Numerical Algorithms

Esteban, Marie Joe, Amirmasoud, Kristin

May 7, 2025

Abstract

This document presents numerical algorithms and methods for solving 2D incompressible Navier-Stokes, with detailed sections on right- and left-hand side formulations, Poisson solver, and validation techniques.

1 Model

We are dealing with the 2D incompressible Navier-Stokes equations, wich gives us:

- For the x-momentum equation:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

- For the y-momentum equation:

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

- Additionally, the continuity equation for incompressible flow ensures mass conservation:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

2 Method

We separate the computation into spatial discretization, time advancement of velocity (predictor), and pressure correction (corrector) to handle the incompressibility constraint. This method decouples the velocity and pressure solutions, solving the momentum equations first to get a tentative velocity (ignoring incompressibility initially), then correcting it by projecting the velocity field onto a divergence-free space via a pressure Poisson equation.

1. Spatial discretization strategies for the right-hand side of the momentum equations. This step involves discretizing the spatial terms (RHS) of the momentum equations, which includes the advection (convection), diffusion, and pressure gradient terms.

2. Then we handle the time derivative (left-hand side) of the momentum equations by simply taking the spatially discretized equations and advancing the velocity fields (u and v) in time without considering the continuity constraint ($\nabla u = 0$). This is the predictor step, computing a tentative velocity field based on the current state, which performs a transport phase that accounts for advection and diffusion without satisfying incompressibility.

3. Finally a new pressure field is computed to enforce incompressibility by solving a Poisson equation for pressure (relating pressure at any point to the entire velocity field). Then the velocity field is corrected to satisfy the continuity equation.

This method was prefered to a coupled (monolithic) method that solves the velocity and pressure fields simultaneously in a fully coupled system at each time step, since it decreases computational cost, although it cannot achieve higher-order temporal accuracy as easily. This projection method also makes it easier to analyze global error and convergence rate, and separate the work in different tasks.

3 Spatial Discretization

3.1 Finite Difference

3.1.1 Central Scheme

3.1.2 Upwind Scheme

3.2 Finite Volume

4 Prediction Step

The prediction step in a pressure-based Navier-Stokes solver estimates the intermediate velocity field before applying the pressure correction. This step advances the momentum equations without enforcing the incompressibility constraint.

4.1 Explicit Schemes

Explicit time integration schemes compute the velocity update using only known values from the current time step. For instance, the Forward Euler method updates the velocity as

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \left(-(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \nu \nabla^2 \mathbf{u}^n \right), \tag{1}$$

where $\mathbf{u} = (u, v)$ is the velocity vector, ν is the kinematic viscosity, and Δt is the time step size.

In the provided code, this update is implemented as:

```
u_new[1:-1, 1:-1] = u[1:-1, 1:-1] + dt * dudt1[1:-1, 1:-1]
v_new[1:-1, 1:-1] = v[1:-1, 1:-1] + dt * dvdt1[1:-1, 1:-1]
```

Here, `dudt1` and `dvdt1` contain the explicit evaluation of the right-hand side terms (advection and diffusion) at time step n.

4.2 Semi-implicit Schemes

Semi-implicit schemes treat some terms explicitly and others implicitly to improve numerical stability without the full cost of solving a global linear system. Typically, the nonlinear advection terms are treated explicitly, while the linear diffusion terms are treated implicitly.

4.2.1 Explicit Advection and Implicit Diffusion

This approach discretizes the momentum equation as

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = -\left((\mathbf{u} \cdot \nabla)\mathbf{u}\right)^n + \nu \nabla^2 \mathbf{u}^{n+1}. \tag{2}$$

The diffusion term $\nu \nabla^2 \mathbf{u}^{n+1}$ is treated implicitly, while the advection term is evaluated explicitly at time n.

For the u-component of velocity at grid point (i, j), using a finite difference discretization on a uniform Cartesian grid with spacings Δx and Δy, the update formula reads

$$u_{i,j}^{n+1} = \frac{u_{i,j}^n + \Delta t \left[-u_{i,j}^n \frac{u_{i,j}^n - u_{i,j-1}^n}{\Delta x} - v_{i,j}^n \frac{u_{i,j}^n - u_{i-1,j}^n}{\Delta y} \right] + \Delta t \nu \left(\frac{u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} \right)}{1 + 2\Delta t \nu \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)}. \tag{3}$$

This formula corresponds to a Jacobi iteration for the implicit diffusion step, avoiding the need to solve a global linear system explicitly.

In code, this update is implemented as:

```
u_new[1:-1, 1:-1] = (
    u[1:-1, 1:-1] + dt * adv_u +
    dt * nu * ((u[1:-1, 2:] + u[1:-1, 0:-2]) / dx**2 +
               (u[2:, 1:-1] + u[0:-2, 1:-1]) / dy**2)
) / (1 + 2 * dt * nu * (1/dx**2 + 1/dy**2))
```

where `adv_u` contains the explicit advection terms evaluated at time n.

This semi-implicit scheme allows for larger stable time steps compared to fully explicit methods while remaining computationally efficient and relatively straightforward to implement.

5 Projection Step

6 Validation

6.1 Taylor-Green Vortex

The Taylor-Green Vortex (TGV) is a canonical benchmark problem for validating numerical solvers of the incompressible Navier-Stokes equations due to its exact analytical solution. This solution describes a decaying vortex flow in a periodic domain, allowing for direct comparison between simulated and analytical results, given by:

$$u(x, y, t) = \sin(\pi x)\cos(\pi y)\exp\left(-2\nu t\pi^2\right),$$

$$v(x, y, t) = -\cos(\pi x)\sin(\pi y)\exp\left(-2\nu t\pi^2\right),$$

$$p(x, y, t) = -\frac{\rho}{4}\left(\cos(2\pi x) + \cos(2\pi y)\right)\exp\left(-4\nu t\pi^2\right),$$

for $x, y \in (-1, 1)^2$, where u and v are the velocity components in the x- and y-directions, respectively, p is the pressure, ν is the kinematic viscosity, and t is the time. The factor π^2 in the exponential decay term accounts for the specific domain scaling used in the implementation.

Validation is performed by computing the kinetic energy of the simulated flow and comparing it with the analytical kinetic energy at a given time, to obtain a quantitative measure of the solver's accuracy. Periodic boundary conditions are applied to maintain the flow's continuity across the domain boundaries.

6.2 Lid-Driven Cavity

The Lid-Driven Cavity flow is another benchmark for validating numerical solvers, particularly for assessing the handling of boundary conditions and steady-state flow features in a confined domain. Unlike the Taylor-Green Vortex, it lacks an analytical solution, so validation relies on comparison with established reference data from literature or prior simulations.

The problem setup involves a square domain of unit length with no-slip boundary conditions on three walls ($u = v = 0$) and a moving top wall (lid) with a constant tangential velocity ($u = 1, v = 0$). The flow is driven by the lid motion, resulting in a primary recirculating vortex and, at higher Reynolds numbers, secondary vortices in the corners. We use Neumann boundary conditions for pressure ($\frac{\partial p}{\partial n} = 0$) at the walls.

Here, validation focuses on the stream function's minimum value, which is indicative of the primary vortex strength in the cavity. According to Logg et al. (2012) this value after 2.5s is -0.061076605. The comparison with a reference value allows for an assessment of numerical accuracy and convergence towards expected flow behavior.

7 Appendix

7.1 Semi-Implicit Scheme Derivation

We begin with the 2D incompressible Navier-Stokes momentum equation for the u-velocity component (ignoring pressure for now):

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

We discretize in time using a semi-implicit scheme:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} + \left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right)^{n} = \nu\nabla^2 u_{i,j}^{n+1}$$

Use first-order upwind or central differences for the advection terms at time n:

$$\left(u\frac{\partial u}{\partial x}\right)_{i,j}^{n} \approx u_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta x}$$

$$\left(v\frac{\partial u}{\partial y}\right)_{i,j}^{n} \approx v_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta y}$$

So the full advection term becomes:

$$\mathsf{Adv}_{i,j}^{n} = u_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta x} + v_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta y}$$

Multiply both sides by Δt and rearrange:

$$u_{i,j}^{n+1} - \Delta t \cdot \nu\nabla^2 u_{i,j}^{n+1} = u_{i,j}^{n} - \Delta t \cdot \mathsf{Adv}_{i,j}^{n}$$

Discretize the Laplacian:

$$\nabla^2 u_{i,j}^{n+1} \approx \frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2}$$

Plug into time discretization:

$$u_{i,j}^{n+1} - \Delta t\cdot\nu\left(\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2}\right) = u_{i,j}^{n} - \Delta t\cdot\mathsf{Adv}_{i,j}^{n}$$

Solve for $u_{i,j}^{n+1}$

$$\left(1 + 2\Delta t\nu\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)\right)u_{i,j}^{n+1} = u_{i,j}^{n} + \Delta t \cdot \left[-u_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta x} - v_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta y}\right] + \Delta t \cdot \nu\left(\frac{u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2}\right)$$

which gives us

$$u_{i,j}^{n+1} = \frac{u_{i,j}^{n} + \Delta t \left[-u_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta x} - v_{i,j}^{n} \cdot \frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta y} \right] + \Delta t \cdot \nu \left(\frac{u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} \right)}{1 + 2\Delta t \nu \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)}$$