FlowBench : A Large Scale Benchmark for Flow Simulation over Complex Geometries

Ronak Tali[1]*, Ali Rabeh[1]*, Cheng-Hau Yang[1]*, Mehdi Shadkhah[1], Samundra Karki[1]
Abhisek Upadhyaya[2], Suriya Dhakshinamoorthy[1], Marjan Saadati[1], Soumik Sarkar[1],
Adarsh Krishnamurthy[1], Chinmay Hegde[2], Aditya Balu[1], Baskar Ganapathysubramanian[1]

[1]**Iowa State University**
{rtali,arabeh,chenghau,mehdish,
samundra,snarayan,marjansd,soumiks,
adarsh,baditya,baskarg}@iastate.edu

[2]**New York University**
{au2216,chinmay.h}@nyu.edu

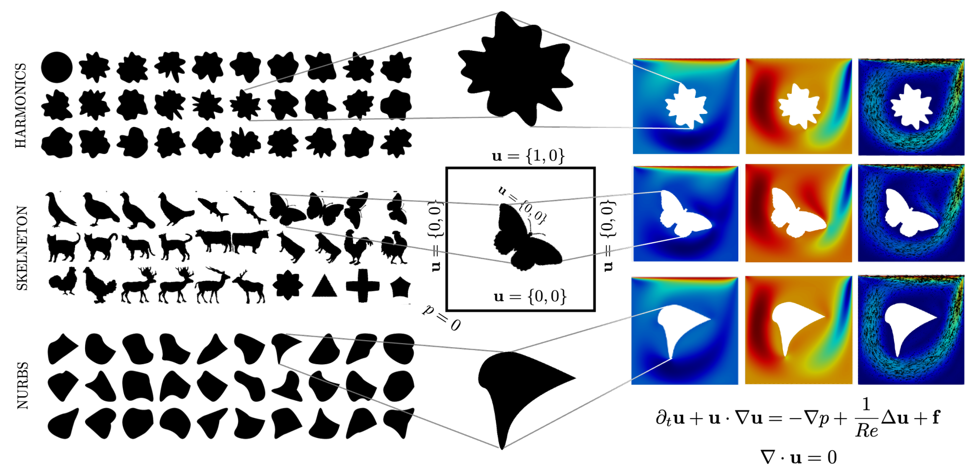


Figure 1: *FlowBench offers comprehensive datasets and metrics for assessing neural PDE solvers designed to model flow phenomena around complex objects. It includes three sets of application-relevant geometries with varying complexities and high-fidelity flow simulation data under different forcing conditions. The left panel in the figure above showcases 30 randomly selected shapes from each geometry group. The middle panel provides a close-up of one geometry within the computational domain, highlighting the boundary conditions. The right panel displays the simulation outputs, including velocity results for three samples.*

Abstract

Simulating fluid flow around arbitrary shapes is key to solving various engineering problems. However, simulating flow physics across complex geometries remains numerically challenging and computationally resource-intensive, particularly when using conventional PDE solvers. Machine learning methods offer attractive opportunities to create fast and adaptable PDE solvers. However, benchmark datasets to measure the performance of such methods are scarce, especially for flow physics across complex geometries. We introduce FlowBench , a benchmark for neural simulators with over 10K samples, which is larger than any publicly available flow physics dataset. FlowBench contains flow simulation data across complex geometries (*parametric vs. non-parametric*), spanning a range of flow conditions (*Reynolds number and Grashoff number*), capturing a diverse array of flow phe-

*These authors contributed equally to this work.

Preprint. Under review.

nomena (*steady vs. transient; forced vs. free convection*), and for both 2D and 3D. FlowBench contains over 10,000 data samples, with each sample the outcome of a fully resolved, direct numerical simulation using a well-validated simulator framework designed for modeling transport phenomena in complex geometries. For each sample, we include velocity, pressure, and temperature field data at 3 different resolutions and several summary statistics features of engineering relevance (such as coefficients of lift and drag, and Nusselt numbers). We envision that FlowBench will enable evaluating the interplay between complex geometry, coupled flow phenomena, and data sufficiency on the performance of current, and future, neural PDE solvers. We enumerate several evaluation metrics to help rank order the performance of current (and future) neural PDE solvers. We benchmark the performance of three baseline methods: Fourier Neural Operators (FNO), Convolutional Neural Operators (CNO), and DeepONets. This dataset(here) will be a valuable resource for evaluating neural PDE solvers that model complex fluid dynamics around 2D and 3D objects.

1 Introduction

Accurate modeling of fluid flow around complicated objects is central to a plethora of applications. In aerospace and automotive applications, flow around wings, and car bodies can significantly impact design and performance [1–3]. In civil and environmental engineering applications, understanding fluid flow patterns around structures like buildings [4] and bridges [5] is critical from a safety perspective, while flow patterns inside buildings are important from a safety and comfort perspective [6]. Bio-flow applications, such as flow around fish [7, 8], birds [9], insects [10], and heart valves [11], are all important fields of study. Sports engineering scientists use fluid flow simulations to optimize equipment design, such as bike helmets and golf ball shapes [12].

Fluid flow is also often intricately connected with thermal effects. A thermal mismatch between the fluid medium and the object geometry can produce buoyancy-driven phenomena, cause thermal plumes, and impact mixing and thermal transport with significant engineering implications. For example, geometry is a critical consideration during the design of heat exchangers [13, 14] in the semiconductor industry. Mitigating urban heat island effects [15, 16] requires modeling and controlling thermal and flow transport across complex urban landscapes. Additionally, ensuring indoor comfort and safety from the transmission of infectious diseases requires a comprehensive understanding of the indoor flow patterns in the built environment [6, 17].

A central challenge in accurately simulating flow patterns in complex geometries is the high resource cost of traditional simulation approaches. High-fidelity flow (and thermal) simulations often require hours to days (or sometimes even months [18]) on high-performance computing (HPC) systems. Scientific machine learning (SciML) has emerged as a promising path towards resolving this challenge. By combining training data with domain-specific information (e.g., physical constraints and smoothness assumptions), SciML approaches offer fast simulation, better extrapolation capabilities, and lower data requirements. This includes impactful applications like weather prediction [19] and canonical flows in simple geometries [20–24].

Despite these successes however, there is a notable lack of datasets for flow (and flow-thermal) interactions with complicated geometries. While databases exist for flow passing through simple shapes such as cylinders [21, 22] and airfoils [20, 24], there remains a significant gap in datasets involving more diverse and complex shapes. We note the availability of a few datasets that include other geometries, but these geometries are limited to drone shapes [23].

In addition to the need for more complex geometries in existing flow datasets, there is also a shortage of multiphysics datasets. By multiphysics, we mean applications that are modeled as a set of (tightly) coupled partial differential equations (PDE), with each PDE modeling a specific physical phenomenon – for instance, Navier-Stokes that models flow phenomena and the advective heat equation that models thermal phenomena. This gap exists because coupling different types of PDEs is inherently challenging and computationally expensive. Solving multiphysics problems requires sophisticated numerical methods and substantial computational resources to accurately simulate each subproblem and capture the interactions between various physical phenomena.

FlowBench seeks to enable the ML community to build the next generation of SciML neural PDE solvers by filling these gaps – complex geometries and multiphysics phenomena. In particular, FlowBench offers:

- **Flow across complex geometries**: We simulate flow and thermal-flow phenomena across a wide range of complex shapes – both parametric and non-parametric – in 2D and 3D. These include simple shapes like ellipses, more complex blobs, and geometries like insects, animals, and birds. Flow across this spectrum of complex objects exhibits a rich array of vortex formation, flow separation, and a range of lift and drag profiles. The diverse shapes and the flow interactions with them provide a rich and intricate dataset for training geometry-aware SciML solvers that can be used for various applications.
- **Multiphysics simulations**: For each shape, we perform a variety of simulations representing flow (i.e., incompressible Navier-Stokes, NS) as well as thermal flow (i.e., coupled Navier-Stokes and Heat Transfer using the Bousinessque coupling, NS-HT) scenarios. Here, we span steady-state and transient behaviors, offering a comprehensive dataset and benchmarks to test and evaluate solvers under various scenarios.
  - For the steady-state case, we consider a variant of the canonical fluid dynamics problem of lid-driven cavity flow (LDC), which is an example of internal flow. The cavity has an object (complex geometry) placed inside (see Figure 3). Additionally, we consider a temperature difference between the cavity walls and the object, producing thermal and flow coupling. For each object, we simulate across a range of Reynolds number $Re \in [10^1, 10^3]$, and Grashof number $Gr \in [10^1, 10^7]$. This range offers a variety of forced $\left(Gr/Re^2 \approx 0.1\right)$, mixed, and free convection $\left(Gr/Re^2 \approx 10\right)$ scenarios. In this case, there are nearly 9000 unique samples for 2D and 500 unique samples for 3D.
  - For the transient case, we consider a variant of another canonical fluid dynamics problem – flow past a bluff object (FPO) – which is an example of external flow. Here, we consider flow moving past a stationary (complex geometry) object that is placed in a large domain. The fluid exhibits various intricate time-dependent patterns as it moves past the object. For each object, we simulate across a range of Reynolds number $Re \in [10^2, 10^3]$, offering an array of vortex-shedding frequencies and other time-dependent patterns. In this case, there are over 1000 unique samples in 2D.

  We provide field data of velocity, pressure, and temperature for each of the $10K+$ simulations. Additionally, we provide summary engineering features, including the coefficient of lift (C_L), drag (C_D), and the average heat transfer (Nusselt number, Nu) from the surface of the complex object. We provide our dataset as a benchmark for others interested in the development and evaluation of SciML models
- **Benchmark metrics and comparisons**: Besides the dataset, we also include workflows to train three types of neural operators – Fourier Neural Operators (FNO), Convolutional Neural Operators (CNO), and Deep Operator Networks (DeepONets). We provide trained operator networks for the steady-state case and comprehensive evaluation metrics for all cases. We also suggest a hierarchy of in-distribution and out-of-distribution tests to evaluate the generalizability of these models.

The dataset (consisting of $10K+$ samples), evaluation metrics, workflows, and trained models together make FlowBench a valuable tool for the ML community to create SciML solvers of coupled phenomena involving complex geometries. All data, visualizations, models, and model evaluations are available on the FlowBench website.

2 Related Work

Most publicly available benchmark datasets are summarized and compared in Table 1. PDEBench [26] presents a dataset consisting of simulations run for a wide range of PDEs, not just flow physics. It includes simulations for both compressible and incompressible Navier-Stokes problems, in both two and three dimensions.

CFDBench [21] is a fluid flow-focused dataset containing a total of 739 cases. These cases are distributed across various setups: lid-driven cavity (with varying density and viscosity over 25 different length and width combinations), tube flow (varying density, viscosity, and geometry, with 50 different inlet velocity conditions), dam flow, and cylinder flow specialties.

Table 1: *Comparison of incompressible flow simulation data in AirfRANS, Curated Dataset, CFDBench, MegaFlow, Eagle, Graph-Mesh, PDEBench, and our FlowBench . An orange check mark indicates a restricted family of shapes (Airfoil: AirRANS and Graph-Mesh; Cylinder: CFDBench, Cylinders and Ellipses: MegaFlow; Drone geometry: Eagle).*

Name	Dimensions	# Simulations	Geometry	Multiphysics
AirfRANS [20]	2	1000	✓	✗
Curated Dataset [25]	2	116	✗	✗
CFDBench [21]	2	739	✓	✗
MegaFlow [22]	2	3000	✓	✗
Eagle [23]	2	1200	✓	✗
Graph-Mesh [24]	2	230	✓	✗
PDEBench [26]	2	1000	✗	✗
ScalarFlow [27]	3	100	✗	✓
Fluid Flow Dataset [28]	2	8000	✗	✗
BubbleML [29]	2,3	79	✗	✓
FlowBench (ours)	2,3	**10650**	✓	✓

Megaflow2D [22] is a comprehensive collection of 3000 cases for 2D Navier-Stokes problems. Each case features different geometrical configurations, including circles, ellipses, and nozzles. All simulations are performed at a fixed Reynolds number of 300.

McConkey et al. [25] focuses exclusively on 2D turbulence simulations, covering 841 different cases across 29 flow scenarios such as a periodic hill, square duct, parametric bump, converging-diverging channel, and curved backward-facing step. Each scenario includes 29 simulations with varying parameters, such as Reynolds numbers, producing 841 data samples.

The Graph-Mesh [24] dataset simulates incompressible 2D Navier-Stokes problems at high Reynolds numbers (greater than 10^6) exclusively for Aerofoils. It varies the angle of attack, inlet velocity, Reynolds number, and Mach number for a total of 230 simulations. A distinguishing feature is the calculation of drag and lift after fitting the SciML models, not just the solutions.

WeatherBench [19] condenses raw weather data from the ERA5 dataset by reducing resolution levels to fit within a GPU. Along with discrete weather-based variables, WeatherBench runs simulations for a range of u, v velocities, temperature, and vorticity. It uses the entire world as its grid and provides both 2D and 3D data post-processed from the ERA5 dataset.

ScalarFlow [27] is the multiphysics simulation dataset coupling Navier-Stokes with density distribution solving, focusing on buoyancy-driven smoke plume reconstructions. It provides volumetric 3D flow reconstructions for complex buoyancy-driven flows transitioning to turbulence.

The Fluid Flow Dataset [28] contains 8000 unsteady 2D fluid flow simulations, each with 1001 time steps, parameterized by Reynolds number.

BubbleML [29] is another multiphysics dataset to couple two-phase Navier-Stokes and energy equations. It includes 79 simulations covering nucleate pool boiling, flow boiling, and sub-cooled boiling under various gravity conditions, flow rates, sub-cooling levels, and wall superheat.

3 FlowBench

3.1 Geometries

Our dataset includes three distinct categories of geometries, namely **G1**, **G2**, and **G3** as illustrated in Figure 2. The first set of geometries, **G1**, consists of parametric shapes generated using Non-Uniform Rational B-Splines (NURBS) curves. NURBS are mathematical representations used in computer graphics and CAD systems to generate and represent curves and surfaces. They offer great flexibility and precision in modeling complex shapes. Each NURBS curve is defined by a set of control points, the degree of the basis function, and knot vectors [30]. We use a uniform knot vector with a second-order (quadratic) basis function, which remains fixed. However, the positions of eight control points are randomly varied to produce a variety of curves. We ensure that the shapes are smooth and

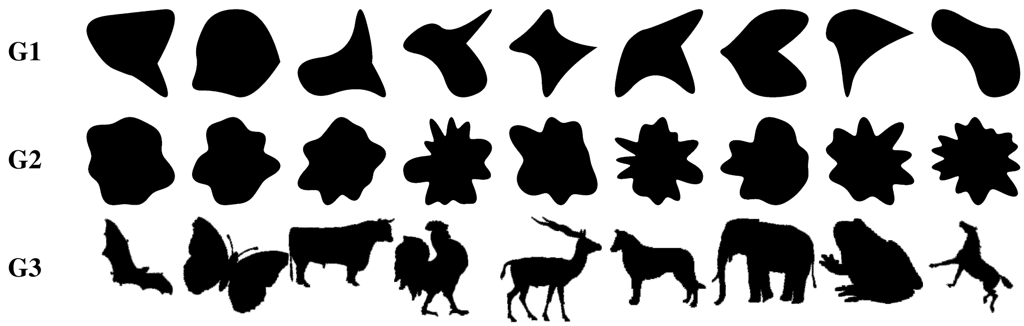


Figure 2: *Examples of the diverse and complex geometries in FlowBench using 9 samples from each of the three groups. The first row corresponds to geometries from the nurbs group G1, the second row to the spherical harmonics group G2, and the third row to the skelneton group G3.*

do not have any discontinuities or self-intersections. All shapes are normalized to fit within the unit hypercube, $[0, 1]^2$. We provide the shapes as well as code for recreating these geometries.

The next set of geometries, **G2**, consists of parametric shapes generated using spherical harmonics [31]. We randomly select $N = 8, ..., 15$ harmonics with amplitudes (a_n, b_n) ranging from 0 to 0.2. The radial function $r(t) = 0.5 + \sum_{n=1}^{N} (a_n \cos(nt) + b_n \sin(nt))$ then defines the shape; and is computed at 500 evenly spaced points in $t \in [0, 2\pi]$. We normalize $r(t)$ so that any surface point is within a distance of 0.5 from the center of the shape, $r(t) = 0.5 \left(\frac{r(t)}{r_{\max}}\right)$. We provide shapes as well as code for recreating these geometries.

The last set of geometries, **G3**, consist of non-parametric shapes sampled from the grayscale dataset in SkelNetOn [32, 33]. We apply a Gaussian blur filter with a scale of 2 to smoothen out some of the thin features of the object. This ensures the shape remains consistent across the three resolutions we provide data for. We then scale the shape to ensure it is contained in the unit hypercube $[0, 1]^2$.

3.2 Flow Physics Problem: Domain, Boundary Conditions, and Outputs

In the 2D LDC setup, a square features three stationary walls and one moving lid, with a domain size of [0, 2] × [0, 2]. An object is placed in the middle of the flow within the chamber. Examples of LDC simulations showing streamlines and y-direction velocities, along with the drag coefficient (C_D) and lift coefficient (C_L) values, are shown in Figure 3. Increasing the Reynolds number brings the vortices closer to the right wall, with additional vortices forming at the bottom-left and bottom-right corners. With increasing Reynolds numbers, the smaller viscous forces acting on the geometries decrease both C_D and C_L.

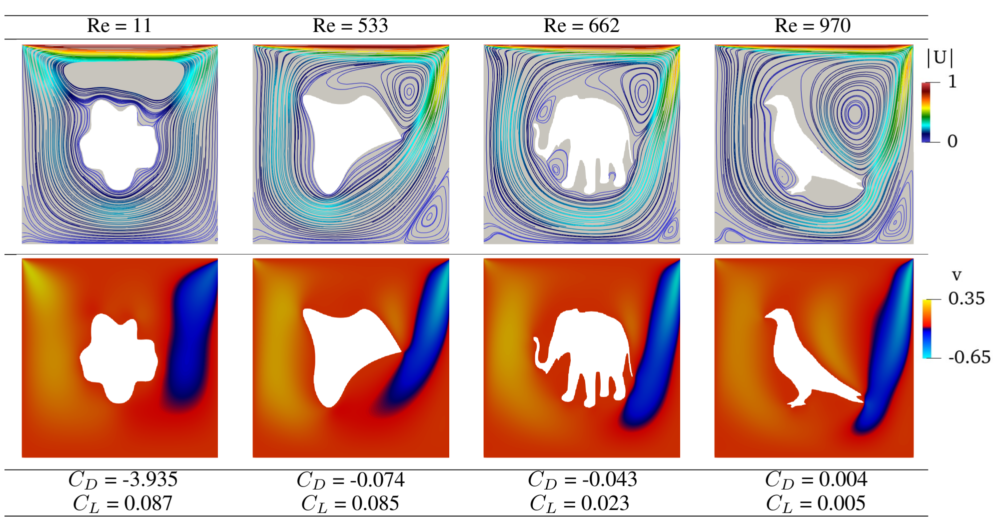


Figure 3: *Drag coefficients (C_D), and lift coefficients (C_L) for different shapes and different Reynolds numbers in pure-NS LDC simulations.*

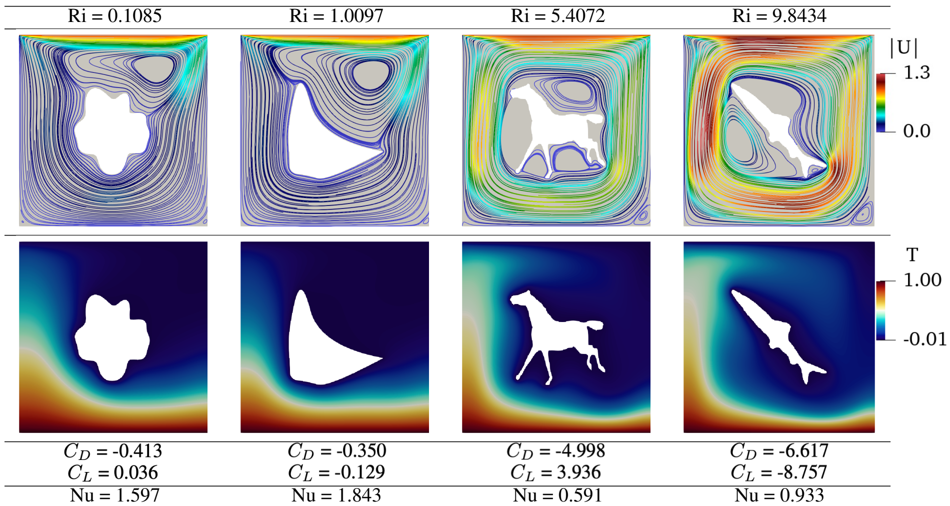
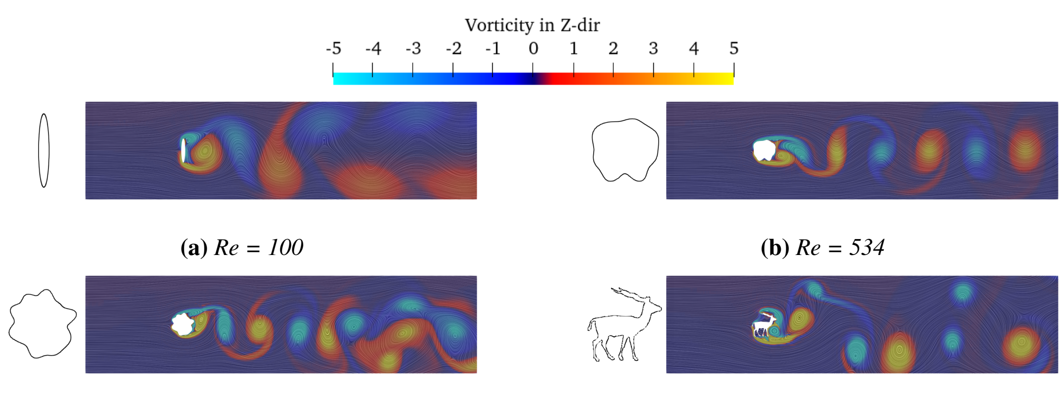


Figure 4: *Drag coefficients (C_D), lift coefficients (C_L), and Nusselt numbers (Nu) for different shapes and different Richardson numbers in NSHT LDC simulations (fixed Re = 100).*

For the 2D Navier-Stokes and heat transfer (NSHT) lid-driven cavity (LDC) problem, the bottom wall is set to a temperature of 1, while the top wall is set to 0. The left and right walls have zero-flux temperature boundary conditions. The object surface is set to a temperature of 0. This setup produces a combination of forced convection (flow due to the moving lid) and Rayleigh-Benard instabilities (flow due to buoyancy-driven natural convection). Examples of NSHT-LDC simulations, showing streamlines, temperatures, and the values of C_D, C_L, and Nu, are presented in Figure 4. We use a non-dimensional number, Richardson number, defined as $Ri = Gr/Re^2$, representing the ratio between buoyancy and inertial force. As Ri increases, we observe higher values of C_D and C_L. A higher Richardson number indicates that buoyancy effects are more significant than the forced flow. The heated fluid rises more strongly, creating greater circulation within the chamber. The increased circulation results in stronger forces acting on the object, leading to higher C_D and C_L.

For the FPO setup, we consider a domain $[0, 64] \times [0, 16]$ with the object placed at (6, 8). A parabolic velocity inlet boundary condition is applied on the left, no-slip boundary conditions are applied on the top and bottom walls, and a zero pressure boundary condition is applied on the right. The large domain size for the FPO problem is chosen to capture as much physical detail as possible in our dataset. We report a smaller cropped-out region of size $[0, 16] \times [0, 4]$ from this dataset, representing a tradeoff between dataset size and the amount of physics captured. Figure 5 shows time snapshots of representative shapes showing vortex shedding, as the flow rotates and stretches around the object. Movies of the time-dependent flow can be seen on the FlowBench website.



(a) *Re = 100* **(b)** *Re = 534* **(c)** *Re = 700* **(d)** *Re = 1000*

Figure 5: *Time snapshots of vorticity for the FPO problem, illustrating vortex shedding profiles across four samples of geometries and Reynolds numbers. Each sample showcases distinct flow characteristics and vortex dynamics.*

The 3D LDC cases are similarly defined and simulated. Due to the computational expense of performing DNS simulations on 3D geometries, we report 500 simulations of complex objects. Here, we focused on parametric shapes, specifically ellipsoids and tori, exhibiting various aspect ratios and orientations.

3.3 Simulation Framework and Compute effort

Our simulation framework is a highly parallel octree-based CFD and multiphysics code. We account for complex geometries using a robust variant of the immersed boundary method, called the Shifted Boundary Method (SBM) [34, 35]. For readers interested in our framework's SBM formulation and implementation, please refer to [36]. Our octree-based framework has been extensively validated and used in various applications like industrial-scale CFD simulations [37], two-phase flow [38], buoyancy-driven flows [39], and coupled multiphysics application [40]. For additional validation of the incompressible flow and thermal incompressible flow solving using SBM, please refer to Section A.5 in the appendix. All the 2D LDC cases are simulated using a uniform mesh of 512×512. All the 2D FPO cases are simulated using an adaptive mesh that resolved to the Kolmogorov length scale close to the object. All the 3D LDC cases are simulated using an adaptive mesh that resolved to 2X the Kolmogorov scale. Illustrations of the mesh are provided in Section A.3. We deployed this framework on one of the largest academic supercomputing clusters in the US, called TACC Frontera [41]. These simulations required about 65K nodehours of runtime.

3.4 MetaData

Input Fields: We have provisioned the following input fields

1. **Reynolds Number**, **Grashof Number**: We feed the Reynolds and Grashof numbers as a concordant (matching the dimensions of the flow domain e.g. for 2D lid driven cavity problem, 512×512) array comprising of a single integer value everwhere.
2. **Geometry Mask**: A geometry mask (g) is a binary representation of a shape or object within a spatial domain. Each element in the mask can have one of two values, typically 0 or 1, where: 0 indicates that the point is outside the object and 1 indicates that the point is inside the object. The geometry mask helps in identifying and isolating the region of interest (the object) from the background or surrounding space. This information is again packaged as a concordant array with each entry marked as 0 or 1.
3. **Signed Distance Field (SDF)**: A Signed Distance Field (s) is a scalar field that represents the shortest distance from any point in space to the surface of a given shape. The "signed" aspect of the signed distance field indicates whether the point is inside or outside the shape. A negative value indicates the point is inside the shape, while a positive value indicates the point is outside the shape. A value of zero indicates the point is exactly on the surface of the shape. The SDF provides additional information through the distance values, which is important for understanding the spatial relationship between locations and the geometric boundary. This method offers richer geometric information compared to the binary geometry mask, as it includes both positional and distance information. Yet again, this information is packaged as a concordant array with each entry representing the nearest distance to the geometrical shape.

Output Fields: We are interested in obtaining field solutions (i.e., solutions at every point in the interior of the domain) for certain cardinal fields. For a 2D solution domain, these are: u - velocity in x direction, v - velocity in y direction, p - pressure and θ - temperature. Additionally, depending on whether we are solving a steady-state or a time-dependent problem, we would have either have one final snapshot of these cardinal fields or a sequence of these fields distributed uniformly over time.

Resolution: We perform simulations at DNS resolution. FlowBench contains postprocessed simulation results at three different resolutions. This serves multiple purposes: First, it allows the community to systematically explore tradeoffs in the amount of data vs resolution vs accuracy. Second, since we perform non-interpolatory sub-sampling, this data allows the community to systematically test PDE super-resolution approaches. Finally, the lower-resolution datasets offer easier opportunities to train the data. For instance, at the time of submission, we could not train the 3D SciML models at the highest resolution available. For the steady state cases, we provide data at resolutions $512 \times 512(\times 512)$, $256 \times 256(\times 256)$, $128 \times 128(\times 128)$. For the time dependant cases, we provide data at resolution 2048×512, 1024×256, and 512×128. We provide 240 snapshots for each time-dependant case.

Dataset Format: All of our datasets are provided as numpy compressed (`.npz`) files. For each steady-state problem, we provide two `.npz` files - one for input and one for output. The input files are suffixed with the marker `"_X.npz"`, and similarly, the corresponding output files are suffixed with the marker `"_Y.npz"`. In 2D, each of these `.npz` files contains a 4D `numpy` tensor of the following form:

$$[\textbf{samples}][\textbf{number\_of\_channels}][\textbf{resolution\_x}][\textbf{resolution\_y}]$$

For time-dependent problems, we provide a single file for each problem. This decision was taken to allow maximum flexibility to the end user in deciding what and how many time steps they want to use to train their models, as these time-dependent problems often take the shape of sequence-to-sequence formulations. In 2D, the resulting `.npz` files take the following form:

$$[\textbf{samples}][\textbf{number\_of\_time\_steps}][\textbf{number\_of\_channels}][\textbf{resolution\_x}][\textbf{resolution\_y}]$$

We provide a total of over 10,000 samples spread across four families of datasets. Table 5 in the Appendix provides a detailed formulaic description of the packaging of the input and output `numpy` tensors for each of these five families.

3.5 Evaluation Metrics and Test Datasets

Evaluation metrics: We recommend a hierarchy of metrics ($M1, M2, M3, M4$) to comprehensively assess the performance of trained models using FlowBench :

- $M1$*: Global metrics*: The Mean Squared Error (MSE) over the entire domain is a good primary metric to measure the accuracy of predicted velocity, pressure, and temperature fields, reflecting how closely the model's predictions match the true values.
- $M2$*: Boundary layer metrics*: The MSE of velocity, pressure, and temperature fields over a tight bounding box encompassing the complex object (we suggest 10%). This tougher metric provides insight into the model's accuracy in predicting near-surface phenomena, which are crucial for applications including flow diagnostics, shape design, and dynamic control.
- $M3$*: Property metrics*: An application-driven metric is to evaluate the accuracy of the summary statistics – C_D, C_L, Nu. The coefficients of lift and drag are critical for assessing the forces acting on the object, while the average Nusselt number predicts heat transfer rates, all of which are key for flow-thermal management and engineering applications. This is a more forgiving metric because it represents spatially averaged properties.
- $M4$*: Residual metrics*: One can also evaluate how well the predicted field satisfies the underlying (set of) PDE. Evaluating how well continuity ($\nabla \cdot u = 0$) is satisfied all over the domain is a measure of respecting the conservation of mass. Similarly, the global average of the PDE residual evaluates how well the model satisfies the underlying PDE in the domain.

Test Datasets: We recommend evaluating trained models on two test scenarios

- *Standard*: This is the standard 80-20 random split of the FlowBench data.
- *Hard*: Here, we recommend splitting the data based on their operating characteristics, i.e. Reynolds number and Grashof (or Richardson) number. Samples with (Re, Gr) in the intermediate range (say, the middle 70%) should be used for training and tested against the out-of-distribution (Re, Gr) samples. While the specifics of the flow patterns in this test dataset may differ from the training set, the global features remain the same, thus providing a good test of generalizability.

We provide code for creating the test data and evaluating these metrics. Together, these metrics and test datasets provide a comprehensive evaluation framework, allowing practitioners to evaluate model accuracy, physical consistency, and practical reliability.

4 Experiments

We report baseline results on training a suite of the most common neural PDE solvers. We studied the following (neural operator) frameworks and report results on the 2D LDC-NS: (a) Fourier Neural Operator (FNO) [42], (b) Convolutional Neural Operators (CNO) [43], (c) DeepONet [44]. The results for the other cases will be available on the FlowBench website. We used the Python packages/code base released by the developers of these three approaches—deepxde (DeepONet), neuraloperators (FNO), ConvolutionalNeuralOperator (CNO)—and closely followed the published code examples. All models were trained on a single A100 80GB GPU, batch size 5 (for the 512×512 resolution case), Adam optimizer with learning rate 10^{-3}, and run for 400 epochs. The validation loss seems to saturate by 400 epochs.

Table 2: *The mean squared errors of CNO and FNO neural operators trained on the 2D LDC dataset at two different resolutions. All errors are reported on the validation dataset. We denote by * the cases in which our implementations diverged. 0^+ indicates an error value comparable to machine precision.*

		Metrics	FNO		CNO	
High Resolution	**G1** 512x512	M1	4.5×10^{-2}		$\mathbf{3.1 \times 10^{-4}}$	
		M2	9×10^{-3}		$\mathbf{2.5 \times 10^{-4}}$	
		M3	0.31	0.31	**0.0072**	**0.0061**
	G2 512x512	M1	4.8×10^{-2}		$\mathbf{3.8 \times 10^{-4}}$	
		M2	1.3×10^{-2}		$\mathbf{3.0 \times 10^{-4}}$	
		M3	2.5	2.5	**0.0066**	**0.0054**
	G3 512x512	M1	4.4×10^{-2}		$\mathbf{6.9 \times 10^{-4}}$	
		M2	9.2×10^{-3}		$\mathbf{4.5 \times 10^{-4}}$	
		M3	2.6	2.6	**0.0051**	**0.0046**
	All (G1, G2, G3) 512x512	M1	4.6×10^{-2}		$\mathbf{2.1 \times 10^{-4}}$	
		M2	3.8×10^{-3}		$\mathbf{2.1 \times 10^{-4}}$	
		M3	0.23	0.23	**0.0052**	**0.0047**
Low Resolution	**G1** 256x256	M1	$O(10^3)^*$		$\mathbf{4.4 \times 10^{-4}}$	
		M2	$O(10^3)^*$		$\mathbf{6.5 \times 10^{-4}}$	
		M3	$O(10^3)^*$	$O(10^3)^*$	**0.0290**	$\mathbf{0^+}$
	G2 256x256	M1	4.9×10^{-2}		$\mathbf{2.7 \times 10^{-4}}$	
		M2	7.7×10^{-2}		$\mathbf{3.1 \times 10^{-4}}$	
		M3	6.1	0^+	**0.0047**	$\mathbf{0^+}$
	G3 256x256	M1	$O(10^3)^*$		$\mathbf{4.9 \times 10^{-4}}$	
		M2	$O(10^3)^*$		$\mathbf{6.8 \times 10^{-4}}$	
		M3	$O(10^3)^*$	$O(10^3)^*$	**0.0220**	$\mathbf{0^+}$
	All (G1, G2, G3) 256x256	M1	$O(10^3)^*$		$\mathbf{2.8 \times 10^{-4}}$	
		M2	$O(10^3)^*$		$\mathbf{1.1 \times 10^{-4}}$	
		M3	$O(10^3)^*$	$O(10^3)^*$	**0.0210**	$\mathbf{0^+}$

While FNO and CNO perform well for the individual datasets, the combined $(G1, G2, G3)$ dataset proved more difficult. Table 2 provides a global view of the performance of the three approaches. We evaluate on three of the metrics, across each of the geometry classes, and at 2 resolutions for FNOs, CNOs, and DeepONets. Unfortunately, we could not get the standard implementation of DeepONets to converge (detailed metrics are reported in the Appendix). Some of the mean squared error of the lift coefficient is very small and close to machine precision ($O(10^{-8})$) and are denoted as 0^+ in Table 2. CNO performed well across all three metrics at both resolutions. In addition, CNO appears to require less tuning (just a single hyperparameter) to produce good results. However, we suspect that fine-tuning the models could improve performance, and we continue to explore this. We encourage the community to further test and build on these results.

5 Conclusions

We introduce a comprehensive benchmark dataset designed for evaluating neural solvers of flow simulations over complex geometries. FlowBench encompasses 2D and 3D simulations, covering many scenarios, from steady-state problems to time-dependent problems. FlowBench will be useful in advancing scientific machine-learning methods for complex geometries by providing datasets and evaluation metrics. Neural PDE solvers that can account for the effect of complex geometries can have a major impact on various applications ranging from bioengineering and power production to automotive and aerospace engineering defined by the interaction of complex geometrical objects with a fluid medium.

Limitations: (1) At the time of submission, our evaluation of existing neural PDE solvers is limited to two of the four FlowBench datasets and on only three of the many promising neural PDE approaches. We invite the community to contribute to evaluating a broader set of approaches using this dataset and suggested metrics. (2) We plan to continue to add more data – and invite the CFD community to add more data – to FlowBench , especially for 3D simulations, and push the range of operating conditions (higher Re and Gr).

References

[1] David Greenblatt and Israel J Wygnanski. The control of flow separation by periodic excitation. *Progress in aerospace Sciences*, 36(7):487–545, 2000.

[2] A Spohn and P Gilliéron. Flow separations generated by a simplified geometry of an automotive vehicle. In *IUTAM Symposium: unsteady separated flows*, volume 1. Kluwer Academic,, 2002.

[3] D You and Parviz Moin. Active control of flow separation over an airfoil using synthetic jets. *Journal of Fluids and structures*, 24(8):1349–1357, 2008.

[4] Rubina Ramponi and Bert Blocken. Cfd simulation of cross-ventilation for a generic isolated building: Impact of computational parameters. *Building and environment*, 53:34–48, 2012.

[5] Tore A Helgedagsrud, Yuri Bazilevs, Kjell M Mathisen, and Ole A Øiseth. Ale-vms methods for wind-resistant design of long-span bridges. *Journal of Wind Engineering and Industrial Aerodynamics*, 191:143–153, 2019.

[6] Kendrick Tan, Boshun Gao, Cheng-Hau Yang, Emily L Johnson, Ming-Chen Hsu, Alberto Passalacqua, Adarsh Krishnamurthy, and Baskar Ganapathysubramanian. A computational framework for transmission risk assessment of aerosolized particles in classrooms. *Engineering with Computers*, pages 1–22, 2023.

[7] Jung-Hee Seo and Rajat Mittal. Improved swimming performance in schooling fish via leading-edge vortex enhancement. *Bioinspiration & Biomimetics*, 17(6):066020, 2022.

[8] Yi Zhu, Fang-Bao Tian, John Young, James C Liao, and Joseph CS Lai. A numerical study of fish adaption behaviors in complex environments with a deep reinforcement learning and immersed boundary–lattice boltzmann method. *Scientific Reports*, 11(1):1691, 2021.

[9] Jialei Song, Haoxiang Luo, and Tyson L Hedrick. Three-dimensional flow and lift characteristics of a hovering ruby-throated hummingbird. *Journal of The Royal Society Interface*, 11(98): 20140541, 2014.

[10] T Engels, D Kolomenskiy, Kai Schneider, F-O Lehmann, and J Sesterhenn. Bumblebee flight in heavy turbulence. *Physical review letters*, 116(2):028103, 2016.

[11] Fei Xu, Emily L Johnson, Chenglong Wang, Arian Jafari, Cheng-Hau Yang, Michael S Sacks, Adarsh Krishnamurthy, and Ming-Chen Hsu. Computational investigation of left ventricular hemodynamics following bioprosthetic aortic and mitral valve replacement. *Mechanics Research Communications*, 112:103604, 2021.

[12] LL Ting. Effects of dimple size and depth on golf ball aerodynamic performance. In *Fluids engineering division summer meeting*, volume 36975, pages 811–817, 2003.

[13] Muhammad Mahmood Aslam Bhutta, Nasir Hayat, Muhammad Hassan Bashir, Ahmer Rais Khan, Kanwar Naveed Ahmad, and Sarfaraz Khan. Cfd applications in various heat exchangers design: A review. *Applied Thermal Engineering*, 32:1–12, 2012.

[14] Chamil Abeykoon. Compact heat exchangers–design and optimization with cfd. *International Journal of Heat and Mass Transfer*, 146:118766, 2020.

[15] Rajagopalan Priyadarsini, Wong Nyuk Hien, and Cheong Kok Wai David. Microclimatic modeling of the urban thermal environment of singapore to mitigate urban heat island. *Solar energy*, 82(8):727–745, 2008.

[16] Jonas Allegrini and Jan Carmeliet. Simulations of local heat islands in zürich with coupled cfd and building energy models. *Urban climate*, 24:340–359, 2018.

[17] Suvanjan Bhattacharyya, Kunal Dey, Akshoy Ranjan Paul, and Ranjib Biswas. A novel cfd analysis to minimize the spread of covid-19 virus in hospital isolation room. *Chaos, Solitons & Fractals*, 139:110294, 2020.

[18] Kumar Saurabh, Masado Ishii, Makrand A Khanwale, Hari Sundar, and Baskar Ganapathysubramanian. Scalable adaptive algorithms for next-generation multiphase flow simulations. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 590–601. IEEE, 2023.

[19] Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11), November 2020. ISSN 1942-2466. doi: 10.1029/2020ms002203. URL http://dx.doi.org/10.1029/2020MS002203.

[20] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

[21] Yining Luo, Yingfa Chen, and Zhen Zhang. Cfdbench: A large-scale benchmark for machine learning methods in fluid dynamics, 2024.

[22] Wenzhuo Xu, Noelia Grande Gutierrez, and Christopher McComb. MegaFlow2D: A parametric dataset for machine learning super-resolution in computational fluid dynamics simulations. In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. Association for Computing Machinery, 2023. URL https://doi.org/10.1145/3576914.3587552.

[23] Steeven Janny, Aurélien Béneteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers, 2023.

[24] Florent Bonnet, Jocelyn Ahmed Mazari, Thibaut Munzer, Pierre Yser, and Patrick Gallinari. An extensible benchmarking graph-mesh dataset for studying steady-state incompressible navier-stokes equations, 2022.

[25] Ryley McConkey, Eugene Yee, and Fue-Sang Lien. A curated dataset for data-driven turbulence modelling. *Scientific Data*, 8(1), September 2021. ISSN 2052-4463. doi: 10.1038/s41597-021-01034-2. URL http://dx.doi.org/10.1038/s41597-021-01034-2.

[26] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2023.

[27] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

[28] Jakob Jakob, Markus Gross, and Tobias Günther. A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1279–1289, 2020.

[29] Sheikh Md Shakeel Hassan, Arthur Feeney, Akash Dhruv, Jihoon Kim, Youngjoon Suh, Jaiyoung Ryu, Yoonjin Won, and Aparna Chandramowlishwaran. BubbleML: A multiphase multiphysics dataset and benchmarks for machine learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=0Wmglu8zak.

[30] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[31] Deheng Wei, Jianfeng Wang, and Budi Zhao. A simple method for particle shape generation with spherical harmonics. *Powder Technology*, 330:284–291, 2018. ISSN 0032-5910. doi: https://doi.org/10.1016/j.powtec.2018.02.006. URL https://www.sciencedirect.com/science/article/pii/S0032591018301189.

[32] Ilke Demir, Camilla Hahn, Kathryn Leonard, Geraldine Morin, Dana Rahbani, Athina Panotopoulou, Amelie Fondevilla, Elena Balashova, Bastien Durix, and Adam Kortylewski. Skelneton 2019: Dataset and challenge on deep learning for geometric shape understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[33] Rowel Atienza. Pyramid u-network for skeleton extraction from shape points. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[34] Alex Main and Guglielmo Scovazzi. The shifted boundary method for embedded domain computations. part i: Poisson and stokes problems. *Journal of Computational Physics*, 372: 972–995, 2018.

[35] Alex Main and Guglielmo Scovazzi. The shifted boundary method for embedded domain computations. part ii: Linear advection-diffusion and incompressible navier-stokes equations. *J. Comput. Phys.*, 372:996–1026, 2018.

[36] Cheng-Hau Yang, Kumar Saurabh, Guglielmo Scovazzi, Claudio Canuto, Adarsh Krishnamurthy, and Baskar Ganapathysubramanian. Optimal surrogate boundary selection and scalability studies for the shifted boundary method on octree meshes. *Computer Methods in Applied Mechanics and Engineering*, 419:116686, 2024.

[37] Kumar Saurabh, Boshun Gao, Milinda Fernando, Songzhe Xu, Makrand A Khanwale, Biswajit Khara, Ming-Chen Hsu, Adarsh Krishnamurthy, Hari Sundar, and Baskar Ganapathysubramanian. Industrial scale large eddy simulations with adaptive octree meshes using immersogeometric analysis. *Computers & Mathematics with Applications*, 97:28–44, 2021.

[38] Makrand A Khanwale, Kumar Saurabh, Masado Ishii, Hari Sundar, James A Rossmanith, and Baskar Ganapathysubramanian. A projection-based, semi-implicit time-stepping approach for the cahn-hilliard navier-stokes equations on adaptive octree meshes. *Journal of Computational Physics*, 475:111874, 2023.

[39] Songzhe Xu, Boshun Gao, Ming-Chen Hsu, and Baskar Ganapathysubramanian. A residual-based variational multiscale method with weak imposition of boundary conditions for buoyancy-driven flows. *Computer Methods in Applied Mechanics and Engineering*, 352:345–368, 2019.

[40] Sungu Kim, Kumar Saurabh, Makrand A Khanwale, Ali Mani, Robbyn K Anand, and Baskar Ganapathysubramanian. Direct numerical simulation of electrokinetic transport phenomena in fluids: Variational multi-scale stabilization and octree-based mesh refinement. *Journal of Computational Physics*, 500:112747, 2024.

[41] Dan Stanzione, John West, R Todd Evans, Tommy Minyard, Omar Ghattas, and Dhabaleswar K Panda. Frontera: The evolution of leadership computing at the National Science Foundation. In *Practice and Experience in Advanced Research Computing*, pages 106–111, 2020.

[42] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[43] Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of pdes, 2023.

[44] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218—229, 2021.

[45] Nabil M Atallah, Claudio Canuto, and Guglielmo Scovazzi. The second-generation shifted boundary method and its numerical analysis. *Computer Methods in Applied Mechanics and Engineering*, 372:113341, 2020.

[46] Nabil Atallah, Claudio Canuto, and Guglielmo Scovazzi. The shifted boundary method for solid mechanics. *International Journal for Numerical Methods in Engineering*, 122(20):5935–5970, 2021.

[47] Charles S Peskin. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics*, 10(2):252–271, 1972.

[48] Lucy Zhang, Axel Gerstenberger, Xiaodong Wang, and Wing Kam Liu. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, 193(21-22):2051–2067, 2004.

[49] Rajat Mittal, Haibo Dong, Meliha Bozkurttas, FM Najjar, Abel Vargas, and Alfred Von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of computational physics*, 227(10):4825–4852, 2008.

[50] J. Parvizian, A. Düster, and E. Rank. Finite cell method: *h*- and *p*- extension for embedded domain methods in solid mechanics. *Computational Mechanics*, 41:122–133, 2007.

[51] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 284:1005–1053, 2015.

[52] Kumar Saurabh, Masado Ishii, Milinda Fernando, Boshun Gao, Kendrick Tan, Ming-Chen Hsu, Adarsh Krishnamurthy, Hari Sundar, and Baskar Ganapathysubramanian. Scalable adaptive pde solvers in arbitrary domains. In *Proceedings of the International Conference for high performance computing, networking, storage and analysis*, pages 1–15, 2021.

[53] Tingting Huang and Hee-Chang Lim. Simulation of lid-driven cavity flow with internal circular obstacles. *Applied Sciences*, 10(13):4583, 2020.

[54] Zhen Chen, Chang Shu, L. Yang, Xiang Zhao, and Ningyu Liu. Immersed boundary–simplified thermal lattice boltzmann method for incompressible thermal flows. *Physics of Fluids*, 32, 01 2020. doi: 10.1063/1.5138711.

[55] Ronald D Henderson. Details of the drag curve near the onset of vortex shedding. *Physics of Fluids*, 7(9):2102–2104, 1995.

[56] Haoxiang Luo, Hu Dai, and Paulo Ferreira de Sousa. A hybrid formulation to suppress the numerical oscillations caused by immersed moving boundaries. In *APS Division of Fluid Dynamics Meeting Abstracts*, volume 62, pages EL–006, 2009.

[57] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 284:1005–1053, 2015.

[58] Alex Main and Guglielmo Scovazzi. The shifted boundary method for embedded domain computations. part ii: Linear advection–diffusion and incompressible navier–stokes equations. *Journal of Computational Physics*, 372:996–1026, 2018.

[59] Soonpil Kang and Arif Masud. A variational multiscale method with immersed boundary conditions for incompressible flows. *Meccanica*, 56(6):1397–1422, 2021.

[60] Yoann Cheny and Olivier Botella. The ls-stag method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. *Journal of Computational Physics*, 229(4):1043–1076, 2010.

[61] W. Jester and Y. Kallinderis. Numerical study of incompressible flow about fixed cylinder pairs. *Journal of Fluids and Structures*, 17(4):561–577, 2003. ISSN 0889-9746. doi: https://doi.org/10.1016/S0889-9746(02)00149-4. URL https://www.sciencedirect.com/science/article/pii/S0889974602001494.

[62] Utkarsh Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.

Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[Yes]**
    (b) Did you describe the limitations of your work? **[Yes]** **We describe the limitations in Section 5.**
    (c) Did you discuss any potential negative societal impacts of your work? [NA] Since the data generated involves fundamental science, the negative impact it has on the society is of similar nature to any fundamental science itself.
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[Yes]**
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? **[NA]**
    (b) Did you include complete proofs of all theoretical results? **[NA]**
3. If you ran experiments (e.g., for benchmarks):
    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[Yes]** **We provide the code, data, and instructions to reproduce the main experimental results on the FlowBench website.**
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[Yes]**
    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[Yes]** **The dataset itself is generated deterministically from simulations. Where possible, we ran multiple experiments and will report the error.**
    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[Yes]** **We mention the simulation efforts in Section 3.3**.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
    (a) If your work uses existing assets, did you cite the creators? **[Yes]** **We cite the code used for SciML in Section 4**
    (b) Did you mention the license of the assets? **[Yes]**
    (c) Did you include any new assets either in the supplemental material or as a URL? **[Yes]** **We have included details of geometry generation and simulation results on our FlowBench website.**
    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[NA]**
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[NA]**
5. If you used crowdsourcing or conducted research with human subjects **[NA]** .
    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [NA]
    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [NA]
    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [NA]

A Details of the CFD simulation framework

Our CFD framework is a well-validated and massively scalable software suite that uses a variant of the immersed boundary method integrated with octree meshes to perform highly efficient and accurate Large-Eddy Simulations (LES) of flows around complex geometries. This framework demonstrates scalability of up to 32,000 processors, achieved through several key innovations, including (a) Rapid in-out tests: These tests quickly determine whether a point is inside or outside a given geometry, significantly speeding up the simulation process. (b) adaptive quadrature: This technique ensures accurate force evaluation by dynamically adjusting the numerical integration based on the local complexity of the geometry, and (c) Tensorized operations: These operations optimize performance by leveraging tensor algebra for computational efficiency. Additional details of the CFD framework, including implementation, are provided in [18, 37].

Our datasets and code are licensed under CC BY NC. Next, we briefly detail the mathematics of the shifted boundary method.

A.1 Formulation of SBM for Navier Stokes

The Shifted Boundary Method (SBM) [34–36, 45, 46] is a numerical approach for solving partial differential equations (PDEs) on complex geometries without the need for body-fitted meshes. It is a robust variant of the immersed boundary methods used in CFD [11, 47–51]. In SBM, the boundary conditions are imposed not on the actual boundary of the immersed object but on a nearby surrogate boundary. This surrogate boundary is chosen to conform to a Cartesian mesh, and the boundary conditions are corrected using Taylor expansions. This method effectively transforms the problem into one on a body-fitted domain, significantly simplifying the mesh generation process while maintaining accuracy and stability. The surrogate boundary is optimally chosen to minimize numerical errors, and the method demonstrates excellent scalability and efficiency, particularly when applied to adaptive octree meshes (see, for instance, Yang et al. [36]). This makes SBM particularly suitable for simulations involving complex geometries and multiphysics couplings. In SBM, additional terms are incorporated into the standard stabilized finite element formulations for solving the Navier-Stokes equations. These terms include a *consistency* term (which appears due to integration by parts operation), an *adjoint consistency* term (which is included to ensure optimal convergence rates), and the *penalty* term (which ensures that as the mesh size is reduced, the boundary conditions asymptote to the true boundary conditions) :

$$
\begin{aligned}
\widetilde{B_{NS}^{VMS}} = B_{NS}^{VMS} & \underbrace{- \left\langle w_i^{c,h}, \frac{1}{Re}\left(\frac{\partial u_i^{c,h}}{\partial x_j} + \frac{\partial u_j^{c,h}}{\partial x_i}\right)\tilde{n_j} - p^{c,h}\tilde{n_i} \right\rangle_{\tilde{\Gamma}_{D,h}}}_{Consistency\ term} \\
& \underbrace{- \left\langle \frac{1}{Re}\left(\frac{\partial w_i^{c,h}}{\partial x_j} + \frac{\partial w_j^{c,h}}{\partial x_i}\right)\tilde{n_j} + q^{c,h}\tilde{n_i}, u_i^{c,h} + \frac{\partial u_i^{c,h}}{\partial x_j} d_j \right\rangle_{\tilde{\Gamma}_{D,h}}}_{Adjoint\ consistency\ term} \\
& \underbrace{+ \frac{\beta}{h \cdot Re} \left\langle w_i^{c,h} + \frac{\partial w_i^{c,h}}{\partial x_j} d_j, u_i^{c,h} + \frac{\partial u_i^{c,h}}{\partial x_j} d_j \right\rangle_{\tilde{\Gamma}_{D,h}}}_{Penalty\ term}, \qquad (1)
\end{aligned}
$$

and

$$
\begin{aligned}
\widetilde{F_{NS}^{VMS}} = F_{NS}^{VMS} & \underbrace{- \left\langle \frac{1}{Re}\left(\frac{\partial w_i^{c,h}}{\partial x_j} + \frac{\partial w_j^{c,h}}{\partial x_i}\right)\tilde{n_j} + q^{c,h}\tilde{n_i}, g_i \right\rangle_{\tilde{\Gamma}_{D,h}}}_{Adjoint\ Consistency\ Term} \\
& \underbrace{+ \frac{\beta}{h \cdot Re} \left\langle w_i^{c,h} + \frac{\partial w_i^{c,h}}{\partial x_j} d_j, g_i \right\rangle_{\tilde{\Gamma}_{D,h}}}_{Penalty\ Term}, \qquad (2)
\end{aligned}
$$

where β is the penalty parameter for the Navier-Stokes equation, B_{NS}^{VMS} is the bilinear weak form for Navier-Stokes without SBM, and F_{NS}^{VMS} is the linear weak for Navier-Stokes form without SBM. The formulation for Navier-Stokes with SBM can be expressed as:

$$\widetilde{B_{NS}^{VMS}} - \widetilde{F_{NS}^{VMS}} = 0. \tag{3}$$

A.2 Formulation of SBM for Heat Transfer

Similar to the Navier-Stokes equations, we use SBM to apply Dirichlet boundary conditions in the energy equation. Essentially, we use SBM to set the temperature to a desired value (T_D) on the geometries. The formulation for energy equation with SBM is:

$$\widetilde{B_{HT}^{VMS}} - \widetilde{F_{HT}^{VMS}} = 0, \tag{4}$$

with

$$\begin{aligned}\widetilde{B_{HT}^{VMS}} =& B_{HT}^{VMS} \underbrace{-\frac{1}{Pe}\left\langle l^{c,h}, \frac{\partial T^{c,h}}{\partial x_j}\tilde{n_j}\right\rangle_{\tilde{\Gamma}_{D,h}}}_{\text{Consistency term}} \underbrace{-\frac{1}{Pe}\left\langle \frac{\partial l^{c,h}}{\partial x_j}\tilde{n_j}, T^{c,h} + \frac{\partial T^{c,h}}{\partial x_j}d_j\right\rangle_{\tilde{\Gamma}_{D,h}}}_{\text{Adjoint consistency term}} \\ & \underbrace{+\frac{\alpha}{h \cdot Pe}\left\langle l^{c,h} + \frac{\partial l^{c,h}}{\partial x_j}d_j, T^{c,h} + \frac{\partial T^{c,h}}{\partial x_j}d_j\right\rangle_{\tilde{\Gamma}_{D,h}}}_{\text{Penalty term}},\end{aligned} \tag{5}$$

and

$$\widetilde{F_{HT}^{VMS}} = F_{HT}^{VMS} \underbrace{-\frac{1}{Pe}\left\langle \frac{\partial l^{c,h}}{\partial x_j}\tilde{n_j}, T_D\right\rangle_{\tilde{\Gamma}_{D,h}}}_{\text{Adjoint consistency term}} \underbrace{+\frac{\alpha}{h \cdot Pe}\left\langle l^{c,h} + \frac{\partial l^{c,h}}{\partial x_j}d_j, T_D\right\rangle_{\tilde{\Gamma}_{D,h}}}_{\text{Penalty term}}, \tag{6}$$

where α is the penalty parameter for energy equation, Pe is Peclet number, which is $0.7Re$ inside our simulations, B_{HT}^{VMS} is the bilinear weak form for Heat Transfer without SBM, and F_{HT}^{VMS} is the linear weak form for Heat Transfer without SBM.

A.3 Solving the CFD equations: Automated creation of meshes involving complex geometries

Tree-based mesh generation, using quadtrees in 2D and octrees in 3D, is common in computational sciences due to its simplicity and parallel scalability. These tree-based data structures enable efficient refinement of regions of interest, facilitating their deployment in large-scale multi-physics simulations. Our mesh generation tool, Dendro-kt [52], provides balanced, partitioned, and parallel tree structures, making it highly effective for large-scale numerical PDE discretizations.

LDC (Lid-driven cavity flow): For the LDC case, we use an octree-based mesh generation framework to create a uniform mesh with an element size of $\frac{1}{2^9}$ over a $[0, 2] \times [0, 2]$ domain. This produces a 512×512 mesh resulting in total degress-of-freedom of $512 \times 512 \times 3 \sim 750K$.

FPO (Flow passing object): Our computational domain is a rectangular region spanning $[0, 64] \times [0, 16]$, with the complex geometry centered at the coordinates (6, 8). Our mesh refinement strategy involves several layers of progressively coarser mesh surrounding the geometry and along the wake. Specifically, we utilize five concentric circles centered at (6, 8), with the innermost circle having a radius of 0.71 and a refinement level of 13 (yielding an element size of $\frac{64}{2^{13}}$), the next circle with a radius of 0.8 and a refinement level of 12 ($\frac{64}{2^{12}}$ element size), the third circle with a radius of 1 and a refinement level of 11 ($\frac{64}{2^{11}}$ element size), the fourth circle with a radius of 2.5 and a refinement level of 10 ($\frac{64}{2^{10}}$ element size), and the outermost circle with a radius of 3 and a refinement level of 9 ($\frac{64}{2^{9}}$ element size). We use a non-dimensional time step of 0.01 for the simulation. Starting from a non-dimensional total time of 392, we begin outputting results every 0.05 non-dimensional time units until reaching a non-dimensional total time of 404. The period from non-dimensional time 392 to 404 is when we output results that are post-processed for use in FlowBench .

Additionally, we define two rectangular refinement regions aligned with the flow direction. The first rectangle has its bottom-left corner at (6, 5.5) and top-right corner at (64, 10.5), with a refinement

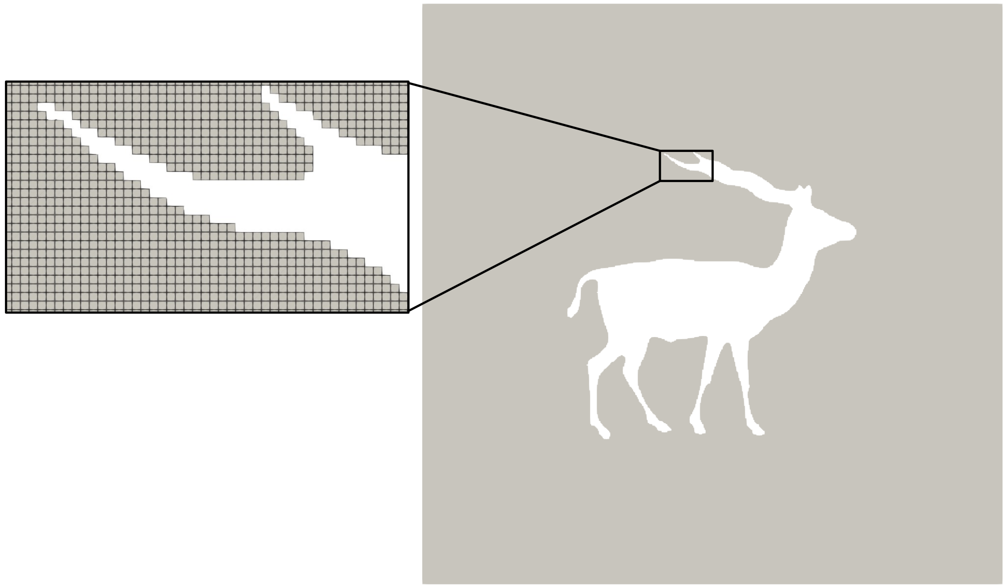


Figure 6: *Uniform octree mesh for the lid-driven cavity flow. We illustrate the mesh with a representative shape from the Skelneton data. This mesh is carved out from a* 512×512 *uniform tessellation of the domain.*

level of 10 ($\frac{64}{2^{10}}$ element size). The second rectangle has its bottom-left corner at (6, 5) and top-right corner at (64, 11), with a refinement level of 9 ($\frac{64}{2^{9}}$ element size). Close to the geometry, to ensure detailed capture of the flow dynamics, we achieve a finer mesh with a refinement level of 14, resulting in an element size of $\frac{64}{2^{14}}$. The mesh with different refinement levels are shown in Figure 7.

The refinement strategy leads to a total nodal point number of 117978. Consequently, the total degrees of freedom (DOFs) amount is 117978 $\times$ 3 (353934). The combination of adaptive refinement and SBM formulation is critical to our ability to solve the PDEs in complex geometries. For comparison, if we were to use a uniform mesh (as in the case of the LDC) instead of an adaptively refined mesh, we would have had a problem with $2^{14} \times 2^{12} \times 3 \sim 201M$ DOFs!

3D LDC: We use a similar adaptive refinement strategy for the 3D LDC case. See Figure 8 for an example of an object and the mesh used. Close to the object's surface, we use a refinement of level 9, producing elements of size $2/2^9$, and away from the object, we progressively coarsen the mesh to a refinement level of 7, for elements of size $2/2^7$. This produces a problem with $2.5M$ DOFs.

Solver configuration: We use the Petsc linear algebra package for solving the system of equations. We utilize the BCGS solver, with an ASM pre-conditioner. The solver continues iterating until a relative tolerance, $rtol$, of 10^{-8}, is reached.

A.4 Postprocessing of the results to compute force coefficients and Nusselt number

After the CFD solve, we also compute three engineering summary variables. The drag coefficient (C_D) represents the non-dimensional force exerted on an object in the direction of the flow. It is calculated using the formula:

$$C_D = \frac{2F_x}{A_{py}}. \tag{7}$$

where F_x is the drag force and A_{py} is the projection area in the y-direction.

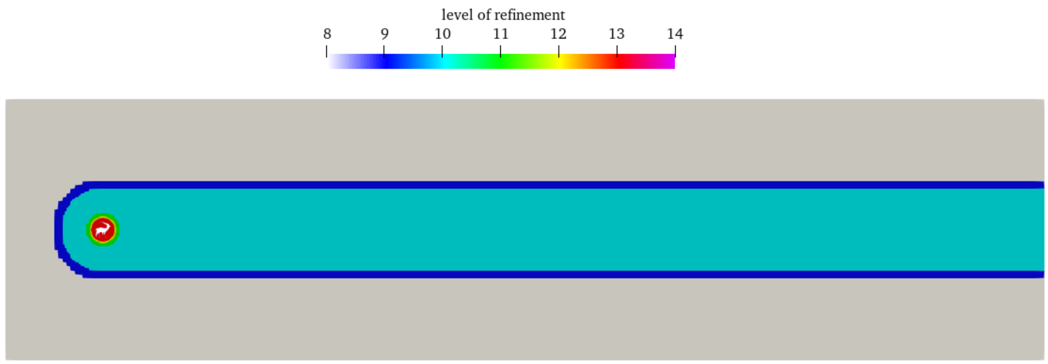
The lift coefficient (C_L) indicates the non-dimensional lift force acting perpendicular to the flow direction. It is defined as:

$$C_L = \frac{2F_y}{A_{px}}. \tag{8}$$

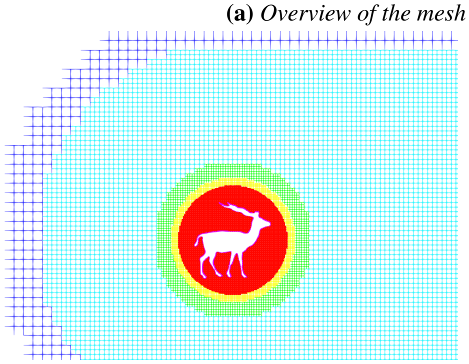
where F_y is the lift force and A_{px} is the projection area in the x-direction.

Finally, we compute the Nusselt number to determine the amount of heat transfer. The local Nusselt number can be defined as:

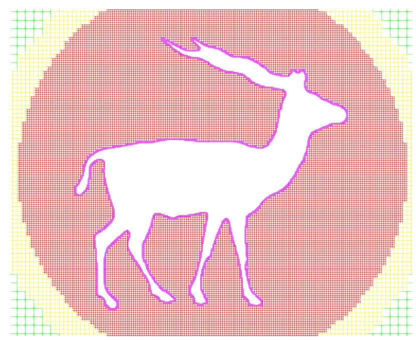
$$Nu = \nabla T \cdot \mathbf{n}. \tag{9}$$



(a) *Overview of the mesh refinement levels.*

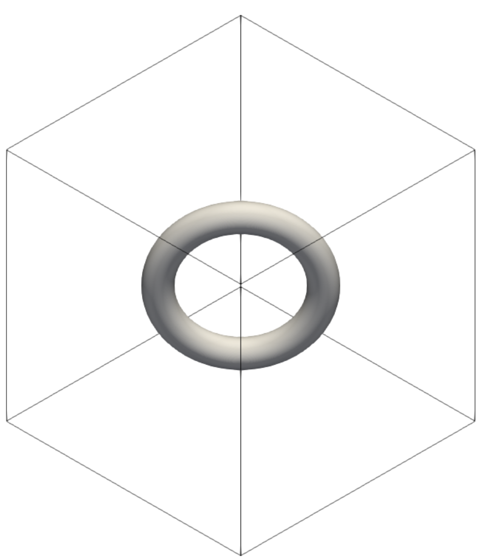


(b) *Zoomed-in view of the local mesh refinement. The five levels of progressively coarser meshes surrounding the geometry are clearly visible.*

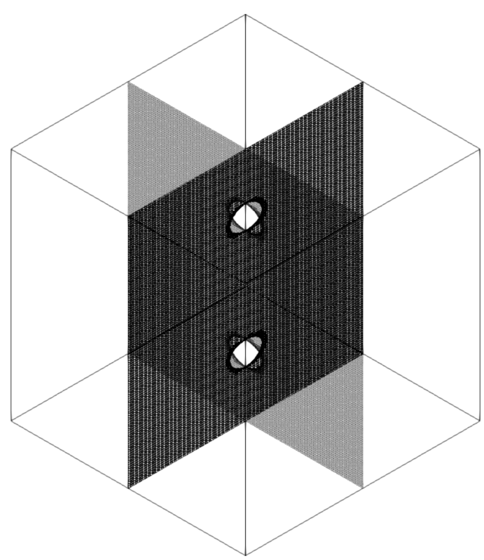


(c) *Zoomed-in view of the local mesh refinement near the geometry.*

Figure 7: *Local refinement octree mesh for the flow passing through the geometry. The geometry is represented by a deer.*



(a) *The object inside the lid driven cavity domain.*



(b) *The octree mesh showing local refinement near the object boundary.*

Figure 8: *An example shape in the 3D LDC case, along with a slice of the computational mesh*

Table 3: *Comparison of drag coefficients and strouhal numbers for flow past a 2D cylinder at Re = 100.*

	Cd	**St**
Current	1.35	0.167
Mittal *et al.* [49]	1.35	0.165
Henderson *et al.* [55]	1.35	-
Luo *et al.* [56]	1.35	0.159
Kamensky *et al.* [57]	1.386	0.170
Main *et al.* [58]	1.36	0.169
Kang *et al.* [59]	1.374	0.168

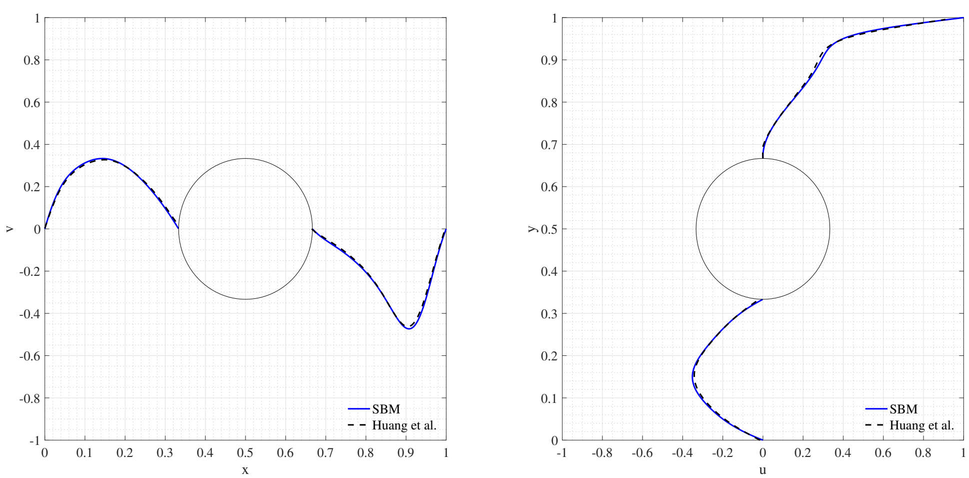
The averaged Nusselt number can be written as:

$$\overline{Nu} = \frac{\int Nu \, d\Gamma}{\int d\Gamma}. \tag{10}$$

We compute the average Nu across both the bottom wall as well as the surface of the object.

A.5 Validation of the CFD framework

LDC (Lid-driven cavity flow), NS: We simulate a disk with a diameter $D = \frac{L}{3}$ placed at the center of the lid-driven cavity, where L is the chamber's box length. Our CFD results match against detailed simulations available in literature [53] shown in Figure 9.



(a) *Horizontal velocity profile.* **(b)** *Vertical velocity profile.*

Figure 9: *Comparison of velocity profiles for lid-driven cavity flow with a circular disk at Re = 1000 against results from the literature.*

LDC (Lid-driven cavity flow), NSHT: To validate our multiphysics simulation framework, we select a case from [54] to compare with. Here, a heated circle is placed at the center of the chamber, with a radius of 0.2L, where L represents the length of the chamber. We evaluate the local Nusselt number on the bottom wall and report an excellent match with [54] in Figure 10 at two distinct operating conditions.

FPO (Flow passing object): For the flow passing through geometries, we tested several shapes. To ensure the reliability of our framework, we used a circle and tested at two different Reynolds numbers: 100 and 1000. Our drag coefficients and Strouhal numbers matched well with the literature, as shown in Table 3 and Table 4.

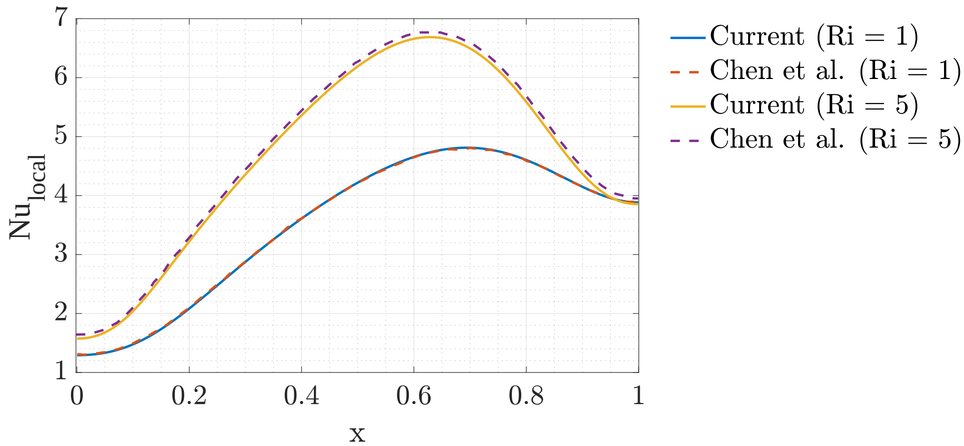


Figure 10: *Comparison of the Nusselt number profile along the bottom wall of the domain. The simulation results are validated against data from [54] to demonstrate the accuracy of the current multiphyics (NSHT) model.*

Table 4: *Comparison of drag coefficients and strouhal numbers for flow past a 2D cylinder at Re = 1000.*

	Cd	**St**
Current	1.52	0.239
Mittal *et al.* [49]	1.45	0.230
Henderson *et al.* [55]	1.51	-
Luo *et al.* [56]	1.56	0.235
Cheny *et al.* [60]	1.61	0.251
Jester *et al.* [61]	1.51	0.25

B Benchmark Setup

B.1 Downsampling

We used our in-house FEM simulation software `DendroKT` to perform forward simulations. In order to capture the physics precisely, we use a very fine resolution – with mesh sizes at the Kolmogorov length scale in 2D, and mesh sizes at $2\times$ the Kolmogorov length scale in 3D – to run the FEM simulations. While this approach allows us to model the physics very precisely, the resulting tensors generated at those resolutions are impractically large in size i.e., running into hundreds of gigabytes. We therefore use the ParaView tool [62] to downsample the original resolution to lower resolutions such as, 512×512, 256×256, 128×128, $128 \times 128 \times 128$ and 512×128. It is important to note that this downsampling is on the fully resolved data, and thus still captures all the larger scale features (as well as the impact of the small scale features on the large scale features).

B.2 Machine Learning

Our Machine Learning pipeline consists of three modules. In the first module, we undertake the tensorization exercise. We group the available downsampled data to a four or five dimensional *numpy* tensor depending on the nature of the physics outlined in Table 2. Tensorization helps us to subset the data using the tensor notation, and it also makes the data ready to be fed into Neural Operators. Data at this point is fed into the second module, where we primarily train the Neural Operator. We have two principal objectives in this module, a) Learn the best hyperparameters for a given dataset and a given Neural Operator. b) Select the best hyperparameter and perform a final training exercise to record the performance of each Neural Operator on every dataset. Finally, we move to the post-processing module, where we report a complete suite of goodness of fit statistics using the trained model on a held-out data sample. We make all our code publicly available.

Table 5: *Formulaic description of the input and output tensors. 3000/6000/1150/500 are sample sizes for the dataset. 240 is the number of equi-spaced time snapshots for the FPO case; $x, y(, z)$ are the dimensions of a field. E.g., $Y[0, 1, :, :]$ indicates the pointwise v velocity over the entire grid.*

Dataset	Dim.	Input Tensor	Output Tensor
LDC - NS	2	$X[3000][Re, g, s][x][y]$	$Y[3000][u, v, p][x][y]$
LDC - NS+HT	2	$X[5990][Re, Gr, g, s][x][y]$	$Y[5990][u, v, p, \theta][x][y]$
FPO - NS	2	$X[1150][Re, g, s][x][y]$	$Y[1150][240][u, v, p][x][y]$
LDC - NS	3	$X[500][Re, g, s][x][y][z]$	$Y[500][u, v, p][x][y][z]$

B.3 DeepONet Results

As mentioned in the main paper, we could not get the standard implementation of DeepONets to converge on our dataset. We still report the overall errors here for completeness.

Table 6: *The mean squared errors of DeepONet neural operators trained on the 2D LDC dataset at two different resolutions. All errors are reported on the validation dataset for three different metrics.*

		Metrics	DeepONet
High Resolution	**G1** 512x512	M1	$O(10^3)^*$
		M2	$O(10^3)^*$
		M3	$O(10^3)^*$
	G2 512x512	M1	$O(10^3)^*$
		M2	$O(10^3)^*$
		M3	$O(10^3)^*$
	G3 512x512	M1	$O(10^3)^*$
		M2	$O(10^3)^*$
		M3	$O(10^3)^*$
	All (G1,G2,G3) 512x512	M1	$O(10^3)^*$
		M2	$O(10^3)^*$
		M3	$O(10^2)^*$
Low Resolution	**G1** 256x256	M1	$O(10^2)^*$
		M2	$O(10^3)^*$
		M3	$O(10^3)^*$
	G2 256x256	M1	$O(10^3)^*$
		M2	$O(10^3)^*$
		M3	$O(10^3)^*$
	G3 256x256	M1	$O(10^2)^*$
		M2	$O(10^2)^*$
		M3	$O(10^3)^*$
	All (G1,G2,G3) 256x256	M1	$O(10^3)^*$
		M2	$O(10^3)^*$
		M3	$O(10^3)^*$