

# APC 524 Final Project:

Implementing a Navier-Stokes Solver and Physics Informed Neural  
Network for Simulating Two-Dimensional Fluid Flow Around a Cylinder

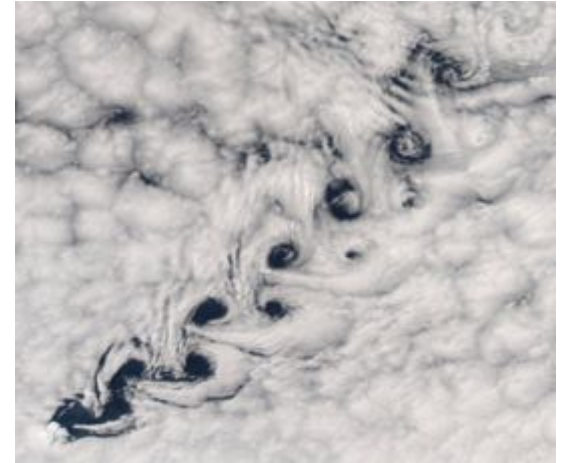
Fairuz Ishraque<sup>1</sup>, Joseph Lockwood<sup>1</sup>, and Aaron Spaulding<sup>2</sup>

<sup>1</sup>Department of Geosciences

<sup>2</sup>Department of Civil and Environmental Engineering

# Derivation Implementation of NS Solver for 2D Cylinder Wake Flow

- Implemented finite-element Navier-Stokes (NS) solver.
- Focus on incompressible, viscous fluid flow.
- No-slip boundary conditions for real fluid behavior.
- Investigated flow separation, wake formation, and vortex shedding.



# Unit Testing: Ensuring Code Reliability

- Comprehensive Unit Tests
  - Robust suite of tests covering various aspects such as Class and method functionality
- Initialization Tests
  - Validates proper class initialization with default values
  - Ensures correct setup of the simulation environment
- Mechanics Validation
  - Validates internal mechanics of the Environment class
  - Essential for computational algorithm accuracy
- Verification and Validation
  - Verification: Code meets requirements and functions correctly
  - Validation: Requirements make sense and serve intended purpose
  - Enhances code reliability and effectiveness
- Adversarial Testing
  - Rigorous challenges to identify potential defects
  - Documentation of code expectations and requirements
  - Builds user confidence in code reliability

# Speed and Efficiency Analysis

- **Profiling with cProfile**
  - Identifying performance bottlenecks
  - Gathering essential performance data
- **SnakeViz for Analysis**
  - Using SnakeViz, a graphical profiler viewer
  - Visualizing and decoding performance bottlenecks
  - Targeting areas for optimization
- **Benefits**
  - Pinpoint resource-intensive functions/methods
  - Informed optimization decisions

# Implementation of a General Navier-Stokes Solver with the Finite Difference Method

- Implemented a customizable environment class that allows arbitrary environments and environmental conditions
- Defined modular boundary conditions allowing for many environment types to be investigated
- Created composable objects that allow for complex environments to be modeled and simulated

# Implementation of a General Navier-Stokes Solver with the Finite Difference Method

## Environmental Variables:

- Spatial Resolution
- Temporal Resolution
- Density
- Kinematic Viscosity
- Environment Size

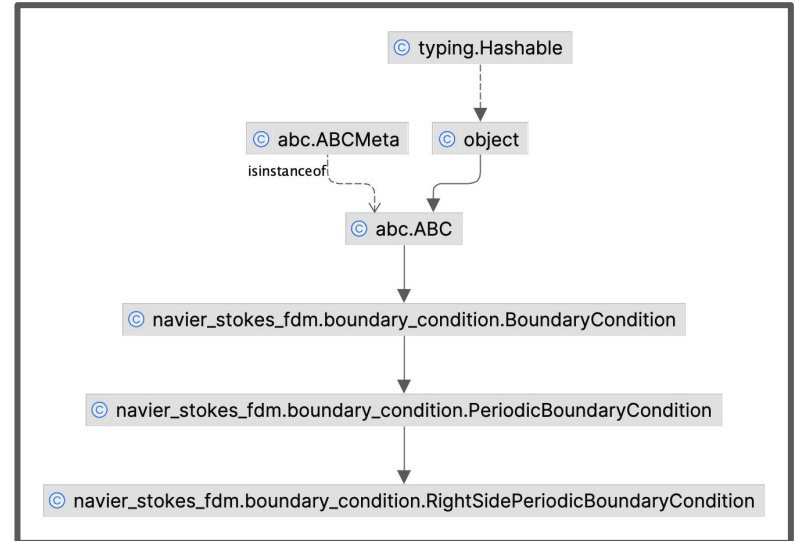
## Boundary Conditions:

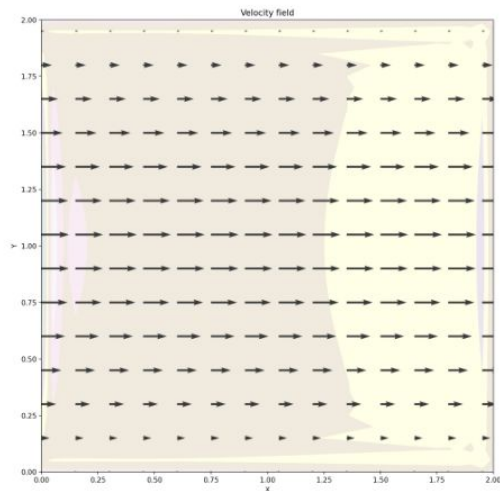
- No-Slip Boundary Condition:
- Fixed Velocity Boundary Condition
- Periodic Boundary Condition
- Free Slip Boundary Condition

## Implemented Objects:

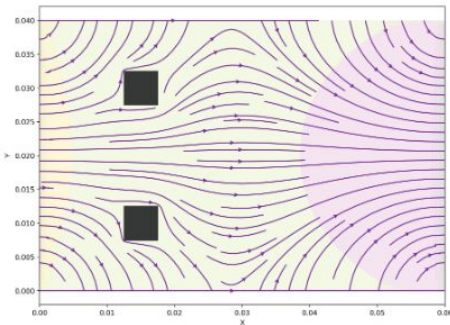
- Rectangle
- Circle

Inheritance Tree For the Right Side  
Periodic Boundary Condition

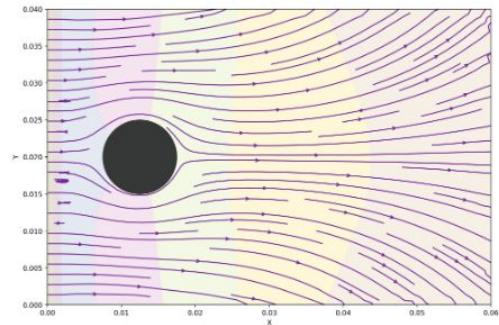




(a) Simulation of fluid flowing in a pipe. The top and bottom boundary conditions are set as no-slip conditions, while the left and right are set as periodic boundary conditions.



(b) Simulation of a fluid flowing around two boxes. Boundary conditions for all sides are set as fixed velocity conditions. The boundary conditions for the boxes are set as no-slip conditions are updated dynamically as each box is added to the environment.



(c) A simulation of fluid flow around a cylinder. Here the right, top, and bottom sides have no slip conditions while the left side has a fixed velocity boundary condition. The boundary conditions around the cylinder are automatically updated at simulation time.

Figure 2: The modular boundary conditions, customizable environment, and composable objects allows for easy simulation of complex environments with very different conditions and requirements.

**Aaron Spaulding's Contributions**

# Example Simulation Setup

Python Module Imports

Modular Boundary Conditions

Composable Objects

Customizable Environment

Automatic Plotting

```
from navier_stokes_fdm import Environment
from navier_stokes_fdm import Rectangle
import navier_stokes_fdm.boundary_condition as bc
```

```
U = 1 # m/s
dimension = 0.005
```

```
boundary_conditions = [
    bc.TopSideFixedVelocityBoundaryCondition(u_value=U, v_value=0),
    bc.BottomSideFixedVelocityBoundaryCondition(u_value=U, v_value=0),
    bc.LeftSideFixedVelocityBoundaryCondition(u_value=U, v_value=0),
    bc.RightSideFixedVelocityBoundaryCondition(u_value=U, v_value=0),
]
```

```
x1, y1 = 0.0125, (0.04 / 2) - (dimension / 2)
objects = [Rectangle(x1, y1, x1 + dimension, y1 + dimension)]
```

```
a = Environment(
    F=(1.0, 0.0),
    len_x=0.06,
    len_y=0.04,
    dt=0.00000015,
    dx=0.0001,
    boundary_conditions=boundary_conditions,
    objects=objects,
    rho=0.6125 # kg/m3
    nu=3e-5 # m2/s
)
```

```
a.run_many_steps(480)
a.plot_streamline_plot(title="", filepath="../Figures/box_example_streamline.png")
```



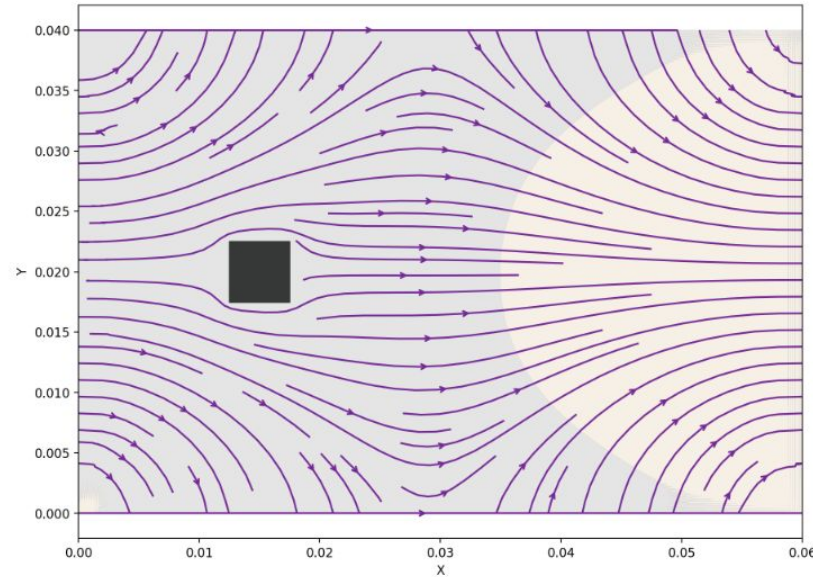
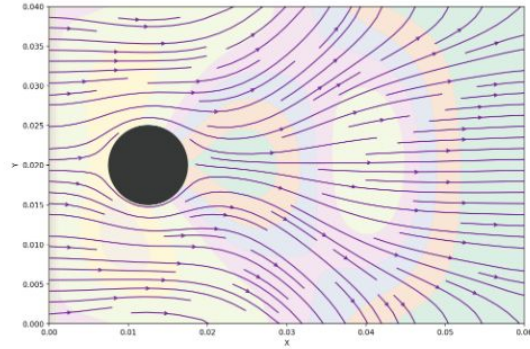
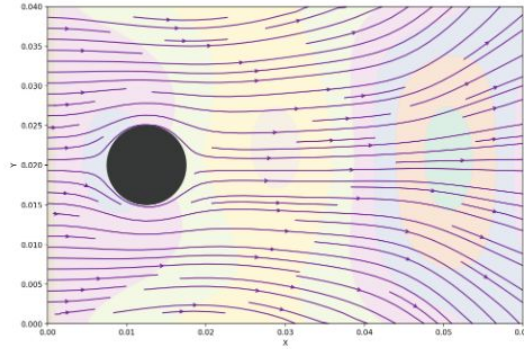


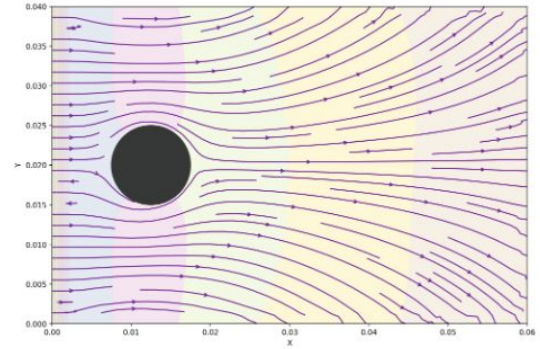
Figure 3: Example streamline plot of a fluid flow around a box. Each boundary is assigned a fixed velocity. Shading represents the pressure field with lighter colors indicating regions of lower pressure. Streamlines are shown in purple.



(a)  $9\mu s$



(b)  $13.5\mu s$



(c)  $45\mu s$

Figure 4: Three time steps of the FD simulation of fluid flow around a cylinder. The pressure field is shown by the shading while streamlines are shown in purple.

# Implementation of a General Navier-Stokes Solver with the Finite Difference Method

- I implemented tests using “pytest” for boundary conditions.
- Implemented automatic testing with GitHub Actions to test every commit for changes that might break core functionality



A screenshot of a GitHub Actions workflow log. The log is displayed on a dark background with light text. It shows a sequence of steps, each with a green checkmark icon, a step name, and a duration. The steps are: 'Set up job' (1s), 'Run actions/checkout@v4.1.1' (3s), 'Set up Python' (0s), 'Install dependencies' (40s), 'Run pytest' (2s), 'Post Set up Python' (0s), 'Post Run actions/checkout@v4.1.1' (0s), and 'Complete job' (0s). The 'Install dependencies' step is the longest, taking 40 seconds.

>	✓ Set up job	1s
>	✓ Run actions/checkout@v4.1.1	3s
>	✓ Set up Python	0s
>	✓ Install dependencies	40s
>	✓ Run pytest	2s
>	✓ Post Set up Python	0s
>	✓ Post Run actions/checkout@v4.1.1	0s
>	✓ Complete job	0s

# References

- NASA / GSFC / Jeff Schmaltz / MODIS Land Rapid Response Team. English: Cloud vortices in the cloud layer off Heard Island, south Indian Ocean. The Moderate Resolution Imaging Spectroradiometer (MODIS) aboard NASA's Aqua satellite captured this true-color image of sea ice off Heard Island on Nov 2, 2015 at 5:02 AM EST (09:20 UTC). Nov. 2015. url: [https://commons.wikimedia.org/wiki/File:Heard\\_Island\\_Karman\\_vortex\\_street.jpg](https://commons.wikimedia.org/wiki/File:Heard_Island_Karman_vortex_street.jpg)
- M. M. Zdravkovich. Flow around Circular Cylinders: A Comprehensive Guide through Flow Phenomena, Experiments, Applications, Mathematical Models, and Computer Simulations. Vol. 1. Oxford University Press, 1997. 8