Flow Past a Cylinder: Solving the Two-dimensional Incompressible Navier-Stokes Equations with MOLE

Yiyue Feng and Miguel A. Dumett

May 9, 2025

Publication Number: CSRCR2025-02

Computational Science & Engineering Faculty and Students Research Articles

Database Powered by the Computational Science Research Center Computing Group & Visualization Lab

COMPUTATIONAL SCIENCE & ENGINEERING

SAN DIEGO STATE UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

Flow Past a Cylinder: Solving the Two-dimensional Incompressible Navier-Stokes Equations with MOLE

Yiyue Feng* and Miguel A. Dumett †‡

May 9, 2025

Abstract

Flow past a cylinder is a standard test in fluid dynamics to check how well a numerical method converges and stays stable. In this project, we solve the two-dimensional incompressible Navier–Stokes equations using MOLE with the projection method. The MOLE library applies higher-order mimetic divergence, gradient, and Laplacian operators, and offers an efficient way to interpolate values on staggered grids. Our results show a clear boundary layer near the channel boundaries and vortex shedding behind the cylinder.

1 Problem setting

In the following subsections we introduce the two-dimensional (2D) governing equations, including parameters and domain, as well as the initial and boundary conditions for solving the 2D incompressible Navier-Stokes equations.

*Computational Science PhD Program at San Diego State University (yfenf9707@sdsu.edu).

†Editor: Jose E. Castillo

‡Computational Science Research Center at San Diego State University (mdumett@sdsu.edu).

1.1 Governing equations, parameters and domain

The equations below represent the 2D incompressible Na-vier–Stokes equations without body forces in differential form, comprising mass conservation (Equation 1) and momentum conservation (Equation 2):

$$\nabla \cdot \vec{U} = 0 \tag{1}$$

$$\frac{\partial \vec{U}}{\partial t} + \left(\vec{U} \cdot \nabla\right) \vec{U} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \vec{U} \tag{2}$$

A divergence (conservative) form ($\vec{U} = (u, v)$) can be written as (in primitive variables):

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3}$$

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{4}$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{5}$$

Reynolds Number is defined as in Equation 6:

$$Re = \frac{U_\infty D}{\nu}, \tag{6}$$

where U_∞ is the free-stream velocity, D is the characteristic length (here, the cylinder diameter/size), and ν is the kinematic viscosity.

The Reynolds number quantifies the ratio of inertial forces to viscous forces in a flow, and vortex shedding starts around $Re \simeq 47$.

In this project, we will set $Re = 100$.

For the flow field size (domain), we assume the flow going along the x-axis from left to right, with the range:

$$x \in [0, 6] \tag{7}$$

$$y \in [-1, 1] \tag{8}$$

The center of the square cylinder is located at $(x, y) = (1, 0)$, with a side length $D = 0.6$.

We set $U_\infty = 1$, which is also the initial condition we will define later. So ν can be calculated accordingly based on Re, D and U_∞.

1.2 Initial conditions

The initial condition for the velocity field $\vec{U}$ is all zero.

1.3 Boundary conditions

First, the velocity field.

At the left inlet ($x = 0$) we have the Dirichlet boundary condition:

$$\vec{U} = (u, v) = (U_\infty, 0) \tag{9}$$

At the right outlet ($x = 6$) we have the Neumann boundary condition:

$$\frac{\partial \vec{U}}{\partial x} = 0 \tag{10}$$

At the top and bottom walls, since they are not far away enough from the cylinder, we apply the "no-slip" condition, where:

$$\vec{U} = (u, v) = (0, 0) \tag{11}$$

But if the top and bottom walls are "far field", we can treat them as "free-slip":

$$\frac{\partial u}{\partial y} = 0 \tag{12}$$

$$v = 0 \tag{13}$$

Or even just treat them as free stream:

$$\vec{U} = (u, v) = (U_\infty, 0) \tag{14}$$

For cylinder walls, we apply the "no-slip" condition, where:

$$\vec{U} = (u, v) = (0, 0) \tag{15}$$

Second, the pressure field.

We can usually leave the input pressure unspecified.

And for the outlet, we have a Dirichlet reference for pressure, which is often set to be zero (gauge):

$$p = 0 \tag{16}$$

For the top and bottom walls, we apply a Neumann boundary condition:

$$\frac{\partial p}{\partial y} = 0 \tag{17}$$

2 Methods

In this section, the projection method is presented, together with a staggered grid and some consideration about time step constraint of the time derivative discretizations.

2.1 Introduction to projection method

The projection method (also called the fractional step or Chorin's method) is a way to decouple the incompressible Navier–Stokes equationsdo nothat we do not have to solve a fully coupled velocity–pressure system at each time step. Instead, we "project" an intermediate velocity onto the divergence-free subspace by a pressure correction.

This splits one coupled system into:

- a momentum update (no pressure)
- a Poisson equation for pressure
- a velocity correction

Equations in each steps of the projection method are listed below.

1. Tentative velocity field $\vec{U^*}$ explicit prediction:

$$\frac{\vec{U^*} - \vec{U}}{\triangle t} = -\left(\vec{U} \cdot \nabla\right)\vec{U} + \nu\nabla^2\vec{U} \tag{18}$$

And we can re-write this in primitive variables form:

$$\frac{u^* - u}{\triangle t} = -(\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y}) + \nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{19}$$

$$\frac{v^* - v}{\triangle t} = -(\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y}) + \nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) \tag{20}$$

2. Pressure Poisson Equation:

$$\nabla^2 p_{new} = \frac{\rho}{\triangle t}\vec{U^*} \tag{21}$$

3. Velocity correction:

$$\vec{U_{new}} = \vec{U^*} - \frac{\triangle t}{\rho}\nabla p_{new} \tag{22}$$

2.2 Space discretizations: staggered grids

MOLE [1, 3] is an implementation of the Corbino-Castillo mimetic differences [2]. Mimetic difference utilizes a staggered grid. It builds discrete analogs of the vector calculus differential operators that are high-order accurate. These discretized operators are constructed such they approximate the extended Gauss divergence theorem.

We apply staggered grids, in which different flow variables are stored at different locations within each computational cell.

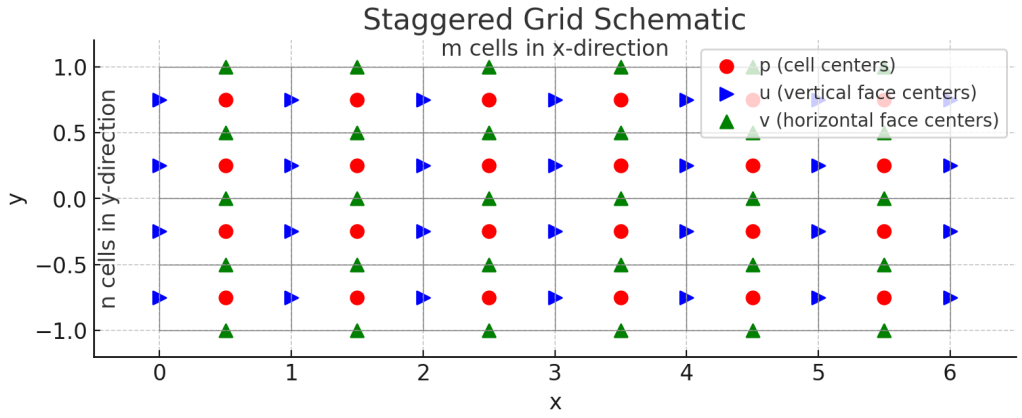

Figure 1: Staggered grids

As illustrated in Fig. 1, pressure lives at cell centers, u components live on the vertical cell faces, and v components live on the horizontal cell faces.

2.3 Time discretizations

For the selection of time step, we should consider multiple conditions.

1. Convective CFL condition:

$$\triangle t \leq \min \left\{ \frac{CFL_{max} \triangle x}{|u|_{max}}, \frac{CFL_{max} \triangle y}{|v|_{max}} \right\} \tag{23}$$

CFL_{max} is set to be 0.5 here.

2. Diffusive stability:

$$\triangle t \leq \frac{1}{2\nu(\frac{1}{\triangle x^2} + \frac{1}{\triangle y^2})} \tag{24}$$

2.4 MOLE: interpolation and mimetic operators

With the application of the staggered grid, there are two types of ways to store flow variables: either cell center, or cell faces.

When storing cell-center data, the Mimetic Operators Library Enhanced (MOLE) also includes boundary values. For the u components, it stores the values on the left and right faces in addition to the m interior cell centers, for a total of $m + 2$ values. Likewise, the v components has $n + 2$ values. Consequently, the full data array forms an $(n + 2) \times (m + 2)$ matrix. MOLE will reshape it into a column vector of length $(m + 2)(n + 2)$ to store.

When storing cell-face data, the array dimensions differ for the u and v components. The u components live on the $m + 1$ vertical faces of each of the n rows of cell centers, yielding an $n \times (m + 1)$ matrix. The v components live on the $n + 1$ horizontal faces of each of the m columns of cell centers, yielding an $(n + 1) \times m$ matrix. MOLE then reshapes each matrix into a single column vector for further computation.

In the code of this project, we use `variableName_flat` to represent these column vectors storing cell-center values, `variableName_on_u` and `variableName_on_v` to represent these column vectors storing cell-face values on vertical faces and horizontal faces.

Listing 1: Matlab code implementation

```
% mimetic operators
L = lap2D(k, m, dx, n, dy);
D = div2D(k, m, dx, n, dy);
G = grad2D(k, m, dx, n, dy);

% interpolation
I = interpolCentersToFacesD2D(k, m, n); % center to
    face
II = interpolFacesToCentersG2D(k, m, n); % face to
    center

% 2D Staggered grid
x_start = 0;    x_end = 6;
y_start = -1;   y_end = 1;
Lx = x_end - x_start; Ly = y_end - y_start;
dx = (x_end - x_start)/m; dy = (y_end - y_start)/n;
xgrid = [0 dx/2:dx:Lx-dx/2 Lx]; % Staggered grid x
ygrid = [0 dy/2:dy:Ly-dy/2 Ly]; % Staggered grid y
[Y, X] = meshgrid(ygrid, xgrid);
U = 0.*X;          V = U;
U_flat = U(:);   V_flat = V(:);

U_stag = I * [U_flat;U_flat]; % center to face
U_on_u = U_stag(1:(m+1)*n); U_on_v = U_stag((m+1)*n+1:
    end);
V_stag = I * [V_flat;V_flat]; % face to center
V_on_u = V_stag(1:(m+1)*n); V_on_v = V_stag((m+1)*n+1:
    end);
UU_on_u = U_on_u.*U_on_u; UV_on_u = U_on_u.*V_on_u;
VV_on_v = V_on_v.*V_on_v; UV_on_v = U_on_v.*V_on_v;

% velocity field prediction
U_star_flat = U_flat - dt*D*[UU_on_u;UV_on_v] + dt*nu*L
    *U_flat;
V_star_flat = V_flat - dt*D*[UV_on_u;VV_on_v] + dt*nu*L
    *V_flat;
```

```
31
32 % velocity correction
33 U_V_flat = [U_star_flat; V_star_flat] - (dt/rho)*(II*G*
       p_new_flat);
34
35 U = reshape(U_flat, m+2, n+2);V = reshape(V_flat, m+2,
       n+2);
```

3 Results and conclusion

As shown in Fig. 2, a well-defined boundary layer forms as the flow develops. In Fig. 3, clear vortex shedding appears. As *Re* increases further, these vortices become progressively stronger.

References

[1] MOLE: Mimetic Operators Library Enhanced, June 2023.

[2] J. Corbino and J. E. Castillo. High-order mimetic finite-difference operators satisfying the extended gauss divergence theorem. *Journal of Computational and Applied Mathematics*, 364, 01 2020.

[3] The MOLE team. MOLE: Mimetic Operators Library Enhanced. https://csrc-sdsu.github.io/mole/, 2025. Accessed: 2025-04-30.

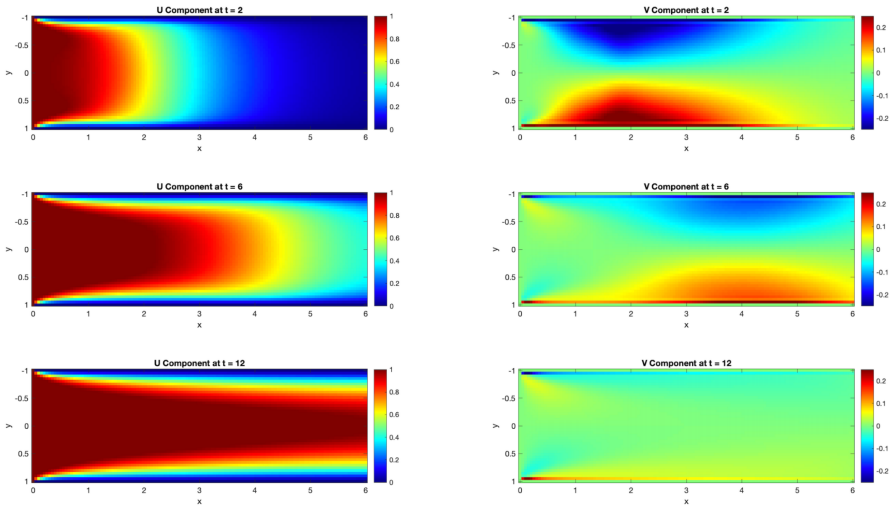

Figure 2: Channel flow developing: t = 2, 6, 12

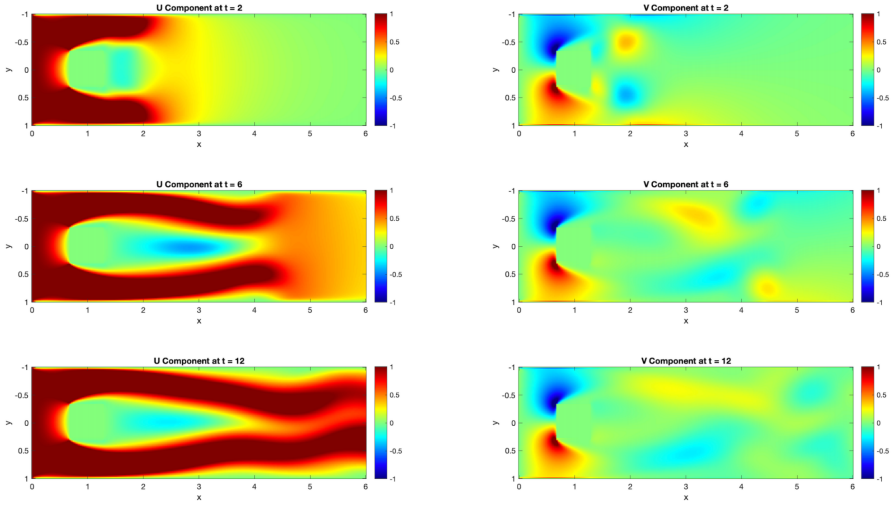

Figure 3: Flow past cylinder: t = 2, 6, 12