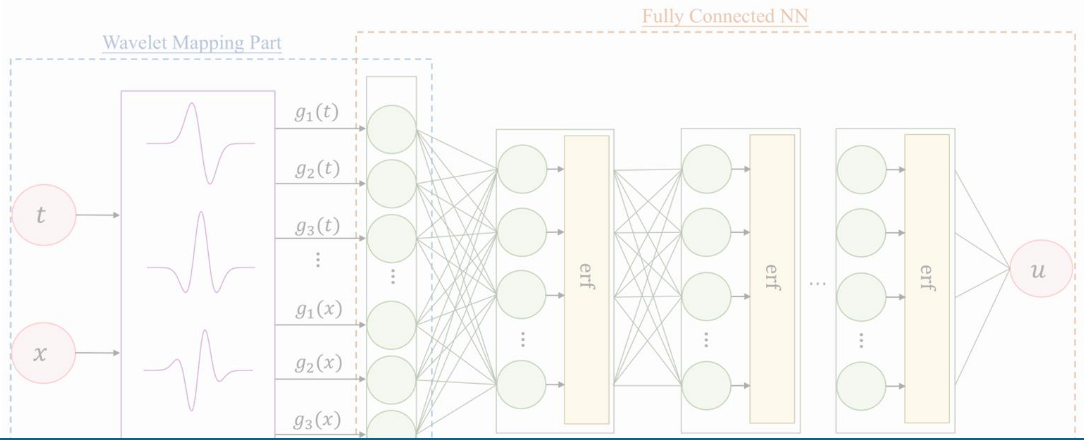
Modelling Fluid Dynamics Using Wavelet-PDE Hybrid Technique

$$u_x = \frac{\partial u}{\partial x}$$

$$u_{xx} = \frac{\partial^2 u}{\partial x^2}$$

$$u_{xy} = \frac{\partial^2 u}{\partial y \partial x} = \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right)$$



By Team 4,
Abdul Hakkim S [CCE21002]
Anirudh B Varma [CCE21007]
Arnav Purohit [CCE21008]

Modelling fluid dynamics using a Wavelet-PDE hybrid technique involves solving partial differential equations (PDEs) while employing wavelet transforms.

Navier–Stokes equations (fundamental PDE in fluid dynamics)
The Navier-Stokes equations describe the motion of viscous fluid substances and are represented as

$$\overbrace{\underbrace{\frac{\partial \mathbf{u}}{\partial t}}_{\text{Variation}} + \underbrace{(\mathbf{u} \cdot \nabla)\mathbf{u}}_{\substack{\text{Convective} \\ \text{acceleration}}}}^{\text{Inertia (per volume)}} = \overbrace{\underbrace{-\nabla w}_{\substack{\text{Internal} \\ \text{source}}} + \underbrace{\nu \nabla^2 \mathbf{u}}_{\text{Diffusion}}}^{\text{Divergence of stress}} + \underbrace{\mathbf{g}}_{\substack{\text{External} \\ \text{source}}} .$$

Why are we using wavelets for this

ANS :
- Wavelets are introduced here as a powerful tool for spatial decomposition and denoising.
- Wavelets excel in capturing local, multi-scale phenomena (e.g., turbulence).

By the way, why PDEs ❓

ANS : Partial Differential Equations (PDEs) are fundamental to modelling physical phenomena like fluid dynamics.

Tools Used:-

Python

- Numpy library
- Scipy library
- Pywt library
- Matplotlib / Seaborn library
- Fipy / Fenics library

Problem Statement and Objectives

Simulating fluid behavior with accuracy and efficiency for complex scenarios like vortices.

Objective of This Project:

- Use a wavelet-PDE hybrid approach to simulate fluid dynamics.
- Apply wavelet transforms for spatial filtering and denoising in Navier-Stokes simulations.

Methodology Overview

Define Computational Grid:

> A 2D grid representing the fluid domain.

Initialize Velocity and Pressure Fields:

> Set initial conditions (e.g., a vortex).

Wavelet Transform:

> Decompose velocity fields into spatial-frequency components.

> Apply denoising or compression.

Solve Navier-Stokes Equations:

> Update velocity and pressure using finite-difference schemes.

> Reapply wavelets at each time step for efficiency.

Visualize Results:

> Plot velocity fields and flow patterns.

Compared with previous works

Wavelet-Galerkin Methods:

Our technique: Uses wavelet transforms for spatial filtering and noise reduction but retains finite-difference schemes to solve PDEs..

This technique: Projects the PDE directly onto a wavelet basis, using wavelets as basis functions in the numerical solution

Adaptive Wavelet Collocation Methods:

Our technique: Applies wavelet transforms globally for spatial decomposition and denoising.

This technique: Dynamically refines the grid using wavelets, focusing computational resources on high-gradient regions.

Wavelet Transform for Data Compression:

Our technique: Uses wavelets to denoise velocity fields and improve computational stability.

This technique: Focuses on reducing the size of simulation datasets while retaining critical flow features.

CODE:

 # Defining the Grid

```
nx, ny = 64, 64  # Grid size
u = np.zeros((nx, ny))  # X-direction velocity
v = np.zeros((nx, ny))  # Y-direction velocity
```

 # Add a smooth initial vortex as a test case

```
u = np.sin(np.pi * X) * np.cos(np.pi * Y)

v = -np.cos(np.pi * X) * np.sin(np.pi * Y)
```

CODE:

 # Applying Wavelet Transform

```
def apply_wavelet_transform(field, wavelet='db1', level=2):
    return pywt.wavedec2(field, wavelet, level=level)
def reconstruct_from_wavelet(coeffs, wavelet='db1’):
    return pywt.waverec2(coeffs, wavelet)
```

 # Update velocity fields using the Navier-Stokes equations

```
u = u_filtered + dt * (-u * dudx - v * dudy + (1 / Re) * laplacian_u)
v = v_filtered + dt * (-u * dvdx - v * dvdy + (1 / Re) * laplacian_v)
```

 # Enforce boundary conditions (no-slip walls)

```
u[:, 0] = u[:, -1] = u[0, :] = u[-1, :] = 0
v[:, 0] = v[:, -1] = v[0, :] = v[-1, :] = 0
```

Applications and Future Scope

- **Applications:**
  - > Turbulence modeling.
  - > Weather forecasting.
  - > Aerodynamic simulations.
- **Future Improvements:**
  - > Extend to 3D simulations.
  - > Include additional phenomena like heat transfer or compressibility.
  - > Optimize wavelet thresholding for different Reynolds numbers.

Challenges

•**Challenges:**

> Balancing accuracy with computational efficiency.

> Fine-tuning wavelet parameters to preserve critical flow details.

•**Insights:**

> Wavelets significantly reduce noise and improve computational performance.

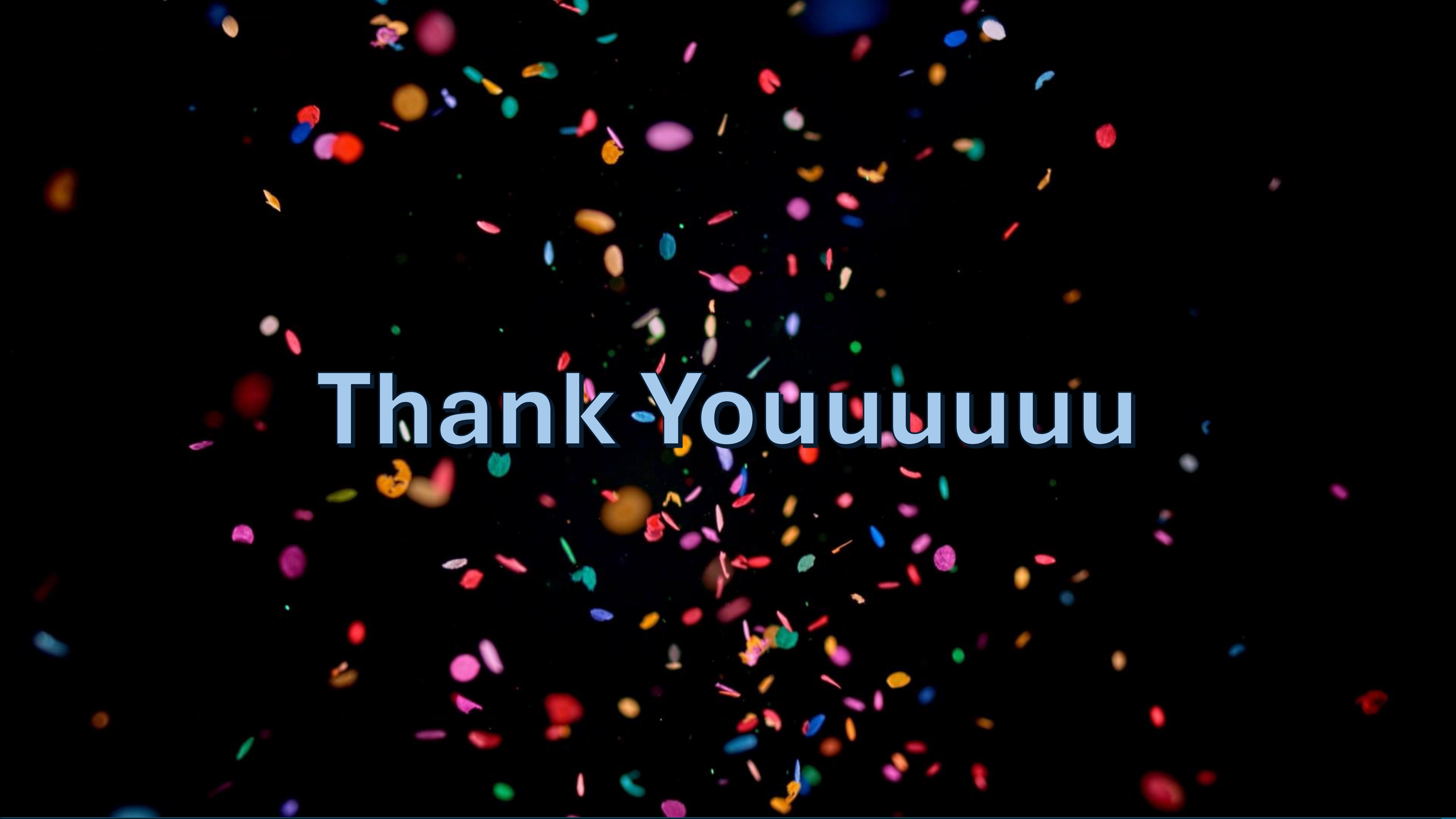
> Hybrid techniques provide a scalable approach for fluid simulations.

Conclusion

> The hybrid wavelet-PDE approach enhances fluid simulation accuracy and efficiency.

> Navier-Stokes equations remain the cornerstone of fluid dynamics modeling.

> Wavelets enable multi-scale analysis, critical for turbulent flows.


Thank Youuuuuu