# 2D Navier-Stokes Solver Code

This document provides a detailed explanation of the Python code that solves the two-dimensional Navier-Stokes equations for incompressible flow in a rectangular domain using the finite-difference method.

## 1. Introduction

The code aims to solve the Navier-Stokes equations to simulate fluid flow in a two-dimensional domain. It employs the finite-difference method to calculate the velocity and pressure fields. The results are visualized using velocity vector fields and pressure contours.

## 2. Domain and Parameters

The domain dimensions and resolution are defined as follows:

* **Domain Dimensions**: Depending on the aspect ratio (`1:1` or `2:1`), the length of the domain (`Lx`) is either 1 or 2 units, while the height (`Ly`) is 1 unit.
* **Grid Resolution**: The grid is discretized into `nx` points in the x-direction and `ny` points in the y-direction. The grid spacing is calculated as `dx = Lx / (nx - 1)` and `dy = Ly / (ny - 1)`.
* **Physical Properties**: The fluid properties include:
* **nu**: Kinematic viscosity, representing the fluid's resistance to deformation.
* **rho**: Density, representing the mass per unit volume of the fluid.

## 3. Boundary Conditions

The boundary conditions are implemented to simulate specific flow scenarios:

* **Inlet (Left Boundary):** The velocity components (`u`, `v`) are prescribed as `**u\_inlet**` and `**v\_inlet**`.
* **Outlet (Right Boundary)**: Velocities (`u`, `v`) are set to zero, indicating a free outflow.
* **Top and Bottom Walls**: No-slip conditions are enforced by setting `**u**` and `**v**` to zero.

## 4. Pressure Poisson Equation

The pressure Poisson equation ensures that the velocity field satisfies the continuity equation for incompressible flow. It uses an iterative finite-difference approach to solve for the pressure field, updating it at each time step.

## 5. Main Solver Loop

The main loop iteratively calculates the velocity and pressure fields over the specified number of time steps (`steps`).

The calculations include:

* **Source Term (`b**`): Computes the divergence of the velocity field and other terms.
* **Velocity Updates**: Accounts for advection, diffusion, and pressure gradient forces.
* **Pressure Update**: Solves the Poisson equation to correct velocity fields.

## 6. Visualization

The code visualizes the flow using matplotlib. It generates:

* **Pressure Contours**: Displayed using filled contours with a color map.
* **Velocity Vectors**: Quiver plots show the direction and magnitude of the velocity field.

## 7. Output

The final velocity **(`u`, `v`)** and pressure **(`p`)** fields are returned for further analysis or visualization.