Penalty Method for Enforcing Zero Velocity in a Subdomain: Application to the Driven Cavity Problem

1 Strong Formulation of the Incompressible Navier-Stokes Equations

The incompressible Navier-Stokes equations in a domain $\Omega \subset \mathbb{R}^2$ are given by:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \Delta \mathbf{u} + \mathbf{f} \quad \text{in } \Omega \times (0, T] \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \times (0, T] \tag{2}$$

where:

- $\mathbf{u} = (u, v)^T$ is the velocity field
- p is the pressure (divided by density)
- ν is the kinematic viscosity
- $\mathbf{f}$ is the body force

In component form, for $\mathbf{u} = (u, v)$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{3}$$

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

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{5}$$

2 The Penalty Method for Immersed Boundaries

2.1 Problem Statement

We want to enforce $\mathbf{u} = \mathbf{0}$ in a subdomain $\Omega_s \subset \Omega$ (e.g., a circular region representing a solid obstacle). This is commonly used in:

- Immersed boundary methods
- Fluid-structure interaction
- Modeling obstacles without body-fitted meshes

2.2 Penalty Formulation

The penalty method adds a forcing term to the momentum equation that penalizes any deviation from the desired velocity in the solid region:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \Delta \mathbf{u} - \eta \chi_s(\mathbf{x})\mathbf{u} \tag{6}$$

where:

- $\eta > 0$ is the **penalty coefficient** (large value)
- $\chi_s(\mathbf{x})$ is the **characteristic function** of the solid region:

$$\chi_s(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_s \\ 0 & \text{if } \mathbf{x} \notin \Omega_s \end{cases} \tag{7}$$

2.3 Physical Interpretation

The penalty term $-\eta \chi_s \mathbf{u}$ acts as a **Darcy-like drag force** that resists fluid motion in the solid region. As $\eta \to \infty$, the velocity $\mathbf{u} \to \mathbf{0}$ in Ω_s.

The modified momentum equation can be rewritten as:

$$\frac{\partial \mathbf{u}}{\partial t} = \underbrace{-(\mathbf{u} \cdot \nabla)\mathbf{u}}_{\text{Convection}} \underbrace{-\nabla p}_{\text{Pressure}} + \underbrace{\nu \Delta \mathbf{u}}_{\text{Diffusion}} \underbrace{-\eta \chi_s \mathbf{u}}_{\text{Penalty}} \tag{8}$$

2.4 Definition of the Solid Region

For a circular obstacle centered at (x_c, y_c) with radius R:

$$\chi_s(\mathbf{x}) = \begin{cases} 1 & \text{if } (x - x_c)^2 + (y - y_c)^2 \leq R^2 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

3 Numerical Discretization

3.1 Projection Method (Chorin's Splitting)

The original code uses a projection method. The time advancement consists of two steps:

Step 1: Predictor (compute intermediate velocity $\tilde{\mathbf{u}}$)

$$\frac{\tilde{u} - u^n}{\Delta t} = -\left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right)^n + \nu \Delta u^n \tag{10}$$

$$\frac{\tilde{v} - v^n}{\Delta t} = -\left(u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right)^n + \nu \Delta v^n \tag{11}$$

Step 2: Pressure Projection (enforce incompressibility)

$$\Delta p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}} \tag{12}$$

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \Delta t \nabla p^{n+1} \tag{13}$$

3.2 Adding the Penalty Term

3.2.1 Explicit Treatment (Unstable)

A naive explicit discretization would be:

$$\tilde{u} = u^n + \Delta t \left(\text{Conv} + \text{Diff} - \eta \chi_s u^n\right) \tag{14}$$

However, for large η, the term $\Delta t \cdot \eta$ becomes very large (e.g., $\Delta t = 0.01$, $\eta = 10^4$ gives $\Delta t \cdot \eta = 100$), causing **numerical instability**.

3.2.2 Implicit Treatment (Stable)

To ensure stability, we treat the penalty term implicitly:

$$\frac{\tilde{u} - u^n}{\Delta t} = \text{Conv}^n + \text{Diff}^n - \eta \chi_s \tilde{u} \tag{15}$$

Rearranging:

$$\tilde{u} + \Delta t \cdot \eta \chi_s \tilde{u} = u^n + \Delta t \left(\text{Conv}^n + \text{Diff}^n\right) \tag{16}$$

$$\tilde{u} \left(1 + \Delta t \cdot \eta \chi_s\right) = u^n + \Delta t \left(\text{Conv}^n + \text{Diff}^n\right) \tag{17}$$

Therefore:

$$\boxed{\tilde{u} = \frac{u^n + \Delta t \left(\text{Conv}^n + \text{Diff}^n\right)}{1 + \Delta t \cdot \eta \chi_s}} \tag{18}$$

Key observations:

- In the fluid region ($\chi_s = 0$): $\tilde{u} = u^n + \Delta t(\text{Conv} + \text{Diff})$ (unchanged)
- In the solid region ($\chi_s = 1$): $\tilde{u} = \frac{u^n + \Delta t(\text{Conv}+\text{Diff})}{1+\Delta t \cdot \eta}$
- As $\eta \to \infty$: $\tilde{u} \to 0$ in the solid region
- This is **unconditionally stable** for any $\eta > 0$

4 Algorithm Summary

Algorithm 1 Projection Method with Penalty

1: **Input:** u^n, v^n, penalty mask χ_s, penalty coefficient η
2: Apply boundary conditions
3: Compute convection and diffusion terms
4: Compute intermediate velocity (without pressure):

$$\tilde{u}^* = u^n + \Delta t(\text{Conv}_u + \text{Diff}_u)$$

$$\tilde{v}^* = v^n + \Delta t(\text{Conv}_v + \text{Diff}_v)$$

5: **Apply implicit penalty:**

$$\tilde{u} = \frac{\tilde{u}^*}{1 + \Delta t \cdot \eta \cdot \chi_s}$$

$$\tilde{v} = \frac{\tilde{v}^*}{1 + \Delta t \cdot \eta \cdot \chi_s}$$

6: Solve pressure Poisson equation: $\Delta p = \frac{1}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}$
7: Project to divergence-free velocity:

$$u^{n+1} = \tilde{u} - \Delta t \frac{\partial p}{\partial x}, \quad v^{n+1} = \tilde{v} - \Delta t \frac{\partial p}{\partial y}$$

5 Code Modifications

5.1 New Function: `define_penalty_region`

This function creates the characteristic function χ_s on the computational grid:

```
def define_penalty_region(nx, ny, region_type='circle',
                          x_center=0.5, y_center=0.5, radius=0.1):
    mask = np.zeros((nx, ny))
    x = np.linspace(0, 1, nx)
    y = np.linspace(0, 1, ny)
    xx, yy = np.meshgrid(x, y)

    if region_type == 'circle':
        mask = ((xx - x_center)**2 + (yy - y_center)**2 <= radius**2)

    return mask.astype(float)
```

5.2 Modified `solve` Function

Added penalty method setup:

```
# Penalty method setup
```

```
penalty_mask_u = np.zeros((ny, nx-1))
penalty_mask_v = np.zeros((ny-1, nx))
penalty_coeff = 1e4  # eta

if penalty_method and penalty_param is not None:
    region_type, region_args = penalty_param
    base_mask = define_penalty_region(nx, ny, region_type, *region_args)
    # Masks adapted to staggered grid
    penalty_mask_u = base_mask[:, :-1]  # u-component grid
    penalty_mask_v = base_mask[:-1, :]  # v-component grid
```

Modified momentum equations with implicit penalty:

```
# Compute intermediate velocity
ut[1:ny+1, 2:nx+1] = u[1:ny+1, 2:nx+1] + dt * (convection + diffusion)

# Apply implicit penalty (Equation 12)
if penalty_method:
    ut[1:ny+1, 2:nx+1] = ut[1:ny+1, 2:nx+1] / (1.0 + dt * penalty_coeff * penalty_mask_u)
```

5.3 Staggered Grid Considerations

On a staggered (MAC) grid:

- Pressure is defined at cell centers: $p_{i,j}$
- u-velocity is defined at vertical cell faces: $u_{i+1/2,j}$
- v-velocity is defined at horizontal cell faces: $v_{i,j+1/2}$

The penalty mask must be interpolated to the appropriate velocity locations:

$$\chi_s^u = \chi_s[:, :-1] \quad \text{(shape: } n_y \times (n_x - 1)) \tag{19}$$

$$\chi_s^v = \chi_s[:-1, :] \quad \text{(shape: } (n_y - 1) \times n_x) \tag{20}$$

6 Parameter Selection

6.1 Penalty Coefficient η

- **Too small** ($\eta \sim 1$): Velocity not effectively suppressed
- **Optimal range** ($\eta \sim 10^3 - 10^5$): Good enforcement with stability
- **Very large** ($\eta \to \infty$): Exact enforcement, but may cause stiff behavior

With implicit treatment, larger values of η can be used without stability issues.

6.2 Error Estimate

The velocity in the penalized region satisfies approximately:

$$|\mathbf{u}|_{\Omega_s} \sim \mathcal{O}\left(\frac{1}{\eta \Delta t}\right) \tag{21}$$

For $\eta = 10^4$ and $\Delta t = 0.01$: $|\mathbf{u}|_{\Omega_s} \sim \mathcal{O}(10^{-2})$

7 Extensions

7.1 Prescribing Non-Zero Velocity

To enforce $\mathbf{u} = \mathbf{u}_d$ instead of $\mathbf{u} = \mathbf{0}$:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \Delta \mathbf{u} - \eta \chi_s (\mathbf{u} - \mathbf{u}_d) \tag{22}$$

The implicit discretization becomes:

$$\tilde{u} = \frac{u^n + \Delta t(\text{Conv} + \text{Diff}) + \Delta t \cdot \eta \chi_s u_d}{1 + \Delta t \cdot \eta \chi_s} \tag{23}$$

7.2 CFL Stability Constraint

Critical Issue: When prescribing large velocities $|\mathbf{u}_d| \gg 1$, the explicit treatment of the convection term imposes a stability constraint:

$$\text{CFL} = \frac{|\mathbf{u}|_{\max} \Delta t}{\min(\Delta x, \Delta y)} \leq C_{\text{crit}} \tag{24}$$

where $C_{\text{crit}} \approx 0.5$ for explicit schemes. This means:

$$\Delta t \leq \frac{C_{\text{crit}} \cdot \min(\Delta x, \Delta y)}{|\mathbf{u}_d|} \tag{25}$$

Example: For $u_d = -2.0$, $\Delta x = 0.02$, we need:

$$\Delta t \leq \frac{0.5 \times 0.02}{2.0} = 0.005 \tag{26}$$

If $\Delta t = 0.01$ is used, the solution becomes unstable and diverges.

7.2.1 Solutions for High Velocity Prescription

Option 1: Reduce Time Step (Simplest)

Adaptively adjust the time step based on the maximum velocity:

$$\Delta t = C_{\text{CFL}} \frac{\min(\Delta x, \Delta y)}{\max(|\mathbf{u}_d|, |\mathbf{u}|_{\text{flow}})} \tag{27}$$

where $C_{\text{CFL}} = 0.3 - 0.5$ for safety.

Option 2: Semi-Implicit Convection

Treat the penalty-induced velocity implicitly in the convection term:

$$\text{Conv} = -\left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) \tag{28}$$

In the penalty region, use u_d for the convection velocity instead of u^n.

Option 3: Ramp Up Velocity Gradually

Instead of immediately imposing $u_d = -2$, use a smooth ramp:

$$u_d(t) = u_d^{\text{final}} \cdot \min\left(1, \frac{t}{T_{\text{ramp}}}\right) \tag{29}$$

This allows the flow to adjust gradually, avoiding initial transients.

7.3 Smooth Transition

Instead of a sharp mask, use a smooth transition:

$$\chi_s(\mathbf{x}) = \frac{1}{2}\left(1 - \tanh\left(\frac{d(\mathbf{x}) - R}{\epsilon}\right)\right) \tag{30}$$

where $d(\mathbf{x})$ is the distance from the obstacle center and ϵ controls the transition width.