DEPARTMENT OF AEROSPACE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

Data-driven and physics-informed deep learning for surrogate modeling of unsteady flows past moving bodies

A Thesis

Submitted by

RAHUL SUNDAR

DEPARTMENT OF AEROSPACE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

Data-driven and physics-informed deep learning for surrogate modeling of unsteady flows past moving bodies

A Thesis

Submitted by

RAHUL SUNDAR

If not now then when, if it is not for all, then why?

– Anonymous

To my mother, father, and sister who always stood like a rock amidst both harsh storms and soothing tides,

To my gurus, who taught me how to learn and distill the essence,

To Vedanta Desikan, the master of all arts and sciences,

To the son of Anjani, who instiled perseverance and reverence,

To the Gitacharyan, who never let go, and

To all those who traverse unexplored paths to make it easy for those who follow after.

THESIS CERTIFICATE

This is to undertake that the Thesis titled **DATA-DRIVEN AND PHYSICS-INFORMED DEEP LEARNING FOR SURROGATE MODELING OF UNSTEADY FLOWS PAST MOVING BODIES**, submitted by me to the Indian Institute of Technology Madras, for the award of **Doctor of Philosophy**, is a bona fide record of the research work done by me under the supervision of **Prof. Sunetra Sarkar**. The contents of this Thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Chennai 600036 **Rahul Sundar**

Date: February 2026

Prof Sunetra Sarkar
Research advisor
Professor
Department of Aerospace Engineering
IIT Madras

LIST OF PUBLICATIONS

I. REFEREED JOURNALS BASED ON THESIS

Sundar, R., Majumdar, D., Lucor, D., & Sarkar, S. (2024). Physics-informed neural networks modelling for systems with moving immersed boundaries: Application to an unsteady flow past a plunging foil. *Journal of Fluids and Structures*, Volume 125, 2024, 104066, ISSN 0889-9746, https://doi.org/10.1016/j.jfluidstructs.2024.104066.

Sundar, R., Lucor, D., & Sarkar, S. (2025). Sequential learning based PINNs to overcome temporal domain complexities in unsteady flow past flapping wings. *Journal of Fluids and Structures*, Volume 139, 2025, 104421, ISSN 0889-9746, https://doi.org/10.1016/j.jfluidstructs.2025.104421.

II. REFEREED JOURNALS (OTHERS)

Thirunavukkarasu, A., **Sundar, R.**, & Sarkar, S. (2024). Comparative analysis of model reduction techniques for flapping wing dynamics. *Physics of Fluids*, 1 June 2024; 36 (6): 067109. https://doi.org/10.1063/5.0209683

III. PUBLICATIONS/BOOK CHAPTERS PART OF CONFERENCE PROCEEDINGS FROM THIS THESIS

Sundar, R., Lucor, D., Sarkar, S. (2025). Understanding the Training of PINNs for Unsteady Flow Past a Plunging Foil Through the Lens of Input Subdomain Level Loss Function Gradients. In: Arun, K.R., Rajesh, G., Arakeri, J.H., Kothadia, H. (eds) *Proceedings of Fluid Mechanics and Fluid Power (FMFP) 2023*, Vol. 5. FMFP 2023. Lecture Notes in Mechanical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-97-7759-4_53

Sundar, R., Majumdar, D., Shah, C.L., Sarkar, S. (2024). Massive Parallelisation and Performance Enhancement of an Immersed Boundary Method Based Unsteady Flow Solver. In: Singh, K.M., Dutta, S., Subudhi, S., Singh, N.K. (eds) *Fluid Mechanics and Fluid Power, Volume 8. FMFP 2022*. Lecture Notes in Mechanical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-97-1033-1_38

IV. PRESENTATIONS IN CONFERENCES FROM THIS THESIS

Sundar, R., Lucor, D., & Sarkar, S. (2024, July). Tackling temporal domain complexity to effectively train PINNs for unsteady flows past moving bodies. In the 16th World Congress on Computational Mechanics and 4th Pan American Congress on Computational Mechanics (WCCM- PANACM 2024), Vancouver,

Canada.

Sundar, R., Majumdar, D., Lucor, D., & Sarkar, S. (2023, April). Data-driven physics-informed and immersed boundary aware surrogate modeling of unsteady flows past moving bodies. In 22nd IACM Computational Fluids Conference 2023 (CFC 2023), Cannes, France.

Sundar, R., Lucor, D., & Sarkar, S. (2023, December). Understanding the training of PINNs for unsteady flow past a plunging foil through the lens of input subdomain level loss function gradients. In 10th International and 50th National Conference on Fluid Mechanics and Fluid Power (FMFP) (No. FMFP2023-12-454).

Sundar, R., Majumdar, D., Shah, C. L., & Sarkar, S. (2022, December). Massive parallelisation and performance enhancement of an immersed boundary method based unsteady flow solver. In Conference on Fluid Mechanics and Fluid Power (pp. 459-472). Singapore: Springer Nature Singapore.

V. PRESENTATIONS IN CONFERENCES (OTHER THAN THESIS)

Moideen, N., **Sundar, R.**, Lucor, D., & Sarkar, S. (2024, July). (Poster) All you need is velocity far away: PINN-PICS based body recovery-load reconstruction framework for unsteady flows. In the 16th World Congress on Computational Mechanics and 4th Pan American Congress on Computational Mechanics (WCCM-PANACM 2024), Vancouver, Canada.

ACKNOWLEDGEMENTS

I offer my obeisance to my parents, my teachers, mentors, the stalwarts of fluid mechanics and applied mathematics, and all the well-wishers who have guided me throughout my academic and personal journey, and who continue to inspire me every day.

I express my deepest gratitude to my advisor, Prof. Sunetra Sarkar, Department of Aerospace Engineering, IIT Madras, for believing in me and for her unwavering support, patient guidance, and incisive scientific wisdom. Her clarity of thought, intellectual rigor, and constant encouragement have shaped my research journey in profound and lasting ways. I am equally grateful to my collaborator, Prof. Didier Lucor, CNRS, France, whose insightful discussions, generosity with his time, and enthusiasm for scientific inquiry have greatly enriched the quality and direction of this work.

I gratefully acknowledge the HPCE Team, P.G. Senapathy Centre for Computing Resources, IIT Madras, for providing access to the high-performance computing facilities that made the high-fidelity simulations in this thesis possible. I also thank NVIDIA and CDAC for their support during the CDAC–GPU Hackathon 2020, which significantly contributed to my understanding of GPU-accelerated computing.

My sincere thanks extend to my doctoral committee members — Prof. R. I. Sujith, Prof. Nandan Kumar Sinha, and Prof. Shaligram Tiwari — for their constructive feedback, technical guidance, and encouragement during various stages of this research. I would also like to acknowledge Prof. Mitesh Khapra and Prof. Balaji Srinivasan, whose courses and discussions strengthened my foundational understanding and played an important role in shaping my research direction.

During my Ph.D. journey, interactions with several industry professionals engaged in translational R&D greatly influenced my transition to industry and exposed me to impactful opportunities at the interface of AI and scientific computing. Their insights

into translational research and the challenges of AI for science and engineering were invaluable.

In these times, computational research is deeply intertwined with open-source frameworks such as OpenFOAM, deal.II, FEniCS, PyTorch, TensorFlow, and JAX. I sincerely thank the open-source communities that sustain these ecosystems. In particular, I thank the OpenFOAM data-driven modeling community — especially the chairs Dr. Andre Weiner and Dr. Tomislav Maric — for introducing me to hybrid CFD–AI modeling through their hackathons and for giving me the opportunity to contribute to their open-source efforts.

I also extend my gratitude to the brilliant student teams at the Centre for Innovation, IIT Madras, who are doing phenomenal work in engineering design and innovation, and for allowing me to be a small part of their creative and technical endeavors. My special thanks go to the iBOT club leads, Mr. Satya Sai and Mr. Sai Krishna, whose depth and breadth of knowledge in robotics, along with the contagious commitment of the entire iBOT team toward problem-solving, have been particularly inspiring.

I am immensely thankful to Prof. Sai Jagan Mohan, Prof. Amol Marathe, Dr. Sandeep Dhar, Prof. P. Srinivasan, and my senior Mr. Anupam Purwar from my undergraduate institution, BITS Pilani, who were instrumental in sowing the early seeds of research in me.

I am grateful to all those who stood by me during the highs and lows of this long journey. Their companionship, humour, and empathy made the most demanding days bearable. My heartfelt thanks go to my labmates, seniors, and friends — Dr. Chandan Bose, Dr. MS Aswathy, Dr. Vishnu, Dr. Dipanjan Majumdar, Dr. Chhote Lal Shah, Varun H. S., Dr. Dhrubajyoti Biswas, and Dr. Rajanya Chatterjee — for intellectually stimulating discussions, constant support, camaraderie, and experiential wisdom throughout this journey. I would like to especially thank Dr. Dipanjan Majumdar for his mentorship, which helped me navigate challenges and refine my research approach.

I am indebted to all the staff members — including the cleaning and hygiene maintenance staff, security personnel, hostel and mess staff, civil and electrical maintenance teams, and the departmental and administrative office staff — for providing a smooth, reliable, and supportive environment conducive to academic work.

Above all, I owe everything to my family. Words are insufficient to express my gratitude to my parents, Mrs. Chandrikha Sundar and Mr. Sundar Kasturirangan, whose sacrifices, unconditional love, and unwavering faith in my abilities have been the foundation on which this entire journey rests. I thank my younger sibling, Harini Sundar, for her constant motivation, perspective, and moral support.

To everyone who has contributed in ways big or small — thank you.

ABSTRACT

KEYWORDS Unsteady flows; Flapping wings; Immersed boundary method; Physics informed neural networks; Deep learning; Sequential learning.

Unsteady flows past rigid or flexible moving bodies, such as flapping wings, are characterized by complex, nonlinear interactions spanning multiple spatial and temporal scales. These intricate flow features can be accurately resolved using high-fidelity computational fluid dynamics (CFD) techniques such as the Immersed Boundary Method (IBM), which enable the simulation of moving bodies on a fixed Eulerian grid. By doing so, IBM eliminates the need for mesh regeneration at every time step, significantly reducing computational overhead compared to Arbitrary Lagrangian–Eulerian (ALE) formulations. Nevertheless, IBM-based simulations remain memory-intensive and computationally expensive, particularly for long-time integrations or large-scale parametric studies. Whether through experiments or numerical simulations, accurately characterizing the underlying flow dynamics often requires resolving the unsteady flow field over extended time horizons. This demand for long-duration, high-fidelity data can impose severe memory and storage constraints, thereby limiting achievable spatiotemporal resolution. Moreover, even when the solid-body kinematics are periodic, the flow field itself may exhibit aperiodic / quasi-periodic behavior due to the inherent nonlinearities of the Navier–Stokes equations, resulting in rich and broadband temporal spectra. In many practical scenarios, such constraints lead to sparse or coarse spatio-temporal flow-field data, or cases where only a subset of flow variables is stored. Rerunning the corresponding simulations or experiments to reconstruct the missing or hidden flow variables can be prohibitively expensive. Furthermore, in cases involving moving bodies within the computational domain where a rich temporal spectra due to aperiodicity / quasi-periodicity is observed, inadequate temporal resolution can degrade the reconstruction accuracy of both observed and unobserved flow-field quantities. Consequently, there is

a strong need for surrogate modeling frameworks capable of efficiently handling these temporal and spatial complexities in unsteady flows past moving bodies.

Classically, researchers have attempted to address moving-boundary problems through rigid body or conformal transformations that map the computational domain to a body-attached reference frame. However, such formulations become cumbersome when extended to multiple moving bodies or flexible, deforming structures. In contrast, frameworks that model moving-body interactions within a fixed Eulerian reference frame—without domain remeshing—offer greater flexibility and generality. At the same time, these frameworks must remain data-efficient, computationally lightweight for querying, and capable of addressing inverse problems such as hidden-variable recovery. In recent years, Physics-Informed Neural Networks (PINNs) have shown remarkable promise in solving a wide range of forward and inverse problems in computational mechanics. Yet, their potential remains underexplored for unsteady flows involving moving boundaries. Given the advantages of working in a fixed Eulerian frame—akin to the Immersed Boundary Method—and the ability of PINNs to operate over arbitrary geometries, they present a powerful foundation for building surrogate modeling frameworks.

This dissertation introduces an immersed-boundary-aware (IBA) surrogate modeling framework based on PINNs for unsteady flows past moving bodies. The framework emphasizes two primary challenges: (i) hidden-variable recovery from sparse data, and (ii) modeling under temporal domain complexities in unsteady flow-field measurements. The methodology is demonstrated on the canonical problem of flow past a plunging airfoil, with training datasets generated using an in-house, GPU-parallelized, discrete-forcing IBM-based unsteady flow solver implemented in C++. Pressure recovery from velocity-field data is considered as the primary representative inverse problem. Two PINN formulations are investigated within the IBA framework: one based directly on the incompressible Navier–Stokes equations and another derived from the Immersed Boundary Method (IBM) formulation. Their relative performance is systematically

evaluated and compared using a novel multi-part physics loss weighting strategy to determine suitable scenarios where these formulations will be applicable. To improve data efficiency while preserving accuracy, a physics-based undersampling technique is introduced.The potential of the framework is further explored to address hidden-boundary estimation, super-resolution reconstruction from coarse or sparse velocity data, and a forward problem prediction. In addition, sequential learning strategies are developed to tackle temporal complexities in unsteady flow datasets, including temporal sparsity, long time horizons, and rich temporal spectra associated with aperiodic dynamics. A preferential spatio-temporal sampling approach combined with physics-based undersampling is proposed to enhance near-field pressure recovery and overall model performance under limited data. Together, these contributions establish a unified, data-efficient, and physics-informed surrogate modeling paradigm for inverse and forward problems in unsteady fluid–structure interaction systems, with strong potential for both numerical and experimental applications.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

BC	Boundary Condition of PDE domain.
CFD	Computational Fluid Dynamics.
DL	Deep Learning.
GPU	Graphics Processing Unit for accelerated computing.
Gradient Imbalance	A condition in PINN training where loss gradients vary significantly across spatial subdomains, leading to inefficient learning in certain regions..
Gradients	Derivatives used for backpropagation.
Hidden Variable Recovery	The process of inferring unmeasured physical quantities, such as pressure fields, from sparse or indirect data using physics-informed models..
HPC	High-Performance Computing infrastructure.
IBM	Immersed Boundary Method for complex geometries.
IC	Initial Condition of PDE domain.
Immersed Boundary Aware Framework	A unified mesh-agnostic surrogate modeling framework based on PINNs that can handle both fluid and solid domains for unsteady flows past moving bodies in an inertial frame of reference..
Leading Edge Vortex (LEV)	A coherent vortex structure formed near the leading edge of a flapping or pitching airfoil, contributing significantly to lift and unsteady aerodynamic forces..
Loss Function	Objective function minimized during

	training.
Loss Relaxation	A training strategy that adaptively relaxes the contribution of physics or boundary condition losses to facilitate convergence under limited training budgets..
MB-IBM-PINN	An immersed boundary aware extension of MB-PINN that operates in both fluid and solid regions, allowing for implicit handling of moving boundaries without explicit geometric tracking..
MB-PINN	A PINN formulation designed to handle unsteady flows with prescribed moving boundaries, operating only in the fluid domain..
ML	Machine Learning.
MLP	Multilayer Perceptron.
MSE	Mean Squared Error metric.
NS	Navier–Stokes equations governing fluid flow.
Operator Learning	A neural approach that learns mappings between functional inputs and outputs, enabling generalization across multiple parametric or boundary condition instances..
PDE	Partial Differential Equation.
PINN	A class of neural networks that incorporate governing physical laws, typically PDEs into their loss function..
Preferential Spatio-Temporal Sampling	A sampling strategy that prioritizes data points in regions or times of high dynamical importance, improving accuracy for unsteady or quasi-periodic flows..
Pressure Recovery	Estimation of pressure fields from sparse velocity data using physics-informed or data-driven surrogate models..

Quasi-Periodic Flow	Flow exhibiting multiple interacting frequencies or modes, leading to non-repeating but structured temporal dynamics..
R^2	Coefficient of determination, accuracy metric.
Re	Reynolds Number, ratio of inertial to viscous forces.
Reduced Order Model (ROM)	A simplified model that captures the essential dynamics of a high-dimensional physical system using a lower-dimensional basis or learned representation..
Residual	Difference between predicted and true PDE values.
Sequential Learning	A training strategy where a model is incrementally trained over temporally or spatially decomposed subdomains to manage long-time dynamics and reduce error accumulation..
Surrogate Modeling	An approach for approximating high-fidelity numerical simulations using computationally efficient models such as neural networks or reduced-order representations..
Time Domain Decomposition	A technique for dividing a long temporal domain into smaller subdomains, each trained with a dedicated neural network, optionally using transfer learning to enhance continuity and stability..
VCS	Vorticity Cutoff Sampling, adaptive sampling near vortical regions.
Vorticity Cutoff Sampling	A physics-based selective data sampling technique where training points are preferentially chosen based on local vorticity magnitude to mitigate vanishing gradients and improve data efficiency..
Zonal Gradient Analysis	A diagnostic approach to quantify

imbalances in gradient magnitudes across spatial zones—such as wake, body, and far-field regions—to guide loss weighting strategies..

ABBREVIATIONS

AI	Artificial Intelligence.
DL	Deep Learning.
DNS	Direct Numerical Simulation.
FSI	Fluid-Structure Interaction.
GPU	Graphics Processing Unit.
IBA	Immersed Boundary Aware Framework.
IBM	Immersed Boundary Method.
LEV	Leading Edge Vortex.
MB-IBM-PINN	Moving Boundary enabled Immersed Boundary Method based Physics-Informed Neural Network.
MB-PINN	Moving Boundary enabled Physics-Informed Neural Network.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
MSE	Mean Squared Error.
PINN	Physics-Informed Neural Network.
PIV	Particle Image Velocimetry.
RMS	Root Mean Square.
RMSE	Root Mean Square Error.
RNN	Recurrent Neural Network.
ROM	Reduced Order Model.
TEV	Trailing Edge Vortex.

NOTATION

$\boldsymbol{u}$ Fluid velocity vector field

Γ_{IB} Solid boundary denoted by Lagrangian markers

Γ_{inlet} Inlet boundary for the overall domain

$\Gamma^r_{left}, \Gamma^r_{top}, \Gamma^r_{bottom}, \Gamma_{right}$ left, right, top and bottom boundaries of the truncated spatial domain.

Γ_{outlet} Inlet boundary for the overall domain

Γ_{wall} Wall boundary for the overall domain

λ_{BC} Loss weight for Dirichlet boundary condition loss components

λ_{Bulk} Loss weight for bulk data loss component

λ_{fluid} Loss weight for physics informed loss computed in the fluid region

λ_{IBvar} weight for immersed boundary auxillary variable (IBVar) loss component

λ_{IB} Loss weight for no-slip boundary condition loss component computed on the lagrangian markers

λ_{IC} Loss weight for initial condition loss

λ_{solid} weight for physics informed loss computed in the solid region

$\mathcal{L}$ Loss function

$\mathcal{L}_{Bulk}$ Bulk data loss function

$\mathcal{L}_{IB}$ No slip boundary condition loss function

$\mathcal{L}_{Phy}$ Physics loss function

Ω Spatial domain

Ω^r	Truncated spatial domain
Ω_f	Spatial domain excluding the moving solid region
Ω_f^r	Truncated spatial domain excluding the moving solid region
$\Omega_s(t)$	Moving solid region in the domain
ω_z	Vorticity
ϕ	Activation function
π	ratio of the circumference of a circle to its diameter
c	Chord length of the airfoil
C_D	Aerodynamic drag coefficient
C_L	Aerodynamic lift coefficient
h	Plunging amplitude
k	Reduced plunging frequency
p	Pressure
Re	Reynolds number
T	Plunging time period
t	Time
U_∞	Free stream velocity
X	Data matrix

CHAPTER 1

INTRODUCTION

Unsteady flow generally manifests in different forms: (i) vortical structures in the wake of a stationary object (Derakhshandeh and Alam, 2019), (ii) impinging flow disturbances on an object (Sørensen, 2011), and (iii) resultant flow field around moving bodies immersed in a fluid medium (Shelley and Zhang, 2011). In the present study, we are interested in the third kind. Such flows are often characterized by the flow field having a strong spatio-temporal dependence. Often, this spatio-temporal dependence manifests through vortices surrounding and trailing the structures (Lewin and Haj-Hariri, 2003; Derakhshandeh and Alam, 2019; Khalid *et al.*, 2018; Lai and Platzer, 1999; Shelley and Zhang, 2011; Young and Lai, 2007). These vortices stretch, rotate, merge, attach, and detach, with each of these mechanisms having varying effects on the loads acting on the structures (Bose and Sarkar, 2018; Majumdar *et al.*, 2020). These mechanisms are also in return affected by displacement or deformation of the structures (Shelley and Zhang, 2011).

Interactions between fluid and flexible structure either submerged or surrounding the fluid are found in many biological and engineering systems. These systems come in different sizes and their study is undeniably intriguing owing to the rich physics they exhibit (Paidoussis, 1998; Dowell and Hall, 2001; Païdoussis *et al.*, 2010). The importance of studying such fluid structure interaction (FSI) systems have only grown over the years owing to the need for efficient, effective, biomimetic practical realizations (Paidoussis, 1998; Bisplinghoff *et al.*, 1996; Heil and Hazel, 2011; Païdoussis *et al.*, 2010; Shyy *et al.*, 2013; Singh *et al.*, 2012; Xiao and Zhu, 2014).

In order to understand fundamental barriers to optimal design of mechanical structures

in flow, in-depth computational or experimental investigations are thus necessary. Experiments and high-fidelity numerical simulations are gateways to understanding the underlying aero (hydro)-dynamic principles that lead to effective load generation in natural flyers and swimmers. However, to build efficient aerial or underwater vehicles, one must not just understand the mechanisms but also harness them. This requires exploration of a vast multi-dimensional design space for determining the optimal system parameters. Once designed, these systems need to actively execute maneuvers in difficult urban/forest environments and adapt to sudden impinging disturbances in the flow requiring performant real time control laws. However, to achieve the above, experiments or high-fidelity simulations are often extremely expensive and are not suitable for real-time query applications like active control and design optimization. Both experimental and numerical approaches demand that the unsteady flow field be resolved over long time horizons to accurately capture the underlying dynamics. This requirement often leads to significant memory constraints, limiting the attainable spatiotemporal resolution of the data. Furthermore, even under periodic solid-body kinematics, the flow field may exhibit aperiodic or quasi-periodic behavior due to the nonlinear nature of the Navier–Stokes equations, resulting in broadband temporal spectra (Lewin and Haj-Hariri, 2003; Khalid *et al.*, 2015; Bose and Sarkar). In practical scenarios, only sparse or coarse spatio-temporal data may be available, and often only a subset of flow variables may be stored—leaving critical hidden flow-field variables unrecorded. Rerunning these experiments or simulations to recover such missing variables can be prohibitively expensive. Moreover, in cases involving temporal sparsity and moving bodies within the computational domain, insufficient temporal resolution could further degrades the reconstruction accuracy of both observed and hidden flow-field quantities. These limitations highlight the pressing need for data-efficient surrogate modeling frameworks that not only accelerate flow-field prediction but also enable hidden-physics recovery, sparse data reconstruction, and robust handling of temporal-domain complexities. Such frameworks can bridge the gap between physical fidelity and

computational tractability—paving the way for the design and control of next-generation bio-inspired vehicles.

In recent years, deep learning (Bishop, 2006; Goodfellow *et al.*, 2016) has demonstrated remarkable success across a wide range of challenging applications, including computer vision (Krizhevsky *et al.*, 2017), time series analysis (Ismail Fawaz *et al.*, 2020), natural language processing (Vaswani *et al.*, 2023), and geospatial analytics (Bodnar *et al.*, 2024). Motivated by these advances, deep learning has increasingly attracted attention for solving forward and inverse problems in unsteady aero- and hydrodynamics. Nevertheless, applying machine learning to unsteady flows past moving bodies remains highly non-trivial due to the intricate nonlinear interactions and multi-scale dynamics inherent in such systems. This study therefore focuses on physics-informed deep learning approaches for surrogate modeling of unsteady flows past moving bodies, emphasizing the recovery of hidden flow-field variables, reconstruction from sparse data, and robust handling of temporal-domain complexities.

1.1 UNSTEADY FLOWS PAST MOVING BODIES AND THEIR APPLICATIONS

Mighty birds soaring high in the skies, insects buzzing around flowers, and fish diving deep into the oceans, are all marvelous beings capable of performing complex aerial or underwater locomotion. Moreover, these organisms execute complex maneuvers with minimal energy expenditure (Chin and Lentink, 2016). Hence, the undying intrigue of these organisms motivated humans to catch up with these natural flyers and swimmers. Over the recent decades, significant efforts have been dedicated by scientists and engineers to design autonomous micro-aerial vehicles (MAVs) and underwater Vehicles (AUVs) that are capable of harnessing aero/hydrodynamic principles to fly or swim. In both the cases, flapping mechanism is used to generate lift and thrust for their efficient locomotion. Flapping wing micro aerial vehicles and autonomous underwater vehicles have received

significant attention owing to their multi-fold applications in surveillance, environmental monitoring, and security (Shelley and Zhang, 2011). These devices are often expected to be small in size and operate at slow flight speeds which is realized only at a low Reynolds (Re) number flow regime. In this low Reynolds number regime, both fixed wing and rotary wing flying models for MAVs are inefficient due to poor aerodynamic performance and extensive noise generation, respectively. Both are impediments to effective design. This is where bio-mimicry is promising, because insects with their flapping wings can execute complex maneuvers in tight spaces while operating at low Re (Chin and Lentink, 2016). However, the aerodynamic principles governing flapping wing flight are complex in nature due to the non-linear interactions between the fluid medium and the flapping wing (Lai and Platzer, 1999; Chin and Lentink, 2016; Bose *et al.*, 2019). Hence, inspired by natural flyers, in recent years, significant research efforts have been dedicated to understand the unsteady flow around flapping wings in the low Re regime, and allow the effective flight devices of flapping wings (Platzer *et al.*, 2008; Lewin and Haj-Hariri, 2003; Khalid *et al.*, 2015; Bose and Sarkar, 2018; Bose *et al.*, 2017, 2019; Majumdar *et al.*, 2020, 2022; Shah *et al.*, 2024*a*,*b*; Xie *et al.*, 2023).

Often the flapping wing system is modeled simplistically as an oscillating foil suspended in a free stream. The interactions between the oscillating foil and the surrounding fluid medium can lead to a diverse set of wake patterns. Earlier works on flow past oscillating foils such as Platzer *et al.* (2008); Gursul and Cleaver (2019); Wu *et al.* (2020) report a variety of wake patterns and its consequence on aerodynamic load generation as an effect of varying kinematic parameters such as the oscillation frequency and amplitude. The topology of this diverse wake patterns can help characterize and classify the flow-field behavior (Majumdar *et al.*, 2020; Bose and Sarkar, 2018). Over the recent years, many works have employed tools from nonlinear time series analysis and dynamical systems theory to characterize the dynamic signatures of the flow. In fact Bose *et al.* (2019); Majumdar *et al.* (2020); Shah *et al.* (2024*a*) investigated the routes to chaos for rigid and flexible flapping wing systems. Specific attention was drawn to quasi-periodic

scenarios where the flow-field exhibits a seemingly regular or periodic behavior but on careful analysis reveals the present of additional incommensurate frequencies leading to the phase space trajectories filling the phase-space. Moreover, quasi-periodic flows demonstrated aerodynamic load enhancing behavior. Such quasi-periodicity can be expected from natural flyers/ swimmers because exact periodicity cannot be maintained by such natural systems due to the constant stimuli-response adaptivity required from these organisms.

Numerous ways exist to understand the physics of such unsteady flow: (1) experimental investigations using wind/water tunnels (Koochesfahani, 1989; Platzer *et al.*, 2008), (2) analytical or phenomenological models (Facchinetti *et al.*, 2004), and (3) numerical approximations (Kim and Choi, 2019). Over the last two decades, the unsteady aerodynamics community has dedicated significant efforts to understand the various load generating mechanisms, and vortex dynamics through physical experiments and numerical simulations across different flow-scales. A comprehensive review of aerodynamics of flapping wing systems are discussed in the works of Ansari *et al.* (2006) and Xie *et al.* (2023).

While the present dissertation primarily focuses on numerical simulations, the role and potential of data-driven and physics-informed learning methods are highlighted in the broader context of both experimental and computational studies involving flows past moving bodies. For the investigations reported here, the flow-field data are obtained from high-fidelity CFD simulations using our in-house, GPU-parallelized code.

The next section discusses the evolution of numerical modeling techniques for unsteady flows past moving bodies, setting the stage for introducing the motivation and applications of machine learning in this domain.

1.2 EVOLUTION OF NUMERICAL MODELING OF UNSTEADY FLOWS PAST MOVING BODIES

The numerical modeling of unsteady flows past moving bodies has undergone substantial evolution over the decades, transforming from simplified analytical approaches to sophisticated computational methods that leverage modern hardware capabilities. This development has allowed researchers to better understand complex fluid-structure interactions that are critical in numerous applications ranging from aircraft design to biomedical engineering.

1.2.1 Historical context and early developments

The quest to understand unsteady flows past moving bodies originated with the human ambition to achieve flight and mimic the aerodynamics of bird flight. As attempts at mechanized human flight progressed, engineers encountered catastrophic structural failures due to flow-induced vibrations of aircraft wings. This challenge gave birth to aeroelasticity (Bisplinghoff *et al.*, 1996; Fung, 2008), which integrated aerodynamics and structural dynamics to model the complex non-linear interactions between fluid and structure. In these early days, experimental investigations using wind and water tunnels served as primary tools for studying unsteady flows, while mathematical models provided an elegant means to formalize the understanding gained from experiments. However, while the structural dynamics equations were relatively straightforward to derive from first principles, fluid forces governed by the more complex nonlinear Navier-Stokes equations, defied analytical solutions in most practical scenarios.

1.2.2 Early analytical models

Due to computational limitations of the era, researchers developed simplified mathematical models that captured essential flow physics while remaining mathematically tractable. These included steady, quasi-steady, and unsteady analytical aerodynamic models for pitch-plunge motion. Researchers such as Wagner (1925) and Theodorsen (1935) developed analytical models for lift and moment that captured the delay and phase effects

inherent in unsteady lift generation on oscillating airfoils. For example, the Wagner function model describes the time-delayed lift response of an airfoil to an abrupt change in angle of attack, and Theodorsen's theory provides a frequency-domain representation for oscillating airfoils by accounting for circulatory effects. These models—although based on simplifying assumptions such as inviscid flow and small-amplitude oscillations—offered significant insight into the dynamics of flight and the onset of aeroelastic issues that led to catastrophic wing failures. Many other authors have further expanded on the thin arifoil theory of Theodorsen (1935) to estimate thrust (Garrick, 1937; Lighthill, 1970). Also, in recent times, adapting from the Wagner theory (Wagner, 1925), and the lifting line theory (Jones, 1939), Boutet and Dimitriadis (2018) modeled the unsteady lift and moment for 3D thin wings undergoing pitch and plunge motions. Directed towards estimating aerodynamic loads for insect flight, many works discuss the quasi-steady models (Sane and Dickinson, 2002; Fung, 2008) which assume that the forces can be predicted solely from the instantaneous angle of attack of the foil while totally ignoring the past history of the wake. In a recent effort, Wang *et al.* (2020*a*) proposed a quasi steady aerodynamic model taking inspiration from the blade element method (BEM) considering the aerodynamic mechanisms of circulation,dissipation, added mass and inertial effects. It is worth noting that these methods can only estimate the aerodynamic loads and not simulate the flow-field. Hence, it is not possible to possible to determine the vortex interactions leading to load generation, both in the near field and in the wake. Moreover, in time, materials became lighter and structures more flexible, resulting in larger oscillations due to fluid forces, and as interest grew in mimicking natural flyers and swimmers, these models proved inadequate for capturing the complete flow physics. This was because the analytical or quasi-steady models were incapable of modeling viscous effects and were valid only for thin airfoils / flat plates and small amplitude oscillations.

1.2.3 Vortex-based methods

As the demands for more accurate and robust predictions of unsteady flow phenomena increased, especially in the context of large-amplitude motions and emerging interests in

biologically inspired flight, researchers transitioned to vortex-based methods addressing the limitations of earlier models. Specifically, this included vortex circulation-based approaches such as the unsteady vortex lattice methods (UVLM) (Katz and Plotkin, 2001). UVLM improved on the classical analytical approaches by modeling the formation and evolution of vortex wakes behind moving bodies and their respective effect on the solid body forces. These methods could capture the shedding and interaction of discrete vortices more realistically than small perturbation models - even though they often retained the assumption of inviscid flow. Seminal works such as Hess and Smith (1967); Katz and Weihs (1978) in the 1976s and 1970s laid the groundwork for later numerical implementations that integrated vortex shedding physics into reduced-order frameworks. Although these methods improved the predictions, they still maintained the inviscid flow assumption potentially limiting their accuracy in low Reynolds number regime flows where viscous effects dominate. Moreover, these methods were incapable of resolving the leading edge vortices which were crucial contributors to load generation in flapping wings. Although some works such as Ramesh *et al.* (2014) proposed artificial modeling of nonlinear viscous separations to capture the LEV growth behavior in terms of a leading-edge suction parameter (LESP), they are inadequate to capture the intricate the complex interactions between different vortex structures such as LEV/ TEV in the flow-field (Lewin and Haj-Hariri, 2003; Bose and Sarkar, 2018; Chin and Lentink, 2016). Hence, there arose the need for high-fidelity simulations of flow-field dynamics which could resolve not just the wake effects, but also account for viscous effects, the boundary layer and the leading edge/ trailing edge vortex formation and their complex dynamical interactions.

1.2.4 Computational fluid dynamics for high-fidelity simulations

With advancements in high-performance computing, high-fidelity computational fluid dynamics (CFD) simulations have become essential tools for understanding complex flow phenomena that are difficult to measure experimentally. Typically, in physical experiments, load cells in physical experiments using wind tunnels, one can measure

the aggregate forces on a body (Platzer *et al.*, 2008). On the other hand, numerical simulations can go further and reveal the underlying causes of these forces by providing detailed velocity and pressure fields throughout the domain.

Despite advancements, high-fidelity simulations remain computationally intensive for unsteady flows past moving bodies due to: the dynamic nature of the solid boundaries, iterative linear algebra routines in Navier-Stokes solvers, mesh quality concerns due to body movement, and memory demands for parametric exploration (Álvarez-Farré *et al.* (2021); Mader *et al.* (2020); Reguly and Mudalige (2020)).

Earlier CFD methods employed either purely Lagrangian or Eulerian approaches for solving the N-S equations. Lagrangian methods such as Discrete vortex methods which allowed following of the solid interfaces accurately, are often limited to two dimensional flow and focused on simulating only the vortex elements and not the entire fluid flow (Ramesh *et al.*, 2014). Eulerian methods on the other hand could resolve this issue but required that the boundary conditions on the moving boundaries be handled carefully. Hence two primary spatial discretization strategies for unsteady flows past moving bodies arose: the arbitrary lagrange eulerian (ALE) and the immersed/embedded boundary method (IBM) based approaches. The ALE methods (Hirt *et al.*, 1974; Sarrate *et al.*, 2001; Wick, 2013) involve body-conformal meshes that must be updated or regenerated as the solid body moves through the fluid domain. The body conformal mesh is designed to strictly follows the moving body. This deformation of the mesh is informed by solving a modified set of N-S equations, where the convective acceleration term in the momentum conservation equation now depends on the relative velocity between the fluid and the moving solid mesh. Although effective, this approach necessitates computationally expensive remeshing operations, and stringent mesh quality concerns as deformations become extreme. Moreover, when there are multiple moving bodies like, say, a school of fish, a cascade of vibrating airfoils, or an array of freely vibrating cylinders, ALE becomes inadequate in handling multiple moving body frequencies or

very large deformations.

Hence, it is desirable to employ body-non-conformal mesh-based approaches as such methods would allow the free movement of single or multiple moving or deforming bodies in the fluid domain. Motivated by such an idea, Peskin (1972) introduced the immersed boundary method to study flow patterns around heart valves. In IBM, a fixed Eulerian grid is used to discretize the domain where the solid boundary (described by a set of Lagrangian marker points), is immersed. This allows one to use simple orthogonal/cartesian grids thereby easing the computations. However, incorporating the boundary conditions on the moving body surface is a non-trivial task. This is why an external momentum forcing term is often added to the N-S equations to account for the moving body. Over the years, different improvements have occurred in the method leading to a broad classification of IBM into continuous and discrete forcing based methods.

In the continous forcing approach as in Peskin (1972, 2002), the momentum forcing is a continuous function of space and time. However, the continuous forcing approach suffered from a diffused boundary problem due to poor mass conservation in the solid region and its vicinity. To tackle this, Peskin and Printz (1993) employed a divergence free finite-difference operators to improve the mass conservation. Noting that the extension of this approach to rigid moving bodies was limiting, a feedback forcing scheme was proposed by Goldstein *et al.* (1993) borrowing from control theory to model the moving solid body. However, this led to spurious oscillations in the flow predictions and necessitated heavy restrictions on the computational time-step size to ensure numerical stability.

To overcome these challenges of continuous forcing approach, a discrete forcing approach was adopted by researchers such as Mohd-Yusof (1997); Kim *et al.* (2001); Farhat *et al.* (2001); Mittal and Iaccarino (2005) which allowed better handling of the boundary

conditions on the moving body. In the discrete forcing approach, the forcing term is essentially not a real physical force as in the continuous forcing approach (Peskin, 1972). Whereas, it is numerically calculated from the discretized momentum equation such that the no-slip boundary condition on the moving body is satisfied. It was later explained by Fadlun *et al.* (2000) that the discrete forcing approach is indeed equivalent to setting velocity values to grid points in such a way that boundary conditions on the solid body are satisfied. However, even the discrete forcing IB methods suffered from spurious force oscillations in the computed fluid forces mainly due to spatio-temporal discontinuity at the fluid-solid interface. Kim *et al.* (2001) developed an improved the approach of Mohd-Yusof (1997) by accompanying the application of momentum forcing with the introduction of a mass source/sink in the continuity equation. This ensured a rigorous satisfaction of the mass continuity, ensured numerical stability . Here, a fractional step method was implemented for pressure-velocity coupling and the momentum forcing was implemented just inside the solid boundary and elsewhere it was fixed to zero. This help overcome spurious force oscillations problem plaguing the IBM earlier (Lee *et al.*, 2011). Ever since, many authors have actively adopted and improved Kim *et al.* (2001)'s approach such as (Kim *et al.*, 2001; Huang and Sung, 2007; Nicolaou *et al.*, 2015; Majumdar *et al.*, 2020) . Nicolaou *et al.* (2015) improved the method by using a semi-implicit discretization of the N-S equations instead of the earlier explicit schemes. In the recent work of Majumdar *et al.* (2020), a flow-solver based on the dicrete forcing IBM was developed in C++ to simulate and capture dynamical transitions in unsteady flow past a harmonically plunging rigid elliptic foil in the low Reynolds number incompressible laminar flow regime. It has been extended and applied to study the unsteady aerodynamics of rigid body flapping (Majumdar *et al.*, 2022), dipteran flight (Shah *et al.*, 2021), chord-wise flexible flapping foils (Shah *et al.*, 2024*a*) and flow-induced vibration based energy harvesting (Chatterjee *et al.*, 2024). In the present thesis, the flow solver developed by Majumdar *et al.* (2020) has been used to generate datasets for unsteady flow past a rigid plunging foil. Overall, IBM enables handling

complex geometries and multiple moving bodies more easily by alleviating the need for mesh motion, re-meshing and the associated mesh quality constraints otherwise required in ALE (see table1.1 for a comparison between IBM and ALE). As a result, IBM incurs comparatively lesser computational and memory costs.

Table 1.1: Comparison of IBM and ALE.

Criterion	IBM	ALE
Mesh Handling	Fixed grid; no remeshing	Dynamic mesh deformation/regeneration
Computational Cost	Lower (avoids mesh motion workflows)	Higher (quality control, Jacobian updates)
Geometry Complexity	Handles sharp/irregular shapes effortlessly	Requires smooth, structured meshes
Moving Boundaries	Native support via Lagrangian markers	Difficult to apply for large deformations or complex geometries

A fluid solid interaction (FSI) system is often studied in contexts involving non-linear flow and structural dynamics, control, and optimization (Shyy *et al.*, 2016). Investigating such systems requires evolving the spatio-temporal dynamics of the system over a long time domain, and frequently require exploring large parameteric spaces (Majumdar *et al.*, 2022). Such a data acquisition becomes computationally intensive, and becomes extremely costly for inverse problems such as hidden physics recovery (Pawar *et al.*, 2020), system identification (Brunton *et al.*, 2020), and model order reduction (Towne *et al.*, 2023). These challenges clearly highlight the need for efficient, fast, accurate numerical solvers, accurate and highly reliable reduced order models and cost effective methods to solve inverse problems in the FSI domain. Following the work of Majumdar *et al.* (2020), solvers were parallelized using OpenMP (Shah *et al.*, 2019), enabling shared-memory CPU acceleration. With the advent of general-purpose GPU computing, the massive parallelism of modern hardware allows significant further speedups (Yuen *et al.*, 2013; Xue and Roy; Kotsalos *et al.*, 2020). While CPU parallelization reduces

turnaround time, offloading linear algebra routines to GPUs can accelerate computations by orders of magnitude, enabling more extensive parametric sweeps and longer temporal simulations. Motivated by this, GPU parallelization, of an in-house IBM based unsteady flow solver is undertaken to accelerate data generation.

1.3 NEED FOR MOVING BOUNDARY AWARE SURROGATE MODELING

Even with GPU-accelerated solvers, high-fidelity simulations are still computationally intensive, especially for long temporal integrations, large parametric sweeps, and inverse problems such as system identification (Brunton *et al.*, 2020) or hidden-variable recovery (Raissi and Karniadakis, 2018; Pawar *et al.*, 2020). Moreover, in practical scenarios, flow-field datasets can often be sparse or incomplete, and critical variables such as pressure may remain unobserved. The reconstruction of such hidden physics, especially under temporal sparsity and complex dynamical behavior, remains a key challenge. This highlights the need for alternative, data-efficient approaches that can complement high-fidelity simulations while enabling accurate recovery of unobserved variables and efficient handling of temporal-domain complexities.

Surrogate modeling frameworks have emerged as low-cost alternatives that complement high-fidelity simulations or experiments (Bhushan *et al.*, 2020). Typically, surrogates are of two kinds: (i) phenomenological model developed using physical assumptions along with some empirical data (Facchinetti *et al.*, 2004), and (ii) data-driven reduced order models (Benner *et al.*, 2015). Whereas, traditional reduced-order modeling approaches are largely based on linear projection-based dimensionality reduction techniques such as Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD). These methods provide low-dimensional representations of high-dimensional flow fields by identifying coherent spatial modes and their temporal evolution (Lumley, 1967; Rowley, 2005). However, their reliance on complete datasets and fixed spatial domains limits their applicability in dynamically evolving domains. More importantly, these

approaches do not generalize well when only partial observables are available. As a result, they may struggle to recover hidden flow variables that are not directly measured but are essential for characterizing the dynamics. In such cases, hidden physics must be inferred indirectly or recovered from governing constraints, as demonstrated in Raissi and Karniadakis (2018); Raissi *et al.* (2020) for a stationary cylinder problem.

Furthermore, the evolving flow topology and domain geometry in moving-body problems violate the spatial coherence and orthogonality assumptions underpinning the POD basis functions/modes. Consequently, snapshot alignment, consistent mode extraction, and basis transferability across parametric regimes become non-trivial, restricting the applicability of these methods in strongly nonlinear and time-varying domains (Goza and Colonius, 2018; Wang and Shoele, 2021). To mitigate these issues, several studies have either excluded near-body regions or transformed the domain to a body-attached frame via affine or conformal mappings—for example, analyzing the flapping-foil flow-field from the wake region (Bose and Sarkar), or performing DMD in transformed coordinate systems (Menon and Mittal, 2020). While such methods simplify the problem, they remain unsuitable for unsteady flow scenarios with deforming structures or multiple moving bodies, where flow features in the near-field/body surface are also crucial. These gaps underscore the need for surrogate modelling frameworks that (i) recover hidden variables along with reconstructing sparse data, (ii) model temporal dependencies inherent in moving-body flows, and are (iii) capable of incorporating knowledge of underlying physics to ensure physical consistency.

However, constructing such surrogates becomes particularly challenging especially when using body non-conformal mesh strategies for CFD simulations such as the Immersed Boundary Method (IBM) (Mittal and Iaccarino, 2005; Kim and Choi, 2019). For incompressible unsteady flows with moving or deforming boundaries, IBM-based simulations inherently produce fictitious flow fields within the solid regions due to the use of body non-conformal meshes and modified Navier–Stokes equations that incorporate

forcing terms to satisfy no-slip boundary conditions (Kim *et al.*, 2001; Majumdar *et al.*, 2020).

With fictitious or physically meaningless data present, traditional decomposition techniques such as POD and DMD cannot be directly applied without domain transformations (e.g., mapping to a body-attached reference frame). Such transformations, are however restrictive when dealing with multiple moving bodies or flexible structures. If one attempts to use IBM data as-is, without transformation, the decomposition lacks a closed-form low-rank solution, as demonstrated by Balajewicz and Farhat (2014). The subsequent section builds on this foundation, formally posing immersed boundary method (IBM) data reconstruction as a machine learning problem, following the theoretical and computational insights of Balajewicz and Farhat (2014) and related works.

1.4 IBM DATA RECONSTRUCTION AS A MACHINE LEARNING PROBLEM

In IBM, the flow-field computations are carried out on a fixed background Eulerian grid while the body is allowed to freely move in this domain. Thus, at any time, there exist grid points in the domain within the envelope of the solid body motion that switch between fluid and solid domains. This results in fictitious flow-field values at the solid grid points at any time unlike a body conformal mesh based methods such as the Arbitrary Lagrange-Eulerian (ALE) framework (Sarrate *et al.*, 2001) where such a problem does not arise. However, IBM is preferred by the fluid-structure interaction community as it can handle complex geometries, multiple moving bodies, and largely deforming structures (Kim *et al.*, 2001; Peskin, 2002; Kim and Choi, 2019; Mittal and Iaccarino, 2005).

Obtaining compact representations of the flow-field data is crucial for developing surrogate models for hidden physics recovery (Pawar *et al.*, 2020). For flow-fields past stationary bodies, such compact representations can be directly obtained using

linear-low rank decomposition techniques such as POD, DMD and Spectral POD (Taira *et al.*, 2017, 2020). Given the fictitious data present in the solid region, one of the approaches to obtaining linear low-rank decompositions and reconstructions of IBM data involves transforming to a body-attached frame of reference using affine or conformal transformation (Wang and Shoele, 2021). Once transformed, the traditional modal analysis techniques such as POD or DMD can be applied (Menon and Mittal, 2020; Wang and Shoele, 2021) making this a case specific transformation. In a recent comparative study of POD, DMD and Spectral POD, Thirunavukkarasu *et al.* (2024) highlighted that while the domain transformation approach is suitable for rigid body motion based flows, multiple moving bodies with varying frequencies or flexible deforming bodies cannot be handled by this approach.

Although IBM provides advantages when it comes to high-fidelity simulations of unsteady flows past moving bodies, Balajewicz and Farhat (2014) showed that a low rank reconstruction and reduction of IBM-obtained flow-field data is a non-trivial task when building low-order or surrogate models. Typically, given the data matrix $\mathbf{X}$ consisting of spatio-temporal snapshots of the flow-field data, a simple linear projection based low-rank decomposition problem is defined as follows

$$\mathcal{J} := \arg\min_{UV} \|\mathbf{X} - UV\|_{L_2}^2 \tag{1.1}$$

Here, U, V are the spatial and temporal modes which can be obtained by solving a singular value decomposition problem (SVD).

However, this is not applicable for IBM generated data because there exist fictitious or zero valued flow-field in the region bounded by the solid boundary at any time instant (see Majumdar *et al.* (2020); Peskin (2002); Kim *et al.* (2001) for more details). As a result, flow-field reconstruction or dimensionality reduction becomes a weighted low-rank matrix decomposition problem.

The weighted-low rank decomposition problem in the context of IBM data can be re-framed as follows

$$\mathcal{J} := \arg\min_{UV} \|\mathbf{M} \odot (\mathbf{X} - UV)\|_{L_2}^2, \tag{1.2}$$

which in a generalized way can be written as

$$\mathcal{J} := \arg\min_{\theta} \|\mathbf{M} \odot (\mathbf{X} - \mathbf{f}(\theta, \mu))\|_{L_2}^2. \tag{1.3}$$

Here, instead of obtaining a linear approximation $\hat{\mathbf{x}} = UV$, a general approximation $\mathbf{f}(\theta, \mu)$ can be modeled by a set of learnable parameters θ which depend on the function's form and input parameters μ which can be the flow/structural parameters or even just the spatiotemporal domain. The weights $\mathbf{M}$ represent the binary valued masks at any given time instant taking the value of one in the fluid region and zero elsewhere. Such a weighted low-rank decomposition problem does not have a closed-form solution as in the form of a Singular Value Decomposition for example. This motivated Balajewicz and Farhat (2014) to solve the weighted low-rank decomposition using alternating least-squares (ALS) based approach to obtain the low-rank modes which still showed spurious values near the solid boundary. Importantly, the ALS approach required multiple iterations to converge to the reconstruction capacity of a standard POD-based reconstruction of ALE-obtained data.

Hence, while it is computationally advantageous to obtain high-fidelity simulations using IBM compared to ALE for complex cases, obtaining low-dimensional models from IBM data is relatively more challenging. As outlined above, IBM data reconstruction can be formulated as a weighted least-squares optimization problem that minimizes the reconstruction error over observed (non-fictitious) grid points. However, when solved using Alternating Least Squares (ALS), the resulting low-rank decomposition still remains a linear approximation of the underlying flow manifold. This restricts its expressivity—particularly under data-sparse or computationally

constrained conditions—where nonlinear correlations and temporal dynamics cannot be adequately captured. This strongly serves as the need to investigate machine learning-based methods for IBM-obtained data. The subsequent section will detail relevant earlier works which have attempted to implement machine learning methods for moving boundary problems. Specific focus will first be diverted towards outlining the mathematical preliminaries of deep learning, then data-driven modeling based approaches involving traditional and deep learning along with their effectiveness so far.

1.5 DEEP LEARNING PRELIMINARIES

Machine learning has proven to be effective in solving various pattern recognition problems in domains like computer vision, bio medical engineering, and time series analysis, etc (Paliouras *et al.*, 2003; Angra and Ahuja, 2017; Bishop, 2006).

Broadly, Machine learning is a set of algorithmic techniques to convert information to experiential knowledge and automate routine tasks that otherwise involve human intervention (Bishop, 2006).

Mathematically, the above definition translates to the following. Given an input data X and its corresponding output/label data Y, machine learning methods help identify a mapping $f : X \rightarrow Y$ such that $f = f(\theta, X)$ where, θ is a vector of unknown parameters of the function. The unknown parameters are to be estimated such that an objective function,

$$\mathcal{L} = \mathcal{L}(\theta)$$

is minimized. The objective function $\mathcal{L}$ is also called as cost/loss function of the machine learning algorithm (Bishop, 2006).

Process of minimising the loss and obtaining the parameters θ of the model is often called training. The design of a loss function for a given machine learning model itself is a detailed field of study, as the accuracy of the results largely depend on the Loss

function (Bishop, 2006; Goodfellow *et al.*, 2016; Haykin, 2009).

Broadly, there are three paradigms of machine learning, which are (1) Supervised learning[Ex: Classification, Regression], (2) Reinforcement learning and (3) Unsupervised learning [Ex: Clustering]

Under supervised learning, we mainly come across the following problem types:

- **Classification** : Given input data X(For example: flow field data of flow over a cylinder), obtain a mapping $f = f(W, b, X)$ such that $f \epsilon [0, 1]$ and maps to the out put data containing corresponding labels(For example, the flow regime being turbulent/laminar)

- **Regression**: Given input data X(For example: flow field data of flow over a cylinder), obtain a mapping $f = f(W, b, X)$ such that f fits to the output data Y which could in this case be the lift/drag forces.

In simpler terms, classification predicts a label, regression predicts a quantity. The latter problem of regression is of focus in this thesis.

The algorithms used to perform machine learning are often called models, and there are many models such as artificial neural networks, decision trees, genetic algorithms, support vector machines, and others (Bishop, 2006; Goodfellow *et al.*, 2016). In this study, however, we shall focus only on artificial neural networks and the associated deep learning theory.

Deep learning is essentially a subset of machine learning where, extremely complex representations/features can be learnt from data through specific stacking of layers of neurons that perform various tensor operations. Deep learning finds its applications in areas such as Computer Vision, Speech Recognition, Natural Language processing, complex dynamical systems and the scope of this field only keeps growing with time (Bishop, 2006; Goodfellow *et al.*, 2016; Haykin, 2009; Hornik *et al.*, 1989; Lagaris *et al.*, 1998; Liu *et al.*, 2017). Most of the topics discussed in this section are covered in the works of Bishop (2006) and Goodfellow *et al.* (2016).

1.5.1 Artificial neuron

Artificial neuron is a mathematical function (Haykin, 1994) conceived from the idea of a biological neuron's capability to transmit information through electrical impulses. Perceptron (see figure 1.1) is an example of an artificial neuron which can perform either binary classification or non-linear regression (Haykin, 1994; Bishop, 2006).

Artificial neural networks consist of a network of artificial neurons as connecting units and they map input data to corresponding output data through a multivariate function. There are many models of artificial neuron but the one in common use nowadays is the perceptron which is an algorithm that learns a binary classifier/ threshold function: a function that maps its input data to a single binary output value based on a threshold condition. Perceptron is the simplest feed forward neural network and multi layer perceptrons are nothing but length and breadth wise stacking of perceptrons that can fit more complex functions. With the well established capability of Deep Neural Networks to work as universal function approximators, it is thus possible to arrive at an alternative approach to solving forward and inverse problems in physical systems.

Mathematical representation of the above artificial neuron is as follows:

$$y = \phi(\mathbf{W}^T\mathbf{X} + b) \tag{1.4}$$

Where, $\mathbf{X} = \{x_i\}_{i=1}^n$ is the input data, y being the Output, $\mathbf{W} = \{w_i\}_{i=1}^n$ the weights and b, the bias. In general, the output variable y could be a vector, and thus, bias b would also be a vector. A bias vector is an additional set of weights in a neural network that requires no input. It corresponds to the output of an artificial neural network when it has zero input. Here, $\phi(.)$ is the activation function that operates on the linear combination of inputs. This function could be linear or nonlinear. More details about activation functions are presented in section 1.5.6.

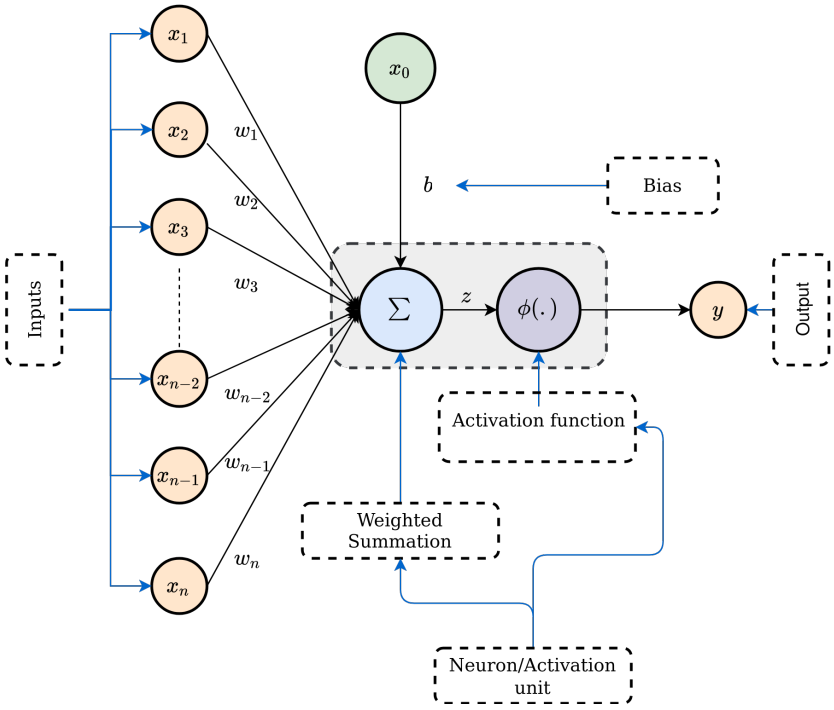


Figure 1.1: An artificial neuron, where, x_i, w_i represent the inputs and synaptic weights and Σ represents a linear combination of the inputs with coefficients as weights and $\phi(.)$ is the activation function and y is the output of the neuron

1.5.2 Artificial neural network

An artificial neural network (Bishop, 2006; Haykin, 2009) is a massively parallel distributed processor made up of artificial neurons as connecting units, which has a natural ability for storing experiential knowledge and making it available for use.

It resembles the brain in two respects:

- Knowledge is acquired by the network from its environment through a learning process.
- Inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

There are a plethora of neural network architectures/models (Liu *et al.*, 2017) that have varied applications and learning rules. In this work, as a stepping stone to understand the more complex model architectures, the Fully connected Feed Forward Neural Networks (Bishop, 2006; Goodfellow *et al.*, 2016) are discussed in in the following

section. An artificial neural network where information is passed in a unidirectional manner and that too from input to output is called a feed forward neural network. Feed back neural networks are where there exists a feedback to the preceeding neuron/layer from the succeeeding layer(s) (Haykin, 1994). In this work we currently shall discuss only fully connected feed forward neural networks (Haykin, 1994).

A three layer fully connected feed forward neural network where there is only a single hidden layer of neurons and it is also called a "shallow FNN" as it contains only one hidden layer. However, if there are $N > 2$ hidden layers of neurons between input and output layers, with same or different activation functions, then they are called deep neural networks (Goodfellow *et al.*, 2016). Deep neural networks form the strong foundation for deep learning (Goodfellow *et al.*, 2016). A very general graphical representation of a deep neural network is given in figure 1.2.

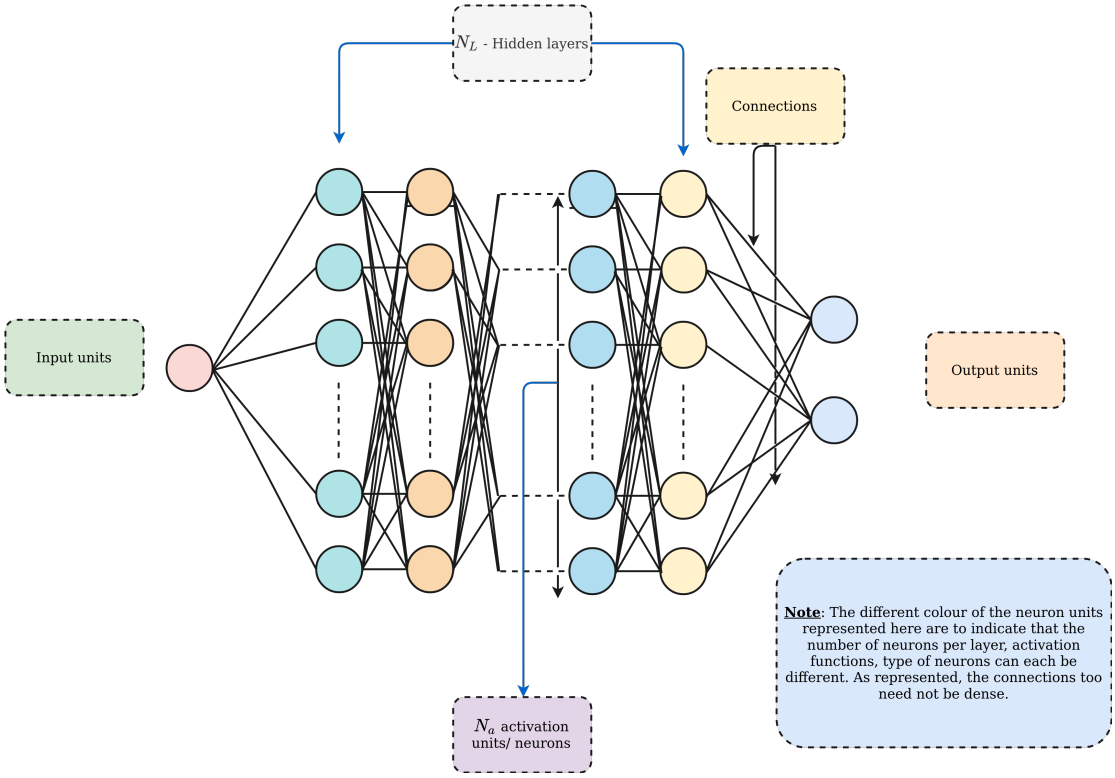


Figure 1.2: A general representation of a deep neural network

1.5.3 Fully connected feed forward neural network

Fully connected feed forward neural network (FNN) is a simple neural network architecture each layer's neurons are completely connected to its preceeding/succeeding layer of neurons. As discussed earlier, FNN can be either shallow or deep, depending on the number of hidden layers between input and output layers(see figures 1.3 and 1.4 for a graphical representation).

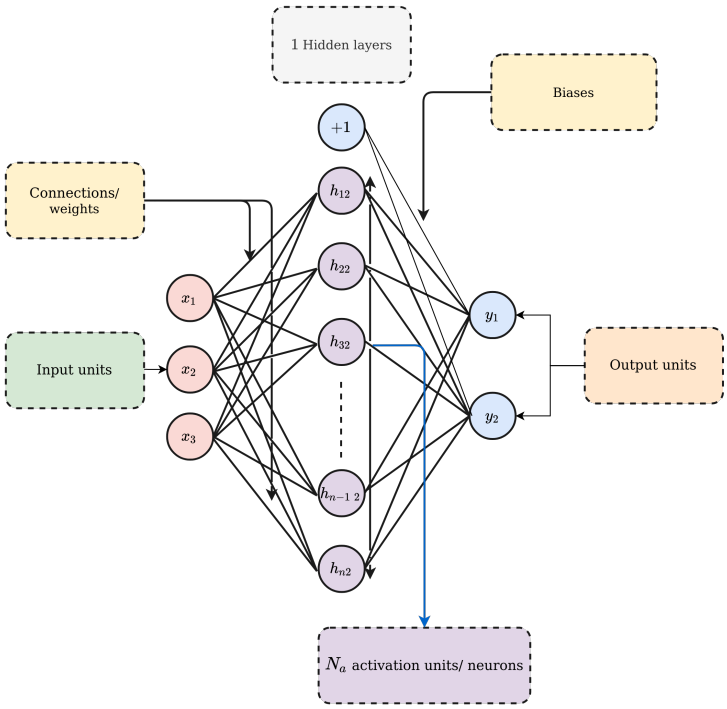


Figure 1.3: A graphical representation of a 3 layer FNN with one hidden layer

The mathematical representation of the above FNN is as follows:

$$h_0 = X \tag{1.5}$$

$$h_1 = \phi(W_x^T h_0 + b_0) \tag{1.6}$$

$$h_2 = \phi(W_{h_1}^T h_1 + b_1) \tag{1.7}$$

$$. \tag{1.8}$$

$$. \tag{1.9}$$

$$h_n = \phi(W^T_{h_{n-1}} h_{n-1} + b_{n-1}) \tag{1.10}$$

$$Y = W^T_{h_n} h_n + b_n \tag{1.11}$$

Here, X and Y are input and output values and h_i is the activated set of values at each layer. Here, W_{h_n} and b_n are the weights and biases of n^{th} hidden layer. $\phi(.)$ is the activation function which can be chosen based on the type of problem being solved (Cyr *et al.*, 2020; Jagtap *et al.*, 2020*a*; Nwankpa *et al.*, 2018).

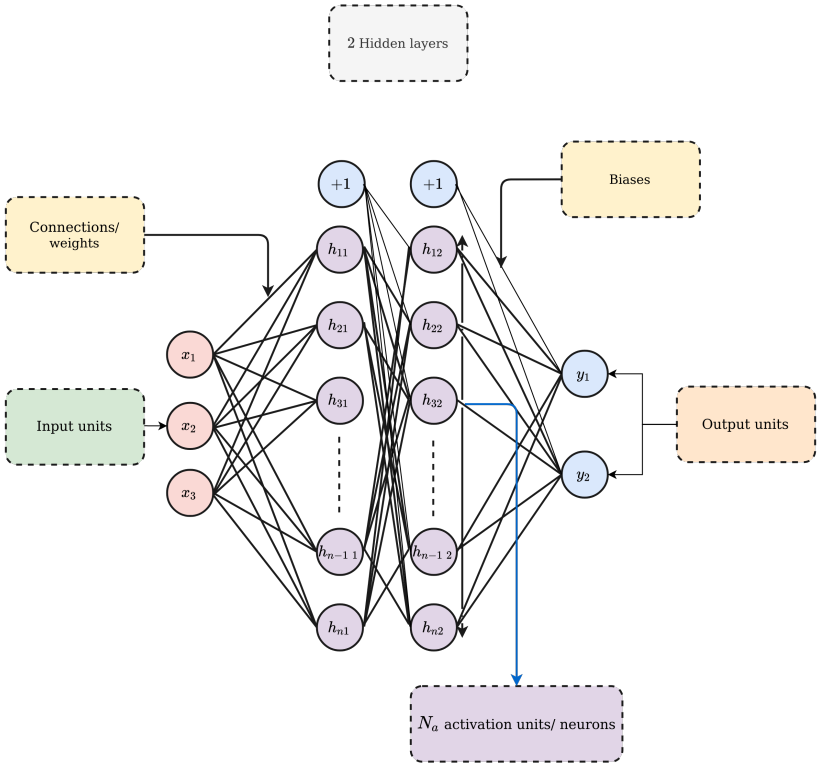


Figure 1.4: A graphical representation of a deep 4 FNN with 2 hidden layer

1.5.4 Loss function and types

Approximation of a function and its derivative can be understood as a regression problem. Hence, in this work, we discuss the basics of Machine learning in the context of regression. For regression problems there are different loss functions that one can choose from and some of the commonly used ones are as follows:

Mean Square Error/Quadratic Loss/L_2 Loss This is the most used loss in regression problems. MSE is the mean of sum of L_2 errors between the actual (y) and predicted values(y^{pred}).

$$\boldsymbol{MSE} = \frac{\sum_{i=1}^{N} \|y_i - y_i^{pred}\|_{L_2}}{N} \tag{1.12}$$

Sum of Squares Error/Quadratic Loss/L_2 Loss This is also used in regression problems. SSE is the sum of L_2 errors between the actual (y) and predicted values(y^{pred}).

$$\boldsymbol{SSE} = \sum_{i=1}^{N} \|y_i - y_i^{pred}\|_{L_2} \tag{1.13}$$

Mean Absolute Error/L_1 Loss Similar to MSE but instead of L2 errors we consider the L1/absolute errors of the actual and predicted values.

$$\boldsymbol{MAE} = \frac{\sum_{i=1}^{N} \|y_i - y_i^{pred}\|_{L_1}}{N} \tag{1.14}$$

Smooth Mean Absolute Error/ Huber Loss

$$L_\delta(y, y^{pred}) = \begin{cases} 1/2\|y - y^{pred}\|_{L_2} & \text{for } |y - y^{pred}| \leq \delta \\ \delta|y - y^{pred}| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{1.15}$$

Log Cosh Loss

$$L(y, y^{pred}) = \sum_{i=1}^{N} \log(\cosh(y_i - y_i^{pred})) \tag{1.16}$$

Each loss function has its own set of merits and demerits. However, importantly a loss function should be continuously differentiability and should be unaffected by outliers in case the data is corrupted (Bishop, 2006). Differentiability of a loss function plays a crucial role while training a neural network as it generally involves calculating the gradients with respect to the unknown parameters (see section 1.5.5).

Algorithm 1: Standard Gradient Descent: Given a loss function $\mathcal{L}((\vec{x}, \vec{y})\vec{W}, \vec{b})$, $\vec{W}$ and $\vec{b}$ are flattened weight and bias vectors respectively and $(\vec{x}, \vec{y})$ is the training data which is known apriori. The learnable parameters being $\vec{W}$ and $\vec{b}$, we can concatenate them together to form $\vec{\theta}$ which is a single vector representing all the unknowns in the loss functions belonging to the R^n space. Here, n is the total number of unknowns/learnable parameters of the neural network and also called degrees of freedom of the neural network (Bishop, 2006). A local minima in the weights space is obtained when the gradients of the function $\nabla\mathcal{L}(\vec{\theta})$, with respect to $\vec{\theta}$ become 0. Given an initial set of weights $(\vec{W}_0)$ and biases $\vec{b}_0$, Gradient descent converges to the local minima by exploring the weight space in the direction of steepest descent at a specified learning rate α.

Require: Loss function $\mathcal{L}((\vec{x}, \vec{y}), \vec{\theta})$
Require: Initialize the weights and biases $\vec{\theta} = \vec{\theta}_0$
Require: Choose a base learning rate α
Require: Choose a tolerance value for $\nabla_{\vec{\theta}=\vec{\theta}_{opt}}\mathcal{L}$
Require: Choose number of iterations N_{iter}
Require: Evaluate the expected value of the gradients, $E[\nabla_{\vec{\theta}=\vec{\theta}_0}\mathcal{L}]$ for the given training data $\{x^i, y^i\}_{i=1}^{N_{train}}$
Require: Initialize counter $i = 0$
  for $i < N_{iter}$ **do**
    if $\nabla_{\vec{\theta}=\vec{\theta}_i}\mathcal{L} \leq tol$ **then**
      $\vec{\theta}_{opt} = \vec{\theta}_i$
      break;
    else
      Simultaneous update: $\vec{\theta}_{i+1} = \vec{\theta}_i - \alpha E[\nabla_{\vec{\theta}=\vec{\theta}_i}\mathcal{L}]$
      $i++$
    end
  end
  Result: $\vec{\theta}_{opt} = \vec{\theta}_{N_{iter}}$ - Optimum weights and biases have been obtained corresponding to a local minima

1.5.5 Training a neural network using gradient based optimisation methods

As discussed earlier, the training /learning of a neural network model involves minimising a loss/cost function $\mathcal{L}$ associated with the unknown parameters(weights and biases) of the neural network. Here, training of a neural network is thus posed as an optimisation problem. Given the loss/cost/objective function $\mathcal{L}$ corresponding to the non-linear map of the neural network f to be approximated, the most popular optimisation algorithms belong to the class of gradient descent methods which require the gradients of the loss with respect to the unknowns(weights and biases). These gradients are required to be

Algorithm 2: Stochastic/Batch Gradient Descent (Bottou, 1999): Given a loss function $\mathcal{L}((\vec{x}, \vec{y})\vec{W}, \vec{b})$, as earlier in Algorithm 1, Stochastic/batch gradient descent is different from standard gradient descent that it performs gradient descent over batches of the randomly shuffled training data instead of the whole data, thus reducing the training time required.

Require: Loss function $\mathcal{L}((\vec{x}, \vec{y}), \vec{\theta})$
Require: Initialize the weights and biases $\vec{\theta} = \vec{\theta}_0$
Require: Number of training data points - N_{train}
Require: Batch size - N_{batch}
Require: Choose a base learning rate α
Require: Choose a tolerance value for $\nabla_{\vec{\theta}=\vec{\theta}_{opt}} \mathcal{L}$
Require: Number of iterations/epochs - N_{iter}
Require: Evaluate the expected value of the gradients, $\nabla_{\vec{\theta}=\vec{\theta}_0} \mathcal{L}^j$ over a randomly shuffled batch j of the given training data $\{x^i, y^i\}_{i=1}^{N_{train}}$
Require: Initialize counter $i = 0$

```
for i < N_iter do
    for j < N_train/N_batch do
        Evaluate ∇_{θ=θ_i} L^j over data points of j^th batch
        Simultaneous update: θ_{i+j+1} = θ_{i+j} − αE[∇_{θ=θ_i} L^j]
        if E[∇_{θ=θ_i} L] ≤ tol then
            θ_opt = θ_{i+j}
            break;
        end
        j++
    end
    i++
end
θ_final = θ_{N_iter+N_train/N_batch}
```

Result: $\vec{\theta}_{opt} = \vec{\theta}_{final}$ - Optimum weights and biases have been obtained corresponding to a local minima

calculated at every pass due to the iterative nature of the optimisation algorithms. The machine learning community has been using automatic differentiation (Baydin *et al.*, 2017), as it is an efficient method to calculate the gradients without having to struggle with pure symbolic or numerical differentiation but at the same time achieving machine level precision.

Standard gradient descent method

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. There have been mulitple improvements of this algorithm and hence most optimizers used in the field of Machine Learning come under the class of gradient descent methods. Given a differentiable multivariable function $f(x_1, x_2, ...x_n) : R^n \rightarrow R$, then gradient of the function f is given by:

$$\nabla_x f = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \vec{e}_i \tag{1.17}$$

Physically, gradient of a function gives the direction in which the function increases the fastest. In optimisation problems, negative gradient is used as we are interested in finding out a local minimum of the function f. The working of standard gradient descent method is mentioned in algorithm 1.

Problem with using the standard gradient descent method is that the training happens sequentially throught the data which generally proves to be time consuming.

Stochastic gradient descent

An alternative to that is Stochastic Gradient Descent where (Bottou, 1999), the gradient of the objective function is evaluated on the entire randomized data batchwise(see algorithm 2). Stochasticity here comes due to evaluation of the loss function /gradients at random subsamples/batches of the training data.

Though Stochastic Gradient descent improves the training speed, the objective/loss functions of neural networks can have other sources of noise other than the shuffling of the data such as regularisation, dropout (Hinton *et al.*, 2012).

Adaptive moment optimisation

Hence, Adaptive Moment optimisation or ADAM (Kingma and Ba, 2017) was proposed which works for stochastic objective functions. ADAM works only with first-order gradients thus having little memory requirement. The method computes individual

Algorithm 3: ADaptive Moment algorithm: Here, g_i^2 indicates the elementwise square of the gradient value g_i at i^{th} iteration. Default settings for the tested machine learning problems (Kingma and Ba, 2017) are $\alpha = 0.001$, $\beta 1 = 0.9$, $\beta 2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With $\beta 1_k$ and $\beta 2_k$ we denote $\beta 1$ and $\beta 2$ to the power k. Either a batch wise apporach(algo2) or full size approach can be adopted(algo1).

Require: Loss function $\mathcal{L}((\vec{x}, \vec{y}), \vec{\theta})$
Require: Base learning rate α
Require: $\beta 1, \beta 2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: Tolerance value for $\nabla_{\vec{\theta}=\vec{\theta}_{opt}} \mathcal{L}$
Require: Number of iterations/epochs = N_{iter}
Require: Initialize counter $i = 0$
Require: Initialize the weights and biases $\vec{\theta} = \vec{\theta}_0$
Initialize 1^{st} moment vector - $\vec{m}_0$
Initialize 2^{nd} moment vector - $\vec{v}_0$
while $E[\nabla_{\vec{\theta}=\vec{\theta}_i} \mathcal{L}] \leq tol$ **do**
  Evaluate $\nabla_{\vec{\theta}=\vec{\theta}_i} \mathcal{L}^j$ over data points of j^{th} batch
  Simultaneous update: $g_i = \alpha \nabla_{\vec{\theta}=\vec{\theta}_i} \mathcal{L}$
  $m_t \leftarrow \beta 1.m_{i-1} + (1 - \beta 1).g_i$(Update biased first moment estimate)
  $v_t \leftarrow \beta 2.v_{i-1} + (1 - \beta 2).g_i^2$(Update biased second moment estimate)
  $\hat{m}_t \leftarrow mt/(1 - \beta 1^i)$(Compute bias-corrected first moment estimate)
  $\hat{v}_t \leftarrow vt/(1 - \beta 2^i)$(Compute bias-corrected second moment estimate)
  $\vec{\theta}_{i+1} = \vec{\theta}_{i+j} - \alpha \hat{m}_t / \sqrt{(\hat{v}_t + \epsilon)}$ (Update parameters)
  $i++$
end
$\vec{\theta}_{final} = \vec{\theta}_{N_{iter}}$
Result: $\vec{\theta}_{opt} = \vec{\theta}_{final}$ - Optimum weights and biases have been obtained corresponding to a local minima

adaptive learning rates for different parameters from estimates of first and second moments of the gradients. We are interested in minimising the $E[\nabla_\theta \mathcal{L}]$ for parameters θ, and let $L(\theta_1), L(\theta_2) \cdots L(\theta_{N_iter})$ represent the realisations of the Loss function at each iteration. ADAM updates the moving averages of the gradient and square of the gradient. In Algorithm 3, $\beta 1, \beta 2 \in [0, 1)$ control the exponential decay rate of the moving averages. See Algorithm 3 for more details on implementation. Advantages of ADAM are that magnitudes of parameter updates are invariant to rescaling of the gradient, its learning rate updates are bounded, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a learning rate annealing.

Though there are other iterative optimisation algorithms of the first (ex: AdaGrad) and second order (ex:L-BFGS), in this work, to retain consistency with earlier works, we will use ADAM optimizer following the works of Raissi *et al.* (2019*b*,*a*); Wang *et al.* (2020*b*).

1.5.6 Activations functions

Activation functions are primal parts of an artificial neural network which decide which neurons need to be activated by a set of weighted combination of input data. had there been no activation function in the artificial neuron, the output would have been just a linear combination of the input data. Hence activation functions play a major role in extending the capabilities of a neuron/neural network to map more complex functions to the output data.

There are mainly two types of activation functions: (1) linear or identity and (2) non-linear. It is well established in literature that for high dimensional data, non-linear regression using neural network performs better than linear polynomial regression (Bishop, 2006; Goodfellow *et al.*, 2016; Hornik *et al.*, 1989).

Linear activation

$$\phi_{linear}(z) = cz, \text{ where, c is a constant} \tag{1.18}$$

Step function

$$\phi_{step}(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{for } z \geq 0 \end{cases} \tag{1.19}$$

Sigmoid

$$\phi_{sigmoid}(z) = \frac{1}{1 + \exp^{(} -z)} \tag{1.20}$$

Hyperbolic tangent

$$\phi_{tanh}(z) = \tanh(z), \text{ for all z} \tag{1.21}$$

Rectified linear unit (ReLU)

$$\phi_{ReLU}(z) = \begin{cases} 0 & \text{for } z < 0 \\ & \\ z & \text{for } z \geq 0 \end{cases}, \text{or,} \tag{1.22}$$

$$\phi_{ReLU}(z) = \max(0, z) \tag{1.23}$$

SiLU/ Swish

$$\phi_{swish}(z) = z\left(\frac{1}{1 + \exp^{-z}}\right) \tag{1.24}$$

Here z is the linear combination of values from the input units/preceding hidden layer.

The above mentioned activation functions are the popular in use, but from the works of Jagtap *et al.* (2020*a*,*b*) and Cyr *et al.* (2020), it is evident that the activation functions can be tailored suitably and can have learnable parameters which adapt as the learning progresses thus becoming appropriate for the problem being solved. With adaptive activations, there are new learnable parameters along with network weights and biases resulting in an additional computational overhead during backpropagation. While adaptive activation functions are performant, the works do not compare with networks of similar order of learnable parameters. From many earlier works such as Ramachandran *et al.* (2017); Dubey *et al.* (2022); Jagtap and Karniadakis (2023); Al Safwan *et al.* (2021), it is noticed that Swish/SiLU activation function is performant in various empirical studies as it is smooth and differentiable at zero, and at the same time has the benefit of monotonicity on the right half of the input plane. Hence, Swish/SiLU activation will be used throughout the rest of the study.

1.5.7 Initialisation of Weights

Initialisation of weights is an important step because inappropriate initialisation would lead to either exploding or vanishing of weights as training progresses (Glorot and Bengio, 2010). This would hence lead to an increased training time. Hence there have been many methods proposed to take care of such issues. One can refer to the survey article by Kumar (2017). Here we mention two novel initialisation techniques popularly used by the machine learning community, namely the Xavier initialisation (Glorot and Bengio, 2010) and Kaiming/He initialisation (He *et al.*, 2015).

Xavier/Glorot initialisation Proposed by (Glorot and Bengio, 2010), this method works for smooth functions like tanh, sigmoid, etc. In this method sets a layer's weights to values chosen from a random uniform distribution that's bounded between

$$\pm \frac{\sqrt{6}}{\sqrt{n_l + n_{l-1}}}. \tag{1.25}$$

This type of intialisation is preferred for continuous activation functions.

Kaiming He initialisation This method works for piecewise continuous activation functions like ReLU, LReLU, etc. Firstly, initialize the biases to be 0 and initialize the weights in each layer from a zero mean gaussian distribution with standard deviation of

$$\sigma_l = \sqrt{\frac{2}{n_l}}, \tag{1.26}$$

where, n_l is the number of units from the preceeding layer

Hyperparameter tuning

Hyperparameters of a neural network are those parameters of the neural network which are set before hand and are not often the learnable parameters of the model. Number of hidden layers, neurons per hidden layer, activation functions, learning rate, number of

epochs, penalties if introduced for any loss function regularisation are all hyper parameters that are to be set before training of the neural network is commenced. Choosing these hyperparameters optimally for a particular is an iterative process though some discretion can be exercised in narrowing down the range of variation (Cyr *et al.*, 2020; Bishop, 2006; Goodfellow *et al.*, 2016) also known as hyperparameter tuning. Now that the key concepts of deep learning are covered, the data-driven modeling techniques in the context of unsteady flows will be discussed in the next section.

1.6 DATA-DRIVEN MODELING AND DEEP LEARNING FOR UNSTEADY FLOWS PAST MOVING BODIES

Data-driven modeling has been evolving over many decades with advancements in numerical methods and compute. Some core applications of data driven modeling are: assimilating data from experimental observations and numerical simulations, solve inverse problems such as hidden physics recovery, design optimization and real time query applications such as control. For all these applications, the core premise is to build a surrogate or reduced-order models that can cost effectively account for parametric variations, and/or spatio-temporal dynamics of the underlying physical system in turn enabling real time query applications mentioned above.

1.6.1 Projection based reduced order modeling

Typically, constructing a ROM involves generating low-dimensional representations (latent space) of the spatiotemporal flow-field data. The governing equations are then projected onto these obtained latent or low-order representations. This is required because CFD simulations often have about a million $O(10^5)$ grid points resulting in a high dimensionality of the spatiotemporal domain. Hence, constructing low-order models or surrogates by extracting meaningful low-dimensional representations from such flow-field data would be beneficial for real-time query applications and aid in the design of experiments and simulations. This has been widely acknowledged in the

comprehensive reviews by Benner *et al.* (2015); Taira *et al.* (2017); Rowley and Dawson (2017); Taira *et al.* (2020) which discuss application of ROMs, and their efficacy in modeling the flow-field accurately.

To a great extent, linear projection-based modal techniques such snapshot proper orthogonal decomposition (POD) and dynamic mode decomposition (DMD) form the backbone of traditional ROMs, leveraging modal analysis to extract dominant flow features from high-fidelity simulations. Originally introduced by Lumley (1967), the POD method has seen multiple advancements, such as the introduction of snapshot-based POD by Sirovich (1987) for analysis of spatiotemporal turbulent flow-field data. Snapshot POD employs singular value decomposition to find the most energetically dominant features from the flow. Towards data-assimilation and reconstructing missing data, Gappy-POD was introduced by Everson and Sirovich (1995) which was later extended by (Venturi and Karniadakis, 2004) to reconstruct missing snapshots of flow past a cylinder. However, in a later work by Gunes *et al.* (2006), it was shown that gappy POD falls short when the temporal resolution of the available data is not enough limiting its application for spatio-temporally sparse datasets. In the reduced order modeling front, POD alone does not form a ROM and it is often coupled with a least squares Galerkin projection approach in an intrusive fashion which requires modifying the governing equations (Benner *et al.*, 2015). Moreover, such techniques could suffer from stability and convergence issues due to discarding higher order modes resulting in unresolved physics (Pawar *et al.*, 2020). While many have proposed closure methods to stabilize POD based ROMs, they are often problem specific, cumbersome due to dependence on discretization and are not generalizable (Benner *et al.*, 2015; Ahmed *et al.*, 2021).

On the other hand, dynamical mode decomposition which was introduced by Rowley *et al.* (2009), is a relatively recent technique that identifies relevant coherent spatiotemporal modes of a dynamical system at discrete frequencies. Unlike POD which only produces a latent space, DMD performs a local linear approximation of the dynamical system

making it a ROM by itself. This is particularly valuable for time-dependent unsteady flows, and hence been adopted across various domains apart from fluid dynamics (Wang and Yu) like neuroscience (Brunton *et al.*, 2016), and climate modeling (Sofi and Oseledets, 2025), among others. However, the construction of low-order representations in DMD is a non trivial task, as there is no particular ranking of among the modes like in POD. The conventional approach to rank DMD modes based on their amplitudes have been ineffective, leading to the proposal of alternative strategies Kou and Zhang (2017); Krake *et al.* (2021); Xu *et al.* (2023). However, these techniques have not been applied to unsteady flows generated around flapping wings. Overall, the POD and DMD family of methods have been applied to a variety of problems including flow-induced vibrations (Perrin *et al.*, 2007; Bose and Sarkar), turbulent flows (Hilberg *et al.*, 1994; Hijazi *et al.*, 2020), and multi-phase flows (Bao and Gildin, 2017; Xiao *et al.*, 2017).

While POD cannot directly provide insights into the dynamics of the flow, DMD cannot directly provide insights into the dominant features. Certain vortical patterns might play crucial roles in determining the dynamical state but contribute less to the flow-field's overall energy content and these patterns may be excluded in the latent space (Noack *et al.*, 2003). Hence, recently, significant attention has been directed towards the spectral proper orthogonal decomposition (SPOD) (Towne *et al.*, 2018) to tackle the challenge of extracting low-dimensional spatio-temporal modes or features of the flow which capture the essence of dynamics as in DMD while also enabling ordering of these modes akin to the energetic dominance of POD. This is achieved in SPOD in the following way: the spatio-temporal flow-field data is first transformed into the frequency domain via a Fourier transform and singular value decomposition (SVD) is applied to the now obtained spectral frequency domain data. The modes now obtained contain the spatio-temporal spectral information. Towne *et al.* (2018) showed that SPOD modes are essentially optimally averaged DMD modes obtained from an ensemble DMD problem in case of stationary flows, thereby establishing a connection between DMD and SPOD. SPOD has found diverse applications in various domains in fluid mechanics, encompassing

pipe flows (Hellström and Smits, 2017), turbulent jets (Schmidt *et al.*, 2018; Pickering *et al.*, 2020), channel flows (Muralidhar *et al.*, 2019), and flow past bluff bodies (Ghate *et al.*, 2020; Nidhan *et al.*, 2020). Similar to POD, even SPOD is not a ROM by itself as in DMD. Hence further projection of governing equations onto these modes might be necessary to build a ROM.

The aforementioned model reduction tools were often employed in the downstream wake of the moving body (Perrin *et al.*, 2007; Bose and Sarkar) or, assumed a thin solid body possessing negligible thickness Goza and Colonius (2018). As a result, the near-field vortices, the LEV in particularly is ignored from the anlaysis although evidence suggests its significant influence on the dynamics of flapping airfoils (Lewin and Haj-Hariri, 2003; Deng *et al.*, 2016; Wang *et al.*, 2020*c*; Majumdar *et al.*, 2020). Application of the above mentioned techniques becomes even more difficult when the flow-field data is generated using body non-conformal mesh based approaches like the Immersed Boundary Methods (IBM) (Peskin, 1972; Mittal and Iaccarino, 2005; Kim and Choi, 2019). The grid points within the bounds of the solid body motion are often time-varying fluid-solid regions. When these model reduction techniques are directly applied to such flow-field data, the resulting modes represent the solid body as an average of the its positions over time. In such a scenario, the reconstructed flow-field using this latent space would be erroneous with unphysical spurious structures close to the solid boundaries (Balajewicz and Farhat, 2014; Wang and Shoele, 2021; Menon and Mittal, 2020; Goza and Colonius, 2018). This requires that the domain be transferred to body attached frame of reference through conformal or affine transformations as in the work of Menon and Mittal (2020). However, such transformations are cumbersome when handling multiple moving bodies or flexible deforming structures. Moreover, studies such as Wang and Shoele (2021); Balajewicz and Farhat (2014) have pointed out the difficulty in handling moving body problems with data generated in an IBM setup. Wang and Shoele (2021) discussed how first of all data generated using IBM is not amenable to conformal or affine transformations of the domain to a body attached frame of reference simply because of using a non-conformal

background mesh. Secondly, it was also pointed out that using POD, DMD or even SPOD over such data would be equivalent to fitting a step function using fourier series resulting in artifacts like Gibbs phenomena at the discontinuity. Such artifacts were also observed by Balajewicz and Farhat (2014) where an alternative Least Squares based approach was used to obtain least squares optimal modes in the Eulerian frame of reference.

To deal with incomplete or corrupted measurements, gappy POD (Everson and Sirovich, 1995) was proposed as a reconstruction framework capable of inferring missing data using a truncated set of precomputed modes. Its extension to unsteady flow reconstruction (Venturi and Karniadakis, 2004; Willcox, 2006) allows interpolation of missing spatial or temporal information by projecting the available data onto the POD basis. However, for non-stationary problems, this reconstruction must be performed at every time step—an approach that quickly becomes computationally cumbersome. Moreover, Gappy POD fundamentally relies on the existence of a well-posed POD basis obtained from a sufficiently rich prior snapshot ensemble. When such a basis is unavailable or only partially representative (as in moving boundary flows), iterative methods have been proposed to refine the basis from incomplete data (Venturi and Karniadakis, 2004). Yet, these remain purely data-driven and lack physical regularization, making them ill-suited for scenarios with sparse measurements or rapidly evolving dynamics.

Recent advances such as gappy spectral POD (Nekkanti and Schmidt, 2023) attempt to overcome some of these limitations by leveraging spectral coherence or nonlinear latent representations to improve reconstruction fidelity. While promising, these methods still assume that the latent manifold inferred from data sufficiently encodes the underlying physical processes; a premise which becomes invalid for transient, non-periodic, or geometry-coupled flow systems. In particular, for immersed boundary formulations, where the solid–fluid interface moves dynamically within the computational domain, enforcing consistent boundary conditions across time-varying snapshots becomes highly

non-trivial. This evolving geometry alters the underlying flow manifold itself, thereby invalidating the fixed basis assumptions of projection-based ROMs.

Thus, despite their algorithmic elegance, traditional ROMs and gappy extensions remain limited in generalizability and physical consistency when applied to unsteady, nonlinear, or sparsely observed flow or FSI systems. These observations build upon the difficulties highlighted previously—where domain deformation, moving interfaces, and near-body flow complexity disrupt the very assumptions enabling low-rank approximations. When coupled with sparsity and nonlinearity, these restrictions make classical ROMs fundamentally unsuitable for reconstructing or predicting unsteady FSI dynamics. This gap underscores the need for surrogate modeling frameworks that are (i) computationally efficient, (ii) capable of incorporating physics priors, (iii) robust to sparse and noisy data, and (iv) able to capture latent temporal dependencies inherent in moving body flows.

Moreover, with traditional intrusive reduced order modeling strategies involve obtaining spatio-temporal modes from data followed by their projection onto the governing equations. While these are data-driven and physics based, the stability of the solutions are not often guaranteed. Many first principle and data-driven closure modeling strategies have been explored in this regard, a comprehensive review of which can be found in (Ahmed *et al.*, 2021). Recognizing this limitation in intrusive reduced order modeling approach such as the POD-Galerkin (Benner *et al.*, 2015), data-driven non-intrusive reduced order modeling strategies have become popular (Gao *et al.*, 2020; Xiao *et al.*, 2017). This includes the integration of POD/DMD and SPOD with other data-driven techniques, including machine learning and deep neural networks (Renganathan *et al.*, 2020; Fu *et al.*, 2023; Kim *et al.*, 2024; Eivazi *et al.*, 2020), enabling the development of non-intrusive ROMs.

1.6.2 Data driven modelling using deep learning

Recently, such machine learning (ML) methods have gained particular interest in the fluid mechanics community towards developing surrogate models as ML is effective in solving complex inverse and ill-posed problems while also suitable for real-time query applications (Brunton *et al.*, 2020; Vinuesa and Brunton, 2022*b*,*a*; Cheng *et al.*, 2023). There are two paradigms in ML based approaches, (i) data-driven (Vinuesa and Brunton, 2022*b*; Brunton *et al.*, 2021), and, (ii) physics informed (Brunton *et al.*, 2020; Willard *et al.*, 2022; Karniadakis *et al.*, 2021; Cuomo *et al.*, 2022).

In some non-intrusive reduced order modeling works (Gao *et al.*, 2020; Xiao and Zhu, 2014; Lee and Carlberg, 2020; Eivazi *et al.*, 2020; El Garroussi *et al.*, 2022; Nony *et al.*, 2023), a variant of an autoencoder neural network is often employed for obtaining low-order representations of the data. Autoencoders are a deep neural networks which trained to compress the input to a bottleneck layer or a latent space and reconstruct the input (Goodfellow *et al.*, 2016). Such autoencoders are often trained using a stochastic optimization algorithm (Kingma and Ba, 2017). Theoretically, an autoencoder can be considered a non-linear generalization of POD as shown by Baldi and Hornik (1989). (Lee and Carlberg, 2020) proposed a convolutional neural network (CNN) based autoencoder architecture for intrusive nonlinear reduction of convection dominated flows which was shown to be more effective than POD based approach. Eivazi *et al.* (2020) proposed an autoencoder-DMD based approach for predictive modeling of flow past a pitching plate and further improved it by replacing DMD with a long-term short memory network (LSTM) (Goodfellow *et al.*, 2016). However, in the case of non-intrusive reduced-order modeling which is purely data-driven, a large amount of data is often required to obtain meaningful in-distribution predictions for the test set while these models often suffer when evaluating on out-of-distribution input parameters.

Although data-driven approaches are well developed, they require large amount of data to capture the essential dynamics of complex physical systems. For example

in the work of Fukami *et al.* (2019), a CNN based super-resolution model was built with $O(10^3)$ flow-field snapshots, and it was demonstrated that the accuracy of their model only improved with increasing number of snapshots for training. Importantly, generalizability of trained models to unseen input data happens to be a concern. Moreover, purely data-driven ROMs, while powerful in pattern recognition, could possibly violate fundamental conservation laws (Eivazi *et al.*, 2020; Gao *et al.*, 2020). Moreover, these data driven deep learning based methods often become inappropriate for coarse or sparse data regime. Since large data acquisition of real engineering systems for inverse problems is often costly, a new paradigm has emerged to integrate the physical laws and machine learning (Willard *et al.*, 2022) such that learning in small data regimes and solving forward/inverse problems of physical systems become possible. By finding a way to encode the physical laws of the system into the neural network, one is able to preseve the inherent invariances and symmetries while performing machine learning tasks.

In a recent survey, Willard *et al.* (2022) presented the emerging trends in integrating physics and machine learning. By combining Machine learning and physics the following objectives are often satisfied: (1) improving predictions beyond that of state-of-the-art physical models (2) parameterization/model order reduction (Lui and Wolf, 2019) (3) forward solving partial differential equations (Avrutskiy, 2020) (4) inverse Modeling (Raissi and Karniadakis, 2018) (5) discovering Governing Equations (Raissi *et al.*, 2019*a*) and (6) multifidelity learning (Meng and Karniadakis, 2020). Willard *et al.* (2022) also point out the key areas in which active research is performed to use a combination of Physics and ML: (1) physics guided or informed Learning - the physical laws are encoded in the objective function to be minimized, (2) physics guided architecture:- the neural network architecture is designed such a way that some properties of a physical system can be imbibed, (3) residual learning : the purpose here is to reduce the errors/residuals of the physical laws (Kani and Elsheikh, 2017) . (4) Hybrid Physics - ML models : the neural network is designed so as to predict intermediate values of a physics model.

While data-driven methods are physics agnostic, physics informed machine learning methods are capable of embedding partial/complete information of the underlying physics into the objective function. The present work focuses on the physics informed machine learning approach, specifically, the physics informed neural networks (PINNs) (Raissi *et al.*, 2019*a*) which is a class of deep neural networks (Goodfellow *et al.*, 2016). More background and related work are discussed in the subsequent section in the context of unsteady flows.

1.7 PHYSICS INFORMED MACHINE LEARNING

Physics informed machine learning is a novel paradigm which traces its origins to the seminal work by Lagaris *et al.* (1998), where neural networks were applied to solve canonical boundary value problems governed by PDEs. Under coarse discretizations of the domain, Lagaris *et al.* (1998) showed that such neural networks with PDE constraints were better interpolants than traditional finite element method based approach. This is especially valuable when dealing with coarse or sparse data. Widely expanding on it, Raissi *et al.* (2019*a*) in their seminal work proposed the physics informed neural networks (PINNs) which have become extremely popular in recent times, finding its applications across various scientific domain.

Formulated as an optimization problem, PINNs embed prior knowledge of the governing physics through the objective/loss function (Raissi *et al.*, 2019*a*). In the context of fluid mechanics, PINNs embed governing equations—such as the Navier-Stokes equations—as soft constraints within neural network loss functions, enabling data-efficient solutions to forward and inverse problems (Raissi *et al.*, 2019*a*). Unlike conventional CFD methods that rely on spatial discretization (e.g., finite volume or spectral methods), PINNs parameterize solutions as continuous functions approximated by deep networks.

PINNs often employ feed forward neural networks to build pointwise space-time / space-time-parameter input based approximations of the underlying physical system. While

there exist various types of neural networks (Goodfellow *et al.*, 2016), the commonly used families of networks in data-driven modeling (Brunton *et al.*, 2020) include feed-forward neural networks (FNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). Among these, CNNs are designed to handle spatially discrete data, while RNNs handle sequential data. They can only provide a discrete approximation of the spatiotemporal data which in turn limits the size/resolution of the spatiotemporal grid one can train these networks on. Once trained on a specific spatio-temporal grid, it wouldn't be possible to predict on another grid using these networks. This is where, although a precursor to these variants, a FNN is promising as one can train on point-wise data, and handle any arbitrary distribution of the spatial grid points as well. This advantage of FNNs coupled with the encoding of the underlying flow-field physics into the network while training through the objective/cost/loss functions makes PINNs using FNNs as backbones, very promising. This approach eliminates mesh generation challenges as one could sample the input grid points from anywhere in the domain and can consider any arbitrary domain (Raissi *et al.*, 2020). This is a valuable feature of PINNs particularly for moving boundaries and complex geometries.

Standard collocation PINNs (Raissi *et al.*, 2019*a*) could be difficult to train due to various failure modes such as competing objectives (Wang *et al.*, 2021*a*), and propagation failures (Krishnapriyan *et al.*, 2021), especially in the case of forward problems. Various strategies have been proposed lately– such as, loss component balancing (Wang *et al.*, 2021*a*; Jin *et al.*, 2021), modified backbone architectures (Wang *et al.*, 2021*a*), adaptive sampling (Wu *et al.*, 2023), domain decomposition (Jagtap *et al.*, 2020*c*) and sequential learning (Krishnapriyan *et al.*, 2021; Penwarden *et al.*, 2023) – to tackle these problems. Many authors have also proposed adaptive loss component weighting (Wang *et al.*, 2021*a*; Heydari *et al.*, 2019), domain decomposition strategies (Jagtap *et al.*, 2020*c*; Dwivedi *et al.*, 2021), adaptive activation functions (Jagtap *et al.*, 2020*b*), and modified architectures (Wang *et al.*, 2021*a*), to mitigate the above failure modes. A more detailed list of scalable algorithms to train PINNs efficiently can be found in Shukla *et al.* (2022),

and Penwarden *et al.* (2023). Recently, Krishnapriyan *et al.* (2021) and Penwarden *et al.* (2023) have shown that standard PINNs are difficult to train in situations when no measurements/simulation data are available, especially for unsteady convection-dominated problems. On the other hand, Gopakumar *et al.* (2023) showed that PINNs become relatively easier to train if at least some sparse/coarse simulation data or partially observed experimental data are available. It has also been shown that when simulation data is available, the expressivity of PINNs can be improved by appropriately relaxing the physics loss components, without the need for any architectural modifications (Lucor *et al.*, 2022). In light of this, with simulation data being available in the present study, it is also of particular interest to investigate strategies that improve the predictive capability of PINNs under a fixed computational budget without the need for architectural changes.

The added physics loss component regularizes learning on the available data and also enables solving different forward/inverse problems with minimal change in the architecture (Raissi *et al.*, 2019*a*). In the context of inverse/ill-posed problems, PINNs are commonly used for, data-driven discovery of PDEs (Raissi *et al.*, 2019*a*), data assimilation (Raissi *et al.*, 2019*b*), uncertainty quantification (Cheng *et al.*, 2023), and, hidden physics discovery (Raissi *et al.*, 2020). Recent surveys have presented the diverse set of domains in which PINNs have been applied (Karniadakis *et al.*, 2021; Cuomo *et al.*, 2022). Once trained on a given spatio-temporal domain, the model allows for query at any temporal/ spatial location within the given domain. Moreover, PINNs are memory efficient; once trained, the model parameters alone are sufficient to be saved to carry out the querying. Other advantages are, they are *automatically* differentiable by obtaining a continuous function of input variables, and extremely flexible to implement using the well developed deep learning (Goodfellow *et al.*, 2016) software platforms capable of automatic differentiation (Baydin *et al.*, 2017) such as Tensorflow (Abadi, 2016), PyTorch (Paszke *et al.*, 2019) and Jax (Bradbury *et al.*, 2018).

A seminal advancement by Rao *et al.* (2020) demonstrated PINNs' capability to solve

Newtonian laminar flows over cylinders using a mixed-variable formulation. By decomposing governing equations into continuum and constitutive components while enforcing divergence-free conditions via stream functions, their method reduced derivative orders, improving training stability and prediction accuracy. This framework was extended to conjugate heat transfer problems in obstructed domains, revealing that architectural choices and equation normalization critically influence solution fidelity. Unsteady flows introduce temporal dependencies that exacerbate computational costs in traditional CFD. PINNs inherently handle spatiotemporal coherence through continuous space-time domain formulations. Recently, Sliwinski and Rigas (2023) and Patel *et al.* (2024) pioneered PINN-based mean flow reconstruction for unsteady flows, leveraging Reynolds-averaged Navier-Stokes (RANS) equations without requiring turbulence closure models. Their approach demonstrated superior performance in reconstructing transient velocity fields from sparse measurements compared to proper orthogonal decomposition methods. In an other study, Sun *et al.* (2020) developed parametric surrogate in a data-free manner with PINNs for hemodynamic flows in irregular geometries, contrasting soft versus hard boundary condition enforcement. Hard constraints, which exactly satisfy Dirichlet/Neumann conditions at input coordinates, proved essential for unique solution identification—a critical requirement for inverse problems. Adaptive activation functions (Jagtap *et al.*, 2020*a*,*b*) further enhanced convergence rates but were secondary to proper boundary treatment. It is however important to note that such a boundary condition enforcement will be non-trivial when working with moving bodies in an inertial frame of reference. PINNs excel in ill-posed problems where boundary conditions or material properties are unknown. Tartakovsky *et al.* (2020) estimated heterogeneous hydraulic conductivity in subsurface flows using PINNs constrained by Darcy's law and sparse measurements. In turbulent flow regimes, Raghu *et al.* (2024) coupled PINNs with the Boussinesq hypothesis to predict Reynolds stresses in free shear flows. Their dual-network architecture separated mean flow variables from eddy viscosity estimation, while dynamic loss weighting balanced

gradient contributions across momentum equations. Non-dimensionalizing governing equations further stabilized training, underscoring the importance of preprocessing in PINN implementations.

Traditional CFD methods like immersed boundary (IBM) or arbitrary Lagrangian-Eulerian (ALE) techniques face mesh quality and remeshing challenges for moving bodies. PINNs can circumvent these issues through meshless formulations, proving particularly advantageous for fluid-structure interaction (FSI) problems involving large deformations. While high-fidelity IBM simulations require GPU acceleration for parametric studies, PINNs can achieve reasonable accuracy at reduced computational cost once trained on the generated data even if sparse. But physics informed methods that handle moving boundaries and complex geometries in a non-conformal mesh setting are still in infancy (Wang and Perdikaris, 2021; Buhendwa *et al.*, 2021; Raissi *et al.*, 2019*b*; Calicchia *et al.*, 2023).

1.7.1 Relevance of PINNs for unsteady flows past moving bodies

Instantaneous pressure field for flow around moving bodies such as natural swimmers or flyers is important due to the complex interactions between pressure associated with the acceleration of the body and the vortices generated in the flow (Daniel, 2015). In experiments, either flow visualization data of a passive scalar such as the concentration of contrast agents/smoke/dye is available (Lai and Platzer, 1999), or, planar velocity data is available through particle image velocimetry (PIV) measurements (Dabiri *et al.*, 2014). Often, pressure field measurements are not taken. Hence, recovering instantaneous pressure non-invasively from the flow visualization/PIV velocity data would be extremely useful in understanding the underlying mechanisms and also for estimating the loads on a moving test specimen (Liu and Katz, 2013; Dabiri *et al.*, 2014; Pirnia *et al.*, 2020). In the case of numerical simulations, too, near-field pressure recovery from velocity data is desirable. In some variants of IBM (Lee *et al.*, 2011; Majumdar *et al.*, 2020), a mathematical artifice is used to enable the computation of the aero-/hydro-dynamic

loads from the total time derivative of velocity and the momentum forcing terms in the momentum conservation equation. In such cases, the pressure data are not even used or stored. Therefore, an appropriate surrogate for accurate physics-consistent near-field pressure recovery and simultaneous velocity reconstruction from the coarse/sparse velocity data alone, would be highly desirable. This also adds to the effort towards further reducing the high memory costs, often incurred in storing high-fidelity simulation data. Indeed, it would be convenient to have a surrogate to query both velocity and pressure once recovered at a specific spatio-temporal coordinate within the region of interest, with minimal effort. Hence, hidden physics recovery using PINNs in the context of moving body flows is a major research question addressed in this thesis.

Surrogate modeling based on simulation data obtained from IBM poses several challenges (Balajewicz and Farhat, 2014). Specifically, as a result of employing a fixed Eulerian background grid, the region of the grid bounded by the Lagrangian markers contains fictitious flow-field values that are necessary to the IBM computation. Thus, at any time instant, the Eulerian grid points falling within the range of the solid body motion could either lie in the fictitious flow domain (that shall be called the *solid* domain from now on) or the fluid domain. Moreover, enforcing the no-slip boundary condition on the solid boundary becomes difficult as the Lagrangian markers are disjoint from the underlying Eulerian grid points. Thus, while handling moving bodies or complex geometries becomes easier with IBM (Kim and Choi, 2019), obtaining a surrogate capable of both recovering pressure and reconstructing the velocity accurately becomes challenging.

PINNs based approaches have also been recently tested for problems involving moving boundaries/interfaces. A PINN was proposed (Wang and Perdikaris, 2021) with a two-network architecture to solve simple benchmark problems on free/moving interfaces (known as Stefan problems), in both forward and inverse settings. One network predicts/learns the previously unknown/known moving interface position, and the other

learns the complete solution of the underlying PDEs with appropriate constraints at the interface, enforced through the loss function. A two-phase incompressible flow problem with moving interface was solved in both forward and inverse settings (Buhendwa *et al.*, 2021), using PINNs with a volume of fluid formulation. Hidden physics recovery was performed using the volume fraction as training variables, where the velocity and pressure fields were recovered as hidden variables. Motivated by the fictitious domain method (FDM), Yang *et al.* (2023*b*) proposed a FDM-PINN approach and demonstrated its capability in solving forward problems with stationary and moving bodies involving linear elliptic/parabolic PDEs. A few studies in the area of vortex induced vibrations of a bluff-body have also been reported (Raissi *et al.*, 2019*b*; Bai and Zhang, 2022; Tang *et al.*, 2022; Bai and Zhang, 2022). These studies considered a body attached frame of reference to model the flow around the body. Note that transferring CFD data from an Eulerian frame of reference to a body attached frame of reference is not feasible in the case of multiple moving bodies or multiple deforming structures. In such situations, a fixed Eulerian frame of reference (as used in IBM) is beneficial and effective. To combine the advantages of IBM and PINNs, Huang *et al.* (2022) proposed a direct forcing based IB-PINNs approach and applied it on a steady flow problem past a stationary cylinder for Reynolds numbers $Re < 40$. The physics losses were minimized in the entire domain considering the solid and fluid regions together. However, it is not clear if in the IB-PINN formulation, including the solid region in the physics loss calculation is necessary when the body position and velocity are known *a priori*. More recently, in another application on moving body (Calicchia *et al.*, 2023), a standard Navier-Stokes (N-S) formulation based PINN considering a non body-attached (inertial) frame of reference was used. PIV data was used and the near-field pressure data around the moving body was recovered effectively in this approach. Given the flexibility of PINNs in choosing collocation points anywhere in the domain, under a body non-conformal approach, it would thus be interesting to test the efficacy of IB-PINN (Huang *et al.*, 2022) in comparison with the standard collocation based PINN in an inertial frame of reference, where the solid region

is discarded. Such a comparison has not been undertaken in the literature, to the best of the author's knowledge. The role of the solid region grid points in an IBM setting on the overall performance of the PINN is also yet to be understood. These are some of the pertinent issues that need to be addressed in light of the above discussion and in the context of handling flow past moving bodies, especially in a non body-attached frame of reference. More recently, in some contemporary works of Zhu *et al.* (2023, 2024), PINNs were employed only in the fluid region to solve forward and inverse problems. Notably, forward problems are extremely challenging to solve. In the work of Zhu *et al.* (2024), to obtain forward solutions with reasonable precision for a flapping wing case study, Zhu *et al.* (2023) resorted to a very small time domain for training such that it contributed only a fraction of the overall flapping time period. Thus forward problems being paricularly challenging to solve using PINNs, GPU accelerated traditional CFD solvers (Chuang and Barba, 2022; Yuen *et al.*, 2013) are preferable for such problems considering the overwhelming benefits of low turnaround times.

Systems with moving boundaries are particularly challenging for PINNs to model because the domain evolves over time, complicating the enforcement of boundary conditions and increasing computational complexity. Additionally, moving boundaries often involve coupled nonlinear dynamics and can introduce sharp gradients or discontinuities in the solution, which are challenging for neural networks to approximate accurately. In this context, some previous studies relied on domain transformations to body-attached frame of reference (Raissi *et al.*, 2019*b*). But this can be limiting when handling multiple moving bodies and deforming flexible structures. Hence, in the context of unsteady flows past moving/flapping wing-like bodies, the use of a fixed background Eulerian grid would be advantageous for training PINNs. This served as the motivation for some of the recent works (Huang *et al.*, 2022; Yang *et al.*, 2023*b*; Calicchia *et al.*, 2023; Zhu *et al.*, 2024). While the results were promising, a deeper understanding of the relative merits of standard NS-based PINNs or IBM-based PINNs was missing.

Some of the major challenges, when working with unsteady flow-field data past moving bodies, arise in the temporal domain of the problem. It is possible that even with periodic actuation of the solid body, the resultant flow-field could be aperiodic in nature, which can bring in multiple time scales (Lewin and Haj-Hariri, 2003). Adequate sampling of the flow-field becomes important to represent all the time scales. Moreover, an understanding of the dynamical behavior of the flow and its interaction with the moving body depends on long time integration of the flow-field, posing a challenge for PINNs as they do not train well on long-time domains (Krishnapriyan *et al.*, 2021). Often, due to memory/storage constraints, the temporal resolution of the obtained data can be coarser than the minimum required Nyquist sampling criterion. In such scenarios, standard interpolation techniques might not suffice for flow-field reconstruction at intermediate time stamps. Moreover, it might not be feasible to rerun the simulations or experiments for the entire spatial domain when the region of interest is often a truncated spatial subdomain encompassing the near-field region and the moving body. Hence, a physics-informed continuous time-space interpolator like a PINN would be beneficial as it can work under data-sparse conditions (Cai *et al.*, 2021*b*), and with any arbitrary domain (Raissi *et al.*, 2020). Importantly, sparsity is a computational source of complexity for PINNs, whereas flow-field aperiodicity necessitates long-time integration and thus strongly tied to the physics of the flow. Aperiodic flows past flapping foils have been investigated by the community owing to their ability to enhance lift/thrust generation (Khalid *et al.*, 2018; Lewin and Haj-Hariri, 2003; Platzer *et al.*, 2008; Bose and Sarkar, 2018; Majumdar *et al.*, 2022; Shah *et al.*, 2021). Such flow-fields contain rich temporal scales although the flow-field might look seemingly regular. A rich spectral content might prove to be challenging for PINNs owing to their frequency bias issues. One way to mitigate this is to use Fourier embeddings (Wang *et al.*, 2021*b*). However, it is impossible to pre-assess the frequency content of the flow-field, before experiments/simulations. Also, even with available data, it might be challenging to determine the spectrum if the temporal resolution is poor.

Considering the above situations, effective training strategies need to be employed to handle the temporal domain complexity of different forms: (1) temporal sparsity, (2) long-time domain integration, and (3) rich spectral content in case of aperiodicity. A unified causal framework for soft/hard causality enforcement through weighting or time domain decomposition strategies was proposed recently (Penwarden *et al.*, 2023). These sequential learning strategies, more than parallel training, can benefit from the transfer learnability of traditional neural networks. But this is applicable only when the networks train on equidistant time/spatial windows in the physical domain. Although it might seem promising to consider varied sizes of networks to train different temporal segments (Jagtap and Karniadakis, 2020), transfer learning would only be beneficial when the time windows/segments chosen are the same, given that spatial features are often quite similar across temporal snapshots of periodic/aperiodic flows. This motivates the present study in which different sequential learning strategies have also been explored, including time marching (Krishnapriyan *et al.*, 2021), backward compatible training (Mattey and Ghosh, 2022), and sequential learning with transfer learning methods (Tang *et al.*, 2022). These have also been investigated in the context of unsteady flows past a flapping airfoil in the current dissertation.

1.8 RESEARCH GAPS AND OPEN PROBLEMS

Based on the review of relevant literature in previous sections, the following open problems have been identified:

- Immersed Boundary Methods (IBM) provide a computationally efficient framework for simulating flows over complex, deforming, and multi-body geometries using fixed Eulerian grids. However, the resulting data contain fictitious or non-physical flow values within the solid region, limiting the direct use of classical projection-based model reduction techniques such as POD or DMD. Reconstructing physically consistent fields in this body-non-conformal, domain-agnostic setting thus remains an open challenge—one that motivates recasting IBM data reconstruction as a physics-constrained learning problem, and developing a framework with broad applicability to single or multiple moving bodies.

- In many experimental and numerical IBM setups, pressure fields or boundary load

distributions are either unrecorded or under-resolved, since momentum forcing alone suffices for force computation in certain IBM variants (Kim *et al.*, 2001; Majumdar *et al.*, 2020). Recovering such hidden physical quantities—particularly pressure—from limited spatio-temporal flow data becomes essential for post-analysis, model validation, and load estimation. Developing non-intrusive, data-efficient surrogate models that can infer these hidden variables without explicit boundary data forms a key focus of this thesis.

- Most of the recent studies (Huang *et al.*, 2022; Yang *et al.*, 2023*b*) which discussed immersed boundary or fictitious domain based PINNs, attempted either steady state problems or time dependent but linear problems. Unsteady flows past moving bodies governed by NS equations and that too with some temporal domain complexities are challenging and not tackled in earlier studies (see table 1.2 for a brief summary of list of earlier and some contemporary contributions).

- In realistic moving-body flow problems, body shape, position, or kinematic quantities such as velocity and acceleration are often unavailable or inaccurately represented. In such cases, inference must proceed in an inertial, body-nonconformal frame while implicitly recovering the hidden boundary motion. The feasibility of surrogate formulations capable of learning both the flow field and boundary evolution simultaneously, ensuring consistent reconstruction of pressure and surface forces needs to be investigated. Similarly, feasibility of the surrogate formulations under no-data scenarios such as forward problems also need to be investigated.

- Unsteady flows past moving bodies can exhibit complex dynamics and a broadband temporal spectra even with periodic actuation of the moving body which challenges development of both data-driven and physics-informed learning frameworks. In addition, sparsity of collected spatio-temporal data or limited observation windows can hinder the ability of the model to capture relevant dynamical modes at intermediate time instants where data is unavailable. Understanding how temporal resolution, physics loss formulation, and time-domain decomposition affect surrogate model performance constitutes an important gap addressed in this study.

1.9 SCOPE AND OBJECTIVES OF THE THESIS

Keeping the need and potential benefits of physics informed neural networks in mind, the primary focus of the thesis is to develop a mesh agnostic and data efficient surrogate modeling framework for unsteady flows past moving bodies. This study will establish an unsteady flow past plunging airfoil as the ideal test bench to develop data-driven or physics-informed models owing to its key characteristics. Hence, throughout this study, flow past a plunging foil is considered at a low Reynolds number $O(100)$. A

Study	Method	Key contribution	Test case
Raissi *et al.* (2019*b*)	PINN	PINN for hidden variable recovery in a body attached frame of reference	Two dimensional flow past vibrating cylinder
Wang and Perdikaris (2021)	PINNs	Two sub-networks. One for the fluid and the other for the moving interface.	Stefan PDEs
Buhendwa *et al.* (2021)	PINNs	Volume of fluid based PINN was proposed	Two phase incompressible flow
Tang *et al.* (2022)	TL-PINNs	Transfer learning was adopted to reconstruct flow past a two dimensional fixed cylinder	Two dimensional fixed cylinder
Huang *et al.* (2022)	IB-PINN	IBM based PINN for time independent BVP	Fixed cylinder steady flow
Yang *et al.* (2023*b*)	FDM-PINN	Fictitious domain PINN for forward problems	Linear elliptic and parabolic steady and unsteady PDEs
Zhu *et al.* (2023)	PINN	Demonstrated the effectiveness of PINNs for dynamic interface problems	Steady state fluid-solid interaction without deformation of the structure

Table 1.2: Summary of contributions of earlier and some contemporary studies on PINNs for moving boundary problems.

high fidelity incompressive Navier Stokes based solver based on the discrete forcing immersed boundary method shall be primarily used to obtain the required training data for the machine learning based models. Although CFD simulations of unsteady flow are possible, they are still computationally intensive (Álvarez-Farré *et al.*, 2021; Mader *et al.*, 2020; Reguly and Mudalige, 2020). This is because, the CFD algorithms are dominated by iterative linear algebra routinesWood *et al.* for the solution of Navier-Stokes (N-S) and allied governing equations. With parallel computing it is possible to accelerate such linear algebraic routines. With the advent of general purpose praphics processing units (GPUs), these compute intensive linear algebraic routines can be parallelised and offloaded onto the GPUs. Hence, the inhouse IBM solver which is written in C++ is parallelized on the GPU using OpenACC (OpenACC-Standard.org, 2015), a directive based parallelization framework by NVIDIA. A directive based paradigm is chosen because of lesser code intrusion and minimal development time to migrate a previous CPU parallel version to the GPU. The machine learning models are built in Python language using the Tensorflow deep learning framework. Machine learning model predictions especially the pressure recovered will be compared against the pressure fields of ALE framework.

Overall, this study aims to provide a novel framework to tackle inverse problems arising in unsteady flow past moving bodies. Specifically, attention will be towards the problem of pressure recovery as a hidden variable using PINNs under different spatio-temporal sparsity and temporal complexity conditions. The study uses a single moving body example, but the framework is not restricted to a single body and is applicable for multiple moving bodies. Note that, moving body problem encompasses key characteristics of the flows past moving bodies: strong vortices with localized regions of strong gradients, and time varying flow behavior. The above mentioned characteristics would pose challenges while training for PINNs in general owing to their failure modes as discussed in earlier sections.

The key contributions of the present study are identified as follows:

- In order to build surrogate models for unsteady flows past moving bodies in a fixed frame of reference, a body non-conformal immersed boundary aware framework based on PINNs has been developed.

- Using this framework, recovery of near-field pressure and reconstruction of velocity data at any spatio-temporal coordinate within a domain of interest while enabling real time query have been done.

- Under the above framework, two PINN formulations, based on the standard N-S equation, and IBM, have been evaluated, specifically for non-intrusive recovery of pressure from coarse velocity data.

- To improve the model performance under a fixed computational budget, the efficacy of physics residual loss relaxation has been investigated.

- A multi-part physics loss weighting strategy, in addition to the standard physics loss relaxation, has been devised for the IBM-based PINN formulation to isolate and explain the effects of the solid region on the model performance.

- To improve data efficiency while maintaining acceptable levels of accuracy, a physics-based undersampling strategy has been proposed.

- To evaluate the framework for velocity reconstruction and pressure recovery under different temporal domain complexities and identify the limitations.

- Extension of the IBA framework to efficiently handle temporal domain complexities using time marching, and temporal decomposition based sequential strategies (Krishnapriyan *et al.*, 2021; Mattey and Ghosh, 2022) coupled with transfer learning (Tang *et al.*, 2022) has been attempted along with a preferential spatio-temporal sampling strategy devised to handle data sparse conditions.

- Detailed evaluation of the proposed methods under a fixed training budget over both periodic and quasi-periodic flow situations past a flapping body depicting the different temporal domain complexities discussed above.

Note that PINNs based modeling of the flow in a body-attached frame of reference could be restrictive and is not amenable for multiple moving bodies or deforming structures. But the generality of the proposed boundary non-conformal approach would allow handling multiple moving bodies/deforming structures in an inertial frame of reference. For the sake of simplicity, a single moving body has been considered in the present study for demonstrating the framework. To the best of the author's knowledge, this is the first time performance of PINNs has been analysed in the context of flapping wings, and that too

in a body non-conformal setting. Moreover, through a set of investigations in a proof of concept manner, the proposed framework would also be amenable to tackle inverse problems such as hidden boundary shape and velocity estimation, super-resolution, and a forward problem involving flows past a single movind body. Note that the forward problem example is attempted in the current study only as a proof of concept to highlight the persistent difficulties in training PINNs over a long time domain for a forward problem (Krishnapriyan *et al.*, 2021; Wang *et al.*, 2024; Zhu *et al.*, 2024) where one has absolutely no data in the time domain except for the initial condition.

1.10 OUTLINE OF THE THESIS

The thesis is structured as follows. The development of the immersed boundary aware framework and its application to the pressure recovery problem is discussed in Chapter 2. Followed the detailed evaluation of the formulations presented in Chapter 2, in Chapter 3, the potential of the IBA framework under different data availability scenarios will be explored and limitations will be highlighted. In Chapter 4, the need to tackle different temporal domain complexities arising in unsteady flow datasets such as (1) temporal sparsity of data sets, (2) long time domains integration to determine the flow physics, and (3) aperiodicity will be discussed. The IBA framework would be subsequently extended to handle these complexities using sequential learning methods and data-efficient physics based sampling techniques. Finally in Chapter 5, the overall summary and conclusion, salient contributions, key limitations and future work will be highlighted. Appendix A presents the development and validation of the GPU accelerated IBM solver used in the study to generate the training data. In addition, Appendix B shall present case studies demonstrating input subdomain level contributions from different spatial zones towards training of PINN models, and discusses the combined effectiveness of the proposed physics-based data sampling and physics loss relaxation.

CHAPTER 2

IMMERSED BOUNDARY AWARE FRAMEWORK OF SURROGATE MODELING USING PHYSICS INFORMED NEURAL NETWORKS

2.1 INTRODUCTION

Instantaneous pressure fields around moving bodies such as natural swimmers or flyers are crucial for understanding the coupled dynamics between body acceleration and vortex-induced flow structures (Daniel, 2015). In experimental studies, however, only indirect measurements—such as flow visualization of passive scalars (smoke, dye, contrast agents) (Lai and Platzer, 1999) or planar velocity fields from particle image velocimetry (PIV) (Dabiri *et al.*, 2014)—are typically available, as these are far easier to obtain than direct pressure data. Consequently, the ability to recover instantaneous pressure non-invasively from flow visualization or PIV-based velocity data becomes invaluable, enabling deeper insight into flow mechanisms and facilitating accurate load estimation on moving bodies (Liu and Katz, 2013; Dabiri *et al.*, 2014; Pirnia *et al.*, 2020).

In numerical simulations, similar challenges arise. For instance, certain variants of the Immersed Boundary Method (IBM) (Lee *et al.*, 2011; Majumdar *et al.*, 2020) compute hydrodynamic loads through the total time derivative of velocity and momentum forcing terms in the Navier–Stokes equations, without explicitly solving for or storing pressure fields. This omission, while computationally efficient, limits post-processing and physical interpretability. Hence, developing accurate, physics-consistent surrogate modeling approaches for near-field pressure recovery—simultaneously reconstructing velocity fields from coarse or sparse data—is of considerable interest. Beyond improving flow analysis, such methods can substantially reduce storage requirements for high-fidelity simulations by eliminating the need to store full pressure fields, allowing efficient

querying of both velocity and pressure at arbitrary spatio-temporal locations.

Nevertheless, surrogate modeling based on IBM-generated simulation data presents several challenges (Balajewicz and Farhat, 2014). Owing to the use of a fixed Eulerian background grid, regions enclosed by Lagrangian markers (representing the moving solid boundary) contain fictitious flow-field values essential for IBM computations. Consequently, at any given time, Eulerian grid points within the range of the solid-body motion may belong either to the fictitious (*solid*) domain or the physical fluid domain. Furthermore, enforcing the no-slip boundary condition is non-trivial, since the Lagrangian markers are not collocated with the Eulerian grid points. Thus, although IBM simplifies the treatment of moving bodies and complex geometries (Kim and Choi, 2019), constructing a surrogate capable of accurately reconstructing velocity and recovering pressure fields remains a significant challenge. In Chapter 1 section 1.4 it was discussed that IBM data reconstruction can be formulated as a weighted least-squares optimization problem. However, the ALS-based decomposition as proposed by Balajewicz and Farhat (2014) offers only a linear approximation of the underlying data structure, which inherently restricts the model expressivity under computational or data constraints. This limitation naturally motivates the adoption of deep learning frameworks, which can capture nonlinear dependencies and complex latent structures beyond the reach of conventional linear methods.

In order to accommodate potential data availability constraints often faced in real datasets, the present work adopts through Physics-Informed Neural Networks (PINNs) (Raissi *et al.*, 2019*a*), a class of deep neural networks (Goodfellow *et al.*, 2016) which incorporate governing equations into the loss function. This way, PINNs regularize learning on available limited/sparse/coarse data and enable the solution of both forward and inverse problems with minimal architectural modification (Raissi *et al.*, 2019*a*). PINNs have been successfully employed in inverse or ill-posed settings, for data-driven PDE discovery (Raissi *et al.*, 2019*a*), data assimilation (Raissi *et al.*, 2019*b*), and hidden

physics recovery (Raissi *et al.*, 2020), with recent surveys highlighting their broad applicability across scientific domains (Karniadakis *et al.*, 2021; Cuomo *et al.*, 2022). The key advantages are that the data points can be selected arbitrarily anywhere in the domain (Raissi *et al.*, 2020) while training, and once trained, a PINN can be queried continuously in space and time within the training domain. Moreover, PINNs are memory-efficient—storing only model parameters suffices for inference—and naturally differentiable owing to automatic differentiation capabilities of modern deep learning frameworks such as TensorFlow (Abadi, 2016), PyTorch (Paszke *et al.*, 2019), and JAX (Bradbury *et al.*, 2018)

As discussed in the Introduction Chapter 1, standard collocation PINNs (Raissi *et al.*, 2019*a*) are sometimes difficult to train for forward problems due to underlying failure modes, broadly classified under: competing optimization objectives, and, propagation failures as discussed in (Wang *et al.*, 2021*a*; Krishnapriyan *et al.*, 2021). Various strategies have been proposed lately– such as, loss component balancing (Wang *et al.*, 2021*a*; Jin *et al.*, 2021), modified backbone architectures (Wang *et al.*, 2021*a*), adaptive sampling (Wu *et al.*, 2023), domain decomposition (Jagtap *et al.*, 2020*c*) and sequential learning (Krishnapriyan *et al.*, 2021; Penwarden *et al.*, 2023) – to tackle these problems. A more detailed list of scalable algorithms to train PINNs efficiently can be found in the works of Shukla *et al.* (2022); Penwarden *et al.* (2023). Inspite of this, forward problems using PINNs are often intractable compared to GPU accelerated CFD simulations. Instead it is more viable to use PINNs only in ill-posed or inverse problems where multiple CFD simulations become intractable as compared to a single training procedure. Recently, Krishnapriyan *et al.* (2021) and Penwarden *et al.* (2023) have shown that standard PINNs are difficult to train in situations when no measurements/simulation data are available, especially for unsteady convection dominated problems. On the other hand, (Gopakumar *et al.*, 2023) showed that PINNs become relatively easier to train if at least some sparse/coarse simulation data or partially observed experimental data are available. It has also been shown that when simulation data is available, the expressivity of PINNs

can be improved by appropriately relaxing the physics loss components, without the need for any architectural modifications (Lucor *et al.*, 2022). In light of this, with simulation data being available in the present study, it is also of particular interest to investigate strategies that improve the velocity reconstruction and pressure recovery capability of PINNs under a fixed computational budget without the need for architectural changes.

PINNs based approaches have been recently tested for problems involving moving boundaries/interfaces. A PINN was proposed (Wang and Perdikaris, 2021) with a two-network architecture to solve simple benchmark problems on free/moving interfaces (known as Stefan problems), in both forward and inverse settings. One network predicts/learns the previously unknown/known moving interface position, and the other learns the complete solution of the underlying PDEs with appropriate constraints at the interface, enforced through the loss function. A two-phase incompressible flow problem with moving interface was solved in both forward and inverse settings (Buhendwa *et al.*, 2021), using PINNs with a volume of fluid formulation. Hidden physics recovery was performed using the volume fraction as training variables, where the velocity and pressure fields were recovered as hidden variables. Motivated by the fictitious domain method (FDM), (Yang *et al.*, 2023*b*) proposed a FDM-PINN approach and demonstrated its capability in solving forward problems with stationary and moving bodies involving linear elliptic/parabolic PDEs. A few studies in the area of vortex induced vibrations of a bluff-body have also been reported (Raissi *et al.*, 2019*b*; Bai and Zhang, 2022; Tang *et al.*, 2022; Bai and Zhang, 2022). These studies considered a body attached frame of reference to model the flow around the body. Note that transferring CFD data from an Eulerian frame of reference to a body attached frame of reference is not feasible in the case of multiple moving bodies or deforming structures. In such situations, a fixed Eulerian frame of reference (as used in IBM) is beneficial and effective. To combine the advantages of IBM and PINNs, Huang *et al.* (2022) proposed a direct forcing based IB-PINNs approach and applied it on a steady flow problem past a stationary cylinder for Reynolds numbers $Re < 40$. The physics losses were minimized in the entire domain

considering the solid and fluid regions together. However, it is not clear if in the IB-PINN formulation, including the solid region in the physics loss calculation is necessary when the body position and velocity are known *a priori*. More recently, in another application on moving body (Calicchia *et al.*, 2023), a standard Navier-Stokes (N-S) formulation based PINN considering a non body-attached (inertial) frame of reference was used. PIV data was used and the near-field pressure data around the moving body was recovered effectively in this approach. Given the flexibility of PINNs in choosing collocation points anywhere in the domain, under a body non-conformal approach, it would thus be interesting to test the efficacy of IB-PINN (Huang *et al.*, 2022) in comparison with the standard collocation based PINN in an inertial frame of reference, where the solid region is discarded. Such a comparison has not been undertaken in the literature, to the best of the authors' knowledge. The role of the solid region grid points in an IBM setting on the overall performance of the PINN is also yet to be understood. These are some of the pertinent issues that need to be addressed in light of the above discussion and in the context of handling flow past moving bodies, especially in a non body-attached frame of reference. Based on the above discussion, the key objectives and outcomes of the present chapter are identified as follows:

- In order to build surrogate models for unsteady flows past moving bodies in a fixed frame of reference, a body non-conformal immersed boundary aware framework based on PINNs has been developed here.
- Under the above framework, two PINN formulations, based on the standard N-S equation, and IBM, have been evaluated, specifically for non-intrusive recovery of pressure from coarse velocity data.
- To improve the model performance under a fixed computational budget, the efficacy of physics residual loss relaxation has been investigated.
- A multi-part physics loss weighting strategy, in addition to the standard physics loss relaxation, has been devised for the IBM-based PINN formulation to isolate and explain the effects of the solid region on the model performance.
- To improve data efficiency while maintaining acceptable levels of accuracy, a physics-based undersampling strategy has been proposed.

Often in flow problems involving a moving body, the solid body is allowed to move either based on a prescribed kinematics or as dictated by the structural dynamical equations coupled with the flow solver. The present study deals with the former scenario. One of the main aims of the present surrogate modeling framework is to recover the near-field pressure as a hidden variable, and reconstruct velocity data at any spatio-temporal coordinate within a domain of interest while enabling real-time query. The present study proposes to achieve this by training a physics informed neural network (PINN) in a body non-conformal manner. As in the IBM, where the flow is investigated in an Eulerian grid and the solid is represented in a Lagrangian fashion, a similar strategy is proposed here and hence can also be called an immersed boundary aware (IBA) approach.

Note that PINNs based modeling of the flow in a body-attached frame of reference could be restrictive and is not amenable for multiple moving bodies or deforming structures. However, using a body non-conformal and an inertial frame of reference alleviates the need for case specific computational domain transformations to a body-attached frame of reference to tackle a moving boundary problem, as was done in earlier studies by Raissi *et al.* (2019*b*); Bai and Zhang (2022). Hence, the generality of the proposed boundary non-conformal approach would allow handling multiple moving bodies/deforming structures in an inertial frame of reference. For the sake of simplicity, incompressible flow past a single moving body has been considered in the present study for demonstrating the framework. To the best of the author's knowledge, this is the first time PINNs' performance has been analysed in the context of flapping wings, and that too in a body non-conformal setting.

The rest of the chapter is organised as follows. In section 2.2 and section 2.3, the modelling of the flapping foil system and unsteady flow around it are discussed. In sections 2.4 and 2.5, the backbone neural network architecture and loss formulation for the immersed boundary aware PINNs based surrogate modeling framework are discussed in the context of pressure recovery and velocity reconstruction. The pre-setup considerations for the

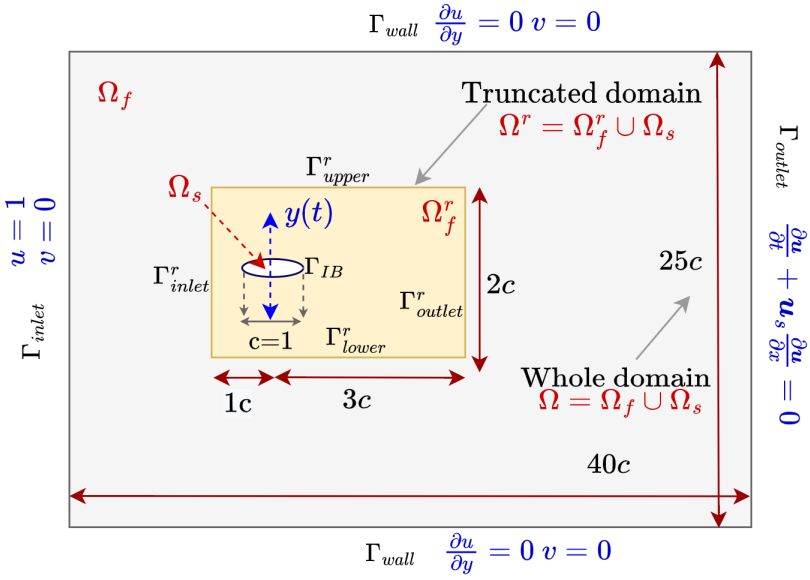


Figure 2.1: Schematics of the problem setup: computational domain Ω is chosen for the IBM solver, and the truncated domain Ω^r is chosen for the surrogate model. Ω_f and Ω^r_f are the fluid regions excluding the solid boundary Γ_{IB} and solid region Ω_s at any given time instant. Γ^r_{inlet}, Γ^r_{outlet}, Γ^r_{upper} and Γ^r_{lower} are the inlet, outlet, upper and lower boundaries of the truncated domain Ω_r, respectively.

PINNs including training-testing database generation and hyperparameter tuning are discussed in section 2.6. Sections 2.7 and 2.8 demonstrate the effects of physics constraint relaxation and a physics based vorticity cut off sampling strategy, respectively. Finally, the discussion and key conclusions are outlined in sections 2.9 and 2.10, respectively.

2.2 THE FLAPPING FOIL SYSTEM

Two-dimensional unsteady incompressible flow past a plunging rigid elliptic airfoil at a low Reynolds number is considered. The foil follows a harmonic kinematics and is considered to be of unit chord length (c = 1) with a thickness to chord ratio of 0.125. It is subjected to uniform free-stream; see figure 2.1(a) for a schematic of the problem. The plunging kinematics is given by, The kinematic model is as follows

$$y(t) = h_a \cos(2\pi f_h t + \phi), \quad \text{and} \tag{2.1}$$

$$\dot{y}(t) = -2\pi f_h h_a \sin(2\pi f_h t + \phi). \tag{2.2}$$

Here, t is dimensional time; h_a and f_h are the plunging amplitude and frequency, respectively. Here, ϕ is the kinematic phase parameter that determines the initial body position at $t = 0$. Aligning with earlier works of Lewin and Haj-Hariri (2003), and Khalid *et al.* (2018), the non-dimensional amplitude $h = h_a/c$ and reduced frequency $k = 2\pi f_h c/U_\infty$ are defined, where, U_∞ is the free stream velocity. Further description of the flapping motion will be made in the terms of k and h.

2.3 MODELING THE UNSTEADY FLOW

The flow around the flapping foil is assumed to be governed by the incompressible Navier-Stokes (N-S) equations given in its non-dimensional form by

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla.(\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{u}, \tag{2.3}$$

$$\nabla.\boldsymbol{u} = 0. \tag{2.4}$$

Here, $\boldsymbol{u}$ represents the non-dimensional velocity vector in the x-y space with u and v being the x and y components of $\boldsymbol{u}$, respectively, and p is the non-dimensional pressure. $Re = \dfrac{U_\infty c}{\nu}$ indicates the Reynolds number with ν being the kinematic viscosity.

A discrete forcing IBM based in-house flow solver developed by Majumdar *et al.* (2020) with GPU acceleration (see Appendix A for details on OpenACC (OpenACC-Standard.org, 2015) based GPU acceleration) has been employed here to solve equations (2.3) and (2.4) for generating the training and testing data for the velocity fields. In IBM approach, the immersed solid boundary Γ_{IB} is represented by a set of Lagrangian markers (figure 2.2(a)) which do not conform exactly with the Eulerian fluid grid. Importantly, at the solid boundary Γ_{IB}, a no-slip boundary condition is to be satisfied such that $\boldsymbol{u}(\boldsymbol{x} = \Gamma_{IB}, t) = \dot{y}(t)$. Since this cannot be directly enforced on Γ_{IB} in the IBM, a momentum forcing term $\boldsymbol{f}$ with f_x and f_y as the respective x and y components must be added in equation (2.3). $\boldsymbol{f}$

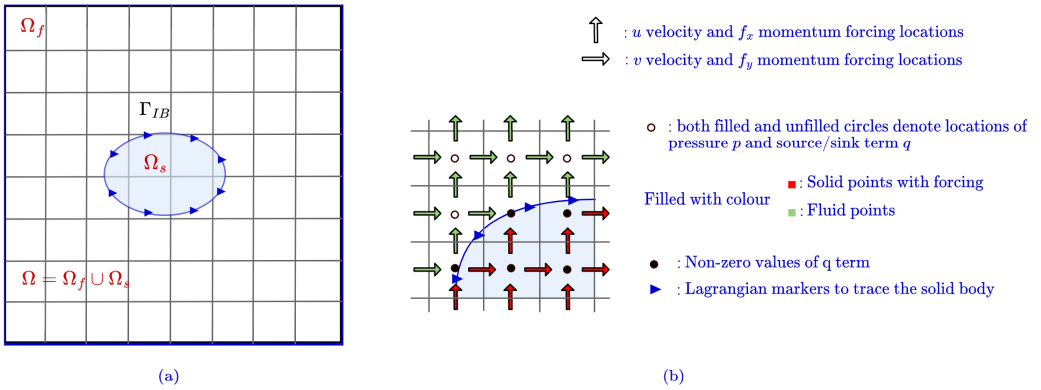


Figure 2.2: Schematics of (a) a representative computational domain $\Omega = \Omega_f \cup \Omega_s$ used in the immersed boundary method. Here Ω_s is the shaded area representing the solid region bounded by the solid boundary Γ_{IB}, and (b) the mesh grid representing the staggered primitive variable arrangement and fluid-solid grid point classifications.

is evaluated such that the no-slip boundary condition is satisfied on Γ_{IB} using appropriate mapping and interpolation between the Eulerian and Lagrangian points. Additionally, a source/sink term q is added in equation (2.4) to strictly satisfy the continuity equation. Both $\boldsymbol{f}$ and q are non-zero inside Ω_s bounded by Γ_{IB}. While $\boldsymbol{f}$ is strictly zero outside Γ_{IB} everywhere in Ω_f, q can be non-zero just outside Γ_{IB} at grid cells that are partially cut by Γ_{IB}, but zero elsewhere in Ω_f. This is depicted by the filled/unfilled markers and colour coded arrows in figure 2.2(b). The governing equations in the current IBM approach takes the form as below,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla.(\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{u} + \boldsymbol{f}, \tag{2.5}$$

$$\nabla.\boldsymbol{u} - q = 0. \tag{2.6}$$

For more details on the IBM solver, please refer to the works of Majumdar *et al.* (2020) and Shah *et al.* (2019).

In the current IBM solver, the pressure-field was not required for computing the aerodynamic loads and hence this data was not stored. For a rigorous testing of the pressure recovery capability of the PINN models, an ALE-based flow solver has been

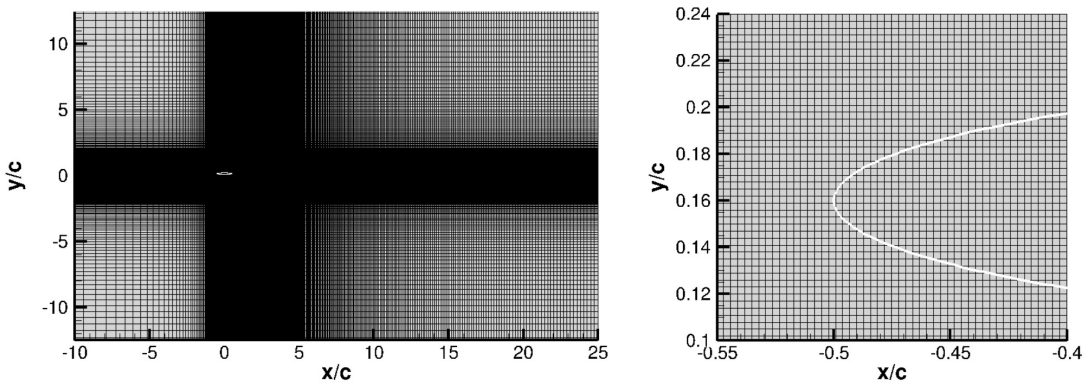


Figure 2.3: (a) Cartesian grid used in IBM solver, and (b) its zoomed view near the solid boundary.

used to generate and test the pressure data. In the ALE technique, the governing equations are solved as it is without the need for any additional forcing or mass source/sink terms in the equations. The ALE based simulations were performed using icoDyMFoam solver from the open-source package foam-Extend (Jasak *et al.*, 2007)).

2.3.1 Solver controls and validation of the CFD solvers

For the sake of completeness, the settings for the IBM and ALE solvers and validation study are discussed here briefly.

The flow simulations were carried out keeping the computational domain (figure 2.1) and IBM solver controls in alignment with the work of (Majumdar *et al.*, 2020). A rectangular computational domain of size $[-10c,\ 25c]\times[-12.5c,\ 12.5c]$ was considered with the elliptic foil being placed at the maximum amplitude position initially. The grid size $\Delta x = \Delta y = 0.004$ and time-step size $\Delta t = 0.0001$ were chosen after performing appropriate grid and time convergence tests for the IBM solver.

The Eulerian mesh was uniformly spaced in the region of the body movement with a minimum cell spacing and then gradually stretched following a geometric progression towards the outer boundaries (resulting in a total of $6.3187e06$ cells) as shown in

figure 2.3(a) with the zoomed view presented in figure 2.3(b). On the other hand, for the ALE solver, the converged unstructured triangulated grid was generated using **gmsh** (Geuzaine and Remacle, 2009) (figure 2.4) and it consisted of $n_{\text{foil}} = 800$ cells on the elliptic foil surface with $3.051e05$ triangular cells in the domain overall. The time step chosen was $\Delta t = 0.0001$, same as was used in the IBM solver. Quantitative and qualitative validation studies of the IBM and ALE solvers were carried out for $Re = 500$ with plunging kinematic parameters $k = 2\pi$ and $h = [0.05, 0.16]$ as in Khalid *et al.*

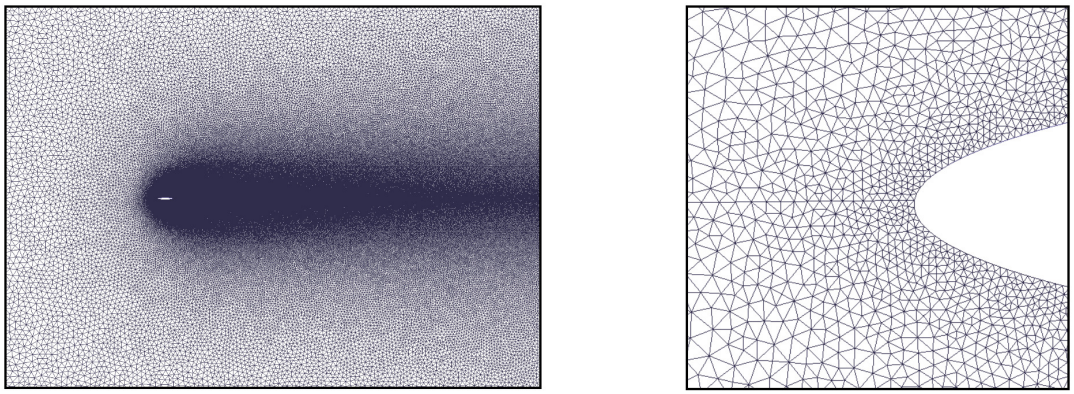


Figure 2.4: (a) Unstructured computational grid used in ALE solver, and (b) its zoomed view near the solid body.

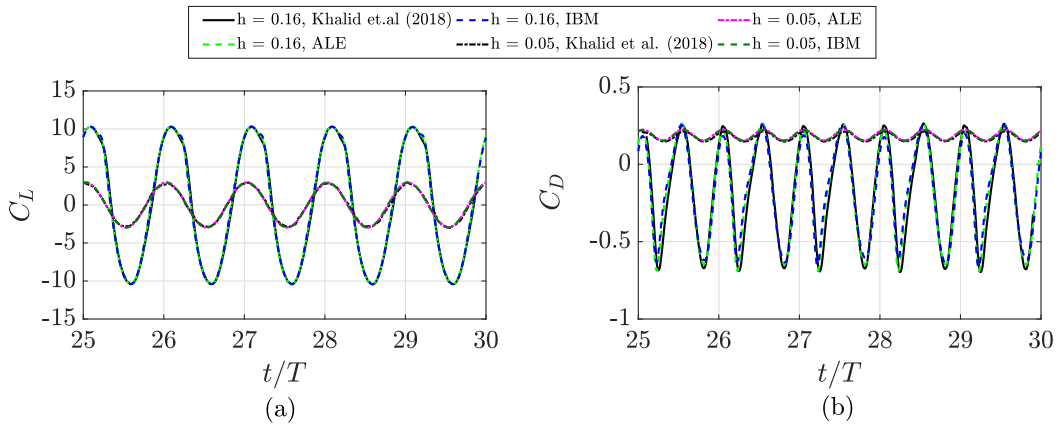


Figure 2.5: Validation of the (a) lift and (b) drag coefficients time histories obtained from IBM and ALE solver with reference to the results presented by Khalid *et al.* (2018).

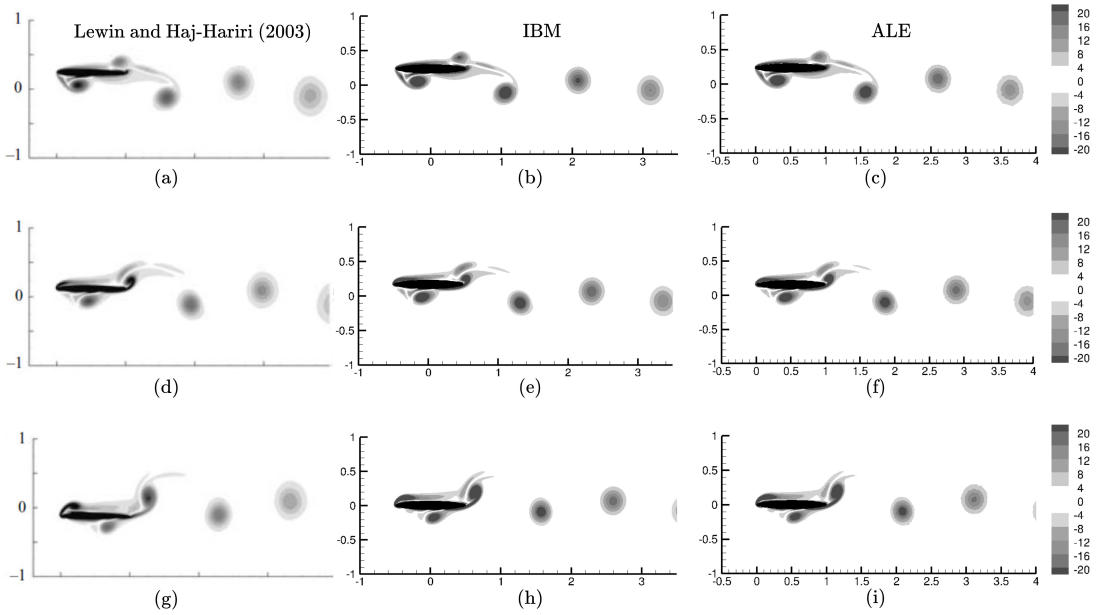


Figure 2.6: Comparison of vorticity fields obtained using the IBM and ALE solvers with that of Lewin and Haj-Hariri (2003) for the downward stroke at $kh = 0.8$ with $k = 3.333$ and $h = 0.24$.

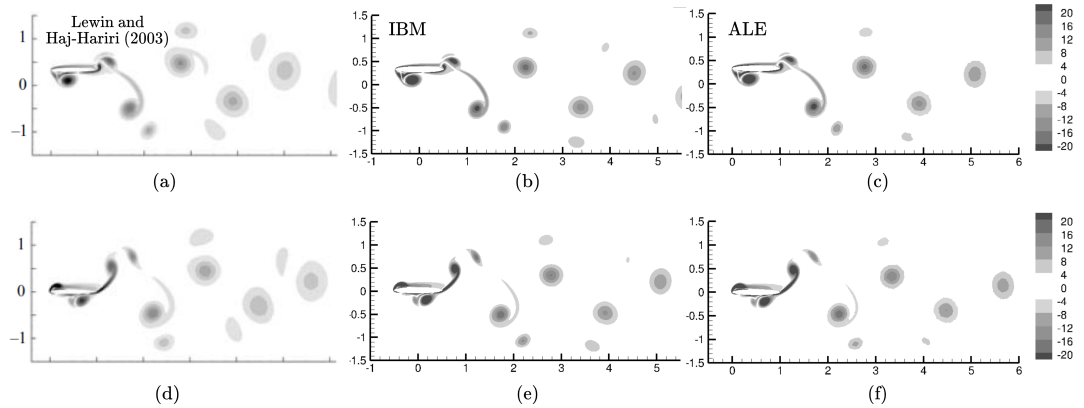


Figure 2.7: Comparison of vorticity fields obtained using the IBM and ALE solvers with that of Lewin and Haj-Hariri (2003) for the downward stroke at $kh = 1.0$ with $k = 3.0$ and $h = 0.333$.

(2018) resulting in $kh = 0.8$ and 1, respectively. The aerodynamic loads obtained from present simulations match very well with that of Khalid *et al.* (2018); see figure 2.5. Additionally, the vorticity snapshots obtained from ALE and IBM solvers are compared with that of Lewin and Haj-Hariri (2003) for $Re = 500$ at $kh = 0.8$ and 1.0 in figures 2.6 and 2.7, respectively. Once again an excellent agreement is seen.

For more extensive validation results of the ALE and the IBM solvers, please refer to earlier works from our group (Badrinath *et al.*, 2017; Majumdar *et al.*, 2020; Shah *et al.*, 2021).

2.4 NEURAL NETWORK ARCHITECTURE

Among the two formulations of PINNs for moving boundaries being presented here, the MB-PINN is formulated only in the fluid region (discarding the solid region) while training; the MB-IBM-PINN is formulated in both the fluid and the solid body regions. The objective is to fit velocity bulk data obtained from IBM simulations and recover pressure as a hidden variable using both MB-PINN and MB-IBM-PINN while investigating their associated efficacy.

For the standard collocation PINN as in Raissi *et al.* (2019*a*), a deep feed forward Neural network (FNN) is chosen as the backbone. Typically, for flow problems, a FNN is formulated to approximate the flow-field variables as outputs, given a spatio-temporal input as in the works of Cai *et al.* (2021*a*); Jin *et al.* (2021). The outputs are expressed as a composition of nonlinear activation functions acting on the spatio-temporal inputs and a network of fully connected hidden layers. Mathematically, this can be expressed as

$$\boldsymbol{h}_0 = (\boldsymbol{x}, t) \tag{2.7}$$

$$\boldsymbol{z}_l = W_l \boldsymbol{h}_{l-1} + b_l, \tag{2.8}$$

$$\boldsymbol{h}_l = \phi(W_l \boldsymbol{h}_{l-1} + b_l), \quad \text{with} \quad l = 1, 2, \cdots L, \text{and} \tag{2.9}$$

$$\boldsymbol{o} = W_{L+1} \boldsymbol{h}_L + b_{L+1}, \tag{2.10}$$

where, $(\boldsymbol{x}, t) \in \mathbb{R}^{n_{\boldsymbol{x}}+1}$ is the spatio-temporal input with $\boldsymbol{h_0} \in \mathbb{R}^{n_{\boldsymbol{x}}+1}$ being the input layer and $n_{\boldsymbol{x}}$ being the spatial dimension. Here, $\boldsymbol{z}_l \in \mathbb{R}^{n_l}$ and $\boldsymbol{h}_l \in \mathbb{R}^{n_l}$ are the pre-activation and activation units of the neural network with L hidden layers and n_l hidden neurons for $1 < l < L$, respectively, with $\phi(.)$ being a non-linear activation function. Throughout the rest of this thesis, the sigmoidal linear unit (SiLU/ Swish) (Ramachandran *et al.*, 2017) activation function has been employed owing to strong evidence of its good performance across different test cases in the deep learning domain (Dubey *et al.*, 2022; Al Safwan *et al.*, 2021; Nwankpa *et al.*, 2018). The quantities $W_l \in \mathbb{R}^{n_{l-1} \times n_l}$ and $b_l \in \mathbb{R}^{n_l}$ for $1 < l < L + 1$, represented by $\boldsymbol{\theta}$ as a whole, are the weights and biases of the neural network to be optimized, respectively.

Under the IBA framework, for the MB-PINN (see schematic figure 2.8(a)), the outputs $\boldsymbol{o}$ are the flow-field variables $\{\boldsymbol{u},\ p\}$. Whereas, in MB-IBM-PINN (see schematic figure 2.8(b)), the outputs $\boldsymbol{o}$ are the flow-field variables and additionally, the body forcing and mass source/sink terms such that $\boldsymbol{o} = \{\boldsymbol{u},\ p,\ \boldsymbol{f},\ q\}$. A FNN is often trained by optimizing the network parameters $\boldsymbol{\theta}$ with a large but potentially sparse or corrupt set of training data consisting of inputs and their corresponding target labels/outputs represented by $\{(\boldsymbol{x}^i, t^i), \hat{\boldsymbol{o}}^i\}_{i=1}^{N_{train}}$ consisting of N_{train} input-output samples with $\hat{\boldsymbol{o}}$ representing the true outputs to be fitted by the network. The optimization is carried out by minimising a loss/cost function, $\mathcal{L}$, to obtain an optimal set of network parameters, $\boldsymbol{\theta}$, such that

$$\boldsymbol{\theta}^{optimal} = \arg\min_{\boldsymbol{\theta}}(\mathcal{L}). \tag{2.11}$$

Numerically, the loss minimization to obtain the optimal network parameters is often performed by first randomly initialising $\boldsymbol{\theta}$ and then iteratively updating it using a variant of a stochastic gradient descent (SGD) based algorithm. Here, the gradients of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ given by $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ are to be calculated. These gradients can be computed using back propagation and automatic differentiation (AD) (Baydin *et al.*, 2017) which is an accurate and efficient way of computing derivatives in a computational graph following the chain rule of differentiation. AD is implemented in most modern deep learning

packages, such as Tensorflow (Abadi, 2016) which is used in the present study. Generally, given K_{MB} mini-batches (sub sets) of the training data containing N_{MB} samples to evaluate the loss and its gradients with respect to the parameters, the algorithm for the parameter updates based on SGD is given by

$$\boldsymbol{\theta}^{i+1} \longleftarrow \boldsymbol{\theta}^{i} - \eta_i \frac{1}{N_{MB}} \sum_{i'=iN_{MB}+1}^{(i+1)N_{MB}} \nabla \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\theta}, (\boldsymbol{x}^{i'}, t^{i'})), \quad i = 1, 2, \cdots K_{MB} \tag{2.12}$$

where η_i is the learning rate corresponding to i^{th} epoch where each epoch corresponds to $K_{MB} = N_{train}/N_{MB}$ SGD iterations. Specifically, ADAM (Kingma and Ba, 2017), a variant of SGD has been used in the literature (Raissi *et al.*, 2019*a*) which will be followed in the present work as well. The detailed loss formulations of MB-PINN and MB-IBM-PINN are discussed in the following section.

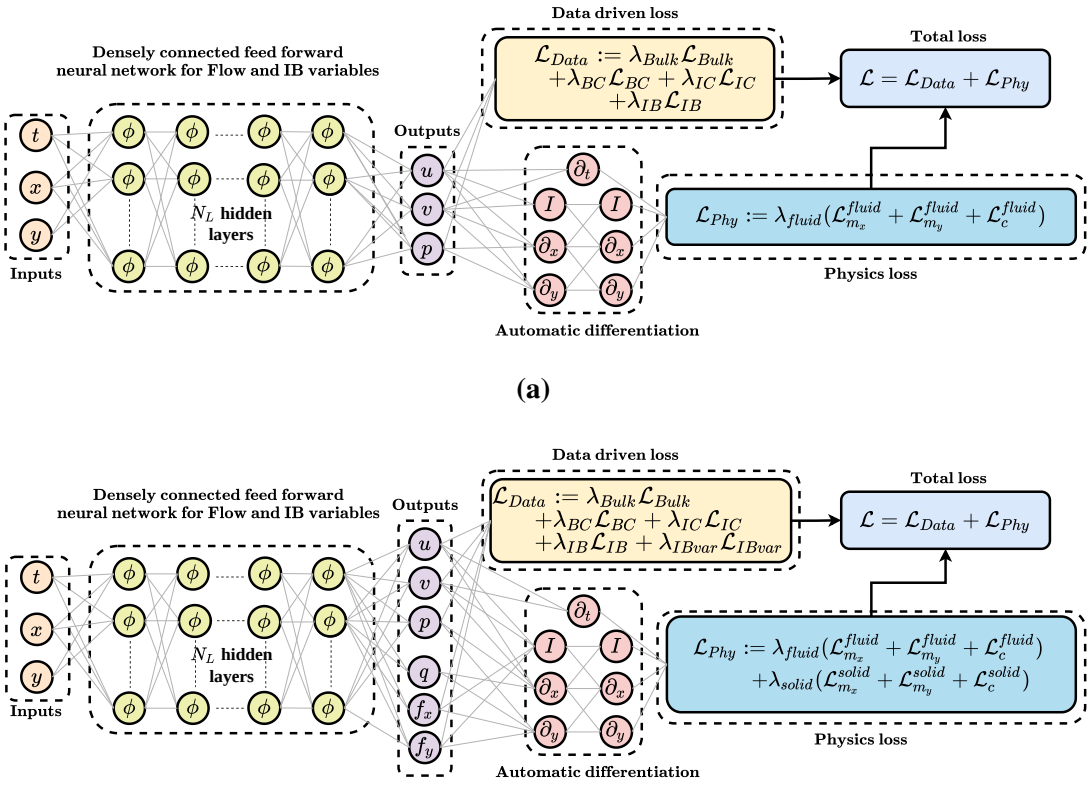


Figure 2.8: Schematic of (a) MB-PINN and (b) MB-IBM-PINN network architectures used under the present IBA framework

2.5 LOSS FORMULATION

The loss function $\mathcal{L}$ of a PINN relying on both data and physical knowledge is represented as

$$\mathcal{L} := \mathcal{L}_{Data} + \mathcal{L}_{Phy}, \tag{2.13}$$

where $\mathcal{L}_{Data}$ and $\mathcal{L}_{Phy}$ correspond to the loss contributions from the data and physics components, respectively. Here, $\mathcal{L}_{data}$ is an indicator of the misfit between the true outputs $\hat{\boldsymbol{o}}^{train}$ and predicted network outputs $\boldsymbol{o}$, respectively; $\mathcal{L}_{Phy}$ uses AD to compute the gradients of $\boldsymbol{o}$ with respect to the input variables $(\boldsymbol{x}, t)$ and compose them together to compute the governing equation residuals. This way, a PINN embeds the prior knowledge of the underlying physics in the training algorithm.

The data and physics loss components can be further split into respective weighted contributions (Heydari *et al.*, 2019; Wang *et al.*, 2021*a*; Bischof and Kraus, 2025) from initial and boundary conditions, interior bulk data and governing equation residuals. The loss formulation in terms of the various components is as follows

$$\mathcal{L}_{Data} := \lambda_{Bulk}\mathcal{L}_{Bulk} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{IC}\mathcal{L}_{IC} + \lambda_{IB}\mathcal{L}_{IB} \tag{2.14}$$

$$\mathcal{L}_{Phy} := \lambda_{Phy}(\mathcal{L}_{m_x} + \mathcal{L}_{m_y} + \mathcal{L}_c). \tag{2.15}$$

Here, $\mathcal{L}_{Bulk}$, $\mathcal{L}_{BC}$, $\mathcal{L}_{IC}$ and $\mathcal{L}_{IB}$ are the loss components corresponding to the predicted interior bulk velocity data, velocity boundary conditions and initial condition on the Eulerian grid. The parameters $\lambda_{Bulk}, \lambda_{IC}$, and λ_{BC} are the weighting coefficients of the bulk data loss, initial and boundary condition loss components, respectively. Here, λ_{IB} is the generic weighting coefficient of the additional no-slip velocity boundary condition loss $\mathcal{L}_{IB}$. It is characteristic to the IBA framework which allows enforcing the no-slip boundary conditions directly on the Lagrangian points similar to Huang *et al.* (2022) instead of following a set of sequential steps and interpolation as in IBM (Majumdar *et al.*, 2020). This is possible owing to the underlying flexibility of PINNs in choosing

collocation points anywhere in the domain to enforce appropriate constraints on them. Thus, both MB-PINN and MB-IBM-PINN allow flexibility in providing both the Eulerian grid points describing the fluid and Lagrangian points describing the solid boundary, respectively. λ_{Phy} is the weighting coefficient of components of the physics informed loss $\mathcal{L}_{Phy}$, which is discussed later in this section, after presenting the data-driven loss components.

The data-driven loss components mentioned above are common to both MB-PINN and MB-IBM-PINN, which can be mathematically expressed as

$$\mathcal{L}_{Bulk} := \frac{1}{N_{Bulk}} \sum_{i=1}^{N_{Bulk}} \|\boldsymbol{u}(\boldsymbol{x}^i_{Bulk}, t^i) - \hat{\boldsymbol{u}}(\boldsymbol{x}^i_{Bulk}, t^i)\|^2_{L_2}, \tag{2.16}$$

$$\mathcal{L}_{BC} := \frac{1}{N_k} \sum_{k=1}^{N_k} \left(\frac{1}{N_{BC_k}} \sum_{i=1}^{N_{BC_k}} \|\boldsymbol{u}(\boldsymbol{x}^i_{BC_k}, t^i) - \hat{\boldsymbol{u}}(\boldsymbol{x}^i_{BC_k}, t^i)\|^2_{L_2} \right), \tag{2.17}$$

$$\mathcal{L}_{IC} := \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \|\boldsymbol{u}(\boldsymbol{x}^i_{IC}, t_0) - \hat{\boldsymbol{u}}(\boldsymbol{x}^i_{IC}, t_0)\|^2_{L_2}, \quad \text{and} \tag{2.18}$$

$$\mathcal{L}_{IB} := \frac{1}{N_{IB}} \sum_{i=1}^{N_{IC}} \|\boldsymbol{u}(\boldsymbol{x}^i_{IB}, t^i) - \hat{\boldsymbol{u}}(\boldsymbol{x}^i_{IB}, t^i)\|^2_{L_2}, \tag{2.19}$$

where, $\hat{\boldsymbol{u}}$ is the true velocity data generated for training by the CFD simulation, whereas, $\boldsymbol{u}$ is the network predicted velocity flow-field data. The spatial and temporal points are represented by $(\boldsymbol{x}, t)$ with $\boldsymbol{x} \in \Omega^r$ and $t \in [0, T]$, respectively. The subscripts shown in the loss expressions for the spatio-temporal points are to indicate independence in sampling the select points of the particular loss component (as depicted by different markers in the schematic figure 2.9). Here, $(\boldsymbol{x}^i_{Bulk}, t^i)$ for $i = \{1, \cdots N_{Bulk}\}$ corresponds to the set of interior velocity data points excluding the solid region. The set of points $(\boldsymbol{x}^i_{BC_k}, t^i)$ for $i = \{1 \cdots N_{BC_k}\}$ with $k = \{1, 2, 3\}$ correspond to the Dirichlet boundary conditions collected at the inlet, Γ^r_{inlet}, upper and lower boundaries, Γ^r_{upper} and Γ^r_{lower}, respectively (see figure 2.1), of the truncated domain Ω^r considered. For simplicity, the inlet and the upper-lower boundary condition losses shall be referred to as $\mathcal{L}_{inlet}$ and $\mathcal{L}_{walls}$, respectively. The set of points $(\boldsymbol{x}^i_{IC}, t_0)$ for $i = \{1, \cdots N_{IC}\}$ are collected at the initial time stamp. In the scenario where the bulk data points are already available at

all times, $\mathcal{L}_{IC}$ need not be enforced additionally. But in the absence of such bulk data, it would be necessary to enforce $\mathcal{L}_{IC}$ in order to infer the flow-fields accurately. Here, $(\boldsymbol{x}^i_{IB}, t^i)$ for $i = \{1, \cdots N_{IB}\}$ correspond to the set of Lagrangian markers lying on the immersed solid boundary as represented in figure 2.9.

In the case of MB-IBM-PINN, additional constraints are required for the momentum forcing term, $\boldsymbol{f}$, and the mass source/sink term, q, which are also additional outputs in the network (see figure 2.8b). It is known that $\boldsymbol{f}$ and q have support only in the solid region bounded by Γ_{IB}, and are zero in the fluid region at any time instant. However, both $\boldsymbol{f}$ and q are not explicitly known inside the solid region as only velocity $\boldsymbol{u}$ data is stored. In Huang *et al.* (2022), in a similar IBM based PINN formulation, velocity penalties were enforced on the Lagrangian markers to satisfy the no-slip boundary condition on the immersed boundary. It was discussed that such an enforcement compensates for the unknown $\boldsymbol{f}$ and q terms in the solid region. Hence, inside the solid region, $\boldsymbol{f}$ and q are recovered as hidden variables given the velocity penalties are enforced on the immersed solid boundary. This is in contrast with the discrete forcing IBM formulation (Majumdar *et al.*, 2020; Kim *et al.*, 2001), where, $\boldsymbol{f}$ and q are computed in a set of sequential steps to satisfy the no-slip boundary condition. However, since $\boldsymbol{f}$ and q are known to be zero in the fluid domain, an additional data-driven constraint $\mathcal{L}_{IBvar}$ is formulated based on Huang *et al.* (2022), such that

$$\mathcal{L}_{IBvar} := \frac{1}{N_f} \sum_{i=1}^{N_{IBvar}} \left(\| f_x(\boldsymbol{x}^i_{IBvar}, t^i) \|^2_{L_2} + \| f_y(\boldsymbol{x}^i_{IBvar}, t^i) \|^2_{L_2} + \| q(\boldsymbol{x}^i_{IBvar}, t^i) \|^2_{L_2} \right), \quad (2.20)$$

where, $(\boldsymbol{x}^i_{IBvar}, t^i)$ for $i = \{1, \cdots N_{IBvar}\}$ is the set of points from the interior of the fluid domain Ω^r_f where the forcing and source/sink terms loss is evaluated. Thus, the data-driven loss formulation of MB-IBM-PINN is as follows

$$\mathcal{L}_{Data} := \lambda_{Bulk}\mathcal{L}_{Bulk} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{IC}\mathcal{L}_{IC} + \lambda_{IB}\mathcal{L}_{IB} + \lambda_{IBvar}\mathcal{L}_{IBvar}, \quad (2.21)$$

with λ_{IBvar} being the weighting coefficient of $\mathcal{L}_{IBvar}$.

The physics informed loss components $\mathcal{L}_{m_x}$, and $\mathcal{L}_{m_y}$ correspond to the x and y momentum residuals and $\mathcal{L}_c$ corresponds to the continuity residual, respectively, which are expressed as follows

$$\mathcal{L}_{m_x} := \frac{1}{N_{Res}} \sum_{i=1}^{N_{Res}} \|r_{m_x}(\boldsymbol{x}^i_{Res}, t^i)\|^2_{L_2}, \tag{2.22}$$

$$\mathcal{L}_{m_y} := \frac{1}{N_{Res}} \sum_{i=1}^{N_{Res}} \|r_{m+y}(\boldsymbol{x}^i_{Res}, t^i)\|^2_{L_2}, \quad \text{and} \tag{2.23}$$

$$\mathcal{L}_c := \frac{1}{N_{Res}} \sum_{i=1}^{N_{Res}} \|r_c(\boldsymbol{x}^i_{Res}, t^i)\|^2_{L_2}. \tag{2.24}$$

For MB-PINN, the following forms of the residuals are used

$$r_{m_x}(\boldsymbol{x}^i_{Res}, t^i) = u_t + uu_x + vu_y - p_x - 1/Re(u_{xx} + u_{yy}), \tag{2.25}$$

$$r_{m_y}(\boldsymbol{x}^i_{Res}, t^i) = v_t + uv_x + vv_y - p_y - 1/Re(v_{xx} + v_{yy}), \quad \text{and} \tag{2.26}$$

$$r_c(\boldsymbol{x}^i_{Res}, t^i) = u_x + v_y. \tag{2.27}$$

In MB-IBM-PINN, the momentum and continuity residuals are expressed as

$$r_{m_x}(\boldsymbol{x}^i_{Res}, t^i) = u_t + uu_x + vu_y - p_x - 1/Re(u_{xx} + u_{yy}) - f_x, \tag{2.28}$$

$$r_{m_y}(\boldsymbol{x}^i_{Res}, t^i) = v_t + uv_x + vv_y - p_y - 1/Re(v_{xx} + v_{yy}) - f_y, \quad \text{and} \tag{2.29}$$

$$r_c(\boldsymbol{x}^i_{Res}, t^i) = u_x + v_y - q, \tag{2.30}$$

where, $(\boldsymbol{x}^i_{Res}, t^i)$ for $i = \{1, \cdots N_{Res}\}$ is the set of points from the interior of the domain Ω^r where the governing equation residuals are evaluated to compute the physics loss $\mathcal{L}_{Phy}$.

In MB-PINN, solid points from Ω_s at any given temporal snapshot are discarded while computing the bulk data and physics losses. This is achieved by classifying the fluid-solid points based on the solid boundary location *a priori* as a data preprocessing step. Thus, the set of bulk data points $(\boldsymbol{x}^i_{Bulk}, t^i)_{i=1}^{N_{Bulk}}$ and residual collocation points $(\boldsymbol{x}^i_{Res}, t^i)_{i=1}^{N^{fluid}_{Res}}$, only from Ω^r_f are fed into the network while training for MB-PINN. It is to be noted that unless the solid boundary points are known *a priori*, it is not possible to classify

the domain as solid or fluid. Thus, MB-IBM-PINN involves a more general formulation based on the governing equations (2.5) and (2.6), where the whole snapshot domain for the residual loss, including Ω_s, is considered. For the pressure recovery problem, the boundary position and the velocity is known *a priori*. In such a scenario, it is of interest to understand the role of the solid region in the performance of MB-IBM-PINN and compare it with MB-PINN.

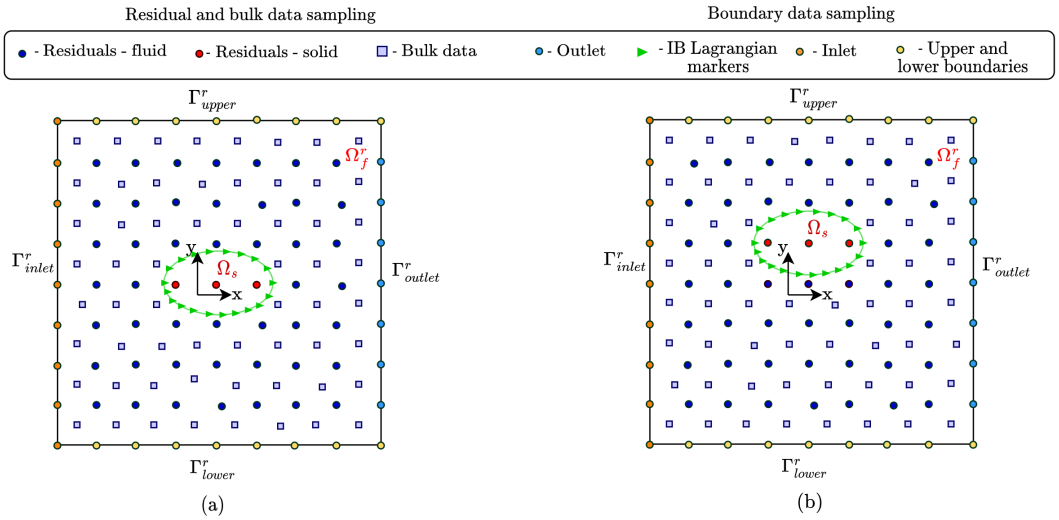


Figure 2.9: A schematic representing sampling of residual, bulk and boundary data points at two different time instants in (a) and (b). Although in the present work, bulk data points and fluid region residual points are sampled from same Eulerian grid, the bulk data points and fluid residual points are shown to be disjoint considering a more general case.

The bulk data and residual collocation points in the IBA framework come from an underlying Eulerian grid. Here, due to solid body motion, the solid region Ω_s and fluid region Ω^r_f are expected to change in time such that $\Omega^r = \Omega^r_f \cup \Omega_s$. Thus, the underlying Eulerian grid points in the region covered by the entire range of solid body motion, can either be found in Ω_s or in Ω^r_f at any time instant as shown in 2.9(a) and 2.9(b). Typically in experimental settings, one would only obtain sparse fluid region data from PIV measurements or smoke/dye visualizations. Keeping this in mind, the solid region data points from IBM simulation are excluded while computing $\mathcal{L}_{Bulk}$ for

MB-IBM-PINN. However, the solid region could still influence the physics loss when taken into account. Earlier works, including Huang *et al.* (2022); Calicchia *et al.* (2023), do not discuss the role of the solid region in the model performance. Thus, in order to determine the exact contribution from each region to the quality of predictions, a fluid-solid residual weighting is proposed in the present study for the MB-IBM-PINN formulation. Here, the residual losses are individually evaluated on the fluid and the solid regions at any time instant respectively with the corresponding loss weights.

Thus, the $\mathcal{L}_{Phy}$ for MB-IBM-PINN is decomposed as,

$$\begin{aligned}\mathcal{L}_{Phy} =& \lambda^{Phy}_{fluid}(\mathcal{L}^{fluid}_{m_x} + \mathcal{L}^{fluid}_{m_y} + \mathcal{L}^{fluid}_{c}) \\ &+ \lambda^{Phy}_{solid}(\mathcal{L}^{solid}_{m_x} + \mathcal{L}^{solid}_{m_y} + \mathcal{L}^{solid}_{c}),\end{aligned} \tag{2.31}$$

where, $\lambda^{Phy}_{\#}$, $\mathcal{L}^{\#}_{m_x}, \mathcal{L}^{\#}_{m_y}$, and $\mathcal{L}^{\#}_{c}$ correspond to the residual loss weighting coefficient, x and y momentum residuals and continuity residual evaluated in the respective # = {solid, fluid} regions. Moving forward, for simplicity, the superscript Phy shall be dropped, and the physics loss weights shall be simply referred to as λ_{fluid} or λ_{solid} for the fluid and solid region, respectively. Hence, in this case, two sets of residual collocation points, one from Ω^r_f given by $(\boldsymbol{x}^i_{Res}, t^i)^{N^{fluid}_{Res}}_{i=1}$, and one from region Ω_s given by $(\boldsymbol{x}^j_{Res}, t^j)^{N^{solid}_{Res}}_{j=1}$, are needed for the evaluation of the physics losses. This allows one to sample different proportions of collocation points from each region. Additionally, the relative weighting allows balancing of the relative contributions from the fluid and the solid regions. This is keeping in mind that density of the points in a particular region is equivalent to weighting the loss locally in that region (Wu *et al.*, 2023). To make a consistent comparison with the MB-IBM-PINN model and given that MB-PINN residuals are evaluated only in the fluid region, its $\mathcal{L}_{Phy}$ can be expressed in terms of # = fluid region such that

$$\mathcal{L}_{Phy} := \lambda_{fluid}(\mathcal{L}^{fluid}_{m_x} + \mathcal{L}^{fluid}_{m_y} + \mathcal{L}^{fluid}_{c}). \tag{2.32}$$

In every mini-batch of collocation points corresponding to $\mathcal{L}_{Bulk}$ and $\mathcal{L}_{Phy}^{fluid}$ from the same fluid region Ω_f^r are mostly disjoint from each other due to random sampling. However, there still exists some overlap since the underlying grid from which these points are sampled is kept the same in this study. Notably, given that PINNs are essentially meshless and there is freedom to select points from anywhere in the domain for the physics losses, one could also sample from a different grid than the Eulerian grid in order to keep the residual and the bulk data points truly disjoint (Wu *et al.*, 2023). This allows finer control over the sampling methodology and sampling levels for each loss component individually. The proportion and refinement levels for each loss component can also be varied for each individual component thereby implicitly weighting the loss components (Wu *et al.*, 2023). Moreover, each individual component can be tracked and their weights can be updated adaptively, as reported in the works of Wang *et al.* (2021*a*); Jin *et al.* (2021); Wu *et al.* (2023). Adaptive methods have not been investigated in the present study to focus on understanding the effect of systematic variations of loss weights and data sampling. Each loss component was monitored independently to understand how every component evolves with iterations.

The details of database generation for training and testing the models and hyperparameter tuning are discussed in section 2.6. The numerical settings considered for MB-PINN and MB-IBM-PINN along with the baseline results, and computational strategies considered to further improve the baseline performance are discussed in sections 2.7 and 2.8.

2.6 PRE-SETUP CONSIDERATIONS FOR PINN

2.6.1 Training and testing database generation

Training and testing data for the velocity-field have been obtained from the IBM solver described in section 2.3. In this regard, the flow around the plunging airfoil was simulated at $Re = 500$, $k = 2\pi$ and $h = 0.16$ corresponding to $kh = 1.0$. This resulted in a periodic flow-filed with a reverse Kármán vortex street in the wake of the flapping foil. An ALE

based OpenFOAM solver was employed to generate the test data set for pressure, as was mentioned in section 2.3. From the recent works of Jagtap *et al.* (2020*c*); Krishnapriyan *et al.* (2021); Mattey and Ghosh (2022); Penwarden *et al.* (2023) it is known that PINNs suffer from the issue of propagation failures, leading to training difficulties on large spatial/temporal domains. Moreover, earlier works such as (Lai and Platzer, 1999; Platzer *et al.*, 2008; Bose and Sarkar, 2018; Majumdar *et al.*, 2022) have shown that the key flow features that largely dictate the dynamics and the aerodynamic loads in flapping problems are strongest in the vicinity of the body.

For the considered flapping kinematic parameters, the resulting flow is periodic (Khalid *et al.*, 2018), making the vortex structures and their interactions repeat in time. Keeping these in mind, to setup the training and testing databases, a truncated spatial domain (figure 2.1) is chosen such that $(x, y) \in [-1c, 3.5c] \times [-1c, 1c]$ ensuring that at least two trailing-wake couples are contained in the truncated domain along with the flapping foil. The time domain is restricted to two plunging cycles for $t/T > 25.0$ and normalized such that $t/T \in [0, 2]$ with a snapshot sampling interval of $\Delta t/T = 0.025$. Here, the snapshot interval is chosen such that the formations of the leading-edge vortex (LEV) and the trailing-edge vortex (TEV) are captured smoothly within one time period of vortex shedding. Note that the transient effects in the flow-field are over by $t/T > 25.0$ and the vortex shedding pattern is stabilized.

The high-resolution IBM velocity data from a non-uniform grid is interpolated onto a coarse uniform grid to bring down the quality of the training data. This serves as a rigorous test of the IBA surrogate since the coarse grid is neither embedded in the IBM nor the ALE grid.

Thus, three benchmark data sets are generated from the CFD simulations, (i) reference IBM velocity data (Ref-IBM), (ii) reference ALE pressure data (Ref-ALE) and (iii) coarsened and interpolated (CI) velocity data. In the case of CI data set, using linear

interpolation, the flow-field data from the IBM solution are interpolated onto a coarse grid containing 270×120 spatial grid points with a uniform spatial resolution (figure 2.10(a)), approximately 4 times lower near the solid boundary than the original IBM grid. However, the resolution of CI grid is sufficient such that the key vortex structures are not lost. Secondly for training, the time interval $\Delta t/T = 0.05$ is chosen such that temporal resolution is two times coarser as compared to Ref-IBM. Different spatial resolutions of CI, Ref-IBM, and Ref-ALE are shown in figures 2.10(b)-(d), respectively. The details of the spatio-temporal resolution of these data sets are summarized in table 2.1. These data sets correspond to the truncated spatio-temporal domain and not the entire domain considered for CFD simulations. The training of the models is performed solely based on the velocity data of CI. The velocity-field reconstructions are compared with Ref-IBM, whereas the pressure recovered by the PINN models is compared with Ref-ALE.

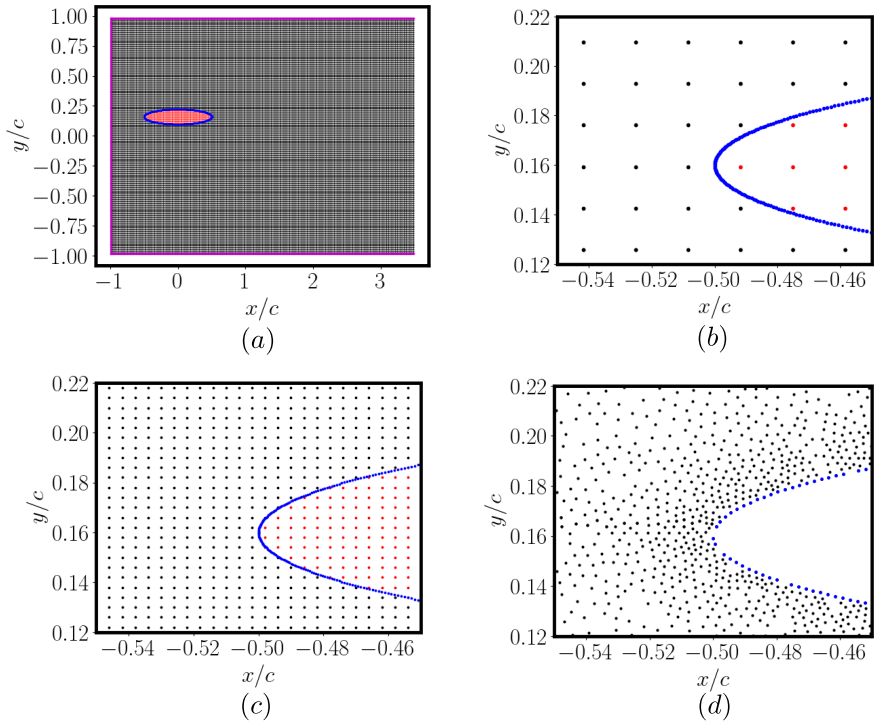


Figure 2.10: (a) Spatial grid of CI data set with the Lagrangian markers in blue, inlet and wall boundary points in magenta, and the solid region in red colours, respectively. Zoomed view of the spatial grid for (b) CI, (c) Ref-IBM and (d) Ref-ALE presents the comparative spatial resolution near the solid boundary.

Table 2.1: Training and testing data set resolution within the truncated domain considered in this study.

Data sets	N_x	N_y	N_t	$\Delta t/T$	$(Nx \times Ny \times N_t)$
Ref-IBM	651	500	81	0.025	2.6365e07
Ref-ALE	-	-	81	0.025	4.05e06
CI	270	120	41	0.05	**1.3284e06**

Among the total $N_x \times N_y \times N_t = 1.3284e06$ data points in CI, $N_{Bulk} = 1.2915e06$ number of points were used to calculate $\mathcal{L}_{Bulk}$ for training after discarding the solid region. This N_{Bulk} is just $S_{Data} = \frac{N_{Bulk}}{N_{Ref}} \times 100 = 4.89\%$ of the $N_{Ref} = N_x \times N_y \times N_t$ flow-field data points available in the Ref-IBM data set. Here, in CI data set, the boundary points resolution is such that $N_{IB} = 1000$, $N_{wall} = 540$, and $N_{inlet} = 120$. At the initial time stamp $t/T = 0$, $N_{IC} = 31500$ flow field data points are sampled discarding the solid region. Here, the residual collocation points are sampled from the CI grid such that there are $N_{res}^{fluid} = 1.2915e06$ and $N_{res}^{solid} = 1.476e04$ spatio-temporal points in the fluid and solid region respectively. Throughout the study, the number of boundary points and the number of residual collocation points are not varied.

By choosing the grid as in CI, there is a significant reduction in the number of training data points that encompass the flow-field information. This is in fact similar to a super-resolution setup with reconstruction of the flow-field from a coarse data (Fukami *et al.*, 2019). It is noted that data coarseness is often more pronounced in the super-resolution studies which might also have a bearing on the prediction accuracy. In the current study, we attempt to train the proposed PINN models only based on velocity data from the coarse CI data set to predict the velocity and pressure to be validated against Ref-IBM and Ref-ALE data sets, respectively. In other words, a data reconstruction problem coupled with hidden variable recovery is at hand. It is worth noting that the pressure data is not used while training throughout the study and is only recovered as a hidden variable.

2.6.2 Evaluation metrics

In the present study, the true velocity data from Ref-IBM data $\{(u(\boldsymbol{x}^i_{Fluid}, t^i), v(\boldsymbol{x}^i_{Fluid}, t^i))\}_{i=1}^{N_{Fluid}}$ and pressure data $p(\boldsymbol{x}^i_{Fluid}, t^i)_{i=1}^{N_{Fluid}}$ are available as testing data. The trained models are then used to predict velocity $\{(\hat{u}(\boldsymbol{x}^i_{Fluid}, t^i), \hat{v}(\boldsymbol{x}^i_{Fluid}, t^i))\}_{i=1}^{N_{Fluid}}$ on the Ref-IBM grid and pressure on Ref-ALE grid $\{\hat{p}(\boldsymbol{x}^i_{Fluid}, t^i)\}_{i=1}^{N_{Fluid}}$, respectively. Here, the subscript "$Fluid$" simply refers to the fact that data at only those spatio-temporal locations are considered which are in the fluid region to evaluate the error measures.

To evaluate the model performance, different error metrics are considered. Here, the mathematical details of the building blocks of the error measures are presented here in the context of x component of the true velocity data $\hat{u}(\boldsymbol{x}^i_{Fluid}, t^i)_{i=1}^{N_{Fluid}}$ and corresponding model predicted data $u(\boldsymbol{x}^i_{Fluid}, t^i)_{i=1}^{N_{Fluid}}$. But the method of calculation apply to v and p as well.

So given $u(\boldsymbol{x}^i_{Fluid}, t^i)_{i=1}^{N_{Fluid}}$ and $\hat{u}(\boldsymbol{x}^i_{Fluid}, t^i)_{i=1}^{N_{Fluid}}$, the root mean squared error (RMSE) or the L_2 error, and overall relative RMSE (rRMSE) evaluated over all time stamps are given by

$$\text{RMSE}_u = \frac{1}{N_{Fluid}} \sum_{i=1}^{N_{Fluid}} (\|u(\boldsymbol{x}^i_{Fluid}, t^i) - \hat{u}(\boldsymbol{x}^i_{Fluid}, t^i)\|_{L_2}) \tag{2.33}$$

$$\text{rRMSE}_u = \frac{1}{N_{Fluid}} \sum_{i=1}^{N_{Fluid}} \frac{(\|u(\boldsymbol{x}^i_{Fluid}, t^i) - \hat{u}(\boldsymbol{x}^i_{Fluid}, t^i)\|_{L_2})}{\frac{1}{N_{Fluid}} \sum_{i=1}^{N_{Fluid}} \|\hat{u}(\boldsymbol{x}^i_{Fluid}, t^i)\|_{L_2}} \tag{2.34}$$

The mean absolute error (MAE) is given by

$$\text{MAE}_u = \frac{1}{N_{Fluid}} \sum_{i=1}^{N_{Fluid}} (\|u(\boldsymbol{x}^i_{Fluid}, t^i) - \hat{u}(\boldsymbol{x}^i_{Fluid}, t^i)\|_{L_1}) \tag{2.35}$$

Here, to additionally estimate how well the model predicts the outputs, coefficient of determination (R^2) (Draper and Smith, 1998) is considered which is given by

$$R^2 = 1 - \frac{\sum_{i=1}^{N_{Fluid}} (u(\boldsymbol{x}^i_{Fluid}, t^i) - \hat{u}(\boldsymbol{x}^i_{Fluid}, t^i))^2}{\sum_{i=1}^{N_{Fluid}} (\hat{u}(\boldsymbol{x}^i_{Fluid}, t^i) - \bar{\hat{u}}(\boldsymbol{x}^i_{Fluid}, t^i))^2} \tag{2.36}$$

In the present study, the above mentioned error measures are first computed for velocity reconstruction over Ref-IBM dataset and for pressure recovery against the Ref-ALE dataset, respectively. The individual velocity component and pressure error measures are then combined to obtain the averaged error measures represented by the prefix "a." As an example, the averaged RMSE (aRMSE) is demonstrated where

$$\text{aRMSE} = \frac{(\text{RMSE}_u + \text{RMSE}_v + \text{RMSE}_p)}{3} \tag{2.37}$$

The above mentioned logic of the prefix "a" applies to the other error quantities described in equations (2.34)-(2.36) as well.

In addition, snapshot wise relative root mean squared errors (rRMSE), and the contours of normalized point-wise normalized absolute errors are presented to determine the temporal and spatial error behavior, respectively. Here, the point-wise absolute errors are normalized with respect to the maximum absolute value of the corresponding flow-field component. Moreover, the capability of the proposed surrogate models in accurately resolving the vorticity field is investigated as well. This is crucial in the context of flow past flapping wings where the vortex structures play an important role in dictating the dynamics.

2.6.3 Hyper parameter tuning

To determine an appropriate number of hidden layers and hidden neurons, the underlying FNN backbone (now referred as MB-FNN) was considered. Here, MB-FNN was trained on the CI data set to reconstruct velocity fields alone in a purely data-driven manner without considering the physics losses. Here, the weights of the data-driven components were unity, except for λ_{IC} which was set to 0. This is because data snapshots are available at all time instances, and a separate loss component for initial condition would only be redundant and only be a competing objective (Wang *et al.*, 2021*a*) in the loss function.

The number of hidden layers and hidden neurons were chosen after a grid search on

combinations of $L = [6, 8, 10]$ layers and $n_l = [60, 80, 100]$ hidden neurons, respectively. The resultant models were evaluated on the basis of velocity reconstruction alone on the Ref-IBM data set (accuracy details presented in table 2.2). It was observed that with the converged $L = 10$ hidden layers and $n_l = 100$ hidden neurons, excellent relative RMSE of 0.51% in $x-$component of velocity u and 1.31% in $y-$component of velocity v were obtained (see table 2.3). This shows that MB-FNN is quite expressive. Throughout the study, these hyper-parameters are kept the same for both MB-PINN and MB-IBM-PINN models as well.

For the MB-PINN model with $L = 10$ layers and $n_l = 100$ hidden neurons, $\theta = 9.1603\text{e}04$ parameters need to be calibrated while training. Whereas, due to three additional outputs in MB-IBM-PINN, $\theta = 9.1906\text{e}04$ parameters need to be calibrated while training. Unless specified, a maximum computational budget of $5e05$ iterations per training cycle in learning rate steps of $[1e-03, 5e-04, 1e-04]$ has been considered with a mini-batch size of $1.5e03$, throughout the study. Training of all the models has been carried out on a single NVIDIA Tesla A100 GPU card with 40GB memory and 6912 CUDA cores. Whereas, testing has been carried out on a desktop grade Quadro Pascal P2200 card with 5GB memory and 1280 CUDA cores. This is to investigate whether a comparable performance from MB-PINN and MB-IBM-PINN can be obtained with the same architecture and computational budget, for not only velocity reconstruction but also pressure recovery as a hidden variable. The multi-part physics loss weighting studies have been carried out to answer this question.

Table 2.2: Accuracy details for the MB-FNN models (without any physical constraints) for different number of hidden layers L and hidden neurons per layer n_l. The model is tested on the Ref-IBM data set.

L	n_l	Accuracy			
		aRMSE	aMAE	aR^2	arRMSE
6	60	1.74e-02	7.9e-03	9.9929e-01	2.47
	80	1.455e-02	6.3e-03	9.9947-01	1.26
	100	1.34e-02	5.55e-03	9.9956e-01	1.16
8	60	1.17e-02	5.15e-03	9.9967e-01	1.67
	80	7.9e-03	3.25e-03	9.9984e-01	1.13
	100	6.75e-03	2.55e-03	9.9987e-01	0.98
10	60	9.75e-05	4.44e-03	9.9976e-01	1.4
	80	7.01e-03	2.8e-03	9.9987e-01	1.01
	100	**6.15e-03**	**2.2e-03**	**9.9989e-01**	**0.91**

Table 2.3: Velocity component wise accuracy of the best MB-FNN model. The model is tested on the Ref-IBM data sets.

Model	Accuracy			
MB-FNN	RMSE	MAE	R^2	rRMSE
u	5.8e-03	2.1e-03	9.9997e-01	0.51
v	6.5e-03	2.3e-03	9.9981e-01	1.31

2.7 HIDDEN PRESSURE RECOVERY AND THE ROLE OF MULTI-PART PHYSICS LOSS WEIGHTING

As stated earlier in the Introduction Chapter 1, hidden pressure recovery in a non-intrusive fashion using sparse velocity field data is valuable for studying flow past moving bodies. To obtain a good model performance on hidden pressure recovery under

a fixed computational budget, an efficient strategy could be relaxing the physics loss components (Buhendwa *et al.*, 2021; Lucor *et al.*, 2022), or inversely, weighting the data-driven loss components higher than the physics counterpart (Wang *et al.*, 2021*a*; Nguyen *et al.*, 2023; Jin *et al.*, 2021). The former approach has been considered in the present study, where the weights of all the data-driven components are taken to be unity and only the physics loss components are relaxed by lowering their appropriate weighting coefficients. Dynamic weighting strategies have been proposed in some of the recent studies by Wang *et al.* (2021*a*); Jin *et al.* (2021); Shukla *et al.* (2022); Heydari *et al.* (2019); Bischof and Kraus (2025). In the present study, a manual tuning strategy of the residual loss weights is adopted to understand their contributions toward pressure recovery in a systematic manner.

To recall from section 2.5, data sets at the solid region are discarded for computing both $\mathcal{L}_{Bulk}$ and $\mathcal{L}_{Phy}$ in MB-PINN, thus only λ_{fluid} is in consideration. For the MB-IBM-PINN results in this section, $\mathcal{L}_{Bulk}$ computation is the same as in MB-PINN, *i.e.*, only the fluid region data points are considered. But the solid region is taken into account in $\mathcal{L}_{Phy}$ computation along with the fluid region, with λ_{solid} and λ_{fluid} as respective weights (see equation (2.31)). In the present study, a multi-part physics loss weighting strategy thus refers to the fluid-solid partitioning of the physics loss and relative weighting between them as discussed in section 2.5. This choice of this fluid-solid partitioning is done to better quantify the impact of the solid region residual losses. As a result, there is an extra hyperparameter, λ_{solid}, to vary in MB-IBM-PINN.

In the beginning, without any loss balancing (*i.e.* $\lambda_{fluid} = \lambda_{solid} = 1.0$) it was observed that the predictions from both MB-PINN and MB-IBM-PINN were significantly worse, with the snapshot wise relative errors being more than 20% for velocity reconstruction (figure 2.11(a)-(b)), and over 30% for pressure recovery (figure 2.11(c)). Thus, reasonable baseline models for MB-PINN and MB-IBM-PINN were first obtained with an overall mean relative error, given by rRMSE, below 10% for velocity reconstruction, and below

20% for pressure recovery. This was achieved in the case of $\lambda_{fluid} = 0.1$ for MB-PINN. To match the MB-PINN's performance, for MB-IBM-PINN with the same $\lambda_{fluid} = 0.1$, a relatively lower $\lambda_{solid} = 0.001$ was required, which suggests the need of undermining the solid region loss. In further discussions, the MB-PINN and MB-IBM-PINN models with $\lambda_{fluid} = 0.1$, and $\lambda_{solid} = 0.001$, shall be referred to as the baseline cases. In the relative error plots (figure 2.11), the MB-PINN and MB-IBM-PINN curves almost overlap each other except at a few later time instances, where MB-IBM-PINN does slightly worse for the selected baseline λ_{fluid} and λ_{solid} values. The equivalence of MB-PINN and MB-IBM-PINN is also confirmed by the similar order of the convergence of the overall training and individual loss components $\mathcal{L}_{Bulk}$, scaled $\mathcal{L}_{Phy}$, $\mathcal{L}_{IC}$, $\mathcal{L}_{inlet}$, and $\mathcal{L}_{wall}$ in figures 2.12(a) and 2.12(b), respectively. Note that, since the bulk data points are available very close to Γ^r_{lower} and Γ^r_{upper} that too across all time, the L_{wall} and $\mathcal{L}_{IC}$ constraints are not enforced additionally. However, the convergence of these loss components is still satisfactory.

The true and predicted velocity and pressure contours are compared in figures 2.13(a) and 2.13(b). The corresponding maximum value normalized point-wise absolute error contours in the near-field region are shown in figure 2.13(c) for a test time stamp $t/T = 0.375$, which is previously unseen during the training. At this time stamp, the velocity and pressure fields are queried on high-resolution Ref-IBM and Ref-ALE data sets test grids, respectively. It can be seen in figure 2.13(c) that the errors are highly pronounced in the region where there is an LEV. This is expected as strong gradients of velocity and pressure exist in such regions of strong vorticity. The issue of difficulty of PINN in capturing strong/sharp gradient has also been reported in the earlier literature (Wang *et al.*, 2021*a*; Dwivedi *et al.*, 2021; Jagtap *et al.*, 2020*c*; Jagtap and Karniadakis, 2020). For the test time $t/T = 0.375$, a characteristic strong LEV on the upper surface of the foil followed by a secondary vortex structure in the mid-region of the upper surface is observed (see figure 2.14(a)). The baseline models are able to resolve the primary vortex structures very well, while the secondary structure is still

blurred in the predictions as seen in figures 2.14(b) and 2.14(c). With regard to the pressure recovery, although the baseline results seem promising, the errors are still high. In order to see if larger training iterations (N_{iter}) per cycle could improve the results, in another set of experiments with $\lambda_{fluid} = 0.1$ and $\lambda_{solid} = 0.001$ (see table 2.4 for accuracy details), it is seen that the improvements are not too significant considering the increased computational budget. It remains to be seen if further relaxation of the residual losses can improve the pressure recovery and the overall model performance.

Table 2.4: Accuracy details of the MB-PINN and MB-IBM-PINN models under the effect of larger N_{iter}.

Model	N_{iter}	Accuracy			
		aRMSE	aMAE	aR^2	arRMSE
MB-PINN ($\lambda_{fluid} = 0.1$)	7.5e05	5.64e-02	3.02e-02	9.95e-01	5.39
	1e06	5.24e-02	2.79e-02	9.96e-01	5.01
MB-IBM-PINN ($\lambda_{fluid} = 0.1$, $\lambda_{solid} = 0.001$)	7.5e05	8.79e-02	4.87e-02	9.87e-01	8.32
	1e06	8.19e-02	4.54e-02	9.89e-01	7.74

The detailed error measures for the relaxation studies are presented in table 2.5. In the case of MB-PINN, there is a significant decrease in the overall error measures as λ_{fluid} is decreased from 1 to 0.1. However, beyond $\lambda_{fluid} = 0.001$, the pressure recovery breaks down as seen in table 2.5. It is observed that at $\lambda_{fluid} = 0.0001$ the snapshot-wise relative errors (see figure 2.15) are either as good as the baseline case or even worse in certain situations. Thus the most optimal results have been obtained for $\lambda_{fluid} = 0.001$ in the case of MB-PINN, with arRMSE = 3.22% which can be considered as the best performing MB-PINN model within the present set of experiments.

For MB-IBM-PINN, even for a fixed λ_{fluid}, the prediction accuracy improves overall when the λ_{solid} is lowered. This is observed for different λ_{fluid} values as shown in table 2.5. To obtain a reasonable accuracy in velocity reconstruction and pressure recovery, indicated by arRMSE, λ_{solid} is expected to be at least one order lower than

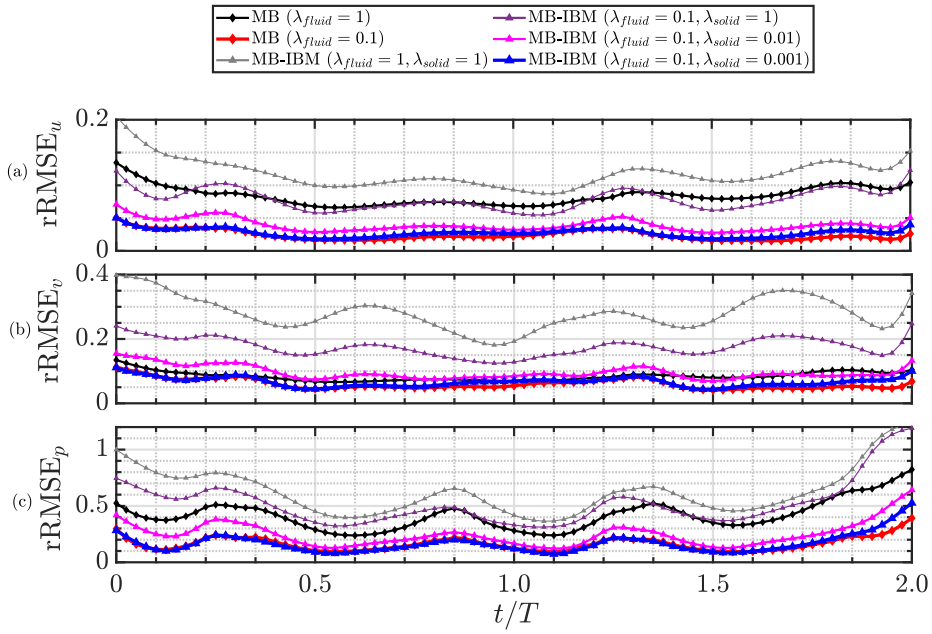


Figure 2.11: Relative errors (rRMSE) in time for (a, b) x and y velocity components with respect to Ref-IBM data set, and (c) for recovered pressure with respect to Ref-ALE data set. Here, MB-PINN (represented as MB) and MB-IBM-PINN (represented as MB-IBM) models are compared, considering $\lambda_{fluid} = 1$ and $\lambda_{fluid} = 0.1$.

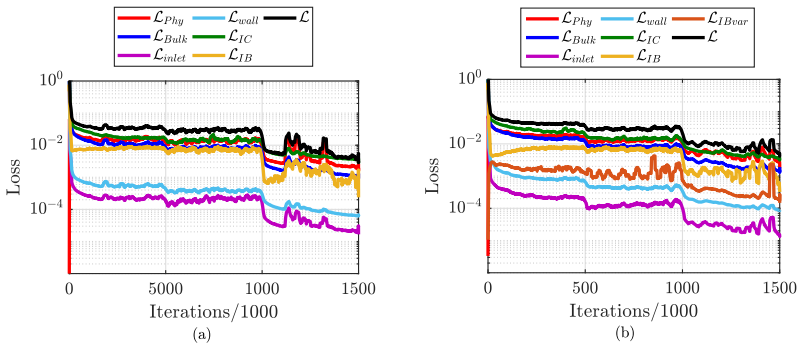


Figure 2.12: Smoothened individual loss component convergence shown for the baseline (a) MB-PINN with $\lambda_{fluid} = 0.1$ and (b) MB-IBM-PINN with $\lambda_{fluid} = 0.1, \lambda_{solid} = 0.001$.

that of λ_{fluid}. This is true for the baseline $\lambda_{fluid} = 0.1$, with $\lambda_{solid} = 0.001$ that is two orders lower, where the accuracy of the model almost matches MB-PINN model for same λ_{fluid} as shown in the relative error plots in figures 2.11(a)-(c). This requirement

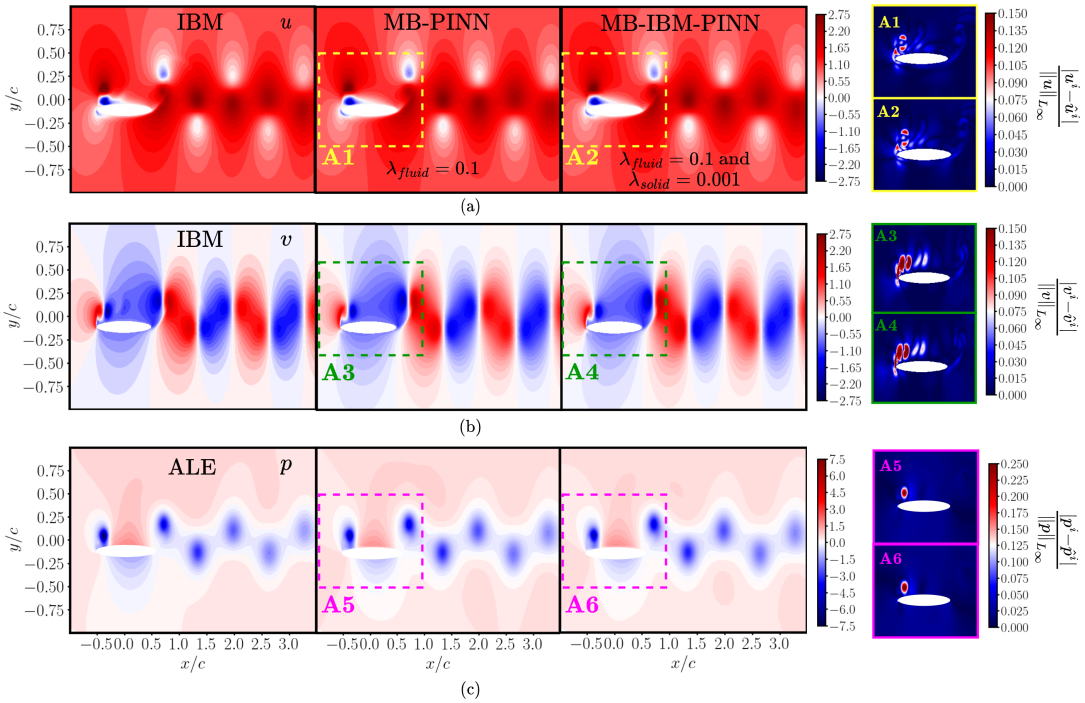


Figure 2.13: Comparison of true and predicted (a, b) velocity and (c) pressure snapshots queried on the Ref-IBM and Ref-ALE grids, respectively, for a test time stamp t/T = 0.375. The inshot boxes **A#** for # = $1, 2, \cdots 6$ correspond to the near-field region, where, the normalized point-wise absolute errors in the flow-field are prominent.

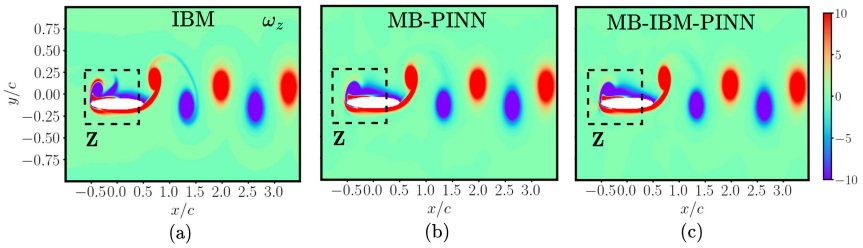


Figure 2.14: Comparison of (a) IBM obtained vorticity snapshot with the predictions of (b) baseline MB-PINN (λ_{fluid} = 0.1) and (c) MB-IBM-PINN (λ_{fluid} = 0.1, λ_{solid} = 0.001) for a test time stamp t/T = 0.375. The inshot boxes **Z** correspond to the near-field region surrounding the LEV and the secondary vorticity structure.

could be because relaxation is also implicitly related to choosing a relatively lower proportion of collocation points in the region (Wu *et al.*, 2023). This is not applicable in the case of MB-PINN since the solid region is not included in the formulation. By fixing

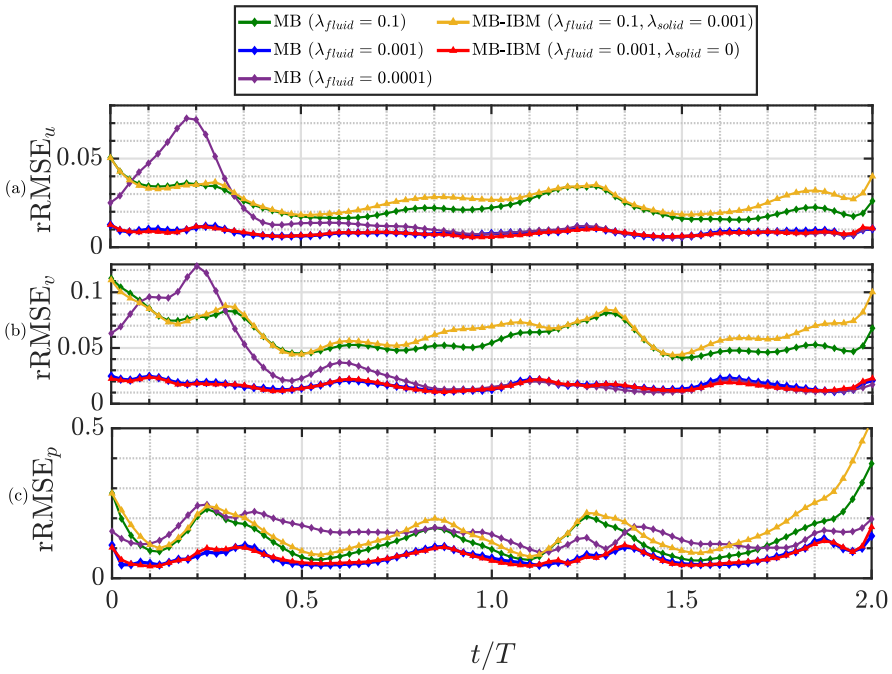


Figure 2.15: Relative errors (rRMSE) in time for (a,b) x and y velocity components, and (c) pressure p predicted by MB-PINN (represented as just MB) and MB-IBM-PINN (represented as MB-IBM) models with different λ_{fluid} and λ_{solid} combinations depicting the effect of further residual relaxation.

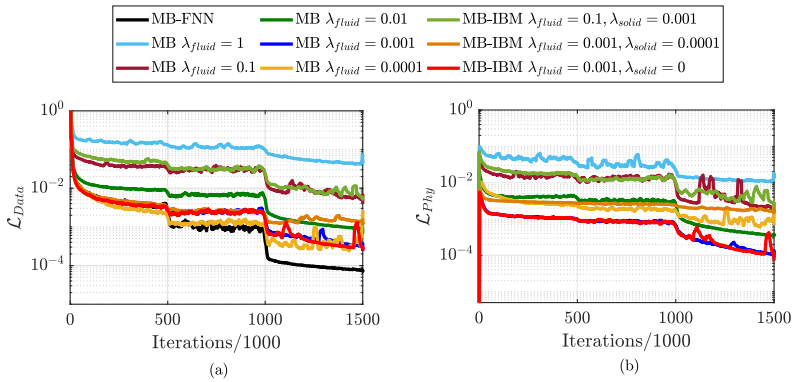


Figure 2.16: Overall effect of relaxation on loss convergence studied for MB-PINN and MB-IBM-PINN models, considering different λ_{fluid} and λ_{solid} values. The combined data-driven losses $\mathcal{L}_{Data}$ are presented in comparison with the MB-FNN model in (a), and the scaled physics-informed losses $\mathcal{L}_{Phy}$ in (b), respectively.

λ_{fluid} = 0.001 as in the optimal MB-PINN model, the percentage arRMSE blows up even at λ_{solid} = 0.0001 and 0.00001. Given that a significantly lower λ_{solid} is indeed required

to improve the accuracy further, the need for solid region losses in MB-IBM-PINN is hence investigated by choosing a limiting $\lambda_{solid} = 0$ keeping $\lambda_{fluid} = 0.001$. In this case, the MB-IBM-PINN model gives the best performance, matching that of MB-PINN. The arRMSE drastically drops as seen in table 2.5.

It can also be seen that for the optimal MB-PINN and MB-IBM-PINN models, $\mathcal{L}_{Data}$ (figure 2.16(a)) comes closer to that of purely data-driven MB-FNN model. Whereas, the scaled $\mathcal{L}_{Phy}$ are the lowest for both variants of PINN at the optimal values of relaxation (see figure 2.16(b)). The relative error in time for velocity reconstruction and pressure recovery are plotted across different selected levels of λ_{fluid} and λ_{solid} for both PINN models in figures 2.15(a)-(c). It is seen that for MB-IBM-PINN, the relative errors in velocity reconstruction and pressure recovery closely follow the optimal MB-PINN model at all time instants for $\lambda_{fluid} = 0.001$ and $\lambda_{solid} = 0$.

Table 2.5: Accuracy of MB-PINN and MB-IBM-PINN for different relaxation coefficients. Best performing models are highlighted in bold-faced fonts.

Model			Accuracy			
	λ_{fluid}	λ_{solid}	aRMSE	aMAE	aR^2	arRMSE (in %)
MB-PINN	1	-	2.38e-01	1.37e-01	9.23e-01	22.96
	0.1	-	7.64e-02	4.09e-02	9.91e-01	7.27
	0.01	-	4.09e-02	2.28e-02	9.97e-01	3.78
	0.001	-	**3.58e-02**	**2.07e-02**	**9.98e-01**	**3.22**
	0.0001	-	7.59e-02	4.87e-02	9.91e-01	6.56
	λ_{fluid}	λ_{solid}	aRMSE	aMAE	aR^2	arRMSE (in %)
MB-IBM-PINN	1	1	0.3.61e-01	2.22e-01	8.24e-01	34.11
	1	0.1	2.60e-01	1.52e-01	9.04e-01	24.81
	1	0.01	2.25e-01	1.32e-01	9.28e-01	21.56
	1	0.001	2.08e-01	1.19e-01	9.41e-01	19.99
	0.1	1	2.83e-01	1.71e-01	8.85e-01	25.76
	0.1	0.1	1.76e-01	9.94e-02	9.54e-01	16.36
	0.1	0.01	1.35e-01	7.60e-02	9.73e-01	12.66
	0.1	0.001	9.14e-02	5.07e-02	9.87e-01	8.63
	0.01	0.001	1.22e-01	7.09e-02	9.61e-01	10.88
	0.01	0.0001	9.53e-02	5.21e-02	9.87e-01	8.45
	0.001	0.001	4.15e-01	2.69e-01	7.27e-01	32.41
	0.001	0.0001	3.27e-01	2.078e-01	8.22e-01	25.91
	0.001	0.00001	2.71e-01	1.719e-01	8.76e-01	21.02
	0.001	**0**	**3.59e-02**	**2.03e-02**	**9.98e-01**	**3.21**

It is thus seen that MB-IBM-PINN can match the performance of MB-PINN under certain combinations of λ_{fluid} and λ_{solid}, under the same training budget and network architecture. At optimal λ_{fluid} and for $\lambda_{solid} = 0$, the solid region points can be effectively discarded in MB-IBM-PINN, for equivalent performance to MB-PINN. This is because the no-slip boundary condition in both the PINN variants is directly enforced through $\mathcal{L}_{IB}$, rather than satisfied indirectly as in IBM (see section 2.3), where the solid region points are required. This also suggests that MB-IBM-PINN might not be necessary when one is already aware of the solid body's position and velocity *a priori*. Moving forward, MB-PINN can be used efficiently for pressure recovery from velocity data when the solid body position and velocity is known *a priori*.

Note that the surrogate model in the present study is able to handle the immersed moving boundary using the Lagrangian markers and the fluid using an Eulerian grid, respectively. The surrogate model with residual relaxation performs very well for queries on the high-resolution grids and additionally recover pressure as a hidden variable. However, the number of data points used are still large even with the coarse CI data sets with about 1.328e06 data points in total. Therefore, given the localized features of the flow-field, the possibility of a physics based sampling of the data points is explored in the next section.

2.8 IMPROVING DATA EFFICIENCY THROUGH PHYSICS-BASED DATA SAMPLING

As mentioned earlier, capturing strong/sharp gradients may become challenging in PINNs (also reported in various works such as Wang *et al.* (2021*a*); Krishnapriyan *et al.* (2021); Dwivedi *et al.* (2021); Jagtap *et al.* (2020*c*); Jagtap and Karniadakis (2020)), and hence, physics constraint relaxation was explored in section 2.7. In the baseline MB-PINN model, the secondary vortex structure was not resolved accurately while also smudging the LEV as shown in figure 2.14. This was also reflected in the locally high point-wise errors in figures 2.13 (marked in rectangular boxes). These errors were

localized around the regions of strong LEVs and TEVs which were representative of strong velocity gradients in the flow-field. To mitigate this, many authors such as Wu *et al.* (2023); Nguyen *et al.* (2023); Gao *et al.* (2023); Peng *et al.* (2022); Tang *et al.* (2023), have proposed various adaptive sampling strategies, albeit with some additional computational overheads. Noting that the sampling of spatial points based on a prior distribution is equivalent to point-wise weighting, Wu *et al.* (2023) dealt with adaptive sampling techniques for PDEs in three dimensions, while Tang *et al.* (2023) proposed a deep adaptive sampling based PINNs (DAS-PINNs) for higher dimensional PDEs.

However, the above approaches entail iterative sampling of newer residual collocation points (and not data points) to mitigate the training difficulty. Recently, Gopakumar *et al.* (2023) showed how the loss landscape of PINN becomes significantly smoother in the presence of very sparse/coarse simulation/experimental data in the bulk, as compared to no data, thereby making PINNs easier to train. In the present study, a data-driven approach was thus adopted. In fact, high number of data points were also used in the optimal MB-PINN model identified in the previous section. To improve data-efficiency and reduce associated data storage requirement, the possibility of using the underlying features of the flow-field data to train the network for MB-PINN is explored in this section.

Table 2.6: Data sets generated through vorticity cutoff based sampling.

Data set	S_{ω_z}	$S_{\omega_z}^{NF}$	N_{Bulk}	N_{res}	S_{Data}
CI	100	100	1.2915e06	1.3062e06	4.89
CI-S50	50		7.5112e05		2.85
CI-S10	10		3.1885e05		1.21
CI-S5	5		2.6481e05		1.00
CI-S1	1		2.2158e05		0.84
CI-S0	0		2.1078e05		0.79
CI-S0-US10	0	10	2.1074e04		0.08

Vorticity magnitude levels are good indicators of strong velocity gradients and the current study proposes a data efficient vorticity cutoff based undersampling approach. Here a

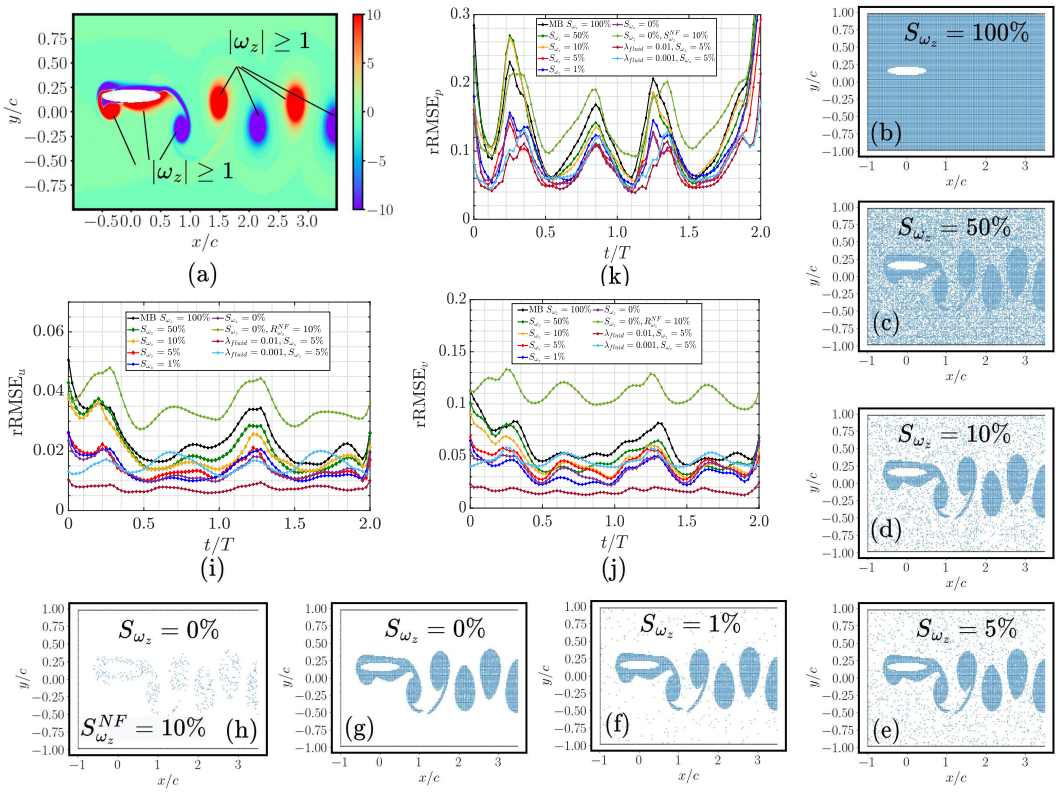


Figure 2.17: Vorticity cutoff based undersampling strategy: (a) the selection of strong and weak vorticity regions is shown for a representative vorticity snapshot at $t/T = 0.0$, which is characterised by a LEV at the bottom surface, (b)-(h) vorticity cutoff based undersampled data locations corresponding to the data sets in table 2.6, (i, j) relative errors in velocity reconstruction, and (k) relative error in recovered pressure.

vorticity cutoff ω_z^* is chosen such that the key features can be retained locally while the regions of low vorticity are undersampled. The vorticity fields are computed from the velocity data using a second-order central-difference scheme. Once the vorticity fields are obtained, a reasonable absolute vorticity cutoff value of $|\omega_z^*| = 1.0$ is chosen. For each vorticity snapshot, only those $N_{|\omega_z| \geq |\omega_z^*|}$ fluid points corresponding to $|\omega_z| \geq |\omega_z^*|$ are first retained, which correspond to the regions of key primary and secondary vortex structures (see figure 2.17(a)). Out of the remaining $N_{|\omega_z| < |\omega_z^*|}$ fluid points corresponding to $|\omega_z| < |\omega_z^*|$, a randomly undersampled set of $N^{sample}_{|\omega_z| < |\omega_z^*|}$ points based on a percentage

Table 2.7: Accuracy details of MB-PINN models for different levels of vorticity cutoff based undersampling. If not mentioned specifically, λ_{fluid} is taken as $\lambda_{fluid} = 0.1$.

Data base	S_{ω_z}(in %)	aRMSE	aMAE	aR^2	arRMSE (in %)
CI	100	7.64e-02	4.09e-02	9.91e-01	7.27
CI-S50	50	7.02e-02	3.93e-02	9.92e-01	6.55
CI-S10	10	6.79e-02	3.99e-02	9.92e-01	6.29
CI-S5	**5**	**5.01e-02**	**2.99e-02**	**9.96e-01**	**4.82**
CI-S1	1	5.41e-02	3.47e-02	9.95e-01	5.01
CI-S0	0	5.11e-02	3.30e-02	9.96e-01	4.83
CI-S0-US10 ($S^{NF}_{\omega_z} = 10\%$)	0	9.6e-02	4.98e-02	9.87e-01	9.90
CI-S5($\lambda_{fluid} = 0.01$)	**5**	**3.59e-02**	**2.25e-02**	**9.97e-01**	**3.22**
CI-S5 ($\lambda_{fluid} = 0.001$)	5	4.78e-02	2.67e-02	9.96e-01	4.77

sampling ratio, $S_{\omega_z} = \dfrac{N^{sample}_{|\omega_z|<|\omega^*_z|}}{N_{|\omega_z|<|\omega^*_z|}} \times 100$ is constructed and combined with the earlier retained data points. Note that $|\omega^*_z|$ does not always need to be 1.0, this may take a different value for a different flow problem and should be selected based on the flow-field vorticity levels of the specific problem of interest. In the present study, $S_{\omega_z} = 100\%, 50\%, 10\%, 5\%, 1\%$ and 0% are considered. For the extreme case $S_{\omega_z} = 0\%$, strong vorticity points in the near-field (NF) are further undersampled with a sampling ratio defined by, $S^{NF}_{\omega_z} = \dfrac{N^{sample}_{|\omega_z|>|\omega^*_z|}}{N_{|\omega_z|>|\omega^*_z|}} \times 100$, such that $S^{NF}_{\omega_z} = 10\%$. The spatial resolutions of the undersampled data sets are presented at a representative time stamp in figures 2.17(b)-(h). For each S_{ω_z}, the corresponding data sets are generated (see table 2.6) with the same number of temporal snapshots as in the CI data set. The resultant models are tested on Ref-ALE for pressure recovery and Ref-IBM for velocity reconstruction. For all the cases considered here, velocity data on the boundaries, Γ^r_{inlet}, Γ^r_{upper} and Γ^r_{lower} (depicted by black marker points in figures 2.17(b)-(h)), are also used during the training step, along with the $N_{|\omega_z|\geq|\omega^*_z|}$ and $N^{sample}_{|\omega_z|<|\omega^*_z|}$ points. Given that the bulk data is available over the entire training time domain, computing the $\mathcal{L}_{IC}$ constraint using the velocity data at the initial time stamp is not necessary as was also noted earlier.

To understand the effectiveness of the vorticity-based undersampling, the MB-PINN

models are first trained with the different data sets mentioned in table 2.6 for $\lambda_{fluid} = 0.1$. Here, $S_{\omega_z} = 100\%$ and $\lambda_{fluid} = 0.1$ corresponds to the baseline MB-PINN model discussed in section 2.7. The detailed error measures are presented in table 2.7. It is observed that, all the undersampled cases except the extreme case of CI-S0-US10, perform better in terms of the arRMSE compared to the baseline MB-PINN. With $\lambda_{fluid} = 0.1$, the best results are obtained for $S_{\omega_z} = 5\%$, with an overall arRMSE of 4.82% which is slightly higher than the earlier reported best case of $\lambda_{fluid} = 0.001$ (without vorticity cutoff based undersampling) in table 2.5 with an arRMSE of 3.217%. Next, keeping $S_{\omega_z} = 5\%$ fixed, as λ_{fluid} is reduced to 0.01, arRMSE again becomes

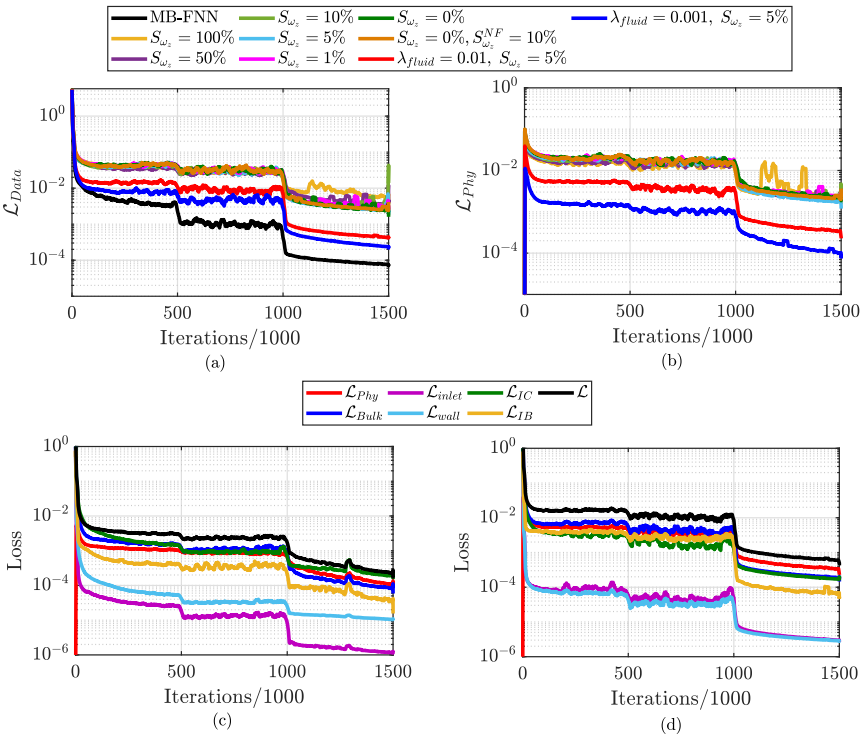


Figure 2.18: Loss convergence plots for select MB-PINN models considering different levels of vorticity undersampling combined with residual relaxation: (a) $\mathcal{L}_{data}$ with that of the best MB-FNN model, and (b) scaled $\mathcal{L}_{Phy}$. Unless otherwise specified, $\lambda_{fluid} = 0.1$ by default. Convergence of individual loss components for the optimal MB-PINN models with (c) $\lambda_{fluid} = 0.001$, $S_{\omega_z} = 100\%$, and (d) $\lambda_{fluid} = 0.01$, $S_{\omega_z} = 5\%$.

comparable to that of $\lambda_{fluid} = 0.001$ in table 2.5, that too with only $S_{Data} = 1.00\%$ for the CI-S5 data set as opposed to 4.89% in the case of the CI data set. The results for $\lambda_{fluid} = 0.001$ are also reasonably good at $S_{\omega_z} = 5\%$. However, the MB-PINN model demonstrates the best performance for $\lambda_{fluid} = 0.01$ and $S_{\omega_z} = 5\%$. In the case of velocity reconstruction, on comparing the accuracy of the optimal MB-PINN models with spatial linear interpolation from CI to Ref-IBM grids, it is observed that linear interpolation could also provide acceptable results for velocity reconstruction. The optimal MB-PINN models are only slightly better than spatial linear interpolation for $u-$ velocity reconstruction (see table 2.8). But the MB-PINN models outperform temporal linear interpolation (see table 2.9 for the test snapshot results for $t/T = 0.375$). Thus, getting a reasonable continuous time-space interpolator using linear interpolation is not plausible, which is however quite nicely achieved with PINNs. Importantly, MB-PINNs provide the added benefit of recovering pressure as a hidden variable, which is not possible with linear interpolation.

Even in the snapshot-wise rRMSE plots for velocity reconstruction (figures 2.17(i) and 2.17(j)) and pressure recovery (figure 2.17(k)), it is seen that $\lambda_{fluid} = 0.01$ and $S_{\omega_z} = 5\%$ case performs the best overall. Interestingly, for the extreme case of $S_{\omega_z} = 0\%$ and for $S^{NF}_{\omega_z} = 10\%$, although the relative velocity reconstruction errors worsen (figures 2.17(i) and 2.17(j)), the relative pressure errors have not as much and are contained below 15% (figure 2.17(k)). This indicates that even with $S_{Data} = 0.08\%$ with respect to Ref-IBM, MB-PINN is still able to recover pressure with reasonable accuracy. Note that both physics loss weighting and vorticity cutoff based sampling contribute towards relaxation of the physics constraints. When both are employed together, a relatively higher value of $\lambda_{fluid} = 0.01$ is found to be sufficient to match the best model performance.

Hence, it is possible to leverage the underlying flow features to improve data efficiency, while maintaining the expressivity. This is also confirmed by the loss convergence plots

in figure 2.18. It can be clearly seen how the combined data-driven losses $\mathcal{L}_{Data}$ for S_{ω_z} = 5% with λ_{fluid} = 0.01, 0.001 are closer to that of MB-FNN model in figure 2.18(a) and correspondingly the scaled $\mathcal{L}_{Phy}$ are lowest in figure 2.18(b). The loss convergence of individual loss components for the best MB-PINN models in the relaxation and undersampling experiments are presented in figures 2.18(c) and 2.18(d). Although the losses are slightly higher for the undersampled case, the order of convergence is still similar for both the optimal models considered.

From the point of view of a fluid dynamicist, the accuracy of the prediction of the key vortex structures are of interest. To this end, flow-field measures, such as the circulation of the LEV (Γ_{LEV}), self-induced velocity (U_{dipole}) of the first and the second dipoles in the wake are computed here. Errors in the predicted and derived quantities, Γ_{LEV} and U_{dipole} are good indicators of how well the primary vortex structures have been resolved

Table 2.8: Accuracy of the best MB-PINN models in comparison with spatial linear interpolation (CI to Ref-IBM grid), the purely data-driven MB-FNN, and the baseline MB-PINN models, respectively.

Model	RMSE	MAE	R^2	rRMSE
Linear		S_{ω_z} = 100%		
u	1.5e-02	1.6e-03	9.998e-01	1.29
v	6.8e-03	8.9e-04	9.998e-01	1.32
MB-FNN		S_{ω_z} = 100%		
u	5.8e-03	2.1e-03	9.999e-01	0.51
v	6.5e-03	2.3e-03	9.998e-01	1.31
MB-PINN		λ_{fluid} = 0.1 (Baseline)		
u	2.7e-02	1.1e-02	9.994e-01	2.41
v	3.0e-02	1.2e-02	9.961e-01	6.01
p	1.7e-01	1.0e-01	9.781e-01	13.41
MB-PINN		λ_{fluid} = 0.001		
u	9.6e-03	3.8e-03	9.999e-01	0.84
v	8.3e-03	3.3e-03	9.997e-01	1.67
p	8.9e-02	5.5e-02	9.943e-01	7.14
MB-PINN		λ_{fluid} = 0.01, S_{ω_z} = 5%		
u	8.6e-03	4.9e-03	9.999e-01	0.75
v	8.2e-03	4.7e-03	9.997e-01	1.64
p	9.1e-02	5.8e-02	9.938e-01	7.28

Table 2.9: Comparing the accuracy details for the best MB-PINN models with temporal linear interpolation (between $t_1 = 0.35$ and $t_2 = 0.4$) of velocity data from the Ref-IBM dataset over the intermediate test snapshot at $t/T = 0.375$.

Model	Accuracy			
Linear	RMSE	MAE	R^2	rRMSE
u	5.3e-02	8.8e-03	9.991e-01	2.92
v	3.9e-02	1.1e-02	9.946e-01	7.36
	$\lambda_{fluid} = 0.001$			
MB-PINN	RMSE	MAE	R^2	rRMSE
u	8.1e-03	3.8e-03	9.999e-01	0.71
v	7.8e-03	3.5e-03	9.997e-01	1.45
	$\lambda_{fluid} = 0.01$, $S_{\omega_z} = 5\%$			
MB-PINN	RMSE	MAE	R^2	rRMSE
u	7.2e-03	4.5e-03	9.999e-01	0.63
v	8.7e-03	5.2e-03	9.997e-01	1.61

by the surrogate models. Given a uniform grid of cell size $dx = dy$, and the vorticity ω_z at any cell center, the calculation of LEV circulation Γ_{LEV} within the bounding box surrounding the LEV is given by

$$\Gamma_{LEV} = \sum_{i=1}^{N_{LEV}} \omega_z^i dx dy, \tag{2.38}$$

where, N_{LEV} corresponds to the number of grid points in a bounding box containing the LEV. Also, the self-induced velocity of a dipole A-B can be calculated as

$$U_{dipole} = \frac{\Gamma_{avg}}{2\pi\xi_{AB}}, \quad \text{where,} \tag{2.39}$$

$$\Gamma_{avg} = \frac{1}{2}(|\Gamma_A| + |\Gamma_B|), \quad \text{and,} \tag{2.40}$$

$$\xi_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}. \tag{2.41}$$

Here, A and B indicate successive vortex cores that together form a dipole in the trailing-wake, with corresponding circulations Γ_A and Γ_B, respectively. The dipoles are marked by the two rectangular boxes in the wake; see figure 2.19(a). ξ_{AB} is the distance between the centers of vortices A and B with (x_A, y_A) and (x_B, y_B) being the respective center coordinates. For more details on the calculation of the above measures, one can

refer to our earlier study (Majumdar *et al.*, 2022). Here, the above flow-field measures are calculated at a typical test time instant, $t/T = 0.375$, which was unseen during the training. The results are presented qualitatively in figure 2.19 and quantitatively in table 2.10.

It is observed that undersampling combined with relaxation ($S_{\omega_z} = 5\%$, $\lambda_{fluid} = 0.01$) improves the LEV resolution and captures the other vortex structures remarkably well. While this is achieved, the data efficiency as compared to the optimal MB-PINN model obtained in the relaxation study, is also maintained. Notably, all the models predict the vortex centers exactly at the locations of the respective true data. The primary and secondary vortex structures and the shear layers are resolved better compared to

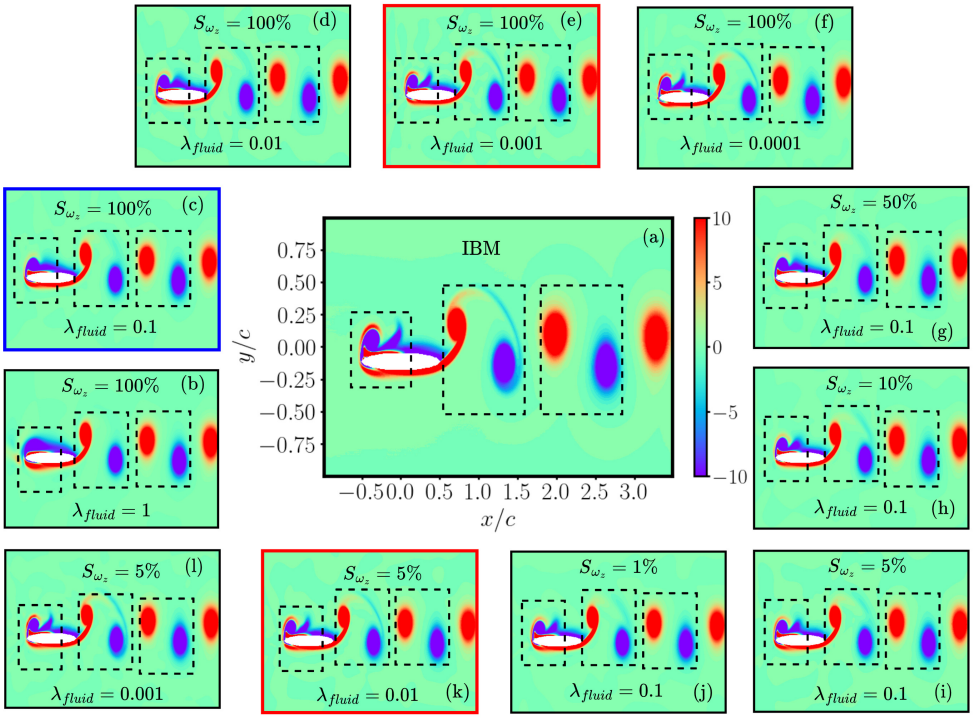


Figure 2.19: Comparison of vorticity contours (a) obtained from IBM data with the predictions of MB-PINN models in (b)-(l) for different λ_{fluid} and S_{ω_z} at a typical test time stamp $t/T = 0.375$. The dotted boxes indicate the regions of strong LEV and vortex dipoles. The baseline MB-PINN model prediction in (c) is highlighted by a blue bounding box, whereas, the best MB-PINN model predictions are highlighted by a red bounding box in (f), and (l), respectively.

Table 2.10: Percentage errors $\epsilon_{\#}$ in predicting LEV circulation Γ_{LEV}, self-induced dipole velocity U_{dipole} for the first two dipoles in the wake at a typical test time stamp $t/T = 0.375$, for MB-PINN models. Here, # = Γ_{LEV}, U^1, U^2. The true values of the derived quantities are $\Gamma_{LEV} = 1.450$, $U^1_{dipole} = 0.394$, and $U^2_{dipole} = 0.379$.

Model	$\epsilon_{\Gamma_{LEV}}$	ϵ_{U^1}	ϵ_{U^2}
Relaxation study (different λ_{fluid})			
$\lambda_{fluid} = 1$	41.006	0.761	0.264
0.1	7.994	0.309	0.248
0.01	1.033	0.129	0.051
0.001	**0.258**	**0.175**	**0.047**
0.0001	0.758	0.132	0.043
Undersampling study (different S_{ω_z} in %))			
$S_{\omega_z} = 50$	10.425	0.397	0.244
10	8.139	0.585	0.188
5	1.381	0.013	0.018
1	1.968	0.105	0.003
0	1.093	0.016	0.149
0, $S^{NF}_{\omega_z} = 10$	3.001	0.399	0.031
5, $\lambda_{fluid} = 0.01$	**0.073**	**0.009**	**0.051**
5, 0.001	0.149	0.079	0.024

the baseline model in figure 2.19(c), either when the residuals are just further relaxed (figure 2.19(f)) or when relaxed and combined with the selective vorticity based sampling strategy (figure 2.19(l)).

A qualitative comparison of the flow-field from the optimal MB-PINN models (figures 2.20(a)-(c)) indicates that even the secondary structure has been captured accurately at time stamp previously unseen by the network during training. As discussed earlier, spatial linear interpolation of the velocity data from CI to Ref-IBM grid is comparable with MB-PINNs (see boxes B0.1, B1 and B2 in figures 2.20(b)-(c)). Temporal linear interpolation (shown for test snapshot prediction at $t/T = 0.375$) performs poorly due to the presence of a moving boundary as seen in the boxes B0.2 in figures 2.20(b)-(c)). The point-wise normalized errors in pressure in the vicinity of the LEV (as seen in the boxes B1 in figures 2.20(b)-(c)) are slightly worse than that of $\lambda_{fluid} = 0.001$ and $S_{\omega_z} = 100\%$ (as seen in the boxes B2 in figures 2.20(b)-(c)). In the

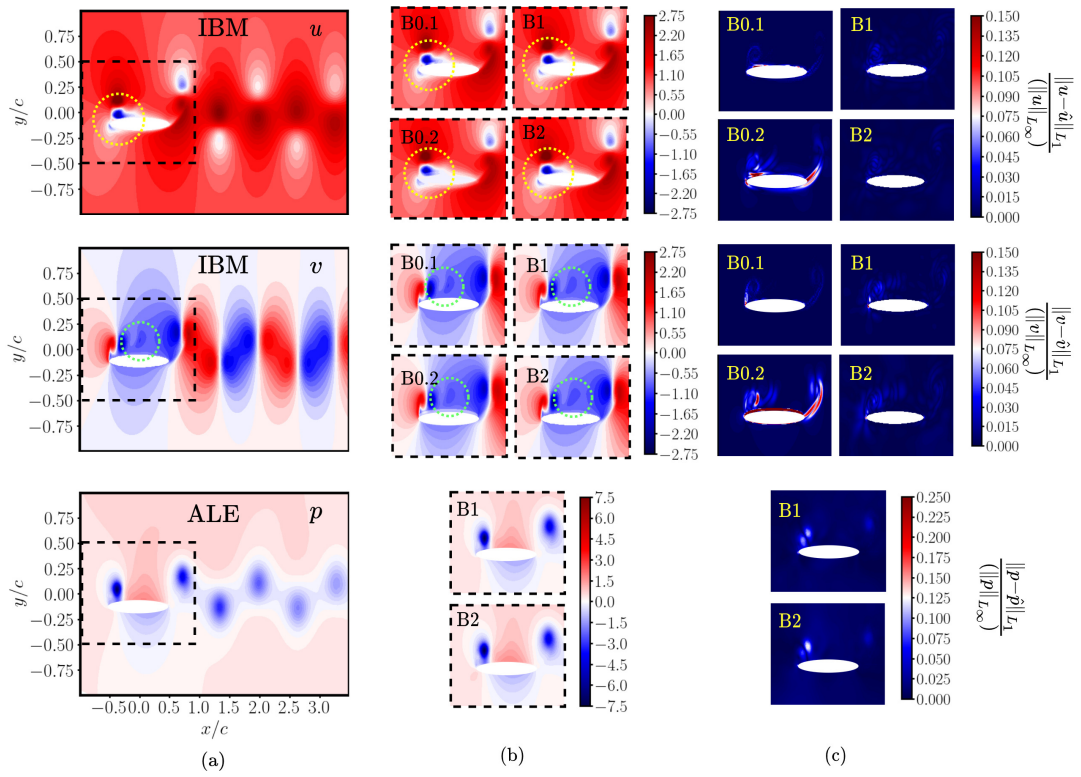


Figure 2.20: Comparison of (a) true IBM velocity and ALE pressure with (b) linear interpolation (velocity data) and the predictions obtained from the best MB-PINN models for a testing time stamp $t/T = 0.375$, and (c) corresponding maximum value normalized point-wise absolute error contours. Here, the rectangular boxes in (b) and (c) representing the near-field region in (a) marked by B0.1 corresponds to spatial linear interpolation from CI to Ref-IBM grid, B0.2 corresponds to temporal linear interpolation, B1 corresponds to λ_{fluid} with $S_{\omega_z} = 100\%$ while that of B2 corresponds to $\lambda_{fluid} = 0.01$ with $S_{\omega_z} = 5\%$.

velocity and pressure slices (see figures 2.21(a)-(c)) taken at $x/c = 0.0c$, for the test time stamp $t/T = 0.375$, queried on the CI grid it is seen that the optimal models obtained in the relaxation and undersampling experiments are equivalent and closely match the true data. Moreover, it can be seen in the insets near the boundary D1 in figure 2.21(a), and D4 in figure 2.21(b) that the no-slip velocity boundary condition is closely satisfied for the optimal MB-PINN models. As seen in figure 2.21(c), the optimal models are very close to that of the true data in pressure recovery while for $\lambda_{fluid} = 0.0001$, it breaks down away from the solid boundary. In the slices of velocity (inshots D11-D14 of

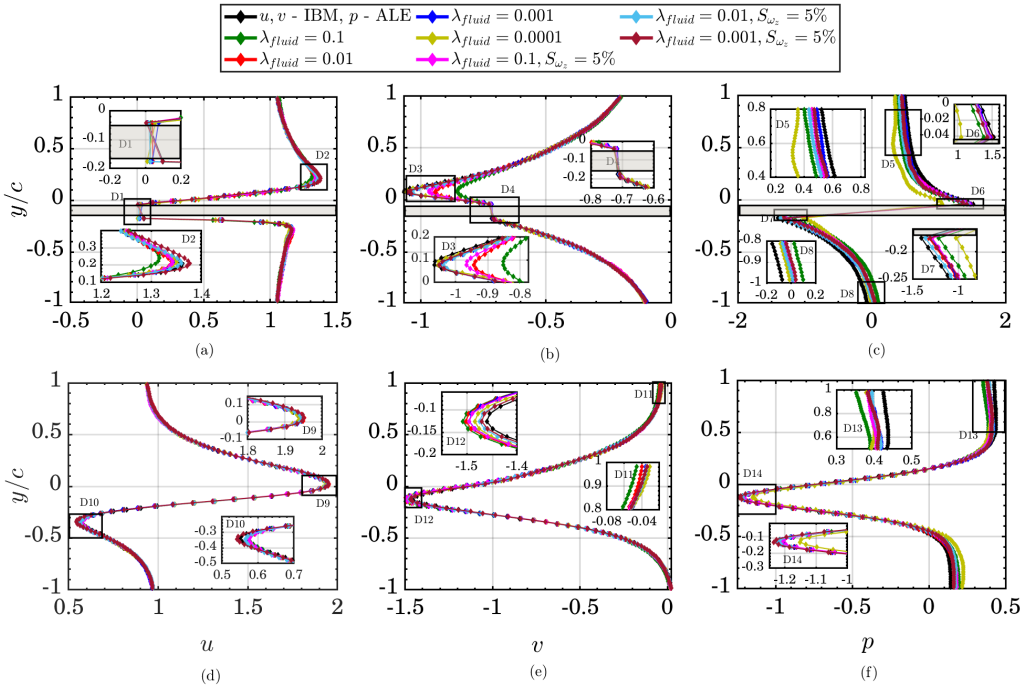


Figure 2.21: Velocity and pressure slices for a test time stamp $t/T = 0.375$ queried on the CI grid for different values of relaxation and vorticity undersampling. The obtained slices are compared with interpolated IBM velocity and ALE pressure data for a slice passing through, (a)-(c) the centre of the foil at $x/c = 0.0c$ where the rectangular region marked by a gray patch represents the solid body and (d)-(f) with slices at a far field location $x/c = 1.5c$. The zoomed inshots are presented using the correspondingly labeled close up shots **D1**-**D14**

figures 2.21(d) and 2.21(e)) taken at a far field location $x/c = 1.5c$ from the foil center, all the models are more or less similar in velocity reconstruction. Whereas, slight differences are seen in pressure recovery (as seen in insets D13 and D14 of figure 2.21(f)) across optimal and sub-optimal cases though it still follows the overall trend of the true data. Stark differences are observed only in the near-field region across parameters. However, it is still remarkable considering that the model was trained using an undersampling approach with about 20% of data points compared to the CI data set and about 1% to that of Ref-IBM, while still maintaining an equivalent level of performance.

Owing to the localized nature of the strong velocity gradients in the flow-field, the

vorticity cutoff based data sampling leads to reasonable data efficiency (requires a significantly less number of data points for training), while maintaining the expressivity of the PINN model. There is almost a 5 times decrease in the data requirement for the optimal combination of $\lambda_{fluid} = 0.01$ and $S_{\omega_z} = 5\%$. Even at $S_{\omega_z} < 5\%$, while reasonable accuracy level can still be obtained. It is also envisioned that, by training efficient PINN models on reduced snapshot domains of interest which have data points stored in an undersampled manner when the solver writes the data, one could potentially obtain huge savings on the memory as well. For a typical example calculation for the present problem, we observed the memory requirement per snapshot to go down by two orders of magnitude.

Moreover, in a related study (see Appendix B), it was also shown with the help of a novel zonal splitting of loss component gradients and comparing their respective gradients that vorticity-based cutoff sampling overcomes the gradient vanishing problem. Note that, gradient vanishing refers to the diminishing of loss gradients during back propagation eventually leading to slowing down and even plateauing of loss convergence (Goodfellow *et al.*, 2016). Additionally, two relative gradient statistical metrics were proposed to determine which zone has a relatively greater contribution during training. It was observed that the region surrounding the moving body has the highest contribution during training, while the updates from the far-field region is relatively lower magnitude. This is because of strong flow-field gradients in the moving body region. For more details, see Appendix B.

2.9 DISCUSSION

In this chapter, an Immersed Boundary Aware (IBA) surrogate modeling framework has been proposed for unsteady flow reconstruction around moving bodies. The framework leverages the flexibility of Physics-Informed Neural Networks (PINNs) while drawing conceptual inspiration from the Immersed Boundary Method (IBM)—where a moving

body is immersed in a fixed Eulerian background grid. This design eliminates the need for case-specific transformations to body-attached reference frames, offering a generalizable approach to modeling complex fluid–structure interactions.

Two formulations were developed under this framework: MB-PINN (based on the standard Navier–Stokes equations) and MB-IBM-PINN (based on the IBM-modified Navier–Stokes formulation). These were tested on flow past a low-Reynolds-number flapping body—a canonical case representative of unsteady wake dynamics with strong vortices, steep gradients, and temporal variations. Both models were trained for non-intrusive pressure recovery (hidden variable inference) and velocity field reconstruction.

A key aspect of the framework was the incorporation of a multi-part physics loss weighting strategy to balance competing physical residuals and improve data efficiency under a fixed computational budget. The MB-PINN approach demonstrated high efficiency when the solid body's position and velocity were known *apriori*, whereas the MB-IBM-PINN achieved comparable reconstruction accuracy but incurred greater computational cost per iteration due to its larger computational graph.

Additionally, physics-based vorticity cut-off sampling was introduced to enhance data efficiency, leveraging regions of strong vorticity as indicators of dynamically significant flow features. The models maintained accurate pressure recovery even when trained on undersampled data, showing strong generalization across solvers.

Overall, the results confirm that MB-PINN provides superior performance in pressure recovery and velocity reconstruction tasks under resource constraints especially when body position, and velocity are known *apriori*. However, the potential of MB-IBM-PINN lies in scenarios where such moving body information is scarce or unavailable. These insights form the basis for extending the IBA framework beyond hidden physics recovery which shall be investigated in the subsequent chapter.

2.10 SUMMARY AND CONCLUSIONS

This chapter introduced and evaluated an Immersed Boundary Aware surrogate modeling framework combining PINNs with the conceptual foundations of the IBM. the IBA formulation using PINNs removes the need for case-specific transformations and enables flexible modeling of moving boundaries within a fixed Eulerian grid indicating the generality of the framework. Of the two PINN formulations proposed, MB-PINN provided efficient, accurate recovery of velocity and pressure fields when solid-body motion is known *apriori*; whereas, MB-IBM-PINN achieved similar accuracy but at a higher computational cost and when the fictitious solid region velocity points were discarded from analysis. To improve data efficiency without compromising accuracy, a vorticity-based sampling strategy which effectively reduced the training data needed. It is important to note here that the IBA framework need not be limited to the pressure recovery problem alone. There are other potential areas in which the proposed models' capabilities can be fully harnessed for future applications. In some experimental scenarios, the body configuration, position, shape and velocity might not be known. This might pose a challenge in recovering near-field pressure and reconstructing data. However, it would be even more challenging to recover not just velocity and pressure, but also the moving boundary configuration. Moreover, in this chapter, the temporal resolution was reasonably high, and the flow was periodic which in reality is quite the opposite. Hence, building upon these challenges, Chapter 3 explores the hidden boundary estimation, a super-resolution and a forward problem examples to highlight the potential of IBA framework beyond pressure recovery.

CHAPTER 3

EVALUATION OF IBA FRAMEWORK UNDER DIFFERENT DATA AVAILABILITY SCENARIOS

3.1 INTRODUCTION

Although pressure recovery was considered as the main focus in Chapter 2, the IBA framework should not be exclusive to just this problem. In certain situations information of the solid body's exact position and its velocity might not be known *apriori* (Dabiri *et al.*, 2014; Calicchia *et al.*, 2023). Also, the fluid velocity data may only be available/measured a little away from the solid surface. For a complete understanding of the flow-field and the interactions with the solid body, estimating the solid body's position, velocity and shape is important. When only velocity flow-field data is available and that too measured/ sampled away from the moving body, this happens to be a hidden boundary estimation problem. However, for a moving body, this poses a challenging problem with the availability of the fluid velocity data alone. Until very recently, to the best of author's knowledge, hidden boundary estimation from sparse velocity flow-field data have not been attempted. Like the hidden pressure recovery problem discussed earlier in Chapter 2, here the body position, velocity and shape are hidden variables. Unlike pressure, this hidden boundary position and shape are not explicitly embedded into the continuous form of the governing equations. Hence a case study has been undertaken here to see how well the proposed IBA framework can handle such problems.

Here, the above scenario is recreated through IBM simulation data. The fluid domain velocity data (downsampled) are collected a little away from the solid surface, and two problems are explored. Firstly, in the absence of the body position and velocity information at training, the efficacy of the IBA framework in reconstructing the bulk velocity and recovering pressure is investigated. Once trained, it is also tested if the solid

body velocity can be recreated as well (to check this, the Lagrangian marker positions of the solid boundary are used). In this part, the body shape is assumed to be known *apriori*. For the second scenario, it is assumed that the exact position, velocity, and shape of the solid body are not known while training or testing. Note that in both problems, it has been assumed that the body is moving but rigid in nature. The results of these investigations are given in the following subsections.

The above scenarios would be investigated under the assumption that flow-field data is resolved in time even if spatially sparse. However, in situations where one might encounter very low-quality data a model that could super-resolve the flow-fields in a physically consistent manner is desirable (Fukami *et al.*, 2023; Aliakbari *et al.*, 2022). Sometimes, with just a single initial time data snapshot of flow-field in a truncated computational domain, solving a forward problem to evolve the flow under certain boundary conditions would also be desirable. Here, the forward problem is an example of no-data scenario and the PINN model would have to hence be trained as a surrogate to a CFD solver. However, early works such as Krishnapriyan *et al.* (2021); Wang *et al.* (2021*a*); Mattey and Ghosh (2022), have reported that forward problems involving nonlinear PDEs are extremely challenging due to propagation failure mode of PINNs where error tends to accumulate over time. In a proof of concept manner, a super-resolution and a forward problem are investigated to identify potential pitfalls of the proposed framework.

Overall, the key objectives and contributions of this chapter are as follows:

- Demonstrate the capability of the IBA framework to recover hidden flow variables (velocity and pressure) when solid body kinematics are *unknown*, by estimating body position and velocity directly from the data.
- Extend the IBA framework to settings where the solid body *shape* is also unknown, using geometric priors to recover body shape, position, and motion alongside velocity and pressure.
- Evaluate the performance of the IBA framework under *low-fidelity or low-resolution* data by testing its ability to perform physically consistent super-resolution of velocity and pressure fields.

- Assess the limits of the framework in a *no-data* scenario through a forward problem, highlighting the challenges of solving the governing PDEs over a restricted spatio-temporal domain without bulk simulation data.

- Provide a unified perspective on how the IBA framework behaves across these varying levels of data availability — from full knowledge of body shape to complete data absence — and outline the potential of the IBA framework and critical problems where IBA framework performs remarkably well.

Note that in all these investigations, the periodically plunging elliptic foil system as previously described in Chapter 2 is considered. This chapter is structured as follows: sections 3.2 investigates simultaneous velocity reconstruction, pressure recovery and estimation of solid body velocity in the absence of body position and velocity information. Here it is assumed that the exact shape of the body is known. In section 3.3, it is assumed that the exact shape is unknown *apriori* but some basic priors on the geometry are enforceable to recover body shape, position, and velocity along with velocity reconstruction and pressure recovery. Since in most cases, one might obtain low-fidelity/low-resolution data, section 3.4 explores a super-resolution problem. In complete absence of any bulk simulation data, section 3.5 explores a complex forward simulation problem over a restricted spatio-temporal domain outlining the challenges. In section 3.6, broad perspectives are drawn and the potential of IBA framework is highlighted in different data availability scenarios drawing attention to certain critical scenarios explored subsequently in the thesis. Finally, the key conclusions are presented in section 3.7

3.2 VELOCITY RECONSTRUCTION AND PRESSURE RECOVERY, AND ESTIMATION OF THE SOLID BODY VELOCITY IN THE ABSENCE OF BODY POSITION AND VELOCITY INFORMATION

It is first investigated whether the PINN models under the present IBA framework are capable of reconstructing the bulk velocity and recovering the pressure. It also tested if the solid body velocity (IB velocity) can be predicted in the absence of any body position or velocity data at training. Note that the body shape/position information is not

used while training but is used subsequently during testing the model (for this part the Lagrangian marker locations are provided as model inputs). Hence, it is assumed here that the shape of the body is known *apriori*.

In both MB-PINN and MB-IBM-PINN, the velocity boundary condition was enforced through the loss component, $\mathcal{L}_{IB}$. Here, the data loss of velocity predictions of the network at the IB Lagrangian marker points were minimized with respect to the true IB velocity. Note that the x component of the IB velocity is zero and the y component is the plunge velocity. In MB-IBM-PINN, in addition to $\mathcal{L}_{IB}$, there was an additional data loss component $\mathcal{L}_{IBvar}$, which enforced the constraints on the forcing ($\boldsymbol{f}$) and the source/sink (q) terms. The loss formulation of MB-PINN remains the same in this case study. Whereas, since the body position and shape are not fed as model inputs while training, the physics loss for MB-IBM-PINN is reformulated without fluid-solid partitioning unlike the one previously adopted in Chapter 2 (see Eqn. (2.31)),

$$\mathcal{L}_{Data} := \lambda_{Bulk}\mathcal{L}_{Bulk} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{IC}\mathcal{L}_{IC}) + \mathcal{L}_{IB} + \lambda_{IBvar}\mathcal{L}_{IBvar}, \tag{3.1}$$

$$\mathcal{L}_{Phy} := \lambda_{fluid \cup solid}(\mathcal{L}_{PDE_{m_x}} + \mathcal{L}_{PDE_{m_y}} + \mathcal{L}_{PDE_c}). \tag{3.2}$$

The data points are obtained from the coarsened IBM simulation dataset CI (see table 2.1 of section 2.6.1). In the present case study, three situations are considered, where the fluid data points are selected about 2 (*i.e.* $0.032c$), 5 (*i.e.* $0.08c$), and 10 (*i.e.* $0.16c$) cells away from the IB. For MB-PINN, the collocation points are taken to be the same as the data points but this is not the case for MB-IBM-PINNs.

As the fluid-solid partitioning of the physics loss cannot be carried out in MB-IBM-PINN, the governing equations considered in MB-IBM-PINN formulation need to be satisfied over the entire domain and not just the fluid region. As a result, note in Eqn. (3.2), there is no λ_{solid}, and instead $\lambda_{fluid \cup solid}$ is used considering the union of fluid and solid

region. Here, residual collocation points are sampled from the entire Eulerian background grid of the CI dataset towards the computation of the physics loss. Throughout this current exercise, the training budget, and network size are kept the same as in the main investigation. Here, the weighting coefficients, $\lambda_{Bulk} = \lambda_{BC} = 1$ (see equations (2.14) and (2.21)), and, the no-slip boundary condition loss coefficient, $\lambda_{IB} = 0$, are considered for both the models. As was done previously as mentioned in the in section 2.7, the $\mathcal{L}_{IC}$ need not be computed here as well. Here, the velocity data only on the inlet boundary is considered for computation of $\mathcal{L}_{BC}$. The data loss component of the weighting coefficient of the IB variables ($\boldsymbol{f}$ and q), $\mathcal{L}_{IBvar}$, are also varied, taking $\lambda_{IBvar} = 1$ or 0.

Table 3.1 presents the accuracy results for bulk velocity reconstruction and pressure recovery. For the sake of comparison, the best-performing MB-PINN cases from the Chapter 2 table 2.8) are also included. Pressure recovery for MB-IBM-PINN without fluid-solid partitioning breaks down below $\lambda_{fluid \cup solid}0.01$, unlike the case of fluid-solid partitioning with $\lambda_{fluid} = 0.001$ and $\lambda_{solid} = 0$. Therefore, $\lambda_{fluid \cup solid}0.001$ for MB-PINN and $\lambda_{fluid \cup solid}0.01$ for MB-IBM-PINN (no fluid-solid partitioning) are considered here. For $\lambda_{IB} = 0$ and $\lambda_{IBvar} = 0$, the pressure recovery by MB-IBM-PINN

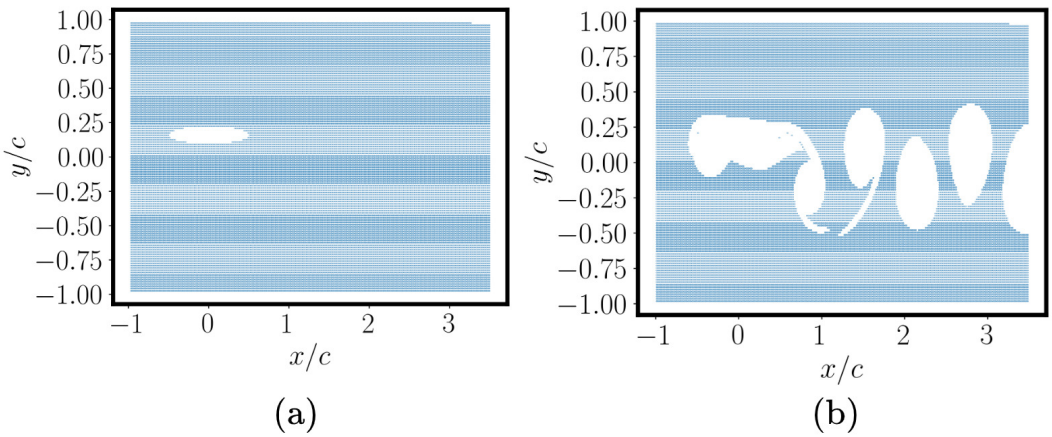


Figure 3.1: Grid points that are considered to compute L_{IBvar} constraints for a representative snapshot in time $t/T = 0.0$: (a) entire fluid region, and (b) region of low vorticity where $|\omega_z| < |\omega^*|$ considered in the VF strategy.

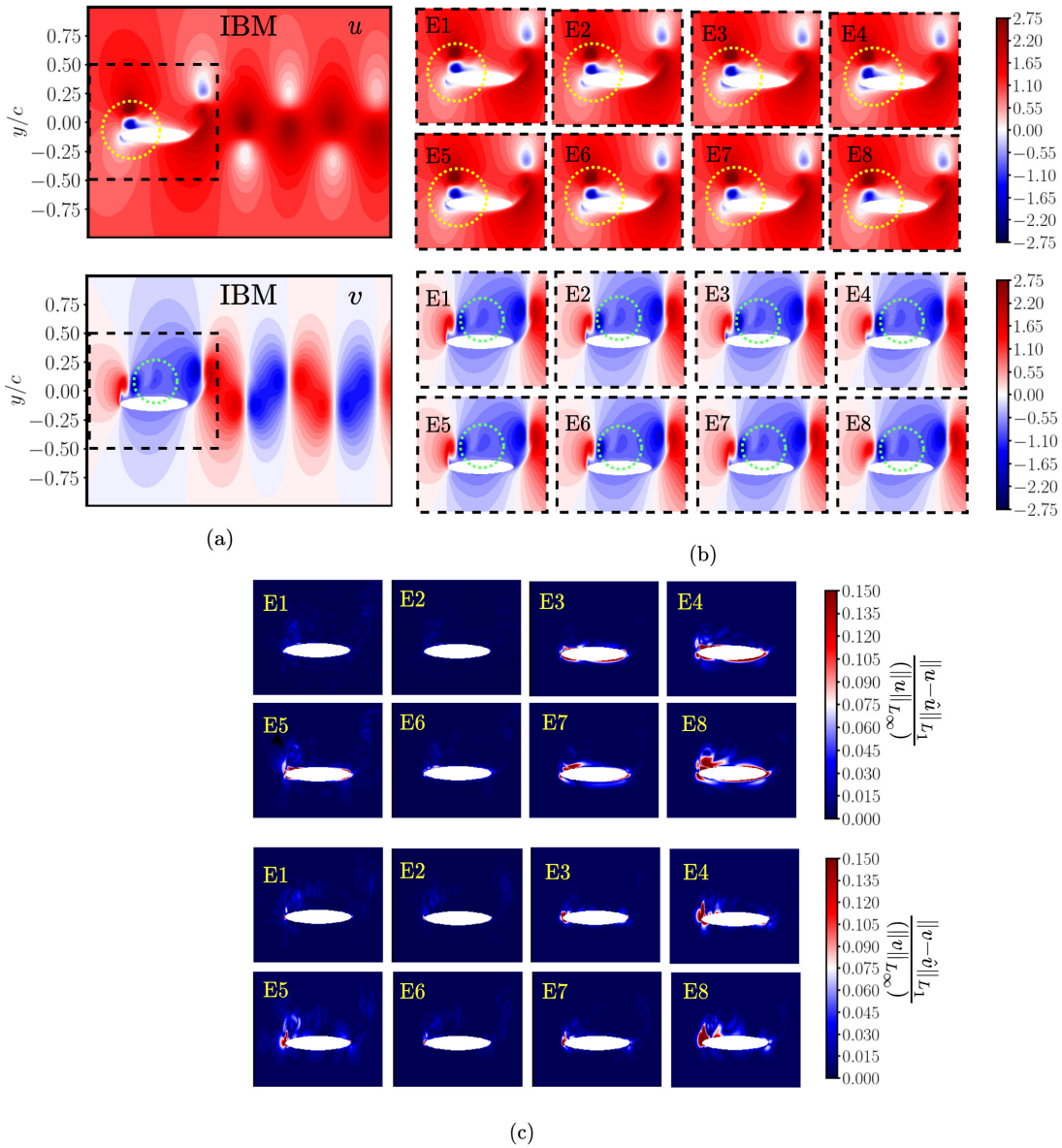


Figure 3.2: Comparison of (a) true IBM velocity with the (b) predictions obtained from optimal MB-PINN (with and without $\mathcal{L}_{IB}$ constraint) and MB-IBM-PINN(without $\mathcal{L}_{IB}$ constraint) models and (c) corresponding maximum value normalized pointwise absolute error contours. The rectangular boxes in (b) and (c) representing the near-field region in (a) marked by E1 $\lambda_{fluid} = 0.001, \lambda_{IB} = 1$), E2-E4 (MB-PINN - $\lambda_{fluid \cup solid} 0.001, \lambda_{IB} = 0$ with fluid data points sampled 2, 5 and 10 cells away from the solid boundary, respectively), E5 (MB-IBM-PINN - $\lambda_{fluid \cup solid} 0.01, \lambda_{IB} = 0, \lambda_{IBvar} = 1$(no VF)), and E6-E8 (MB-IBM-PINN - $\lambda_{fluid \cup solid} 0.001, \lambda_{IB} = 0, \lambda_{IBvar} = 1$ (VF) with fluid data points sampled 2, 5 and 10 cells away from the solid boundary, respectively).

Table 3.1: Accuracy details of MB-PINN and MB-IBM-PINN when trained without $\mathcal{L}_{IB}$ (*i.e.* $\lambda_{IB} = 0$) and without solid-fluid partitioning. (*)-marked results are for the data-driven MB-FNN and the optimal MB-PINN models (with $\mathcal{L}_{IB}$, *i.e.* $\lambda_{IB} = 1$), taken tables 2.3 and 2.8 of Chapter 2 for comparison.

Model	Accuracy				
MB-FNN*	RMSE	MAE	R^2	rRMSE (in%)	arRMSE (in%)
u	5.8e-03	2.1e-03	9.999e-01	0.51	0.91
v	6.5e-03	2.3e-03	9.998e-01	1.31	
MB-PINN*	$\lambda_{fluid \cup solid}$0.001				
u	9.6e-03	3.8e-03	9.999e-01	0.84	
v	8.3e-03	3.3e-03	9.997e-01	1.67	3.22
MB-PINN*	$\lambda_{fluid \cup solid}$0.01, $S_{\omega_z} = 5\%$				
u	8.6e-03	4.9e-03	9.999e-01	0.75	
v	8.2e-03	4.7e-03	9.997e-01	1.64	3.22
p	9.1e-02	5.8e-02	9.938e-01	7.28	
MB-PINN	$\lambda_{fluid \cup solid}$0.001, $\lambda_{IB} = 0$, 2 cells away				
u	4.5e-03	1.9e-03	9.999e-01	0.39	
v	4.9e-03	2.0e-03	9.999e-01	0.99	2.23
p	6.6e-02	3.6e-02	9.968e-01	5.32	
MB-PINN	$\lambda_{fluid \cup solid}$0.001, $\lambda_{IB} = 0$, 5 cells away				
u	2.0e-02	5.6e-03	9.994e-01	2.39	
v	1.4e-02	4.7e-03	9.989e-01	2.89	4.26
p	1.3e-01	8.2e-02	9.893e-01	7.48	
MB-PINN	$\lambda_{fluid \cup solid}$0.001, $\lambda_{IB} = 0$, 10 cells away				
u	4.5e-03	1.9e-03	9.999e-01	5.58	
v	4.9e-03	2.0e-03	9.999e-01	9.47	10.84
p	6.6e-02	3.6e-02	9.968e-01	17.49	
MB-IBM-PINN	$\lambda_{fluid \cup solid}$0.01, $\lambda_{IB} = 0, \lambda_{IBvar} = 0$				
u	3.6e-02	1.7e-02	9.989e-01	3.22	
v	5.3e-02	2.1e-02	9.885e-01	10.67	32.75
p	1.12	7.5e-01	2.824e-01	84.44	
MB-IBM-PINN	$\lambda_{fluid \cup solid}$0.01, $\lambda_{IB} = 0, \lambda_{IBvar} = 1$				
u	2.4e-02	7.9e-03	9.995e-01	2.09	
v	2.2e-02	7.3e-03	9.979e-01	4.39	7.25
p	1.9e-01	1.1e-01	9.691e-01	15.25	
MB-IBM-PINN	$\lambda_{fluid \cup solid}$0.01, $\lambda_{IB} = 0, \lambda_{IBvar} = 1$ (VF), 2 cells away				
u	9.5e-03	2.9e-03	9.999e-01	0.83	
v	7.5e-03	2.6e-03	9.998e-01	1.49	3.74
p	1.1e-02	5.2e-02	9.914e-01	8.92	
MB-IBM-PINN	$\lambda_{fluid \cup solid}$0.01, $\lambda_{IB} = 0, \lambda_{IBvar} = 1$ (VF), 5 cells away				
u	3.4e-02	5.6e-03	9.991e-01	2.92	
v	1.7e-02	2.9e-03	9.987e-01	3.42	8.73
p	1.0e-01	3.9e-02	9.597e-01	19.86	
MB-IBM-PINN	$\lambda_{fluid \cup solid}$0.01, $\lambda_{IB} = 0, \lambda_{IBvar} = 1$ (VF), 10 cells away				
u	5.8e-02	1.0e-02	9.974e-01	5.05	
v	3.6e-02	5.8e-03	9.946e-01	7.23	15.17
p	4.4e-01	5.2e-02	8.875e-01	33.22	

is not satisfactory as seen in table 3.1, due to the lack of constraints on $\boldsymbol{f}$ and q. Thus, $\boldsymbol{f}$ and q could take any non-zero values in the fluid domain. This directly affects the minimization of the physics loss and the pressure recovered becomes inaccurate. With

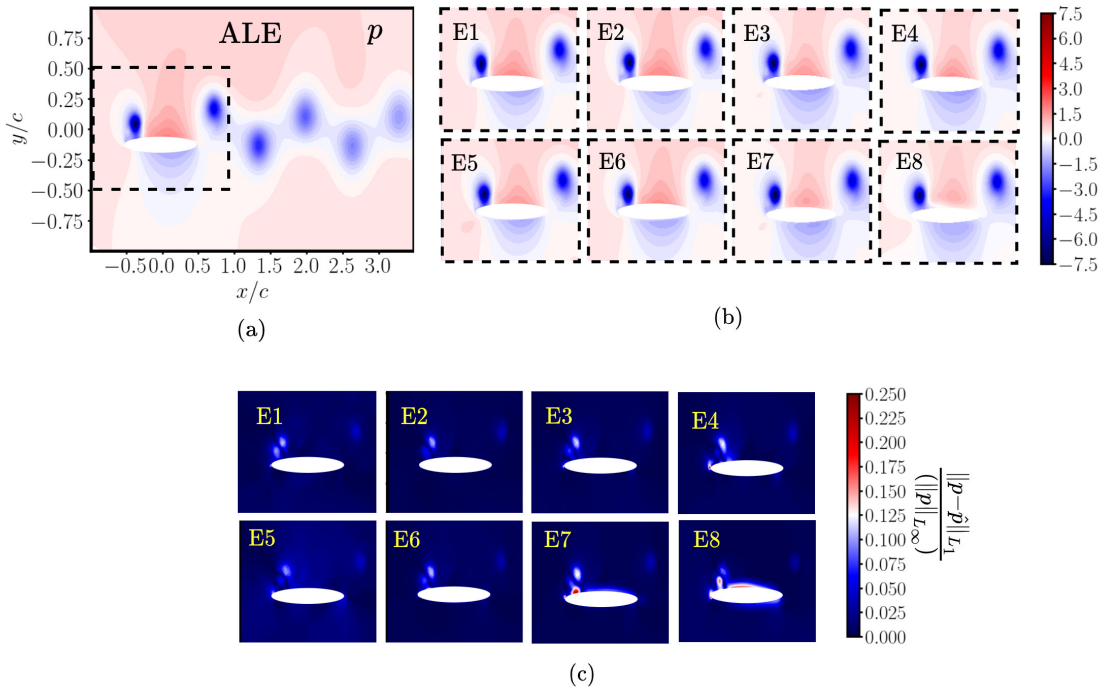


Figure 3.3: Comparison of (a) true ALE pressure with the (b) predictions obtained from optimal MB-PINN (with and without $\mathcal{L}_{IB}$ constraint) and MB-IBM-PINN (without $\mathcal{L}_{IB}$ constraint) models and (c) corresponding maximum value normalized pointwise absolute error contours. The rectangular boxes in (b) and (c) representing the near-field region in (a) marked by E1 $\lambda_{fluid} = 0.001, \lambda_{IB} = 1$), E2-E4 (MB-PINN - $\lambda_{fluid \cup solid} 0.001, \lambda_{IB} = 0$ with fluid data points sampled 2, 5 and 10 cells away from the solid boundary, respectively), E5 (MB-IBM-PINN - $\lambda_{fluid \cup solid} 0.01, \lambda_{IB} = 0, \lambda_{IBvar} = 1$(no VF)), and E6-E8 (MB-IBM-PINN - $\lambda_{fluid \cup solid} 0.001, \lambda_{IB} = 0, \lambda_{IBvar} = 1$ (VF) with fluid data points sampled 2, 5 and 10 cells away from the solid boundary, respectively).

$\lambda_{IBvar} = 1$, the pressure recovery is found to be dependent on the sampling of grid points where $\boldsymbol{f}$ and q are to be constrained to zero. The regions of strong flow-field gradients (specifically, the regions of strong vorticity) should ideally be avoided to compute $\mathcal{L}_{IBvar}$ in order to ease the training. Using the same vorticity cut-off value $|\omega^*| = 1$ as was done in the undersampling study (see section 2.8 of Chapter 2), the grid points to compute $\mathcal{L}_{IBvar}$ are sampled from the fluid region where $|\omega| < |\omega^*|$ (see figure 3.1(a) and (b) for an example). This sampling strategy (called VF hereon) improves the predictions of MB-IBM-PINN as seen in table 3.1 and figure 3.2.

Both MB-PINN and MB-IBM-PINN (with VF) models with fluid data sampled 2 cells away perform satisfactorily in bulk velocity reconstruction and pressure recovery and are comparable in their accuracy, though not having $\mathcal{L}_{IB}$ during training. In fact, the reconstructed velocity and pressure are almost indistinguishable from the true data (see figure3.2(a), figure3.3(a), and boxes E1-E8 in figure 3.2(b), and figure 3.3(b), respectively). Notably, the MB-PINN model trained on fluid data sampled 2 cells away from the IB is slightly better than the optimal MB-PINN models presented in Chapter 2. This could be due to one lesser competing objective/ constraint in the loss function, *i.e.* the absence of the $\mathcal{L}_{IB}$ constraint at training. This is in alignment with earlier works (Wang *et al.*, 2021*a*; Heydari *et al.*, 2019; Bischof and Kraus, 2025) which have reported that additional constraints enforcing the boundary conditions often make the training slightly more difficult. Moreover, since the fluid data points are quite close to the IB, the velocity on the IB is reconstructed reasonably well. Interestingly, this also corroborates with the results presented in the earlier chapter 2 section 2.7, where the $\mathcal{L}_{wall}$ was converging reasonably well in spite of not explicitly minimising it while training (see figures 2.12 and2.18(c) from sections 2.7 and 2.8 in Chapter 2, respectively). This was because bulk velocity data points were available quite close to the upper and lower boundaries of the truncated domain.

However, as the fluid data points are sampled further away from the solid body, the bulk velocity reconstruction and pressure recovery suffer at regions closer to the solid boundary as seen in figure 3.2(c), and figure 3.3(c). The highest errors in velocity reconstruction and pressure recovery are observed for the MB-IBM-PINN and MB-PINN models with fluid data sampled 10 cells away from the IB.

Next, it is checked whether both the PINN models can satisfactorily reconstruct the velocity on the IB, when $\lambda_{IB} = 0$. The Lagrangian marker points are used as model inputs to validate the velocity predictions at the IB locations. It is observed that MB-PINN and MB-IBM-PINN could recover the body velocity with an rRMSE of around 8% and 14%,

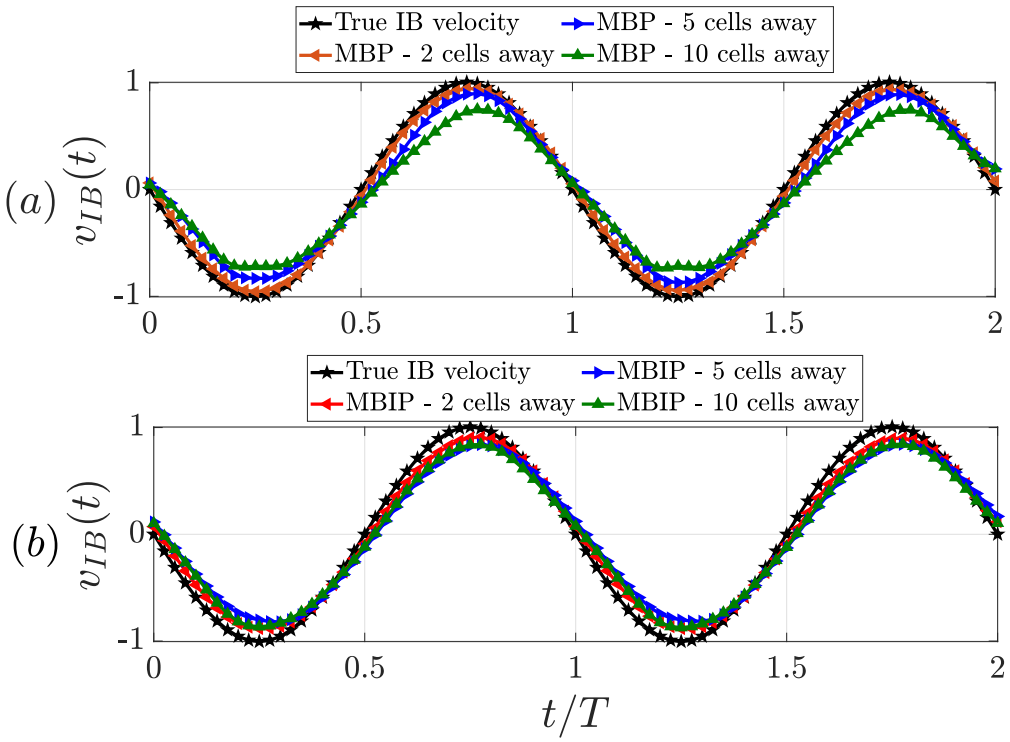


Figure 3.4: Comparison of true IB velocity with IB velocity predictions of (a) MB-PINN (MBP as in the legend) models and (b) MB-IBM-PINN (MBIP) models with $\mathcal{L}_{IB}$ constraint absent ($\lambda_{IB} = 0$). Here, fluid data points are sampled 2, 5, and 10 cells away from the IB. The MB-IBM-PINN results are presented for cases with $\lambda_{IBvar} = 1$ considering a low vorticity strength-based sampling (VF) of fluid region points such that $|\omega_z| < |\omega^*|$ (as in figure 3.1(b)) for the forcing constraint.

respectively, for the data availability case of 2 cells away (table 3.2). With fluid data points sampled further away from the IB, the errors worsen (see figure 3.4), though errors in MB-IBM-PINN do not increase as rapidly as in MB-PINN (table 3.2 and figure 3.4(b)). This could be due to the physics loss being minimized in the entire domain including the solid region in MB-IBM-PINN which further regularises the overall predictions. From the above findings, it can be concluded that both the PINN models in their current form perform satisfactorily for bulk velocity reconstruction, pressure recovery, and recovery of the solid body velocity in the absence of body position and velocity data at training, provided the fluid velocity data is measured close to the solid surface.

Table 3.2: Accuracy details obtained over IB velocity (y component) predictions of the MB-PINN and MB-IBM-PINN models with $\mathcal{L}_{IB}$ constraint, enforced ($\lambda_{IB} = 1$) or absent ($\lambda_{IB} = 0$) while training.

Model	RMSE	MAE	R^2	rRMSE (in %)
MB-PINN ($\lambda_{fluid} = 0.001, \lambda_{IB} = 1$)	6.23e-03	1.51e-03	9.999e-01	**0.88**
MB-PINN ($\lambda_{fluid} = 0.001, \lambda_{IB} = 0$)	5.69e-02	5.12e-02	9.934e-01	**8.06**
MB-PINN ($\lambda_{fluid} = 0.001, \lambda_{IB} = 0$ - 5 cells away)	1.41e-01	1.26e-01	9.599e-01	20.03
MB-PINN ($\lambda_{fluid} = 0.001, \lambda_{IB} = 0$ - 10 cells away)	2.18e-01	1.94e-01	9.044e-01	30.91
MB-IBM-PINN ($\lambda_{fluid} = 0.01, \lambda_{IB} = 1, \lambda_{IBvar} = 1$)	4.87e-03	2.47e-03	9.999e-01	0.69
MB-IBM-PINN ($\lambda_{fluid} = 0.01, \lambda_{IB} = 0, \lambda_{IBvar} = 1$)	1.40e-02	1.22e-01	9.604e-01	19.87
MB-IBM-PINN ($\lambda_{fluid} = 0.01, \lambda_{IB} = 0, \lambda_{IBvar} = 1$(VF))	9.42e-02	8.51e-02	9.822e-01	13.33
MB-IBM-PINN ($\lambda_{fluid} = 0.01, \lambda_{IB} = 0, \lambda_{IBvar} = 1$(VF) - 5 cells away)	1.59e-01	1.43e-01	9.493e-01	22.52
MB-IBM-PINN ($\lambda_{fluid} = 0.01, \lambda_{IB} = 0, \lambda_{IBvar} = 1$(VF) - 10 cells away)	1.36e-01	1.24e-01	9.629e-01	19.25

3.3 BODY SHAPE ESTIMATION USING MB-IBM-PINN

In the previous case study, the body shape and exact geometry were known *apriori* while its position and velocity were unknown during training. However, in this case study, it is assumed that the exact shape is also unknown at training or testing. Hence, the objective here is to determine the solid body shape along with its position, and velocity with the help of the proposed framework. Assuming a rigid body, the solid body's center position needs to be first determined. In MB-IBM-PINN, the momentum forcing term $\boldsymbol{f}$ is expected to be non-zero in the solid region and zero everywhere in the fluid region outside the IB. When the body position and as a result the boundary conditions on the IB are not known, one could use $\boldsymbol{f}$ values to approximately estimate the body location. This is possible because the physics loss constraint in MB-IBM-PINN is enforced throughout the domain including the solid region, and without the fluid-solid partitioning of the constraint. Once the body center position is estimated, a finite differencing can be carried out to obtain the body velocity $\dot{\boldsymbol{x}}_{cg}(t)$.

Here, the squared magnitude of momentum forcing ($|\boldsymbol{f}|^2$) is considered in order to isolate regions of very high $\boldsymbol{f}$. Across all the cases, the MB-IBM-PINN predictions of $|\boldsymbol{f}|^2$ are still localized in the solid region in spite of using the VF sampling strategy for $\mathcal{L}_{IBvar}$(see figure 3.7). To estimate the body center position, for each snapshot of $|\boldsymbol{f}|^2$, one of the following approaches can be used.

1. **Maximum value approach:** Locate where $|\boldsymbol{f}|^2$ is maximum to obtain $\boldsymbol{x}_{cg}(t_j)$ for $j = 1, 2, \cdots N_t$ snapshots.

2. **Simple mean approach:** Locate the N_{cg} grid points located at $\boldsymbol{x}^i(t_j)$ for $i = 1, 2, \cdots N_{cg}$, such that $|\boldsymbol{f}(\boldsymbol{x}^i, t_j)|^2 \geq 0.05(\max\{|\boldsymbol{f}(t_j)|^2\})$. Then calculate a mean of the grid points such that,

$$\boldsymbol{x_{cg}}(t_j) = \frac{\sum_{i=1}^{N_{cg}}(\boldsymbol{x}^i(t_j))}{N_{cg}}, \quad \text{for } j = 1, 2, \cdots N_t. \tag{3.3}$$

3. **Weighted mean approach:** Locate the N_{cg} grid points located at $\boldsymbol{x}^i(t_j)$ for $i = 1, 2, \cdots N_{cg}$, such that $|\boldsymbol{f}^i(t_j)|^2 \geq 0.05(\max\{|\boldsymbol{f}(t_j)|^2\})$. Then calculate a

weighted mean of the grid points such that,

$$\boldsymbol{x_{cg}}(t_j) = \frac{\sum_{i=1}^{N_{cg}}(|\boldsymbol{f}^i(t_j)|^2 \boldsymbol{x}^i(t_j))}{\sum_{i=1}^{N_{cg}}(|\boldsymbol{f}^i(t_j)|^2)}, \quad \text{for } j = 1, 2, \cdots N_t. \tag{3.4}$$

The solid body identification exercise is carried out with the same MB-IBM-PINN models discussed above in subsection 3.1 (without fluid-solid partitioning of physics loss taking $\lambda_{fluid \cup solid}0.01$, and without the $\mathcal{L}_{IB}$ constraint enforced). While training, $\mathcal{L}_{IBvar}$ is enforced in locations with low vorticity strength as described earlier (see the VF sampling approach in figure 3.1(b)). Note that for a rigid body, the velocity of the center of the body can designate the body velocity. The accuracy of the body center position and velocity estimation are presented in tables 3.3 and tables 3.4, respectively. Their time histories for the different approaches mentioned above are presented in figures 3.5 and 3.6, respectively. It can be seen that the mean-based approach outperforms the maximum and the weighted mean-based approaches in estimating the body center. It is seen that as the fluid data points are sampled further away from the IB, the rRMSE values worsen. However, up to 5 cells away from the IB, the rRMSE of body center

Table 3.3: Accuracy details for rigid body center position ($\boldsymbol{x}_{cg}(t)$) estimation using MB-IBM-PINN model with $\mathcal{L}_{IB}$ constraint enforced ($\lambda_{IB} = 1$) and $\mathcal{L}_{IBvar}$ constraint enforced in regions of low vorticity strength (VF sampling) while training with fluid data points away from the IB.

Method	RMSE	MAE	R^2	rRMSE (in %)
2 cells (0.032c) away from IB				
Maximum value	1.73e-02	1.23e-02	9.768e-01	15.23
Simple mean	8.71e-03	6.56e-03	9.941e-01	7.65
Weighted mean	1.05e-02	8.09e-03	9.915e-01	9.21
5 cells (0.08c) away from IB				
Maximum value	2.82e-02	1.88e-02	9.386e-01	24.77
Simple mean	1.60e-02	1.43e-02	9.801e-01	14.09
Weighted mean	6.05e-02	5.01e-02	7.173e-01	53.17
10 cells (0.16c) away from IB				
Maximum value	4.73e-02	4.33e-02	8.269e-01	41.59
Simple mean	3.15e-02	2.98e-02	9.232e-01	27.71
Weighted mean	3.58e-02	3.41e-02	9.011e-01	31.44

Table 3.4: Accuracy details for body velocity estimation (specifically, $\dot{y}(t)$) based on second order finite differencing of the position estimated using the simple mean approach (see figures 3.5(c)-(d) and table 3.3).

Distance from IB	RMSE	MAE	R^2	rRMSE (in %)
2 cells (0.032c)	1.33e-01	8.50e-02	9.645e-01	18.84
5 cells (0.08c)	2.47e-01	1.57e-01	8.772e-01	35.03
10 cells (0.16c)	3.09e-01	2.23e-01	8.088e-01	43.71

position is still within 15% in the simple mean approach.

There are some irregularities in $x_{cg}(t), \dot{x}_{cg}(t)$, and near the peak values of $\dot{y}_{cg}(t)$ which could be due to the higher non-uniformity in $|\boldsymbol{f}|^2$ derived mask at those times (see figures 3.7(b), 3.5 and 3.6 corresponding to $t/T = 0.25$). As a result, the errors are higher than 10% as seen in table 3.4, even for the case with fluid data sampled 2 cells away. Given that the fluid data (especially in the wake) is smooth across time, by assuming that such sudden jumps in $x_{cg}(t)$, and $\dot{\boldsymbol{x}}_{cg}(t)$ might not be physical, smoothing the time histories is possible. However, to test the capability of the framework, any such smoothing has been avoided here.

As the data points are sampled further and further away from the IB, information of the actual shape of the body is lost resulting in a smudging of $|\boldsymbol{f}|^2$ around the true IB (see figure 3.7). In spite of this, the body center position is estimated reasonably well. Moreover, the simple mean approach estimates the body center more closely to the true body center. This is because it gives equal weightage to the grid points in the masked region extracted using the thresholding of $|\boldsymbol{f}|^2$. However, the other approaches did not estimate the body position so well because of $|\boldsymbol{f}|^2$ being non-uniform in the region where the solid body could possibly lie.

Subsequently, the shape estimation can be carried out as an optimization problem in the following manner.

1. From $|\boldsymbol{f}(t)|^2$, extract snapshot-wise masks $M_f(t)$ such that $M_f(t) = 1$ wherever

$|\boldsymbol{f}(\boldsymbol{x}^i, t_j)|^2 \geq 0.05(\max\{|\boldsymbol{f}(t_j)|^2\})$, and zero otherwise.

2. Using the estimated rigid body velocity, extract the edges $E_{\boldsymbol{u}_{IB}}$ from the velocity contours such that $E_{\boldsymbol{u}_{IB}} = 1$, where the contours match with either the individual components or the magnitude of body velocity.

3. Then, for each snapshot, extract grid points wherever both $M_f(t) = 1$ and $E_{\boldsymbol{u}_{IB}}(t) = 1$. This gives the potential IB marker locations.

4. At each time step, subtract the corresponding body center position from the potential IB marker location coordinates and center the body around $(0, 0)$.

5. Then, superimpose them to obtain the cluster of $N_{IB'}$ possible IB marker locations $(\{\boldsymbol{x}^i_{IB'} = (x^i_{IB'}, y^i_{IB'})\}_{i=1}^{N_{IB'}})$.

6. From this cluster of $N_{IB'}$ points, one can either assume a parameterized geometry form or obtain a best fit using splines. Here, the former approach is adopted as an

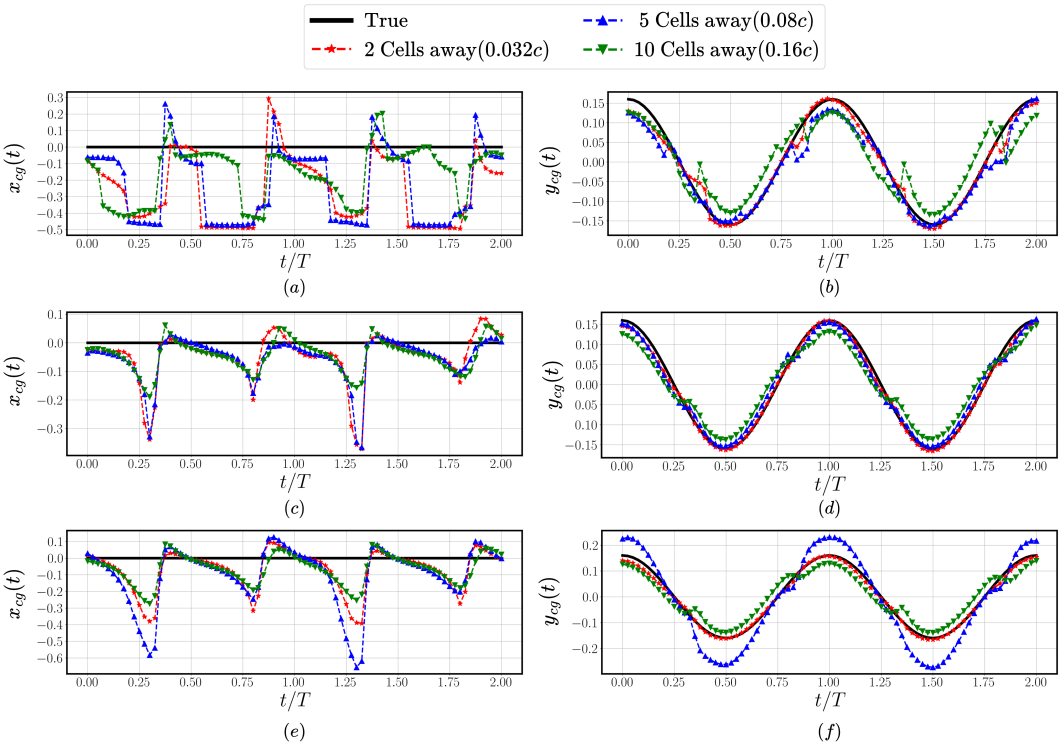


Figure 3.5: Estimation of body position $(x_{cg}(t), y_{cg}(t))$ from the squared magnitude of momentum forcing term $(|\boldsymbol{f}|^2)$ using the (a-b) maximum value, (c-d) simple mean and (e-f) weighted mean approaches, respectively for fluid data sampled at different distances from the IB. The simple mean approach notably outperforms the other approaches.

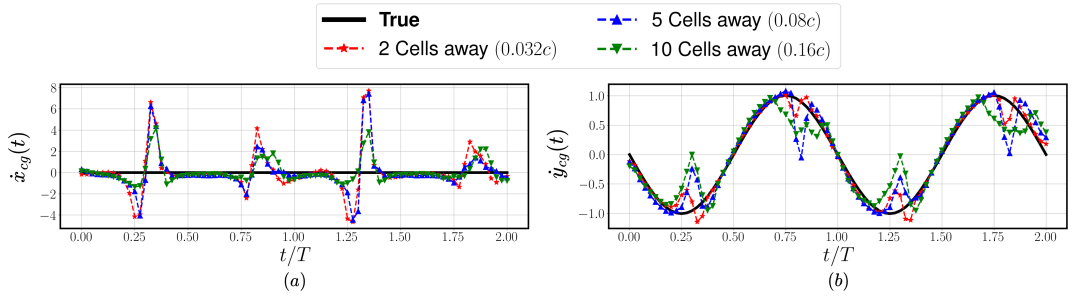


Figure 3.6: Estimation of body velocity $(\dot{x}_{cg}(t), \dot{y}_{cg}(t))$ based on second-order finite differencing of the position estimated using the simple mean approach (see figures 3.5(c)-(d) and table 3.3).

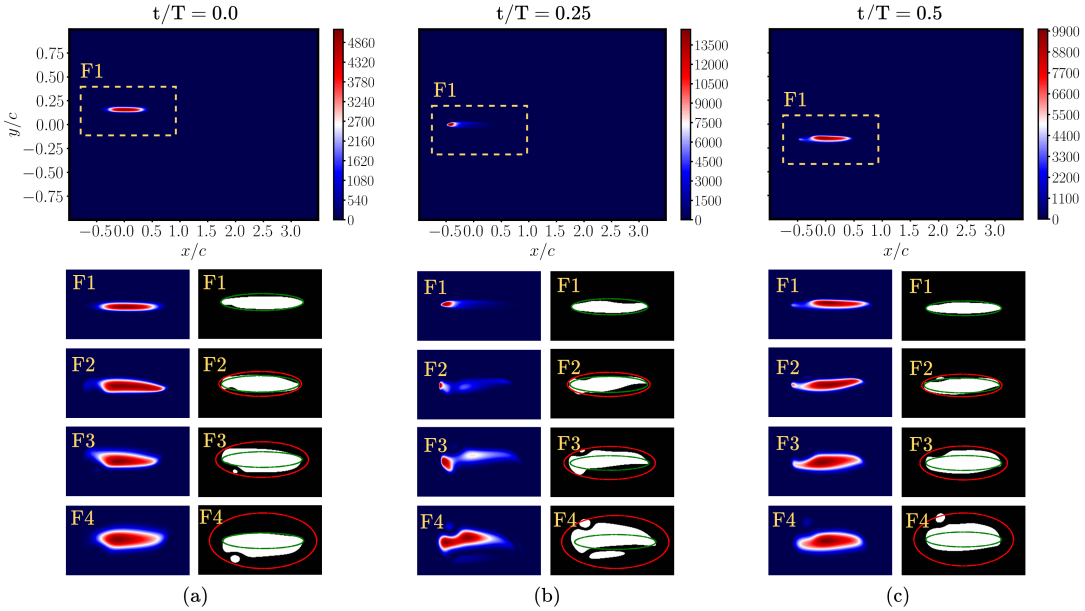


Figure 3.7: MB-IBM-PINN predicted contours of squared magnitude of momentum forcing term ($|\boldsymbol{f}|^2$) and associated masks generated for the condition $|\boldsymbol{f}^i(t)|^2 > 0.05 \max\{|\boldsymbol{f}(t)|^2\}$, for $i = 1, 2. \cdots N_{cg}$ presented for one plunging downstroke at timestamps (a) $t/T = 0.0$, (b) $t/T = 0.25$, and (c) $t/T = 0.5$. The topmost row and the box F1 correspond to the MB-IBM-PINN model with $\lambda_{IB} = 1, \lambda_{IBvar} = 1$ and no VF based sampling of grid points for enforcing $\mathcal{L}_{IBvar}$. The boxes F2-F4 represent the MB-IBM-PINN models with $\lambda_{IB} = 0, \lambda_{IBvar} = 1$ but with VF and fluid data points sampled 2(0.032c), 5 (0.08c) and 10 (0.16c) cells way from the IB, respectively. Note in the mask contours that the green color elliptical boundary is the actual IB and outside the red elliptical boundary is where the fluid data are sampled for cases in F2-F4.

example.

7. To estimate the geometry parameters, an objective function $\mathcal{L}_{shape}$ dependent on the geometry parameters can be minimized using an optimization algorithm.

To demonstrate the above steps, the MB-IBM-PINN model trained with fluid data points sampled about 5 cells away is chosen. First, the snapshot-wise masks and the edges from velocity contours matching body velocity are extracted as described above (see figures 3.7 and 3.8). A set of $N_{IB'}$ possible IB marker locations ($\{\boldsymbol{x}^i_{IB'}\}_{i=1}^{N_{IB'}}$) are then obtained (see figure 3.9).

Notably, these grid points when obtained using the edges extracted by matching individual body velocity components, seem to be contained within an elliptical boundary (see figures 3.9(a)-(b)). This is because MB-IBM-PINN within the solid region, predicts a uniform velocity close to the body velocity (see figures 3.8(a)-(b)). By using the body velocity magnitude for edge extraction, the grid points towards the trailing edge are seen to define the body geometry very crisply (see figures 3.9(c) and 3.10).

Based on the above findings, an elliptical geometry is assumed for the body. The following objective function $\mathcal{L}_{shape}$ is designed to estimate the unknown geometric parameters using the extracted IB points $\boldsymbol{x}_{IB'}$.

$$\mathcal{L}_{shape} := \arg_{[a,b]} \min\{|(x_{IB'}/a)^2 + (y_{IB'}/b)^2 - 1|^2\}, \tag{3.5}$$

where, a and b denote the lengths of semi-major and semi-minor axes, respectively. The $\mathcal{L}_{shape}$, is optimized using a simple grid search algorithm over the parameter space, chosen such that $a \in [0.01c, 1c]$ and $b \in [0.01c, 1c]$ are discretised into a 201×201 uniform grid. The errors in the prediction of the semi-major and semi-minor axes are approximately, 11% and 35%, respectively. While the body thickness is over-predicted, considering the fact that the fluid data points are sampled 5 cells *i.e.* $0.08c$ away from the IB, the overshoot of the thickness prediction is still less than this value. This is quite impressive and is expected to improve with further improvements in the velocity

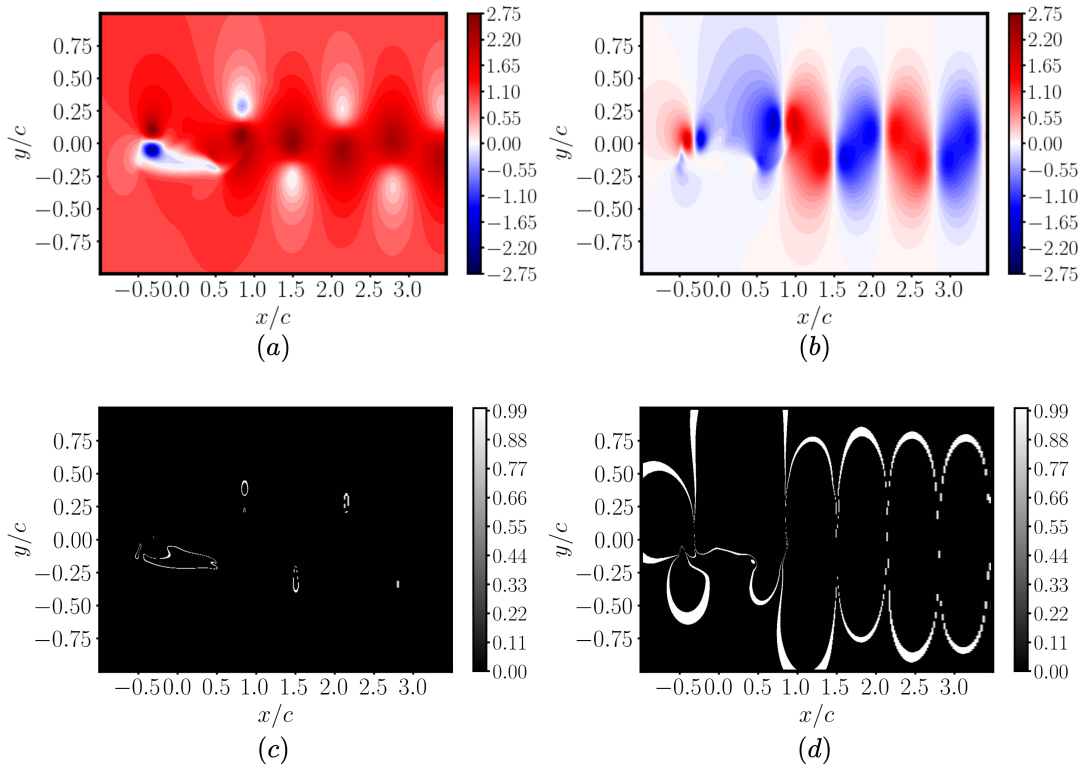


Figure 3.8: Edge extraction from MB-IBM-PINN predicted velocity contours by finding the contours matching the estimated rigid body velocity. MB-IBM-PINN Reconstructed contours of (a)x-component and (b) y-component velocity at $t/T = 0.5$. Here, $\mathcal{L}_{IB}$ constraint is not enforced, and $\lambda_{fluid \cup solid}0.01$ with fluid data points 5 cells away from IB. Note how the edges are not unique to the solid body region.

reconstruction of MB-IBM-PINN. However, in case of more complex kinematics or geometries, this would be quite a challenging task.

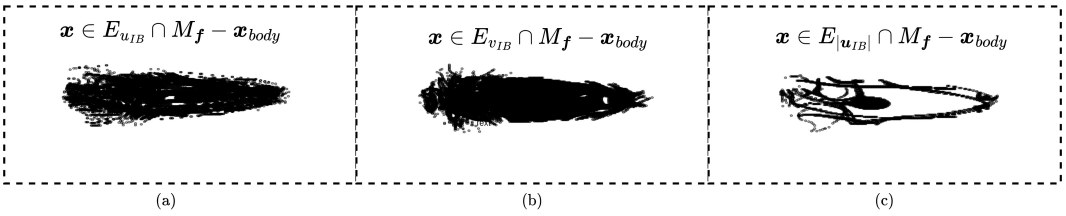


Figure 3.9: Comparison of different edge extraction approaches to obtain the potential IB points by matching (a) $u_{IB} = \dot{x}_{cg}(t)$, (a) $v_{IB} = \dot{y}_{cg}(t)$ and (c) $|\boldsymbol{u}_{IB}| = |\dot{\boldsymbol{x}}_{cg}(t)|$.

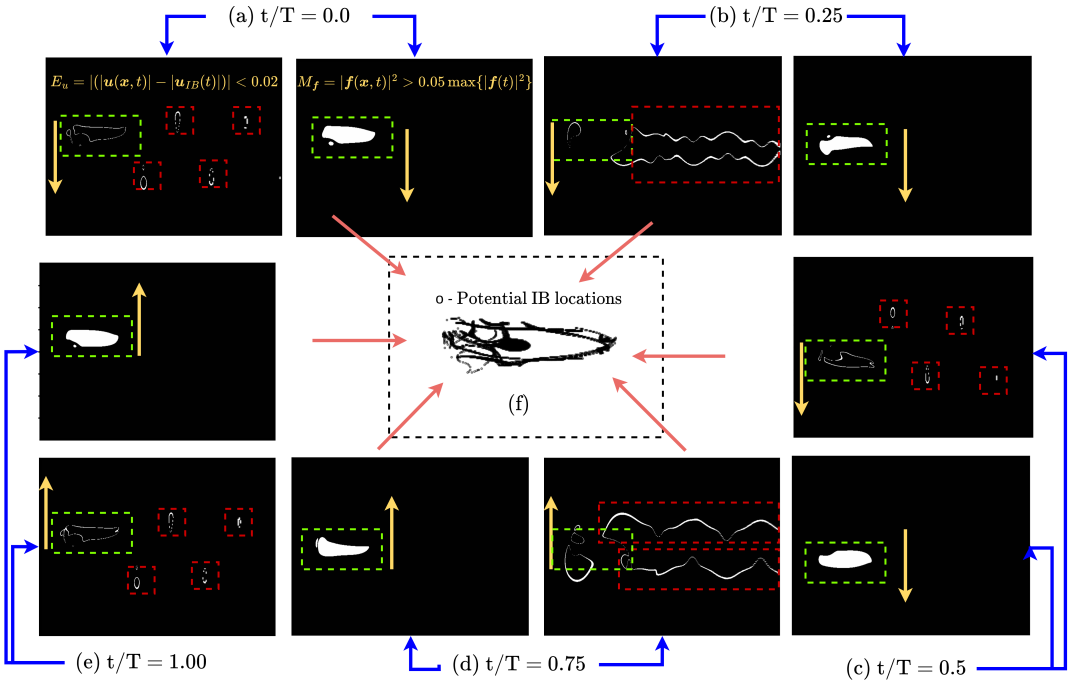


Figure 3.10: Detailed representation of (a)-(e) extracting the potential IB marker locations by combining the body velocity magnitude based edge extraction ($E_{|\boldsymbol{u}_{IB}|}$) with the masks $M_f(t)$ at different time instants. The extracted potential IB marker locations are depicted in (f).

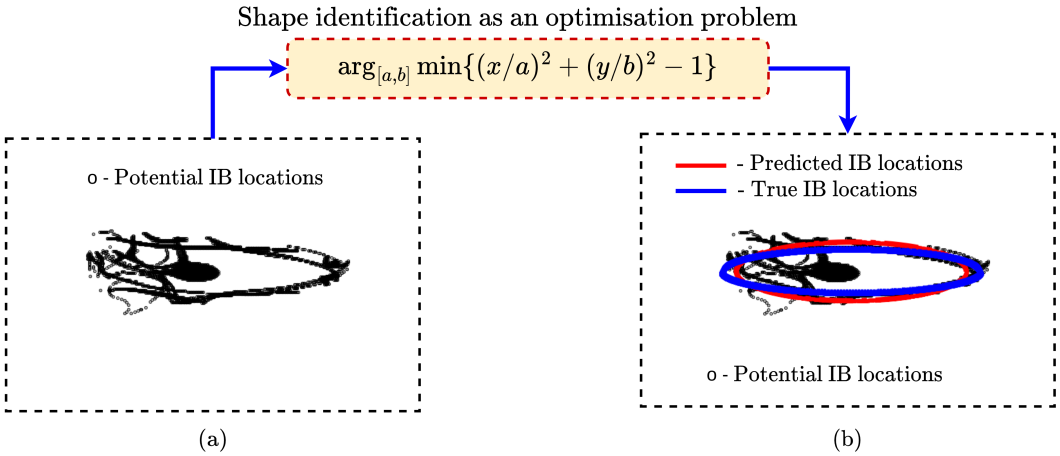


Figure 3.11: Shape estimation as an optimization problem. (a) The extracted potential IB marker locations and (b) overlay of the true and predicted IB.

In such complex situations, further improvements of the proposed framework in terms of model architecture, and loss formulations with additional shape-related constraints would be necessary to identify the shape of the IB. It might also be possible to make use of the trained model in conjunction with image segmentation/edge detection techniques such as Sobel filters (Gao *et al.*, 2010) or active contour matching algorithms (Menet *et al.*, 1990) to extract the approximate shape of the body. Such techniques to estimate the body shape and location, from coarse fluid velocity data in conjunction with PINN models are left for future work. Having explored the potential of the framework for identifying hidden boundaries and estimating body shapes, we now turn to another fundamental challenge—recovering fine-scale flow features from coarse data. The next section investigates the use of the IBA framework for physics-consistent super-resolution, focusing on high-resolution velocity and pressure reconstruction from low-fidelity inputs.

3.4 HIGH-RESOLUTION PRESSURE RECOVERY AND VELOCITY RECONSTRUCTION USING LOW-FIDELITY SIMULATION DATA

Super-resolution of flow-fields in a physically consistent manner is a challenging task, where given low-fidelity training data, one employs deep learning to effectively predict the high-resolution data (Fukami *et al.*, 2019; Aliakbari *et al.*, 2022; Fukami *et al.*, 2023). However, it would be worthwhile to investigate the efficacy of the IBA framework for such tasks. Thus, the objective here is to thus to test the framework in pressure recovery and high-resolution velocity reconstruction from low-fidelity data. Here, MB-PINNs are considered since they were shown to be excellent in bulk velocity reconstruction and pressure recovery. To test this, IBM simulations are performed with a low-resolution mesh 4 times coarser than the Ref-IBM grid (the high-resolution grid from our earlier investigations). The coarseness level was chosen to maintain parity with the grid from the coarse interpolated dataset (i.e. CI from the earlier investigations when downsampling was done). Notably, the CI grid considered in section 2.6.1 was 4 times coarser than the Ref-IBM grid (with $\Delta x = \Delta y = 0.016$ for CI and with $\Delta x = \Delta y = 0.004$ for Ref-IBM,

respectively). The low-fidelity velocity flow-field data is then extracted for the truncated domain as described in section 2.6.1. An MB-PINN model is subsequently trained on this low-fidelity data (referred to as Ref-IBM-LF, where LF stands for low fidelity) while minimizing physics loss on the high-resolution Ref-IBM grid (as shown in the workflow in figure 3.12). Finally, the model predictions are made on the high-resolution Ref-IBM grid for validating velocity reconstruction and on Ref-ALE for testing pressure recovery.

The model architecture, hyperparameters, testing datasets, and the training budget are kept the same as in earlier studies presented in Chapter 2. Since the underlying data quality is fundamentally much lower than that of the downsampled high-resolution data, the same settings might not be optimal for obtaining predictions with similar order of accuracy. Note that the Ref-IBM-LF data is less accurate compared to the CI dataset in

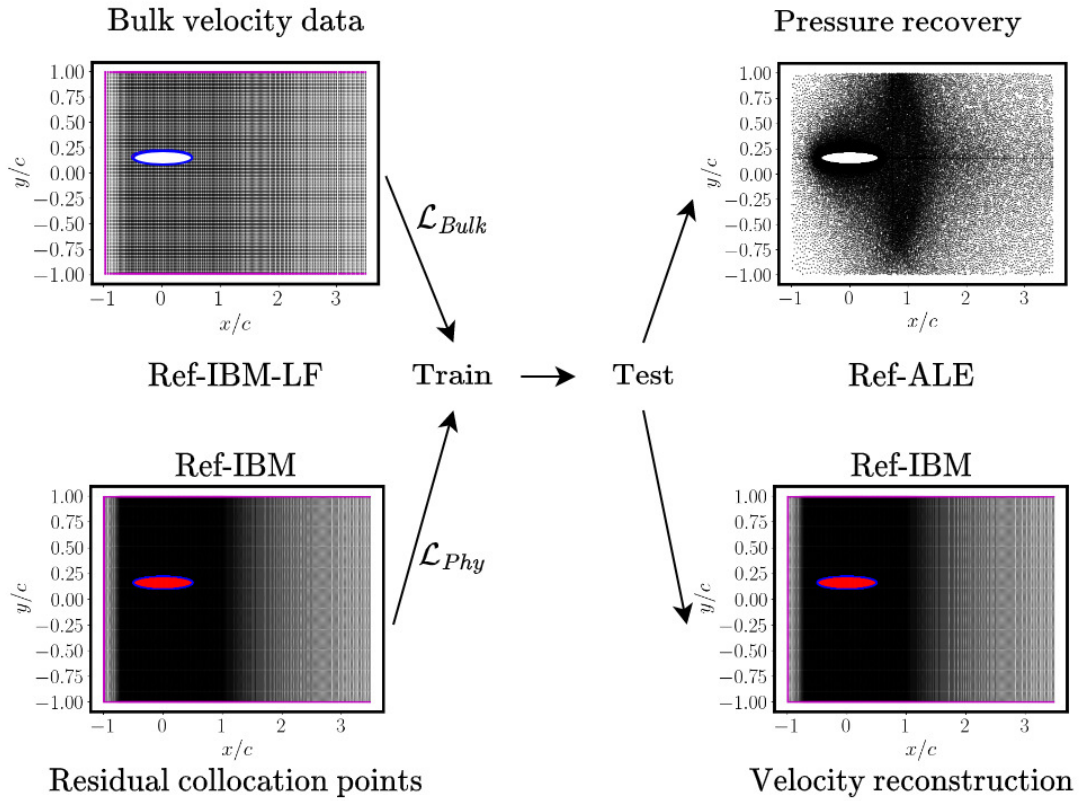


Figure 3.12: Workflow for the super-resolution example study

Chapter 2 as the CI data set was obtained by downsampling the high-resolution dataset. Therefore, the linear interpolation of velocity data from CI to Ref-IBM results in lower errors as compared to that of linear interpolation of velocity data from Ref-IBM-LF to Ref-IBM (see table 3.5). Even though the velocity reconstruction and pressure recovery errors are found to be less than 10% and 15%, respectively, for the models trained on Ref-IBM-LF (the coarse mesh data), they are higher than those studies reported in sections 2.7 and 2.8.

Here, velocity reconstruction from PINN models is found to be still comparable with linear interpolation and not significantly better (see table 3.5). In point-wise error contour plots shown in figure 3.13 for the test time stamp, $t/T = 0.375$, the errors are high in the regions of strong vorticity for both velocity reconstruction and pressure recovered

Table 3.5: Accuracy details for the super-resolution example case study. Here, C2F stands for training on the low-fidelity dataset Ref-IBM-LF and predicting on the high-resolution Ref-IBM grid.

Model	RMSE	MAE	R^2	rRMSE (in %)
Linear-C2F		$S_{\omega_z} = 100\%$		
u	5.6e-02	2.2e-02	9.974e-01	4.89
v	4.9e-03	2.0e-02	9.899e-01	9.89
MB-PINN-C2F		$\lambda_{fluid} = 0.001$		
u	5.5e-02	2.2e-02	9.976e-01	4.82
v	4.9e-02	2.0e-02	9.891e-01	9.86
p	1.7e-01	8.3e-02	9.810e-01	12.97
MB-PINN-C2F		$\lambda_{fluid} = 0.01$, $S_{\omega_z} = 5\%$		
u	5.6e-02	2.2e-02	9.975e-01	4.87
v	5.0e-02	2.2e-02	9.897e-01	10.02
p	1.9e-01	1.0e-01	9.776e-01	14.30
MB-PINN-C2F		$\lambda_{fluid} = 0.001$, $N_{iter} = 7.5e05$		
u	5.5e-02	2.2e-02	9.976e-01	4.82
v	4.9e-02	2.0e-02	9.901e-01	9.81
p	1.6e-01	7.4e-02	9.825e-01	12.26
MB-PINN-C2F		$\lambda_{fluid} = 0.01$, $S_{\omega_z} = 5\%$ $N_{iter} = 7.5e05$		
u	5.5e-02	2.3e-02	9.976e-01	4.82
v	4.9e-02	2.2e-02	9.898e-01	9.94
p	1.7e-01	8.8e-02	9.807e-01	13.00

on the high-resolution Ref-IBM and Ref-ALE grids, respectively. Notably, in table 3.5 and the error contours in figure 3.13, larger training iterations and vorticity cut-off based sampling do not improve the predictions significantly, and the error contours are in fact indistinguishable from each other.

It is hypothesized that to truly achieve super-resolution, the physics constraints over the high-resolution Ref-IBM grid must be better satisfied. This can only be attested by a significant improvement in pressure recovery. Moreover, since the underlying data quality is fundamentally much lower than before, the same hyperparameter settings used earlier might not be optimal as observed here. As there are many hyperparameters that the model predictions might be sensitive to, instead of hand-tuning these, adaptive

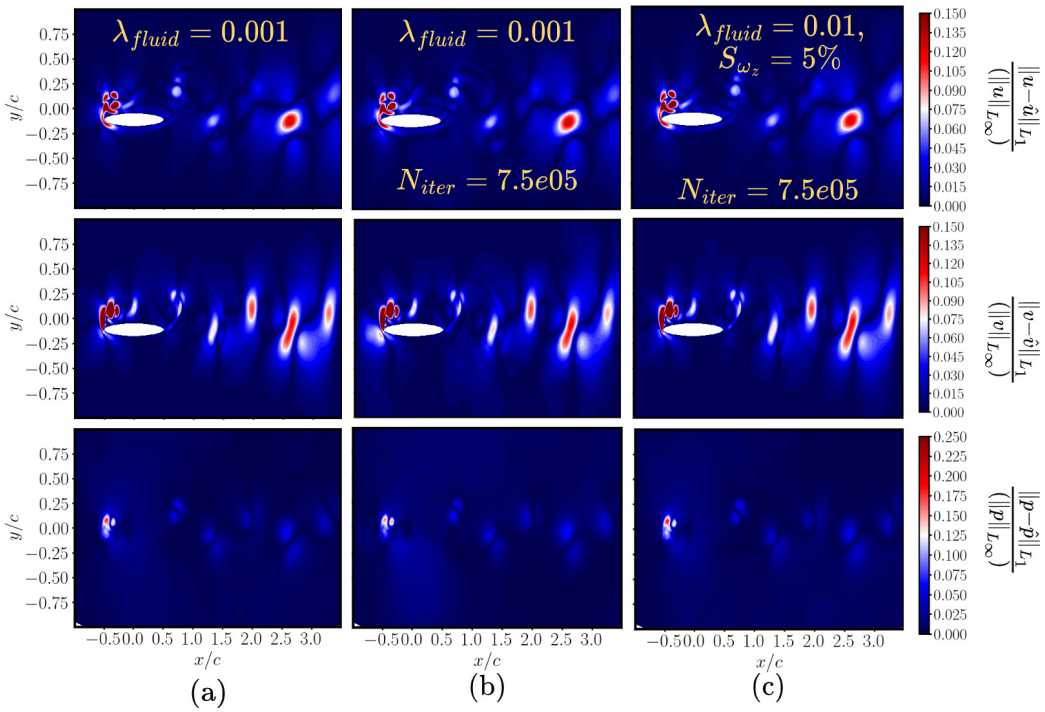


Figure 3.13: Comparison of maximum value normalized pointwise absolute error contours obtained over MB-PINN-C2F predictions on the Ref-IBM (velocity data) and Ref-ALE (pressure data) considering (a) $\lambda_{fluid} = 0.001, N_{iter} = 5e05$, (b)$\lambda_{fluid} = 0.001, N_{iter} = 7.5e05$, and (c) $\lambda_{fluid} = 0.01, S_{\omega_z} = 5\%, N_{iter} = 7.5e05$, respectively.

approaches need to be used. Especially, with a larger network size, batch size along with adaptive loss component weighting and adaptive residual sampling, one could obtain better results than those presented here with the same model architecture. Following Aliakbari *et al.* (2022), one could also adopt a transfer learning approach to pre-train on low-fidelity data, and then, fine-tune the physics on the high-resolution grid. This requires an expansive investigation for unsteady flows past moving bodies which however, is out of scope in the present study.

While the preceding sections focused on data-driven reconstruction tasks—ranging from hidden boundary estimation to super-resolution—the natural next step is to evaluate the potential of the proposed framework in a data-free setting. To this end, the following section examines the performance of IBA framework in addressing a forward problem in a truncated spatio-temporal domain, where the governing equations are solved solely from initial and boundary conditions, without any velocity training data.

3.5 PINNS FOR A FORWARD PROBLEM WITH A MOVING BOUNDARY

In a forward problem, the system governing PDEs are solved, given only the initial and boundary conditions and no data (Raissi *et al.*, 2019*a*; Sun *et al.*, 2020). All the different situations investigated above, and in Chapter 2 were data-driven. However, it would be interesting to test the capability of the proposed PINN framework in solving a forward problem as well, where no velocity training data are available. So far, such a problem has not been attempted in the context of moving body flows. As a proof of concept exercise, given only initial and boundary conditions on the truncated domain and moving solid boundary, MB-PINN and MB-IBM-PINN are used here to infer the solution to the governing equations (see equations (2.5) and (2.6) of section 2). Results suggest that the models are extremely sensitive to the number of iterations and model initialization. In the current form, the performance of the MB-PINN and MB-IBM-PINN models are not very satisfactory. Although MB-PINN is seen to significantly outperform

Table 3.6: Testing data set resolution within the truncated domain considered for the forward problem.

Data sets	N_x	N_y	N_t	$\Delta t/T$	N_{Bulk}
Ref-IBM	651	500	5	0.025	1.6274e06
Ref-ALE	-	-	5		2.5e05

MB-IBM-PINN, the errors are still high due to the challenging nature of the unsteady flow problem involving a moving body. Initially a time domain of $t/T \in [0, 2]$ (as used for the data-driven PINNs studies in Chapter 2) was considered, but the forward problem failed to converge to the true solution in spite of training for $N_{iter} = 1.5e05$ iterations, whereas, for the data-driven PINNs, the models started converging to less than 10% error by about $N_{iter} = 2e04$ iterations in the first training cycle itself. Subsequently, a smaller temporal domain of $t/T \in [0, 0.1]$ has been chosen for the forward problem, as standard PINNs are often difficult to train on longer-time domains for solutions having strong gradients or evolving boundaries. This is due to the propagation failure mode of PINNs as it has been reported in the literature (Krishnapriyan *et al.*, 2021; Mattey and Ghosh, 2022; Penwarden *et al.*, 2023). Sequential/causal learning frameworks have been proposed for such problems (Wang *et al.*, 2024; Penwarden *et al.*, 2023).

The spatial domain considered for this study is $\Omega = [-1c, 3.5c] \times [-1c, 1c]$ (which is the same as what was used for the data-driven PINNs studies in Chapter 2). The CI grid (270×120 grid points) is considered and a time step size of $\Delta t/T = 0.025$ is selected. Overall, in the fluid region, there are about 1.593e05 residual collocation points. The details of the test data set are provided in table 3.6. Keeping the network width and depth, the batch size, and the number of training stages the same as in the data-driven studies, the results are sensitive to the number of iterations in each training stage, despite the smaller time domain chosen. Although not presented here for all the $N_{iter} \in [2.5e04, 5e04, 1e05, 5e05]$, $N_{iter} = 5e04$ is found to be better than the rest. The accuracy details for the $N_{iter} = 5e04$ case are presented in table 3.7 for both MB-PINN and MB-IBM-PINN (without fluid-solid partitioning of the physics loss). Although

the qualitative results of the velocity show some similarity with that of the true data at an intermediate time stamp, $t/T = 0.075$ (figure 3.15), the relative RMSEs are still reasonably high (table 3.7), and these errors grow over every successive snapshot (figure 3.14). The models are not able to predict the velocity field with high accuracy due to the presence of a moving body and strong gradients surrounding the moving body for the time domain considered in the example case. This in turn affects the pressure recovery significantly. The error contours in figure 3.16 indicate that the highest errors are mostly localized around the moving body for MB-PINN, but they extend into the surrounding regions for MB-IBM-PINN especially for pressure. Recent studies by Krishnapriyan *et al.* (2021); Mattey and Ghosh (2022); Wang *et al.* (2024); Penwarden *et al.* (2023) suggest that adaptive sampling/weighting, or a causal/sequential learning framework could mitigate the propagation failures in time for PDEs exhibiting solutions with strong time-varying gradients/localized features. This has been taken up in the next chapter where, sequential learning techniques are used to enhance the proposed IBA framework to tackle various temporal domain complexities arising in the context of unsteady flows past moving bodies.

Table 3.7: Accuracy details of the best MB-PINN and MB-IBM-PINN models for the forward problem.

Model	Accuracy			
	$\lambda_{fluid} = 0.01, N_{iter} = 5e04$			
MB-PINN	RMSE	MAE	R^2	rRMSE (in %)
u	6.4e-02	2.7e-02	9.958e-01	5.65
v	6.7e-02	2.6e-02	9.708e-01	14.96
p	5.3e-01	3.8e-01	8.688e-01	35.19
	$\lambda_{fluid} = 0.01, N_{iter} = 5e04$			
MB-IBM-PINN	RMSE	MAE	R^2	rRMSE
u	1.1e-01	7.3e-02	9.883e-01	9.35
v	1.4e-01	8.2e-02	8.821e-01	29.97
p	2.26	1.87	-1.2883	151.18

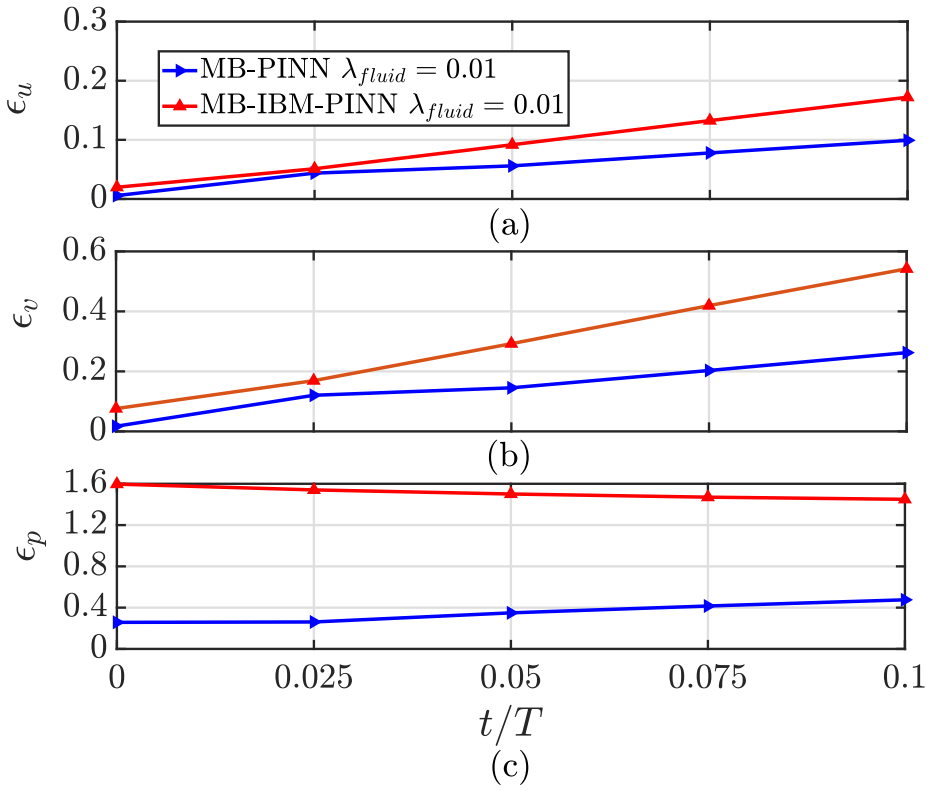


Figure 3.14: Snapshot-wise relative error for velocity and pressure predictions obtained from MB-PINN and MB-IBM-PINN models for $N_{iter} = 5e04$ for the forward problem.

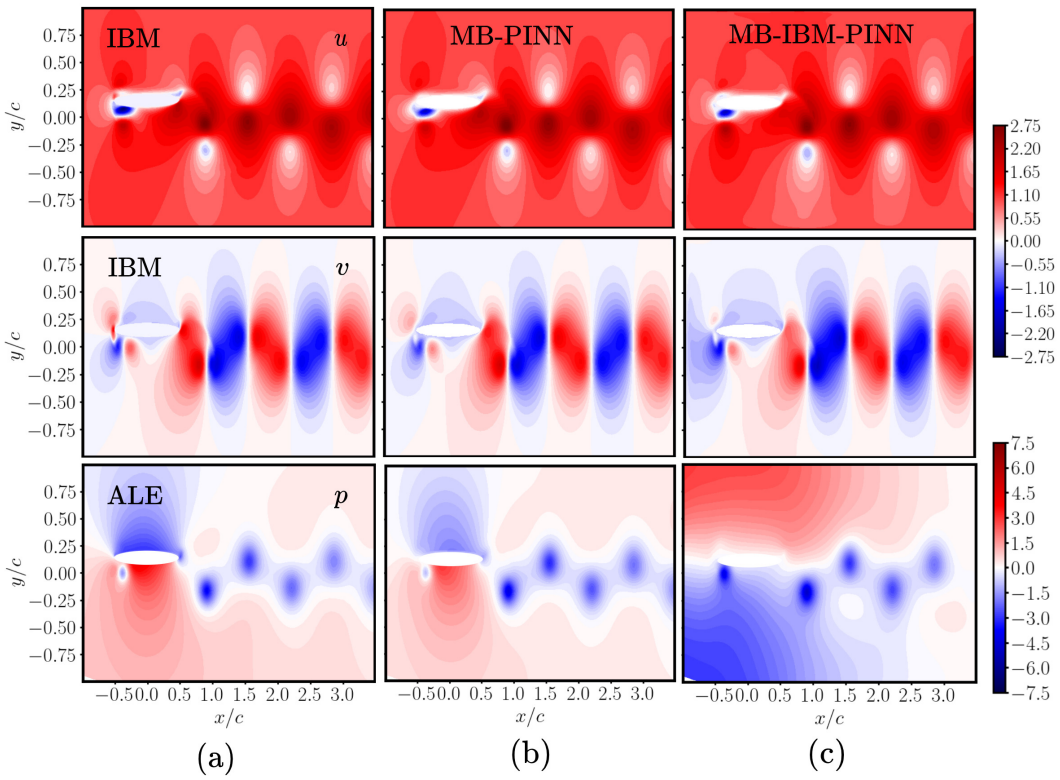


Figure 3.15: Comparison of (a) true and (b)-(c) predicted velocity and pressure contours at $t/T = 0.075$ obtained from MB-PINN and MB-IBM-PINN models for the forward problem.

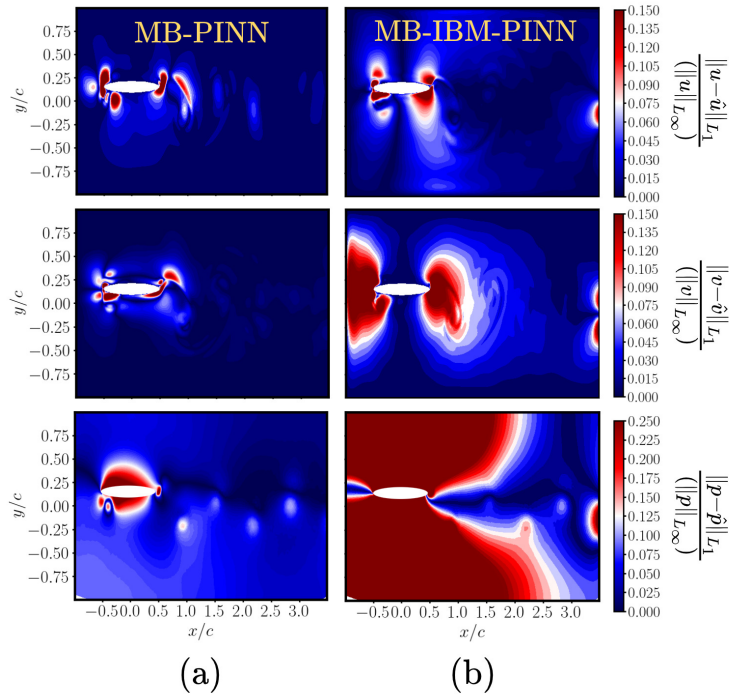


Figure 3.16: Comparison of maximum value normalized absolute error contours at $t/T = 0.075$ for MB-PINN and MB-IBM-PINN models for the forward problem.

3.6 DISCUSSION AND PERSPECTIVES

In this chapter, the efficacy of the PINN variants under the IBA framework proposed earlier in Chapter 2 was evaluated through a series of proof-of-concept studies addressing progressively challenging data scenarios. The first investigation focused on a hidden boundary estimation problem where neither the solid-body position nor velocity data were known a priori. Such conditions are typical in experimental and field situations, where only fluid velocity measurements, taken away from the body surface, are available. The goal was to reconstruct the surrounding flow field, recover the body kinematics, and in the second problem approximate the body shape with some geometry priors.

For these situations, the MB-IBM-PINN variant proved advantageous because, in addition to velocity and pressure, it also predicts the momentum forcing ($\boldsymbol{f}$) and mass source/sink (q) terms that have support predominantly in the solid region. These predictions can serve as useful indicators of the immersed body's approximate position and motion. Using this property, the model was able to estimate the body center, velocity, and shape, even when only fluid-domain data were provided. A rigid body assumption simplified the extraction of motion characteristics, and a grid-based optimization enabled the estimation of elliptical shape parameters with promising accuracy. The predicted semi-minor axis error remained smaller than the minimum distance from the nearest available data point, underlining the potential of IBA framework in non-ideal scenarios where MB-PINNs alone would fail.

Nevertheless, for more complex kinematics or deforming geometries, shape estimation remains challenging. Future extensions could incorporate shape-aware loss formulations or use the trained PINN models in conjunction with image segmentation or edge-detection algorithms (Menet *et al.*, 1990; Gao *et al.*, 2010) to improve solid-boundary reconstruction.

A third problem explored was super-resolution of flow fields (Fukami *et al.*, 2023),

where the objective was to reconstruct high-fidelity velocity and recover pressure from low-resolution simulation data. In this case, low-resolution IBM simulations were directly used for training, making the underlying data quality fundamentally poorer than before. The MB-PINN model employed low-resolution velocity data for the bulk loss ($\mathcal{L}Bulk$), while the physics loss ($\mathcal{L}Phy$) was computed using collocation points from the high-resolution reference grid. Results indicated that truly achieving super-resolution requires the model to more accurately satisfy the physics residuals on the fine grid, likely demanding adaptive loss weighting (Wang *et al.*, 2021*a*) and residual sampling strategies (Wu *et al.*, 2023). Although such adaptive schemes could improve accuracy, they introduce additional computational overheads.

Finally, to test the framework in a purely physics-driven context, a forward problem was formulated in which no training data were provided in the bulk domain. Here, the governing PDEs were solved given only the boundary and initial conditions over a truncated spatial domain. As expected, this proved significantly more difficult, especially for the flapping-wing flow-field characterized by strong gradients and moving boundaries. The models yielded moderate errors in velocity (10–15%) and pressure (30%), pointing to the need for more stable and temporally aware training schemes such as sequential or causal learning frameworks (Wang *et al.*, 2024; Penwarden *et al.*, 2023).

Overall, the studies reaffirm that MB-PINNs are more efficient when body position and velocity are known *apriori*, while MB-IBM-PINNs extend applicability to data-sparse or uncertain boundary conditions. The experiments collectively underscore the adaptability of the IBA framework to a wide range of real-world data scenarios—from hidden boundaries to low-resolution inputs and fully data-free simulations.

3.7 SUMMARY AND CONCLUSIONS

This chapter demonstrated the versatility of the Immersed Boundary Aware (IBA) framework for several inverse and forward flow problems involving moving bodies.

It established that in non-ideal scenarios, where body position, velocity and shape are unknown, MB-IBM-PINNs have potential in simultaneous velocity reconstruction, pressure recovery, and hidden boundary estimation making them valuable for experimental and partially observed systems. The framework can, in principle, be extended in future to handle high-fidelity reconstruction from low-resolution data through adaptive loss formulations and transfer learning. However, in a data-free setting such as the forward problem, the IBA framework performance highlights the need for temporally consistent learning strategies. It is important to also note that, to solve the forward problem meaningfully, a full spatial domain might be needed, and the time domain needs to be really small over which the network is trained. This challenge makes it intractable to train the PINN models for forward problems, and traditional GPU accelerated solvers would still triumph over the PINN models in such scenarios. However, given the importance of pressure recovery to the unsteady aerodynamics community, one would encounter different temporal domain complexities when working with unsteady flow-field datasets. These temporal domain complexities typically manifest as temporal sparsity, long time domain and rich temporal spectra due to aperiodicity in the flow. Hence, in the subsequent chapter, the IBA framework is extended using sequential learning techniques to handle temporal domain complexities arising in unsteady flows keeping MB-PINNs as the backbone.

CHAPTER 4

SEQUENTIAL LEARNING BASED PINNS TO OVERCOME TEMPORAL DOMAIN COMPLEXITIES IN UNSTEADY FLOW PAST FLAPPING WINGS

4.1 INTRODUCTION

In the previous chapter, the Immersed Boundary Aware (IBA) framework and its variants, namely MB-PINN and MB-IBM-PINN, were shown to effectively reconstruct velocity and recover pressure fields for moving-body flows, even under data-sparse or unknown moving boundary conditions. While these methods addressed the spatial complexities introduced by immersed and moving boundaries, the next major challenge arises from temporal domain complexities, which severely impact the learnability and generalization capability of physics-informed models.

As shown previously, Physics-Informed Neural Networks (PINNs) (Raissi *et al.*, 2019*a*) offer a powerful approach to learning spatiotemporal fields governed by partial differential equations, blending data-driven fitting with physical regularization. However, their training becomes increasingly unstable for long temporal domains or unsteady flows due to well-documented issues such as spectral bias (Wang *et al.*, 2021*b*), and the inherent difficulty of learning consistent temporal dynamics (Krishnapriyan *et al.*, 2021; Wang *et al.*, 2024). These problems can become particularly pronounced for moving-body systems, where domain evolution, nonlinear coupling, and sharp flow gradients make global time-domain optimization highly ill-conditioned.

For such flows, particularly unsteady or aperiodic wake interactions past flapping foils (Khalid *et al.*, 2018; Bose and Sarkar, 2018; Majumdar *et al.*, 2022), the challenge stems from the multi-scale temporal dynamics and data sparsity inherent in data collected

from simulations or experiments. Due to many constraints, the obtained data may often fail to meet the Nyquist sampling criterion, leaving only temporally coarse snapshots not suitable for complete flow characterization. Standard interpolation techniques cannot faithfully reconstruct flow fields at intermediate time steps, especially when the flow exhibits rich spectral content and when the available data snapshots are quite far apart in time. Moreover, re-running simulations or conducting experiments to densify the data is often impractical. Thus, continuous space-time PINNs provide a compelling alternative—they can interpolate missing time states while respecting the governing physics (Cai *et al.*, 2021*b*; Raissi *et al.*, 2020).

However, such problems demand training strategies that respect temporal causality while maintaining stability over long integrations. Recent works have explored sequential and causal learning frameworks (Mattey and Ghosh, 2022; Penwarden *et al.*, 2023) that divide the temporal domain into smaller segments or time windows and train networks either in a time-marching or transfer-learning manner. Unlike parallel decomposition methods, sequential schemes preserve temporal continuity and can exploit the transferability of neural network representations across adjacent time segments. At the same time, more discrete-time formulations—such as explicit time-domain PINNs or physics-informed LSTMs (Su *et al.*, 2024; Liu *et al.*, 2023)—offer advantages for purely forward problems but sacrifice the continuous space-time modeling capability critical for data-sparse reconstruction tasks involving moving bodies.

In light of these challenges, the present chapter explores sequential learning-based extensions of MB-PINNs to efficiently handle temporal domain complexities arising from: 1) temporal sparsity of flow-field data, 2) long time-domain integration, and 3) rich spectral content associated with aperiodicity.

For flow past a plunging foil at a low Reynolds number having known kinematics and velocity flow-field data from IBM simulations (sparsely sampled in time), the MB-PINNs

formulation proposed in Chapter 2 have been extended here using sequential training strategies to obtain surrogate models. The aim of the surrogates is to perform velocity reconstruction and simultaneous pressure recovery. In light of the above discussed challenges, the scope, and the contributions of the present study are as follows.

- To begin with, evaluation of the earlier proposed MB-PINN for velocity reconstruction and pressure recovery under different temporal domain complexities to identify its limitations.
- Extension of MB-PINN to efficiently handle temporal domain complexities using time marching, and temporal decomposition based sequential strategies coupled with transfer learning.
- Detailed evaluation of the proposed methods under a fixed training budget over both periodic and quasi-periodic flow situations past a flapping body, depicting the three temporal domain complexities discussed above.
- Devising an effective preferential spatiotemporal sampling strategy to train sequential learning variants for accurate near-field pressure recovery.

In the present investigation, the efficacy of the sequential training strategies, and computational efficiency have been compared under a fixed training budget. A physics-based sampling,has been used to train the networks in a data-efficient manner.

The general structure of this Chapter is as follows. Note that, throughout the investigations presented here, the periodically plunging elliptic foil system is considered as described in section 2.2 following the kinematic model as described in equations (2.1) and (2.2). The details of unsteady flow past this plunging foil system modeled as two dimensional incompressible flow and the underlying IBM solver follow from section 2.3. In section 4.2, details of the sequential learning framework extending the previously proposed MB-PINNs are discussed in the context of pressure recovery from velocity data. The IBM and ALE-based CFD solver settings are used for data generation, and details of the training and testing data sets are given in section 4.3. The key results and discussion for the periodic and aperiodic test cases are presented in sections 4.4 and 4.5, respectively. Finally, the key conclusions and further work are outlined in section 4.6.

4.2 EXTENDING THE IMMERSED BOUNDARY AWARE FRAMEWORK USING SEQUENTIAL TRAINING

As mentioned in the Introduction, physical and numerical complexities related to the temporal dynamics of the system (see figure 4.1), such as, (a) temporal sparsity, (b) long-time integration, and (c) rich temporal spectrum, need to be handled while training PINNs for moving body flows in question. In the absence of any prior knowledge of the frequency content of the flow, the negative impact of these temporal complexities can be alleviated by turning to sequential learning (Krishnapriyan *et al.*, 2021; Shukla *et al.*, 2022; Penwarden *et al.*, 2023).

Surrogate modeling of unsteady flows past moving bodies using a fixed frame of reference is beneficial as it can allow handling multiple moving bodies, and possible deforming bodies. The immersed boundary aware (IBA) framework of PINNs as presented in Chapter 2 considered a fixed Eulerian frame of reference and was inspired by the benefits of the immersed boundary method. Note that, it would not be feasible to enable coordinate transformations when multiple moving bodies or flexible bodies are involved. The IBA framework in Chapter 2 (see sections 2.4, and 2.5 for more details) alleviates the need for transformations of the computational domain to body-attached frame of reference. The fluid velocity data points were obtained on a fixed Eulerian grid, whereas, the solid body was immersed in the Eulerian grid and described using a set of Lagrangian markers. Pressure recovery from velocity data in the modeling of unsteady flows past moving boundaries (see figure 4.2) was one of the major outcomes. Although the data was generated using an IBM solver, in a scenario where body position and velocity are known *a priori*, the NS-equations-based formulation, MB-PINN, was best suited for pressure recovery. In this scenario, it was also shown that, the IBM-based formulation MB-IBM-PINN could perform at par with MB-PINN only when the solid region was discarded from physics loss computation and that too with an additional computational overhead. However, when body position, shape, and velocity were not exactly known and the solid region could not be determined *a priori*, MB-IBM-PINN could be useful as

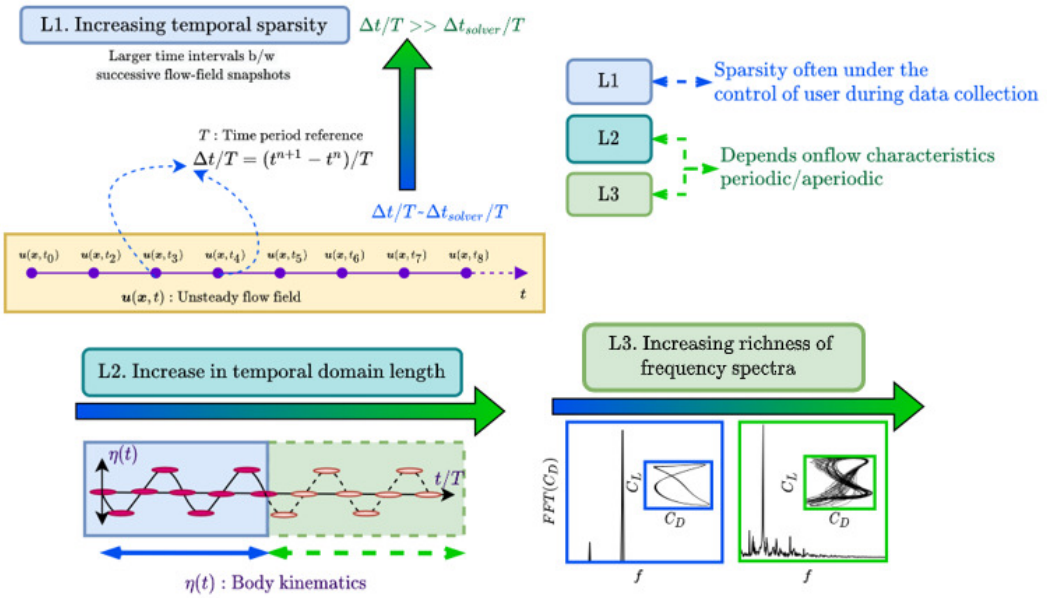


Figure 4.1: Schematic depicting different computational and physical sources of temporal domain complexities with application to unsteady flow past a sinusoidally plunging airfoil.

seen in Chapter 3. In the current case study, it is assumed that such body position, shape and velocity information are available to us, and the NS based MB-PINN formulation is considered for further evaluation under different temporal complexities.

The loss formulation for the standard MB-PINN is described in Chapter 2 section 2.5 (see equations 2.14 and (2.32) for the exact mathematical loss formulation). In this study, only the inlet is considered for Dirichlet boundary conditions and therefore, we have

$$\mathcal{L}_{Data} := \lambda_{Bulk}\mathcal{L}_{Bulk} + \lambda_{Inlet}(\mathcal{L}_{Inlet}) + \lambda_{IB}\mathcal{L}_{IB}, \tag{4.1}$$

$$\mathcal{L}_{Phy} := \lambda_{fluid}(\mathcal{L}_{m_x} + \mathcal{L}_{m_y} + \mathcal{L}_c). \tag{4.2}$$

The loss components, $\mathcal{L}_{Bulk}$ and $\mathcal{L}_{Inlet}$ correspond to predicted interior bulk velocity data and inlet velocity boundary conditions on the Eulerian grid. The additional no-slip velocity boundary condition loss, $\mathcal{L}_{IB}$, is computed directly on the solid boundary, Γ_{IB}, described by a set of Lagrangian markers. The coefficients, λ_*, for $* \in \{\text{Bulk, Inlet, IB}\}$ are the weighting coefficients of the bulk data loss, inlet, and no-slip boundary condition

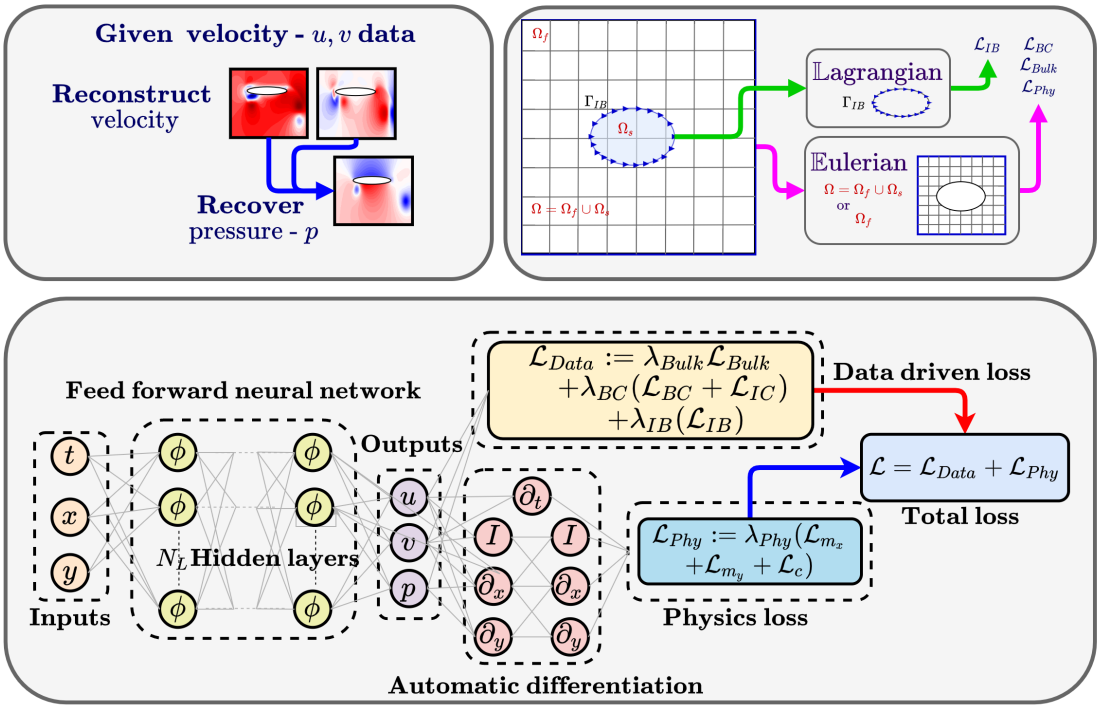


Figure 4.2: A schematic depicting the MB-PINN formulation under the immersed boundary aware (IBA) framework for pressure recovery from velocity data.

loss components, respectively. Here, λ_{fluid} is the weight of the physics loss component which operates only in the fluid region by design.

A general mathematical formulation of the above-mentioned data-driven and physics loss components is as follows

$$\mathcal{L}_* := \frac{1}{N_*} \sum_{i=1}^{N_*} \|\boldsymbol{u}(\boldsymbol{x}_*^i, t^i) - \hat{\boldsymbol{u}}(\boldsymbol{x}_*^i, t^i)\|_{L_2}^2 \tag{4.3}$$

$$\mathcal{L}_@ := \frac{1}{N_{Phy}} \sum_{i=1}^{N_{Phy}} \|r_@^x(\boldsymbol{x}_{Phy}^i, t^i)\|_{L_2}^2 \tag{4.4}$$

where, $\hat{\boldsymbol{u}}$ is the true velocity data generated for training by the CFD simulation, whereas, $\boldsymbol{u}$ is the network-predicted velocity flow-field data. The spatial and temporal points are represented by $(\boldsymbol{x}, t)$ with $\boldsymbol{x} \in \Omega^r$ and $t \in [0, T]$, respectively. The subscripts shown in the loss expressions for the spatiotemporal points are to indicate independence in sampling the select points of the particular loss component. Here, $(\boldsymbol{x}_*^i, t^i)$ for $i = \{1, \cdots N_*\}$ corresponds to the set of velocity data points in the respective fluid interior bulk region,

at the inlet or on the solid boundary, respectively. The physics-informed loss components, $\mathcal{L}_{m_x}$, $\mathcal{L}_{m_y}$, and $\mathcal{L}_c$ correspond to the mean squared errors of x and y momentum equations (equation 2.3) and continuity equation (equation 2.4) residuals $\boldsymbol{r}_{@}$, respectively. Here, the subscript @ denotes the different physics loss components - the momentum and continuity equation residuals as described in equation (2.25) in Chapter 2.

As already stated, the focus of the present study is on different temporal domain complexities and recovery of pressure from velocity data under them. In many experimental scenarios the body can be tracked with a camera and thus its position and velocity can be known with reasonable accuracy (Calicchia *et al.*, 2023). On the other hand, velocity data could be noisy or sparse, and the instantaneous pressure fields not being available at all. It is hence assumed in this study that the available velocity flow-field data is temporally sparse, while the body position and velocity are available in a time-resolved manner.

Extending MB-PINN proposed in Chapter 2, a general loss formulation is proposed, which under suitable conditions morphs to sequential learning variants of the MB-PINN model. In the present investigation, the efficacy of the sequential training strategies, and computational efficiency have been compared under a fixed training budget. Additionally, an improvement of the previously proposed physics-based sampling (see section 2.8) has been used to train the networks in a data-efficient manner. The sequential learning based extensions of the MB-PINN will be considered for the pressure recovery problem. The details of the framework and a general loss formulation are discussed in the subsequent section.

4.2.1 Network architecture and loss formulation

For spatiotemporal coordinates, $\{(\boldsymbol{x}, t)\}_{i=1}^{N} \in \Omega_{\boldsymbol{x}} \times \Omega_t$, N_d domain decompositions can be obtained such that, $\Omega_{\boldsymbol{x}} \times \Omega_t = \bigcup_{i=1}^{N_d} \Omega_{\boldsymbol{x}}^i \times \Omega_t^i$, with common stationary or moving interfaces $\{\partial\Omega_{ij}\}_{i=1,j=1}^{N_d,N_d} \forall i \neq j$. The temporal domain is broken down into subdomains and these

interfaces/common data points are only considered in the temporal domain context. Now, an array of N_{nets} number of neural network backbones, $\{_{\#_j}\mathcal{N}(\theta_j)\}_{j=1}^{N_{nets}}$, can be defined on each subdomain or a combination of subdomains, such that $\#_j$ is a feed-forward neural network. One can choose variants of a feed-forward neural network as well, such as the modified MLP (mMLP) (Wang *et al.*, 2021*a*), or, a Fourier feature-based multiscale network (Wang *et al.*, 2021*b*). A simple feed-forward network is considered in the present study. Within each subdomain, multiple sets of corresponding spatio-temporal coordinates and target variables are used for training which have been described in the following.

For any i^{th} subnetwork being trained, the overall loss formulation is mathematically expressed as:

$$\mathcal{L}(\theta_i) := \mathcal{L}_{Data}(\theta_i) + \mathcal{L}_{Phy}(\theta_i) + \mathcal{L}_{Backward}(\theta_i), \tag{4.5}$$

$$\mathcal{L}_{Data}(\theta_i) := \lambda^i_{Bulk}\mathcal{L}_{Bulk}(\theta_i) + \lambda^i_{inlet}\mathcal{L}_{inlet}(\theta_i) + \lambda^i_{IB}\mathcal{L}_{IB}(\theta_i) \tag{4.6}$$

$$\mathcal{L}_{Phy}(\theta_i) := \lambda^i_{fluid}(\mathcal{L}_{m_x}(\theta_i) + \mathcal{L}_{m_y}(\theta_i) + \mathcal{L}_c(\theta_i)), \tag{4.7}$$

$$\mathcal{L}_{Backward}(\theta_i) = \lambda^i_{Back}\sum_{j=1}^{N_d-1}\mathcal{L}^j_{Bulk}. \tag{4.8}$$

Here, N_d are the number of subdomains and the loss coefficients, λ^i_*, indicate that one can choose different weights for each i^{th} subdomain independently. These weights can be determined either globally or locally (i.e. in a point-wise manner) as in the works of McClenny and Braga-Neto (2023); Xiang *et al.* (2022); Wang *et al.* (2021*a*); Anagnostopoulos *et al.* (2024). In the present study, aligning with the earlier investigations in section 2.7, and following the work of Lucor *et al.* (2022), a global physics loss relaxation is adopted.

Different MB-PINN variants are obtained from the generalized loss formulation shown above (see figure 4.3 for a detailed schematic and table 4.1). In addition to the baseline standard MB-PINN shown in the figure 4.3, the sequential learning variants of MB-PINN

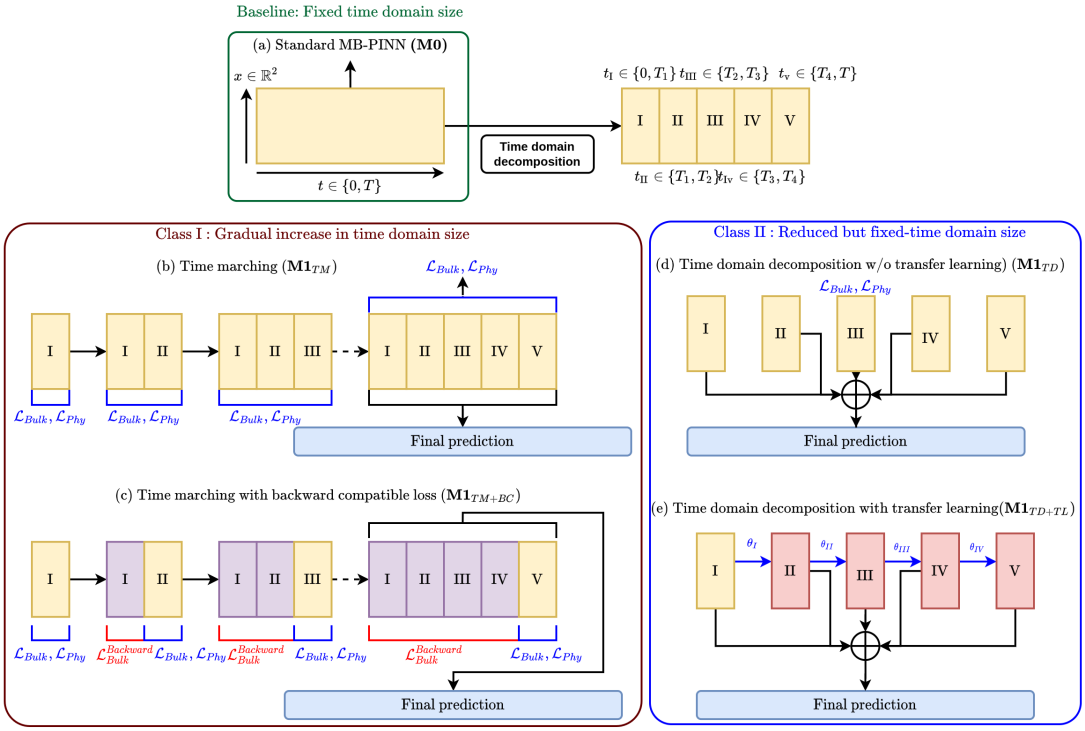


Figure 4.3: Schematics of different types of sequential learning variant of MB-PINN considered. The entire time domain is partitioned into subdomains (five, as an example) of equal length. The green, red, and blue outlined boxes refer to broad classification in terms of the time domain size the model sees during training; ClassI: $\mathbf{M1}_{TM}$ and $\mathbf{M1}_{TM+BC}$ fall under this for which a single network sees a gradual increase in the time domain by appending the subdomains one by one during training; Class II: $\mathbf{M2}_{TD}$ and $\mathbf{M2}_{TD+TL}$ fall under this class for which multiple networks form the model, each network inheriting the data of a subdomain.

have been further classified into two broad categories: Class I, and Class II, depending on whether the time domain is gradually increased in size, or decomposed into small subdomains during training (figure 4.3).

***Baseline* (M0)** In this case, MB-PINN is trained over the entire time domain effectively as a single domain and a single network under consideration. As a result, $N_{nets} = 1$ and $N_d = 1$ (see figure 4.3(a)).

Class I Models

In this variant, starting with the first subdomain, a *single* network ($N_{nets} = 1$) is trained in stages, and with each passing training stage the time domain size increases gradually by the addition of the subsequent temporal subdomain. Once trained, only this *single* network is necessary to infer the predictions.

Time marching (*$\mathbf{M1}_{TM}$*) In this approach, the time domain is decomposed into N_d sequentially ordered input time slabs/subdomains for simplicity. Note that while the model size is fixed, the time domain size is gradually increased (see figure 4.3(b)) by adding the time slabs one by one during training.

Time marching + Backward compatible training(*$\mathbf{M1}_{TM+BC}$*) This approach incorporates elements from the backward compatible PINNs proposed by Mattey and Ghosh (2022). $\mathbf{M1}_{TM+BC}$ is trained such that $\mathcal{L}_{Data}(\theta_i)$ and $\mathcal{L}_{Phy}(\theta_i)$ are computed over $i = 1, 2 \cdots N_d^{data}$ temporal subdomains. In addition, backward compatible data losses ($\mathcal{L}_{Backward}$) are introduced to fit the predictions obtained over the previous $i-1$ time slabs using the network used in the current i^{th} time slab. (see figure 4.3(c)).

Class II Models

In this category, the size of the time domain is reduced by splitting the time domain into smaller subdomains and training individual networks over each. Post-training, these networks need to be strung together to obtain the predictions over the entire time domain. While this increases the number of networks to be trained, due to reduced time domain size for each network potentially a smaller network can be trained with a lower computational time.

Time domain decomposition ((*$\mathbf{M2}_{TD}$*) + Transfer learning (*$\mathbf{M2}_{TD+TL}$*)) The time domain is decomposed into N_d subdomains in this approach as well. But, unlike $\mathbf{M1}_{TM}$

or $\mathbf{M1}_{TM+BC}$, there are $N_{nets} = N_d$ number of networks trained subsequently over each subdomain (see figures 4.3(d) and (e)). Note that $\mathbf{M2}_{TD}$ does not incorporate transfer learning of weights, while $\mathbf{M2}_{TD+TL}$ does so. In $\mathbf{M2}_{TD+TL}$, the final weights of the trained network from the previous $(i-1)^{th}$ subdomain are used (transferred) as initialization for training over the subsequent i^{th} subdomain consecutively, hence termed as a transfer learning approach.

Table 4.1: Overview of the standard and different sequential learning variants of MB-PINN

Model	Time Marching	Backward compatible	Time domain decomposition	Transfer learning
M0	✗	✗	✗	✗
$\mathbf{M1}_{TM}$	✓	✗	✗	✗
$\mathbf{M1}_{TM+BC}$	✓	✓	✗	✗
$\mathbf{M2}_{TD}$	✗	✗	✓	✗
$\mathbf{M2}_{TD+TL}$	✗	✗	✓	✓

The loss function is optimized over the network parameters, $\theta = \{\theta_i\}_{i=1}^{N_{nets}}$, using an ADAM optimiser (Kingma and Ba, 2017). For a fair comparison, baseline, Class I and II models (sub-networks) are trained under a fixed training budget of 1.5e06 iterations with a step decay of learning rate after every 5e05 iterations, starting from $\eta = 1e-03$, $5e-04$ over the second step, and finally to $1e-04$. Based on the extensive studies performed earlier in Chapters 2 and 3, $\lambda_{fluid} = 0.01$ was chosen throughout the rest of the study.

4.3 DATABASE GENERATION

4.3.1 Test cases

Two unsteady flow scenarios were considered in the present study to evaluate the performance of the sequential learning variants under different temporal domain complexities mentioned earlier. The details of the flapping foil system and unsteady flow modeling follow from sections 2.2 and 2.3 of Chapter 2. In the first scenario, the flow was simulated at $Re = 500$, reduced frequency, $k = 2\pi$ and plunge amplitude, $h = 0.16$,

corresponding to $kh = 1.0$ (Khalid *et al.*, 2015). The flow-field in this case was completely periodic. The second scenario involved a more challenging case of a quasi-periodic flow as in Majumdar *et al.* (2020). It was simulated with a slightly lower Reynolds number $Re = 300$ at $k = 4$, $h = 0.4125$ resulting in $kh = 1.65$. At higher Reynolds numbers, the transitional route from the periodic dynamics is extremely quick with slight changes in the kinematic parameters and more difficult to capture (Lewin and Haj-Hariri, 2003; Khalid *et al.*, 2015, 2018). We would like to investigate a quasi-periodic flow scenario where the flow pattern is not exactly repeating (but seemingly regular), incommensurate frequencies are introduced in the temporal spectra due to underlying nonlinearity of the flow, LEV-TEV interactions. Hence, $Re = 300$ is chosen for the quasi-periodic case. This case is more challenging because of two main reasons: lack of repeatability of the flow-field, and its spectral richness. Note that, the combination of a relatively large plunging amplitude and lower Reynolds number make the vortices larger leading to a relatively larger domain extent than in the periodic case if

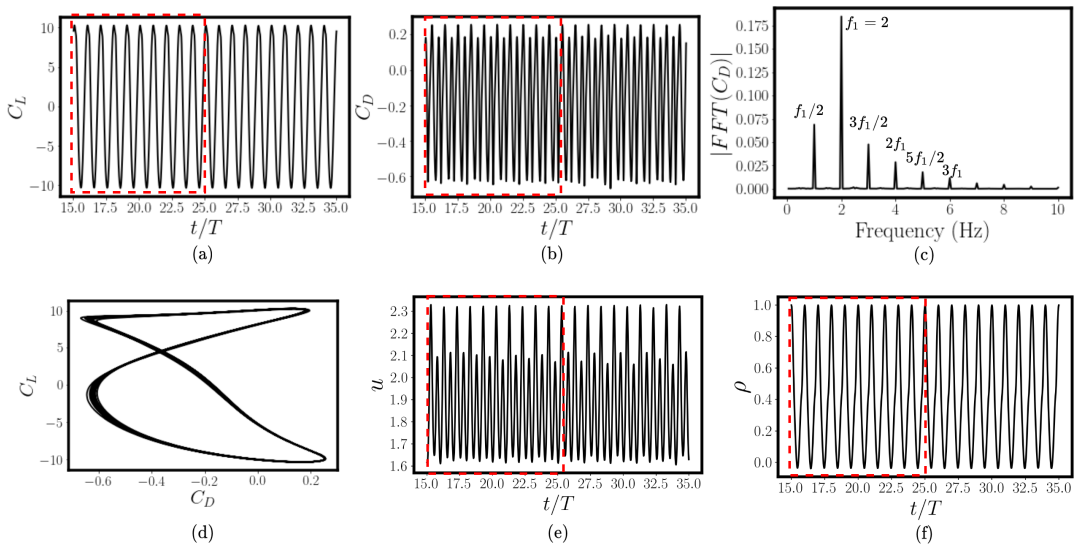


Figure 4.4: Dynamics and data analysis of the periodic case: (a-b) coefficients of lift (C_L) and drag (C_D) with time, (c) C_D Fourier spectrum, (d) $C_L - C_D$ phase portrait, (e) streamwise velocity (u) at probe location $x/c = 2.5$, and, (f) corresponding streamwise velocity correlation (ρ) as a function of time (t/T).

one were to capture at least two vortex couples. All the unsteady flow simulations were carried out using an in-house IBM solver (Majumdar *et al.*, 2020). IBM based solvers in general are ideal for handling large movements of solid boundaries as they do not require mesh movement due to which the accuracy could suffer as in some other mesh conformal approaches, like the ALE (Sarrate *et al.*, 2001). Simulation data were coarsened in space-time for training (more on this will be discussed in section 4.3.3), and the original high-resolution data were used for testing. To establish the dynamical states of the associated flow fields, a detailed characterization in terms of the aerodynamic loads and the stream-wise correlation of the $x-$ component of velocity has been shown in figures 4.4 and 4.5. The time regular periodic case was chosen as a good test case for a thorough evaluation of the proposed variants under temporal sparsity, as well as the challenge of training over large time domains. Based on these evaluations, the most suited variants was identified and was then evaluated against the more challenging quasi-periodic case. Note that, for aperiodic flows minor differences in field conditions could potentially grow, resulting in ALE and IBM obtained solutions slightly differing from each other mainly owing to differences in the underlying numerical methods (Majumdar *et al.*, 2020). This could potentially bring small discrepancies in the aerodynamic loads in the aperiodic case. Thus, a straightforward cross-comparison of model predictions might be difficult for the aperiodic case.

For the quasi-periodic situation, the frequency spectrum of the load (see in figure 4.5(c)) exhibits incommensurate frequencies. The presence of the incommensurate second frequency is not related to the input kinematics and is a direct result of the non-linearity in the flow governing equations. Moreover, the streamwise correlation of axial velocity component u at a probe location $x/c = 2.5c$ (see figure 4.5(c)) shows a decreasing trend with an oscillatory behavior indicating quasi-periodicity (Majumdar *et al.*, 2020). Both case studies are expected to serve as benchmarks to evaluate the performance of standard (baseline) and sequential MB-PINN variants.

4.3.2 Revisiting vorticity cutoff based sampling

In order to improve the data efficiency of the MB-PINN model, the vorticity cut-off-based spatial undersampling strategy (VS) proposed in section 2.8 has been used in the present study. In this approach, given the bulk velocity data $\{\boldsymbol{u}(\boldsymbol{x}^i, t^i)\}_{i=1}^{N_{Bulk}}$, vorticity values are calculated as, $\omega = \nabla \times \boldsymbol{u}$. Since vorticity can be considered a proxy for regions with strong flow-field gradients, a suitable cutoff $|\omega| = |\omega^*| > 0$, needs to be chosen to differentiate between the strong and weak gradient regions. A desired percentage ($S_{|\omega|<|\omega^*|}$ and $S_{|\omega|\geq|\omega^*|}$) of bulk data points can then be sampled from these regions respectively.

4.3.3 Details of training and testing datasets

Coarsened simulation data, and subsequently vorticity cutoff based spatial sampling, with a sampling ratio of $S_{\omega_z} = 5\%$, were used for training, as was also considered in our earlier study in section 2.8. For the periodic case, $|\omega^*| = 1$ was chosen. For the quasi-periodic case, where the Reynolds number was lower, a lower cutoff of $|\omega^*| = 0.1$ was chosen due to

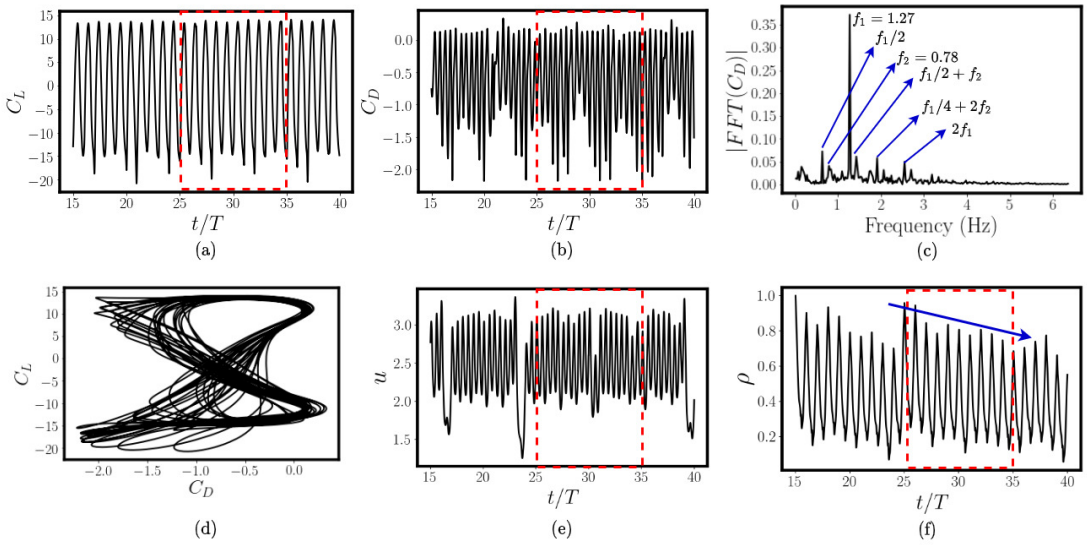


Figure 4.5: Dynamics and data analysis of the quasi-periodic case: (a-b) lift and drag coefficient time histories, (c) C_D Fourier spectrum, (d) $C_L - C_D$ phase portrait, (e) streamwise velocity (u) at probe location, $x/c = 2.5$, and, (f) corresponding streamwise velocity correlation (ρ) as a function of time (t/T).

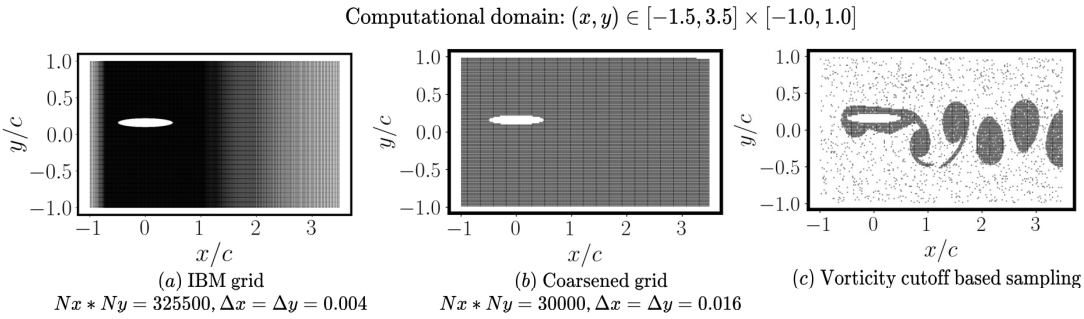
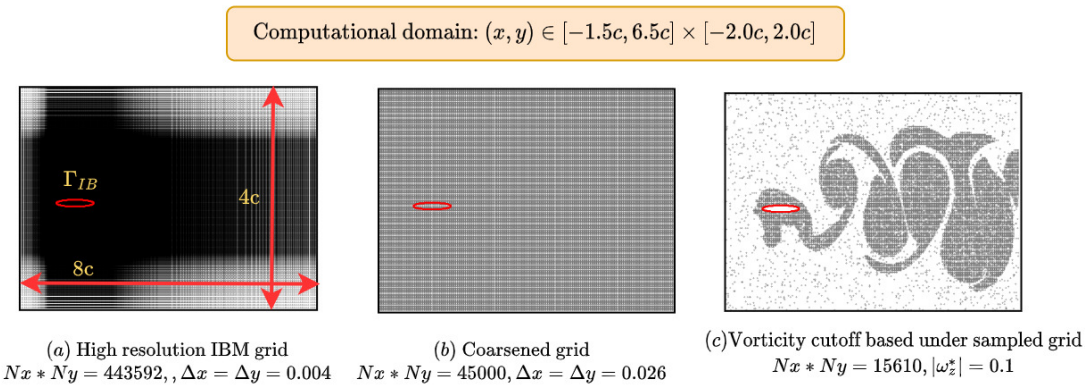


Figure 4.6: Computational domain and grids used for training and testing for the periodic case; (a) truncated high-resolution grid for IBM on which the models are tested, (b) 4x coarsened grid with 10x lower number of data points than in the high-resolution case, (c) vorticity cut off based undersampled grid.

Table 4.2: Details of the training and testing database for the domain considered for the periodic case. CI-*: coarse interpolated dataset; Ref-IBM and Ref-ALE: high-resolution testing data sets generated using IBM and ALE solver, respectively. Further details on 'Ref-*' datasets can be found in section 2.6.1. Here, the number of bulk data grid points (N_{Bulk} and residual collocation points (N_{Phy}) are specified per snapshot as the respective temporal resolutions of sampling $\Delta t_{Bulk}/T$ and $\Delta t_{Phy}/T$ are later varied over different examples in the study.

Datasets	N_x	N_y	N_{Bulk}	N_{Phy}
Training datasets				
CI	270	120	3.186e04	**3.186e04**
CI-S5	-	-	**6.458e03**	
Testing datasets				
Ref-IBM	651	500	3.255e05	-
Ref-ALE	-	-	5.0e04	-

the possibility of relatively larger sizes of the flow-field vortices. This also led to a larger number of grid points sampled in the wake region, compared to the near-field region surrounding the moving body. A schematic of the truncated domain and a representative snapshot of the undersampled grids have been presented for the periodic and quasi-periodic cases in figures 4.6 and 4.7, respectively. Note that the problem deals with a moving discontinuity in the domain and in that light, evaluating the accuracy of near-field velocity reconstruction and pressure recovery under different temporal complexities to identify the suitable sequential learning variants is crucial. Towards that aim, the



(a) High resolution IBM grid
$Nx * Ny = 443592, , \Delta x = \Delta y = 0.004$

(b) Coarsened grid
$Nx * Ny = 45000, \Delta x = \Delta y = 0.026$

(c)Vorticity cutoff based under sampled grid
$Nx * Ny = 15610, |\omega_z^*| = 0.1$

Figure 4.7: Computational domain and grids used for training and testing for the quasi-periodic case; (a) truncated high-resolution grid for IBM on which the models are tested, (b) 6x coarsened grid with 10x lower number of data points than in the high-resolution case, (c) vorticity cut off based undersampled grid.

coarsened undersampled training and high-resolution ground truth testing databases were generated to investigate the efficacy of the sequential learning strategies: for the periodic case, the details of the dataset are shown in table 4.2, and for the quasi-periodic case in table 4.3. These databases were generated for different temporal resolutions of bulk data ($\Delta t_{Bulk}/T$) and residual collocation grid points ($\Delta t_{Phy}/T$), respectively.

Table 4.3: Details of the training and testing databases for the domain considered in this study for the quasi-periodic flow. CI-*: coarse interpolated datasets; Ref-IBM-QS and Ref-ALE-QS: high-resolution testing data sets generated using an IBM and an ALE solver, respectively. The domain size considered here is $[-1.5c, 6.5c] \times [-2c, 2c]$. Here, N_t^{Bulk} corresponds to the number of bulk data snapshots over the time domain of ten plunging time periods. Unlike table 4.2, here, N_{Bulk} and N_{Phy} correspond to the overall number of grid points over the entire time domain.

Data sets	N_x	N_y	N_t^{Bulk}	$\Delta t_{Bulk}/T, \Delta t_{Phy}/T$	N_{Bulk}	N_{Phy}
Training datasets						
CI-QP81	300	150	81	0.125, 0.015625	**3.594e06**	
CI-S5-QP81	-	-	81	0.125, 0.015625	**1.219e06**	**2.875e07**
CI-S5-QP21	-	-	21	0.5, 0.015625	**3.219e05**	
Testing datasets						
Ref-IBM-QP	732	606	641	0.015625, -	2.805e08	-

4.4 TACKLING TEMPORAL COMPLEXITIES IN THE PERIODIC CASE

The sequential learning strategies described in the previous section are first evaluated for the periodic flow case. As the flow-field is regular and repeats exactly in this case, it would be possible to identify the underlying data-independent deficiencies of the models. Specifically, two isolated training scenarios for the periodic case study have been considered: one with temporal sparsity of the velocity data over a short time domain, and the other with a long time domain but with temporally well resolved data. We wish to determine the efficacy of the proposed sequential learning variants of MB-PINN under each case. Although MB-PINN is used here as the body position and its velocity information are known beforehand, these sequential learning methods can also be directly adopted for MB-IBM-PINN as was proposed in section 2.5, in case temporal domain complexities arise for hidden boundary estimation problems such as the one discussed in Chapter 3 which is left for future work.

4.4.1 Importance of physics loss, and residual collocation points

For an accurate reconstruction of temporal signals, Nyquist criterion suggests that the sampling rate should be at least twice the maximum frequency (f_{max}) observed in the system, that is, $\Delta t_{Bulk}/T \leq f_h/2f_{max}$, with $f_h = 1/T$ being the plunging frequency. Purely data-driven methods could fail when the temporal resolution does not adhere to the Nyquist criterion. In the context of flow-field reconstruction, especially with a moving body in the domain, temporal interpolation/reconstruction of the velocity field under temporal sparsity could be challenging. The interpolation errors would often be the highest in the vicinity of the moving body as seen earlier in Chapter 2 (see figure 2.20, and tables 2.8 and 2.9 from section 2.8). Hence, it is important to determine if there are any advantages in using the physics informed baseline MB-PINN (**M0**) model for temporal interpolation, over its purely data-driven standard feed-forward neural network (MB-FNN) counterpart also studied in Chapter 2. To begin with, MB-FNN would require a large amount of data to perform satisfactorily. Given that PINNs are expected to work well under data-sparsity conditions, the need for a physics

informed loss is demonstrated by comparing the spatio-temporal velocity reconstruction through MB-FNN and **M0** model of MB-PINN under different temporal sparsity levels ($\Delta t_{Bulk}/T \in [0.2, 0.25, 0.3, 0.4, 0.5]$). In most cases, one can acquire the position and the velocity of the body with high resolution through certain means, but it might be difficult to obtain the flow-field data. Hence, frequency content of the flow from temporally sparse data cannot be estimated accurately *a priori* and hence the Nyquist criterion cannot be defined exactly for the PINN models. However, it is known that the vortex shedding frequency would most often be equal to the plunging frequency (Majumdar *et al.*, 2020), and the drag force would be characterized by twice that frequency. Hence, a proxy Nyquist criterion can still be defined based on twice the plunging frequency as, $\Delta t_{Bulk}/T \leq f_h/2f_{max} = f_h/4f_h = 0.25$ such that $f_{max} = 2f_h$. The reason for defining a proxy Nyquist criterion stems from the need to define a relative metric for oversampling or undersampling of the near-field or the far-field data. Since only the kinematics is

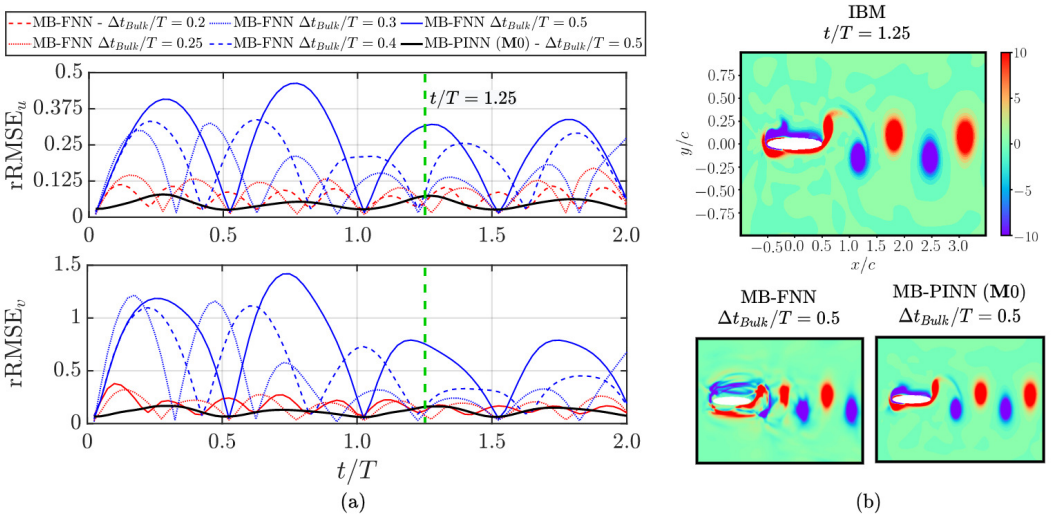


Figure 4.8: Comparison of standard MB-PINN (baseline, $\mathbf{M}_0$) with purely data-driven MB-FNN for velocity reconstruction at different $\Delta t_{Bulk}/T \in [0.2, 0.25, 0.3, 0.4, 0.5]$; (a) relative errors over time for velocity reconstruction, and, (b) comparison of a representative vorticity snapshot obtained from (top) IBM ground truth, and (bottom) model predictions at $t/T = 1.25$, where, a significant improvement in the near-field is observed with MB-PINN (**M0**) as opposed to MB-FNN for $\Delta t_{Bulk}/T = 0.5$.

Table 4.4: Comparison of the accuracy of velocity component (v) reconstructed for the periodic case by linear interpolation (LI), purely data-driven feedforward neural network (MB-FNN), and standard MB-PINN (**M0**) when trained with varying levels of temporal sparsity without satisfying the Nyquist criterion ($\Delta t/T \leq 1/(2f_{max}) = 0.25$). The errors are evaluated over two plunging cycles.

Model	$\Delta t_{Bulk}/T$	**Network size**	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
LI	0.5	-	5.17e-01	3.03e-01	-1.01e-01	104.96
MB-FNN	0.5	100×10	3.56e-02	1.87e-02	3.77e-01	69.87
M0	0.5	100×10	5.62e-02	2.38e-02	9.87e-01	**11.07**

known, and given the temporally sparse nature of the data, the frequency spectra of the flow cannot be identified *a priori* during training. Hence, the only way was to make use of the plunging frequency to define a Nyquist criterion acknowledging that the drag frequency goes twice that of lift/vortex shedding. Note that the predictions are made at a temporal resolution of $\Delta t/T = 0.025$ which is atleast 8 times finer depending on the $\Delta t_{Bulk}/T$ mentioned above.

The relative reconstruction errors in time (figure 4.8(a)) for purely data driven MB-FNN across the temporal sparsity levels were notably worse than **M0**, trained with $\Delta t_{Bulk} = 0.5$ not satisfying the Nyquist criterion. Moreover, the near-field was distorted for the MB-FNN interpolated vorticity contour at a test time stamp, whereas, **M0** outperformed in this regard (see figure 4.8(b)). This clearly shows that the physics informed approach is capable of significantly improving the spatio-temporal interpolation in the limited data scenario. A detailed comparison of the average error metrics of velocity reconstruction has been presented in table 4.4 for $\Delta t_{Bulk}/T = 0.5$.

With the help of its physics-informed approach in **M0**, it is also possible to recover pressure as a hidden variable (unlike MB-FNN). But the temporal resolution of the residual collocation points ($\Delta t_{Phy}/T$) could also play an important role in improving the accuracy of pressure recovery as seen in figure 4.9. Notably, in figure 4.9 at very coarse $\Delta t_{Phy}/T \in [0.5, 0.2, 0.1]$, the errors were above 50% on average, indicating a

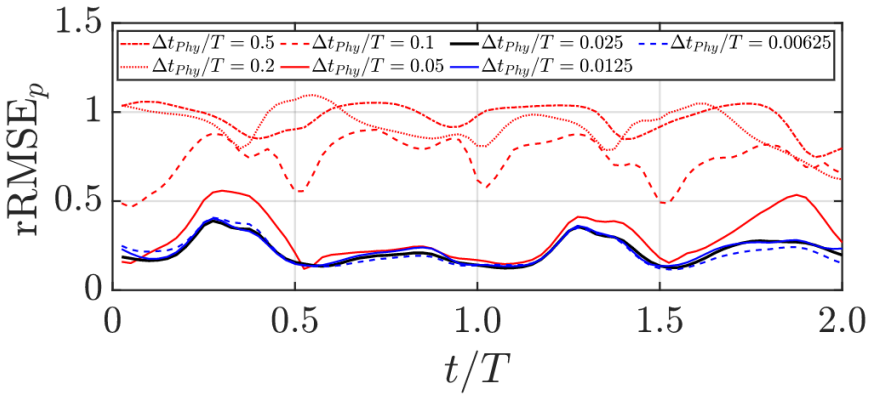


Figure 4.9: Comparison of the time behavior of relative RMSE for pressure recovery with MB-PINN (**M0**) models trained over two plunging periods for the periodic case. Effect of different temporal resolutions of residual collocation points, $\Delta t_{Phy}/T \in [0.5, 0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625]$ is shown. The data snapshots are available at a temporal resolution of $\Delta t_{Bulk}/T = 0.5$. As temporal resolution increases ($\Delta t_{Phy}/T$ decreases), the pressure recovery improves overall.

poor pressure recovery. Whereas, there was a significant improvement in the relative error at $\Delta t_{Phy}/T = 0.05$, which then plateaued at $\Delta t_{Phy}/T = 0.025$ with insignificant improvements at finer temporal resolutions. For practical reasons, having a coarser temporal resolution for residual collocation points and bulk data points, is computationally efficient. This is because, with coarser resolution the overall size of the entire dataset is reduced. Thus, during training, given a fixed mini-batch size, the model can see more number of epochs of the data at coarser resolutions. Therefore, a favorable trade-off needs to be achieved between accuracy and computational efficiency while selecting the temporal resolution. In this case, $\Delta t_{Phy} \in [0.0125, 0.025]$ was reasonable as indicated in figure 4.9. For more information on the effect of temporal resolution of residual collocation points, see tables 4.5 and 4.6 for the accuracy details.

Temporal resolution of residual collocation points

The accuracy details for velocity reconstruction and pressure recovery for baseline $M0$ models trained with different temporal resolution of residual collocation points $\Delta t_{Phy}/T$ are presented in tables 4.5 and 4.6. Here, the bulk data is sampled at $\Delta t_{Bulk}/T = 0.5$. It

is observed that the velocity reconstruction errors improve significantly up to $\Delta t_{Phy}/T$ = 0.0125 beyond which the improvement is not significant. The significant improvement in velocity reconstruction is met with only a slight degradation of pressure recovery. But further increase in the temporal resolution of residual collocation points would make it computationally expensive when training over large time domains.

Table 4.5: Comparison of accuracy details of y velocity component (v) reconstructed by the standard MB-PINN when trained with a varying number of PDE collocation points in time ($\Delta t_{Phy}/T$). Here, the temporal sparsity of data snapshots is such that $\Delta t_{Bulk}/T = 0.5$ not satisfying the Nyquist criteria ($\Delta t/T \leq 1/(2f_{max}) = 0.25$). The errors are evaluated over two plunging cycles. The network is of depth $l = 10$, and width $n = 100$.

$\Delta t_{Phy}/T$	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
Linear Interpolation				
-	5.17e-01	3.03e-01	-1.01e-01	104.96
MB-FNN				
-	3.56e-01	1.87e-01	3.77e-01	69.87
M0				
0.5	2.18e-01	1.22e-01	7.68e-01	42.67
0.2	1.50e-01	6.51e-02	8.94e-01	29.82
0.1	1.37e-01	5.59e-02	9.11e-01	27.27
0.05	6.46e-02	2.69e-02	9.81e-01	12.72
0.025	5.62e-02	2.38e-02	**9.87e-01**	**11.07**
0.0125	4.32e-02	1.95e-02	**9.91e-01**	**8.45**
0.00625	4.21e-02	1.94e-02	**9.92e-01**	**8.22**

Table 4.6: Comparison of accuracy details of pressure (p) recovered by the standard MB-PINN when trained with a varying number of PDE collocation points in time ($\Delta t_{Phy}/T$). Here, the temporal sparsity of data snapshots is such that $\Delta t_{Bulk}/T = 0.5$ not satisfying the Nyquist criteria ($\Delta t/T \leq 1/(2f_{max}) = 0.25$). The errors are evaluated over two plunging cycles. The network is of depth $l = 10$, and width $n = 100$.

$\Delta t_{Phy}/T$	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
0.5	1.28	8.25e-01	5.77e-02	96.72
0.2	1.22	7.32e-01	1.53e-01	91.35
0.1	9.72e-01	5.45e-01	4.43e-01	73.77
0.05	3.79e-01	2.20e-01	8.95e-01	29.75
0.025	2.82e-01	1.77e-01	**9.47e-01**	**21.74**
0.0125	2.93e-01	1.82e-01	**9.44e-01**	**22.55**
0.00625	2.76e-01	1.70e-01	**9.49e-01**	**21.17**

4.4.2 Efficacy of models under temporal sparsity

It is evident that with physics-informed loss, and appropriate temporal resolution of residual collocation points, both spatio-temporal interpolation of velocity data, and pressure recovery are much better than the pure data-driven counterpart. Recent works of Penwarden *et al.* (2023); Mattey and Ghosh (2022); Wang *et al.* (2024) suggest sequential learning methods result in improved performance due to gradual increase, or reduced temporal domain size (see section 4.2 for an explanation on the classification of the methods and the different ways in which they reduce the problem complexity) as compared to the simple standard learning. These investigations mostly relied on the ability to solve forward problems for some canonical PDEs. For most practical applications, traditional physics based solvers are often triumphant in forward problems, owing to their computational efficiency. It is only for inverse problems PINNs are powerful, owing to their flexiblity and ability to work with limited data. In addition, in most cases, some form of observational/simulation data are available albeit not of desired level of spatio-temporal resolution and quality. This could be due to computational/memory constraints, which inhibit experimentalists and CFD engineers alike to deal with truncated spatial domains and coarse temporal resolutions. When data becomes limited in space-time, PINNs could prove to be useful for pressure recovery as a hidden variable. However, the standard learning based **M0** can be hard to train on temporally sparse data, collected over long time domains especially for moving body flow-fields. In such a scenario, it is important to study the efficacy of sequential learning over standard learning methods. For this purpose, the baseline, Class I (**M1**), and Class II (**M2**) models were first evaluated under temporal sparsity for velocity reconstruction and pressure recovery over a short time domain $t/T \in [0, 2]$ (two plunging time periods); this was with $\Delta t_{Bulk}/T = 0.5$ that did not meet the Nyquist criterion ($\Delta t_{Bulk}/T < 0.25$).

Here in table 4.7, the accuracy details for pressure recovery using **M0** models of different sizes ((100×10), (100×5), and (50×5) were chosen as potential candidates) across different temporal sparsity levels is presented. Clearly, as long as the temporal resolution

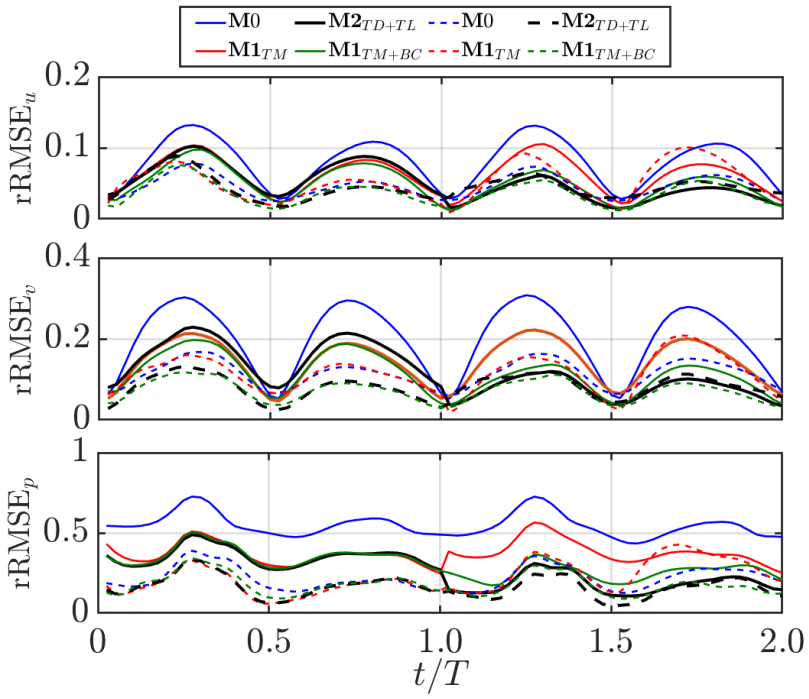


Figure 4.10: Relative RMSE over different times for (top-middle) reconstructed velocity, and (bottom) recovered pressure fields compared for different standard and sequential learning model variants for the periodic case. Solid lines represent shallow models with $l = 5$ hidden layers, dashed lines represent deeper models with $l = 10$ layers.

satisfies Nyquist criteria, the pressure recovery errors are lover than 15% for the 100 x 10 network. Importantly, it was also observed that in the case of a forward problem ($\Delta t_{Bulk}/T = \infty$, with velocity data available only at the initial time stamp $t/T = 0$), the pressure recovery drastically suffers even over a time domain of two plunging time periods. Forward problems involving moving boundaries need special attention in terms of initialization, training, architecture, loss formulation, and architecture. This requires an in depth independent investigation which is out of the scope of the present work. Clearly, among the network candidates, the network of size 50×5 could not yield a good pressure recovery which could be due to lack of model expressivity.

The relative errors in time (see figure 4.10) for the reconstructed velocity and recovered pressure fields indicate the highest average errors for time stamps that are equidistant

Table 4.7: Comparison of pressure recovery accuracy details of the standard MB-PINN (**M0**) model when trained with varying levels of temporal sparsity. Here, in the table (F) stands for forward problem when $\Delta t_{Bulk}/T = \infty$

$\Delta t_{Bulk}/T$	**Network size**	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
0.2	100×10	1.372e-01	8.16e-02	9.861e-01	10.84
0.25	100×10	1.78e-01	1.07e-01	9.79e-01	13.86
0.3	100×10	2.22e-01	1.25e-01	9.68e-01	16.99
0.4	100×10	2.31e-01	1.48e-01	9.62e-01	18.05
0.5	100×10	2.82e-01	1.77e-01	9.47e-01	21.74
∞(F)	100×10	9.46e-01	6.84e-01	4.32e-01	73.70
0.2	100×5	2.88e-01	1.85e-01	9.48e-01	22.12
0.25	100×5	3.02e-01	1.90e-01	9.43e-01	23.4
0.3	100×5	4.19e-01	2.73e-01	8.69e-01	32.66
0.4	100×5	4.98e-01	3.27e-01	8.36e-01	38.32
0.5	100×5	7.22e-01	4.87e-01	6.95e-01	54.71
∞(F)	100×5	1.39	1.06	-1.85e-01	107.49
0.2	50×5	5.25e-01	3.66e-01	8.38e-01	39.79
0.25	50×5	5.66e-01	3.93e-01	8.09e-01	43.15
0.3	50×5	6.99e-01	4.87e-01	6.93e-01	53.45
0.4	50×5	8.56e-01	5.97e-01	5.59e-01	64.84
0.5	50×5	1.05	7.41e-01	3.72e-01	79.07
∞(F)	50×5	1.66	1.31	-6.66e-01	127.04

from the previous and next available data snapshots. It is evident that by design, in class I methods, the errors accumulate over the later time stamps ($\mathbf{M1}_{TM}$), or, over the earlier time stamps ($\mathbf{M1}_{TM+BC}$), respectively which is undesirable. In such cases class II models, by design, are triumphant over the standard MB-PINN. Since the kinematics is periodic and the time domain decomposition is such that initial position for both time segments align exactly, transfer learning was seen to be beneficial over the variants that see the problem complexity growing gradually.

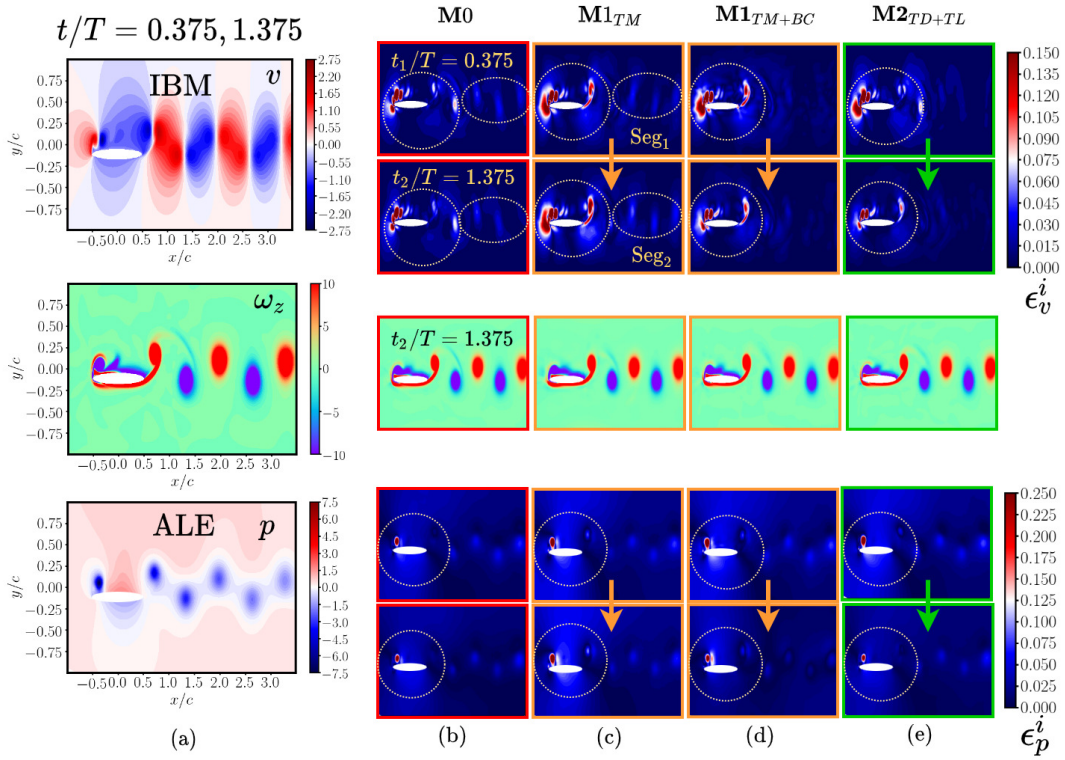


Figure 4.11: Comparison of point-wise maximum value normalized absolute errors in (top) velocity, and (bottom) pressure predictions and (middle) corresponding vorticity contours obtained from standard and sequential learning MB-PINN variants at representative test time stamps from first and last time segments at $\Delta t_{Bulk}/T = 0.5$ for the periodic case,

The velocity, pressure and vorticity contours in figure 4.11 reveal that highest errors are observed consistently in the near-field region surrounding the moving body, across different model variants. Whereas, the far-field predictions are almost qualitatively

indiscernible from the ground truth. Based on the averaged error metrics in tables 4.8 and 4.9, out of Class I and Class II models, $\mathbf{M1}_{TM+BC}$, and $\mathbf{M2}_{TD+TL}$ were promising for handling temporally sparse short time domain datasets. With the benefit of transfer learning, class II variant ($\mathbf{M2}_{TD+TL}$) performed significantly better than $\mathbf{M1}_{TM+BC}$, especially when the network size was smaller (n = 100 and l = 5), which is a desirable feature in terms of computational efficiency.

Table 4.8: Comparison of the accuracy of velocity component v of the periodic case reconstructed by linear interpolation (LI), and the standard, sequential, and backward compatible MB-PINN models when trained with $\Delta t_{Bulk}/T = 0.5$ without satisfying the Nyquist criteria ($\Delta t_{Bulk}/T \leq 1/(2f_{max}) = 0.25$). The errors are evaluated over two plunging cycles.

Model	Network size	RMSE	MAE	R^2	rRMSE(in %)
LI	-	5.17e-01	3.03e-01	-1.01e-01	104.96
M0	100×5	1.02e-01	4.77e-02	9.53e-01	20.13
	100×10	5.62e-02	2.38e-02	9.87e-01	11.07
$\mathbf{M1}_{TM}$	100×5	7.04e-02	3.29e-02	9.78e-01	13.84
	100×10	5.73e-02	2.04e-02	9.85e-01	11.29
$\mathbf{M1}_{TM+BC}$	100×5	7.14e-02	3.25e-02	9.77e-01	14.02
	100×10	4.30e-02	1.76e-02	9.91e-01	8.45
$\mathbf{M2}_{TD+TL}$	100×5	4.84e-02	2.09e-02	9.88e-01	**9.49**
	100×10	4.23e-02	1.52e-02	9.91e-01	**8.29**

Table 4.9: Comparison of the pressure recovery accuracy of the standard, sequential, and backward compatible MB-PINN models for the periodic case, The models are trained with $\Delta t_{Bulk}/T = 0.5$ without satisfying the Nyquist criteria ($\Delta t_{Bulk}/T \leq 1/(2f_{max}) = 0.25$). The errors are evaluated over two plunging cycles.

Model	Network size	RMSE	MAE	R^2	rRMSE(in %)
M0	100×5	7.22e-01	4.87e-01	6.95e-01	54.71
	100×10	2.82e-01	1.77e-01	9.47e-01	21.74
$\mathbf{M1}_{TM}$	100×5	4.89e-01	3.12e-01	8.57e-01	37.12
	100×10	2.89e-01	1.57e-01	9.41e-01	22.11
$\mathbf{M1}_{TM+BC}$	100×5	4.63e-01	2.96e-01	8.72e-01	35.23
	100×10	2.48e-01	1.48e-01	9.57e-01	**19.21**
$\mathbf{M2}_{TD+TL}$	100×5	3.19e-01	1.96e-01	9.28e-01	24.37
	100×10	2.60e-01	1.42e-01	9.50e-01	**20.19**

4.4.3 Efficacy of models when trained over a long time domain

The models were trained over a time domain encompassing 10 plunging time periods. While there is sufficient temporal resolution, it is important to determine which variant is suitable especially for pressure recovery when trained over long-time domain data. Long-time domains might pose training difficulties to PINNs when there are strong flow-field gradients and a moving boundary, it is expected that sequential training variants will perform better as they reduce the problem complexity while keeping the models expressive.

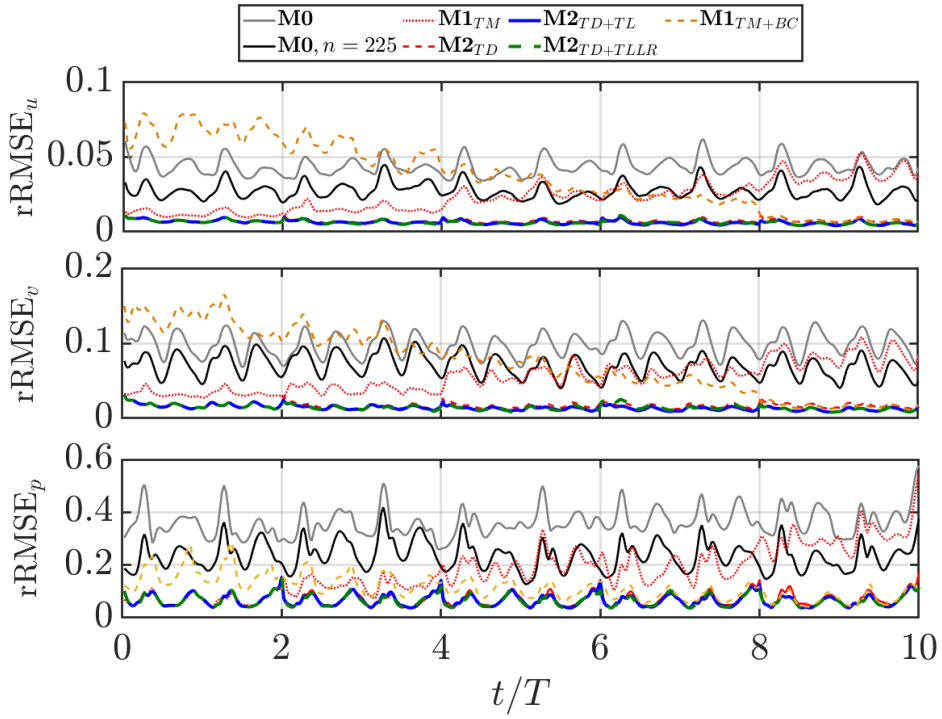


Figure 4.12: Comparison of snapshot-wise relative error in (a-b) velocity reconstruction and (c) pressure recovery obtained for the standard and sequential learning based MB-PINN models for the long time domain case study.

Sequential learning methods in the earlier subsections have so far been tested for sparsity, however, it is equally important to note that practical applications require these methods to be evaluated over long time domains. To isolate the effects of a long time domain usage, class I and II model variants were evaluated with $t/T \in [0, 10]$, and $\Delta t_{Bulk}/T = 0.05$ for

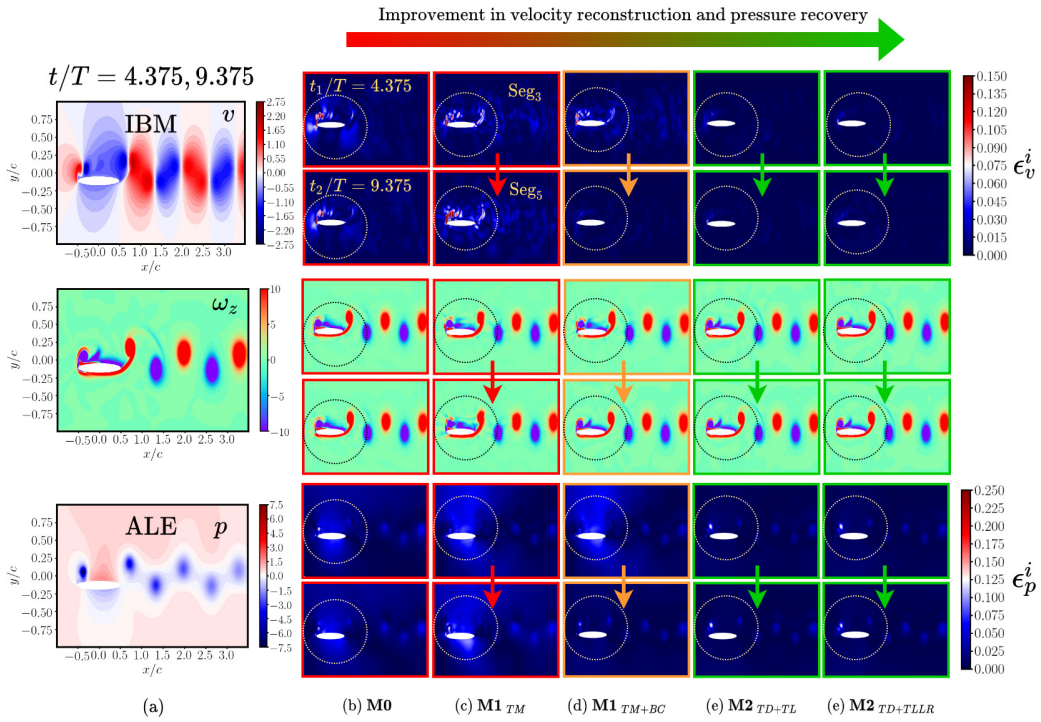


Figure 4.13: Comparison of (a) ground truth (top) y velocity component v pointwise error, (mid) vorticity, and (bottom) pressure pointwise error contours for predictions of (b) standard and (c-f) sequential learning based models across two different time domain segments

the periodic case. The choice of $\Delta t_{Bulk}/T$ ensured the Nyquist criteria was satisfied by a large margin.

Figure 4.12 shows that the baseline **M0**, of size $n = 100$ and $l = 10$, struggled to reconstruct the velocity, and even more so to recover pressure over this long time domain. This was the case even when a five times larger network was considered ($n = 225$ vs $n = 100$ hidden neurons with a fixed $l = 10$ hidden layers.) This was a consequence of the increased time domain size, which subsequently was not met with a similar increase in model expressivity, requiring the problem to be broken into simpler parts and that the model should not be too large. Sequential learning methods are promising since they either increase the problem complexity gradually (Class I), or, reduce the problem and model complexity through temporal domain decomposition (Class II).

In the sparsity case study presented earlier, Class I model variants performed relatively better than the **M0** models (see tables 4.8 and 4.9). But, the problem of error accumulation still persisted in Class I models, which was more evident in the long time domain case. This has been very clearly revealed in the relative error plots in figure 4.12. Here $\mathbf{M1}_{TM}$ can be seen accumulating error over time, for the later segments, while $\mathbf{M1}_{TM+BC}$ accumulating error over all the prior time segments. However, Class II models, $\mathbf{M2}_{TD}$, $\mathbf{M2}_{TD+TL}$, or the more computationally efficient $\mathbf{M2}_{TD+TLLR}$, performed better than Class I and **M0**, overall. The $\mathbf{M2}_{TD+TLLR}$ variant involved training the subsequent time segments over only two learning rate decay stages, with the starting learning rate being $2e-04$ as opposed to $1e-03$ for the other variants. As a result, $\mathbf{M2}_{TD+TLLR}$ required lower number of iterations ($2/3\times$), while still performing comparably to $\mathbf{M2}_{TD+TL}$ indicating a good computational efficiency.

This is confirmed in further qualitative inspection of the point-wise errors in velocity, vorticity and pressure as presented in figure 4.13. Quantitatively, the averaged error metrics in tables 4.10 and 4.11 also reveal that class II models are significantly better in performance.

Table 4.10: Long temporal domain: Comparison of accuracy details of reconstructed velocity component (v) by the standard, sequential, and backward compatible MB-PINNs when trained over 10 plunging cycles for the periodic case. Here, ($\Delta t_{Bulk}/T = \Delta t_{Phy}/T = 1/20$). The sequential and backward compatible variants are trained over 5 segments of 2 plunging cycles each.

Model	Network size	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
M0	100×10	4.96e-02	2.5e-02	9.86e-01	9.86
M0	225×10	3.51e-02	1.68e-02	9.95e-01	6.97
$\mathbf{M1}_{TM}$	100×10	2.75e-02	1.42e-02	9.96e-01	5.49
$\mathbf{M1}_{TM+BC}$	100×10	3.73e-01	1.88e-02	9.93e-01	7.45
$\mathbf{M2}_{TD}$	100×10	**8.39e-03**	**4.67e-03**	**9.99e-01**	**1.68**
$\mathbf{M2}_{TD+TL}$	100×10	**7.00e-03**	**3.87e-03**	**9.99e-01**	**1.41**
$\mathbf{M2}_{TD+TLLR}$	100×10	**7.59e-03**	**3.99e-03**	**9.99e-01**	**1.52**

Overall, it was observed in the periodic flow case that $\mathbf{M1}_{TM}$ and $\mathbf{M1}_{TM+BC}$ are conclusively less efficient, in case of temporal sparsity, or, training over long time domain.

Table 4.11: Long temporal domain: Comparison of the accuracy of recovered pressure (p) by the standard, sequential, and backward compatible MB-PINNs when trained over 10 plunging cycles for the periodic case. Here, ($\Delta t_{Bulk}/T = \Delta t_{Phy}/T = 1/20$). The sequential and backward compatible variants are trained over 5 segments of 2 plunging cycles each.

Model	Network size	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
M0	100×10	4.79e-01	3.43e-01	8.66e-01	36.25
M0	225×10	3.09e-01	2.16e-01	9.41e-01	23.62
$\mathbf{M1}_{TM}$	100×10	2.31e-01	1.65e-01	9.59e-01	17.94
$\mathbf{M1}_{TM+BC}$	100×10	1.45e-01	8.14e-02	9.85e-01	11.39
$\mathbf{M2}_{TD}$	100×10	**8.81e-03**	**5.58e-03**	**9.94e-01**	**7.05**
$\mathbf{M2}_{TD+TL}$	100×10	**8.36e-03**	**5.11e-02**	**9.95e-01**	**6.66**
$\mathbf{M2}_{TD+TLLR}$	100×10	**8.25e-03**	**4.99e-03**	**9.95e-01**	**6.60**

This is because they train over a gradually increasing time domain. As a result, a larger network might be required to accommodate the entire time domain, towards the end of the training stage (when training over the last time segment). Here the major impediment is the error accumulation problem of Class I models, which can be avoided with Class II ones. Moreover, the transfer learning based variants of Class II allow the overall problem size to be broken down, so that smaller networks could be trained to obtain a better accuracy, compared to **M0** or Class I models.

In both individual scenarios, the case of temporally sparse data, and the long time domain data well resolved in time, the time domain decomposition based Class II models with transfer learning ($\mathbf{M2}_{TD+TL}, \mathbf{M2}_{TD+TLLR}$) were successful. Specifically, the $\mathbf{M2}_{TD+TLLR}$ model was more computationally efficient.

In the next section, these models will be used to reconstruct a quasi-periodic flow case. In addition, a simple but efficient preferential spatio-temporal sampling will also be proposed to improve the reconstructions in a data-efficient manner.

4.5 TACKLING QUASI-PERIODIC FLOW WITH PREFERENTIAL SPATIO-TEMPORAL SAMPLING

Quasi-periodicity is encountered in certain kinematic regime in the context of unsteady flows past flapping wings due to the aperiodic behaviour of the near-field vortex structures, such regimes may enhance the aerdynamic load generation (Bose *et al.*, 2017; Majumdar *et al.*, 2020, 2022). Enhanced load generation might also lead to a higher performance efficiency (Majumdar *et al.*, 2022), which could be significant in the design of bio-mimetic devices. Quasi-periodic flow-field around a flapping body may involves minor variations in the position/organization of the vortices though it may look seemingly regular. However, one of the most important characteristics of the dynamics is the presence of incommensurate frequencies (and there combinations) in the system during quasi-periodicity. Thus, quasi-periodic flow serves as a good example of temporal complexity that involves a rich temporal spectrum, to evaluate the sequential learning based models proposed.

In the backdrop of our earlier discussion in section 4.4, for the quasi-periodic test case only $\mathbf{M2}_{TD+TL}$ and $\mathbf{M2}_{TD+TLLR}$ was evaluated against the **M0** model. For multi-scale problems, earlier studies have suggested Fourier feature embeddings (Wang *et al.*, 2021*b*), however, when the rich frequency content cannot be estimated *a priori* with temporally sparse data snapshots, such embeddings are not feasible without knowing the exact frequency scales. Even when a sufficiently large frequency scale was chosen to initialize the learnable Fourier feature layers, preliminary results indicated that the predictions could not improve significantly. Hence, Fourier feature embeddings were not employed in this study.

4.5.1 Importance of temporal resolution in the near-field

As mentioned previously, undersampling with the help of a vorticity cut-off was proposed in Chapter 2 to improve data efficiency, while maintaining accuracy. This cut-off-based undersampling gives uniform weight to all the grid points, with data where the

absolute vorticity $|\omega|$ is greater than a cut-off value $|\omega^*|$. For the quasi-periodic case study, a cut-off value $|\omega^*|= 0.1$ was chosen and a baseline domain Ω_1^r was chosen $\Omega_1^r : [-1.5c, 6.5c] \times [-2c, 2c]$ to cover two vortex couples in the wake. In this case the domain is larger than the periodic case study, due to larger vortices involved (an effect of lower Reynolds number and larger plunging amplitude).

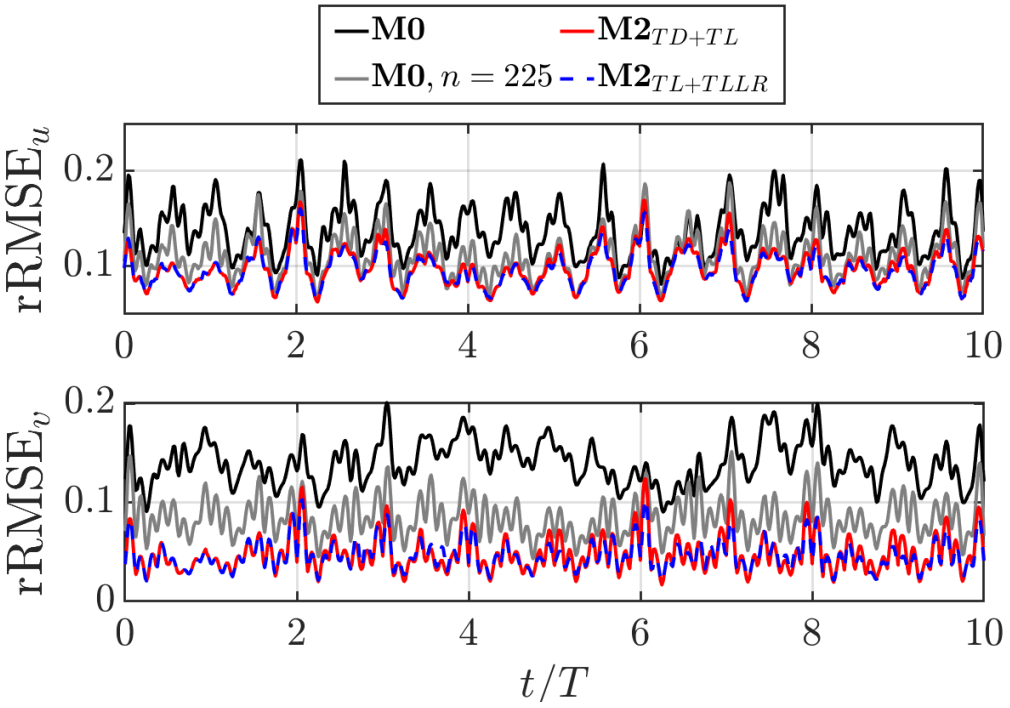


Figure 4.14: Comparison of relative error in time obtained for (top) $x-$ velocity component u, and (bottom) $y-$ velocity component v, reconstruction for $\Delta t_{Bulk}/T = 0.125$. Unless otherwise specified, the networks consist of 10 hidden layers with $n = 100$ neurons per hidden layer.

The **M0**, **M2**$_{TD+TL}$, and **M2**$_{TD+TLLR}$ models were trained over a long time domain for different $\Delta t_{Bulk}/T \in [0.125, 0.5]$, respectively. For $\Delta t_{Bulk}/T = 0.125$, the relative RMSE (rRMSE) over time (see figure 4.14) indicated that both **M2**$_{TD+TL}$, and **M2**$_{TD+TLLR}$ are equivalent and infact better than the baseline **M0** trained with different network sizes. Recall that, as per the Nyquist sampling criterion, $\Delta t_{Bulk}/T \leq 0.25$. The rRMSE worsened when this criterion was not satisfied (see table 4.12), with the best performance at $\Delta t_{Bulk}/T = 0.125$. A closer inspection of the contours in figure 4.15 reveals the

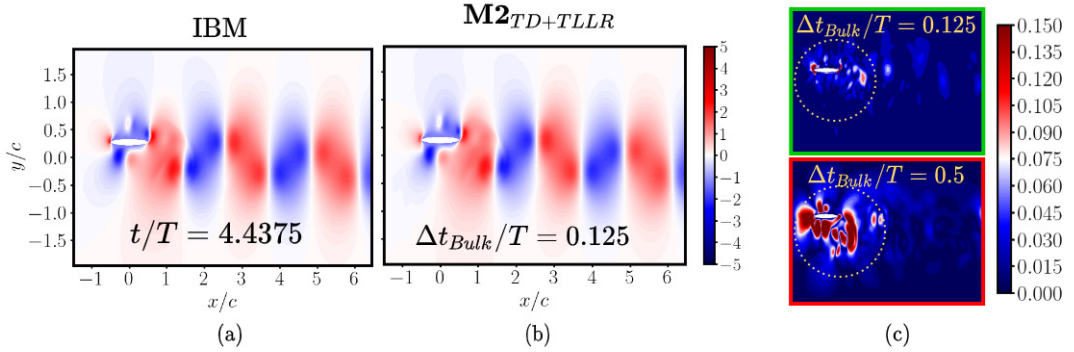


Figure 4.15: Comparison of (a) ground truth $y-$velocity and pressure data with (b) **M2**$_{TD+TLLR}$ reconstruction at $\Delta t_{Bulk}/T = 0.125$. The pointwise errors in (c) are compared for **M2**$_{TD+TLLR}$ cases with $\Delta t_{Bulk}/T = 0.125, 0.5$, respectively. Higher the sparsity the near-field reconstructions suffer the most while the wake is still reasonably reconstructed due to smoothness of the solution and no moving boundary in that region.

Table 4.12: Comparison of the accuracy of velocity component v reconstructed by standard MB-PINN and variants of seq-MB-PINN when trained with varying levels of temporal sparsity over 10 plunging cycles (Quasi-periodic case) considering the spatial domain Ω_1^r as previously described, and $\lambda_{IB} = 1$.

$\Delta t_{Bulk}/T$	**Network size**	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
		M0			
0.125	100×10	1.27e-01	7.33e-02	9.79e-01	14.03
0.125	225×10	7.72e-02	4.02e-02	9.92e-01	8.52
		M2$_{TD+TL}$			
0.125	100×10	6.34e-02	3.13e-02	9.95e-01	**7.02**
0.5	100×5	2.14e-01	1.03e-01	9.38e-01	23.67
		M2$_{TD+TLLR}$			
0.125	100×10	6.49e-02	3.09e-02	9.94e-01	**7.18**
0.5	100×5	2.21e-01	1.11e-01	9.33e-01	24.34

pointwise errors to be highest in the near-field region surrounding the moving boundary, and significantly lower errors in the wake region for both $\Delta t_{Bulk}/T$ considered. This points to the ability of sequentially trained MB-PINNs to reconstruct the wake even when $\Delta t_{Bulk}/T = 0.5$ does not satisfy the Nyquist criteria. However, accurate reconstruction of the velocity field and recovery of pressure in the near-field and moving body are crucial for the computation of the aerodynamic loads.

The near-field is affected possibly due to a relatively lower proportion of grid points in this region compared to the wake region. This results in PINNs automatically getting more updates from the wake region during training. Moreover, since the flow-field is smoother in the wake region without discontinuity like the moving body, the far-field data reconstruction and pressure recovery are more robust during temporal sparsity, long time domains, and spatially sparse data. Further, the moving body region faces three competing objectives in the form of physics, bulk data, and the no-slip-boundary-condition loss components. Whereas, the wake region has only two competing objectives, the bulk-data and the physics loss components, respectively. Thus, multiple competing objectives, lack of sufficient data, and strong flow-field gradients make the training difficult in the near-field region.

To tackle these challenges and improving the pressure recovery and subsequent load reconstruction, a preferential spatio-temporal sampling strategy (PVS), in addition to vorticity cut-off sampling has been proposed in the next section.

4.5.2 Preferential spatiotemporal sampling

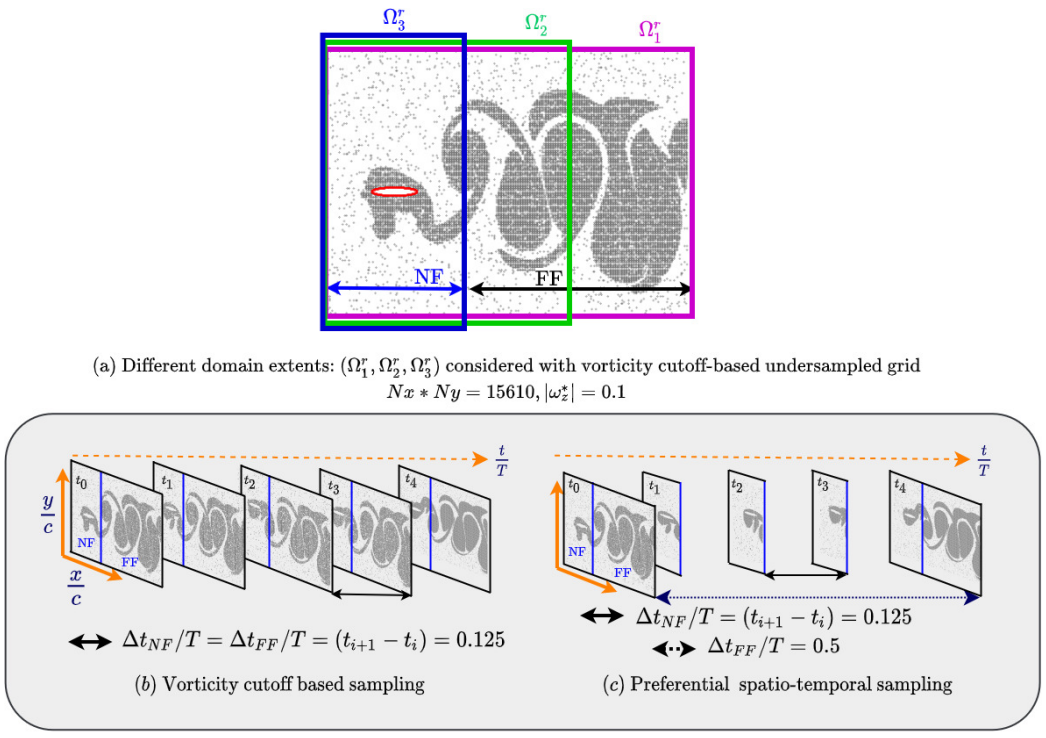
To improve the overall velocity reconstruction and pressure recovery, the preferential spatiotemporal sampling strategy (PVS) is proposed. In statistics, stratified sampling involves sampling from a population of data stratified or partitioned into sub-populations. The proposed PVS approach has similarities with such stratified sampling strategies. Here, the whole domain is first stratified/partitioned into near-field and wake region sub-domains, allowing flexibility to sample preferentially in space and time from the respective sub-domains.

Here, bulk data points are first undersampled using the vorticity cut-off based sampling technique previously described. Once obtained, one needs to keep the temporal resolution for the near-field data points such that, $\Delta t_{Bulk}/T >= 1/2f_{max}$. On the other hand, the far-field does not have to adhere to such strict resolution restriction and can be,

$\Delta t/T < 1/2f_{max}$. This is possible as PINNs are capable of recovering smoother features even under high temporal sparsity as seen earlier.

Data points in the near-field were sampled in time with $\Delta t^{NF}_{Bulk} = 0.125$; in the wake, data points were undersampled such that, $\Delta t^{FF}_{Bulk} = 4 \times \Delta t^{NF}_{Bulk} = 0.5$. This indirectly could give more weightage to the near-field data in time and space. As a result, the velocity flow-field reconstruction and pressure recovery are expected to improve overall. It will be later shown through experiments that increasing temporal sparsity in the near-field can be quite detrimental, especially when the spectral content in the flow becomes richer.

To demonstrate the efficacy of PVS, three different domains and datasets have been considered, $\Omega^r_1 : [-1.5c, 6.5c] \times [-2c, 2c]$, $\Omega^r_2 : [-1.5c, 3.5c] \times [-2c, 2c]$, and $\Omega^r_3 : [-1.5c, 1.5c] \times [-2c, 2c]$. Here, the domain size is decreased to study the importance of



(a) Different domain extents: $(\Omega^r_1, \Omega^r_2, \Omega^r_3)$ considered with vorticity cutoff-based undersampled grid $Nx * Ny = 15610, |\omega^*_z| = 0.1$

(b) Vorticity cutoff based sampling

(c) Preferential spatio-temporal sampling

Figure 4.16: Schematic comparing the vorticity-based sampling (b) and preferential spatiotemporal sampling - combined with the vorticity-based sampling - in the near and far-field regions (c).

Table 4.13: Details of training dataset for vorticity cutoff (VS), and preferential spatio-temporal (PVS) sampling over different domains Ω_1^r, Ω_2^r, and Ω_3^r for the quasi-periodic case considered. The training datasets are generated with vorticity-based selective sampling ratio $S_{\Omega_z} = 5\%$ with $|\omega_z^*| = 0.1$, the near-field region (NF) being $[-1.5c, 1c]$ in the x direction; $\Delta t/T^{NF}_{Bulk} = 0.125$; $\Delta t/T^{FF}_{Bulk} \in 0.125, 0.5$, depending on the far-field (FF) spatio-temporal sampling.

Sampling	Domain	$(\Delta t/T)^{NF}_{Bulk}, (\Delta t/T)^{FF}_{Bulk}$	N_{Bulk}	N_{Phy}	N^{NF}_{Bulk}/N_{Bulk} (in %)	N_{IB}/N_{Bulk} (in %)
Training datasets						
VS	Ω_1^r	(0.125,0.125)	$1.219e06$	$2.875e07$	14.31	52.57
PVS	Ω_1^r	(0.125,0.5)	$4.756e05$	$2.875e07$	36.67	134.76
VS	Ω_2^r	(0.125,0.125)	$5.805e05$	$1.808e07$	30.05	110.42
PVS	Ω_2^r	(0.125,0.5)	$2.904e05$	$1.808e07$	60.05	220.78
VS	Ω_3^r	(0.125,0.125)	$2.148e05$	$1.077e07$	100.0	298.40

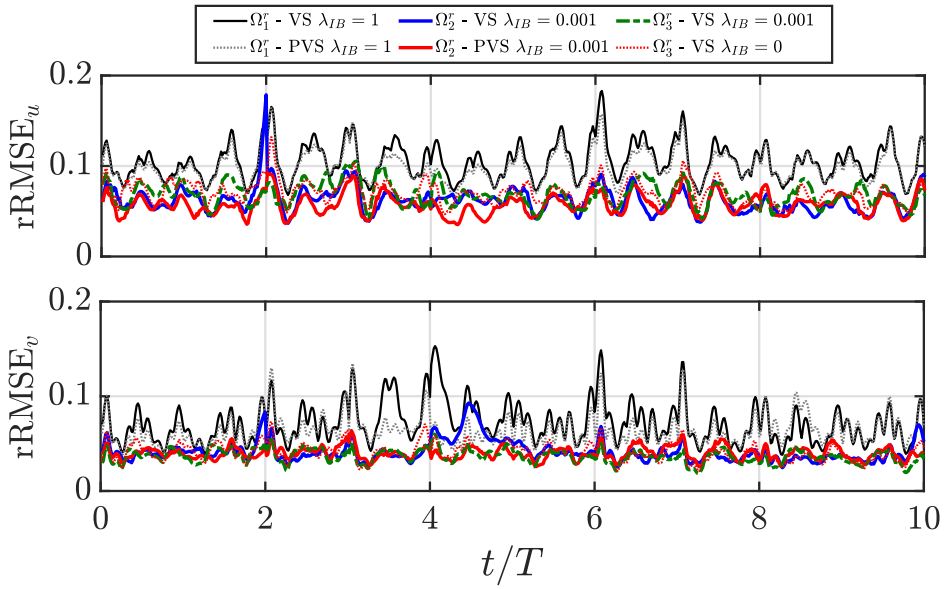


Figure 4.17: Comparison of relative errors in time for velocity reconstruction for **M2**$_{TD+TLLR}$ models trained with and without preferential sampling.

the near-field over wake region. Figure 4.16 shows the schematic, and table 4.13 presents more details of the datasets. Note that, Ω_1^r was earlier chosen so as to keep at least two vortex couples within the extent. Whereas, the horizontal (x-axis) extent of Ω_2^r is smaller and same as that of the periodic case study but captures only a single vortex couple; Ω_3^r is only the near-field region encompassing the moving body. The wake/far-field regions for Ω_1^r and Ω_2^r domains are, $\Omega_1^r - \Omega_3^r$, and $\Omega_2^r - \Omega_3^r$, respectively.

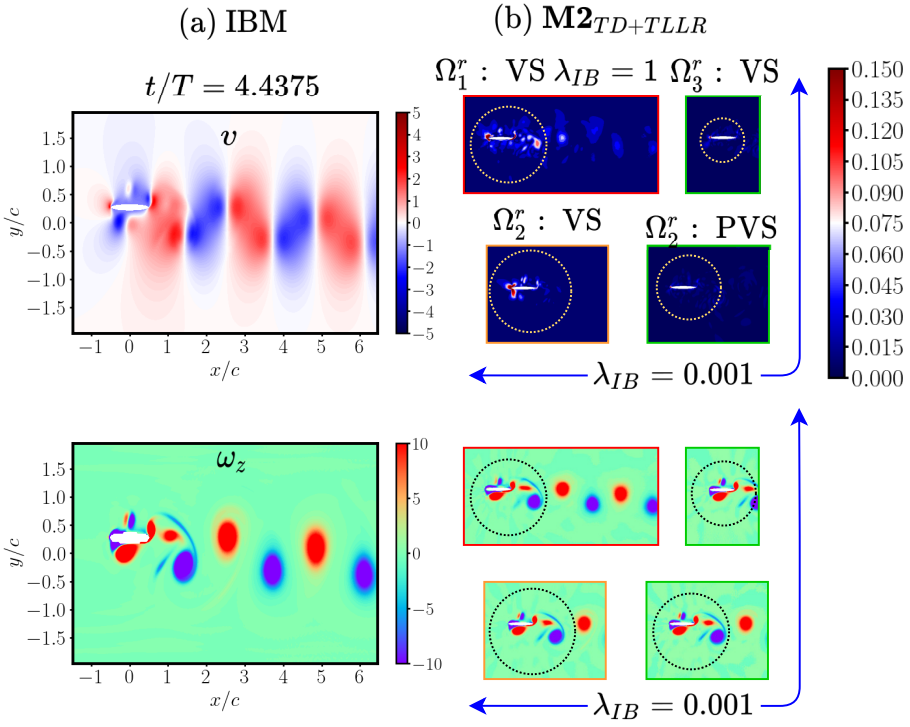


Figure 4.18: Comparison of, (a) ground truth IBM of velocity component (v), and, (b) point-wise errors in model reconstruction at a test time stamp of t/T = 4.4375. Corresponding vorticity snapshots are in the bottom row.

In addition to a relaxed $\mathcal{L}_{Phy}$, weight coefficient λ_{IB} of the $\mathcal{L}_{IB}$ was also relaxed. This was to ease the training by relaxing the competition between the three objectives, $\mathcal{L}_{IB}$ on the moving boundary, and $\mathcal{L}_{Bulk}$, $\mathcal{L}_{Phy}$ near the body. Relative velocity reconstruction errors in figure 4.17 and the average error metrics in table 4.14 show the remarkable power of combining the preferential spatio-temporal vorticity based sampling (PVS) along with the $\mathcal{L}_{IB}$ loss relaxation. The efficacy of this approach is even more evident

from figure 4.18, where the point-wise velocity reconstruction errors are seen to come down significantly as the domain size reduces along with PVS and λ_{IB} relaxation. Note that in the case of Ω_3^r, the near-field region alone suffices for good velocity reconstruction as indicated by the vorticity contours.

Table 4.14: Comparison of the accuracy of velocity component (v) reconstructed by the standard MB-PINN and variants of seq-MB-PINN when trained with varying levels of temporal sparsity over 10 plunging cycles (Quasi-periodic case) Unless otherwise specified, $\Delta t_{Bulk}/T = 0.125$ in case of vorticity cut off sampling (VS).

Domain	λ_{IB}	Sampling	**Network size**	**RMSE**	**MAE**	R^2	**rRMSE**(in %)
			M2$_{TD+TL}$				
Ω_1^r	1	$\Delta t_{Bulk}/T = 0.5$, VS	100×5	2.14e-01	1.03e-01	9.38e-01	23.67
Ω_1^r	1	VS	100×10	6.34e-02	3.13e-02	9.95e-01	7.02
Ω_2^r	1	VS	100×10	5.43e-02	2.72e-02	9.96e-01	6.00
Ω_3^r	1	VS	100×10	5.8e-01	2.85e-02	9.951e-01	6.58
Ω_1^r	1	PVS	100×5	5.23e-01	2.69e-02	9.964e-01	5.79
Ω_2^r	1	PVS	100×10	5.35e-02	2.86e-02	9.963e-01	5.92
Ω_2^r	0.001	VS	100×10	3.5e-02	1.49e-02	9.984e-01	3.88
Ω_2^r	0.001	PVS	100×10	3.35e-02	1.62e-02	9.986e-01	**3.7**
			M2$_{TD+TLLR}$				
Ω_1^r	1	$\Delta t_{Bulk}/T = 0.5$, VS	100×5	2.21e-01	1.11e-01	9.33e-01	24.34
Ω_1^r	1	VS	100×10	6.49e-02	3.09e-02	9.94e-01	7.18
Ω_2^r	1	VS	100×10	5.33e-02	2.55e-02	9.962e-01	5.9
Ω_3^r	1	VS	100×10	5.71e-02	2.76e-02	9.952e-01	6.47
Ω_1^r	1	PVS	100×5	5.86e-02	3.28e-02	9.956e-01	6.47
Ω_2^r	1	PVS	100×10	5.15e-02	2.5e-02	9.965e-01	5.7
Ω_2^r	0.001	VS	100×10	3.74e-02	1.57e-02	9.982e-01	4.15
Ω_2^r	0.001	PVS	100×10	3.6e-02	1.55e-02	9.984e-01	**3.99**
Ω_3^r	0.001	VS	100×10	3.23e-02	1.17e-02	9.986e-01	**3.66**
Ω_3^r	0	VS	100×10	3.57e-02	1.25e-02	9.983e-01	4.05

In quasi periodic flows, the dynamical behavior obtained across solvers would largely remain the same, though small local differences can exist due to differences in the underlying numerical schemes. This was also discussed in Majumdar *et al.* (2020), where minor differences were observed in the time histories of aerodynamic load coefficients obtained from IBM and ALE solvers, as the solutions evolved over time. As a result, the pressure recovered by PINN models trained on IBM data could still demonstrate high point-wise errors, when compared with other solvers (say an ALE solver).

To further demonstrate this, the pressure predictions of **M2**$_{TD+TLLR}$ models trained

with and without preferential sampling were compared with ALE pressure data in the same spirit as in section 4.4. The aforementioned discrepancy between solvers is clearly

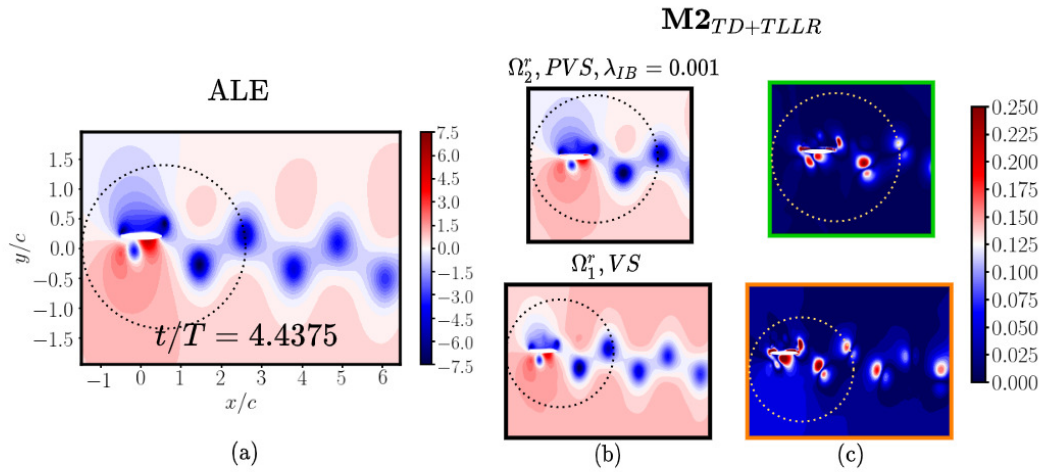


Figure 4.19: Comparison of pressure contours of (a) ALE and (b) $\mathbf{M2}_{TD+TLLR}$ model predictions at a representative test time stamp $t/T = 4.4375$, and (c) corresponding pointwise maximum-value normalized absolute error contours. In (b-c) Top row corresponds to the best $\mathbf{M2}_{TD+TLLR}$ model trained on Ω_2^r domain with preferential sampling and $\lambda_{IB} = 0.001$. In the bottom row, the predictions correspond to $\mathbf{M2}_{TD+TLLR}$ model trained on Ω_1^r domain with just vorticity cutoff sampling and $\lambda_{IB} = 1$.

inferred from the high pointwise errors wherever vortices are present in the flow (see figure 4.19) even for the best model obtained using preferential sampling and $\mathcal{L}_{IB}$ weighting. Given that pressure is a hidden variable, it is still necessary to determine whether the pressure recovered from PINNs are reasonable.

Although pressure is not directly available from the IBM results, the aerodynamic loads can be computed in the IBM routine using the momentum forcing term and temporal derivative terms of the momentum conservation equations (Majumdar *et al.*, 2020). Since computation of C_L directly depends on the pressure distribution on the moving body, it can serve as a good metric to indirectly determine the accuracy of the pressure recovery. Here, C_D is not chosen for quantitative validation of the time history because it can be quite sensitive to LEV and TEV reconstruction inaccuracies. The magnitude differences

in C_L and C_D can also be factor, with the PINNs models reconstructing the higher magnitude values better. Hence, errors in pressure recovery at the leading and trailing edges, and C_L - C_D magnitude differences could directly affect their reconstruction which have not been reported in recent works (Zhu *et al.*, 2024; Calicchia *et al.*, 2023; Wang *et al.*, 2025) for flows past moving bodies. This requires a deeper investigation into physics-based data normalization techniques which can improve C_L and especially C_D reconstructions irrespective of their magnitude differences, however, this is currently out of scope for the present study. In the rest of the discussion, C_L obtained from IBM and reconstructed from the PINN model have been quantitatively compared to evaluate the performance of pressure recovery (see table 4.15 for the accuracy details and figure 4.20 for C_L reconstruction plots for select cases). For a qualitative validation, the shape of the $C_L - C_D$ phase portraits has been considered in figure 4.21. Additionally, the Fourier spectrum of the C_D time histories have been considered to validate the dynamics of the flow (see figure 4.22).

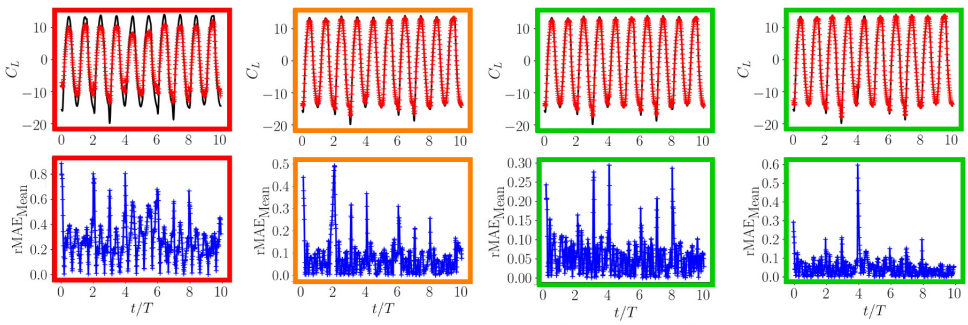


Figure 4.20: Comparison of (top) IBM ground truth and **M2**$_{TD+TLLR}$ reconstructed C_L time histories represented by black and red color lines, respectively, (a-d) across different bulk data sampling approaches and λ_{IB} values; and (bottom) corresponding relative mean absolute errors in time. Unless otherwise specified, λ_{IB} = 0.001

Note in table 4.15 that, improving the structure of the time series comes at the cost of under prediction of the C_L^{Peak}. Further training/fine tuning of the models can improve this. But notably, the errors have certainly dropped by relaxing the $\mathcal{L}_{IB}$. The effect of $\lambda_{IB} = 0$

Table 4.15: Comparison of accuracy of loads reconstructed by the seq-MB-PINN (TL and TLLR) for the quasi-periodic case, The models are trained with varying λ_{IB} at a fixed $\Delta t_{Bulk}/T = 0.125(1/8)$) satisfying the Nyquist criteria ($\Delta t/T \leq 1/(2f_{max}) = 0.25$). The sub-networks are of depth, $l = 10$, and width, $n = 100$ trained over 5 segments of 2 plunging periods each. Here, true $\hat{C}_L^{Peak} = 14.043$

Domain	λ_{IB}	Sampling	C_L^{Peak}	$\frac{C_L^{Peak} - \hat{C}_L^{Peak}}{\hat{C}_L^{Peak}}$	$\frac{\textbf{MAE}}{\lvert\hat{C}_L\rvert_{L_\infty}}$ (in %)	$\frac{\textbf{MAE}}{\lvert\hat{C}_L\rvert_{L_1}}$ (in %)
			TL			
Ω_1^r	1	Δt_{Bulk} = 0.5, VS	5.98	59.89	52.17	76.13
Ω_1^r	1	VS	10.91	22.32	25.04	36.54
Ω_2^r	1	VS	11.40	18.80	17.30	25.25
Ω_3^r	1	VS	12.21	13.04	13.58	19.83
Ω_1^r	1	PVS	10.65	25.54	22.07	32.21
Ω_2^r	1	PVS	12.27	12.61	15.59	22.76
Ω_2^r	0.001	VS	13.28	5.41	5.54	8.086
Ω_2^r	**0.001**	PVS	**13.37**	**4.80**	**4.48**	**6.54**
			TLLR			
Ω_1^r	1	Δt_{Bulk} = 0.5, VS	8.19	41.67	46.23	67.47
Ω_1^r	1	VS	9.83	30.50	25.66	37.46
Ω_2^r	1	VS	11.64	17.12	17.80	25.25
Ω_3^r	1	VS	10.44	25.59	18.91	22.76
Ω_1^r	1	PVS	11.95	14.87	18.79	27.42
Ω_2^r	1	PVS	12.31	12.18	15.46	22.56
Ω_2^r	0.001	VS	13.33	5.01	5.33	7.77
Ω_2^r	0.001	PVS	13.41	4.52	3.81	5.56
Ω_3^r	0.001	VS	13.60	3.14	3.44	5.03
Ω_3^r	0.0	VS	13.63	2.94	3.09	4.51

is such that the errors in load reconstruction become higher in the first subdomain which drop over subsequent domains. With a slight non-zero weighting, such as $\lambda_{IB} = 0.001$, the overall errors in all segments remain low as compared to a case when $\lambda_{IB} = 1$ (see figure 4.20). Qualitatively, the $C_L - C_D$ phase-portraits in figure 4.21 also show a distinct improvement when $\mathcal{L}_{IB}$ relaxation is coupled with preferential spatio-temporal sampling. This deduction is especially useful in the scenario when one needs to train MB-PINNs without any information of the moving body velocity. In this case, one would have to discard the predictions in the first subdomain and consider the results from the subsequent subdomains where transfer learning has come into effect. However, if one has the moving body velocity information *a priori*, then enforcing it through $\mathcal{L}_{IB}$ with a reasonably low λ_{IB} proves to be beneficial in load reconstruction over the entire time domain considered. Comparison of the frequency spectra from the reconstructed and IBM ground truth C_D in figure 4.22 indicates a good recovery of the rich temporal spectra by the **M2**$_{TD+TLLR}$ models compared to **M0**. Here, **M2**$_{TD+TLLR}$ is able to capture the peak locations (right frequencies), while **M0** fails to do so. This is possible because time domain decomposition simplifies the problem and makes it easy for the model to train on a relatively smaller subdomain.

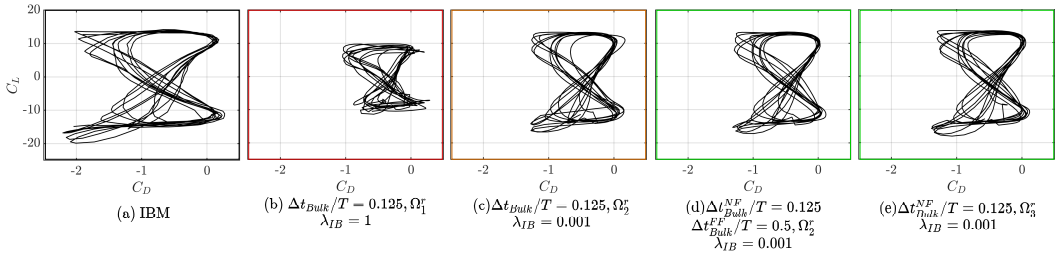
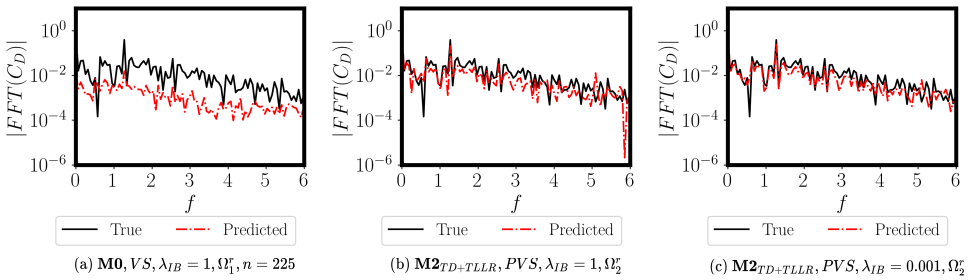


Figure 4.21: Qualitative of comparison of $C_L - C_D$ phase portraits reconstructed from (a) IBM ground truth data, and (b-e) predictions of **M2**$_{TD+TLLR}$ models across different bulk data sampling approaches and $\mathcal{L}_{IB}$ values.

Overall, under a fixed training budget, Class II models, especially, sequential learning using time domain decomposition with transfer learning are performant in dealing

with different temporal domain complexities. Moreover, it is beneficial to combine preferential spatio-temporal sampling, and relaxation of $\mathcal{L}_{IB}$ loss component, in addition to physics loss relaxation significantly, to improve the overall pressure recovery and load reconstruction for the quasi-periodic case.



(a) $\mathbf{M0}, VS, \lambda_{IB} = 1, \Omega_1^r, n = 225$ (b) $\mathbf{M2}_{TD+TLLR}, PVS, \lambda_{IB} = 1, \Omega_2^r$ (c) $\mathbf{M2}_{TD+TLLR}, PVS, \lambda_{IB} = 0.001, \Omega_2^r$

Figure 4.22: Comparison of Fourier spectrum of C_D for (a) **M0** model trained with $n = 225$ hidden neurons, $\lambda_{IB} = 1$ with vorticity cutoff sampling (VS), and (b-c) $\mathbf{M2}_{TD+TLLR}$ models trained with different λ_{IB} over Ω_2^r with preferential sampling (PVS).

4.6 SUMMARY AND CONCLUSIONS

The investigations in this chapter highlight the limitations of traditional PINNs for long-time integration of unsteady flow and flow-structure interaction problems, and demonstrates the benefits of decomposition-based strategies for addressing error accumulation and complex quasi-periodic dynamics. The earlier developed moving boundary-enabled PINN (MB-PINN) formulation (see section 2.5) under the immersed boundary aware (IBA) framework has been utilized for velocity reconstruction and pressure recovery.

Key difficulties for PINNs in long-time integration include temporal sparsity, long temporal domains and rich spectral content. In order to tackle these temporal domain complexities in the context of moving bodies using MB-PINN, two sequential learning strategies have been explored: - time marching with gradual increase in time domain size, and time domain decomposition with an overall reduced problem size. While the former

strategy struggles with error accumulation over long time domains, the latter approach combined with transfer learning effectively reduces error propagation and computational complexity. For the present problem of incompressible unsteady flows past a flapping airfoil exhibiting periodic and quasi-periodic dynamics, the time decomposition approach with preferential spatio-temporal sampling successfully improves both accuracy and efficiency for the pressure recovery and aerodynamic load reconstruction. Note that quasi-periodic dynamics brings with it rich temporal spectra which cannot be determined from temporally sparse data beforehand. This poses a serious challenge for the PINN models, and hence a proxy Nyquist criterion was defined to have safeguards while training the model to obtain satisfactory results. Sequential learning with time domain decomposition approach simplified the problem and a good reconstruction of the rich temporal spectra was obtained. The models were able to capture the right frequencies while the standard PINN model failed to do so. However, these methods are not yet directly applicable for strongly aperiodic, such as chaotic flows, due to the underlying neural networks' spectral bias, also the flow-field trajectories being vastly different under small variations in the initial condition.

When flow-field data is available—even if only sparsely—the time domain should be partitioned into subdomains that span one or two plunging time periods. Over-partitioning into smaller subdomains would necessitate training an excessive number of sub-networks, thereby increasing computational expenses and heightening the risk of over-fitting.

In this study the models trained operated only on a single parametric instance, however, in future, operator learning techniques can be adopted to handle multiple parametric instances, or different input conditions such as gusty inflows. This would be motivating towards building parametric surrogates for unsteady flow control. Also, the models need to become viable for training on more complex and realistic 3D flows, which requires further research on multiple fronts: the underlying neural network architecture, optimization of collocation points sampling, time-adaptive labeled data sampling, adaptive loss balancing

and optimization techniques which can accelerate loss convergence and training. The present study also holds promise for experimentalists, where PIV measurements of planar velocity fields can be utilized to recover hidden quantities in the same flavor as in the very recent works of Volk *et al.* (2025); Wang *et al.* (2025), and infer the underlying load generation mechanisms.

CHAPTER 5

OVERALL CONCLUSIONS AND FUTURE WORK

5.1 SUMMARY AND CONCLUSIONS

In the present work,while highlighting the gaps in earlier and contemporary literature in Chapter 1, the need for a unified, mesh-agnostic, immersed boundary–aware surrogate modeling framework using physics-informed neural networks (PINNs) was established for unsteady flows past moving bodies. The immersed boundary–aware (IBA) framework developed in Chapter 2 addressed this need by adopting an inertial frame of reference for the spatial domain and a body non-conformal setting inspired by the immersed boundary method. Two principal PINN formulations were proposed: the moving-boundary PINN (MB-PINN), which operates solely in the fluid region leveraging the Navier-stokes equations, and the moving-boundary immersed-boundary–based PINN (MB-IBM-PINN) leveraging the IBM formulation, which operates in both the fluid and solid regions. A non-intrusive pressure recovery task—treated as a hidden variable inference problem from coarse/sparse velocity measurements—was selected to assess the capability of the proposed formulations. Using a novel multi-part physics loss weighting, it was demonstrated that the MB-PINN and MB-IBM-PINN are equivalent for the pressure recovery problem when the MB-IBM-PINN is trained with data and collocation points restricted to the fluid region. The introduction of physics loss relaxation further proved to be an effective strategy to enhance loss convergence and the overall near-field velocity and pressure predictions under a fixed training budget. A novel physics based vorticity-cutoff sampling technique was implemented which reduced the requirement of overall data points while still maintaining good accuracy levels. These results collectively demonstrated that an immersed-boundary–aware PINN framework reconstructs velocity while recovering physically consistent pressure fields in unsteady flow–body interactions with good accuracy. Notably, of the two formulations, MB-PINN was favorable for

scenarios where the moving body shape, position and velocity are known *apriori*, where as MB-IBM-PINN would be suitable for scenarios where such information might not be available. The flow-field datasets were generated using an in-house GPU accelerated (see Appendix A for more details on the OpenACC based GPU acceleration strategy) discrete forcing IBM based unsteady flow solver.

To deepen the understanding of MB-PINN training behavior in moving-body problems, an additional investigation was carried out. A novel zonal splitting methodology and gradient-statistics–based metrics were proposed to analyse spatial imbalances in the loss component contributions across three regions: the near-field surrounding the moving body, the wake, and the outer far-field. Three trained models were considered for evaluation: a baseline MB-PINN without any loss relaxation of physics based sampling, an optimal MB-PINN trained with physics loss relaxation alone, and a MB-PINN model trained with both physics loss relaxation and physics-based sampling. In the first and third case, the contributions are highest from the near-field for both physics and bulk data losses. However, for the second case, the contribution for data loss still remains to be near-field, whereas the contribution from near-field dips and wake region contributes more to the physics loss component. Importantly, the vorticity-based under-sampling alleviated vanishing gradient issues, validating the utility of selective physics-based sampling. These insights provided a generalizable diagnostic framework for quantifying imbalanced contributions from spatial (or input) subdomains explaining the results observed in Chapters 2 and 3. It is important to note that this diagnostic approach is agnostic to neural network architecture and can thus be applied to both physics-based and purely data-driven models in future to discern which subdomains play an active role in training. This analysis can contribute towards further design of adaptive loss-weighting strategies and more balanced optimization procedures in the context of unsteady flows past moving bodies.

The potential of the IBA framework was also demonstrated for other relevant inverse

problems under different data availability scenarios in the context of unsteady flows past moving bodies in Chapter 3. A specific problem of interest was whether the PINN models are capable of recovering near-field pressure, body position, velocity and shape in the absence of such information (See sections 3.2 and 3.3). It was shown that in such scenarios where body position, velocity and configuration are not known, MB-PINN is not applicable as one cannot distinguish between fluid and solid regions in the spatial domain. Whereas, the IBM based formulation MB-IBM-PINN was found to be most suitable because the IBM formulation allows handling both fluid and solid regions together using a single set of governing equations. For this problem especially, the momentum forcing and source/sink terms were considered as hidden variables in addition to pressure. The obtained predictions for momentum forcing terms were used as masks to filter out the fluid and solid region. In another case study, super-resolution based pressure recovery was considered (see section 3.4) where velocity data obtained from coarse simulations was used to train the MB-PINN model. Once trained, the model was tested on the fine resolution grid. While the models performed only slightly better than linear interpolation, it is important to note that simultaneous super-resolution and pressure recovery is a challenging problem. It was hypothesized that for true super-resolution and accurate pressure recovery on high-resolution grids, the physics constraints must be more stringently satisfied—an aspect highly sensitive to hyperparameter settings, data sparsity, and sampling strategies. To overcome these limitations, adaptive and transfer learning–based approaches were identified as promising future pathways. This would greatly benefit the unsteady aerodynamics community as coarse simulations would suffice to obtain reliable high resolution predictions. It was also clearly shown in section 3.5 that the PINN models are not feasible for the forward problem simply due to the requirement of very small time steps which can drastically increase the training time. Hence, it is opined that CFD simulations are always a better option for forward problems especially if well-posed as compared to PINNs. Recent advancements in architecture, adaptive activation functions, sampling, loss weighting, domain decomposition and optimization

algorithms could possibly benefit the forward problem (see review papers of Karniadakis *et al.* (2021); Shukla *et al.* (2022); Patel *et al.* (2024); Toscano *et al.* (2025)). However, it is still left to investigate whether these approaches would result in faster training convergence and whether they can converge to a solution under ill-posed conditions such as a truncated spatial domain without exact boundary conditions for pressure as considered in the example in section 3.5.

It is important to recognize that complexities can not just arise in problem setups as above, but even in the temporal domain of unsteady flow-field datasets. In the context of unsteady flows past moving bodies, temporal domain complexities are of focus in this thesis. The datasets could become temporally sparse, or have been collected over long time domains to correctly characterize the flow-field dynamics, and the flow-field might may contain rich temporal spectra due to aperiodic flow dynamics. Note that although the plunging kinematics can be periodic, the resultant flow-field need not. To tackle these temporal domain complexities, sequential learning strategies were explored in Chapter 4. Along with the baseline MB-PINN from earlier studies in Chapters 2 and 3, two classes of sequential learning strategies were developed under the Immersed boundary aware framework: Class I - time marching and Class II - time domain decomposition. These strategies were first evaluated for a periodic case study (see section 4.4). The analysis showed that physics loss plays a crucial role under temporal sparsity—significantly improving temporal interpolation and aiding in pressure recovery. While both Class I and Class II models performed satisfactorily for short-duration temporally sparse datasets, Class II models consistently outperformed Class I for long time domains, even when the flow was periodic. The primary reason is the error accumulation inherent in Class I approaches due to progressive time-domain expansion, whereas Class II models mitigate this by leveraging domain decomposition and independent sub-networks for each temporal segment. Under a fixed training budget, Class II models thus emerged as the most accurate and computationally efficient option.

To further evaluate the robustness of the proposed Class II framework, a quasi-periodic case study was undertaken (see section 4.5) where the flow-field exhibits richer temporal spectra. In discrete-forcing IBM formulations, aerodynamic loads are typically computed as the volume integral of the momentum forcing terms and the total derivative terms on the left-hand side of the momentum conservation equation (Majumdar *et al.*, 2020). Pressure fields recovered from PINN-based reconstructions of IBM velocity data were used to estimate aerodynamic loads, which were subsequently validated against IBM simulation results. It is worth noting that, even in experimental studies, direct pressure measurements are difficult; instead, aerodynamic loads or surface pressures are typically obtained using load cells or piezoelectric sensors embedded in the test section.

For the quasi-periodic case, despite the periodic plunging motion, the near-field flow—particularly dominated by the leading-edge vortex (LEV)—introduces strong aperiodicity (Bose and Sarkar). In this region, three competing loss components coexist: bulk data loss, physics loss, and no-slip boundary condition loss. While these did not interfere significantly in the periodic case, their interaction in the quasi-periodic regime severely degraded pressure recovery, even under relaxed physics loss weighting. A joint relaxation of both the physics and no-slip boundary condition loss components markedly improved pressure reconstruction, as reflected in the recovered lift coefficients. Observing that the near-field contributed most actively to the training, while the far-field tolerated higher temporal sparsity, a preferential spatio-temporal sampling strategy was developed—building upon the earlier vorticity cutoff sampling technique. This enhanced sampling approach demonstrated that for accurate load reconstruction, the temporal resolution of the near-field velocity data must satisfy a Nyquist-like criterion based on the plunging frequency. Moreover, the results revealed that near-field velocity data alone could suffice for reliable load estimation; however, incorporating wake data extending up to approximately 2.5 chord lengths downstream yielded optimal reconstruction accuracy. These findings collectively underscore the strength of the IBA framework—particularly when equipped with vorticity cutoff and preferential sampling strategies—in handling

complex unsteady flow phenomena with rich temporal dynamics.

The IBA framework is mesh agnostic, and capable of handling moving boundaries in a body non-conformal setting from an inertial frame of reference. This core capability is crucial towards handling multiple moving bodies, or deforming structures without having to carry out case specific computational domain transformations which could impede the transferability of the developed models to other cases. Through various examples, it was demonstrated that IBA framework has strong capabilities in addressing hidden variable recovery problems and shows promising potential in complex hidden boundary estimation tasks relevant to the unsteady aerodynamics community. The proposed framework uniquely integrates physics-informed learning, physics loss relaxation, and preferential sampling strategies, enabling data-efficient model training. Based on the studies carried out so far, in order to obtain the best results under a fixed training budget and with minimal computational overheads, the combination of physics loss and no-slip boundary condition loss relaxation, combined physics based preferential spatio-temporal and vorticity cutoff sampling, and sequential learning are recommended. As part of ongoing work, this framework is being extended to tackle the hidden boundary estimation problem, highlighting the versatility and potential of the IBA framework as a generalizable surrogate modeling paradigm for inverse problems in unsteady flows past moving bodies.

5.2 SALIENT CONTRIBUTIONS

The salient contributions of the present thesis are discussed below, and also summaries in light of earlier and contemporary work are presented in table 5.1.

- Development of a mesh agnostic immersed boundary aware framework using physics informed neural networks for surrogate modeling of unsteady flows past moving bodies in a body non-conformal setting from an inertial frame of reference. Two PINN formulations were developed under this framework: MB-PINN which operates in the fluid region alone leveraging the Navier-Stokes equation, and the MB-IBM-PINN which leverages the IBM formulation operating in both fluid and

solid regions.

- Introduction of a novel multi-part or subdomain based loss weighting strategy to relatively explain the equivalence between two PINN formulations MB-PINN and MB-IBM-PINN. It was conclusively found that MB-PINN and MB-IBM-PINN are equivalent when the solid region points are discarded from training MB-IBM-PINN.

- It was evidently shown that MB-PINN is suitable under scenarios where body position and velocity are known. Whereas MB-IBM-PINN formulation is more useful when such body position and velocity information are not available.

- Demonstration of a novel hidden boundary estimation proof of concept, where the MB-IBM-PINN does not just recover pressure from velocity data but in addition estimate body position, velocity and shape. Albeit under the assumption of the rigid body translation, the MB-IBM-PINN holds promise for such scenarios but requires more advanced object detection algorithms which is part of a continuing work.

- Established the potential and limitations of the IBA framework for super-resolution and forward problems where the bulk velocity data is either of low-fidelity or not available at all.

- Development of a novel physics-based vorticity cutoff sampling and an extended version using preferential spatio-temporal sampling strategies allow training of models in a data efficient manner while also maintaining remarkable accuracy levels.

- Introduction of time domain decomposition based sequential learning coupled with transfer learning under the IBA framework to tackle temporal domain complexities arising in unsteady flow datasets such as (1) temporal sparsity, (2) long time domain, and (3) rich temporal spectra due to quasi-periodicity.

- Conclusive demonstration of the importance of physics loss under temporal sparsity of unsteady flow-field data. It was shown that pure data-driven models suffer in velocity data reconstruction while totally incapable of pressure recovery when pressure is not used at training. Employing a physics loss was shown to significantly improve temporal interpolation which is critical for reconstruction of unsteady flow-fields past moving bodies under temporal sparsity of data.

5.3 LIMITATIONS AND FUTURE SCOPE

During the course of the present study, a few contemporary works such as Huang *et al.* (2022); Zhu *et al.* (2023, 2024); Calicchia *et al.* (2023) along similar lines had been

Study	Method	Key contribution	Test case
Raissi *et al.* (2019*b*)	PINN	PINN for hidden variable recovery in a body attached frame of reference	Two dimensional flow past vibrating cylinder
Wang and Perdikaris (2021)	PINNs	Two sub-networks. One for the fluid and the other for the moving interface.	Stefan PDEs
Buhendwa *et al.* (2021)	PINNs	Volume of fluid based PINN was proposed	Two phase incompressible flow
Tang *et al.* (2022)	TL-PINNs	Transfer learning was adopted to reconstruct flow past a two dimensional fixed cylinder	Two dimensional fixed cylinder
Huang *et al.* (2022)	IB-PINN	IBM based PINN for time independent BVP	Fixed cylinder steady flow
Yang *et al.* (2023*b*)	FDM-PINN	Fictitious domain PINN for forward problems	Linear elliptic and parabolic steady and unsteady PDEs
Zhu *et al.* (2023)	PINN	Demonstrated the effectiveness of PINNs for dynamic interface problems	Steady state fluid-solid interaction without deformation of the structure
Chapter 2	MB-PINN/ MB-IBM-PINN	Immersed boundary aware framework, multi-part loss weighting, vorticity cutoff sampling	Two dimensional plunging foil system
Appendix B	MB-PINN	Zonal splitting approach and relative gradient statical analysis for analyzing input subdomain level contributions to training.	Two dimensional plunging foil system
Zhu *et al.* (2024)	PINN	Moving boundary problems including flow-field reconstruction, pressure recovery and forward problem (very short time domain).	Two and Three dimensional moving body forward and inverse problems
Chapter 4	MB-PINN and sequential learning variants	Extension of the IBA framework to handle different temporal domain complexities. Preferential spatio-temporal sampling also proposed.	Two dimensional plunging foil system

Table 5.1: Summary of author contributions including the present work.

published. Huang *et al.* (2022) proposed IB-PINN motivated by IBM, and demonstrated a forward problem use case with steady state fixed cylinder. However, the importance and role of solid region grid points was not investigated which was found to play a crucial role in determining the trainability of MB-IBM-PINN and its performance in pressure recovery (Chapter 2). In an other study, Calicchia *et al.* (2023) demonstrated the ability of the simple Navier Stokes based PINNs akin to MB-PINN to recover pressure from PIV obtained planar velocity measurements for swimming fish. Zhu *et al.* (2024) explored a PINN framework similar to MB-PINNs for forward and inverse problems involving single and multiple moving bodies in 2D and 3D showing the extended potential of what the proposed IBA framework could achieve when handling experimental data or 3D simulation data. Given how experimental data can be noisy, some earlier works on PINNS Raissi *et al.* (2020, 2019*b*,*a*); Wong *et al.* (2022) have discussed the robustness of PINNs under noise showing how physics loss acts as a regularizer during training allowing the deep neural network to learn key patterns within the data in spite of the noise as compared to pure data-driven neural network training. This adds potential promise to the IBA framework in its extensibility to noisy experimental data. In the present study, 2D incompressible flow past a plunging foil was considered as an example only to establish the capability of the IBA framework. Since the contemporary works showed the possibility of handling 3D problems, it would be worthwhile to consider more complex scenarios involving 3D periodic and aperiodic flows past multiple moving and deforming bodies such as a school of fish or cascading array of flapping foils/ vibrating cylinders. However, for the best performance, extensive investigations into the effectiveness of more recently proposed neural network architectures, loss formulations, optimization strategies, adaptive loss weighting, sampling, and activation functions would be required to ensure minimal human intervention (Karniadakis *et al.*, 2021; Shukla *et al.*, 2022; Toscano *et al.*, 2025; Jagtap *et al.*, 2022; McClenny and Braga-Neto, 2023).

It is important to note that in most practical scenarios, one might operate under a fixed time budget. Hence, as a practitioner, these methods need to be tailored keeping that in

mind. Note that the quasi-periodic case was considered in this study owing to its load enhancing characteristics discussed in literature (Lewin and Haj-Hariri, 2003; Majumdar *et al.*, 2020; Bose *et al.*, 2019). In the case of multiple moving bodies, deforming structures, and three dimensional flow, the temporal spectra could be much richer. Hence, suitable care might have to be taken while employing the sequential learning methods. In case of strongly aperiodic or chaotic flows, it might be prudent to develop stochastic or generative physics based deep learning models based on the score function based diffusion algorithm (Yang *et al.*, 2023*a*). This is because the flow-fields may vary solver to solver, and with varying spatio-temporal resolution within a solver. Hence, instead of reconstructing a single realization of the flow-field, capturing the flow-field statistics using generative models would be more beneficial for querying applications as in typical climate modeling applications (Mardani *et al.*, 2024; Sundar *et al.*, 2025). Moreover, chaotic flows require very long time domain integration. Hence sample efficient generative physics based models would be of interest here. Diffusion models are good candidates for this purpose.

High-fidelity CFD remains the gold standard for forward prediction in well posed settings where complete governing equations, initial-boundary conditions, and domain information are available. However, applicability of CFD becomes severely constrained in scenarios involving incomplete, sparse, or noisy data, or when certain physical quantities are not measured or stored. In contrast, the immersed boundary–aware (IBA) framework offers a complementary capability by enabling physics-consistent inference under such constraints. By embedding governing equations within a learning framework, IBA can reconstruct flow fields from partial observations, recover hidden variables such as pressure, and infer missing boundary information directly from data without requiring a fully specified problem setup. Moreover, operating on a fixed Eulerian grid, the framework avoids the complexities of mesh generation and domain transformation for moving or deforming bodies. As a result, IBA is particularly valuable in experimental settings, storage-limited simulations, and inverse problems, where CFD alone cannot be directly

applied or would require prohibitively expensive re-simulation. An important direction for future work is to extend the IBA framework towards identifying model discrepancies or unresolved physics from partial observations. In many practical scenarios—particularly in experimental settings or under coarse numerical resolution—the governing equations used may not fully capture all relevant physical effects, such as unresolved turbulence, structural coupling, or modeling assumptions. While conventional CFD assumes the governing equations to be exact, physics-informed frameworks offer the possibility to relax this assumption. In this context, the governing equations may be augmented with a learnable residual term, enabling the model to infer effective forcing or correction terms consistent with both data and known physics (Ahmed *et al.*, 2021; Pawar *et al.*, 2020; Raissi and Karniadakis, 2018). Such an approach would allow the IBA framework to identify regions in space and time where the assumed physics is inadequate, thereby providing insight into latent flow mechanisms or modeling errors. To ensure physical interpretability and avoid overfitting, appropriate regularization strategies—such as sparsity-promoting constraints (Bao and Gildin, 2017; Hoefler *et al.*, 2021; Huang *et al.*, 2026) or smoothness priors (Duhme *et al.*, 2026; Zhou and Wu, 2020)—would need to be incorporated. Additionally, the physics-based sampling strategies proposed in the present work, such as vorticity-guided sampling, could play a key role in localizing such discrepancies to dynamically important regions of the flow. This extension would position the IBA framework not only as a surrogate modeling tool, but also as a diagnostic framework for understanding and improving governing models in complex unsteady flows.

An other potential extension of the now well validated IBA framework would be to consider parametric input as in the domain of Operator learning (Kovachki *et al.*, 2024; Karniadakis *et al.*, 2021). For this purpose, the IBA framework can be very easily extended to incorporate physics informed deep operator networks (DeepONets) (Goswami *et al.*, 2023) or fourier neural operators (FNO) (Li *et al.*, 2021) as backbones. Although pointwise and continuous time-space models were considered in this study, one could

also consider building parametric surrogates using convolutional neural networks or transformer models which might not be grid agnostic. In this regard, the recently proposed universal physics transformer (UPT) (Alkin *et al.*, 2025) which is grid agnostic can be used for moving body flows. Over just the last two years, many foundational models with emergent zero-shot capabilities have been developed for text based image generation (Stable Diffusion (Rombach *et al.*, 2022)), zero-few shot generalisable large language model (the popular GPT-3 (Brown *et al.*, 2020)), and more recently a foundation model for the Earth system - Aurora (Bodnar *et al.*, 2024). With more open source scientific data sets available recently, such as 'The Well' (Ohana *et al.*, 2025), and the fundamental technology for foundational models being mostly relying on the transformer architecture developed first in the seminal work by Vaswani *et al.* (2023), IBA framework based foundational models can be developed for unsteady flows past moving bodies. However, it is important to keep in mind that foundational models are often expensive to train simply due to a large number of GPUs required (>O(100)), and huge datasets in TBs (Ohana *et al.*, 2025). Hence, being prudent, significant and simultaneous attention needs to be given towards developing high-quality datasets and resource efficient training of such models. While the capabilities and potential of the IBA framework using PINNs were demonstrated for two-dimensional unsteady flows, in future, given the advancements in architectures, training strategies, sampling strategies, and generative diffusion models, the IBA framework sets nothing but a cautious baseline to extend beyond two-dimensional flows and handle multiple-moving bodies, turbulent and three-dimensional flows which come with high data-level and modeling complexity with multiple spatio-temporal scales.

APPENDIX A

GPU ACCELERATION OF UNSTEADY FLOW SOLVER

A.1 INTRODUCTION

Recently, Majumdar *et al.* (2020) developed a discrete forcing type IBM (Kim *et al.*, 2001; Kim and Choi, 2019) solver to simulate and capture dynamical transitions in unsteady flow past a sinusoidally plunging rigid elliptic foil in the low reynolds number incompressible laminar flow regime. This solver was later parallelized by Shah *et al.* (2019) using OpenMP, a compiler directive based shared memory parallel programming framework, where a speed up of three times over the serial solver was reported. Although parallelization on CPUs using OpenMP allows for reduction in turn around time, these linear algebra routines can be accelerated further and significantly by offloading the computations on to GPUs(Xue and Roy). This is because, GPUs with their large number of processor cores and high throughput capacity promise massive performance enhancement of parallelisable linear algebra routines.

Typically, there exist three pathways to port a code to GPU: (i) performance tuned architecture specific third party libraries - CuSPARSE and CuBLAS (Naumov, 2011) to name a few, (ii) compiler directive frameworks - OpenACC (Chandrasekaran and Juckeland, 2017; Farber, 2016), and (iii) architecture specific parallel programming language extensions such as CUDA (Hoshino *et al.*, 2013) and OpenCL (Sugawara *et al.*, 2013). Of these pathways, compiler directive frameworks ensure minimum code intrusion and ease of parallelization cutting down the development time (Li *et al.*, 2016) as compared to full fledged parallel programming extensions like CUDA. The present work involves parallelization and performance enhancement of an in-house unsteady flow solver by offloading computationally intensive routines onto the GPU using OpenACC

framework. The Chapter outline is as follows: a discussion on the validation case setup and IBM solver is presented in section A.1. In section A.2, baseline solvers chosen for validation, the code parallelization strategy and code development cycle are discussed. Performance analysis of the GPU ported solver in the context of overall speedup and input scaling is presented and compared with that of the baseline solvers in section A.3. Finally in section A.4, conclusions and future work are discussed.

To validate the GPU ported solver, the plunging foil system in Eqs. 2.1 and 2.2 is considered (see Section 2.2 for more details). The flow equations (Eqs. (2.5) and (2.6)) are solved using finite volume based semi-implicit fractional step method (Choi and Moin, 1994) with the primitive variables being arranged in a staggered grid. Second order spatial and temporal discretizations are achieved using Adams-Bashforth and Crank-Nicolson schemes respectively. Pressure and velocity correction equations are iteratively solved using modified Gauss-Seidel successive over relaxation method with red-black tagging as used by Shah *et al.*Shah *et al.* (2019). A detailed description of the IBM algorithm used in the present study is presented in Majumdar *et al.* (2020). The computational domain considered is same as in Chapter 2.

A.2 PARALLELIZATION STRATEGY AND IMPLEMENTATION

A.2.1 Earlier work and baseline solvers

Two earlier versions of the in-house solver written in the C++ language are considered for validation and speedup analysis. These are, a serial implementation as in Majumdar *et al.* (2020), and an OpenMP based parallel implementation as in Shah *et al.* (2019), henceforth referred as **SOL0** and **SOL1**, respectively. **SOL1** uses an improved Gauss-Seidel successive over relaxation algorithm with red-black tagging of the mesh points to solve the pressure and velocity correction equations in order to avoid race condition and data dependency. Owing to interspersed parallel and sequential regions, a fork-join based parallelism was adopted by Shah *et al.* (2019). As a result, a speedup of almost

three times over **SOL0** was reported. The simulations using **SOL1** were executed on 16 CPU threads. However, the turn around time was still high. Hence, there was a pertinent need to accelerate the solver by offloading already parallel or parallelisable computations to the GPU. In the current work, using the OpenACC framework, a GPU compatible solver henceforth referred as **SOL2** is obtained.

A.2.2 System configuration

Development and testing were first carried out on a local system with an Intel i7 10th Gen processor with 6 processor cores and 12 threads, 16Gb Memory and a NVIDIA RTX 2060 Max Q GPU card with 1920 CUDA cores and 6GB memory. All the validation cases were however run on AQUA super cluster hosted at the P.G Senapathy Computing Centre for Computing resource in IIT Madras. With a multithreaded 20 core Xeon Gold 6248 processor that has a clock speed of 2.6GHz, AQUA's GPU nodes consist of 2 NVIDIA Tesla V100 GPU cards each with 5120 CUDA cores and 32GB GPU memory.

A.2.3 Software stack

Throughout the present work, NVIDIA's high performance computing software development kernel (HPC-SDK) version - 20.7 has been used both on the local system and AQUA super cluster. Amongst a variety of libraries and tools in HPC-SDK 20.7, Nsight-Systems and Nsight-Compute were used here for code profiling and NVTX for code annotation. Portable Batch System (PBS) was used for job scheduling on the cluster.

A.2.4 GPU implementation

The iterative code development cycle adopted in the present work using OpenACC to develop **SOL2** typically involves four steps: (i) analysis and profiling of the code for parallelisable hot-spots, (ii) parallelization of the identified code hotspots, (iii) testing and validation of the modification, and (iv) further optimization of the parallelized loops. Since OpenMP and OpenACC are both compiler directive based parallel programming

frameworks, the GPU implementation involved very minimal source code intrusions as shown in the schematic in figure A.1.

Once the code sections are ported using appropirate OpenACC **pragmas**, the compiler automatically decides how to offload the code sections onto the GPU at the time of compilation. The implicit decisions made by the compiler can be inferred from the compiler trace. Further optimizations can also be made from the suggestions and feedback offered by the compiler trace. This is one reason why OpenACC has a quick learning curve and cuts down the code development time (Chandrasekaran and Juckeland, 2017). The code development cycle described earlier was followed for each code block in the parallelisable regions iteratively until a satisfactory performance was achieved. Details of the incremental GPU implementation are presented below.

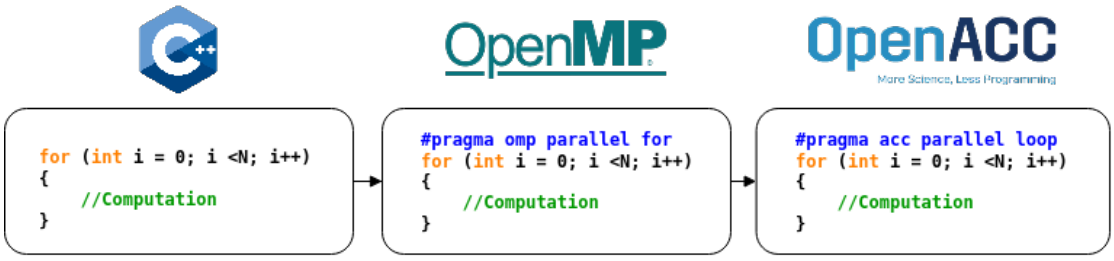


Figure A.1: Schematics representing the minimal code changes when using OpenMP and OpenACC

Analysis

Using Nsight-systems, the serial CPU solver **SOL0** was initially profiled on the local system in which the functions: (i) flagging/classification of solid and fluid points, (ii) iterative solvers for pressure and velocity corrections were identified to be the most time consuming regions followed by the code block involving (iii) body-force interpolation function as seen in figure A.2 and table A.1. The remaining regions of the code are executed serially and consume very little time as compared to the parallelisable sections.

As a preliminary analysis, the theoretical speedup S_{theory} possible through parallelization

Table A.1: Sequential CPU time for the functions identified to be parallelisable hot spots

Function	Computational time per time step (in seconds)
P iterative solver	3.835
Fluid/solid flagging	2.5515
U-V iterative solver	0.5518
Body force interpolation	0.0758

of code regions over the serial execution is given by the Amdahl's law, such that Amdahl (1967)

$$S_{theory} = \frac{1}{(1-p) + p/n}, \tag{A.1}$$

where, n is the number of processor cores and $p = T_{par}/T_{wall}$ is the parallel portion of the code in terms of wall time taken for execution of the code respectively. Here, T_{ser} and T_{par} are time taken by serial and parallel sections of the code, respectively. Whereas, total wall time of the code, T_{wall} is given by, $T_{wall} = T_{ser} + T_{par}$. In an ideal scenario, with effective parallelization of the parallel portion, the total wall time T_{wall} comes down closer to T_{ser}, with $T_{par} << T_{ser}$. In the present study, potentially parallelisable regions in **SOL0** comprise 99.8% of the total wall time when executed serially and hence, $p = T_{par}/T_{wall} = 0.998$. Thus, the maximum speedup possible in the limit of extremely large number of processor cores (i.e $n \to \infty$, therefore $p/n \to 0$) would be $S_{theory} = 1/((1 - 0.998) + 0) = 500$ times the sequential code. However, the theoretical speedup is not always achievable owing to various factors such as finite number of processor cores, hardware latency, data transfers between CPU and GPU, synchronization of threads and lower processor clock speed of GPUs as compared to CPU. Hence, the obtained speedup is much lesser. But, the effects of these factors can by minimized by profiling the code to identify the code hot spots needing optimization and appropriate OpenACC pragmas can be added.

Parallelization of loops

The parallelisable code blocks are often characterised by **for** loops. In the OpenMP

implementation **SOL1**, the directive **parallel for** along with clauses **default(shared)** for data sharing of shared variables and **schedule(dynamic)** for balancing the workload distrbution on the CPU threads were used for the hotspot regions earlier noted in figure A.2 and in Shah *et al.* (2019). In the GPU implementation, OpenACC directives such as **kernels** or **parallel for** can be added one by one before each potentially parallelisable code block /for loop/statement in the hotspots identified earlier (see Section A.2.4). The **kernels** directive is more flexible than the **parallel for** directive as it lets the compiler decide which regions within the scope of the construct are parallelisable and the compiler trace sheds more light on the specific optimizations and degrees of parallelism that can be brought in. To parallelize **for** loops with backward data dependencies, temporary variables can be allocated to swap and update the values. Although this would increase the memory requirement, it is offset by the improvement in speedup. Also, to avoid race conditions, selected variables in the scope of the code block that are offloaded onto GPU need to be appropriately privatised using the **private(variable list)** clause.

In the GPU implementation **SOL2**, classification/flagging of solid-fluid points function was first ported to GPU and then the pressure/velocity correction solver functions were ported using appropriate OpenACC directives and clauses. As mentioned in section A.2.4, body force interpolation procedure was also parallelized and offloaded onto the GPU in **SOL2**.

Optimization of loops

Following the feedback from compiler trace, appropriate directives/clauses were further added for optimization. Specifically, **routine** and **seq** directives were added wherever there were sequential user defined functions that needed to be executed on the GPU to avoid unnecessary data transfers between CPU and the GPU. Also, care was taken to ensure that these offloaded sequential operations are not computationally intensive. Additionally, **collapse** and **reduction** clauses were added for vectorization and to avoid race condition through simultaneous read and write operations respectively. Further

optimizations such as explicit mapping of for loops to the GPU threads and other intricate data management constructs can also be implemented (for more details, refer to Chandrasekaran and Juckeland (2017); Xue and Roy). However, in the present study, the code development cycle was stopped when a satisfactory speedup was obtained for **SOL2**, and the intricate optimizations were left for future work.

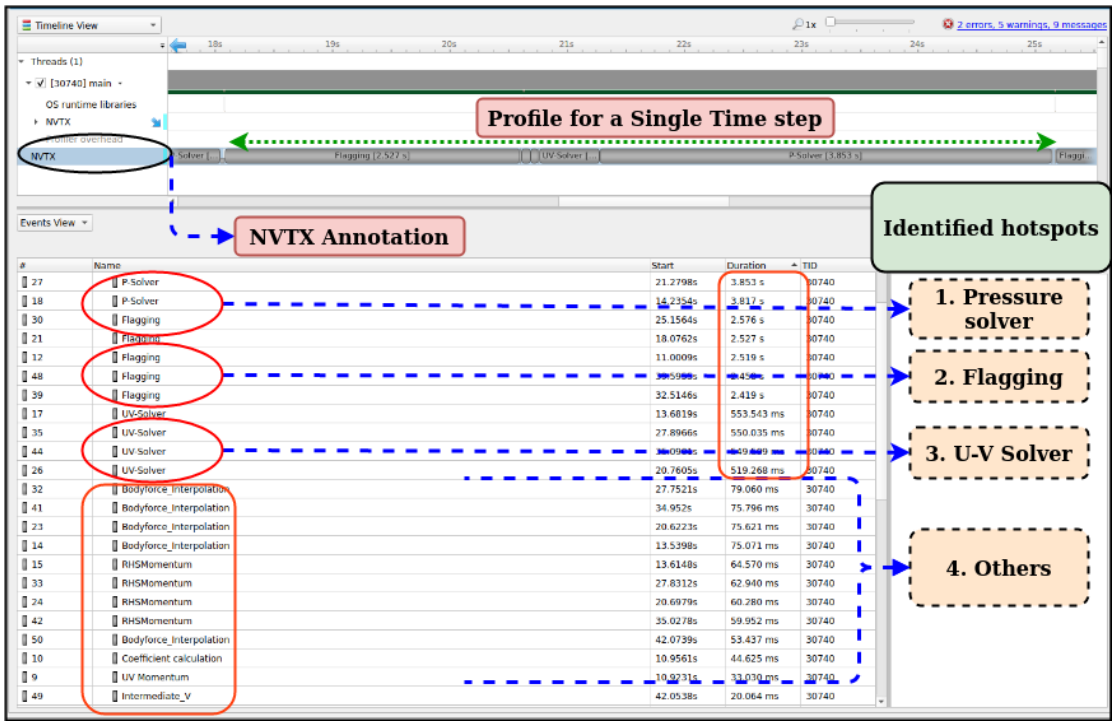


Figure A.2: Nsight-systems profiler output for the NVTX annotated serial version of the IBM solver for a single time marching step.

Data management

Data management is crucial in OpenACC implementation as CPUs and GPUs have different memory architectures. Because of this, variables are first allocated on the CPU and then copied to GPU where computations are then carried out. This leads to multiple data transfers between CPU and GPU which gets limited by latency overheads and the memory bandwidth of GPU, thereby impeding the solver performance. This can be avoided by minimising data transfers using OpenACC's data management directive (Chandrasekaran and Juckeland, 2017) appended with appropriate **copyin**,

copyout or **copy** clauses. This drastically improves the performance because all computations can be carried out at once when the required data for all of them are hosted on the GPU at the time of execution. In the present study, all the variables were copied to the GPU at the beginning of evey time marching step of the solver. This is because file write operations that need to be carried out at the end of every few time steps take place only on the CPU. Hence, data transfers although significantly minimized, were necessary at the beginning and end of every time step for **SOL2** as seen in figure A.3.

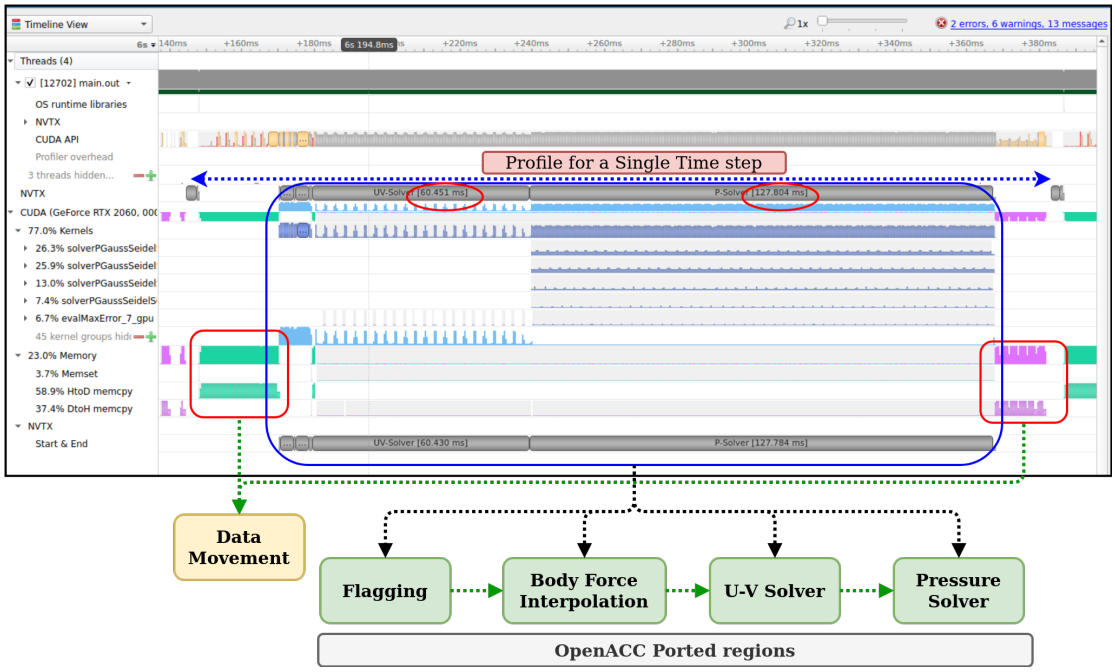


Figure A.3: Final Nsight-systems profiler outputs for the NVTX annotated OpenACC version of the IBM solver for a single time marching step.

A.3 RESULTS AND DISCUSSION

A.3.1 Validation studies

A Reynolds number, $Re = 500$, non-dimensional plunging velocity $k\bar{h} = 1.0$ with $k = 2\pi$ and $\bar{h} = 0.16$ are chosen for validation. The aerodynamic force coefficient time histories obtained from the simulations were compared with that of the baseline solvers **SOL0**, **SOL1** and the work of Khalid *et al.* (2018) in figure A.4. The results are in good

agreement with each other for **SOL0**, **SOL1** and **SOL2** solvers with almost no deviations, and also corroborate well with the results of Khalid *et al.* (2018). The small discrepancies observed in the aerodynamic coefficients between the present numerical studies and the literature (Khalid *et al.*, 2018), especially in drag coefficient (see figure A.4(b)), is attributed to underlying differences in the numerical method used in Khalid *et al.* (2018) and Majumdar *et al.* (2020).

A.3.2 Performance analysis

Speedup characterization

Cases for each solver setting are run thrice to obtain the average wall times for **SOL0**, **SOL1** and **SOL2**. In order to characterise the speedup of the parallel solvers **SOL1** and **SOL2**, the wall times of **SOL0** is considered as reference. The speedup S can be calculated using the wall times as follows

$$S_{SOLi} = T_{SOL0}/T_{SOLi} \quad \text{for i = 1,2} \tag{A.2}$$

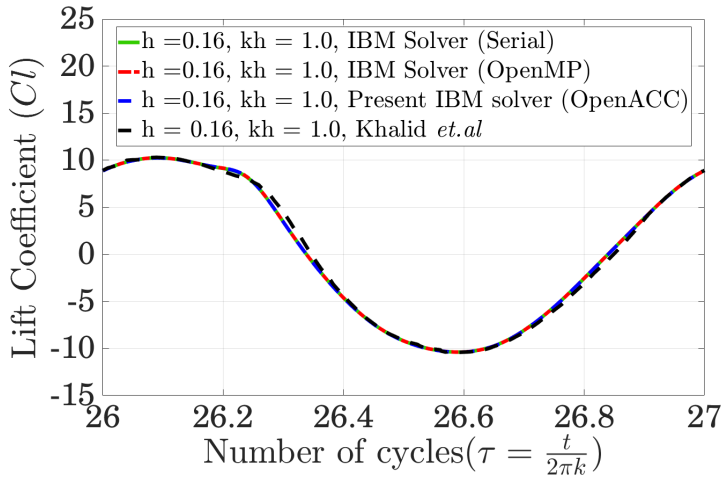
The relative speedup obtained by **SOL2** over **SOL1** is calculated as follows

$$S_{relative} = S_{SOL2}/S_{SOL1} \tag{A.3}$$

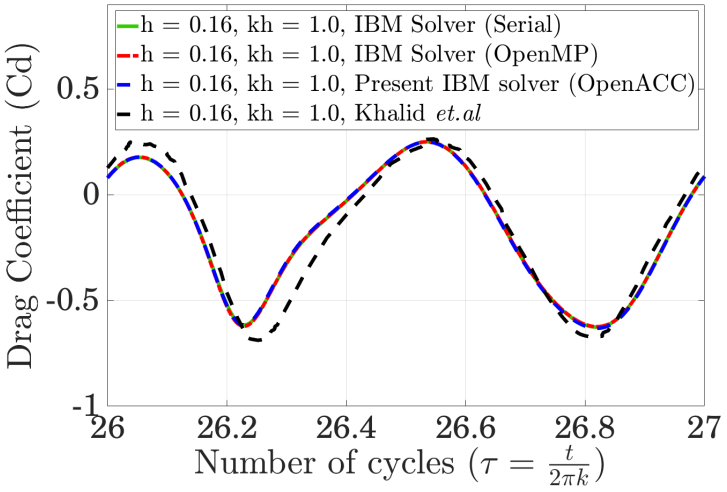
Additionally, the evolution of average speedup of **SOL2** over **SOL0** with the increasing number of time steps is considered for the first 10, 100, 1000 and 10000 time steps respectively (see figure A.5(a)). On considering the overall speedup for the first 1000 time steps alone, it is observed that a significant reduction in turn around times (see table A.2) is achieved that results in almost a speedup of the order $O(10^2)$ and $O(10)$ over **SOL0** and **SOL1**, respectively.

Input scaling performance

Along with speedup, given the massive number of GPU cores, it is important that scaling of speedup with respect to increasing mesh levels of the computation domain is studied. To this effect, three levels of mesh were considered, M1, M2 and M3 with 6, 12 and 18



(a)



(b)

Figure A.4: Plots comparing the (a) lift and (b) drag coefficient time histories for serial, OpenMP, OpenACC implementations with the results of Khalid *et al.*Khalid *et al.* (2018) for a sinusoidally plunging rigid elliptic foil.

Mesh Levels	M1 (6L)	M2 (12L)	M3 (18L)
Solver version			
SOL0	13140s	24663s	39994s
SOL1	4232s	8175s	11953s
SOL2	**244.4s**	**378s**	**368.3s**

Table A.2: Average wall time for serial CPU, OpenMP and OpenACC GPU solvers executed for the first 1000 time steps

lakh grid cells respectively. In figure A.5(a), as number of time steps are increased, a flattening trend is observed in the case of meshes M1 and M2 but in the case of M3, the speedup scales linearly with time upto 1000 time steps and then dips suddenly. The significant difference in speedup pattern for the mesh M3 compared to M1 and M2 is attributed to the uncertainty in job allocation on the GPU nodes of AQUA cluster. The jobs pertaining to M1 and M2 were often run on one GPU node while those of M3 ran on other GPU node with multiple concurrent jobs running simultaneously.

The overall and relative speedup obtained for **SOL2** over **SOL1** and **SOL0** for the first 1000 time steps are presented for the three mesh levels in A.5(b). The OpenACC based solver scales almost linearly with increasing mesh size compared to the sequential and OpenMP versions. This indicates effective utilization of GPU cores with increasing mesh size. This is in contrast to the OpenMP version of the solver where speedup over serial code is almost constant across mesh levels.

A.4 CONCLUSION

In order to reduce the turn around time of an IBM based unsteady flow solver, OpenACC was chosen as the GPU porting pathway owing to its similarity with OpenMP, minimal code intrusion and development time. Computationally intensive parallel routines

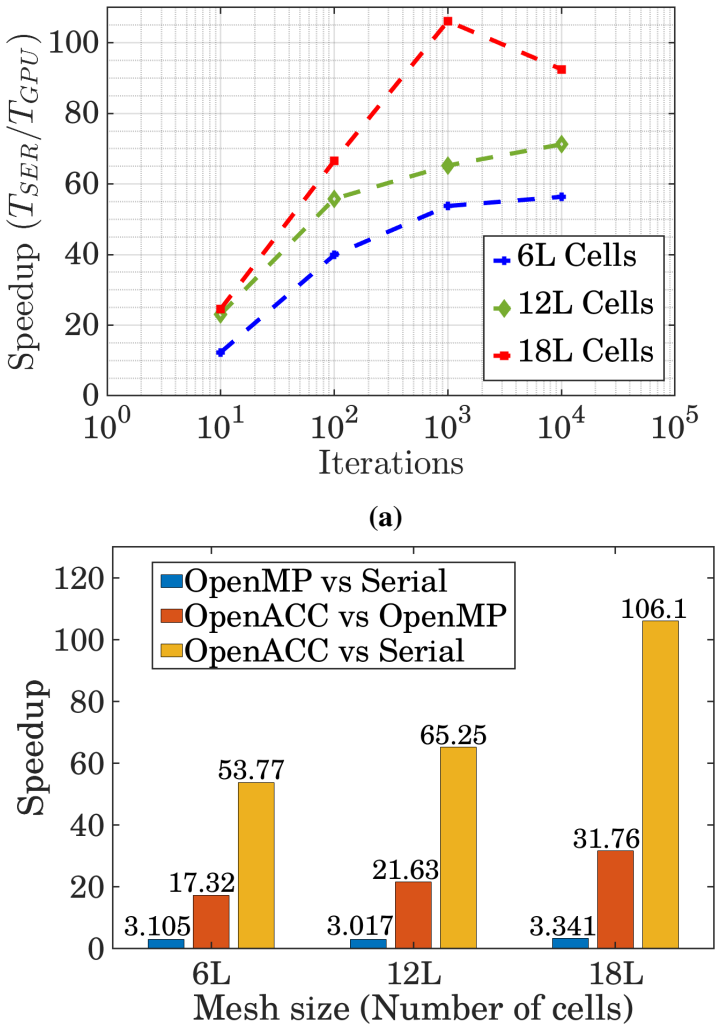


Figure A.5: (a) Evolution of speedup as a function of timestep for the GPU ported IBM solver at various mesh levels, and (b) bar graph depicting the relative speedup of OpenACC version obtained over serial and openMP versions at different mesh levels and for the first 1000 time steps

were offloaded onto the GPU with minimal data transfers using appropriate directives and clauses provided in the OpenACC framework by adopting an incremental code development cycle. Finally, significant speedups upto the order $O(10)$ and $O(10^2)$ over the OpenMP and serial solver versions were obtained, respectively. Improved performance of the GPU ported solver is a result of (i) parallelized code regions that were otherwise executed sequentially in the openMP version, (ii) optimized code blocks / for loops / functions with specifc directives and clauses ensuring no data dependency, data conflicts

and race condition, and (iii) minimized data transfers between CPU and GPU. With increasing mesh size, the speedup of OpenACC version scaled almost linearly owing to the effective utilization of the GPU cores as compared to the constant speedup of OpenMP version of the code. The parallelization strategy followed in the present work has been extended to other in-house IBM based non-linear fluid structure interaction solvers developed (Shah *et al.*, 2021, 2024*a*,*b*; Chatterjee *et al.*, 2024) **#TODO: add new citations of the other solvers developed.** .

APPENDIX B

UNDERSTANDING THE TRAINING OF PINNS FOR UNSTEADY FLOW PAST A PLUNGING FOIL

B.1 INTRODUCTION

Physics-informed neural networks have become promising for their use in solving complex inverse problems such as hidden physics recovery, data-driven equations discovery, uncertainty quantification, Plain vanilla PINNs (Raissi *et al.*, 2019*a*) are however difficult to train when the systems exhibit strong spatiotemporal gradients. Hence, recently, authors have proposed different strategies to train PINNs better, such as adaptive sampling, modified architectures, static and dynamic loss weighting.

Surrogate modeling of unsteady flow past moving boundaries such as flapping wings become challenging if the underlying high-fidelity simulation data used for such an endeavour is obtained using the immersed boundary method (Peskin, 2002; Balajewicz and Farhat, 2014). This is because, IBM uses a fixed Eulerian background grid for the fluid, whereas the solid boundary is described by a set of Lagrangian markers. At any time instant, there exist eulerian grid cells bounded by the solid boundary which consists of fictitious flow field data.

Flows past flapping wings are often characterized by strong flow-field gradients which makes it challenging for PINNs to train. Hence, in this thesis, an Immersed boundary aware framework based on PINNs inspired by the IBM was proposed (see Chapter 2 and also see Sundar *et al.* (2024)). Specifically within IBA framework, a moving boundary enabled PINNs (MB-PINNs) (Sundar *et al.*, 2024) formulation utilized a fixed Eulerian grid for the fluid domain discarding the solid region points at any given time, and the no-slip boundary condition was enforced directly on the Lagrangian markers unlike in

discrete forcing IBM (Majumdar *et al.*, 2020) where a sequence of interpolation steps is required to enforce boundary conditions. A combination of global physics loss relaxation and vorticity cutoff based under sampling was shown in Chapter 2 to improve data efficiency while maintaining good accuracy in velocity reconstruction and simultaneous pressure recovery as a hidden variable.

Since loss component weighting improves the loss balancing globally as reported by Wang *et al.* (2021*a*), it would be worthwhile to investigate if there exist local imbalances in the spatial domains of interest. This would also in a way determine which spatial region drives the training. One way of understanding how PINNs train is to look into the layer-wise loss component gradients as in Wang *et al.* (2021*a*) to understand the effect of competing objectives on the training. While these gradients are often visualized for an entire full batch setting, it is not visualized however in a localized context to understand the contributions from respective spatiotemporal zones. Visualizing the layer-wise gradients obtained for the loss components over specific spatiotemporal zones would highlight any imbalances in the gradient updates and also validate the need for localized weighting strategies further like in the self-adaptive weighting or residual-based attention approaches (Xiang *et al.*, 2022; Anagnostopoulos *et al.*, 2024).

Hence, the objectives of the current study are to investigate how loss contributions spatially are affected by physics loss relaxation combined with a physics-based undersampling, and devise metrics to quantify which spatial zone drives the training of the network. As an example, the flow past a plunging airfoil at a low Reynolds number is considered following Chapter 2. From Chapter 2, three MB-PINN cases are considered: one being a baseline without any loss weighting or under-sampling, and the other two with loss relaxation and undersampling.

The outline of this chapter is as follows. The methodology of MB-PINNs and understanding the loss component gradients is revisited in section B.2. The numerical

experiments and results are discussed in section B.3 and finally the conclusions are presented in section B.4.

B.2 METHODOLOGY

In this section, the problem setup, MB-PINN formulation, zonal splitting of loss component gradients, and metrics to quantify the zonal imbalances and determine the spatial zone driving the training will be discussed.

B.2.1 Problem setup

In the present study, the plunging foil example from Section 2.2 with $Re = 500$, $k = 2\pi$ and $h_a = 0.16$ is chosen aligning with Khalid *et al.* (2018) for gradient-based analysis of the MB-PINN models. The training data has been generated using the discrete forcing IBM-based unsteady flow solver (Majumdar *et al.*, 2020) where, additional momentum forcing ($\boldsymbol{f}$) and source/sink (q) terms are added to Eqs. (2.3) and (2.4), respectively to satisfy the no-slip boundary condition (see Eqs. (2.5 and (2.6) for the modified governing equations solved in IBM.).

B.2.2 Moving boundary enabled PINNs (MB-PINNs)

Inspired by the immersed boundary method, an immersed boundary aware framework using PINNs for surrogate modeling of unsteady flows past moving boundaries was explored in Chapter 2. A truncated spatial domain excluding solid region points Ω_f^r was chosen as shown in figure 2.1 to train the moving boundary enabled PINN(MB-PINN) models. It was shown in Chapter 2 that the MB-PINN (see figure B.1) was efficacious against solving velocity reconstruction and simultaneous pressure recovery given IBM data. A feed forward neural network with Swish activation as a backbone was used in Chapter 2 to map the spatio-temporal coordinates to the corresponding velocity and

pressure predictions. The loss function $\mathcal{L}$ can be written as follows

$$\begin{aligned}\mathcal{L} = \lambda_{Bulk}\mathcal{L}_{Bulk} + \lambda_{BC}\mathcal{L}_{BC} + \lambda_{IC}\mathcal{L}_{IC} \\ +\lambda_{IB}\mathcal{L}_{IB} + \lambda_{Phy}\mathcal{L}_{Phy}.\end{aligned} \tag{B.1}$$

Here, $\mathcal{L}_{\#}$ with the subscripts # = {Bulk, BC, IC, IB, Phy} denote the loss contributions from mean squared errors in interior bulk velocity data, boundary condition, initial conditions, no-slip boundary condition on the immersed boundary and the physics constraints, respectively and $\lambda_{\#}$ denotes the corresponding loss weights. Here, the boundary conditions include Dirichlet values at the inlet, upper and lower boundaries of the truncated computational domain Ω_f^r (see figure 2.1).

Throughout the study, $\lambda_{Bulk} = \lambda_{BC} = \lambda_{IC} = \lambda_{IB} = 1$, and the physics loss alone is relaxed. In addition to physics loss relaxation, a vorticity cutoff based sampling was proposed in Section 2.8 (also see Sundar *et al.* (2024)). Here, a vorticity cutoff $|\omega_z^*|$, is chosen and all those fluid data points retained such that $|\omega_z| \geq |\omega_z^*|$. Then based on a percentage sampling ratio $S_{\omega_z} = \dfrac{N^{sample}_{|\omega_z|<|\omega_z^*|}}{N_{|\omega_z|<|\omega_z^*|}} \times 100$, the remaining fluid points are undersampled.

Considering K_{MB} mini batches of the data, the network weights and biases together represented by θ are updated using the back propagated loss component gradients as follows

$$\begin{aligned}\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} - \eta_i \frac{1}{N_{MB}} \sum_{i'=iN_{MB}+1}^{(i+1)N_{MB}} \Big(& \lambda_{Bulk}\nabla_{\boldsymbol{\theta}}\mathcal{L}^{i'}_{Bulk} \\ & + \lambda_{BC}\nabla_{\boldsymbol{\theta}}\mathcal{L}^{i'}_{BC} \; + \; \lambda_{IB}\nabla_{\boldsymbol{\theta}}\mathcal{L}^{i'}_{IB} \\ & + \; \lambda_{IC}\nabla_{\boldsymbol{\theta}}\mathcal{L}^{i'}_{IC} \; + \; \lambda_{Phy}\nabla_{\boldsymbol{\theta}}\mathcal{L}^{i'}_{Phy} \Big),\end{aligned} \tag{B.2}$$

where η_i is the learning rate corresponding to i^{th} iteration and each epoch corresponds to $K_{MB} = N_{train}/N_{MB}$ iterations. Here, the MB-PINN models were trained using the stochastic gradient descent based optimization algorithm ADAM (Kingma and Ba, 2017). Unless otherwise specified, a mini-batch size of $N_{MB} = 1500$ was chosen with a learning

rate step decay $lr = [1e-03, 53-04, 1e-04]$ where each learning rate step consisted of 5e05 training iterations.

The loss components $\mathcal{L}_{IB}$, $\mathcal{L}_{BC}$ don't depend on the interior spatial coordinates, and $\mathcal{L}_{IC}$ considers a data snapshot only at $t/T = 0$. They play an important role in ensuring the hidden variables are recovered appropriately (Buhendwa *et al.*, 2021; Lucor *et al.*, 2022). However, $\mathcal{L}_{Bulk}$ and $\mathcal{L}_{Phy}$ are computed from spatial coordinates interior to Ω^r_f. Given that spatially, there are zones where strong, weak or no vortices exist, it would thus be interesting to investigate the contributions from different spatial regions internally in the domain.

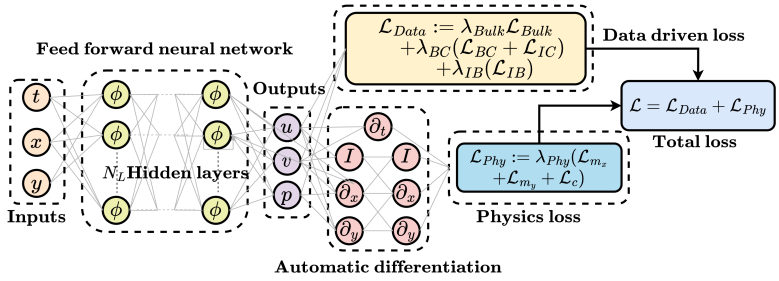


Figure B.1: MB-PINN schematic (Image adapted from Sundar *et al.* (2024).)

B.2.3 Zonal splitting of layer-wise gradients

While global loss weighting improves training and achieves global loss component balancing, that might not be the case locally. This could be due to varied contributions from each spatial region leading to different effective learning rates in each spatial region. To analyse this, once the MB-PINNs are trained or even at any intermediate learning stage, the layer-wise gradients can be computed overall for samples over the entire domain and for points sampled from select spatial zones. In the present study, three spatial zones are defined (see figure B.2), namely, the moving body zone ($Z1 \subset \Omega^r_f$ such that $-1 < x < 1$ and $-0.5 < y < 0.5$), the wake zone ($Z2 \subset \Omega^r_f$ such that $1 < x$ and $-0.5 < y < 0.5$), and the outer zone ($Z3 \subset \Omega^r_f$ such that $Z3 = \Omega^r_f - Z1 \cup Z2$).

Note that the overall spatial domain is Ω^r_f which is $\Omega^r = [-1c, 3.5c] \times [-1c, 1c]$) minus the solid region points at any time $t \in [0, 2]$. Also note that, zonal splitting is not carried out in temporal domain.

Given training velocity data $\hat{\boldsymbol{u}}$ at these spatial coordinates across all time, the components $\mathcal{L}_{Bulk}$ and $\mathcal{L}_{Phy}$ in Eq. B.1 can hence be split into contributions from respective spatial-zones as follows

$$\mathcal{L}_{Bulk} = \frac{1}{N_{MB}} \left(\sum_{i=1}^{3} \left(\sum_{j=1}^{N_{Zi}} \|\boldsymbol{u}^j - \hat{\boldsymbol{u}}^j\|^2 \right) \right) \tag{B.3}$$

$$= \left(\sum_{i=1}^{3} \left(\frac{N_{Zi}}{N_{MB}} \sum_{j=1}^{N_{Zi}} \frac{\|\boldsymbol{u}^j - \hat{\boldsymbol{u}}^j\|^2}{N_{Zi}} \right) \right) \tag{B.4}$$

$$= \left(\sum_{i=1}^{3} \left(p_{Zi} \sum_{j=1}^{N_{Zi}} \frac{\|\boldsymbol{u}^j - \hat{\boldsymbol{u}}^j\|^2}{N_{Zi}} \right) \right) \tag{B.5}$$

$$= \left(\sum_{i=1}^{3} \left(p_{Zi} \mathcal{L}^{Zi}_{Bulk} \right) \right) \tag{B.6}$$

It is seen that $\mathcal{L}_{Bulk}$ is a weighted sum of zonal mean squared errors where p_{Zi} for $i = 1, 2, 3$ is the proportion of sample points from zone Zi with respect to the overall mini-batch sample. Similarly, $\mathcal{L}_{Phy}$ can be split into respective contributions from the spatial zones as follows

$$\mathcal{L}_{Phy} = \left(\sum_{i=1}^{3} \left(p_{Zi} \sum_{j=1}^{N_{Zi}} \frac{\|\boldsymbol{r}(\boldsymbol{u}^j)\|}{N_{Zi}} \right) \right) \tag{B.7}$$

The above expressions can be simplified as

$$\mathcal{L}_* = \left(\sum_{i=1}^{3} \left(p_{Zi} \mathcal{L}^{Zi}_* \right) \right) \tag{B.8}$$

for $* = \{\text{Bulk, Phy}\}$. Now the gradient vector $\nabla_\theta \mathcal{L}_*$ required to be computed at every training iteration to update the network parameters can be further expressed in terms of the zonal contributions as

$$\nabla_\theta \mathcal{L}_* = \left(\sum_{i=1}^{3} \left(p_{Zi} \nabla_\theta \mathcal{L}^{Zi}_* \right) \right). \tag{B.9}$$

The above zonal splitting further indicates that in addition to loss component weighting, the data and collocation points sampling can modify the contributions from spatial zones or even spatio-temporal zones of interest. Hence, the zonal contributions can't be investigated independent of the sample proportions as the overall loss gradient is the weighted sum of the gradients obtained over each zone.

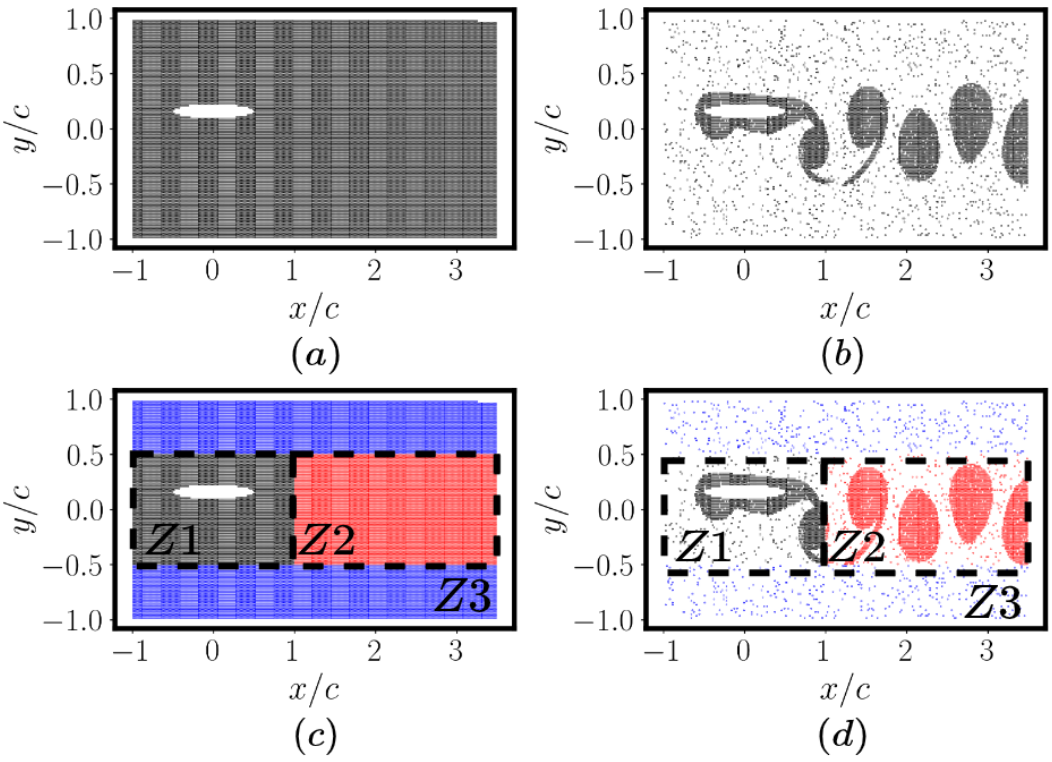


Figure B.2: Spatial grids of (a) CI and (b) CI-S5 data sets and the corresponding (c-d) zonal splitting of the grids into Z1- moving body, Z2 - wake and Z3 - outer zones, respectively

B.2.4 Metrics to diagnose zonal imbalances

Visualising and comparing the distribution of layerwise loss component gradients often sheds light on the extent of balance between the components (Wang *et al.*, 2021*a*) globally or locally. In addition, as seen previously, since gradient updates also depend on the proportion of zone specific coordinates with respect to the overall sample $p_{Z\#}$, this has to be weighed in while quantifying the relative contributions from each spatial zone. To

quantify which zone drives the training for components $\mathcal{L}_*$ with $* = \{\text{Bulk, Phy}\}$, the following zonal proportion weighted relative gradient statistics based metrics are thus computed

$$\mu_*^{Zi} = p_{Zi} \frac{\overline{|\nabla_\theta \mathcal{L}_*^{Zi}|}}{\overline{|\nabla_\theta \mathcal{L}_*|}} \tag{B.10}$$

$$\sigma_*^{Zi} = p_{Zi} \frac{\text{std}\{|\nabla_\theta \mathcal{L}_*^{Zi}|\}}{\text{std}\{|\nabla_\theta \mathcal{L}_*|\}}. \tag{B.11}$$

Once MB-PINNs are trained, the mean and standard deviation of the gradient magnitudes $\overline{|\nabla_\theta \mathcal{L}_*|}$, $\text{std}\{|\nabla_\theta \mathcal{L}_*|\}$ are computed over the network parameters for a mini-batch sample from the entire spatio-temporal domain and also for samples from respective spatial zones. The metrics $\mu_*^{Zi}, \sigma_*^{Zi}$ for $i = \{Z1, Z2, Z3\}$, and $* = \{\text{Bulk, Phy}\}$ are then computed.

B.3 RESULTS AND DISCUSSION

In the present study, three MB-PINN test cases from Chapter 2 are considered for analysis; a model without any physics loss relaxation or under sampling (case 1), a model with physics loss relaxation alone (case 2), and a model with both physics loss relaxation and vorticity cutoff based undersampling (case 3). The true and predicted velocity x-component, associated point-wise errors, and vorticity contours for these cases are presented in figure B.3 where the errors are prominent in zone $Z1$ and $Z2$ for case 1 but they almost vanish for cases 2 and 3 (see Figs. B.3(c-e)). The accuracy details are also presented in table B.3. To further understand which spatial zones in the flow contribute to the training and determine if there exist spatial imbalances, the loss component wise gradient distributions are analysed. The metrics μ_*^{Zi} and σ_*^{Zi} for $i = 1, 2, 3$ and $* = \{\text{Bulk, Phy}\}$ described in Eqs. (B.10) and (B.11) are presented in table B.4 for all the test cases discussed below.

B.3.1 Case 1: Baseline

It is observed that the first and last layer gradient distributions overall (see Figs. B.4(a)and B.4(d)) indicate an imbalance in contributions more prominent in the first layer. The

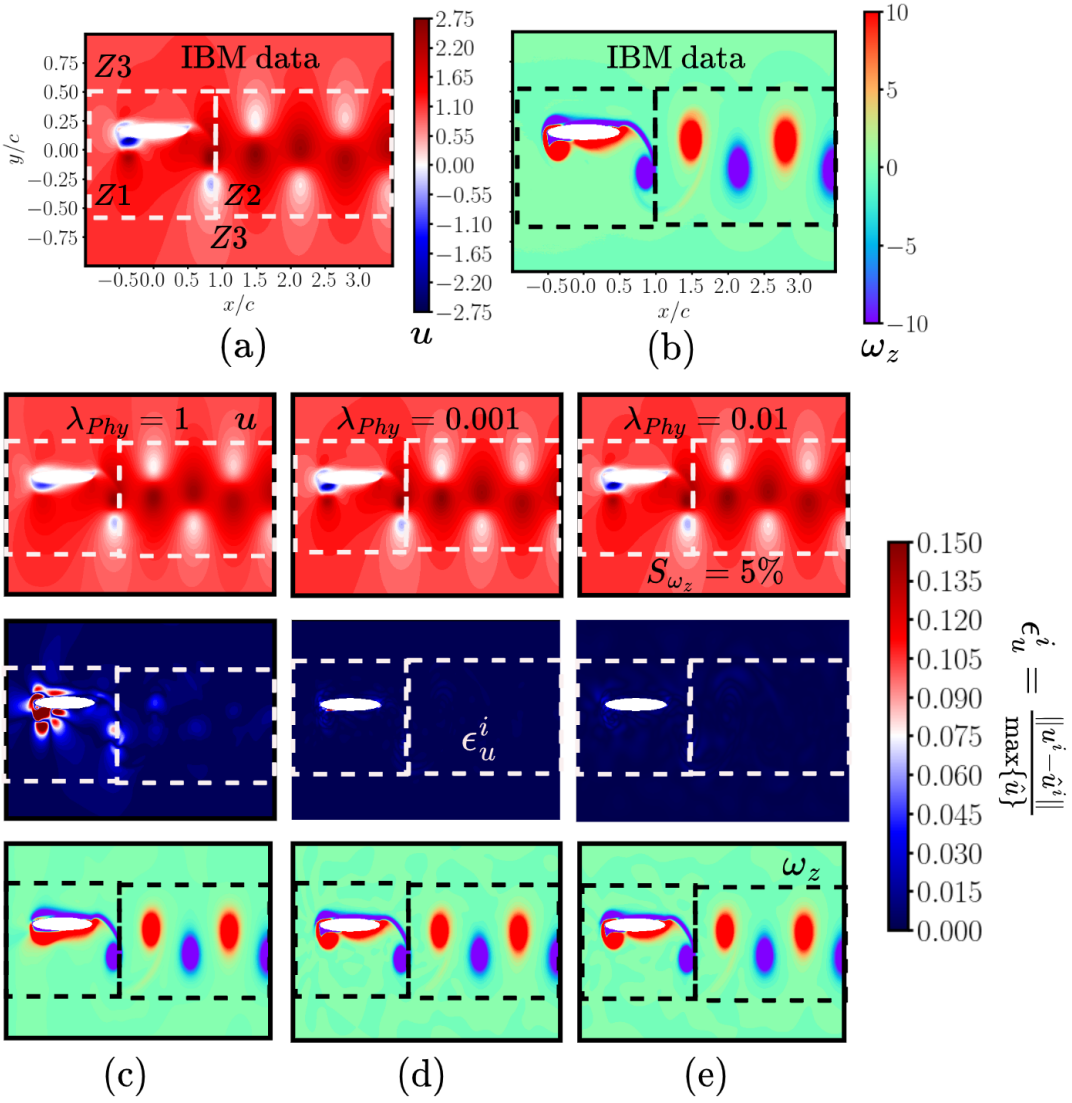


Figure B.3: Comparison of (a-b) true and (c-e) MB-PINN predicted velocity x-component, corresponding point-wise maximum value normalized absolute errors and vorticity contours at $t/T = 1.0$.

Table B.1: Training and testing data set resolution within the truncated domain considered in this study. Ref-IBM and Ref-ALE are the high resolution testing data-sets generated using a IBM and an ALE solver, respectively. Further details about the 'Ref-*' data-sets can be found in Section 2.6.1 and also see Sundar *et al.* (2024).

Data sets	N_x	N_y	N_t	$\Delta t/T$	$(Nx \times Ny \times N_t)$
CI	270	120	41	0.05	**1.3284e06**
CI-S5	-	-	-	0.05	**2.648e05**
Ref-IBM	651	500	81	0.025	2.6365e07
Ref-ALE	-	-	81	0.025	4.05e06

Table B.2: Training data-sets and associated proportion p_{Zi} of data points from each spatial zone Zi with $i = 1, 2,$ and 3 for a mini-batch sample at$1e06$ iterations.

Data set	N_{Bulk}	p_{Z1}	p_{Z2}	p_{Z3}
CI	1.2915e06	2.175e-01	2.857e-01	4.967e-01
CI-S5	2.6481e05	2.936e-01	5.841e-01	1.223e-01

Table B.3: Accuracy of MB-PINN for different relaxation coefficients evaluated on Ref-IBM and Ref-ALE datasets. Best performing models are highlighted in bold-faced fonts.

Test case	λ_{fluid}	S_{ω_z} (in %)	aMAE	arRMSE (in %)
Case 1	1	100	1.37e-01	22.96
Case 2	**0.001**	100	**2.07e-02**	**3.22**
Case 3	**0.01**	**5**	**2.25e-02**	**3.22**

$\nabla_\theta L_{Phy}$ distributions across all zones are at least one order higher than that of $\nabla_\theta L_{Bulk}$ (see Figs. B.5(a)and B.5(d) and Figs. B.6(a)and B.6(d), respectively). Although the gradients have a higher μ and σ (subscripts dropped for simplicity) in $Z1$ (see table B.4) the gradients for the overall domain Ω_f still receive significant contributions from $Z2$ and $Z3$ (see figure B.5(a) and B.5(d)) since $p_{Z1} < p_{Z2} < p_{Z3}$.

B.3.2 Case 2: With physics loss relaxation

With global physics loss relaxation, the MB-PINN model trained on the CI dataset was reported to be optimal for $\lambda_{Phy} = 0.001$. It is seen that the gradient magnitudes are much lower than that of the baseline case indicating some sort of vanishing gradient problem

(see Figs. B.4(b)and B.4(e)). Even though the gradient magnitudes are very low, the loss component gradients are relatively well balanced across all the zones unlike in case 1 (see Figs. B.5(b)and B.5(e), and Figs. B.6(b)and B.6(e), respectively). Here, as seen in the table B.4, $\nabla_\theta \mathcal{L}_{Bulk}$ is dominated by contributions from $Z2$, while $\nabla_\theta \mathcal{L}_{Phy}$ receives most of its contributions from $Z1$. In spite of the good accuracy, it is seen that the gradients almost vanish in the last layer as opposed to the first layer.

B.3.3 Case 3: With global physics loss relaxation and vorticity cutoff based under sampling

Combining the physics loss relaxation with a physics based vorticity cutoff sampling was identified as a useful strategy in Chapter 2 (also see Sundar *et al.* (2024)) to reduce the data requirement while still obtaining similar accuracy as in case 2 (see table B.3). It is observed that in case 3 the magnitudes of $\nabla_\theta \mathcal{L}_{Bulk}$ and $\nabla_\theta \mathcal{L}_{Phy}$ are higher and they don't vanish unlike in case 2 (see Figs. B.4(c) and B.4(f)). In a way, since vorticity cutoff based under sampling retains more data points only in those regions with strong gradients, it is possible that this prevents the gradients from vanishing. Unlike Case 1 and Case 2, due to selective under sampling, $p_{Z3} < p_{Z1} < p_{Z2}$ which in turn affects the gradient back propagation. As a result, in table B.4 it is seen that $\nabla_\theta \mathcal{L}_{Bulk}$ and $\nabla_\theta \mathcal{L}_{Phy}$ are both dominated by contributions from $Z1$ followed by that from $Z2$. The first and last layer gradient distributions as well indicate that across all zones, the gradients are balanced with $Z1$ driving the training as seen by the gradient distributions on Ω_f following $Z1$.

B.4 CONCLUSIONS

In this Chapter, to understand the training of MB-PINNs for flow past a plunging foil, a novel zonal splitting methodology and gradient statistics-based metrics were proposed to analyze spatial imbalances in loss component contributions. Three zones consisting of the moving body, the wake and the outer far-field were considered. Three test cases from earlier Chapter 2 were considered for analysis with and without physics

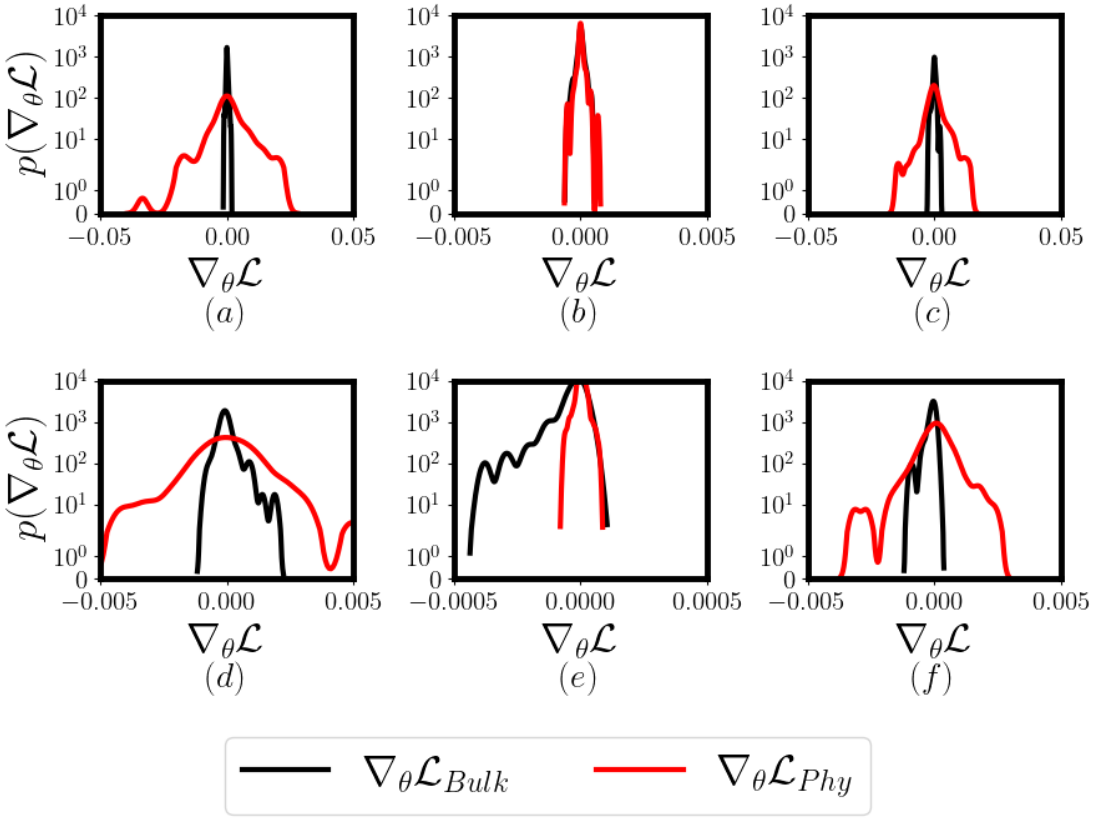


Figure B.4: First layer (top row) and last layer (bottom row) $\nabla_\theta \mathcal{L}_{Bulk}$ and $\nabla_\theta \mathcal{L}_{Phy}$ distributions obtained after $1e06$ training iterations for (a,d) Case 1: $\lambda_{Phy} = 1$, (b,e) Case 2: $\lambda_{Phy} = 0.001$ and $S_{\omega_z} = 100\%$ and (c,f) Case 3: $\lambda_{Phy} = 0.01$ and $S_{\omega_z} = 5\%$.

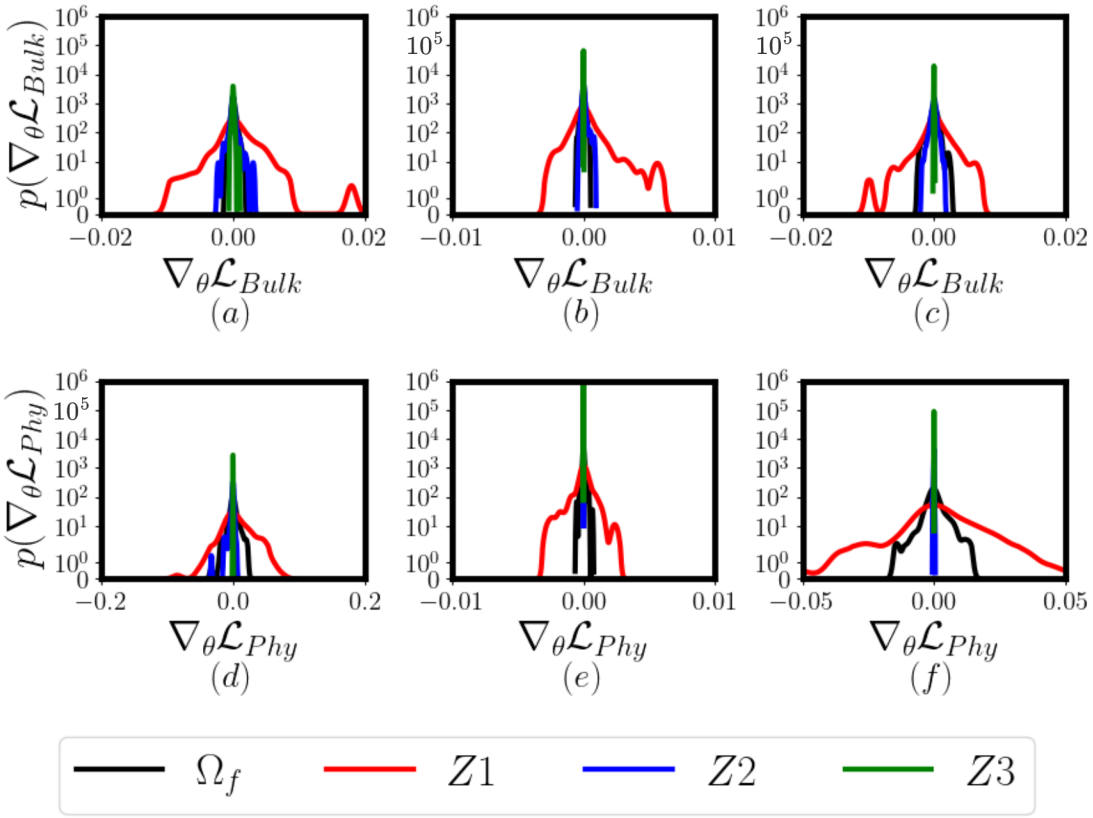


Figure B.5: First layer $\nabla_\theta \mathcal{L}_{Bulk}$ (top row) and $\nabla_\theta \mathcal{L}_{Phy}$ (bottom row) distributions obtained after $1e06$ training iterations for (a,d) Case 1: $\lambda_{Phy} = 1$, (b,e) Case 2: $\lambda_{Phy} = 0.001$ and $S_{\omega_z} = 100\%$ and (c,f) Case 3: $\lambda_{Phy} = 0.01$ and $S_{\omega_z} = 5\%$.

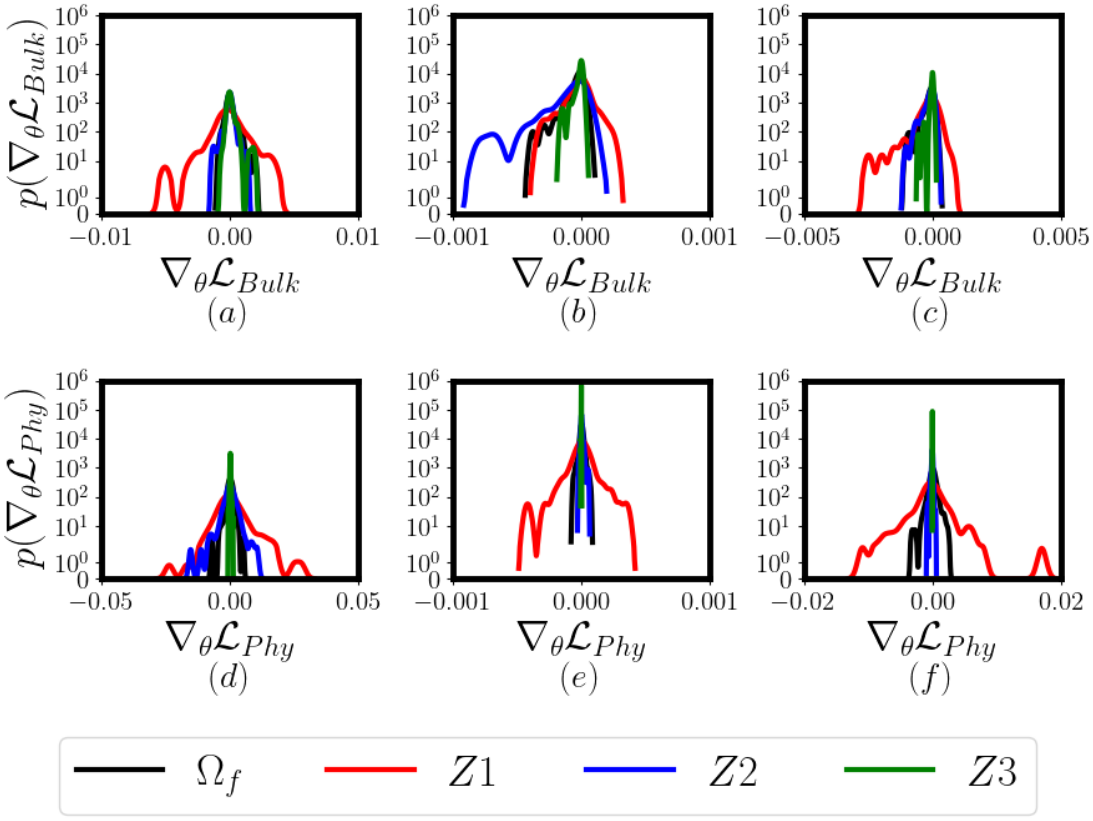


Figure B.6: Comparison of last layer $\nabla_\theta \mathcal{L}_{Bulk}$ (top row) and $\nabla_\theta \mathcal{L}_{Phy}$ (bottom row) distributions obtained after $1e06$ training iterations for (a,d) Case 1: $\lambda_{Phy} = 1$, (b,e) Case 2: $\lambda_{Phy} = 0.001$ and $S_{\omega_z} = 100\%$ and (c,f) Case 3: $\lambda_{Phy} = 0.01$ and $S_{\omega_z} = 5\%$.

Table B.4: Zonal relative gradient statistics across test cases. Dominant zones are highlighted in bold.

Zone	$\mu_{Bulk}^{Z\#}$	$\sigma_{Bulk}^{Z\#}$	$\mu_{Phy}^{Z\#}$	$\sigma_{Phy}^{Z\#}$
Case 1: $\lambda_{Phy} = 1$				
Z1	**7.932e-01**	**7.466e-01**	**1.256**	**2.153**
Z2	2.179e-01	2.669e-01	1.292e-01	1.368e-01
Z3	2.292e-01	2.459e-01	6.503e-02	5.801e-02
Case 2: $\lambda_{Phy} = 0.001$				
Z1	3.207e-01	4.769e-01	**5.565e-01**	**5.608e-01**
Z2	**7.552e-01**	**8.477e-01**	6.906e-02	5.608e-02
Z3	1.297e-01	1.676e-01	9.610e-3	5.823e-03
Case 3: $\lambda_{Phy} = 0.01$, $S_{\omega_z} = 5\%$				
Z1	**7.054e-01**	**1.194**	**4.929**	**7.555**
Z2	4.872e-01	4.392e-01	1.359e-01	1.073e-01
Z3	7.959e-03	5.917e-03	1.432e-03	7.506e-04

loss relaxation and vorticity cutoff based under-sampling. The analysis confirmed the existence of zonal imbalances and also determined which zone dictated training. In the test case with the relaxation of the physics loss alone, the wake zone dominated the gradient updates coming from the loss of data. Whereas, the moving body zone dominated the updates from physics loss. However, for the test case with both physics loss relaxation and physics-based undersampling, the moving body zone dominated the gradient updates for both the loss components. It was also observed that a vorticity cutoff based under-sampling alleviates the problem of vanishing gradients as opposed to the case of no under-sampling, further validating the importance of selective physics-based sampling. It is envisioned that quantifying the imbalanced contribution from spatial or even more generally input subdomains would enable the design of effective loss component weighting strategies. This approach can be extended effectively for other problems as well to evaluate imbalances in contributions from the input sub-domains of interest. Moreover, being agnostic to the model architecture, this approach can be applied to physics-based or even purely data-driven methods to determine which input subdomain plays a relatively active role in training.

REFERENCES

1. **Abadi, M.**, Tensorflow: learning functions at scale. *In Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, ICFP 2016. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450342193. URL https://doi.org/10.1145/2951913.2976746.

2. **Ahmed, S. E.**, **S. Pawar**, **O. San**, **A. Rasheed**, **T. Iliescu**, and **B. R. Noack** (2021). On closures for reduced order models—a spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, **33**(9). URL https://doi.org/10.1063/5.0061577.

3. **Al Safwan, A.**, **C. Song**, and **U. B. Waheed**, Is it time to swish? Comparing activation functions in solving the Helmholtz equation using PINNS. *In 82nd EAGE Annual Conference & Exhibition*, volume 2021. European Association of Geoscientists & Engineers, 2021. URL https://doi.org/10.3997/2214-4609.202113254.

4. **Aliakbari, M.**, **M. Mahmoudi**, **P. Vadasz**, and **A. Arzani** (2022). Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks. *International Journal of Heat and Fluid Flow*, **96**, 109002. URL https://doi.org/10.1016/j.ijheatfluidflow.2022.109002.

5. **Alkin, B.**, **A. Fürst**, **S. Schmid**, **L. Gruber**, **M. Holzleitner**, and **J. Brandstetter** (2025). Universal physics transformers: A framework for efficiently scaling neural operators. URL https://arxiv.org/abs/2402.12365.

6. **Álvarez-Farré, X.**, **A. Gorobets**, and **F. X. Trias** (2021). A hierarchical parallel implementation for heterogeneous computing. application to algebra-based CFD simulations on hybrid supercomputers. *Computers & Fluids*, **214**, 104768. URL https://doi.org/10.1016/j.compfluid.2020.104768.

7. **Amdahl, G. M.**, Validity of the single processor approach to achieving large scale computing capabilities. *In Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring). Association for Computing Machinery, New York, NY, USA, 1967. ISBN 9781450378956. URL https://doi.org/10.1145/1465482.1465560.

8. **Anagnostopoulos, S. J.**, **J. D. Toscano**, **N. Stergiopulos**, and **G. E. Karniadakis** (2024). Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **421**, 116805. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782524000616.

9. **Angra, S.** and **S. Ahuja**, Machine learning and its applications: A review. *In 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*. 2017. URL 10.1109/ICBDACI.2017.8070809.

10. **Ansari, S.**, **R. Żbikowski**, and **K. Knowles** (2006). Aerodynamic modelling of insect-like flapping flight for micro air vehicles. *Progress in Aerospace Sciences*, **42**(2), 129–172. ISSN 0376-0421. URL https://doi.org/10.1016/j.paerosci.2006.07.001.

11. **Avrutskiy, V. I.** (2020). Neural networks catching up with finite differences in solving partial differential equations in higher dimensions. *Neural Computing and Applications*, 1–16. URL https://doi.org/10.1007/s00521-020-04743-8.

12. **Badrinath, S.**, **C. Bose**, and **S. Sarkar** (2017). Identifying the route to chaos in the flow past a flapping airfoil. *European Journal of Mechanics - B/Fluids*, **66**, 38–59. ISSN 0997-7546. URL https://www.sciencedirect.com/science/article/pii/S0997754616300656.

13. **Bai, X.-D.** and **W. Zhang** (2022). Machine learning for vortex induced vibration in turbulent flow. *Computers & Fluids*, **235**, 105266. ISSN 0045-7930. URL https://www.sciencedirect.com/science/article/pii/S0045793021003649.

14. **Balajewicz, M.** and **C. Farhat** (2014). Reduction of nonlinear embedded boundary models for problems with evolving interfaces. *Journal of Computational Physics*, **274**, 489–504. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999114004458.

15. **Baldi, P.** and **K. Hornik** (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, **2**(1), 53–58. ISSN 0893-6080. URL https://www.sciencedirect.com/science/article/pii/0893608089900142.

16. **Bao, A.** and **E. Gildin**, Data-driven model reduction based on sparsity-promoting methods for multiphase flow in porous media. volume SPE Latin America and Caribbean Petroleum Engineering Conference of *SPE Latin America and Caribbean Petroleum Engineering Conference*. 2017. URL https://doi.org/10.2118/185514-MS.

17. **Baydin, A. G.**, **B. A. Pearlmutter**, **A. A. Radul**, and **J. M. Siskind** (2017). Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, **18**(1), 5595–5637. ISSN 1532-4435. URL https://dl.acm.org/doi/abs/10.5555/3122009.3242010.

18. **Benner, P.**, **S. Gugercin**, and **K. Willcox** (2015). A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, **57**(4), 483–531. URL https://doi.org/10.1137/130932715.

19. **Bhushan, S.**, **G. W. Burgreen**, **J. L. Bowman**, **I. D. Dettwiller**, and **W. Brewer**, Predictions of steady and unsteady flows using machine-learned surrogate models. *In 2020 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)*. 2020. URL 10.1109/MLHPCAI4S51975.2020.

00016.

20. **Bischof, R.** and **M. A. Kraus** (2025). Multi-objective loss balancing for physics-informed deep learning. *Computer Methods in Applied Mechanics and Engineering*, **439**, 117914. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782525001860.

21. **Bishop, C. M.**, *Pattern recognition and machine learning*. springer, 2006. URL https://link.springer.com/9780387310732.

22. **Bisplinghoff, R. L.**, **H. Ashley**, and **R. L. Halfman**, *Aeroelasticity*. Courier Corporation, 1996. URL 10.1017/S0001924000122213.

23. **Bodnar, C.**, **W. P. Bruinsma**, **A. Lucic**, **M. Stanley**, **A. Vaughan**, **J. Brandstetter**, **P. Garvan**, **M. Riechert**, **J. A. Weyn**, **H. Dong**, **J. K. Gupta**, **K. Thambiratnam**, **A. T. Archibald**, **C.-C. Wu**, **E. Heider**, **M. Welling**, **R. E. Turner**, and **P. Perdikaris** (2024). A foundation model for the earth system. URL https://arxiv.org/abs/2405.13063.

24. **Bose, C.**, **S. Gupta**, and **S. Sarkar** (2019). Transition to chaos in the flow-induced vibration of a pitching–plunging airfoil at low reynolds numbers: Ruelle–Takens–Newhouse scenario. *International Journal of Non-Linear Mechanics*, **109**, 189–203. URL https://doi.org/10.1016/j.ijnonlinmec.2018.11.012.

25. **Bose, C.**, **V. Reddy**, **S. Gupta**, and **S. Sarkar** (2017). Transient and stable chaos in dipteran flight inspired flapping motion. *Journal of Computational and Nonlinear Dynamics*, **13**(2), 021014. ISSN 1555-1415. URL https://doi.org/10.1115/1.4038447.

26. **Bose, C.** and **S. Sarkar**, Flow periodicity analysis past a flapping airfoil using proper orthogonal decomposition. *In 47th AIAA Fluid Dynamics Conference*. URL https://arc.aiaa.org/doi/abs/10.2514/6.2017-3647.

27. **Bose, C.** and **S. Sarkar** (2018). Investigating chaotic wake dynamics past a flapping airfoil and the role of vortex interactions behind the chaotic transition. *Physics of Fluids*, **30**(4), 047101. ISSN 1070-6631. URL https://doi.org/10.1063/1.5019442.

28. **Bottou, L.**, *On-line learning and stochastic approximations*. Cambridge University Press, USA, 1999. ISBN 0521652634, 9–42.

29. **Boutet, J.** and **G. Dimitriadis** (2018). Unsteady lifting line theory using the Wagner function for the aerodynamic and aeroelastic modeling of 3D wings. *Aerospace*, **5**(3). ISSN 2226-4310. URL https://www.mdpi.com/2226-4310/5/3/92.

30. **Bradbury, J.**, **R. Frostig**, **P. Hawkins**, **M. J. Johnson**, **C. Leary**, **D. Maclaurin**, **G. Necula**, **A. Paszke**, **J. VanderPlas**, **S. Wanderman-Milne**, and **Q. Zhang** (2018). JAX: composable transformations of Python+NumPy programs. URL http://github.com/google/jax.

31. **Brown, T. B.**, **B. Mann**, **N. Ryder**, **M. Subbiah**, **J. Kaplan**, **P. Dhariwal**, **A. Neelakantan**, **P. Shyam**, **G. Sastry**, **A. Askell**, **S. Agarwal**, **A. Herbert-Voss**, **G. Krueger**, **T. Henighan**, **R. Child**, **A. Ramesh**, **D. M. Ziegler**, **J. Wu**, **C. Winter**, **C. Hesse**, **M. Chen**, **E. Sigler**, **M. Litwin**, **S. Gray**, **B. Chess**, **J. Clark**, **C. Berner**, **S. McCandlish**, **A. Radford**, **I. Sutskever**, and **D. Amodei**, Language models are few-shot learners. *In Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., Red Hook, NY, USA, 2020. ISBN 9781713829546. URL https://dl.acm.org/doi/abs/10.5555/3495724.3495883.

32. **Brunton, B. W.**, **L. A. Johnson**, **J. G. Ojemann**, and **J. N. Kutz** (2016). Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of Neuroscience Methods*, **258**, 1–15. ISSN 0165-0270. URL https://www.sciencedirect.com/science/article/pii/S0165027015003829.

33. **Brunton, S. L.**, **J. Nathan Kutz**, **K. Manohar**, **A. Y. Aravkin**, **K. Morgansen**, **J. Klemisch**, **N. Goebel**, **J. Buttrick**, **J. Poskin**, **A. W. Blom-Schieber**, **T. Hogan**, and **D. McDonald** (2021). Data-driven aerospace engineering: Reframing the industry with machine learning. *AIAA Journal*, **59**(8), 2820–2847. URL https://doi.org/10.2514/1.J060131.

34. **Brunton, S. L.**, **B. R. Noack**, and **P. Koumoutsakos** (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, **52**(Volume 52, 2020), 477–508. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-010719-060214.

35. **Buhendwa, A. B.**, **S. Adami**, and **N. A. Adams** (2021). Inferring incompressible two-phase flow fields from the interface motion using physics-informed neural networks. *Machine Learning with Applications*, **4**, 100029. ISSN 2666-8270. URL https://www.sciencedirect.com/science/article/pii/S2666827021000104.

36. **Cai, S.**, **Z. Mao**, **Z. Wang**, **M. Yin**, and **G. E. Karniadakis** (2021*a*). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 1–12. URL https://doi.org/10.1007/s10409-021-01148-1.

37. **Cai, S.**, **Z. Wang**, **S. Wang**, **P. Perdikaris**, and **G. E. Karniadakis** (2021*b*). Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, **143**(6), 060801. ISSN 0022-1481. URL https://doi.org/10.1115/1.4050542.

38. **Calicchia, M. A.**, **R. Mittal**, **J.-H. Seo**, and **R. Ni** (2023). Reconstructing the pressure field around swimming fish using a physics-informed neural network. *Journal of Experimental Biology*, **226**(8), jeb244983. URL https://doi.org/10.1242/jeb.244983.

39. **Chandrasekaran, S.** and **G. Juckeland**, *OpenACC for Programmers: Concepts and Strategies*. Addison-Wesley Professional, 2017. URL https://dl.acm.org/doi/

book/10.5555/3175812.

40. **Chatterjee, R.**, **C. L. Shah**, **S. Gupta**, and **S. Sarkar** (2024). Energy harvesting from wake-induced vibration of flexible flapper behind a bluff body. *Physics of Fluids*, **36**(11), 117130. ISSN 1070-6631. URL https://doi.org/10.1063/5.0234607.

41. **Cheng, S.**, **C. Quilodrán-Casas**, **S. Ouala**, **A. Farchi**, **C. Liu**, **P. Tandeo**, **R. Fablet**, **D. Lucor**, **B. Iooss**, **J. Brajard**, **D. Xiao**, **T. Janjic**, **W. Ding**, **Y. Guo**, **A. Carrassi**, **M. Bocquet**, and **R. Arcucci** (2023). Machine learning with data assimilation and uncertainty quantification for dynamical systems: A review. *IEEE/CAA Journal of Automatica Sinica*, **10**(6), 1361–1387. URL https://doi.org/10.1109/JAS.2023.123537.

42. **Chin, D. D.** and **D. Lentink** (2016). Flapping wing aerodynamics: from insects to vertebrates. *Journal of Experimental Biology*, **219**(7), 920–932. ISSN 0022-0949. URL https://doi.org/10.1242/jeb.042317.

43. **Choi, H.** and **P. Moin** (1994). Effects of the computational time step on numerical solutions of turbulent flow. *Journal of Computational Physics*, **113**(1), 1–4. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999184711120.

44. **Chuang, P.-Y.** and **L. A. Barba** (2022). Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration. URL https://doi.org/10.25080/majora-212e5952-005.

45. **Cuomo, S.**, **V. S. Di Cola**, **F. Giampaolo**, **G. Rozza**, **M. Raissi**, and **F. Piccialli** (2022). Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing*, **92**(3), 88. URL https://doi.org/10.1007/s10915-022-01939-z.

46. **Cyr, E. C.**, **M. A. Gulian**, **R. G. Patel**, **M. Perego**, and **N. A. Trask**, Robust training and initialization of deep neural networks: An adaptive basis viewpoint. *In* **J. Lu** and **R. Ward** (eds.), *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*. PMLR, 2020. URL https://proceedings.mlr.press/v107/cyr20a.html.

47. **Dabiri, J. O.**, **S. Bose**, **B. J. Gemmell**, **S. P. Colin**, and **J. H. Costello** (2014). An algorithm to estimate unsteady and quasi-steady pressure fields from velocity field measurements. *Journal of Experimental Biology*, **217**(3), 331–336. ISSN 0022-0949. URL https://doi.org/10.1242/jeb.092767.

48. **Daniel, T. L.** (2015). Unsteady aspects of aquatic locomotion. *American Zoologist*, **24**(1), 121–134. ISSN 0003-1569. URL https://doi.org/10.1093/icb/24.1.121.

49. **Deng, J.**, **L. Sun**, **L. Teng**, **D. Pan**, and **X. Shao** (2016). The correlation between wake

transition and propulsive efficiency of a flapping foil: A numerical study. *Physics of Fluids*, **28**(9), 094101. ISSN 1070-6631. URL https://doi.org/10.1063/1.4961566.

50. **Derakhshandeh, J.** and **M. M. Alam** (2019). A review of bluff body wakes. *Ocean Engineering*, **182**, 475–488. ISSN 0029-8018. URL https://www.sciencedirect.com/science/article/pii/S0029801818307418.

51. **Dowell, E. H.** and **K. C. Hall** (2001). Modeling of fluid-structure interaction. *Annual Review of Fluid Mechanics*, **33**(Volume 33, 2001), 445–490. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev.fluid.33.1.445.

52. **Draper, N. R.** and **H. Smith**, *Applied regression analysis*, volume 326. John Wiley & Sons, 1998. URL https://doi.org/10.1007/b98890.

53. **Dubey, S. R.**, **S. K. Singh**, and **B. B. Chaudhuri** (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, **503**, 92–108. URL https://doi.org/10.1016/j.neucom.2022.06.111.

54. **Duhme, C.**, **F. Eilers**, and **X. Jiang** (2026). Exploring sparsity and smoothness of arbitrary ℓ_p norms in adversarial attacks. URL https://arxiv.org/abs/2602.06578.

55. **Dwivedi, V.**, **N. Parashar**, and **B. Srinivasan** (2021). Distributed learning machines for solving forward and inverse problems in partial differential equations. *Neurocomputing*, **420**, 299–316. URL https://doi.org/10.1016/j.neucom.2020.09.006.

56. **Eivazi, H.**, **H. Veisi**, **M. H. Naderi**, and **V. Esfahanian** (2020). Deep neural networks for nonlinear model order reduction of unsteady flows. *Physics of Fluids*, **32**(10), 105104. ISSN 1070-6631. URL https://doi.org/10.1063/5.0020526.

57. **El Garroussi, S.**, **S. Ricci**, **M. De Lozzo**, **N. Goutal**, and **D. Lucor** (2022). Tackling random fields non-linearities with unsupervised clustering of polynomial chaos expansion in latent space: application to global sensitivity analysis of river flooding. *Stochastic Environmental Research and Risk Assessment*, **36**(3), 693–718. URL https://doi.org/10.1007/s00477-021-02060-7.

58. **Everson, R.** and **L. Sirovich** (1995). Karhunen–loeve procedure for gappy data. *Journal of the Optical Society of America A*, **12**(8), 1657–1664. URL https://doi.org/10.1364/JOSAA.12.001657.

59. **Facchinetti, M. L.**, **E. De Langre**, and **F. Biolley** (2004). Coupling of structure and wake oscillators in vortex-induced vibrations. *Journal of Fluids and structures*, **19**(2), 123–140. URL https://doi.org/10.1016/j.jfluidstructs.2003.12.004.

60. **Fadlun, E. A.**, **R. Verzicco**, **P. Orlandi**, and **J. Mohd-Yusof** (2000). Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of computational physics*, **161**(1), 35–60. URL https://doi.org/10.1006/

jcph.2000.6484.

61. **Farber, R.**, *Parallel programming with OpenACC*. Newnes, 2016. ISBN 978-0-12-410397-9. URL https://www.sciencedirect.com/book/edited-volume/9780124103979/parallel-programming-with-openacc.

62. **Farhat, C.**, **P. Geuzaine**, and **C. Grandmont** (2001). The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids. *Journal of Computational Physics*, **174**(2), 669–694. URL https://doi.org/10.1006/jcph.2001.6932.

63. **Fu, J.**, **D. Xiao**, **R. Fu**, **C. Li**, **C. Zhu**, **R. Arcucci**, and **I. M. Navon** (2023). Physics-data combined machine learning for parametric reduced-order modelling of nonlinear dynamical systems in small-data regimes. *Computer Methods in Applied Mechanics and Engineering*, **404**, 115771. URL https://doi.org/10.1016/j.cma.2022.115771.

64. **Fukami, K.**, **K. Fukagata**, and **K. Taira** (2019). Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, **870**, 106–120.

65. **Fukami, K.**, **K. Fukagata**, and **K. Taira** (2023). Super-resolution analysis via machine learning: a survey for fluid flows. *Theoretical and Computational Fluid Dynamics*, 1–24. URL https://doi.org/10.1017/jfm.2019.238.

66. **Fung, Y. C.**, *An introduction to the theory of aeroelasticity*. Courier Dover Publications, 2008. ISBN 0486678717.

67. **Gao, H.**, **J.-X. Wang**, and **M. J. Zahr** (2020). Non-intrusive model reduction of large-scale, nonlinear dynamical systems using deep learning. *Physica D: Nonlinear Phenomena*, **412**, 132614. URL https://doi.org/10.1016/j.physd.2020.132614.

68. **Gao, W.**, **X. Zhang**, **L. Yang**, and **H. Liu**, An improved sobel edge detection. *In 2010 3rd International conference on computer science and information technology*, volume 5. IEEE, 2010. URL https://doi.org/10.1109/ICCSIT.2010.5563693.

69. **Gao, Z.**, **L. Yan**, and **T. Zhou** (2023). Failure-informed adaptive sampling for PINNs. *SIAM Journal on Scientific Computing*, **45**(4), A1971–A1994. URL https://doi.org/10.1137/22M1527763.

70. **Garrick, I. E.** (1937). Propulsion of a flapping and oscillating airfoil. Technical report. URL https://ntrs.nasa.gov/api/citations/19930091642/downloads/19930091642.pdf.

71. **Geuzaine, C.** and **J.-F. Remacle** (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, **79**(11), 1309–1331. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579.

72. **Ghate, A.**, **A. Towne**, and **S. Lele** (2020). Broadband reconstruction of inhomogeneous turbulence using spectral proper orthogonal decomposition and gabor modes. *Journal of Fluid Mechanics*, **888**. URL https://doi.org/10.1017/jfm.2020.78.

73. **Glorot, X.** and **Y. Bengio**, Understanding the difficulty of training deep feedforward neural networks. *In* **Y. W. Teh** and **M. Titterington** (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*. PMLR, Chia Laguna Resort, Sardinia, Italy, 2010. URL https://proceedings.mlr.press/v9/glorot10a.html.

74. **Goldstein, D.**, **R. Handler**, and **L. Sirovich** (1993). Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*, **105**(2), 354–366. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999183710818.

75. **Goodfellow, I.**, **Y. Bengio**, and **A. Courville**, *Deep learning*. MIT press, 2016. URL http://www.deeplearningbook.org.

76. **Gopakumar, V.**, **S. Pamela**, and **D. Samaddar** (2023). Loss landscape engineering via data regulation on PINNs. *Machine Learning with Applications*, **12**, 100464. ISSN 2666-8270. URL https://doi.org/10.1016/j.mlwa.2023.100464.

77. **Goswami, S.**, **A. Bora**, **Y. Yu**, and **G. E. Karniadakis**, Physics-informed deep neural operator networks. *In Machine Learning in Modeling and Simulation: Methods and Applications*. Springer International Publishing, Cham, 2023. ISBN 978-3-031-36644-4, 219–254. URL https://doi.org/10.1007/978-3-031-36644-4_6.

78. **Goza, A.** and **T. Colonius** (2018). Modal decomposition of fluid–structure interaction with application to flag flapping. *Journal of Fluids and Structures*, **81**, 728–737. URL https://doi.org/10.1016/j.jfluidstructs.2018.06.014.

79. **Gunes, H.**, **S. Sirisup**, and **G. E. Karniadakis** (2006). Gappy data: To Krig or not to Krig? *Journal of Computational Physics*, **212**(1), 358–382. URL https://doi.org/10.1016/j.jcp.2005.06.023.

80. **Gursul, I.** and **D. Cleaver** (2019). Plunging oscillations of airfoils and wings: Progress, opportunities, and challenges. *AIAA Journal*, **57**(9), 3648–3665. URL https://doi.org/10.2514/1.J056655.

81. **Haykin, S.**, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994. URL https://dl.acm.org/doi/book/10.5555/1213811.

82. **Haykin, S. S.** (2009). Neural networks and learning machines.

83. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun**, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15. IEEE Computer

Society, USA, 2015. ISBN 9781467383912. URL https://doi.org/10.1109/ICCV.2015.123.

84. **Heil, M.** and **A. L. Hazel** (2011). Fluid-structure interaction in internal physiological flows. *Annual Review of Fluid Mechanics*, **43**(Volume 43, 2011), 141–162. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-122109-160703.

85. **Hellström, L. H.** and **A. J. Smits** (2017). Structure identification in pipe flow using proper orthogonal decomposition. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **375**(2089), 20160086.

86. **Hess, J.** and **A. Smith** (1967). Calculation of potential flow about arbitrary bodies. *Progress in Aerospace Sciences*, **8**, 1–138. ISSN 0376-0421. URL https://www.sciencedirect.com/science/article/pii/0376042167900036.

87. **Heydari, A. A.**, **C. A. Thompson**, and **A. Mehmood** (2019). SoftAdapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions. *arXiv preprint arXiv:1912.12355*. URL https://doi.org/10.48550/arXiv.1912.12355.

88. **Hijazi, S.**, **G. Stabile**, **A. Mola**, and **G. Rozza** (2020). Data-driven pod-galerkin reduced order model for turbulent flows. *Journal of Computational Physics*, **416**, 109513. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999120302874.

89. **Hilberg, D.**, **W. Lazik**, and **H. Fiedler** (1994). The application of classical pod and snapshot pod in a turbulent shear layer with periodic structures. *Applied scientific research*, **53**(3-4), 283–290. URL https://doi.org/10.1007/BF00849105.

90. **Hinton, G. E.**, **N. Srivastava**, **A. Krizhevsky**, **I. Sutskever**, and **R. R. Salakhutdinov** (2012). Improving neural networks by preventing co-adaptation of feature detectors. URL https://arxiv.org/abs/1207.0580.

91. **Hirt, C.**, **A. Amsden**, and **J. Cook** (1974). An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, **14**(3), 227–253. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/0021999174900515.

92. **Hoefler, T.**, **D. Alistarh**, **T. Ben-Nun**, **N. Dryden**, and **A. Peste** (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, **22**(241), 1–124. URL http://jmlr.org/papers/v22/21-0366.html.

93. **Hornik, K.**, **M. Stinchcombe**, and **H. White** (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**(5), 359–366. ISSN 0893-6080. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

94. **Hoshino, T.**, **N. Maruyama**, **S. Matsuoka**, and **R. Takaki**, CUDA vs OpenACC: Performance case studies with kernel benchmarks and a memory-bound CFD application. *In 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. IEEE, 2013. URL 10.1109/CCGrid.2013.12.

95. **Huang, C.**, **H. Ge**, **Y. Zhang**, and **X. Su** (2026). Deep plug-and-play priors with structural properties for tensor compressive sensing. *Neurocomputing*, **683**, 133509. ISSN 0925-2312. URL https://www.sciencedirect.com/science/article/pii/S0925231226009069.

96. **Huang, W.-X.** and **H. J. Sung** (2007). Improvement of mass source/sink for an immersed boundary method. *International journal for numerical methods in fluids*, **53**(11), 1659–1671. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.1458.

97. **Huang, Y.**, **Z. Zhang**, and **X. Zhang** (2022). A Direct-Forcing Immersed Boundary Method for incompressible flows based on Physics-Informed Neural Network. *Fluids*, **7**(2), 56. URL https://doi.org/10.3390/fluids7020056.

98. **Ismail Fawaz, H.**, **B. Lucas**, **G. Forestier**, **C. Pelletier**, **D. F. Schmidt**, **J. Weber**, **G. I. Webb**, **L. Idoumghar**, **P.-A. Muller**, and **F. Petitjean** (2020). InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, **34**(6), 1936–1962. URL https://doi.org/10.1007/s10618-020-00710-y.

99. **Jagtap, A. D.** and **G. E. Karniadakis**, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. volume 28. Global Science Press, 2020. ISSN ISSN 1815-2406. URL https://www.osti.gov/biblio/2282003.

100. **Jagtap, A. D.** and **G. E. Karniadakis** (2023). How important are activation functions in regression and classification? a survey, performance comparison, and future directions. *Journal of Machine Learning for Modeling and Computing*, **4**(1). URL https://doi.org/10.1615/JMachLearnModelComput.2023047367.

101. **Jagtap, A. D.**, **K. Kawaguchi**, and **G. Em Karniadakis** (2020*a*). Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **476**(2239), 20200334. ISSN 1364-5021. URL https://doi.org/10.1098/rspa.2020.0334.

102. **Jagtap, A. D.**, **K. Kawaguchi**, and **G. E. Karniadakis** (2020*b*). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, **404**, 109136. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999119308411.

103. **Jagtap, A. D.**, **E. Kharazmi**, and **G. E. Karniadakis** (2020*c*). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to

forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, **365**, 113028. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782520302127.

104. **Jagtap, A. D.**, **Y. Shin**, **K. Kawaguchi**, and **G. E. Karniadakis** (2022). Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions. *Neurocomputing*, **468**, 165–180. ISSN 0925-2312. URL https://www.sciencedirect.com/science/article/pii/S0925231221015162.

105. **Jasak, H.**, **A. Jemcov**, **Z. Tukovic**, *et al.*, OpenFOAM: A c++ library for complex physics simulations. *In International workshop on coupled methods in numerical dynamics*, volume 1000. IUC Dubrovnik Croatia, 2007. URL https://www.croris.hr/crosbi/publikacija/prilog-skup/538019.

106. **Jin, X.**, **S. Cai**, **H. Li**, and **G. E. Karniadakis** (2021). NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, **426**, 109951. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999120307257.

107. **Jones, R. T.**, *The unsteady lift of a finite wing*. 682. National Advisory Committee for Aeronautics, 1939. URL https://ntrs.nasa.gov/citations/19930081468.

108. **Kani, J. N.** and **A. H. Elsheikh** (2017). DR-RNN: A deep residual recurrent neural network for model reduction. *arXiv preprint arXiv:1709.00939*. URL https://doi.org/10.48550/arXiv.1709.00939.

109. **Karniadakis, G. E.**, **I. G. Kevrekidis**, **L. Lu**, **P. Perdikaris**, **S. Wang**, and **L. Yang** (2021). Physics-informed machine learning. *Nature Reviews Physics*, **3**(6), 422–440. URL https://doi.org/10.1038/s42254-021-00314-5.

110. **Katz, J.** and **A. Plotkin**, *Low-Speed Aerodynamics*. Cambridge Aerospace Series. Cambridge University Press, 2001, 2 edition.

111. **Katz, J.** and **D. Weihs** (1978). Hydrodynamic propulsion by large amplitude oscillation of an airfoil with chordwise flexibility. *Journal of Fluid Mechanics*, **88**(3), 485–497. URL https://doi.org/10.1017/S0022112078002220.

112. **Khalid, M. S. U.**, **I. Akhtar**, **H. Dong**, **N. Ahsan**, **X. Jiang**, and **B. Wu** (2018). Bifurcations and route to chaos for flow over an oscillating airfoil. *Journal of Fluids and Structures*, **80**, 262–274. ISSN 0889-9746. URL https://doi.org/10.1016/j.jfluidstructs.2018.04.002.

113. **Khalid, M. S. U.**, **I. Akhtar**, and **N. I. Durrani** (2015). Analysis of strouhal number based equivalence of pitching and plunging airfoils and wake deflection. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, **229**(8), 1423–1434. URL https://doi.org/10.1177/0954410014551847.

114. **Kim, J.**, **D. Kim**, and **H. Choi** (2001). An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, **171**(1), 132–150. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999101967786.

115. **Kim, W.** and **H. Choi** (2019). Immersed boundary methods for fluid-structure interaction: A review. *International Journal of Heat and Fluid Flow*, **75**, 301–309. URL https://www.sciencedirect.com/science/article/pii/S0142727X1830691X.

116. **Kim, Y.**, **Y. Choi**, and **B. Yoo** (2024). Gappy AE: A nonlinear approach for gappy data reconstruction using auto-encoder. *Computer Methods in Applied Mechanics and Engineering*, **426**, 116978. URL https://doi.org/10.1016/j.cma.2024.116978.

117. **Kingma, D. P.** and **J. Ba** (2017). ADAM: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*. URL https://doi.org/10.48550/arXiv.1412.6980.

118. **Koochesfahani, M. M.** (1989). Vortical patterns in the wake of an oscillating airfoil. *AIAA Journal*, **27**(9), 1200–1205. URL https://doi.org/10.2514/3.10246.

119. **Kotsalos, C.**, **J. Latt**, **J. Beny**, and **B. Chopard** (2020). Digital blood in massively parallel cpu/gpu systems for the study of platelet transport. *Interface Focus*, **11**(1), 20190116. ISSN 2042-8898. URL https://doi.org/10.1098/rsfs.2019.0116.

120. **Kou, J.** and **W. Zhang** (2017). An improved criterion to select dominant modes from dynamic mode decomposition. *European Journal of Mechanics - B/Fluids*, **62**, 109–129. ISSN 0997-7546. URL https://www.sciencedirect.com/science/article/pii/S0997754616302990.

121. **Kovachki, N. B.**, **S. Lanthaler**, and **A. M. Stuart** (2024). Chapter 9 - operator learning: Algorithms and analysis. URL https://www.sciencedirect.com/science/article/pii/S1570865924000097.

122. **Krake, T.**, **S. Reinhardt**, **M. Hlawatsch**, **B. Eberhardt**, and **D. Weiskopf** (2021). Visualization and selection of dynamic mode decomposition components for unsteady flow. *Visual Informatics*, **5**(3), 15–27. ISSN 2468-502X. URL https://www.sciencedirect.com/science/article/pii/S2468502X21000309.

123. **Krishnapriyan, A.**, **A. Gholami**, **S. Zhe**, **R. Kirby**, and **M. W. Mahoney** (2021). Characterizing possible failure modes in physics-informed neural networks. **34**, 26548–26560. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

124. **Krizhevsky, A.**, **I. Sutskever**, and **G. E. Hinton** (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, **60**(6), 84–90. ISSN 0001-0782. URL https://doi.org/10.1145/3065386.

125. **Kumar, S. K.** (2017). On weight initialization in deep neural networks. URL https:

//arxiv.org/abs/1704.08863.

126. **Lagaris, I.**, **A. Likas**, and **D. Fotiadis** (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, **9**(5), 987–1000.

127. **Lai, J. C. S.** and **M. F. Platzer** (1999). Jet characteristics of a plunging airfoil. *AIAA Journal*, **37**(12), 1529–1537. URL https://doi.org/10.2514/2.641.

128. **Lee, J.**, **J. Kim**, **H. Choi**, and **K.-S. Yang** (2011). Sources of spurious force oscillations from an immersed boundary method for moving-body problems. *Journal of Computational Physics*, **230**(7), 2677–2695. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S002199911100012X.

129. **Lee, K.** and **K. T. Carlberg** (2020). Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, **404**, 108973. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999119306783.

130. **Lewin, G. C.** and **H. Haj-Hariri** (2003). Modelling thrust generation of a two-dimensional heaving airfoil in a viscous flow. *Journal of Fluid Mechanics*, **492**, 339. URL 10.1017/S0022112003005743.

131. **Li, X.**, **J. Overbey**, **C. Seals**, **A. Lim**, and **P.-C. Shih** (2016). Comparing programmer productivity in openACC and cuda : An empirical investigation. *International Journal of Computer Science, Engineering and Applications*, **6**, 1–15.

132. **Li, Z.**, **N. B. Kovachki**, **K. Azizzadenesheli**, **B. liu**, **K. Bhattacharya**, **A. Stuart**, and **A. Anandkumar**, Fourier neural operator for parametric partial differential equations. *In International Conference on Learning Representations*. 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

133. **Lighthill, M. J.** (1970). Aquatic animal propulsion of high hydromechanical efficiency. *Journal of Fluid Mechanics*, **44**(2), 265–301. URL 10.1017/S0022112070001830.

134. **Liu, F.**, **J. Li**, and **L. Wang** (2023). PI-LSTM: Physics-informed long short-term memory network for structural response modeling. *Engineering Structures*, **292**, 116500. ISSN 0141-0296. URL https://www.sciencedirect.com/science/article/pii/S014102962300915X.

135. **Liu, W.**, **Z. Wang**, **X. Liu**, **N. Zeng**, **Y. Liu**, and **F. E. Alsaadi** (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, **234**, 11–26. ISSN 0925-2312. URL https://www.sciencedirect.com/science/article/pii/S0925231216315533.

136. **Liu, X.** and **J. Katz** (2013). Vortex-corner interactions in a cavity shear layer elucidated by time-resolved measurements of the pressure field. *Journal of Fluid Mechanics*, **728**,

417–457. URL 10.1017/jfm.2013.275.

137. **Lucor, D.**, **A. Agrawal**, and **A. Sergent** (2022). Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection. *Journal of Computational Physics*, **456**, 111022. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999122000845.

138. **Lui, H. F.** and **W. R. Wolf** (2019). Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, **872**, 963–994. URL 10.1017/jfm.2019.358.

139. **Lumley, J. L.** (1967). The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*. URL https://cir.nii.ac.jp/crid/1571980075051475712.

140. **Mader, C. A.**, **G. K. W. Kenway**, **A. Yildirim**, and **J. R. R. A. Martins** (2020). ADflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. *Journal of Aerospace Information Systems*, **17**(9), 508–527. URL https://doi.org/10.2514/1.I010796.

141. **Majumdar, D.**, **C. Bose**, and **S. Sarkar** (2020). Capturing the dynamical transitions in the flow-field of a flapping foil using immersed boundary method. *Journal of Fluids and Structures*, **95**, 102999. ISSN 0889-9746. URL https://www.sciencedirect.com/science/article/pii/S0889974620300128.

142. **Majumdar, D.**, **C. Bose**, and **S. Sarkar** (2022). Transition boundaries and an order-to-chaos map for the flow field past a flapping foil. *Journal of Fluid Mechanics*, **942**, A40. URL https://doi.org/10.1017/jfm.2022.385.

143. **Mardani, M.**, **N. Brenowitz**, **Y. Cohen**, **J. Pathak**, **C.-Y. Chen**, **C.-C. Liu**, **A. Vahdat**, **M. A. Nabian**, **T. Ge**, **A. Subramaniam**, **K. Kashinath**, **J. Kautz**, and **M. Pritchard** (2024). Residual corrective diffusion modeling for km-scale atmospheric downscaling. URL https://arxiv.org/abs/2309.15214.

144. **Mattey, R.** and **S. Ghosh** (2022). A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, **390**, 114474. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782521006939.

145. **McClenny, L. D.** and **U. M. Braga-Neto** (2023). Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, **474**, 111722. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999122007859.

146. **Menet, S.**, **P. Saint-Marc**, and **G. Medioni**, Active contour models: overview, implementation and applications. *In 1990 IEEE International Conference on Systems,*

Man, and Cybernetics Conference Proceedings. 1990.

147. **Meng, X.** and **G. E. Karniadakis** (2020). A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, **401**, 109020. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999119307260.

148. **Menon, K.** and **R. Mittal** (2020). Dynamic mode decomposition based analysis of flow over a sinusoidally pitching airfoil. *Journal of Fluids and Structures*, **94**, 102886. ISSN 0889-9746. URL https://www.sciencedirect.com/science/article/pii/S0889974619308746.

149. **Mittal, R.** and **G. Iaccarino** (2005). Immersed boundary methods. *Annual Review of Fluid Mechanics*, **37**(Volume 37, 2005), 239–261. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev.fluid.37.061903.175743.

150. **Mohd-Yusof, J.** (1997). Combined immersed boundary/b-spline methods for simulations of flows in complex geometries. *CTR annual research briefs, NASA Ames/Stanford Univ.*, 317–327. URL https://cir.nii.ac.jp/crid/1570009750172078080.

151. **Muralidhar, S. D.**, **B. Podvin**, **L. Mathelin**, and **Y. Fraigneau** (2019). Spatio-temporal proper orthogonal decomposition of turbulent channel flow. *Journal of Fluid Mechanics*, **864**, 614–639. URL 10.1017/jfm.2019.48.

152. **Naumov, M.** (2011). Incomplete-LU and Cholesky preconditioned iterative methods using CUSPARSE and CUBLAS. *Nvidia white paper*, **3**.

153. **Nekkanti, A.** and **O. T. Schmidt** (2023). Gappy spectral proper orthogonal decomposition. *Journal of Computational Physics*, **478**, 111950. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999123000451.

154. **Nguyen, T. N. K.**, **T. Dairay**, **R. Meunier**, **C. Millet**, and **M. Mougeot** (2023). Fixed-budget online adaptive learning for physics-informed neural networks. towards parameterized problem inference, 453–468. ISSN 1611-3349. URL http://dx.doi.org/10.1007/978-3-031-36027-5_36.

155. **Nicolaou, L.**, **S. Jung**, and **T. Zaki** (2015). A robust direct-forcing immersed boundary method with enhanced stability for moving body problems in curvilinear coordinates. *Computers & Fluids*, **119**, 101–114. ISSN 0045-7930. URL https://www.sciencedirect.com/science/article/pii/S0045793015002212.

156. **Nidhan, S.**, **K. Chongsiripinyo**, **O. T. Schmidt**, and **S. Sarkar** (2020). Spectral proper orthogonal decomposition analysis of the turbulent wake of a disk at re = 50 000. *Phys. Rev. Fluids*, **5**, 124606. URL https://link.aps.org/doi/10.1103/PhysRevFluids.5.124606.

157. **Noack, B. R.**, **K. Afanasiev**, **M. Morzyński**, **G. Tadmor**, and **F. Thiele** (2003). A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, **497**, 335–363. URL https://doi.org/10.1017/S0022112003006694.

158. **Nony, B. X.**, **M. C. Rochoux**, **T. Jaravel**, and **D. Lucor** (2023). Reduced-order modeling for parameterized large-eddy simulations of atmospheric pollutant dispersion. *Stochastic Environmental Research and Risk Assessment*, **37**(6), 2117–2144. URL https://doi.org/10.1007/s00477-023-02383-7.

159. **Nwankpa, C.**, **W. Ijomah**, **A. Gachagan**, and **S. Marshall** (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*. URL https://arxiv.org/abs/1811.03378.

160. **Ohana, R.**, **M. McCabe**, **L. Meyer**, **R. Morel**, **F. J. Agocs**, **M. Beneitez**, **M. Berger**, **B. Burkhart**, **K. Burns**, **S. B. Dalziel**, **D. B. Fielding**, **D. Fortunato**, **J. A. Goldberg**, **K. Hirashima**, **Y.-F. Jiang**, **R. R. Kerswell**, **S. Maddu**, **J. Miller**, **P. Mukhopadhyay**, **S. S. Nixon**, **J. Shen**, **R. Watteaux**, **B. R.-S. Blancard**, **F. Rozet**, **L. H. Parker**, **M. Cranmer**, and **S. Ho** (2025). The well: a large-scale collection of diverse physics simulations for machine learning. URL https://arxiv.org/abs/2412.00568.

161. **OpenACC-Standard.org** (2015). OpenACC programming and best practices guide.

162. **Paidoussis, M. P.**, *Fluid-structure interactions: slender structures and axial flow*, volume 1. Academic press, 1998.

163. **Païdoussis, M. P.**, **S. J. Price**, and **E. De Langre**, *Fluid-structure interactions: cross-flow-induced instabilities*. Cambridge University Press, 2010.

164. **Paliouras, G.**, **V. Karkaletsis**, and **C. D. Spyropoulos**, *Machine learning and its applications: advanced lectures*, volume 2049. Springer, 2003. URL https://doi.org/10.1007/3-540-44673-7.

165. **Paszke, A.**, **S. Gross**, **F. Massa**, **A. Lerer**, **J. Bradbury**, **G. Chanan**, **T. Killeen**, **Z. Lin**, **N. Gimelshein**, **L. Antiga**, **A. Desmaison**, **A. Köpf**, **E. Yang**, **Z. DeVito**, **M. Raison**, **A. Tejani**, **S. Chilamkurthy**, **B. Steiner**, **L. Fang**, **J. Bai**, and **S. Chintala** (2019). PyTorch: an imperative style, high-performance deep learning library.

166. **Patel, Y.**, **V. Mons**, **O. Marquet**, and **G. Rigas** (2024). Turbulence model augmented physics-informed neural networks for mean-flow reconstruction. *Physical Review Fluids*, **9**(3), 034605. URL https://doi.org/10.1103/PhysRevFluids.9.034605.

167. **Pawar, S.**, **S. E. Ahmed**, **O. San**, and **A. Rasheed** (2020). Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, **32**(3), 036602. ISSN 1070-6631. URL https://doi.org/10.1063/5.0002051.

168. **Peng, W.**, **W. Zhou**, **X. Zhang**, **W. Yao**, and **Z. Liu** (2022). RANG: A Residual-

based Adaptive Node Generation method for physics-informed neural networks. URL https://arxiv.org/abs/2205.01051.

169. **Penwarden, M.**, **A. D. Jagtap**, **S. Zhe**, **G. E. Karniadakis**, and **R. M. Kirby** (2023). A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions. *Journal of Computational Physics*, **493**, 112464. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999123005594.

170. **Perrin, R.**, **M. Braza**, **E. Cid**, **S. Cazin**, **A. Barthet**, **A. Sevrain**, **C. Mockett**, and **F. Thiele** (2007). Obtaining phase averaged turbulence properties in the near wake of a circular cylinder at high reynolds number using POD. *Experiments in Fluids*, **43**, 341–355. URL https://doi.org/10.1007/s00348-007-0347-6.

171. **Peskin, C. S.** (1972). Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, **10**(2), 252–271. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/0021999172900654.

172. **Peskin, C. S.** (2002). The immersed boundary method. *Acta numerica*, **11**, 479–517. URL https://doi.org/10.1017/S0962492902000077.

173. **Peskin, C. S.** and **B. F. Printz** (1993). Improved volume conservation in the computation of flows with immersed elastic boundaries. *Journal of Computational Physics*, **105**(1), 33–46. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S002199918371051X.

174. **Pickering, E.**, **G. Rigas**, **P. A. Nogueira**, **A. V. Cavalieri**, **O. T. Schmidt**, and **T. Colonius** (2020). Lift-up, Kelvin–Helmholtz and Orr mechanisms in turbulent jets. *Journal of Fluid Mechanics*, **896**. URL https://doi.org/10.1017/jfm.2020.301.

175. **Pirnia, A.**, **J. McClure**, **S. Peterson**, **B. Helenbrook**, and **B. Erath** (2020). Estimating pressure fields from planar velocity data around immersed bodies; a finite element approach. *Experiments in Fluids*, **61**(2), 1–16. URL https://doi.org/10.1007/s00348-020-2886-z.

176. **Platzer, M. F.**, **K. D. Jones**, **J. Young**, and **J. C. S. Lai** (2008). Flapping wing aerodynamics: Progress and challenges. *AIAA Journal*, **46**(9), 2136–2149. URL https://doi.org/10.2514/1.29263.

177. **Raghu, S.**, **R. Nayek**, and **V. Chalamalla** (2024). Physics informed neural networks for free shear flows. URL https://arxiv.org/abs/2404.03542.

178. **Raissi, M.** and **G. E. Karniadakis** (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, **357**, 125–141. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999117309014.

179. **Raissi, M.**, **P. Perdikaris**, and **G. Karniadakis** (2019*a*). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, **378**, 686–707. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

180. **Raissi, M.**, **Z. Wang**, **M. S. Triantafyllou**, and **G. E. Karniadakis** (2019*b*). Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, **861**, 119–137. URL https://doi.org/10.1017/jfm.2018.872.

181. **Raissi, M.**, **A. Yazdani**, and **G. E. Karniadakis** (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, **367**(6481), 1026–1030. URL https://www.science.org/doi/abs/10.1126/science.aaw4741.

182. **Ramachandran, P.**, **B. Zoph**, and **Q. V. Le** (2017). Searching for activation functions. URL https://arxiv.org/abs/1710.05941.

183. **Ramesh, K.**, **A. Gopalarathnam**, **K. Granlund**, **M. V. Ol**, and **J. R. Edwards** (2014). Discrete-vortex method with novel shedding criterion for unsteady aerofoil flows with intermittent leading-edge vortex shedding. *Journal of Fluid Mechanics*, **751**, 500–538. URL https://doi.org/10.1017/jfm.2014.297.

184. **Rao, C.**, **H. Sun**, and **Y. Liu** (2020). Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, **10**(3), 207–212. ISSN 2095-0349. URL https://www.sciencedirect.com/science/article/pii/S2095034920300350.

185. **Reguly, I. Z.** and **G. R. Mudalige** (2020). Productivity, performance, and portability for computational fluid dynamics applications. *Computers & Fluids*, **199**, 104425. ISSN 0045-7930. URL https://www.sciencedirect.com/science/article/pii/S0045793020300013.

186. **Renganathan, S. A.**, **R. Maulik**, and **V. Rao** (2020). Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil. *Physics of Fluids*, **32**(4), 047110. ISSN 1070-6631. URL https://doi.org/10.1063/1.5144661.

187. **Rombach, R.**, **A. Blattmann**, **D. Lorenz**, **P. Esser**, and **B. Ommer** (2022). High-resolution image synthesis with latent diffusion models. URL https://arxiv.org/abs/2112.10752.

188. **Rowley, C. W.** (2005). Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, **15**(03), 997–1013. URL https://doi.org/10.1142/S0218127405012429.

189. **Rowley, C. W.** and **S. T. Dawson** (2017). Model reduction for flow analysis

and control. *Annual Review of Fluid Mechanics*, **49**(Volume 49, 2017), 387–417. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-010816-060042.

190. **Rowley, C. W.**, **I. Mezić**, **S. Bagheri**, **P. Schlatter**, and **D. S. Henningson** (2009). Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, **641**, 115–127. URL https://doi.org/10.1017/S0022112009992059.

191. **Sane, S. P.** and **M. H. Dickinson** (2002). The aerodynamic effects of wing rotation and a revised quasi-steady model of flapping flight. *Journal of Experimental Biology*, **205**(8), 1087–1096. ISSN 0022-0949. URL https://doi.org/10.1242/jeb.205.8.1087.

192. **Sarrate, J.**, **A. Huerta**, and **J. Donea** (2001). Arbitrary lagrangian–eulerian formulation for fluid–rigid body interaction. *Computer Methods in Applied Mechanics and Engineering*, **190**(24), 3171–3188. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S004578250000387X. Advances in Computational Methods for Fluid-Structure Interaction.

193. **Schmidt, O. T.**, **A. Towne**, **G. Rigas**, **T. Colonius**, and **G. A. Brès** (2018). Spectral analysis of jet turbulence. *Journal of Fluid Mechanics*, **855**, 953–982. URL https://doi.org/10.1017/jfm.2018.675.

194. **Shah, C. L.**, **D. Majumdar**, **C. Bose**, and **S. Sarkar** (2024*a*). Controlling the chaotic wake of a flapping foil by tuning its chordwise flexibility. *Journal of Fluids and Structures*, **127**, 104134. ISSN 0889-9746. URL https://www.sciencedirect.com/science/article/pii/S0889974624000690.

195. **Shah, C. L.**, **D. Majumdar**, and **S. Sarkar**, Performance enhancement of an immersed boundary method based FSI solver using OpenMP. *In Annual CFD Symposium*. NAL, Bangalore, 2019.

196. **Shah, C. L.**, **D. Majumdar**, and **S. Sarkar**, Investigating the dynamical behaviour of Dipteran flight-inspired flapping motion using immersed boundary method-based FSI solver. *In Recent Advances in Computational Mechanics and Simulations*. Springer, 2021, 259–270. URL https://doi.org/10.1007/978-981-15-8315-5_23.

197. **Shah, C. L.**, **K. Sourav**, and **S. Sarkar** (2024*b*). Dynamic coupling of wing mechanics and aerodynamics in dipteran-inspired flapping wing systems. *Physics of Fluids*, **36**(9). URL https://doi.org/10.1063/5.0224091.

198. **Shelley, M. J.** and **J. Zhang** (2011). Flapping and bending bodies interacting with fluid flows. *Annual Review of Fluid Mechanics*, **43**(Volume 43, 2011), 449–465. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-121108-145456.

199. **Shukla, K.**, **M. Xu**, **N. Trask**, and **G. E. Karniadakis** (2022). Scalable algorithms for

physics-informed neural and graph networks. *Data-Centric Engineering*, **3**, e24. URL https://doi.org/10.1017/dce.2022.24.

200. **Shyy, W.**, **H. Aono**, **C.-k. Kang**, and **H. Liu**, *An introduction to flapping wing aerodynamics*, volume 37. Cambridge University Press, 2013.

201. **Shyy, W.**, **C.-k. Kang**, **P. Chirarattananon**, **S. Ravi**, and **H. Liu** (2016). Aerodynamics, sensing and control of insect-scale flapping-wing flight. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **472**(2186), 20150712. ISSN 1364-5021. URL https://doi.org/10.1098/rspa.2015.0712.

202. **Singh, K.**, **S. Michelin**, and **E. de Langre** (2012). Energy harvesting from axial fluid-elastic instabilities of a cylinder. *Journal of Fluids and Structures*, **30**, 159–172. ISSN 0889-9746. URL https://www.sciencedirect.com/science/article/pii/S0889974612000321.

203. **Sirovich, L.** (1987). Turbulence and the dynamics of coherent structures. I. coherent structures. *Quarterly of applied mathematics*, **45**(3), 561–571. URL https://doi.org/10.1090/qam/910462.

204. **Sliwinski, L.** and **G. Rigas** (2023). Mean flow reconstruction of unsteady flows using physics-informed neural networks. *Data-Centric Engineering*, **4**, e4. URL https://doi.org/10.1017/dce.2022.37.

205. **Sofi, S. S.** and **I. Oseledets** (2025). A case study of spatiotemporal forecasting techniques for weather forecasting. *GeoInformatica*, **29**(2), 275–298. URL https://doi.org/10.1007/s10707-024-00530-y.

206. **Sørensen, J. N.** (2011). Aerodynamic aspects of wind energy conversion. *Annual Review of Fluid Mechanics*, **43**(Volume 43, 2011), 427–448. ISSN 1545-4479. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-122109-160801.

207. **Su, C.**, **J. Liang**, and **Z. He** (2024). E-PINN: A fast physics-informed neural network based on explicit time-domain method for dynamic response prediction of nonlinear structures. *Engineering Structures*, **321**, 118900. ISSN 0141-0296. URL https://www.sciencedirect.com/science/article/pii/S0141029624014627.

208. **Sugawara, M.**, **S. Hirasawa**, **K. Komatsu**, **H. Takizawa**, and **H. Kobayashi**, A comparison of performance tunabilities between OpenCL and OpenACC. *In 2013 IEEE 7th International Symposium on Embedded Multicore Socs*. 2013. URL https://doi.org/10.1109/MCSoC.2013.31.

209. **Sun, L.**, **H. Gao**, **S. Pan**, and **J.-X. Wang** (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, **361**, 112732. ISSN 0045-7825. URL https:

//www.sciencedirect.com/science/article/pii/S004578251930622X.

210. **Sundar, R.**, **Y. Hu**, **N. Parashar**, **A. Blanchard**, and **B. Dodov** (2025). Taudiff: Highly efficient kilometer-scale downscaling using generative diffusion models. URL https://arxiv.org/abs/2412.13627.

211. **Sundar, R.**, **D. Majumdar**, **D. Lucor**, and **S. Sarkar** (2024). Physics-informed neural networks modelling for systems with moving immersed boundaries: Application to an unsteady flow past a plunging foil. *Journal of Fluids and Structures*, **125**, 104066. ISSN 0889-9746. URL https://www.sciencedirect.com/science/article/pii/S088997462400001X.

212. **Taira, K.**, **S. L. Brunton**, **S. T. Dawson**, **C. W. Rowley**, **T. Colonius**, **B. J. McKeon**, **O. T. Schmidt**, **S. Gordeyev**, **V. Theofilis**, and **L. S. Ukeiley** (2017). Modal analysis of fluid flows: An overview. *AIAA Journal*, **55**(12), 4013–4041. URL https://doi.org/10.2514/1.J056060.

213. **Taira, K.**, **M. S. Hemati**, **S. L. Brunton**, **Y. Sun**, **K. Duraisamy**, **S. Bagheri**, **S. T. M. Dawson**, and **C.-A. Yeh** (2020). Modal analysis of fluid flows: Applications and outlook. *AIAA Journal*, **58**(3), 998–1022. URL https://doi.org/10.2514/1.J058462.

214. **Tang, H.**, **Y. Liao**, **H. Yang**, and **L. Xie** (2022). A transfer learning-physics informed neural network (TL-PINN) for vortex-induced vibration. *Ocean Engineering*, **266**, 113101. ISSN 0029-8018. URL https://www.sciencedirect.com/science/article/pii/S0029801822023848.

215. **Tang, K.**, **X. Wan**, and **C. Yang** (2023). DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, **476**, 111868. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999122009317.

216. **Tartakovsky, A. M.**, **C. O. Marrero**, **P. Perdikaris**, **G. D. Tartakovsky**, and **D. Barajas-Solano** (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, **56**(5), e2019WR026731. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019WR026731. E2019WR026731 10.1029/2019WR026731.

217. **Theodorsen, T.** (1935). General theory of aerodynamic instability and the mechanism of flutter. *NACA Technical Report*, (496).

218. **Thirunavukkarasu, A.**, **R. Sundar**, and **S. Sarkar** (2024). Comparative analysis of model reduction techniques for flapping wing dynamics. *Physics of Fluids*, **36**(6), 067109. ISSN 1070-6631. URL https://doi.org/10.1063/5.0209683.

219. **Toscano, J. D.**, **V. Oommen**, **A. J. Varghese**, **Z. Zou**, **N. Ahmadi Daryakenari**, **C. Wu**, and **G. E. Karniadakis** (2025). From PINNs to PIKANS: Recent advances in

physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, **1**(1), 1–43. URL https://doi.org/10.1007/s44379-025-00015-1.

220. **Towne, A.**, **S. T. M. Dawson**, **G. A. Brès**, **A. Lozano-Durán**, **T. Saxton-Fox**, **A. Parthasarathy**, **A. R. Jones**, **H. Biler**, **C.-A. Yeh**, **H. D. Patel**, and **K. Taira** (2023). A database for reduced-complexity modeling of fluid flows. *AIAA Journal*, **61**(7), 2867–2892. URL https://doi.org/10.2514/1.J062203.

221. **Towne, A.**, **O. T. Schmidt**, and **T. Colonius** (2018). Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis. *Journal of Fluid Mechanics*, **847**, 821–867. URL https://doi,org/10.1017/jfm.2018.283.

222. **Vaswani, A.**, **N. Shazeer**, **N. Parmar**, **J. Uszkoreit**, **L. Jones**, **A. N. Gomez**, **L. Kaiser**, and **I. Polosukhin** (2023). Attention is all you need. URL https://arxiv.org/abs/1706.03762.

223. **Venturi, D.** and **G. E. Karniadakis** (2004). Gappy data and reconstruction procedures for flow past a cylinder. *Journal of Fluid Mechanics*, **519**, 315–336. URL https://doi.org/10.1017/S0022112004001338.

224. **Vinuesa, R.** and **S. L. Brunton** (2022*a*). Emerging Trends in Machine Learning for Computational Fluid Dynamics . *Computing in Science & Engineering*, **24**(05), 33–41. ISSN 1558-366X. URL https://doi.ieeecomputersociety.org/10.1109/MCSE.2023.3264340.

225. **Vinuesa, R.** and **S. L. Brunton** (2022*b*). Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, **2**(6), 358–366. URL https://doi.org/10.1038/s43588-022-00264-7.

226. **Volk, M.-C.**, **A. Sergent**, **D. Lucor**, **M. Mommert**, **C. Bauer**, and **C. Wagner** (2025). A PINN methodology for temperature field reconstruction in the PIV measurement plane: Case of Rayleigh-Bénard convection. URL https://arxiv.org/abs/2503.23801.

227. **Wagner, H.** (1925). Über die entstehung des dynamischen auftriebes von tragflügeln. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, **5**(1), 17–35. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19250050103.

228. **Wang, B.** and **M. Yu**, *Analysis of Wake Structures Behind an Oscillating Square Cylinder Using Dynamic Mode Decomposition*. URL https://arc.aiaa.org/doi/abs/10.2514/6.2016-3779.

229. **Wang, C.**, **W. Zhang**, **J. Hu**, **J. Zhao**, and **Y. Zou** (2020*a*). A modified quasisteady aerodynamic model for a sub-100 mg insect-inspired flapping-wing robot. *Applied Bionics and Biomechanics*, **2020**(1), 8850036. URL https://doi.org/10.1155/

2020/8850036.

230. **Wang, H.**, **F. Wu**, **Y. Liu**, **X. He**, **S. Feng**, and **S. Wang** (2025). Machine-learning-based pressure reconstruction with moving boundaries. *Journal of Fluid Mechanics*, **1008**, A21. URL 10.1017/jfm.2025.91.

231. **Wang, R.**, **K. Kashinath**, **M. Mustafa**, **A. Albert**, and **R. Yu**, Towards physics-informed deep learning for turbulent flow prediction. *In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20. Association for Computing Machinery, New York, NY, USA, 2020*b*. ISBN 9781450379984. URL https://doi.org/10.1145/3394486.3403198.

232. **Wang, S.** and **P. Perdikaris** (2021). Deep learning of free boundary and stefan problems. *Journal of Computational Physics*, **428**, 109914. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/S0021999120306884.

233. **Wang, S.**, **S. Sankaran**, and **P. Perdikaris** (2024). Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **421**, 116813. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782524000690.

234. **Wang, S.**, **Y. Teng**, and **P. Perdikaris** (2021*a*). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, **43**(5), A3055–A3081. URL https://doi.org/10.1137/20M1318043.

235. **Wang, S.**, **H. Wang**, and **P. Perdikaris** (2021*b*). On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **384**, 113938. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782521002759.

236. **Wang, T.-K.** and **K. Shoele** (2021). Geometrically weighted modal decomposition techniques. *Journal of Fluid Mechanics*, **911**, A41. URL 10.1017/jfm.2020.1090.

237. **Wang, Z.**, **L. Du**, **J. Zhao**, **M. C. Thompson**, and **X. Sun** (2020*c*). Flow-induced vibrations of a pitching and plunging airfoil. *Journal of Fluid Mechanics*, **885**, A36. URL 10.1017/jfm.2019.996.

238. **Wick, T.** (2013). Solving monolithic fluid-structure interaction problems in arbitrary lagrangian eulerian coordinates with the deal. ii library. *Archive of Numerical Software*, **1**(1), 1–19. URL https://doi.org/10.11588/ans.2013.1.10305.

239. **Willard, J.**, **X. Jia**, **S. Xu**, **M. Steinbach**, and **V. Kumar** (2022). Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Comput. Surv.*, **55**(4). ISSN 0360-0300. URL https://doi.org/10.1145/3514228.

240. **Willcox, K.** (2006). Unsteady flow sensing and estimation via the gappy proper orthogonal

decomposition. *Computers & Fluids*, **35**(2), 208–226. ISSN 0045-7930. URL https://www.sciencedirect.com/science/article/pii/S0045793005000113.

241. **Wong, J. C.**, **P.-H. Chiu**, **C. C. Ooi**, and **M. H. Da** (2022). Robustness of physics-informed neural networks to noise in sensor data. URL https://arxiv.org/abs/2211.12042.

242. **Wood, S. L.**, **K. Jacobson**, **W. T. Jones**, and **W. K. Anderson**, *Sparse Linear Algebra Toolkit for Computational Aerodynamics*. URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-0317.

243. **Wu, C.**, **M. Zhu**, **Q. Tan**, **Y. Kartha**, and **L. Lu** (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, **403**, 115671. ISSN 0045-7825. URL https://www.sciencedirect.com/science/article/pii/S0045782522006260.

244. **Wu, X.**, **X. Zhang**, **X. Tian**, **X. Li**, and **W. Lu** (2020). A review on fluid dynamics of flapping foils. *Ocean Engineering*, **195**, 106712. ISSN 0029-8018. URL https://www.sciencedirect.com/science/article/pii/S0029801819308261.

245. **Xiang, Z.**, **W. Peng**, **X. Liu**, and **W. Yao** (2022). Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, **496**, 11–34. ISSN 0925-2312. URL https://www.sciencedirect.com/science/article/pii/S092523122200546X.

246. **Xiao, D.**, **Z. Lin**, **F. Fang**, **C. C. Pain**, **I. M. Navon**, **P. Salinas**, and **A. Muggeridge** (2017). Non-intrusive reduced-order modeling for multiphase porous media flows using smolyak sparse grids. *International Journal for Numerical Methods in Fluids*, **83**(2), 205–219. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4263.

247. **Xiao, Q.** and **Q. Zhu** (2014). A review on flow energy harvesters based on flapping foils. *Journal of Fluids and Structures*, **46**, 174–191. ISSN 0889-9746. URL https://www.sciencedirect.com/science/article/pii/S0889974614000140.

248. **Xie, C.**, **N. Gao**, **Y. Mend**, **Y. Wu**, and **C. Yang** (2023). A review of bird-like flapping wing with high aspect ratio. *Chinese Journal of Aeronautics*, **36**(1), 22–44. ISSN 1000-9361. URL https://www.sciencedirect.com/science/article/pii/S1000936122001170.

249. **Xu, L.**, **Z. Liu**, **X. Li**, **M. Zhao**, and **Y. Zhao** (2023). An improved mode time coefficient for dynamic mode decomposition. *Physics of Fluids*, **35**(10), 105106. ISSN 1070-6631. URL https://doi.org/10.1063/5.0166272.

250. **Xue, W.** and **C. J. Roy**, *Heterogeneous Computing of CFD Applications on CPU-GPU Platforms using OpenACC Directives*. URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-1046.

251. **Yang, L.**, **Z. Zhang**, **Y. Song**, **S. Hong**, **R. Xu**, **Y. Zhao**, **W. Zhang**, **B. Cui**, and **M.-H. Yang** (2023*a*). Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.*, **56**(4). ISSN 0360-0300. URL https://doi.org/10.1145/3626235.

252. **Yang, Q.**, **Y. Yang**, **T. Cui**, and **Q. He** (2023*b*). FDM-PINN: Physics-informed neural network based on fictitious domain method. *International Journal of Computer Mathematics*, **100**(3), 511–524. URL https://doi.org/10.1080/00207160.2022.2128674.

253. **Young, J.** and **J. C. S. Lai** (2007). Vortex lock-in phenomenon in the wake of a plunging airfoil. *AIAA Journal*, **45**(2), 485–490. URL https://doi.org/10.2514/1.23594.

254. **Yuen, D. A.**, **L. Wang**, **X. Chi**, **L. Johnsson**, **W. Ge**, and **Y. Shi**, *GPU solutions to multi-scale problems in science and engineering*. Springer, 2013.

255. **Zhou, L.** and **X. Wu** (2020). Globally optimal surface segmentation using deep learning with learnable smoothness priors. URL https://arxiv.org/abs/2007.01217.

256. **Zhu, X.**, **X. Hu**, and **P. Sun** (2023). Physics-informed neural networks for solving dynamic two-phase interface problems. *SIAM Journal on Scientific Computing*, **45**(6), A2912–A2944. URL https://doi.org/10.1137/22M1517081.

257. **Zhu, Y.**, **W. Kong**, **J. Deng**, and **X. Bian** (2024). Physics-informed neural networks for incompressible flows with moving boundaries. *Physics of Fluids*, **36**(1), 013617. ISSN 1070-6631. URL https://doi.org/10.1063/5.0186809.

CURRICULUM VITAE

NAME Rahul Sundar

DATE OF BIRTH 31 August 1995

EDUCATION QUALIFICATIONS

2017 **B.E. (Honors)**

Institution	Birla Institute of Technology and Science, Pilani
Specialization	Mechanical Engineering

Doctor of Philosophy

Institution	Indian Institute of Technology Madras
Specialization	Aerospace Engineering
Registration Date	9 July 2018

DOCTORAL COMMITTEE

Chairperson

Prof H S N Murthy
Head of the Department
Department of Aerospace Engineering
Indian Institute of Technology Madras

Guide(s)

Prof. Sunetra Sarkar
Biomimetics and Dynamics Lab
Department of Aerospace Engineering
Indian Institute of Technology Madras

Member(s)

Prof. R I Sujith
Department of Aerospace Engineering
Indian Institute of Technology Madras

Prof. Nandan Kumar Sinha
Department of Aerospace Engineering
Indian Institute of Technology Madras

Prof. Shaligram Tiwari
Department of Mechanical Engineering
Indian Institute of Technology Madras