + \nabla\left(\rho\vec{u}\right)\right) dV = 0 \tag{2}$$

Where $\vec{u}$ is the normal velocity of the boundary enclosing the material volume. Derivation of above equation can be found at appendix A.1. The above equation 2 can be written in a differential form as

$$\frac{\partial \rho}{\partial t} + \nabla\left(\rho\vec{u}\right) = 0 \tag{3}$$

For an incompressible fluid the material volume does not deform as during the course of motion, it just moves along with the local velocity field. This means $\nabla\vec{u} = 0$. This gives the mass conservation for an incompressible fluid as

$$\frac{\partial \rho}{\partial t} + \vec{u} \bullet \nabla\rho = 0 \tag{4}$$

3 Conservation of momentum

The calculation of appendix A.1 can also be done for any smooth vector field, repeating the same calculation for momentum density field $\rho\vec{u}$ we get the equations in the component form as

$$\frac{\partial(\rho u_i)}{\partial t} + \partial_j(\rho u_j u_i) + \frac{\partial P}{\partial x_i} = 0 \tag{5}$$

which can also be written as

$$\frac{\partial(\rho u_i)}{\partial t} = -\partial_j T_{ij} \tag{6}$$

where $T_{ij} = \rho u_i u_j + \delta_{ij} P$ is the stress energy tensor for ideal fluid and equation(5) can also be thought of as the Newton's equation of motion for ideal fluid. To model a fluid with viscosity we add another term in the stress energy tensor which is $-\eta S_{ij}$ where η is coeffiecient of kinematic viscosity.

and $S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the strain rate tensor. For incompressible flow $S_{ii} = 0$ where summetion convention is implied. Using this we get the Navier stokes equation for viscous incomressible fluid as

$$\frac{\partial u_i}{\partial t} = -u_j \partial_j u_i - \frac{1}{\rho}\partial_i P + \mu \partial^2 u_i \tag{7}$$

$$\partial_j u_j = 0 \tag{8}$$

Where $\mu = \eta/\rho$ is coefficient of dynamic viscosity. Above set of equation constitute two coupled non linear partial differential equations and in order to get the complete solution of a incompressible viscous fluid flow problem above two equation have to be solved simulatanously.

4 Numerical solution in 2D

In this section I will discribe the stategy for simultanously solving equation(7) and (8) numerically with arbitrary boundary conditions, but before that let me discuss briefly some of the challanges in doing so and how can they be handled.

- The Navier stokes momentum equation is non-linear multivariate boundary value problem with the noslip boundary condition on the boundary, we will assume the boundaries to be stationary so that the no slip condition is simply $u_i = 0$ at the boundary points.
- The non linearity can be handled numerically but there is bigger problem here since the momentum equation contains a pressure gradient term in RHS but there is no governing equation for pressure and we also cannot use an equation of state for it since a moving fluid is never in a thermodynamic equillibruim. There are two ways to tackle this issue numerically (1) Vorticity stream-function approach also known as non-primitive variable approach and (2) primitive varibable approach. We will not use Vorticity stream-function approach for our simulation therefore the section 4.1 may be skipped. and the reader can go directly to section 4.2

4.1 Vorticity stream-function approach

In this method we make use of the fact that $\nabla \times \nabla P = 0$ and try to eliminate the pressure term by taking curl of NS mometum equation. Doing this we get transport equation for vorticity $\omega = \partial_1 u_2 - \partial_2 u_1$

$$\frac{\partial \omega}{\partial t} = -u_1 \partial_1 \omega - u_2 \partial_2 \omega + (\partial_1^2 + \partial_2^2)\omega$$

Derivation of above equation is given in appendix A.2. In 2D the incompressibility condition can be satisfied by assuming $u_1 = \partial_2 \psi$ and $u_2 = -\partial_1 \psi$. Substituting this in above equation we get

$$\frac{\partial \omega}{\partial t} = -(\partial_2 \psi)(\partial_1 \omega) + (\partial_1 \psi)(\partial_2 \omega) + (\partial_1^2 + \partial_2^2)\omega \tag{9}$$

and ψ can be obtained from the poisson equation

$$\omega = \partial_1 u_2 - \partial_2 u_1 = -(\partial_1^2 + \partial_2^2)\psi \tag{10}$$

Now since the pressure has been eliminated equation(9) and (10) can be solved numerically to get ω and ψ and the velocity field can be derived from these. Even though pressure has been eliminated from these equation it can be obtained once velocity field is known by solving the pressure poisson equation derived in appendix A.3. The algorithm to solve the above equation is the following

1. Initialize u_1, u_2 and impose noslip boundary conditions.
2. Using u_1, u_2 derive boundary conditions on ω and ψ.
3. Use discrtized version of equation(9) to update ω
4. Solve discretized version of poisson equation for ψ
5. Get velocity field from ψ
6. Go back to step 2.

The method handles incompressibility quite well and resolves the pressure issue, but it is only applicable in 2D and not extendable to 1D or 3D, since a stream function can be defined only in 2D. Also the boundary conditions on the secondary varibles ψ and ω are not given, they are derived from boundary conditions on velocity field by using Taylor's expansion at the boundary points. This type is of initialization of boundary values is prone to truncation error. Also solving poisson equation numerically for arbitrary boundary conditions is an iterative process and adds it's own roundoff error. Due to these reasons we do not consider this method any further and use primitve variable method for our solution.

4.2 Pressure corrector or Primitive variable approach

This method makes use of the fact that the pressure in momentum equation can derived from the incompressibility condition. Therefore we do not eliminate pressure but use it to correct the velocity field such that it satisfies the incompressibility condition. The algorithm we use is known as MAC scheme.

We use finite difference method (FDM) to discritize NS equation, these method is based on approximating a function by it's Taylor series expansion upto first order i.e.

$$f(x_i \pm \Delta x) = f(x_i) \pm \left.\frac{df(x)}{dx}\right|_{x=x_i} \Delta x + O(\Delta x^2)$$

This gives the approximation of the derivative as

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = \frac{f(x_i) - f(x_i - \Delta x)}{\Delta x} + O(\Delta x) \tag{11}$$

The formula above is known as backward difference formula for derivative of a function similarly there is also a forward difference formula given by

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x} + O(\Delta x) \tag{12}$$

The above formulas are derived by using the Taylor series expansion of $f(x_i \pm \Delta x)$. Note that both the formulas are have the accuracy of order Δx. However if we subtract $f(x_i - \Delta x)$ from $f(x_i + \Delta x)$ we get central difference formula

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = \frac{f(x_i + \Delta x) - f(x_i - \Delta x)}{2\Delta x} + O(\Delta x^2). \tag{13}$$

Which has the order of accuracy $O(\Delta x^2)$ but as shown in appendix A.4 that central space difference schemes can make the solutions of convection diffusion problems unstable for low values of viscosity and gives no stable solution for zero viscosity. For time derivatives we will use forward difference formula since we want your solution to march forward in time. This type of discretization scheme is known as forward time central space (FTCS) scheme. The second derivative is discretized as

$$\left.\frac{d^2f(x)}{dx^2}\right|_{x=x_i} = \frac{f(x_i + \Delta x) - 2f(x_i) + f(x_i - \Delta x)}{\Delta x} + O(\Delta x) \tag{14}$$

From now on we will use the follwing notation.

- u and v to denote x and y component of velocities.
- subscript will denote discretized space variable and superscript will denote discretized time variable.
- i,j will be used to denote the discretized location on x and y axis respectively.
- P = (i,j), N = (i,j+1), S = (i,j-1), W = (i-1,j), E= (i+1,j)

The algorithm works as follows. For incompressible flows the NS momentum equation can be written as

$$\frac{\partial u_i}{\partial t} = -\partial_j(u_j u_i) - \frac{1}{\rho}\partial_i p + \mu \partial^2 u_i$$

Using our initial guess for pressure ($\bar{p}$) we can update the velocities as

$$\bar{u}^{n+1} = u^n + \Delta t \left(-\partial_x(uu) - \partial_y(uv) + \mu\partial^2 u\right)^n - \frac{\Delta t}{\rho}\partial_x \bar{p} \tag{15}$$

$$\bar{v}^{n+1} = v^n + \Delta t \left(-\partial_x(uv) - \partial_y(vv) + \mu\partial^2 u\right)^n - \frac{\Delta t}{\rho}\partial_v \bar{p} \tag{16}$$

The bar on the velocities and pressure indicate that this is just our guess for u,v and p since they may not satisfy incompressibility condition.

$$\partial_x u^{n+1} + \partial_y v^{n+1} = 0 \tag{17}$$

The correct velocites can be obtained if we have correct values of pressure i.e.

$$u^{n+1} = u^n + \Delta t \left(-u\partial_x u - v\partial_y u + \mu\partial^2 u\right)^n - \frac{\Delta t}{\rho}\partial_x p \tag{18}$$

$$v^{n+1} = v^n + \Delta t \left(-u\partial_x v - v\partial_y v + \mu\partial^2 u\right)^n - \frac{\Delta t}{\rho}\partial_v p \tag{19}$$

Subtracting the two sets of equation we get

$$u^{n+1} = \bar{u}^{n+1} - \frac{\Delta t}{\rho}\partial_x p' \tag{20}$$

$$v^{n+1} = \bar{v}^{n+1} - \frac{\Delta t}{\rho}\partial_y p' \tag{21}$$

Where $p' = p - \bar{p}$ is the pressure correction required to make velocities diverenceless. Taking derivatives and adding above two equations and using the incompressibility condition we get

$$(\partial_x^2 + \partial_y^2)p' = \frac{\rho}{\Delta t}\left(\partial_x \bar{u}^{n+1} + \partial_y \bar{v}^{n+1}\right) \tag{22}$$

This is poisson's equation for pressure correction, once it is solved we can update the pressure as $p = p' + \bar{p}$ and from it we can calculate the correct velocities. Using central difference for space derivatives we can write

$$(\partial_x^2 + \partial_y^2)p' = \frac{p'_E - 2p'_P + p'_W}{\Delta x} + \frac{p'_N - 2p'_P + p'_S}{\Delta y} \tag{23}$$

Solving poisson equation is a recursive and time consuming process which introduces it's own numerical errors. Many of the pressure corrector schemes are same till this point. Now comes the crutial step where various pressure corrector schemes differ. MAC scheme assumes that velocities at a point correct pressures only at that point that is to say $p'_W = p'_N = p'_E = p'_S = 0$. This gives

$$p'_P = -\frac{\rho}{4\Delta t}\frac{\left(\frac{\bar{u}_E - \bar{u}_W}{2\Delta x} + \frac{\bar{v}_N - \bar{v}_S}{2\Delta y}\right)}{\left(\frac{1}{\Delta x} + \frac{1}{\Delta y}\right)} \tag{24}$$

One this pressure correction is known the correct pressure can be calculated from $p = p' + \bar{p}$. From which the corrected velocities can also be calculated from equation (20) (21). We also need to make sure that the numerical solution we get from this explicit scheme actually corresponds to actual solution and that the roundoff error do not make the solution explode and give unpysical results. For this the space and time must be discretized by following criteria. Which is derived in appendix A.4.

$$\Delta t \leq \frac{2\mu/\rho}{u^2 + v^2} \tag{25}$$

$$\Delta x \geq \sqrt{u^2 + v^2}\Delta t \tag{26}$$

Therefore our algorithm for solving the NS equations in 2D is as follows

1. Initialize u, v and p.
2. Check wheather the parameters satisfy the conditions for stability. Continue the simulation iff they do.
3. Impose noslip boundary conditions on velocities.
4. Predict the new u and v from the discretized NS equations (15), (16) with FTCS scheme.
5. Impose noslip boundary conditions on velocities.
6. Calculate the pressure correction from equation(24) and calculate the corrected pressure from $p = p' + \bar{p}$
7. Calculate the updated velocites from equation (20) (21).
8. Calculate the average divergence of the velocity field per unit grid point, if it is not less than the required accuracy then go back to step 3. Break this loop once it is withing required accuracy limit.
9. Update u and v for next time step and go back to step 3.

5 Simulation

Till now we have described the equations to be solved and methodology to solve them numerically. In this section we put these methods described to solve the equations numerically and simulate the results, we also explain the code which does so and how to interpret various outputs produced by the code. The whole project is written in C++ and the source files can be found at this github link.

The project comprices mainly of two C++ classes named "fluid" and "geometry". The fluid class is created for doing the follwing things.

- Defining an N by M grid on which the fluid flow equations will be solved.
- Initializing fluid parameters such as grid spacing ($\Delta x = \Delta y$), time step for evolution of solution (Δt) , density of the fluid (ρ) and coefficient of viscocity (η). The spacial domation of analysis is $0 \leq x \leq N\Delta x$, $0 \leq y \leq M\Delta y$.
- Defining fields vx, vy, temp_vx, temp_vy for velocities. Temp variables hold intemediate results for the calculations.
- Defining scalar field P for pressure.
- Defining arrays bx and by which hold the locations of all the boundary points.
- Giving initial values of boundary points and velocity fields using functions init_boundary(), init_velocity().
- Imposing noslip boundary conditions using the function noslip().
- Updating velocity field from Navier stokes equation using the function Vel_evolve().
- Evolving velocity and pressure fields using MAC scheme using function Evolve().
- Evolving a dye places in our fluid field using the function Den_evolve().

The class geometry does the following jobs for us.

- Gives locations geomerical shapes in 2D such as a point, or a line between two points, a rectangle, circle. These points are used as boundary points in our simulation. For example a flow inside a pipe can be simulated using two parrallel lines as boundaries.
- Draws scalar field on each location screen, the brightness of the pixel denotes the magnitude of the field at that location and color indicates the sign i.e. red for positive values and blue for negative value.
- Draws vector fields on screen where each arrow is of equal length and the magnitude of the vector is denoted by the brightness of the arrow on screen.
- Calculates partial derivatives, divergnce, laplacian of the vector fields using either central or backwards difference scheme depending on the choise of the user.
- Draws streamlines for the flow.

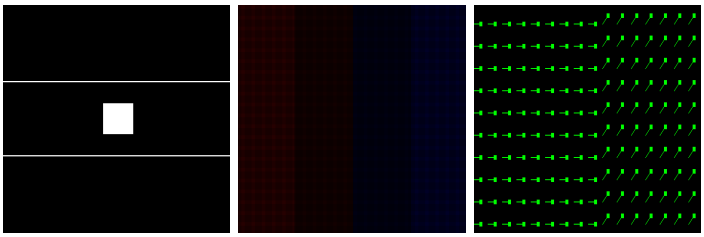

Figure1: Left figure shows boundary drawn by two lines and a solid square, central figure shows a scalar field where the magnitude of the field is 1, 0.5,-0.5,-1 in the four subdomains, and the right figure shows a vector field which is given by $\hat{x}$ and $\hat{x} + \hat{y}$ in the two subdomains.

5.1 Solved examples

(a): Now let us look at some examples using everything we discussed till now. Our first example is the flow inside a pipe. We take initial velocity field to be the $\vec{v} = 5\hat{x}$ everywhere inside the pipe as shown in the figure 2.a. We use 256×256 grid with uniform grid spacing $\Delta x = \Delta y = 1$ and $\Delta t = 0.05$. There is noslip boundary conditions where the fluid is in contact with the pipe. The boundary of the grid is taken to be periodic i.e. left and right, top and bottom of the grid are connected like Klein bottle. The density is uniform eevrywhere $\rho = 1$. We evolve the field using pressure corrector technique from t=0 to t=50, the solution becomes almost steady upto t=50 i.e. $\partial_t \vec{v} = 0$. The absolute error in the divergence of the field is no more than 0.0001 at any time. We put a non diffusive dye in our final fluid configuration and see how it moves under the velocity field of the fluid. The outputs for different values of viscosity are shown

in figure 3. Note that in final output the magnitude(brightness of the arrow) of the field is 0 at the boundary and increases slowly till it reaches a maximum value at the center. This velocity diffusion stronger for higher viscosity. The user may run the executable files " ./exec_dye_pipe_v1 " and "./exec_dye_pipe_v1 " to see the simulation for the two cases.

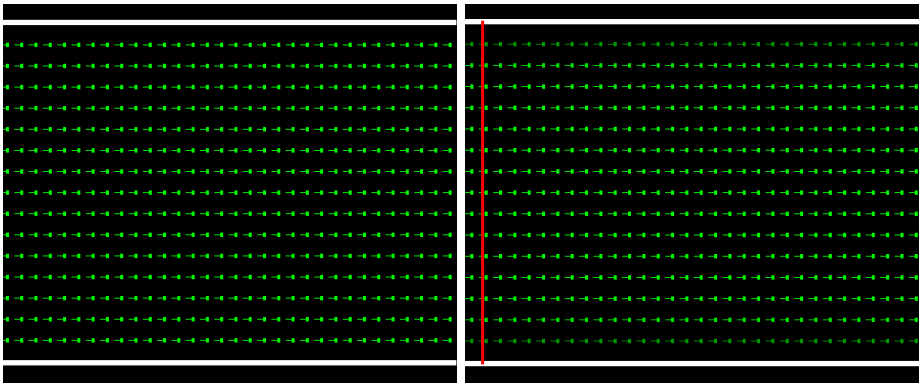

Figure2: The left figure shows the velocity field at t=0, the right figure shows the field at t=50 when the steady state is reached. It also shows the intial placement of the dye.

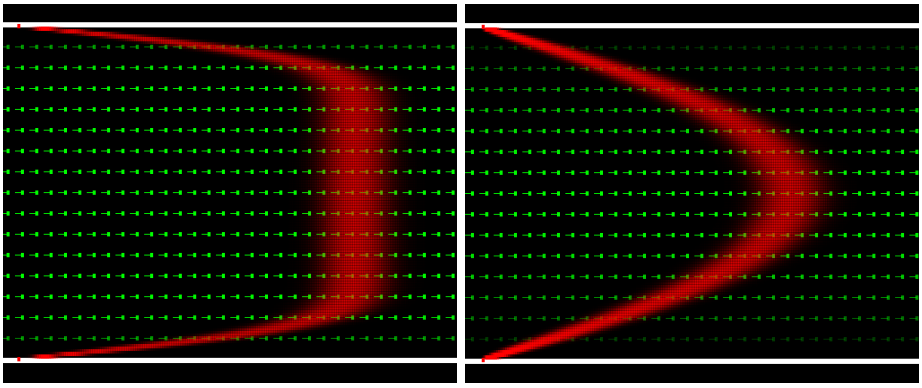

Figure3: Left and right figure show the final field configuration at t=50 for $\eta = 1, 10$ respectively. They also show how the evolution of the dye in the fluid for equal times.

(b): For our next example consider the fluid inside a two dimentional box, the initial field configuration is shown in figure4. This intial condition could arrive when someone blows air in a closed room in flatland. As can be seen from the images taken at different time that this blow of air creates a negative pressure region behind it which sucks in air and a positive pressure region in front which creates a circular outgoing wavefront. After some time two vortecies form which rotate in opposite direction and move forward with the air flow maintaining the distance between them. When these eddies come near the right wall they move apart. The user may run the executable "./air_in_2dbox " to see

the simulation for the velocity field, and "./air_in_2d_box_P " for the simulation of the velocity and the pressure fields.

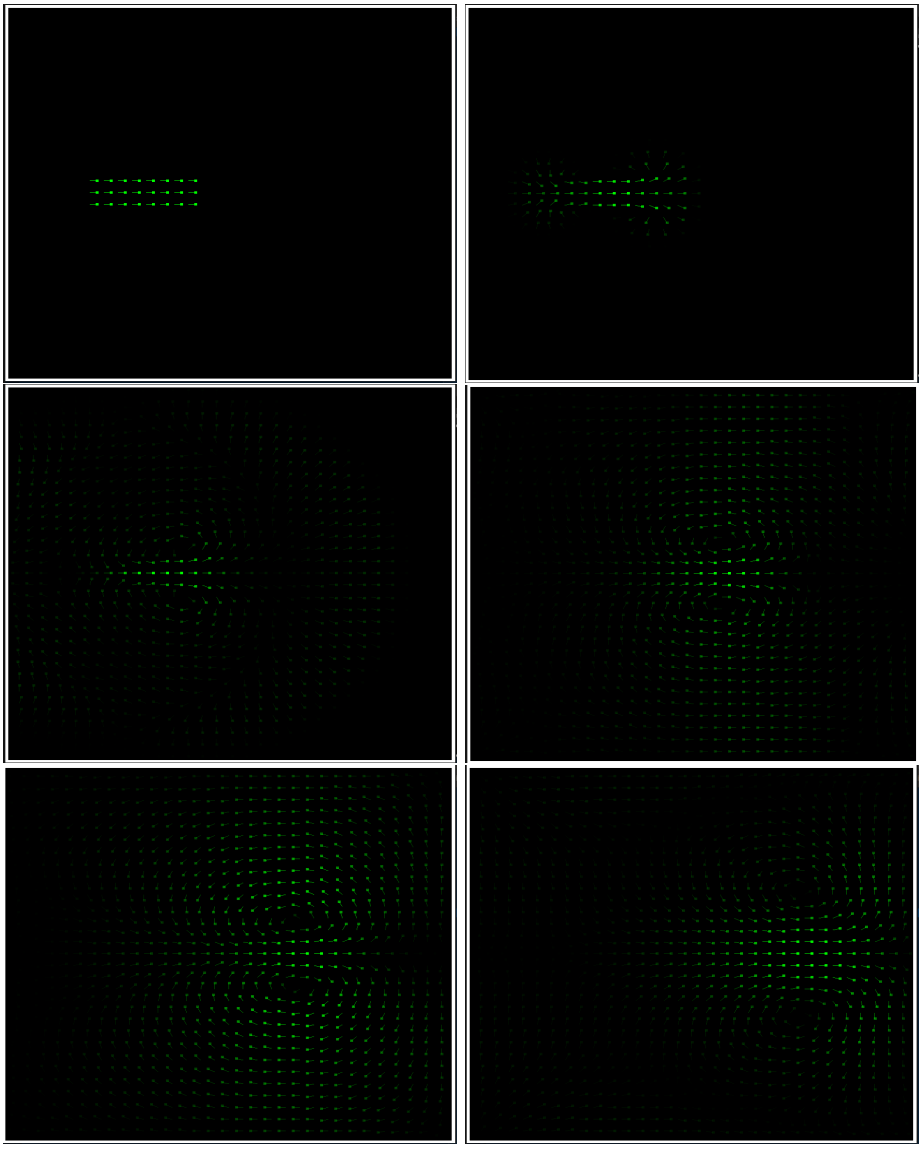

Figure4: First figure shows the initial field configuration, the rest show the snapshot at different times.

(c): For our next example consider the flow inside a pipe around a square block. The field cofigurations at different times are shown in figure 5 below. The user may run the executable file "./flow_around_square" to see the simulation for the velocity field, and "./flow_around_square_P " for the simulation of the velocity and the pressure fields.

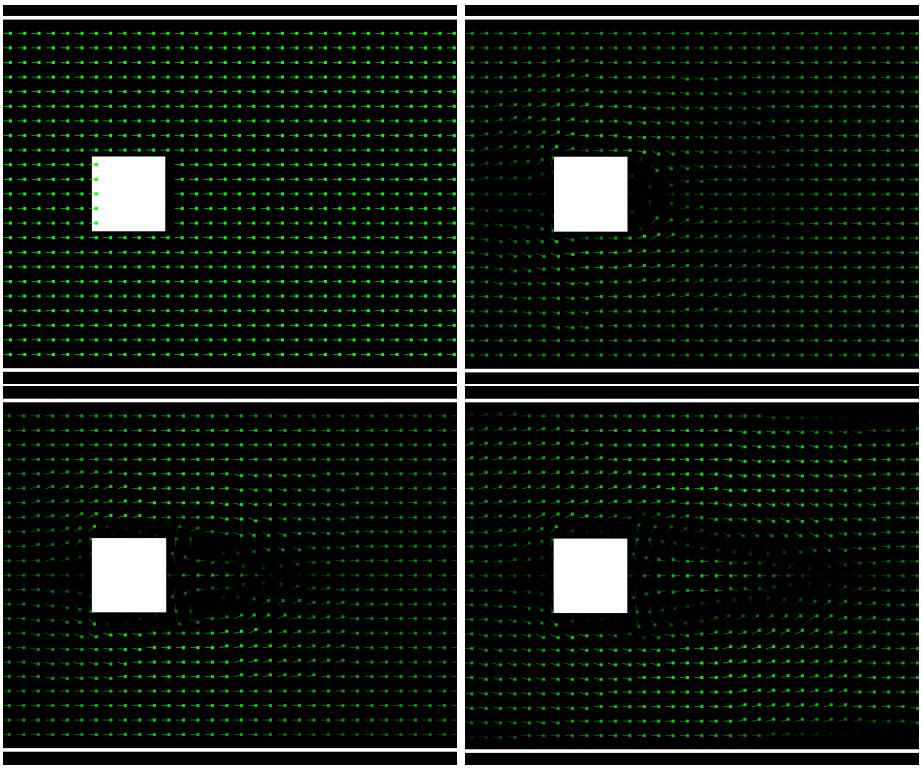

Figure5: First figure shows the initial field configuration, the rest show the snapshot at different times.

Appendix

A.1 Integration of a field on a changing volume

$$\frac{d}{dt}\int_{V(t)} F(x,t)dV = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left[\int_{V(t+\Delta t)} F(x,t+\Delta t)dV - \int_{V(t)} F(x,t)dV\right]$$

where

$$\int_{V(t+\Delta t)} F(x,t+\Delta t)dV \approxeq \int_{V(t)} F(x,t)dV + \int_{V(t)} \Delta t\frac{\partial F(x,t)}{\partial t}dV + \int_{\Delta V} F(x,t)dV + \int_{\Delta V} \Delta t\frac{\partial F(x,t)}{\partial t}dV$$

$$\implies \frac{d}{dt}\int_{V(t)} F(x,t)dV = \lim_{\Delta t\to 0}\left[\int_{V(t)}\frac{\partial F(x,t)}{\partial t}dV + \frac{1}{\Delta t}\int_{\Delta V} F(x,t)dV\right]$$

$$\frac{d}{dt}\int_{V(t)} F(x,t)dV = \int_{V(t)}\frac{\partial F(x,t)}{\partial t}dV + \int_{A(t)} F(x,t)\vec{u}\bullet\hat{n}dA$$

$$\frac{d}{dt}\int_{V(t)} F(x,t)dV = \int_{V(t)}\left(\frac{\partial F(x,t)}{\partial t} + \nabla\left(F(x,t)\vec{u}\right)\right)dV \tag{27}$$

A.2 vorticity transport equation

Taking curl of momentum equation we get

$$\nabla\times\frac{\partial\vec{u}}{\partial t} = -\nabla\times(\vec{u}\bullet\nabla\vec{u}) + \nabla\times\nabla^2\vec{u}$$

$$\frac{\partial}{\partial t}(\nabla\times\vec{u}) = -\nabla\times(\nabla u^2 + \vec{\omega}\times\vec{u}) + \nabla^2(\nabla\times\vec{u})$$

$$\frac{\partial\vec{\omega}}{\partial t} = -\nabla\times(\vec{\omega}\times\vec{u}) + \nabla^2\vec{\omega}$$

$$\frac{\partial\vec{\omega}}{\partial t} = \vec{\omega}\bullet\nabla\vec{u} - \vec{u}\bullet\nabla\vec{\omega} + \nabla^2\vec{\omega} \tag{28}$$

A.3 Pressure poisson equation from incopressible flow

Taking divergence of momentum equation

$$\frac{\partial}{\partial t}(\partial_i u_i) = -(\partial_i u_j)(\partial_j u_i) - (u_j \partial_j)(\partial_i u_i) - \frac{1}{\rho}\partial^2 P + \mu \partial^2(\partial_i u_i) = 0$$

using incompressbility condition we get

$$\partial^2 P = -\rho(\partial_i u_j)(\partial_j u_i) \tag{29}$$

A.4 Stability of explicit numerical solution to convection diffusion equation

Consider the one-dimensional convection diffusion equation.

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = D\frac{\partial^2 T}{\partial x^2} \tag{30}$$

The dicretized version of this equation with FTCS scheme is

$$T_i^{n+1} = T_i^n - \alpha\left(T_{i+1}^n - T_{i-1}^n\right) + \beta\left(T_{i+1}^n - 2T_i^n + T_{i-1}^n\right) \tag{31}$$

where $\alpha = u\frac{\Delta t}{2\Delta x}$ and $\beta = D\frac{\Delta t}{\Delta x^2}$

A numerical solution is said to be stable if the roundoff errors in each step decay over time and do not affect the solution. Let us consider the error at a grid point

$$e_i^n = T_i^n - \bar{T}_i^n \tag{32}$$

Where T_i^n is the exact solution and $\bar{T}_i^n$ is the approximate numerical solution due to roundoff. Since the equation is linear in T the error satisfies the same equation i.e.

$$e_i^{n+1} = e_i^n - \alpha\left(e_{i+1}^n - e_{i-1}^n\right) + \beta\left(e_{i+1}^n - 2e_i^n + e_{i-1}^n\right) \tag{33}$$

Let us take the Fourier transform of the error and since the governing equation is linear let us study only one component of the decomposition i.e.

$$e_i^n \sim g^n(k)e^{i\pi k x_i} \tag{34}$$

$$e_i^{n+1} \sim g^{n+1}(k)e^{i\pi k x_i} \tag{35}$$

Substituting these into the error equation we get

$$\frac{g^{n+1}}{g^n} = \left[(\alpha+\beta)e^{-i\pi k\Delta x} + (1-2\beta) + (\beta-\alpha)e^{i\pi k\Delta x}\right] \tag{36}$$

The condition for stability is that the magnitude of the error should decay with time i.e.

$$\left|\frac{g^{n+1}}{g^n}\right| \leq 1 \tag{37}$$

This gives

$$\left(1 - 4\beta sin^2\left(\frac{\theta}{2}\right)\right)^2 + 4\alpha^2 sin^2\theta \leq 1 \tag{38}$$

Where $\theta = k\pi\Delta x$. The stability condition can also be expressed as

$$0 \leq 4\alpha^2 \leq 2\beta \leq 1 \tag{39}$$

Putting the values for α and β we get

$$0 \leq u^2\frac{\Delta t^2}{\Delta x^2} \leq 2D\frac{\Delta t}{\Delta x^2} \leq 1 \tag{40}$$

Which gives

$$\Delta t \leq \frac{2D}{u^2} \tag{41}$$

$$\Delta x \geq |u|\Delta t \tag{42}$$

Note that for D=0 no value to Δt can give stable solution. To get stable solution in that case one must use either forward or backward difference formula for space derivatives.