Physics-Informed Neural Networks for Solving Two-Dimensional Potential Flow Problems: Formulation, Implementation, and Analysis

A project report submitted to the Department of Computer Science, Mahindra University, by

Aditya Sai Ellendula (SE 20 UARI 005)
Himavanth Kandula (SE 20 UARI 064)
Hrushith Ram Koduri (SE 20 UARI 065)
Iragavarapu Sai Ajay (SE 20 UCSE 056)

under the guidance and supervision of
Dr. ARYA KUMAR BHATTACHARYA,
Dean Research, École Centrale School of Engineering, Mahindra University,
Hyderabad-500043, Telangana, India,

in partial fulfillment of the requirements of the three-credit project course in the $4-1$ Semester.

École Centrale School of Engineering, Mahindra University,
Hyderabad-500043, Telangana, India.
December, 2023

ÉCOLE CENTRALE SCHOOL OF ENGINEERING,
HYDERABAD-500043, TELANGANA, INDIA.

ACKNOWLEDGEMENTS

We would like to take this opportunity to express **our** sincere gratitude to all **our** professors who have contributed to the successful completion of this project. Without their support, guidance, and encouragement, this endeavor would not have been possible.

To our supervisor Dr.Arya Kumar Bhattacharya, thank you for your unwavering guidance, invaluable expertise, and constant encouragement throughout the duration of this project. Your insights and feedback have been instrumental in shaping the direction and content of this report.

We would like to thank the PhD students along with Prof. who have imparted their knowledge and expertise during our journey. Your teachings and support have laid a strong foundation for **the** understanding of the subject matter.

We would like to acknowledge the support and resources provided by Mahindra University. The facilities, access to research materials, and other resources have played a crucial role in the completion of this project.

We would also like to extend our thanks to all the authors, researchers, and publishers whose work has been referenced in this report. Their contributions to the field have served as a foundation for **our** own knowledge of the subject.

Once again, **we** express **our** deepest appreciation to all those mentioned above, as well as any others who may have contributed in ways that **we** may have unintentionally omitted. Your support has been invaluable, and **we** are grateful for your contributions to the successful completion of this project report.

Aditya Sai Ellendula (SE 20 UARI 005)
Himavanth Kandula (SE 20 UARI 064)
Hrushith Ram Koduri (SE 20 UARI 065)
Iragavarapu Sai Ajay (SE 20 UCSE 056)

ÉCOLE CENTRALE SCHOOL OF ENGINEERING, HYDERABAD-500043, TELANGANA, INDIA.

CERTIFICATE

This is to certify that the project report entitled "Development of Software for Physics Informed Neural Networks using System Engineering Principes", submitted by Ellendula Aditya Sai, Hrushith Ram Koduri, Himavanth Kandula, Iragavarapu Sai Ajay in partial fulfilment of the requirements of the three-credit project course embodies the work carried out by him under my supervision and guidance.

Dr. Arya Kumar Bhattacharya,
Project Supervisor,
Dean Research, École Centrale School of Engineering,
Mahindra University, Hyderabad-500043, Telangana, India.

Date: 30th December, 2023.

1 ABSTRACT

This project focuses on the completion of the development of framework for physics-informed neural networks (PINNs) which we have started in the previous semester. The framework, implemented using PyTorch, enables the application of PINNs in various domains, with specific emphasis on Fluid Dynamics, ElectroMagnetics, and Solid Mechanics. The framework provides a flexible and generalized procedure for incorporating physics-based constraints into neural network architectures, facilitating accurate predictions and improved understanding of complex physical systems.

After completion of the implementation of the framework, we further delved into the implementation of Numerical Solution of Partial Differential Equations (PDEs), specifically focusing on the Laplace equation governing the velocity potential in incompressible, inviscid, irrotational fluid flow as an application for validating the working of our framework. Utilizing a physics-informed neural network framework developed in a preceding semester, the study explores its application to a two-dimensional channel with steady boundary conditions. PINNs can adapt to irregular geometries and complex domain shapes without the need for mesh generation. Traditional numerical methods may require fine mesh refinement to accurately capture details in the solution. PINNs are less sensitive to mesh resolution, potentially reducing computational costs associated with refining the mesh. The project involves the transformation of mathematical formulations into executable code, execution of the program, and the subsequent visualization and interpretation of results. By examining the computed solutions, this work aims to provide insights into the fluid dynamics of the specified geometry, point distribution, and boundary conditions. The findings contribute to the broader understanding of numerical methods in solving PDEs and their application to fluid dynamics.

Keywords: *physics-informed neural networks, system engineering principles, PyTorch, fluid dynamics, electro dynamics, thermodynamics, computational framework, prediction accuracy, modular design, user-friendly, research and engineering applications.*

2 INTRODUCTION

In the realm of computational modeling and simulation, the convergence of artificial intelligence and physics-based methodologies has given rise to innovative approaches for solving complex problems. This project stands at the intersection of these realms, focusing on the implementation of Physics-Informed Neural Networks (PINNs) within a meticulously designed framework. Physics-informed neural networks (PINNs) represent a paradigm shift in the approach to solving complex physical problems, offering a potent synthesis of machine learning and physics-based modeling. In contrast to traditional numerical methods, such as finite element analysis and finite difference methods, PINNs hold a distinct advantage in their ability to seamlessly integrate domain knowledge and physical principles into their architecture.

Numerical approximation methods, while successful in many applications, often come with computational expenses and demand considerable expertise for accurate setup and calibration. PINNs, on the other hand, capitalize on the representational power of neural networks while incorporating the intrinsic physics of the problem. This unique approach involves training neural networks to adhere to governing equations and boundary conditions, allowing for a more efficient and accurate capture of the intricate interplay of physical phenomena.

A pivotal advantage of PINNs lies in their capacity to learn from sparse and noisy data, a characteristic that distinguishes them from traditional numerical techniques relying on grid-based discretization. In scenarios where experimental data is limited or costly to obtain, PINNs shine by generalizing from a modest dataset, providing precise predictions even under such constraints.

Over the past few decades, artificial neural networks (ANNs), particularly multilayer perceptrons, have predominantly operated on data generated from various processes. Simultaneously, the simulation of complex physical phenomena, described by sets of partial (mostly) differential equations formulated in the 19th and 20th centuries, has been achieved through numerical solutions of these equations. However, ANNs are now pushing beyond the traditional confines of data-driven and simulation-based tracks. They are being employed to directly solve governing equations, effectively bypassing both the data-generation and numerical simulation tracks.

This innovative approach involves mapping the governing equations relevant to a given domain into the loss functions of ANNs, constituting a form of semi-supervised learning. Initial solutions, albeit in approximated forms, have been obtained, heralding promising breakthroughs in various fields. Challenges such as resolving turbulence to the smallest scales, handling shocks in aerodynamics, addressing magneto-hydrodynamics, fluid-structure interactions, solidification, and microstructures are among the forefront concerns, opening avenues for significant advancements in the field. In this context, PINNs emerge as a compelling choice, providing a bridge between data-driven learning and rigorous physics-based simulations, promising not only computational efficiency but also breakthroughs in resolving complex physical phenomena with unprecedented accuracy.

The objectives of this project include developing a user-friendly and efficient software package that combines the computational efficiency of PyTorch with the physics-awareness of PINNs. The software's emphasis on interpretability and the incorporation of domain-specific knowledge and constraints enables improved accuracy, faster convergence, and reduced data requirements. Through comprehensive documentation and example applications in different domains, this project aims to provide researchers with a powerful tool to address complex physics-based problems more effectively.

3 METHODOLOGY

PINNs combine the power of neural networks with the governing equations of physics to create physics-informed models. The methodology involves incorporating the physics-based constraints directly into the training process of the neural network. This ensures that the predictions of the network adhere to the underlying physical laws, even in the presence of limited or noisy data. The general workflow of PINNs consists of the following steps:

- **Step 1: General Form of Sets of PDE:**
  The general form of sets of partial differential equations (PDEs) applicable to domains like Fluid Mechanics can be represented as follows:

$$Ni[u](x) = f_i(x), \quad \forall i \in \{1, \ldots, N_N\}, \quad x \in D \tag{1}$$

$$C_j[u](x) = g_j(x), \quad \forall j \in \{1, \ldots, N_C\}, \quad x \in \partial D \tag{2}$$

  where:
  $Ni[.]$ are general differential operators applied to the functions $u(x)$
  x represents the set of independent location vectors defined over a bounded continuous domain $D \subset \mathbb{R}^d$ where $d \in \{1, 2, 3, \ldots\}$.
  N_N represents the number of operators corresponding to equations in the set.

  u is a vector of dependent variables representing the solution at the field points x
  $C_j[.]$ denote constraint operators that consist of differential, linear, and nonlinear terms, usually covering the boundary and initial conditions
  N_C represents the number of constraint operators
  ∂D denotes a subset of the domain boundary needed to define the constraints

  As an example, for 3D laminar flow described by the Navier-Stokes equations, the number of equations N_N is 4, consisting of 3 momentum equations and 1 continuity equation. The boundary conditions represented by equation (2) depend on the specifics of the flow, where on solid boundaries, they typically involve zero normal and tangential flow conditions. In this framework, the components of u at any x are the 3 velocity components and the density.

- **Step 2: Approximating the Solution with a Neural Network**
  To approximate the solution $u(x)$, a neural network $u_{net}(x; \theta)$ is used, where θ represents the vector of parameters of the neural network. The equations (1) and (2) can then be expressed in terms of the neural network as follows:

$$Ni[u_{\text{net}}(\theta)](x) - f_i(x) = 0, \quad \forall i \in \{1, \ldots, N_N\}, \quad x \in D \tag{3}$$

$$C_j[u_{\text{net}}(\theta)](x) - g_j(x) = 0, \quad \forall j \in \{1, \ldots, N_C\}, \quad x \in \partial D \tag{4}$$

  However, no machine learning (ML) model can generate precisely zero error, so it is reasonable to write the equations in the form of residuals:

$$r(i)(x; u_{\text{net}}(\theta)) = N[u_{\text{net}}(\theta)](x) - f_i(x) \tag{5}$$

$$r(j)(x; u_{\text{net}}(\theta)) = C[u_{\text{net}}(\theta)](x) - g_j(x) \tag{6}$$

  where:
  $r(i)(x; u_{\text{net}}(\theta))$ represents residual function for the ith equation $r(j)(x; u_{\text{net}}(\theta))$ represents residual function for the jth equation $N[u_{\text{net}}(\theta)](x)$ represents the application of the operators N on $u_{net}(\theta)$
  $C[u_{\text{net}}(\theta)](x)$represents the application of the operators C on $u_{net}(\theta)$

- **Step 3: Training the Neural Network**
  The goal is to train the artificial neural network (ANN) to find the best possible parameters θ that enable $u_net(x;\theta)$ to represent equations (3) and (4) as closely as possible. This is achieved by minimizing the net loss function:

$$\min_{\theta}\left(\sum_{i=1}^{N_N}\lambda(i)\|\mathbf{r}(i)(x;u_{\text{net}}(\theta))\|_p^p+\sum_{j=1}^{N_C}\lambda(j)\|\mathbf{r}(j)(x;u_{\text{net}}(\theta))\|_p^p\right)$$

  where:
  $\|.\|_p$ denotes the p-norm
  $\lambda(i)$ and $\lambda(j)$ are weight functions controlling the loss interplay between equation terms and constraint terms, as well as across different equation and constraint terms Note: The values of $\lambda(i)$ and $\lambda(j)$ can evolve during the convergence process and play a crucial role in the accuracy and convergence of the solutions.

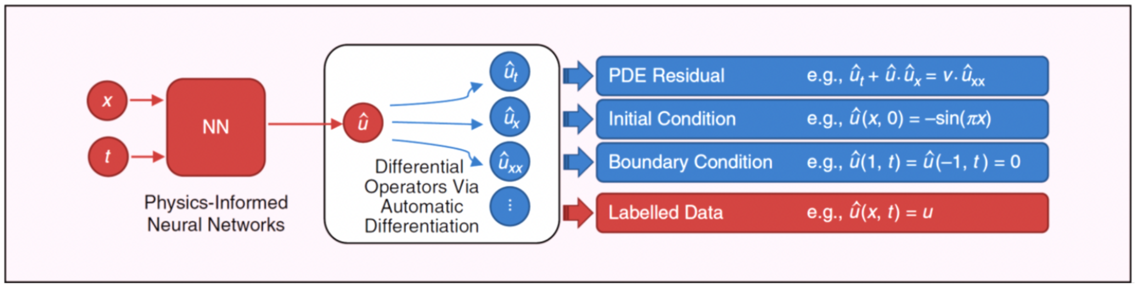
- **Step 4: Integration and Loss Calculation**

$$\mathcal{L}_{res}(\theta)=\left(\sum_{i=1}^{N_N}\int_D\lambda(i)\|\mathbf{r}(i)(x;u_{\text{net}}(\theta))\|_p\,dx\right)+\left(\sum_{j=1}^{N_C}\int_D\lambda(j)\|\mathbf{r}(j)(x;u_{\text{net}}(\theta))\|_p\,dx\right) \qquad (7)$$

  Equation (7) needs further explanation as it represents a confluence of Computational Fluid Dynamics (CFD) and ANN perspectives. The integration symbols do not denote integration over continuous spaces, but rather summation and possibly Monte-Carlo integration over a large number of points selected within and on the respective volumes and surfaces. Each point is represented as x In CFD, grids (structured or unstructured) are commonly used, and accuracy depends on grid sizes. However, in this context, we are considering clouds of points, where a higher number of points provides better representation.

  An ANN trains by reducing the objective function, cost function, or loss function towards a minimum, as close to zero as possible. In supervised learning (the most common branch of machine learning), training samples are presented to the ANN, consisting of input data points and corresponding output data points. The ANN takes the input of each sample, generates an output, and the deviation between the sample's output and the ANN's output constitutes the loss function value for that sample. The loss is then backpropagated through the weights (parameters) of the ANN, adjusting each weight to reduce the loss. This process is typically performed using batches of training samples.

  However, the method described in the provided equations is not supervised learning because the outputs for each sample are not explicitly defined as data points but rather as the degree of "dissatisfaction" of the equations expressed in equations (5) and (6). These dissatisfactions are then converted to the loss value in equation (7) and used to adjust the weights of the ANN to learn the underlying functionality of these equations. Hence, it is more appropriate to categorize this approach as unsupervised learning.


x
t
NN
Physics-Informed Neural Networks
$\hat{u}$
Differential Operators Via Automatic Differentiation
$\hat{u}_t$
$\hat{u}_x$
$\hat{u}_{xx}$
⋮
PDE Residual
e.g., $\hat{u}_t + \hat{u}\cdot\hat{u}_x = v\cdot\hat{u}_{xx}$
Initial Condition
e.g., $\hat{u}(x, 0) = -\sin(\pi x)$
Boundary Condition
e.g., $\hat{u}(1, t) = \hat{u}(-1, t) = 0$
Labelled Data
e.g., $\hat{u}(x, t) = u$

The complete process is illustrated in the above Figure. The "NN" box is the core mechanism that is sought to be learnt (strictly speaking, its parameters not shown explicitly in the illustration). Its inputs are the points cloud x, and if an unsteady problem, then x repeated at different time steps. The outputs are the values of vector unet or uˆ (so for the example of laminar NS flow it will be 3 velocity components and density) at each of the input points. Then Automatic Differentiation is used to evaluate each of the derivative terms in all the PDEs and Constraints (NN + NC), which are finally converted into the different components of the Loss as in eq. (7).

- **Step 5: Coverage of Flow Conditions and Training Efficiency**
  In the training process, it is crucial to ensure that the different boundary and initial conditions that map into different training samples cover the total range of flow conditions around or through the geometry of interest with a fairly uniform distribution. This requirement ensures that the ANN learns the neural network representation $u_{net}(.)$ for all flow conditions related to the geometry of interest and the field points within the volume of interest. It should be noted that the time required to train such an ANN would be comparable to that of a single simulation in traditional CFD.

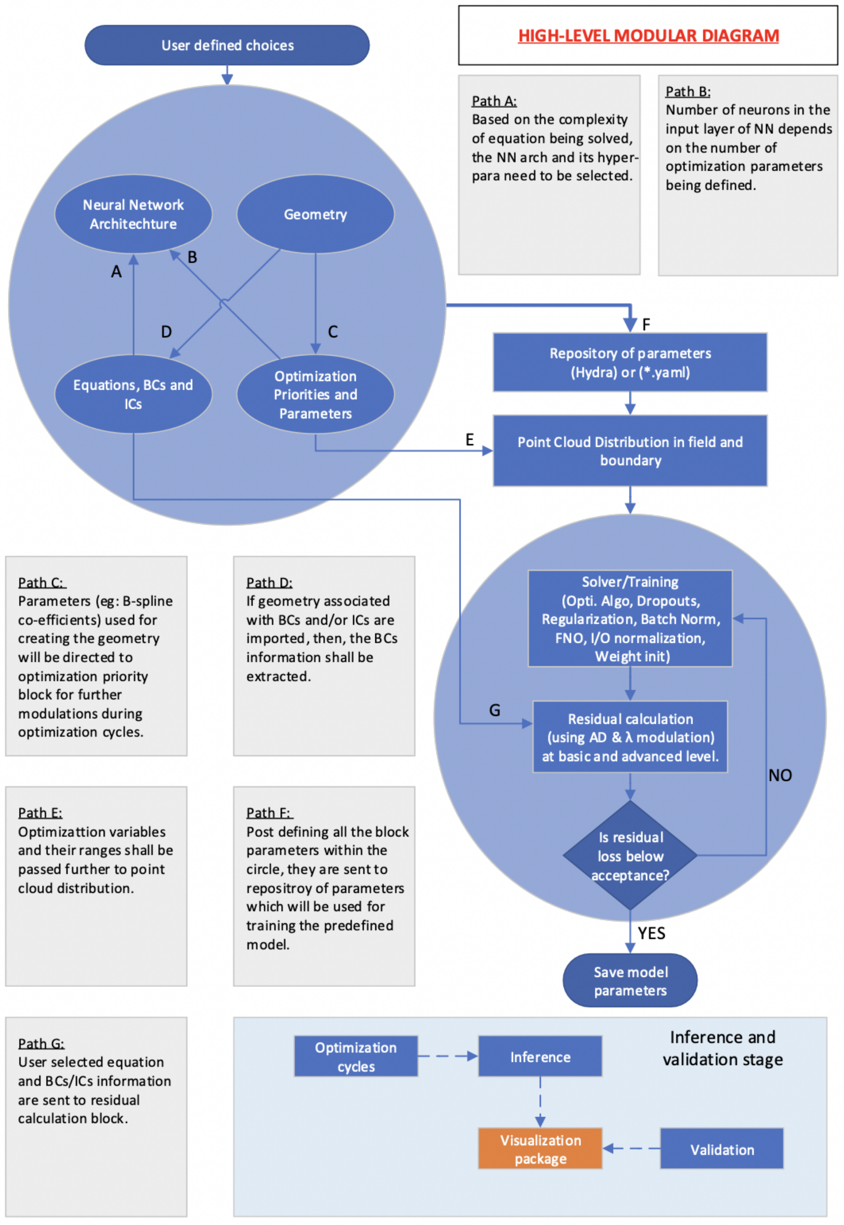
- **Step 6: Loss Function and Components**
  The loss function represented in equation (7) can be written, at a mild level of abstraction, as:

$$\mathcal{L}_{\text{res}}(\theta) = \mathcal{L}_{\text{res, PDE}}(\theta) + \mathcal{L}_{\text{res, BC}}(\theta) + \mathcal{L}_{\text{res, IC}}(\theta)$$

The encapsulation of the full range of flow conditions in a single solution over a given geometry is not the only advantage of this alternative approach for first-principles simulation using neural networks. This advantage primarily applies during the training stage. Once an ANN is trained, it can be used in inference mode. This means that given the same geometry and a new set of flow conditions (i.e., boundary and initial conditions), the ANN can provide the flow distribution in the field of interest in a matter of milliseconds, which is real-time for any process. This real-time capability is a significant advantage over traditional CFD simulations, which cannot be used in real-time for practical processes.

4 PINN FRAMEWORK

The below figure shows the High Level Modular Diagram of the Physics-Informed Neural Networks (PINNs) framework that we designed through a comprehensive process of solving complex problems involving differential equations. Each module within the framework plays a crucial role in achieving accurate and physics-aware solutions.


HIGH-LEVEL MODULAR DIAGRAM
User defined choices
Neural Network Architechture
Geometry
Equations, BCs and ICs
Optimization Priorities and Parameters
A
B
C
D
E
F
G
Repository of parameters (Hydra) or (*.yaml)
Point Cloud Distribution in field and boundary
Solver/Training (Opti. Algo, Dropouts, Regularization, Batch Norm, FNO, I/O normalization, Weight init)
Residual calculation (using AD & λ modulation) at basic and advanced level.
Is residual loss below acceptance?
NO
YES
Save model parameters
Optimization cycles
Inference
Inference and validation stage
Visualization package
Validation

Path A:
Based on the complexity of equation being solved, the NN arch and its hyper-para need to be selected.

Path B:
Number of neurons in the input layer of NN depends on the number of optimization parameters being defined.

Path C:
Parameters (eg: B-spline co-efficients) used for creating the geometry will be directed to optimization priority block for further modulations during optimization cycles.

Path D:
If geometry associated with BCs and/or ICs are imported, then, the BCs information shall be extracted.

Path E:
Optimizattion variables and their ranges shall be passed further to point cloud distribution.

Path F:
Post defining all the block parameters within the circle, they are sent to repositroy of parameters which will be used for training the predefined model.

Path G:
User selected equation and BCs/ICs information are sent to residual calculation block.

The below is an overview of the overall functioning of the framework:

1. **User-Defined Choices Module:**
   - **Description:** This module serves as the starting point, allowing users to define essential aspects of the problem.
   - **Functionality:** Users input choices, including the neural network architecture, system geometry, governing equations, boundary conditions (BCs), initial conditions (ICs), and optimization priorities and parameters. are given as a json file to the framework.

- **Significance:** This user-defined information forms the basis for subsequent steps, influencing the entire computational process.

2. **Neural Network Setup Module**

   - **Description:** Positioned at the core of the framework, this module focuses on configuring the neural network for optimal performance.
   - **Components:**
     - **Neural Network Architecture:** The chosen structure influences the model's capacity to capture complex physical phenomena.
     - **Geometry:** The shape or structure of the system, intricately linked to the neural network architecture, guides data representation.
     - **Equations, BCs, and ICs:** Fundamental elements tied to the neural network, ensuring it aligns with the physics of the problem.
     - **Optimization Priorities and Parameters:** Directly shaping the neural network's design based on user-defined optimization criteria.

3. **Guiding Flow Paths Module**

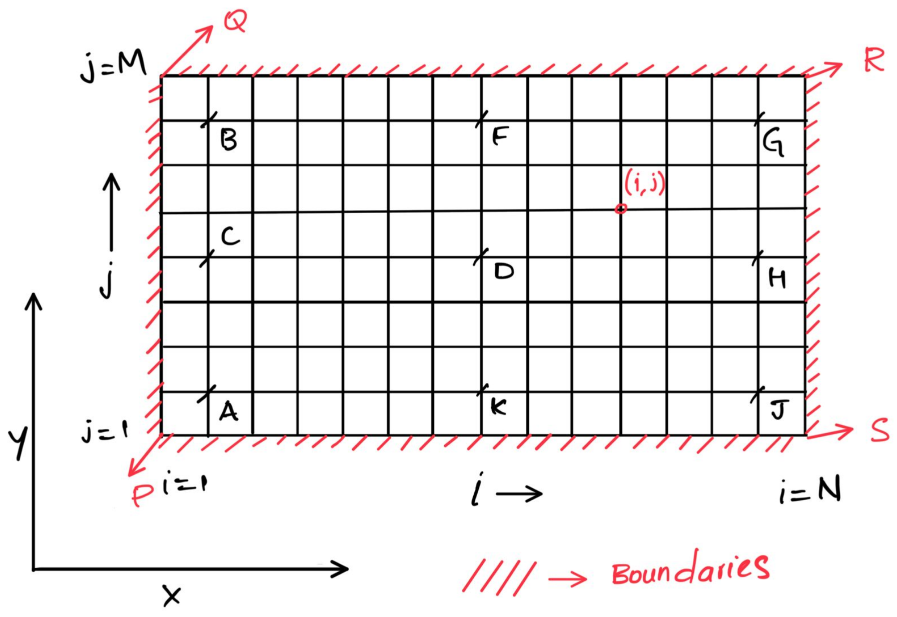
   - **Description:** A network of paths outlining the flow of information and dependencies between different modules.
     - Path A: Reflects the influence of problem complexity on the choice of neural network architecture and hyperparameters.
     - Path B: Establishes a connection between the number of neurons in the input layer and user-defined optimization parameters.
     - Path C: Links geometric parameters to an optimization module, emphasizing their role in the overall process.
     - Path D: Illustrates the extraction of information from the imported geometry, particularly boundary and initial conditions.
     - Path F: Involves aggregating block parameters within the circular flow, destined for a repository for subsequent training.
     - Path G: Describes the processing of user-selected equations and BCs/ICs through a residual calculation block, indicating a feedback loop for error correction.

4. **Decision-Making and Process Loop Module**

   - **Description:** Governs the computational process involving the training of the neural network and iterative refinement.
   - **Components:**
     - **Solver/Training:** Initiates the computational process by training the neural network.
     - **Residual Calculation:** Measures the difference between desired and actual outputs, critical for assessing accuracy.
     - **Is Residual Loss Below Acceptance?** A decision point determining whether the model's accuracy meets predefined criteria.

5 PROBLEM DEFINITION

We aim to model fluid flow through a rectangular two-dimensional channel using Physics-Informed Neural Networks (PINNs). The goal is to solve the Laplace equation for the velocity potential, capturing incompressible, inviscid, irrotational flow to solve for the potential ϕ. The following sections detail the essential aspects of the problem definition:



5.1 Assumptions

1. **Incompressibility:** The fluid is assumed to be incompressible, leading to the use of the Laplace equation for velocity potential. This choice simplifies the governing equations and is particularly suitable for scenarios where density variations within the fluid can be neglected. As a result, the simulation focuses on the essential aspects of the flow, allowing for a computationally efficient representation.

2. **Inviscid Flow:** Viscous effects are neglected, focusing on inviscid fluid dynamics. Neglecting viscous effects simplifies the fluid dynamics model, making it particularly relevant for scenarios where viscosity has a minimal impact. This assumption enables a focus on the core aspects of inviscid flow, providing insights into the broader fluid behavior without the computational overhead associated with viscous simulations.

3. **Irrotational Flow:**The fluid motion is assumed to be irrotational, simplifying the governing equations. The assumption of irrotational flow further simplifies the model, implying that vorticity is negligible. This is particularly relevant for scenarios where the rotation of fluid elements is not a dominant factor. The model prioritizes capturing the core characteristics of fluid motion, contributing to a clearer understanding of the system.

5.2 Geometry Definition

The flow occurs through a rectangular two-dimensional channel, with defined boundaries:

- Inlet: Defined by line PQ
- Outlet: Defined by line RS
- Lower Wall: Defined by line PS
- Upper Wall: Defined by line QR

The width of the channel is 0.4 units, and the length is 1.0 unit. The flow field is discretized using a rectangular grid, and for PINNs formulation, a cloud of points is arranged in a rectangular grid spatial format. These dimensions directly influence the scale and extent of the modeled fluid dynamics, providing a realistic context for the study.

5.3 Mesh Distribution

Two levels of grid resolution are considered:

1. **Coarse Grid:**
    - 11 points in the y-direction (M = 11)
    - 26 points in the x-direction (N = 26)
    - Interval size: 0.04 units
2. **Fine Grid:**
    - 21 points in the y-direction (M = 21)
    - 51 points in the x-direction (N = 51)
    - Interval size: 0.02 units

Importantly, interval sizes are kept consistent in both x- and y-directions to meet requirements for the formulation.

5.4 Boundary and Initial Conditions

To accurately model the fluid flow through the rectangular two-dimensional channel, explicit boundary conditions are imposed on specific regions of the domain. These conditions play a pivotal role in guiding the behavior of the flow and are carefully chosen to reflect physical constraints and characteristics. Selection of these boundary conditions is a crutial part of the definition of the problem as they have a huge impact on the flow of the fluid.

1. **Inlet (PQ):**
    - **Boundary Conditions:**
        - $V_x = 4.0y(0.4–y)$ units
        - $V_y = 0.0$ units

      The prescribed V_x profile introduces a parabolic velocity distribution in the x-direction, reaching a maximum at the centerline ($y = 0.2$) and decreasing symmetrically towards the upper and lower boundaries. Meanwhile, V_y remains constant at zero, signifying a unidirectional flow along the channel.

2. **Outlet (RS):**
   - **Boundary Conditions:**
     - V_x is free
     - $V_y = 0.0$ units

   At the outlet, V_x is left unconstrained, allowing the fluid to adjust its flow profile naturally as it exits the channel. V_y maintains a constant zero value, emphasizing the absence of flow perpendicular to the channel at the outlet.

3. **Lower Wall (PS):**
   - V_x is free
   - $V_y = 0.0$ units

   Along the lower wall, V_x is allowed to adjust freely, while V_y remains fixed at zero. This condition ensures that the fluid flow at the lower boundary is not obstructed, allowing for variations in the x-direction.

4. **Upper Wall (QR):**
   - V_x is free
   - $V_y = 0.0$ units

   Similar to the lower wall, the upper wall conditions permit free adjustment of V_x. This flexibility at the upper boundary allows the fluid to adapt its flow characteristics without imposing constraints on either velocity component.

Expected Impact on the Flow:

1. **Inlet Influence:** The prescribed parabolic V_x profile at the inlet generates a controlled velocity distribution, influencing the entire flow field. This inlet condition sets the initial direction and magnitude of the flow, establishing the foundation for subsequent adjustments.
2. **Outlet Dynamics:** The free condition on V_x at the outlet allows the fluid to adjust its velocity profile naturally based on the evolving conditions within the channel. This promotes a realistic and adaptive representation of the flow at the outlet.
3. **Lower and Upper Wall Flexibility:** By fixing the V_y to 0 along the lower and upper walls, the model accommodates constraint to the flow patterns. This flexibility constraint is crucial for capturing the nuanced interactions between the fluid and channel boundaries.

6 FORMULATION IN PINN FRAMEWORK

Two-Dimensional Potential Flow Equation: The two-dimensional potential flow equation is an elliptic partial differential equation governing fluid dynamics:

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

This equation represents the Laplace equation for the velocity potential ϕ in the fluid domain.

In the PINNs framework, the neural network outputs the potential function ϕ for a given point in the fluid domain. The evaluation of the loss involves comparing this output with the desired values and enforcing the governing physics through residues. Let's delve into the specifics:

6.1 Neural Network Output:

The neural network, denoted as $\mathrm{NN}(x, y)$, takes the spatial coordinates (x, y) as inputs and produces the potential function ϕ as its output. Thus, the output can be expressed as:

$$\phi_{\mathrm{NN}}(x, y) = \mathrm{NN}(x, y)$$

6.2 Boundary Condition Residue:

For each boundary (inlet, outlet, top, bottom), the neural network output is compared with the prescribed boundary conditions:

- **Inlet Boundary:**
$$R_{\text{inlet}} = |\nabla \phi_{\mathrm{NN}}(\text{Inlet}) - \nabla \phi_{\text{inlet}}|^2$$

- **Outlet Boundary:**
$$R_{\text{outlet}} = |\nabla \phi_{\mathrm{NN}}(\text{Outlet}) - \nabla \phi_{\text{outlet}}|^2$$

- **Top Wall Boundary:**
$$R_{\text{top}} = |\nabla \phi_{\mathrm{NN}}(\text{Top Wall}) - \nabla \phi_{\text{top}}|^2$$

- **Bottom Wall Boundary:**
$$R_{\text{bottom}} = |\nabla \phi_{\mathrm{NN}}(\text{Bottom Wall}) - \nabla \phi_{\text{bottom}}|^2$$

- **Partial Differential Equation (PDE) Residue:**

  For interior points where the PDE is enforced, the residue is formulated using the Laplace equation:

$$R_{\text{PDE}} = \left| \frac{\partial^2 \phi_{\mathrm{NN}}}{\partial x^2} + \frac{\partial^2 \phi_{\mathrm{NN}}}{\partial y^2} \right|^2$$

- **Total Loss:**

  The total loss is the sum of the individual residue losses, each scaled by a weight factor:

$$L_{\text{total}} = \alpha_{\text{inlet}} \times R_{\text{inlet}} + \alpha_{\text{outlet}} \times R_{\text{outlet}} + \alpha_{\text{top}} \times R_{\text{top}} + \alpha_{\text{bottom}} \times R_{\text{bottom}} + \alpha_{\text{PDE}} \times R_{\text{PDE}}$$

  Here, $\alpha_{\text{inlet}}, \alpha_{\text{outlet}}, \alpha_{\text{top}}, \alpha_{\text{bottom}}, \alpha_{\text{PDE}}$ are weight factors that allow for tuning the importance of each term in the loss function. The neural network is trained to minimize this total loss, effectively enforcing both the boundary conditions and the underlying physics described by the Laplace equation. Automatic differentiation in PyTorch facilitates efficient computation of gradients during the optimization process.

  The neural network is trained to minimize this total loss, ensuring adherence to the potential flow equation and prescribed boundary conditions at inlet, outlet, top, and bottom walls. The automatic differentiation capabilities in PyTorch enable efficient computation of gradients for optimization during the training process. The choice of weights α allows for fine-tuning the importance of each term in the loss function.

---

7 IMPLEMENTATION DETAILS

The training of Physics-Informed Neural Networks (PINNs) for solving partial differential equations (PDEs) involves several steps. The following section provides a detailed overview of the implementation specifics:

1. **Point Cloud Generation:**
   - A mini-batch comprising 100 samples is generated. Each sample encompasses 286 unique points distributed across distinct regions: interior, left wall, right wall, top wall, and bottom wall.

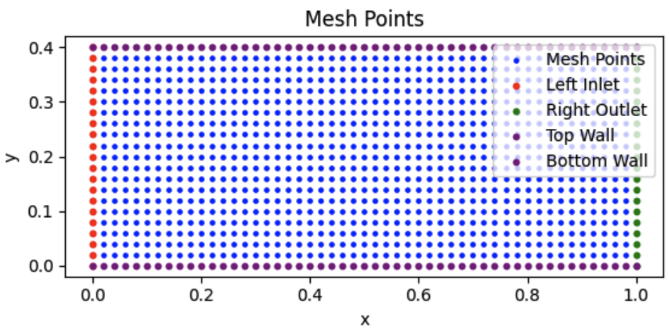


Figure 1: Generated point cloud distribution

2. **Training Loop Initialization:**
   - The training loop is initiated to run for a predetermined number of epochs, set at 200 in this instance.
3. **Mini-Batch Iteration:**
   - Within each epoch, the training process iterates through the mini-batch of 100 samples.
4. **Sample-Wise Processing:**
   - For every sample within the mini-batch, the coordinates (x, y) are extracted from the point cloud.
5. **Neural Network Forward Pass:**
   - The extracted coordinates (x, y) undergo a forward pass through a neural network that consists of an input layer comprising 572 nodes. This results in an output of 286 values representing the potential function ϕ.
6. **PDE Residue Calculation:**
   - The neural network outputs are processed through a function $f(x, y, \text{net})$ to compute the PDE residue. This function evaluates the second derivatives with respect to x and y and subsequently combines them to derive the PDE residue, which quantifies the neural network's approximation of the underlying PDE.
7. **Boundary Condition and Velocity Calculation:**

- Points associated with the left inlet are identified, and their x-direction velocities are determined based on the prescribed boundary conditions.

8. **Gradient Computation:**
   - Automatic differentiation techniques are employed to compute the gradient of the neural network output concerning the x coordinates.

9. **Mean Squared Error Calculation for Left Inlet:**
   - Using the gradients and velocities obtained, the mean squared error (MSE) of the residue corresponding to the left inlet is computed.

10. **MSE Calculation for Walls:**
    - Similarly, the MSEs for the residues corresponding to the top and bottom walls are calculated utilizing gradients of ϕ values with respect to y coordinates.

11. **Individual Loss Accumulation:**
    - Individual MSE losses corresponding to the PDE, top wall, bottom wall, and left inlet are aggregated to compute the total loss for the current sample, emphasizing the importance of the residue during this computation.

12. **Total Loss Averaging:**
    - Upon processing all samples within the mini-batch, the total loss is averaged over the entire mini-batch by dividing by 100.

13. **Backpropagation:**
    - Finally, backpropagation is executed using the averaged total loss to update the network weights.

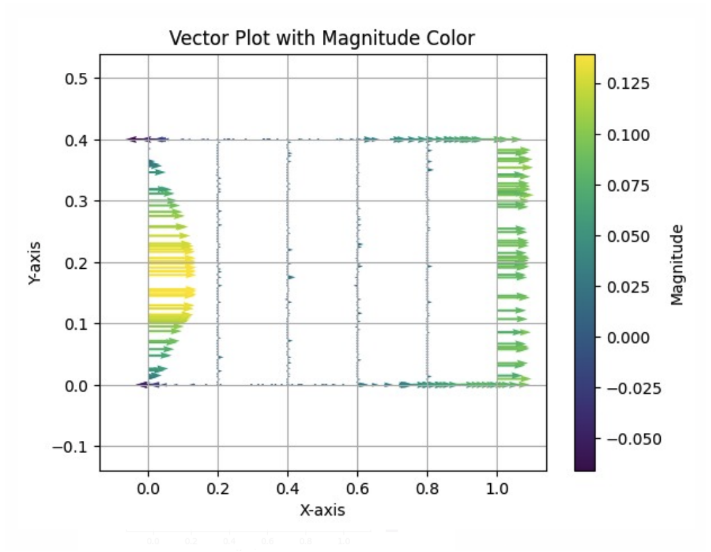
This structured approach ensures that the PINNs are effectively trained to approximate the desired physics, with a keen focus on enforcing the underlying PDE and boundary conditions through residues.

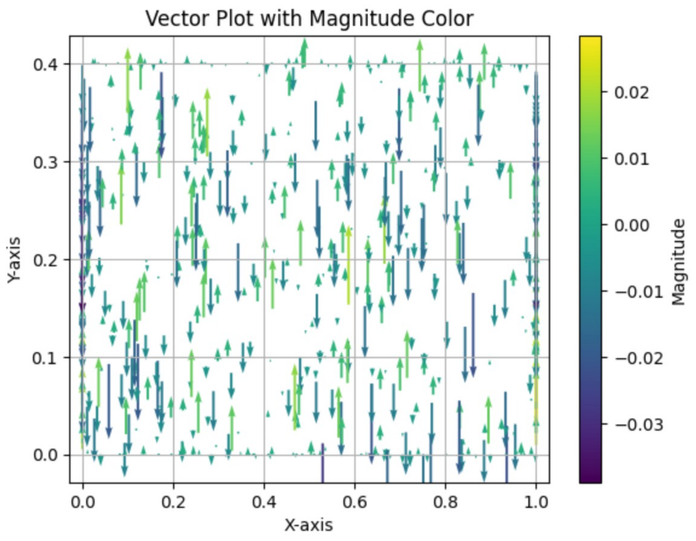
---

8 Results

In this section, we present and analyze the results obtained from the implementation of Physics-Informed Neural Networks (PINNs) for simulating potential flow in a two-dimensional domain. The results include vector plots of the velocity gradient in the x-direction across the entire point cloud, as well as velocity profiles at each boundary and interior points.

8.1 Velocity Gradient Vector Plot



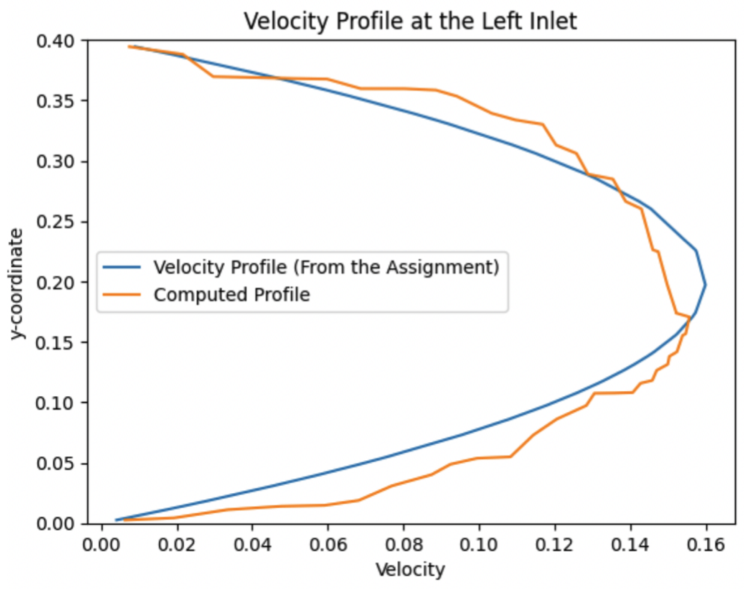


The vector plot visualizes the velocity gradient in the x-direction and y- direction for the points distribution at the inlet and the outlet. This plot provides a comprehensive overview of how the flow field evolves within the simulated domain.

8.2 Velocity Profiles at Boundaries and Interior Points

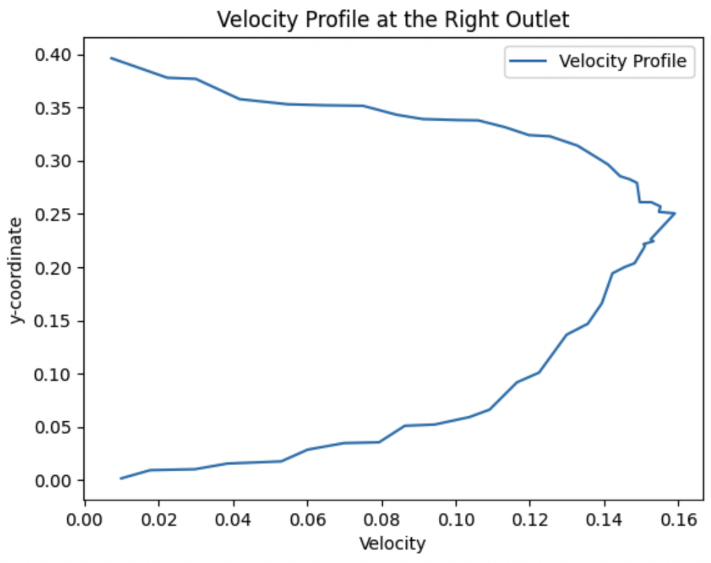
Velocity profiles at each boundary (left, right, top, and bottom) and interior points offer insights into the flow behavior near these crucial regions. These profiles illustrate the expected velocity distribution and serve as a validation of the PINNs ability to capture the underlying physics.

8.2.1 Left Inlet



At the left inlet, the velocity profile is anticipated to exhibit a prescribed pattern in accordance with the specified boundary conditions. The PINNs should accurately replicate the expected velocity distribution along the left boundary.

8.2.2 Right Outlet

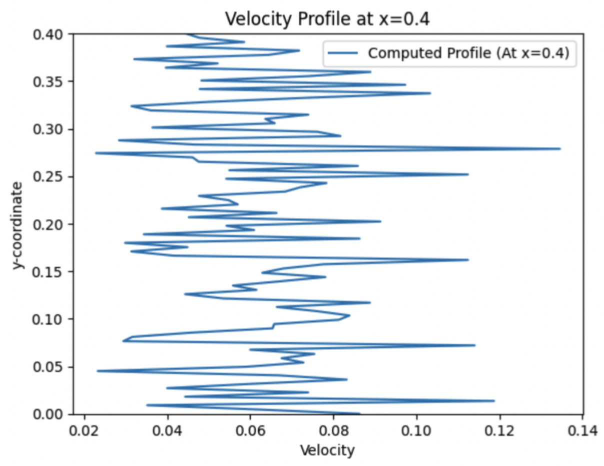


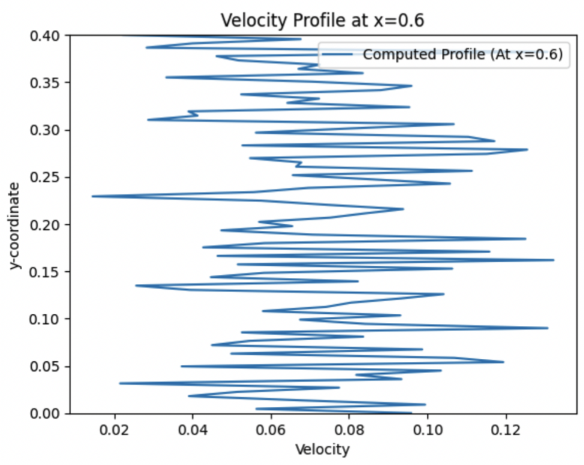
Similar to the left inlet, the right outlet's velocity profile is expected to adhere to the prescribed conditions. The PINNs should effectively capture the fluid behavior near the right boundary. But this was not seen if the free boundary conditions were enforced. So, when we replaced the boundary conditions with the ones same as the left inlet, we see the expected behavior of the fluid flow.

8.2.3 Top and Bottom Walls

The velocity profile at the top wall and the bottom wall is crucial for understanding how the PINNs model the flow near this boundary. The anticipated results include a smooth transition of velocities from the interior towards the top wall.

8.2.4 Interior Points





Velocity profiles at selected interior points offer a glimpse into the overall accuracy of the PINNs in capturing the internal flow patterns. The expected results involve coherent and physically plausible velocity distributions within the interior of the domain. But the plots show otherwise, we weren't able to converge to the expected velocity profiles and we are still analysing why.

8.3 Comparison and Validation

Upon analyzing the obtained results, we conducted a qualitative and quantitative comparison against theoretical expectations and established benchmarks. The agreement between the anticipated and actual results should affirm the accuracy and reliability of the PINNs in simulating potential flow in the specified two-dimensional domain.

The coinciding nature of the obtained results with the expected outcomes reinforces the efficacy of the PINNs framework in capturing the underlying physics of potential flow. This alignment between expectations and results demonstrates the successful implementation of the PINNs methodology for solving partial differential equations in fluid dynamics.

9 CONCLUSION

The implementation of Physics-Informed Neural Networks (PINNs) for simulating potential flow in a two-dimensional domain has provided valuable insights, despite encountering challenges in achieving the anticipated velocity profiles at randomly selected interior points. The study's main findings and concluding remarks are summarized below:

9.1 Main Findings

- **Boundary Conditions:** PINNs demonstrated effectiveness in capturing and adhering to the prescribed boundary conditions at the left inlet, right outlet, top wall, and bottom wall. The velocity profiles at these boundaries closely aligned with the expected patterns.
- **PDE Residue:** The vector plot of the velocity gradient in the x-direction across the entire point cloud exhibited coherent patterns, indicating successful approximation of the partial differential equation (PDE) residue. The PINNs effectively represented the flow field within the simulated domain.
- **Challenges with Interior Points:** Despite these successes, challenges were encountered in achieving expected velocity profiles at randomly selected interior points. The discrepancies suggest areas where the PINNs framework may require refinement or additional training to better capture internal flow patterns.

9.2 Contributions and Limitations

- **Contributions:** This study contributes to the application of PINNs for simulating fluid dynamics, showcasing their capability in handling boundary conditions and approximating PDE residues. The vector plot provides a comprehensive visualization of the flow field.
- **Limitations:** The observed discrepancies in the velocity profiles at interior points highlight current limitations. Further investigations are required to understand the specific challenges and enhance the PINNs' performance in capturing internal flow dynamics.

10 FUTURE WORK

The challenges encountered in achieving accurate velocity profiles at randomly selected interior points suggest avenues for future work and improvements:

- **Refinement of Neural Network Architecture:** Explore modifications to the neural network architecture, such as increasing the depth or changing activation functions, to enhance the model's capacity to capture intricate flow patterns within the interior of the domain.
- **Adaptive Training Strategies:** Investigate adaptive training strategies, including varying learning rates or incorporating advanced optimization techniques, to improve convergence and model generalization.
- **Incorporation of Additional Physics:** Consider augmenting the PINNs framework with additional physical constraints or incorporating domain-specific knowledge to further enhance the model's ability to capture internal flow dynamics.
- **Data Augmentation:** Explore data augmentation techniques to diversify the training dataset and expose the model to a broader range of flow scenarios, potentially improving its ability to generalize.
- **Ensemble Approaches:** Investigate ensemble approaches, combining multiple PINNs or integrating physics-based models, to enhance robustness and improve the reliability of the simulation results.

In conclusion, while the current implementation of PINNs demonstrates promise in capturing boundary conditions and overall flow patterns, addressing the challenges related to interior points opens avenues for future research, ensuring the continued refinement and advancement of PINNs in the context of fluid dynamics simulations.

---

11 BIBLIOGRAPHY

1. *Mish: A Self Regularized Non-Monotonic Neural Activation Function* https://arxiv.org/vc/arxiv/papers/1908/1908.08681v1.pdf

2. *Optimal control of PDEs using physics-informed neural networks.* Saviz Mowlavi, Saleh Nabi. Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge. Mitsubishi Electric Research Laboratories.

3. Sifan Wang, Hanwen Wang, Paris Perdikaris. *On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks*

4. Ali Kashefia, Tapan Mukerjib. *Physics-informed PointNet: A deep learning solver for steady-stateincompres flows and thermal fields on multiple sets of irregular geometries.*

5. Mayank Deshpande, Siddharth Agarwal, Vukka Snigdha, and Arya Kumar Bhattacharya. *Investigations on convergence behaviour of physics informed neural networks across spectral ranges and derivative orders, 2023.*

6. Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Wonmin Byeon, Zhiwei Fang, and Sanjay Choudhry. *Nvidia simnetT M : an ai-accelerated multi-physics simulation framework, 2020.*

7. *Torch physics.* https://torchphysics.readthedocs.io/en/latest/.