Applications Of Physics Informed Neural Network - Option B (Empirical evaluation):

Mohit Garg 2020AM10657
Department of Computational Mechanics
IIT Delhi

Mudit Chopra 2020AM10658
Department of Computational Mechanics
IIT Delhi

Editor: Mohit and Mudit

Abstract

Traditional approach to solve partial differential equations is computationally expensive and most of physics governed equations are partial differential equations in nature.So, we can see lack of satisfactory approach to our problem.The great challenge is to find a set of functions that will be able to approximate the solution.So, we tend to try machine learning based methods to deal with such problems. A method for solving partial differential equations in continual mechanics is the variational technique to obtaining the extremum of an objective functional. Finding a group of functions that can accurately approximate the solution is the main issue in the direct calculus of variations methods.

1. What is the problem?

In recent years, physics-informed neural networks (PINNs) have emerged as a promising approach for solving partial differential equations (PDEs) by incorporating the physical laws of the problem into the neural network architecture. In this work, we investigate the performance of different variations of PINN on linear elasticity problem.

Linear elasticity is a fundamental problem in engineering mechanics, which describes the deformation and stresses in a solid under external loads. We consider a 2D linear elasticity problem with homogeneous Dirichlet boundary conditions, where the goal is to compute the displacement field given a body force and a prescribed geometry.

To compare the accuracy and efficiency of the different PINN methods, we perform a objective study and present a detailed analysis of the results.The results of our study show that GPINN outperforms standard PINN and WPINN in terms of accuracy and convergence rate.

Overall, this study provides valuable insights into the performance of different PINN methods for linear elasticity problems and highlights the potential of adaptive PINN algorithms for efficient and accurate solutions of PDEs.

2. Why is it an important problem?

The problem of solving partial differential equations (PDEs) is an important challenge in many scientific and engineering fields, such as physics, mechanics, fluid dynamics, and materials science. Traditional numerical methods, such as finite element and finite difference methods, have been widely used for solving PDEs. However, these methods are often computationally expensive, especially for problems with complex geometries and boundary conditions.

Physics-informed neural networks (PINNs) offer a promising alternative to traditional methods, by incorporating the governing equations and boundary conditions directly into the neural network architecture. This approach has several advantages, including reduced computational cost, higher accuracy, and the ability to handle complex geometries and boundary conditions.

In the context of linear elasticity, PINNs can be used to accurately predict the deformation and stresses in a solid under external loads, given the geometry and material properties. This is an important problem in many engineering applications, such as structural design, mechanical engineering, and aerospace engineering. Accurate predictions of the deformation and stresses can help engineers to optimize the design and performance of structures, reduce material usage, and ensure the safety and reliability of the system.

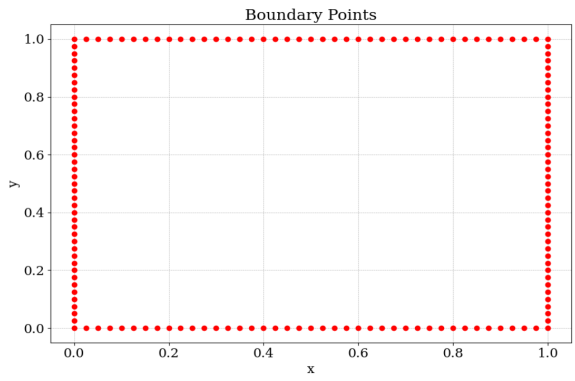
Moreover, the use of PINNs for solving linear elasticity problems has several advantages over traditional numerical methods. PINNs can handle complex geometries and boundary conditions more easily, without the need for mesh generation and remeshing. PINNs can also provide more accurate solutions, especially for problems with high-frequency solutions, due to their ability to approximate complex nonlinear functions.

Overall, the problem of solving PDEs, including linear elasticity, is an important and challenging problem in many scientific and engineering fields. The development and application of PINNs offer a promising avenue for tackling these problems, and our study provides insights into the performance of different PINN techniques for solving linear elasticity problems with Dirichlet boundary conditions.

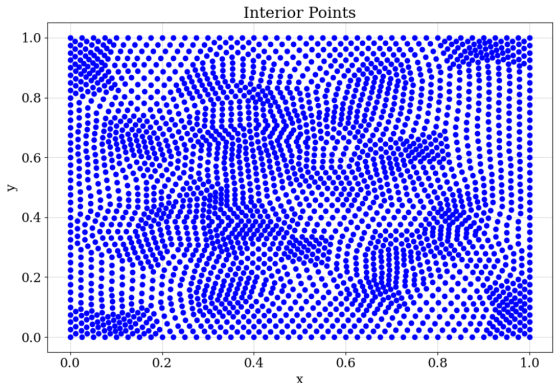
3. What is the intuition behind PINN and its variants being considered?

The intuition behind Physics-Informed Neural Networks (PINNs) is to incorporate the physical laws and constraints directly into the neural network architecture, allowing the network to learn the underlying physics of the problem. This is achieved by defining a loss function that consists of two parts: a data-driven loss, which measures the difference between the predicted solution and the observed data, and a physics-informed loss, which enforces the governing equations and boundary conditions of the problem.

In the context of our problem of linear elasticity with Dirichlet boundary conditions, the governing equations are the linear elasticity equations, which relate the stresses and



(a) Boundary mesh from x,y in [0,1]



(b) Interior mesh from x,y in [0,1]

strains in the material, and the boundary conditions specify the behavior of the solution at the boundary of the domain. The PINN architecture incorporates these equations and boundary conditions as constraints on the output of the neural network, effectively enforcing the physics of the problem during the training process.

Our study futher tries to improve performance by making some changes to loss as although PINN can be somewhat tackled by tuning optimizers(adding LBFGS after Adam , scheduling operations like Chain scheduler,weight decay and other hyperparameters, etc.), but the focus of our study is on handling the loss function. Now observe that our total loss is a multi loss system consisting of residue of PDE, boundary conditions. The focus is on effectively formulating ways of loss function.

Overall, the intuition behind the PINN technique is to combine the power of deep neural networks with the physical laws and constraints that govern the problem, resulting in a highly flexible and accurate approach for solving PDEs. Our study provides insights into the strengths and limitations of different variants of the PINN technique for solving linear elasticity problems with Dirichlet boundary conditions.

4. Defining problem for training and evaluation

We solve the following linear elasticity boundary value plane stress problem on Square Domain

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + G\left(\frac{1+\nu}{1-\nu}\right)\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y \partial x}\right) + f_x = 0$$

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + G\left(\frac{1+\nu}{1-\nu}\right)\left(\frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 u}{\partial x \partial y}\right) + f_y = 0$$

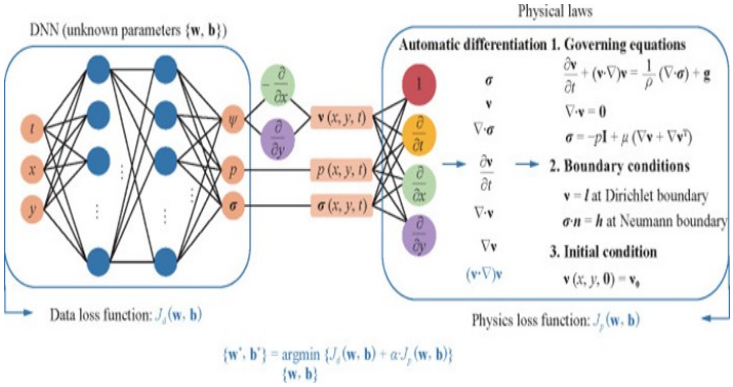
where $u_1 = u$ and $u_2 = v$ are the deformation in the x and y direction respectively, $G = E/2(1+\nu)$, $E = 1$ GPa is the Young's modulus, and $\nu = 0.3$ is the Poisson ratio of the material. We prescribe fixed boundary conditions, hence, we imagine a plate that is fixed

at the boundary, with periodic loading in the interior. Additionally, the test problem will have zero stresses in the z-direction, hence, $\sigma_{3j} = \sigma_{i3} = 0$, for $i, j = 1, 2, 3$.

Here we consider f_x and f_y based on solution $\mathrm{u} = \sin(2\pi x)\sin(2\pi y)$ and $\mathrm{v} = \sin(2\pi x)\sin(\pi y)$ and propose Forward Problem for Plane Stress Linear Elasticity Boundary Value Problem on given mesh (2193 points) to find u,v at a given x,y.

5. Brief review of research papers

The popularity of solving differential equations/ numerical equations via machine learning has gained popularity and is being seen as an effective/ computationally inexpensive support to the traditional scientific numerical computing methods like CFD, FEM.



(a) Physics Informed Neural Network Architecture[3]

$$L = L_{data} + L_{PDE},$$

where

$$L_{data} = \frac{1}{N_u}\sum_i^{N_u} \| u(\mathbf{x}^i_{data}, t^i_{data}) - u^i_{data} \|^2 + \frac{1}{N_v}\sum_i^{N_v} \| v(\mathbf{x}^i_{data}, t^i_{data}) - v^i_{data} \|^2 + \frac{1}{N_w}\sum_i^{N_w} \| w(\mathbf{x}^i_{data}, t^i_{data}) - w^i_{data} \|^2,$$

and

$$L_{PDE} = \frac{1}{N_f}\sum_i^{N_f}\sum_j^{4} \| f_j(\mathbf{x}^i_f, t^i_f) \|^2,$$

$$f_{1,2,3} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u},$$

$$f_4 = \nabla \cdot \mathbf{u}.$$

(b) Loss function[4]

Raissi et al, 2019[1] in their study describes in detail the application of deep learning for numerical solution of PDEs via various deep learning architecture like CNN, feedforward MLP etc in solving the Partial Differential Equations. They described concept of PINN and its application in forward problems and inverse problems in physics domain. The learning process involved both data driven and physics-based loss approach (equation residue loss). The results achieved via PINN closely matched with the analytical solution.

The finite element method (FEM) (Chen, 2013) is typically the method of choice to approximate solutions of LEBVP on a macroscale. It consists of discretizing the weak form of the PDE, and solving the discretize formulation on shape elements. In our case though, we have already started with periodic solution with which we can compare our final results.

Since we are dealing with PINNs, so we have investigated some current results on improving PINN performance based on loss function.

It is well-established that PINNs have much difficulty resolving boundary conditions for static PDEs with continuous solutions. Alexandros Papados[2] has introduced solving compressible Euler equations that model gas and fluid dynamics using an original PINNs formulation, Weighted-Physics-Informed Neural Networks with Domain Extension (W-PINNs-DE). In addition, we solve a plane stress linear elasticity boundary value problem (LEBVP)

from solid mechanics, using a variation of W-PINNs-DE, W-PINNs where we allot different weights to different type of losses while calculating total loss for optimization.For example, in LEBVP problem, we have two types of losses - residual and boundary loss.The modification here is to incorporate weights to the total loss function, such that the loss of the boundary condition is minimized faster than the loss of the PDE. Allowing the loss of boundary condition to decrease faster than that of the PDE ensures the neural network learns the boundary condition data accurately.

Paris Perdikaris[3] present a learning rate annealing algorithm that utilizes gradient statistics during model training to balance the interplay between different terms in composite loss functions.The key benefits of the proposed automated procedure is that this adaptive method can be easily generalized to loss functions consisting of multiple terms (e.g., multi-variate problems with multiple boundary conditions on different variables), while the extra computational overhead associated with computing the gradient statistics is small, especially in the case of infrequent updates.

Algorithm 1: Learning rate annealing for physics-informed neural networks

Consider a physics-informed neural network $f_\theta(\boldsymbol{x})$ with parameters θ and a loss function

$$\mathcal{L}(\theta) := \mathcal{L}_r(\theta) + \sum_{i=1}^{M} \lambda_i \mathcal{L}_i(\theta),$$

where $\mathcal{L}_r(\theta)$ denotes the PDE residual loss, the $\mathcal{L}_i(\theta)$ correspond to data-fit terms (e.g., measurements, initial or boundary conditions, etc.), and $\lambda_i = 1$, $i = 1, \ldots, M$ are free parameters used to balance the interplay between the different loss terms. Then use S steps of a gradient descent algorithm to update the parameters θ as:

for $n = 1, \ldots, S$ **do**

(a) Compute $\hat{\lambda}_i$ by

$$\hat{\lambda}_i = \frac{\max_\theta\{|\nabla_\theta \mathcal{L}_r(\theta_n)|\}}{\overline{|\nabla_\theta \mathcal{L}_i(\theta_n)|}}, \quad i = 1, \ldots, M, \tag{40}$$

where $\overline{|\nabla_\theta \mathcal{L}_i(\theta_n)|}$ denotes the mean of $|\nabla_\theta \mathcal{L}_i(\theta_n)|$ with respect to parameters θ.

(b) Update the weights λ_i using a moving average of the form

$$\lambda_i = (1 - \alpha)\lambda_i + \alpha \hat{\lambda}_i, \quad i = 1, \ldots, M. \tag{41}$$

(c) Update the parameters θ via gradient descent

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_r(\theta_n) - \eta \sum_{i=1}^{M} \lambda_i \nabla_\theta \mathcal{L}_i(\theta_n) \tag{42}$$

end

The recommended hyper-parameter values are: $\eta = 10^{-3}$ and $\alpha = 0.9$.

Jeremy Yu[4] propose a new method, gradient-enhanced physics-informed neural networks (gPINNs), for improving the accuracy of PINNs. gPINNs leverage gradient information of the PDE residual and embed the gradient into the loss function. In PINNs, we only enforce the PDE residual f to be zero; because f (x) is zero for any x, we know that the derivatives of f are also zero. Now the concept is, we assume that the exact solution of the PDE is smooth enough such that the gradient of the PDE residual $\nabla f(x)$ exists, and then propose the gradient-enhanced PINNs to enforce the derivatives of the PDE residual to be zero as well, i.e., $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_d}\right) = 0, x \in \Omega$.

The loss function of gPINNs is: $\mathrm{L} = \mathrm{w}_f L_f + w_b L_b + \sum_{i=1}^{d} w_{gi} L_{gi}(\theta; T_{gi})$
where $L_{gi}(\theta; T_{gi}) = \frac{1}{|T_{gi}|} \sum_{x \in T_{gi}} \left|\left| \frac{\partial f}{\partial x_i} \right|\right| |^2$.

By enforcing the gradient of the PDE residual, gPINN improves the accuracy of the predicted solutions for u and requires less training points. One motivation of gPINN is that the PDE residual of PINNs usually fluctuates around zero, and penalizing the slope of the residual would reduce the fluctuation and make the residual closer to zero

So, we are finally going to study W-PINN, gPINNs and adaptive PINN (learning rate annealing algorithm) along with original implementation of PINN in our research paper.

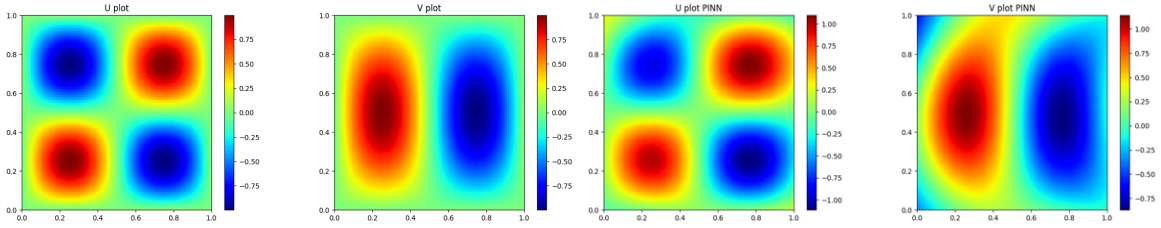
6. Architecture and parameters

No. of hidden layers	5
No. of neurons in each layer	30
Activation function	Tanh
Optimizer	Adam
Learning rate	0.001
No. of epochs	10000

We use NN with 5 layers having 30 neurons each and Tanh as activation function with lr = 0.001 in Adam optimizer.In gPINN, weight assigned to gradient loss term is 0.01 which generally gives decent results as per research paper. Initializations are 0 in general.

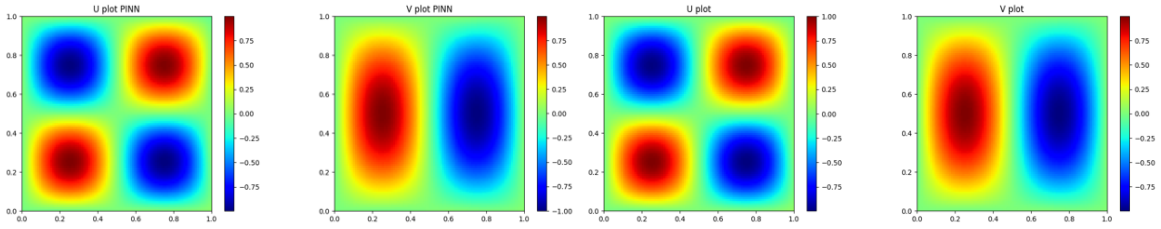
7. Results and comparisons

7.1 u-v solution



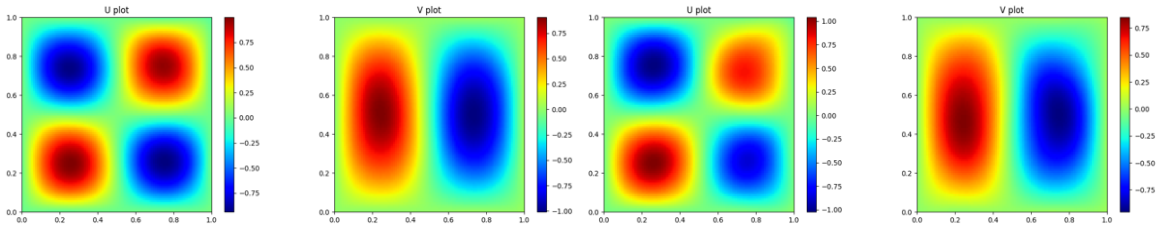
(a) Original solution

(b) PINN solution



(c) wPINN solution

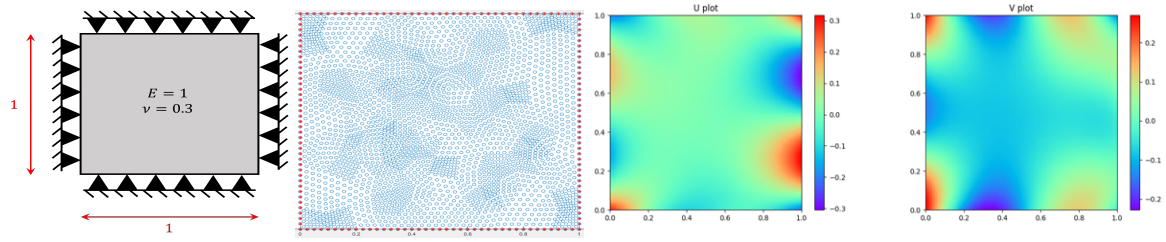
(d) gPINN solution



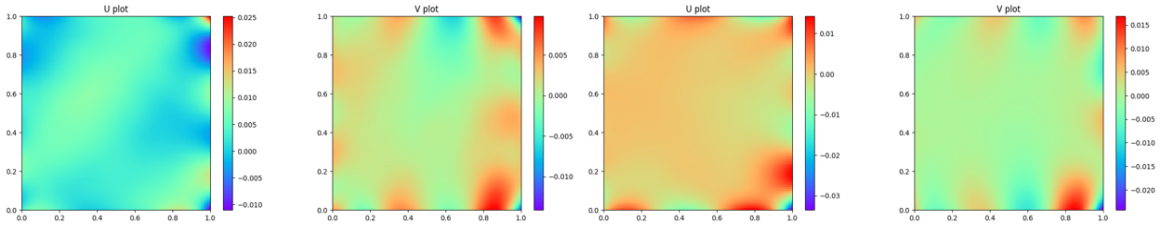
(e) Adaptive PINN solution

(f) Adaptive gPINN solution

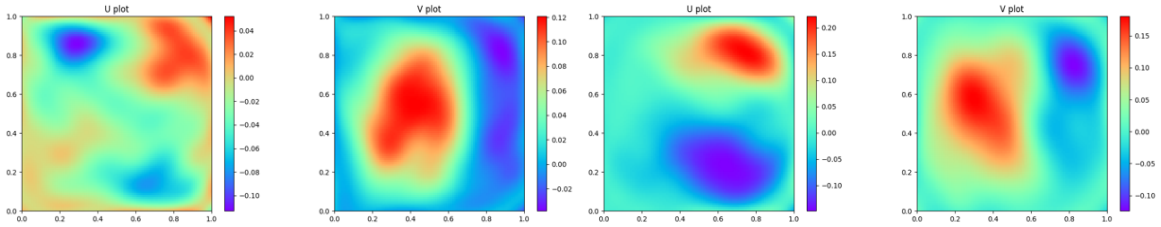
7.2 Error plots (exact - numerical)



(a) PINN solution



(b) wPINN solution (c) gPINN solution



(d) Adaptive PINN solution (e) Adaptive gPINN solution

7.3 MSE loss for u and v

We compare different MSE loss i.e. mean squared average loss for u and v using exact and predicted values of u,v by different variants of PINN.

Simple PINN implementation

MSE loss for u = 0.00620190

MSE loss for v = 0.00674600

WPINN - Weighted PINN implementation

MSE loss for u = 0.00063600

MSE loss for v = 0.00017275

Adaptive PINN

MSE loss for u = 0.00114598

MSE loss for v = 0.00305061

gPINN-Gradient-enhanced PINN

MSE loss for u = 9.89493e-6

MSE loss for v = 8.546528e-6

gPINN + Adaptive PINN - a try to combine the both algorithms

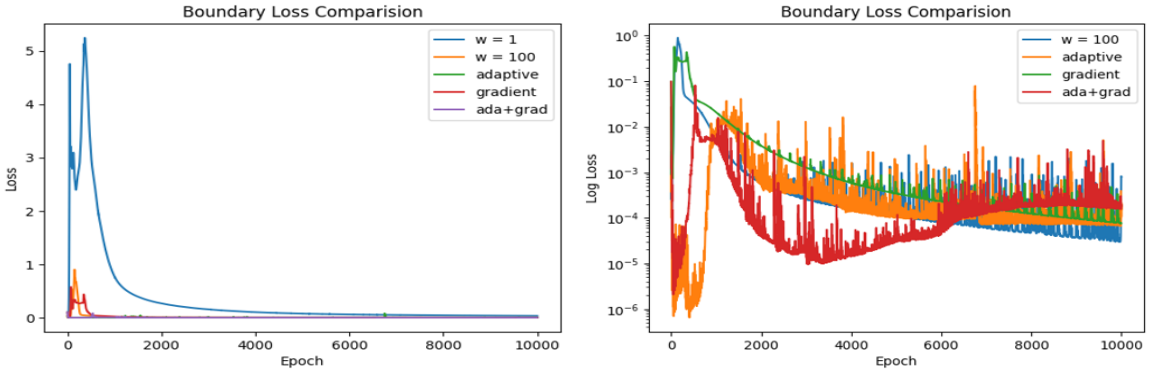
MSE loss for u = 0.00557421

MSE loss for v = 0.00517552

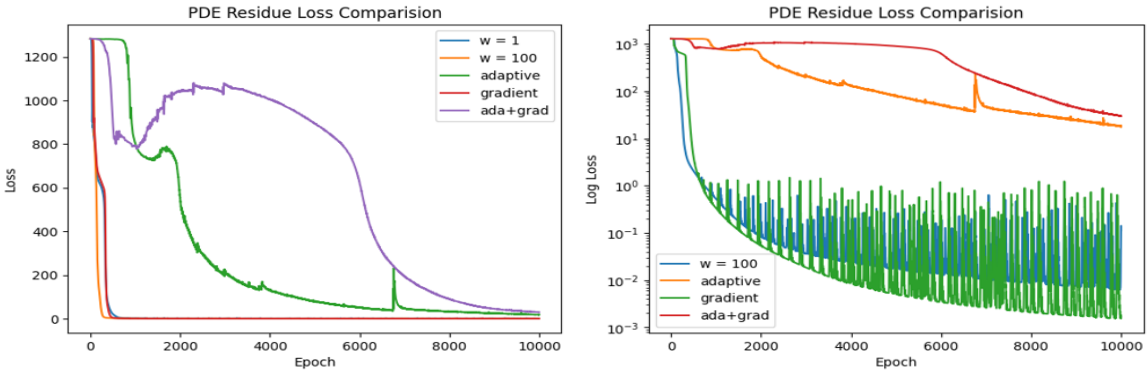
We get PINN>gPINN + Adaptive>Adaptive>WPINN>gPINN in terms of loss values so in terms of overall performance over given domain, we observe opposite trend PINN<gPINN + Adaptive<Adaptive<WPINN<gPINN.

7.4 Loss History and inferences

7.4.1 Boundary Loss History



7.4.2 PDE residue Loss History



Finally from loss history,one can make following conclusions based on 5 given variants of PINN:

7.4.3 Simple PINN implementation

No extra factor multiplied with boundary loss - It is clear from the obtained results that the boundary condition is not properly reinforced(as can be seen from boundary loss curve) and results are not effective. It is ineffective to reduce the overall loss and hence the results of displacement field obtained have much comparative error.

7.4.4 WPINN - Weighted PINN implementation

100 factor multiplied with boundary loss - It is clear from the obtained results that the boundary condition is effectively imposed, from boundary loss curve). It reduce the boundary loss but adding weight factor of large magnitude creates a bias towards boundary loss. It can be seen from the training process and the PDE residue plots that model is not that effective to handle PDE residue loss. Hence this requires other algorithm to achieve overall optimal convergence.

7.4.5 Adaptive PINN

In this the weight factor is adaptively decided based on the relative magnitudes of loss gradient terms. As can be seen from the loss curves it requires much more epochs as compared to regular PINN but it is less noisy and handles the bias towards weight factor part. It requires more resources and hence showing less effective result in 10000 epoch.

7.4.6 gPINN-Gradient-enhanced PINN

It uses gradient of residue loss terms to provide stability. It can be seen from the PDE loss and boundary loss curves that it effectively handles both the components present in loss. It reduces boundary value loss just behind w = 100 but greatly reduces the PDE loss as compared to other models.The gradient PINN loss minimizes all loss components effectively and there is difference in convergence values with gradient PINN having the least. Also, from the slope of gradient PINN it is evident that it must on further training reach lower value (optimally) which is the main issue for PINN.

7.4.7 gPINN + Adaptive PINN - a try to combine the both algorithms

In this the weight factor is adaptively decided based on the relative magnitudes of loss gradient terms while it also uses gradient of residue loss terms to provide stability . As can be seen from the loss curves it requires much more epochs as compared to regular PINN but it is less noisy and handles the bias towards weight factor part. One can observe that although it is too slow initially but after 5000 iterations we observe significant improvement trend which leads to our guessing while large number of iterations or different optimizers may lead to better results.

8. Computational resources

While we can see different variants of PINN here but still we need into take consideration computational resources involved.PINN and WPINN have no difference in computational resources while the extra computational overhead associated with computing the gradient statistics is also small (adaptive PINN), especially in the case of infrequent updates. gPINN has nearly twice computational resources involved as compared to PINN but extra resources are worth it considering better performance of gPINN specially at higher iterations.

9. Implementation and future path

Implementation has been shared at Google Drive link `https://drive.google.com/drive/folders/1VDDgZSHVi3zGQJJZxTyzS8u-bWPFHQW5?usp=sharing`.

Now based on our empirical evaluation, we conclude that GPINN is the best technique for solving linear elasticity problems with Dirichlet boundary conditions, due to its superior accuracy and convergence rate. However, adaptive PINN methods can provide a useful tool for reducing the computational cost and improving the accuracy of the solution. While the problem of solving linear elasticity is not completely solved, our study demonstrates the potential of PINNs for tackling complex PDE problems. Future research could focus on investigating the performance of PINNs for more general boundary conditions, as well as exploring the use of other types of neural networks for solving PDEs over different domains. If one could solve faster convergence for combination of adaptive and gradient-enhanced PINN like based on different learning rate and optimizer, we could get a new efficient techniques to be added in successful modifications of PINN.

10. Author Contributions

We both believe that the final completion of this project was based on equal effort of both of us.All parts from Algorithm design, coding and result inference to presentation and report preparations were jointly done by both of us and it is not possible to accurately mention who did what part of the project.

References

[1]Deep Learning for the Numerical Solution of Partial Differential Equations: A Review" by Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, published in the Journal of Computational Physics in 2019.

[2] Physics Informed Deep Learning and their Application in Computational Solid and Fluid Mechanics by Alexandros Papados

[3]Wang, S., Teng, Y., Perdikaris, P. (2020). Understanding and mitigating gradient pathologies in physics-informed neural networks. Journal of Computational Physics, 415, 109461.

[4]Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems by Jeremy Yu, Lu Lu, Xuhui Meng, George Em Karniadakis,Computer Methods in Applied Mechanics and Engineering, Volume 393, 2022, 114823, ISSN 0045-7825, https://doi.org/10.1016/j.cma.20

[5]Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism by Levi McClenny and Ulisses Braga-Neto,https://arxiv.org/abs/2009.04544