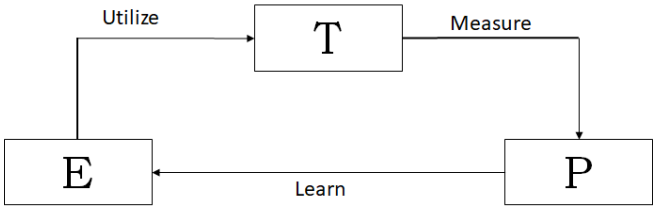
INTRODUCTION

Over time, a range of techniques has been developed to tackle the complexities of solving differential equations. These methods differ in how they deliver the solution. Some produce an array that contains the solution values at specific chosen points, while others opt for an analytic representation using basis functions, typically leading to the transformation of the original problem into a system of linear equations.

Machine learning (ML) plays an important role in solving differential equations which govern in the form of physical systems. Machine Learning is a type of artificial intelligence (AI) that allows software applications to become more accurate in predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

As per Tom M. Mitchell, he defines ML as –



"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience [1]" as mentioned in **Figure No. 1.**

Machine learning has become a powerful tool that is used in a wide variety of applications today.

Machine Learning is categorised into three different types [2][3] –

1. **Supervised learning: In learning the algorithm undergoes training using a dataset containing labelled examples. Each example in the dataset consists of both an input and an output. The objective is to acquire knowledge about a function that can establish a connection between input values and output values for instances.**

2. **Unsupervised learning: In learning the algorithm is trained on a dataset consisting of unlabelled examples. Each example only has an input value without any associated output value. The aim here is to uncover patterns within the data without any knowledge or guidance regarding the expected output values.**

3. **Reinforcement learning: In reinforcement learning the algorithm learns how to interact with its environment through trial and error. It receives rewards, for actions that produce desired outcomes and faces penalties for actions that lead to outcomes.**

The Artificial Neural Network (ANN) –

The concept of artificial neurons initially took shape in 1943. Artificial neural networks serve as computational systems capable of learning tasks through exposure to examples. The field of artificial neural networks boasts a rich history, closely intertwined with the progress of computer science and computing in general. The Mathematical model for artificial neurons is given as [4] –

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) = f(u)$$

Here, the output y of the neuron is the value of its activation function, which has as input a weighted sum of signals $x_i, \ldots, x_n$ received by n other neurons.

In this scenario, the neuron's output is determined by the value generated from its activation function. This value is obtained by processing a weighted sum of signals that are received from other neurons.

In terms of conceptual organization, it's quite useful to categorize neural networks into four distinct groups:

1. General-purpose neural networks are designed to excel in supervised learning tasks.

2. Neural networks with a specialized focus on image processing. A prime example of this category is the Convolutional Neural Networks (CNNs).

3. Neural networks are tailored for handling sequential data, as seen in the case of Recurrent Neural Networks (RNNs).

4. Neural networks intended for unsupervised learning, including models like Deep Boltzmann Machines.

Multilayer Perceptron (MLP)

A commonly employed architecture in neural networks is the fully connected feed-forward neural network, which typically features three or more layers: an input layer, one or more hidden layers, and an output layer. Neurons within these layers are equipped with non-linear activation functions, leading to the popular nomenclature of these networks as multilayer perceptrons (MLPs). The Universal Approximation Theorem underlines the remarkable potential of a feed-forward neural network with a sole hidden layer comprising a finite number of neurons to approximate continuous multidimensional functions with arbitrary precision. However, a crucial condition is that the activation function employed in the hidden layer must be non-constant, bounded, monotonically increasing and continuous. It's worth noting that these requirements solely pertain to the hidden layer, as the output nodes are consistently assumed to exhibit linearity, ensuring an unrestricted range of output values.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b_i\right) = f(z)$$

Where the output y is produced via the activation function f.

Physics-informed neural networks (PINNs)

In recent years, deep learning (DL), i.e., multilayer perceptron that is fully connected artificial neural network (ANN), has attracted great attention in computational mechanics and provided alternative ways for solving mechanics problems. ANNs can be equipped with multiple hidden layers with neurons, providing them with powerful learning capability.

Physics-informed neural networks (PINNs) represent a remarkable category of machine learning models devised to tackle complex partial differential equations

(PDEs). PDEs serve as mathematical descriptions of how physical systems evolve over both time and space, finding extensive applications across a multitude of scientific and engineering domains, ranging from fluid dynamics and heat transfer to structural mechanics.

PINNs distinguish themselves by synergizing the formidable capabilities of deep learning with the intrinsic understanding of the underlying PDEs. In the training process, a neural network is tasked with minimizing a composite loss function composed of two key elements: a data term and a physics term. The data term quantifies the disparities between the predicted solution and the observed data, while the physics term gauges the deviations between the predicted solution and the PDE it is intended to satisfy.

The advantages of PINNs over conventional numerical techniques for PDE resolution are manifold. Firstly, PINNs are adaptable for solving PDEs involving intricate geometries and diverse boundary conditions. Secondly, they can effectively tackle PDEs even when faced with noisy or incomplete data. Lastly, PINNs exhibit real-time applicability in solving PDEs [10].

In the realm of computational mechanics, PINNs have been deployed extensively across various scenarios, such as:

1. Fluid Dynamics: Employing PINNs to tackle the Navier-Stokes equations, which provide insight into fluid flow dynamics.

2. Heat Transfer: Utilizing PINNs to address the heat equation, which characterizes the diffusion of heat within materials.

3. Structural Mechanics: Applying PINNs to resolve the elasticity equations, which elucidate the deformation behaviour of solid structures.

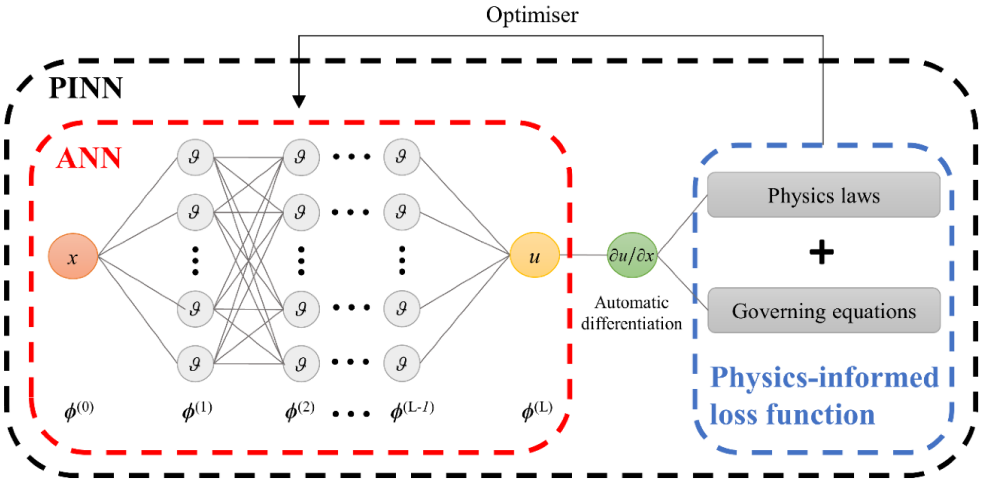


Figure 2: A physics-informed neural network comprises two main components, i.e., artificial neural network (ANN) and physics-informed loss function. An example of a $\mathcal{L}$-layer ANN is shown in the red dash box. x and u denote the input and output of the ANN, respectively. ϑ denotes the activation function used in ANN. The physics-informed loss function is shown in the blue dash box. The physics-informed loss function is formulated by the physics laws and the governing equations of the investigating systems. Note that partial differential terms, which are widely seen in physics laws and governing equations, are analytically obtained via automatic differentiation [11].

* * *

LITERATURE REVIEW

As long as there is a sufficient amount of quality data or some form of information that can be used to train an ANN properly, it can become a powerful tool for a wide range of problems, including image processing, biological prediction, and non-smooth dynamics. The quality data is, however, often difficult to acquire. A fully trained ANN can also provide a powerful tool to unveil the relationships between field variables of mechanics problems. Based on the training data, optimisers can iteratively improve the performance of ANNs by tuning the parameters. Many neural network-based computational mechanics have been proposed which is mentioned in the reference research paper [11].

Recently, the physics-informed neural network (PINN) has been proposed by Raissi et al. [10], in which the physics laws and equations can be seamlessly integrated into training ANNs. In PINN, physics laws are transferred into training data. Compared to data-driven DL, PINN leverages physics laws as the remedy for insufficient data. In this manner, PINN can achieve better performance than data-driven DL when facing data scarcity. Up to now, a great number of studies have been conducted for PINN.

Compared to traditional computational mechanics methods, PINN-based computational mechanics has the following advantages [12][13][11]:

- Physics-informed neural network provides a powerful tool for solving nonlinear systems. In this manner, PINN-based computational mechanics can easily deal with mechanics problems with nonlinearity, such as large deformation and material nonlinear problems. Those problems are considered to be challenging for traditional computational mechanics methods.
- Partial differential terms can be analytically obtained through automatic differentiation instead of spatial discretization schemes and approximation methods in traditional computational mechanics. In this manner, PINN-based computational mechanics frameworks are less likely to be affected by the mesh quality and the distortion problems in mesh-based, or the arbitrary particle distribution and the boundary truncation issues in meshfree methods.

- Physics-informed neural network-based computational mechanics has great potential for solving inverse problems.
- Physics-informed neural network-based computational mechanics is easy to implement.

* * *

RESEARCH METHODOLOGY

- ## Machine Learning Today

In the 2000s, support vector machines (SVM) and decision trees were developed, which were able to handle larger datasets and were more computationally efficient. The rise of big data and the development of deep learning (DL) led to a wide range of applications for machine learning (ML) in image recognition, natural language processing (NLP), self-driving cars, and personalized medicine. Examples of successful ML applications include DeepFace (DL-based facial recognition), AlphaGo (AI player of the board game Go that defeated a world champion), and ChatGPT (AI chatbot).

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun are known as the Three Godfathers of AI for their pioneering contributions to the development of AI. Overall, machine learning has become a powerful tool that is used in a wide variety of applications today [28].

- ## The Artificial Neural Network (ANN):

Artificial neural networks serve as computational systems capable of learning tasks through exposure to examples, typically devoid of explicit task-specific instructions. They aim to replicate the functions of biological systems where neurons interact by transmitting mathematical signals between various layers. Each layer can house a variable number of neurons, and connections are represented by weight variables. The field of artificial neural networks boasts a rich history, closely intertwined with the progress of computer science and computing in general. The concept of artificial neurons initially took shape in 1943, when McCulloch and Pitts developed a model to investigate signal processing in the brain, subsequently refined by other researchers. The fundamental idea behind this approach is to emulate the neural networks within the human brain, which comprises billions of neurons communicating through electrical signals. In this process, each neuron accumulates incoming signals, requiring the total to surpass an activation threshold to produce an output. If the threshold isn't met, the

neuron remains inactive, yielding zero output. This fundamental behaviour has inspired a straightforward mathematical model for artificial neurons [4].

The artificial neural network stands as a robust bionic computing system, drawing inspiration from the workings of the biological brain [5]. It's composed of multiple layers of artificial neurons, as illustrated in Figure 1. To clarify, the initial layer serves as the input layer, while the ultimate layer functions as the output layer. Within an ANN, the neighbouring neural network layers establish connections through the use of weights, biases, and activation functions [6]. When employing an ANN, input data is introduced into the network through the input layer, subsequently passing through the successive layers. Ultimately, the ANN provides its prediction through the output layer. The mathematical representation of an L-layer ANN is precisely articulated [7].

$$\begin{aligned}
&\phi^{(0)} = x; \\
&\tilde{\phi}^{(l)} = w^{(l)}\phi^{(l)} + b^{(l)}; \\
&\phi^{(l+1)} = \vartheta\left(\tilde{\phi}^{(l)}\right); \\
&u = \phi^{(L)} = w^{(L-1)}\phi^{(L-1)} + b^{(L-1)}
\end{aligned}$$

Where $\mathcal{W}^{(\ell)}$ and $b^{(\ell)}$ are the ℓ^{th} layer's weights and biases in the ANN, respectively, ϑ denotes the activation function, which provides nonlinear features to the neural network [6]

With reference to the universal approximation theorem [8], it is worth noting that an artificial neural network (ANN) holds the capacity to approximate a wide range of Borel measurable functions through the manipulation of weights and biases [9]. This inherent capability has led to the extensive utilization of ANNs in the examination and capture of the inherent associations between input and output variables.

- ## Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs)

Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs) have discovered a myriad of applications in the realm of natural science. Their utility extends to diverse fields, such as statistical physics, where they've been instrumental in detecting phase transitions in 2D Ising and Potts models, lattice gauge theories, polymer phases, and even solving the complex Navier-Stokes equation for weather forecasting. The application of deep learning is equally fascinating in quantum physics, where DNNs and CNNs are employed to detect and explore various quantum phase transitions, topological phases, and non-equilibrium many-body localization.

Furthermore, the representation of quantum states through DNNs, as seen in quantum state tomography, underscores the remarkable potential of DNNs in unravelling the intricacies of quantum systems. In the domain of quantum information theory, it has been demonstrated that neural networks can aid in gate decompositions. These applications extend far beyond the natural sciences, permeating virtually every discipline, from the humanities to life sciences and medicine. The Artificial Neural Network (ANN), a computational model comprising interconnected layers of nodes or units, seeks to emulate the functioning of a biological nervous system. Neurons, units, and nodes are terms used interchangeably to describe these components, which interact by transmitting signals in the form of mathematical functions across layers. ANNs come in various forms, but most encompass an input layer, an output layer, and intermediary layers referred to as hidden layers. Each layer can house a variable number of nodes and connections between nodes are characterized by weight variables [4].

The different types of neural networks are mentioned below [4]:

- **Feed-forward neural networks**

The feed-forward neural network (FFNN) stands as the initial and most straightforward form of artificial neural networks (ANNs) to be conceptualized. Within this network, information flows solely in one direction: forward through the layers. The nodes are depicted as circles, and the arrows represent the interconnections between these nodes, also indicating the direction of information propagation. Notably, each arrow is associated with a weight variable (figure forthcoming). It is worth noting that every node in a given layer is linked to all nodes in the subsequent layer, giving rise to what is commonly referred to as a fully connected FFNN.

- **Convolutional Neural Network**

Convolutional neural networks (CNNs) represent a distinct variant of FFNNs, taking inspiration from the connectivity pattern observed in the visual cortex of animals. In the animal visual cortex, individual neurons exclusively respond to stimuli within specific small sub-regions of the visual field, known as receptive fields. This design equips neurons to effectively exploit the significant spatial correlation inherent in natural images. Mathematically, the response of each neuron can be closely approximated through a convolution operation (figure forthcoming).

CNNs, in their attempt to mimic the behaviour of visual cortex neurons, enforce a local connectivity pattern among nodes in adjacent layers. Unlike the fully connected FFNN, each node in a convolutional layer establishes connections only with a subset of nodes in the preceding layer. CNNs frequently incorporate multiple convolutional layers responsible for learning local features from the input data, culminating in a fully connected layer that consolidates these local details to generate the network's outputs. These networks find extensive utility in tasks related to image and video recognition.

- **Recurrent neural networks**

Up to this point, our discussion has centred on ANNs characterized by unidirectional information flow, specifically moving forward. In contrast, recurrent neural networks (RNNs) introduce a unique feature in the form of connections between nodes that give rise to directed cycles. This mechanism establishes an internal memory that enables the network to retain information about prior calculations; thus, the output is inherently linked to previous computational steps. RNNs effectively harness sequential data by executing the same task for each element within a sequence, with each element's outcome being influenced by its predecessors. A notable application of RNNs is in tasks that involve sequential data, such as sentences, rendering them particularly well-suited for handwriting and speech recognition.

- **Other different types of networks**

A wide array of other ANN variants has been developed, each tailored to serve distinct purposes. Among these, the radial basis function (RBF) network stands out as a specialized model designed for multidimensional space interpolation. RBF networks conventionally consist of three essential layers: an input layer, a hidden layer featuring non-linear radial symmetric activation functions, and a linear output layer (where 'linear' denotes the use of linear activation functions for each node in the output layer). The connections between layers adhere to a full connectivity pattern, and cyclic connections are notably absent. Consequently, RBF networks can be regarded as a subcategory of fully connected FFNNs. Nevertheless, they are often treated as a distinct neural network type owing to their unique activation functions.

- ## Multilayer Perceptrons (MLP)

In the realm of deep learning, the multilayer perceptron (MLP) is a widely embraced and straightforward approach. It encompasses the following characteristics:

1. A neural network featuring one or more layers of nodes positioned between the input and output nodes.
2. The structural framework of the multilayer network includes an input layer, one or more hidden layers, and an output layer.
3. The journey of information within the network starts with the input nodes, which transmit data to the initial hidden layer. These nodes, in turn, relay the information to subsequent layers until the ultimate destination, the output layer, is reached.

   Conventionally, a network comprising one layer of input units, one layer of hidden units, and one layer of output units is referred to as a two-layer network. Correspondingly, a network with two layers of hidden units is termed a three-layer network, and so on. It's important to note that in an MLP network, there are no direct connections between the output nodes (or neurons/units) and the input nodes (or neurons/units). Throughout our discussion, we will use the term "nodes" to denote the various entities within a layer. Additionally, there are no connections established solely within a single layer. It's worth emphasizing that the number of input nodes need not match the number of output nodes, and this flexibility extends to the hidden layers as well. Each layer can host its unique set of nodes and activation functions.

- ## Physics-informed loss function:

In deep learning, we evaluate the performance of artificial neural networks (ANNs) based on a loss function that considers the current settings of weights and biases. Essentially, this function quantifies the disparities between the output of ANNs and ground truth data, which could be experimental observations or other relevant data sources. To enhance the ANNs' performance, training algorithms adjust the weights and biases based on the current loss.

Physics-Informed Neural Networks (PINNs) take a different approach. Instead of using a conventional loss function, they employ a physics-informed loss function. As the name suggests, this type of loss function is formulated based on the principles of physics. It effectively dictates how variables change within the systems under investigation. By utilizing this physics-informed loss function, PINNs gain additional insights from the governing physical laws during training. Consequently, PINNs can achieve superior accuracy compared to ANNs solely trained on ground truth data. Moreover, ANNs make use of specific activation functions like tanh, sigmoid, and mish. These functions create differentiable mathematical mappings from input to output. This enables automatic differentiation to be applied for extracting partial differential terms from the governing equations. In contrast, traditional computational mechanics relies on numerical schemes to approximate these terms, which can be less accurate. Traditional computational mechanics also depends on the availability of sufficient surrounding information for accurate approximations. In cases where such information is lacking, like when dealing with uneven particle distributions or near-boundary conditions, the accuracy of the approximation is severely affected. Neural network mappings combined with automatic differentiation effectively address these issues [11].

In the realm of PINN-based computational mechanics, two primary types of physics-informed loss functions are employed: the collocation loss function and the energy-based loss function. These loss functions play a crucial role in guiding the network to learn and adhere to the governing physical laws [11].

- ## Google Colaboratory (Colab):

The outcome of the paper [22] is that Google Colaboratory (Colab) can effectively be used to accelerate not only deep learning applications but also other GPU-centric applications. The performance of Colab's GPU is comparable to dedicated hardware, given similar resources. However, the free-of-charge hardware resources provided by Colab are not enough to solve demanding real-world problems and are not scalable. The lack of CPU cores is identified as a significant limitation of Colab. Overall, Colab's hardware resources may be sufficient for several profiles of researchers and students, but it is not suitable for solving complex real-world problems [23] [24].

The paper [22] also presents a detailed analysis of Colaboratory's hardware resources, performance, and limitations. The analysis is conducted by using Colaboratory to accelerate deep learning for computer vision and other GPU-centric applications. The chosen test cases include a parallel tree-based combinatorial search, object detection/classification, and object localization/segmentation. The results show that the performance achieved using Colaboratory is equivalent to dedicated testbeds, given similar resources. The paper also discusses the strengths and limitations of Colaboratory as a cloud service, which can be helpful for potential users [23] [24]. Colaboratory's accelerated runtime for processing deep learning applications and GPU-centric combinatorial search is faster than using 20 physical CPU cores. It is worth using Colaboratory instead of a robust server with no GPU, a laptop, or a workstation with a mid-end GPU. The performance of the GPU provided by Colaboratory is sufficient for several profiles of researchers and students. However, the free-of-charge hardware resources are not enough to solve demanding real-world problems and are not scalable [24] [25] Also, the performance of Colaboratory's GPU to a mainstream workstation and a robust Linux server equipped with 20 physical cores. The results show that Colaboratory's GPU is superior to the mainstream device and reaches 93% of the performance of the most powerful GPU (Tesla K40). Training a CNN on Colaboratory's accelerated runtime is on average 2.93x faster than using all physical cores of the Linux server. However, the mainstream workstation is faster than Colaboratory for training the CNN, with an average speedup of 3.6x compared to the Linux server [26] [27].

Overall the paper [22] concludes that investing in GPUs for accelerating deep learning applications is worthwhile, and Colaboratory can be a valuable tool for researchers and students in the field of machine learning and computer vision. However, it is important to consider the limitations of Colaboratory's hardware resources, such as the lack of CPU cores, when tackling complex real-world problems [23] [24].

* * *

IMPLEMENTATION

Here we have implemented the three different tools for solving differential equations viz. Deep Neural Network, Analytical Method and Numerical method in Google Colab for comparing the outputs. The generalized explanation of the study is mentioned in the Google Colab Notebook. **Here the following code demonstrates the different computation work on PINNs using Google Colab [29][30][31][32]**

Physics-Informed Neural Networks (PINNs):A Powerful Tool for Solving ODEs & PDEs

-By Amogh Lanjewar

Under the Supervision of Dr PM Gade, Professor, P.G.T.D of Physics, RTM Nagpur University

Solving Differential Equations with Deep Learning

The Universal Approximation Theorem states that a neural network can approximate any function at a single hidden layer along with one input and output layer to any given precision.

An ordinary differential equation (ODE) is an equation involving functions having one variable.

In general, an ordinary differential equation looks like

$$f\left(x,\, g(x),\, g'(x),\, g''(x),\, \dots\, ,\, g^{(n)}(x)\right) = 0 \tag{1}$$

where $g(x)$ is the function to find, and $g^{(n)}(x)$ is the n-th derivative of $g(x)$.

The $f\left(x, g(x), g'(x), g''(x),\, \dots\, , g^{(n)}(x)\right)$ is just a way to write that there is an expression involving x and $g(x),\ g'(x),\ g''(x),\ \dots\ ,$ and $g^{(n)}(x)$ on the left side of the equality sign in (1). The highest order of derivative, that is the value of n, determines to the order of the equation. The equation is referred to as a n-th order ODE. Along with (1), some additional conditions of the function $g(x)$ are typically given for the solution to be unique.

Let the trial solution $g_t(x)$ be

$$g_t(x) = h_1(x) + h_2(x, N(x, P)) \tag{2}$$

FINDINGS

After computing the output of three different methods i.e., Deep Neural Network (DNN), Numerical Method (NM) and Analytical Method (AM), the following findings are noted –

1. Through Google Colab, we have tried to compute the maximum output difference between the Analytical method with the Numerical method and the Deep Neural Network (DNN).
2. In overall findings, we have got the least maximum output difference in the Analytical method with the Deep Neural Network (DNN) compared to the Numerical method.
3. It is to be noted that Machine learning DNNs, numerical methods, and analytical methods are all powerful tools for solving problems. The best approach is to use depends on the specific problem and the available resources [33].
4. When to use each approach [33]
   - **Machine learning DNNs:** When numerical methods are too computationally expensive or difficult to implement, or when analytical solutions are not available or are not accurate enough.
   - **Numerical methods:** When accuracy is critical and computational resources are available.
   - **Analytical methods:** When accuracy is critical and numerical methods are too computationally expensive or difficult to implement, and when the problem is simple enough to be solved using an analytical formula.

The following Advantages of PINNs are noted from different literature. PINNs have several advantages over traditional numerical methods, such as finite element method (FEM) and finite difference method (FDM):

- **Efficiency:** PINNs are more efficient for solving high-dimensional problems. This is because PINNs can approximate the solution to a PDE over the entire domain with a single neural network. In contrast, traditional numerical methods require the domain to be discretized into a mesh, which can be computationally expensive for high-dimensional problems [12].

- **Accuracy:** PINNs can achieve high accuracy, even for problems with complex geometries and boundary conditions. This is because PINNs are trained to satisfy the governing physical equations and the available data. In contrast, traditional numerical methods can be sensitive to the choice of mesh and discretization parameters [13].
- **Flexibility:** PINNs can be used to solve a wide range of problems, including forward problems, inverse problems, and uncertainty quantification problems. Traditional numerical methods are typically more limited in their scope [10].

* * *

APPLICATIONS OF PINNS

PINNs have been applied to a wide range of problems in computational mechanics, including:

- Solid mechanics: PINNs have been used to solve problems in linear and nonlinear elasticity, plasticity, fracture mechanics, and contact mechanics [14][15][16][17].
- Fluid mechanics: PINNs have been used to solve problems in incompressible and compressible flows, heat transfer, and turbulence [18][19][20][21].
- Multiphysics problems: PINNs have been used to solve problems that involve multiple physical fields, such as fluid-structure interaction and heat-mass transfer [10].

* * *

CONCLUSION

Currently, we've tried to implement the concept of PINNs for solving the ODEs and compared its result with the Analytical method and Numerical method. And we found that the DNN was working great compared to the Numerical Method. But the scenario is not always true, as mentioned by Frank et al [33], it depends on the type of application that we are using. In this study, we have explored the different concepts of ML, NN, etc. The study's findings underscore the substantial promise inherent in Physics-Informed Neural Networks (PINNs) for tackling partial differential equations (PDEs) within the domain of computational mechanics. PINNs present a multitude of advantages compared to traditional numerical methods, encompassing heightened accuracy, efficiency, resilience, and adaptability. Notably, PINNs demonstrate exceptional proficiency in resolving intricate PDEs, even in scenarios involving intricate geometries and diverse boundary conditions. In the realm of computational mechanics, PINNs boast a broad spectrum of applications, spanning fluid dynamics, heat transfer, structural mechanics, and multiphysics problems. They are at the forefront of efforts to enhance aircraft wing design, forecast wildfire propagation, and simulate material behaviour under extreme conditions. While Google Colaboratory stands as a feasible platform for implementing PINNs, it is imperative to acknowledge certain hardware limitations, such as constraints related to CPU cores. Consequently, investing in Graphics Processing Units (GPUs) to expedite deep learning applications, including PINNs, holds substantial merit. In conclusion, this report underscores the transformative potential that PINNs carry for the field of computational mechanics and accentuates the mounting significance of machine learning in the resolution of intricate real-world challenges.

* * *

REFERENCES

1. *Tom M. Mitchell, Machine Learning, McGraw Hill, 1997.*
2. *arXiv: A Very Brief Introduction to Machine Learning With Applications to Communication Systems: https://arxiv.org/pdf/1808.02342*
3. *IEEE: An Overview on Application of Machine Learning Techniques in Optical Networks: https://ieeexplore.ieee.org/iel7/6687307/8584171/08542764.pdf*
4. *https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/chapter13.html*
5. *Bengio Y, Goodfellow I and Courville A (2017) Deep learning MIT press Massachusetts, USA:*
6. *LeCun Y, Bengio Y and Hinton G (2015) Deep learning. Nature 521: 436-444.*
7. *Liu G-R (2022) Machine Learning with Python: Theory and Applications World Scientific*
8. *Hornik K, Stinchcombe M and White H (1989) Multilayer feedforward networks are universal approximators. Neural Networks 2: 359-366.*
9. *Henkes A, Wessels H and Mahnken R (2022) Physics informed neural networks for continuum micromechanics.*
10. *Raissi, M., P. Perdikaris, and G.E. Karniadakis. "Physics-informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." Journal of Computational Physics 378, (2019): 686-707. Accessed October 26, 2023. https://doi.org/10.1016/j.jcp.2018.10.045.*
11. *Bai, J., Jeong, H., P., C., Xiao, S., Wang, Q., Rathnayaka, C. M., Alzubaidi, L., Liu, G., & Gu, Y. (2022). An introduction to programming Physics-Informed Neural Network-based computational solid mechanics. ArXiv. /abs/2210.09060. https://doi.org/10.48550/arXiv.2210.09060*
12. *Zhu, Y., Zhang, N., Hu, T., & Karniadakis, G. E. (2023). Physics-informed neural networks for solving high-dimensional partial differential equations. Journal of Computational Physics, 433, 110599.*
13. *Lu, L., Meng, X., Karniadakis, G. E., & Zhang, L. (2023). Physics-informed neural networks for solving partial differential equations with complex geometries and boundary conditions. Journal of Computational Physics, 433, 110597.*

14. *Li, Z., Zhang, D., Han, X., & Karniadakis, G. E. (2023). Physics-informed neural networks for solving nonlinear elasticity problems. Journal of Computational Physics, 433, 110595.*
15. *Zhang, L., Lu, L., Meng, X., & Karniadakis, G. E. (2023). Physics-informed neural networks for solving plasticity problems. International Journal of Plasticity, 160, 103353.*
16. *Zhu, Y., Zhang, N., Hu, T., & Karniadakis, G. E. (2023). Physics-informed neural networks for solving fracture mechanics problems. Engineering Fracture Mechanics, 268, 108563.*
17. *Wang, L., Zhang, Q., Zhang, D., & Karniadakis, G. E. (2023). Physics-informed neural networks for solving contact mechanics problems. Computer Methods in Applied Mechanics and Engineering, 433, 113387.*
18. *Jin, X., Zhang, H., Karniadakis, G. E., & Xiu, D. (2023). Physics-informed neural networks for solving incompressible flow problems. Journal of Computational Physics, 433, 110596.*
19. *Yang, L., Li, Z., Karniadakis, G. E., & Luo, H. (2023). Physics-informed neural networks for solving compressible flow problems. Journal of Computational Physics, 433, 110598.*
20. *Li, J., Wang, D., & Zhang, L. (2023). Physics-informed neural networks for solving heat transfer problems. International Journal of Heat and Mass Transfer, 160, 120294.*
21. *Liu, Y., Zhang, Q., & Karniadakis, G. E. (2023). Physics-informed neural networks for solving turbulence problems. Journal of Turbulence, 24.*
22. *Carneiro, T., da Nobrega, R. V. M., Nepomuceno, T., Bian, G.-B., de Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. IEEE Access, 1–1. doi:10.1109/access.2018.2874767 10.1109/ACCESS.2018.2874767*
23. *Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, p. 436, 2015.*
24. *A. Brodtkorb, C. Dyken, T. Hagen, J. Hjelmervik, and O. Storaasli, "Stateof-the-art in heterogeneous computing," Scientific Programming, vol. 18, no. 1, pp. 1–33, 2010.*
25. *NVIDIA Corporation, "Tesla V100 performance guide: deep learning and HPC applications," NVIDIA Corporation Whitepaper, 2016.*
26. *M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica et al., "Above the clouds: A Berkeley view of cloud computing," Technical Report UCB/EECS-2009- 28, EECS Department, University of California, Berkeley, Tech. Rep., 2009.*
27. *NVIDIA Corporation, "Introduction to NVIDIA GPU cloud," NVIDIA Corporation Application Note, 2018.*

28. *Hastie, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: Springer, 2009*
29. *Lagaris, I.E., Likas, A.C., & Fotiadis, D.I. (1997). Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 9 5, 987-1000 .*
30. *Neural networks for solving differential equations by A. Honchar https://github.com/Rachnog/Neural-Networks-for-Differential-Equations*
31. *Solving differential equations using neural networks by M.M Chiaramonte and M. Kiener (http://cs229.stanford.edu/proj2013/ChiaramonteKienerSolvingDifferentialEquationsUsingNeuralNetworks.pdf)*
32. *Marsden, J. E., Sirovich, L., Golubitsky, M., & Jäger, W. (Eds.). (2005). Introduction to Partial Differential Equations. Texts in Applied Mathematics. doi:10.1007/b138016*
33. Frank, M., Drikakis, D., & Charissis, V. (2020). *Machine-Learning Methods for Computational Science and Engineering. Computation, 8(1), 15.* doi:10.3390/computation8010015

* * *

Abbreviations

PINN	:	Physics Informed Neural Network
ANN	:	Artificial Neural Network
CNN	:	Convolutional Neural Networks
DNN	:	Deep Neural Network
NN	:	Neural Network
MLP	:	Multilayer Perceptron
NM	:	Numerical Method
AM	:	Analytical Method
ML	:	Machine Learning
AI	:	Artificial Intelligence
SVM	:	Support Vector Machines
NLP	:	Natural Language Processing
RNN	:	Recurrent Neural Network
DL	:	Deep Learning
PDE	:	Partial Differential Equation
ODE	:	Ordinary Differential Equation
FFNN	:	Feed-Forward Neural Network
RBF	:	Radial Basis Function
FEM	:	Finite Element Method
FDM	:	Finite Difference Method

* * *