Physics-Informed Neural Networks for Solving Navier-Stokes Equations: Modeling Fluid Flow Around a Cylinder

Group Members: Ahmed-Wassim Benzerga
Willow Stenglein
Mateo Garcia-Rosell

October 6, 2024

1 A Primer: Navier-Stokes Equations

The Navier-Stokes equations describe the motion of fluid substances like liquids and gases. These equations are fundamental in fluid mechanics and are used to model various phenomena such as ocean currents, weather patterns, airflow around airplane wings, and blood flow through arteries. Here's a breakdown of the equations and their significance.

1.1 Understanding Fluid Motion

Two key principles govern fluid motion:

- **Conservation of Momentum (Newton's Second Law)**: The rate of change of momentum of a fluid element is equal to the sum of forces acting on it.
- **Conservation of Mass (Continuity Equation)**: The mass of fluid remains constant over time.

The Navier-Stokes equations combine these principles to model fluid flow.

1.2 Incompressible Navier-Stokes Equations

For many practical problems, we can assume the fluid is incompressible, meaning its density remains constant (like water). Under this assumption, the Navier-Stokes equations consist of:

- Momentum equations in the x and y directions.
- The incompressibility constraint (continuity equation).

For a 2D fluid flow, these equations are written as:

1.3 Momentum in x-direction (u)

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

where:

- u: velocity in the x-direction
- v: velocity in the y-direction
- ν: kinematic viscosity
- p: pressure
- ρ: fluid density

This equation states that the change in momentum in the x-direction is due to the forces from pressure gradients and viscous diffusion.

1.4 Momentum in y-direction (v)

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

This is similar to the x-momentum equation but describes momentum in the y-direction.

1.5 Continuity Equation (Incompressibility Condition)

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

This ensures that fluid mass is conserved (no net gain or loss of fluid).

2 Key Components of Navier-Stokes

The terms in the Navier-Stokes equations represent different physical processes:

- **Inertial terms**: These terms involve $u\frac{\partial u}{\partial x}$ and $v\frac{\partial u}{\partial y}$, describing how the velocity changes due to the fluid's motion itself.
- **Pressure gradient terms**: These are $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$, representing the forces exerted by pressure differences.
- **Viscous terms**: The second-order derivatives $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial y^2}$ account for the friction between fluid layers.
- **Time derivative terms**: $\frac{\partial u}{\partial t}$ describes how the velocity changes over time.

3 Boundary Conditions

To solve the Navier-Stokes equations, we need to specify boundary conditions. These can include:

- **No-slip condition**: The fluid velocity is zero at solid boundaries (e.g., the walls of a pipe).
- **Inflow/outflow conditions**: Velocity or pressure specified at the entry or exit points of the flow domain.

4 Applications of Navier-Stokes Equations

The Navier-Stokes equations are used to model a wide range of fluid phenomena, including:

- **Vortex shedding**: Occurs when fluid flows past a bluff body, like a cylinder, and forms a repeating pattern of vortices. This is an example of a complex flow that the Navier-Stokes equations can capture.
- **Airflow around an aircraft wing**: The lift and drag forces experienced by the wing are governed by the Navier-Stokes equations.
- **Weather prediction models**: These equations are part of the equations used in climate and weather simulations.

5 Challenges in Solving the Navier-Stokes Equations

The Navier-Stokes equations are non-linear and coupled, making them challenging to solve. Analytical solutions are rare, and most real-world problems require numerical methods such as:

- **Finite Element Method (FEM)**
- **Finite Difference Method (FDM)**
- **Computational Fluid Dynamics (CFD)**

In the case of the code provided, a **Physics-Informed Neural Network (PINN)** is used to approximate the solution of the Navier-Stokes equations for the wake of a cylinder. The residuals f and g represent how well the neural network satisfies these equations at various points in the flow domain.

6 Advantages of PINNs in Solving the Cylinder Wake Problem

Physics-Informed Neural Networks (PINNs) offer significant advantages over traditional numerical methods like Finite Element Methods (FEM) and Computational Fluid Dynamics (CFD) for solving fluid flow problems such as the wake behind a cylinder. Below are some key advantages of PINNs in this context:

6.1 No Need for Mesh Generation

Traditional numerical methods require discretizing the problem domain into a mesh, which can be particularly challenging for complex geometries, such as the wake behind a cylinder. PINNs are mesh-free and solve the equations in continuous space, eliminating the need for this costly step.

6.2 Integration of Physical Laws

PINNs incorporate the governing physics directly into their loss function. In the case of fluid flow, the Navier-Stokes equations are used to inform the neural network's training, ensuring the solution adheres to physical constraints across the entire domain. Traditional methods approximate these laws through discretization, which can lead to numerical errors, especially near boundaries.

6.3 Handling of Complex Geometries

Due to their mesh-free nature, PINNs can handle complex geometries and boundary conditions more flexibly than traditional methods. This is particularly advantageous when modeling the wake behind a cylinder, where flow structures can vary greatly in size and shape.

6.4 Fewer Data Requirements

PINNs combine physical constraints with data, allowing them to work effectively even with sparse data. For the cylinder wake problem, this means that the network can predict flow patterns even if experimental data or simulation results are limited, unlike purely data-driven machine learning models that require large datasets.

6.5 Parallelism and Scalability

Neural networks can be trained in parallel on GPUs or TPUs, making PINNs highly scalable. This is particularly beneficial for large-scale simulations of fluid dynamics, where traditional methods may struggle with computational costs.

7 Workflow

This code implements a physics-informed neural network (PINN) to solve the Navier-Stokes equations, which describe fluid flow (e.g., velocity field and pressure field). The problem solved here involves fluid flow around a cylinder (cylinder wake problem). Let's break down the code section by section:

7.1 Imports

```python
import torch
import torch.nn as nn
import numpy as np
import scipy.io
from matplotlib import pyplot as plt
import matplotlib.animation as animation
```

Description:

- `torch`: Used to build the neural network and compute gradients automatically.
- `numpy`: For numerical operations.
- `scipy.io`: For loading .mat files (which store MATLAB data).

- `matplotlib`: For visualization and animation of the results.

7.2 Constants

```
nu = 0.01
```

Description:

- `nu`: This is the kinematic viscosity of the fluid, a constant used in the Navier-Stokes equations.

7.3 NavierStokes Class

This class encapsulates the PINN model for solving the Navier-Stokes equations.

7.4 Constructor (__init__)

```
def __init__(self, X, Y, T, u, v):
    ...
```

Description:

- Takes spatial (`X`, `Y`) and temporal (`T`) coordinates as inputs, along with the corresponding velocity components u and v.
- Initializes the tensors, sets up the neural network, and configures the optimizer.

7.5 Network Architecture (network)

```
def network(self):
    self.net = nn.Sequential(
        nn.Linear(3, 20), nn.Tanh(),  # input: x, y, t
        nn.Linear(20, 20), nn.Tanh(),
        nn.Linear(20, 2))  # output: psi (stream function), p (pressure
            )
```

Description: This is a fully connected neural network with 3 inputs (space x, y and time t) and 2 outputs (stream function ψ and pressure p). The stream function is used to derive velocity components u and v.

7.6 Physics-based Loss Calculation (function)

```
def function(self, x, y, t):
    res = self.net(torch.hstack((x, y, t)))
    psi, p = res[:, 0:1], res[:, 1:2]
    ...
```

Description: The stream function ψ is used to compute the velocity components u and v via automatic differentiation. It also calculates the partial derivatives needed for the loss terms that enforce the Navier-Stokes equations. These are calculated using `torch.autograd.grad`. The outputs are the velocity components u, v, pressure p, and two residuals f and g, which measure how well the current neural network approximates the Navier-Stokes equations.

7.7 Training Step (closure)

```
def closure(self):
    ...
```

Description: This function calculates the loss as the sum of four components:

- Velocity loss: Matches the predicted velocity u, v to the given data.
- Residual losses f and g: Ensure that the predicted velocities satisfy the Navier-Stokes equations.

The optimizer minimizes this loss.

7.8 Significance of Residuals

The residuals f and g in the Navier-Stokes equations are critical components that assess how well the physics-informed neural network (PINN) satisfies the governing physical laws. These residuals represent the errors in the conservation of momentum in the x- and y-directions, which correspond to the two primary equations of the Navier-Stokes system. Ideally, if the neural network perfectly satisfies the equations, the residuals should converge to zero.

7.8.1 Residual f

The residual f corresponds to the error in the x-momentum equation of the Navier-Stokes system. It is formulated as follows:

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

Where:

- $\frac{\partial u}{\partial t}$: Time derivative of the x-component of velocity.
- $u\frac{\partial u}{\partial x}$: Convective term representing fluid movement in the x-direction.
- $v\frac{\partial u}{\partial y}$: Cross-term due to the interaction with the y-component of velocity.
- $\frac{\partial p}{\partial x}$: Pressure gradient in the x-direction.
- ν: Kinematic viscosity, which influences the diffusive term involving the second-order spatial derivatives of velocity.

7.8.2 Residual g

The residual g represents the error in the y-momentum equation of the Navier-Stokes system and is given by:

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

Where:

- $\frac{\partial v}{\partial t}$: Time derivative of the y-component of velocity.
- $u\frac{\partial v}{\partial x}$: Convective term for fluid motion in the x-direction affecting the y-component.
- $v\frac{\partial v}{\partial y}$: Convective term for fluid movement in the y-direction.
- $\frac{\partial p}{\partial y}$: Pressure gradient in the y-direction.

7.8.3 Purpose of Residuals in the Code

The residuals f and g serve multiple purposes in the PINN framework:

- **Enforcing Physical Laws:** By minimizing the residuals f and g, the network is compelled to obey the Navier-Stokes equations at each point within the domain. These residuals are integral to the total loss function used for training the neural network.
- **Physics-Informed Loss:** The network learns from both observational data (measured velocities u and v) and physical laws (via residuals f and g). When these residuals approach zero, it indicates that the learned solution closely adheres to the Navier-Stokes equations.

7.8.4 Residual-Based Loss in the Code

In the `closure()` function of the code, the residuals f and g are utilized to compute the physics-informed loss:

```
f_loss = self.mse(f_prediction, self.null)  # Enforces f      0
g_loss = self.mse(g_prediction, self.null)  # Enforces g      0
self.loss = u_loss + v_loss + f_loss + g_loss  # Total loss combines
    data and physics
```

Here, $f_{\text{prediction}}$ and $g_{\text{prediction}}$ denote the network's current estimates of the residuals. By minimizing these residuals alongside the velocity data, the network learns a solution that not only fits the training data but also complies with the governing physical laws represented by the Navier-Stokes equations.

7.9 Training and Data Handling

```
N_train = 5000
data = scipy.io.loadmat('cylinder_wake.mat')
...
idx = np.random.choice(N * T, N_train, replace=False)
```

Description: Loads precomputed simulation data of the cylinder wake problem (`cylinder_wake.mat`). Selects a random subset of the data for training (5000 samples) from x, y, t, u, v, p.

7.10 Training/Testing the PINN

```
pinn = NavierStokes(x_train, y_train, t_train, u_train, v_train)
pinn.train()
```

Description: Training: If the code block is uncommented, it would train the neural network on the sampled data.

```
pinn.net.load_state_dict(torch.load('model.pt'))
pinn.net.eval()
```

Description: Testing: The model is loaded from a previously trained state (saved in `model.pt`) and switched to evaluation mode. Predictions are made on test data.

7.11 Plotting

```
fig, ax = plt.subplots()
...
```

Description: After evaluating the model, the predicted pressure field p is plotted using matplotlib's contour plot.

7.12 Animation

```
def animate(i):
    u_out, v_out, p_out, f_out, g_out = pinn.function(x_test, y_test, i
        *t_test)
    ...
```

Description: This function updates the plot dynamically to show how the pressure field changes over time. `FuncAnimation` from matplotlib is used to create an animated plot of the pressure field $p(x, y, t)$.