Fast Convergence PINNs Using Pseudo-Density Embedding: A study on Solid Mechanics

Melvin Wong[1]*, Jiao Liu[1]*, Ge Jin[2]*, Kunpeng Li[1,3], Doan Ngoc Chi Nam[4]

[1]*College of Computing & Data Science, Nanyang Technological University (NTU), Singapore*
[2]*School of Aerospace Engineering, Beijing Institute of Technology, Haidian, Beijing, China*
[3]*School of Physical and Mathematical Sciences (SPMS), Nanyang Technological University (NTU), Singapore*
[4] *Singapore Institute of Manufacturing Technology (SIMTech), A*STAR, Singapore*

{wong1357, jiao.liu}@ntu.edu.sg, jinge52293@gmail.com, kunpeng.li@ntu.edu.sg, doanncn@simtech.a-star.edu.sg

***Abstract*—Physics-informed neural networks (PINNs) have shown the potential to incorporate physical laws into machine learning models. However, their widespread adoption is limited due to significant convergence issues, particularly on complex geometries. This paper presents a first study on training a model in the form of a pseudo-density embedding that encodes geometry information and subsequently extending from this baseline model in future PINN training process. Our study on complex geometry involving the science of solid mechanics demonstrates that such an embedding not only streamlines preprocessing tasks but also produces precise physical outcomes and notably accelerates convergence compared to conventional PINNs. Empirical findings indicate that our proposed method, pseudo-density embedding PINN (PD-PINN), achieves a significant 3% to 8% reduction in error rates within a defined computational budget on a linear elastic solid mechanics example, surpassing performance benchmarks set by traditional methodologies.**

***Index Terms*—Physics-informed neural networks, solid mechanics, linear elasticity, pseudo-density embedding**

I. Introduction

The simulation and modeling of physical systems have long been central to numerous scientific and engineering domains [1]. With the recent advancements in deep neural networks, researchers are exploring the application of these networks in these domains [2]. Among the promising methods for such tasks, physics-informed neural networks (PINNs) have received considerable attention [3]. A distinctive feature of PINNs is their ability to integrate the governing physics laws described by partial differential equations into the learning objectives of the neural networks. This ensures that their outputs are consistent with the underlying governing physics.

Although PINNs have been widely studied in recent years, they still face the challenge of slow convergence, especially when dealing with irregular geometry boundaries [4]. The performance of PINNs can be improved by designing effective training algorithms [5]. Over the past five years, several such algorithms have been proposed, broadly categorized into loss re-weighting schemes [6], [7], collocation points re-sampling [8], [9], and curriculum training [10]. In this paper, we introduce a new category of training algorithm for linear elastic solid mechanics called pseudo-density embedding PINN (PD-PINN). While traditional methods characterize the shape of a geometry only by sampling points at its boundaries, we propose using a pseudo-density embedding network as a baseline model makes complementary use of pseudo-density fields inside and outside the geometry to provide additional global geometric information. This additional geometric information enhances the ability of PINN to handle irregular geometric problems and significantly accelerates its convergence performance. Additionally, we highlight that the performance of PINNs is highly sensitive to hyperparameter settings, especially the loss weights for boundary conditions [5]. This often necessitates users of PINN to make multiple adjustments to loss weights to achieve an optimal predictive ability for the model. In contrast, our method alleviates the need for explicitly setting these loss weights for boundaries not subject to forces or hinge joints, simplifying the specification of boundary conditions. Moreover, we also found that the proposed PD-PINN can simplify the data preprocessing by allowing the utilization of collection points sampled only in a regular geometric space and the traction boundaries, thereby avoiding the need to sample at the irregular traction-free boundaries. While various methodologies akin to pseudo-density embedding have been introduced within the realm of topology optimization [11], our study demonstrates that deploying a pseudo-density embedding baseline model yields favorable outcomes even in the context of forward problems.

The remainder of the paper is organized as follows. Section II provides the technical background on linear elastic solid mechanics and PINNs. The proposed method is elaborated in Section III. Section IV presents the results of experimental studies. Finally, Section V concludes the paper.

II. Preliminaries

A. Governing Equations in Linear Elastic Solid Mechanic

The linear elastic problem is a fundamental and extensively studied issue in solid mechanics [12]. Here the 2-dimensional problems are considered in this paper. The governing equations for linear elastic problems are described as follows:

$$\sigma_{ij,j} + f_i = 0, \tag{1}$$

$$\sigma_{ij} = \lambda \delta_{ij} \epsilon_{kk} + 2\mu \epsilon_{ij}, \tag{2}$$

*Equal contribution as the first author

$$\epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}), \tag{3}$$

$$\mathbf{x} \in \Omega,$$
$$i, j, k \in \{1, 2\},$$

where σ_{ij} is the Cauchy stress tensor, ϵ_{ij} is the strain tensor, u_i is the displacement, f_i is the body force per unit mass, δ_{ij} is the Kronecker delta function, and λ and μ are the Lamé constants. Note that both σ_{ij}, ϵ_{ij} and u_i are written in Einstein notation and are functions of a point $\mathbf{x}$ inside of the geometry Ω. The governing equations above are closed by the following boundary conditions:

$$\begin{aligned} u_i &= \bar{u}_i, \quad \mathbf{x} \in \Gamma_u, \\ \sigma_{ij} n_j &= \bar{f}_i, \quad \mathbf{x} \in \Gamma_f, \end{aligned} \tag{4}$$

where $\bar{u}_i$ and $\bar{f}_i$ are the displacement and force at the corresponding boundaries Γ_u and Γ_f, respectively, and n_j is the unit outward normal vector on the corresponding boundaries Γ_f.

B. Physics-Informed Neural Networks

Following the function provided by Raissi *et al.* [3], in this subsection, we introduce how to use the PINN to calculate the physical dynamics of linear elastic solid mechanics. We use a fully connected neural network to approximate the solutions of the displacements and record them as $\hat{u}_i, i \in \{1, 2\}$. Then, following the governing equations (2) and (3), we can calculate the approximated Cauchy stress tensor $\hat{\sigma}_{ij}$ and strain tensor $\hat{\epsilon}_{ij}$ using automatic differentiation, respectively. Based on these approximations, the PINN loss function is then defined as the composition of a PDE loss component (L_{pde}) and a boundary condition loss component (L_{bc}):

$$L_{pinn} = w_{pde} L_{pde} + w_{bc} L_{bc}, \tag{5}$$

$$L_{pde} = ||\hat{\sigma}_{ij,j} + f_i||^2_{\Omega}, \tag{6}$$

$$L_{bc} = ||\hat{u}_i - \bar{u}_i||^2_{\Gamma_u} + ||\hat{\sigma}_{ij} n_j - \bar{f}_i||^2_{\Gamma_f}, \tag{7}$$

where w_{pde} and w_{bc} are the weights of the corresponding losses. Note that the PDE and boundary condition loss components are defined over a continuous domain. In particular, it is hard to directly calculate them since they are intractable. Therefore, we need to estimate the loss components using Monte Carlo based on a set of collection points. For linear elastic solid mechanics, we need three subsets of collection points, denoted as $\mathcal{D}_{\Omega} = \{\mathbf{x}^{(l)}_{\Gamma_u}\}^{N_{\Omega}}_{l=1}$, $\mathcal{D}_{\Gamma_u} = \{\mathbf{x}^{(l)}_{\Gamma_u}\}^{N_{\Gamma_u}}_{l=1}$, and $\mathcal{D}_{\Gamma_f} = \{\mathbf{x}^{(l)}_{\Gamma_f}\}^{N_{\Gamma_f}}_{l=1}$. These three subsets contain collection points in the geometry Ω, on the boundary Γ_u, and on the boundary Γ_f, respectively. Meanwhile, for the points in $\mathcal{D}_{\Gamma_f}$, the corresponding unit outward normal vectors should also be provided. Based on the loss components estimated according to the above subsets, gradient-based optimizers such as stochastic gradient descent and Adam can train the PINN.

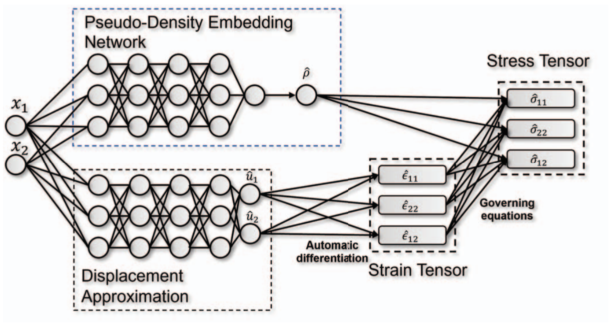


Fig. 1. The PD-PINN architecture comprises two neural networks: the pseudo-density embedding network, which describes the geometry, and the second network, which predicts displacement solutions, strain tensors, and the Cauchy stress tensor. PD-PINN training involves incorporating the pseudo-density embedding network into the overall training process.

III. Proposed Method

In this section, we present the details of PD-PINN. We begin by outlining the training process for the pseudo-density embedding network employed to describe the geometry. We then present the training algorithm for incorporating the pseudo-density embedding baseline model into the PINN.

A. Pseudo-Density Embedding Network

During the pseudo-density embedding baseline model training, we assume that a regular geometric space Ω^+ is capable of covering the original geometry Ω. Subsequently, we sample a set of collection points $\mathcal{D}_{\Omega^+} = \{(\mathbf{x}^{(l)})\}^{N_{\Omega^+}}_{l=1}$ within Ω^+. Each collection point is associated with a corresponding label $\rho^{(l)}$, indicating whether the point $\mathbf{x}^{(l)}$ resides in Ω. If $\rho^{(l)} = 1$, then $\mathbf{x}^{(l)}$ is located inside Ω; otherwise, it is situated outside Ω. Thus, we can train the pseudo-density embedding baseline model $\hat{\rho}(\mathbf{x}) = \text{Sigmoid}(C \cdot t_{\theta}(\mathbf{x}))$ using the dataset $\mathcal{D}^{\rho}_{\Omega^+} = \{(\mathbf{x}^{(l)}, \rho^{(l)})\}^{N_{\Omega^+}}_{l=1}$ to predict the inclusion status of a point within Ω, where $t_{\theta}(\mathbf{x})$ is a multilayer perceptron and C is a predefined parameter.

It is important to note that $\mathcal{D}_{\Omega^+}$ can be easily generated based on any conventional geometry representation, such as mesh, level set function, or point cloud. Furthermore, this baseline model $\hat{\rho}(\mathbf{x})$ itself can be considered as a type of geometry representation.

B. Training of the PD-PINN

Similar to conventional PINNs, we use a neural network $\hat{u}$ to approximate the displacement solution and subsequently provide the predicted strain tensor $\hat{\epsilon}_{ij}$ using automatic differentiation. In contrast to the conventional PINNs, the predicted Cauchy stress tensor is obtained by incorporating the pseudo-density embedding baseline model into (2), i.e.,

$$\hat{\sigma}^+_{ij} = \hat{\rho}(\lambda \delta_{ij} \hat{\epsilon}_{kk} + 2\mu \hat{\epsilon}_{ij}), \tag{8}$$

and the equilibrium equation (1) is modified as:

$$\sigma^+_{ij,j} + \hat{\rho} f_i = 0, \tag{9}$$

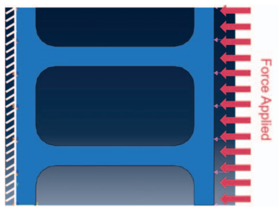
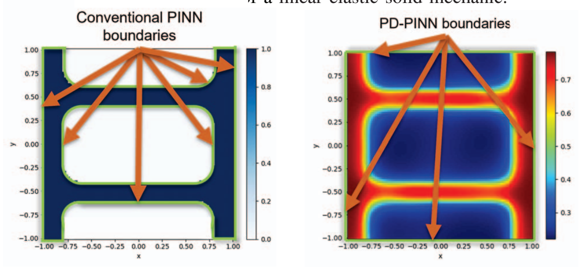


Fig. 2. An example of a linear elastic solid mechanic.



(a) Conventional PINN (b) PD-PINN

Fig. 3. The boundaries of the conventional PINN and the PD-PINN. The conventional PINN defines the boundary conditions at the geometry boundaries (highlighted in green). In contrast, we use the design space boundaries in PD-PINN.

Then, the loss of the PD-PINN is defined as follows:

$$L_{PD-PINN} = w_{pde^+}L_{pde^+} + w_{bc^+}L_{bc^+}, \tag{10}$$

$$L_{pde^+} = ||\hat{\sigma}^+_{ij,j} + \hat{\rho} f_i||^2_{\Omega^+}, \tag{11}$$

$$L_{bc^+} = ||\hat{u}_i - \bar{u}_i||^2_{\Gamma_u} + ||\hat{\sigma}^+_{ij} n_j - \bar{f}_i||^2_{\Gamma^+_f}, \tag{12}$$

where Γ^+_f contains all of the points on the traction force boundaries of Ω and the traction-free boundaries of Ω^+.

Note that the loss function $L_{PD-PINN}$ of the PD-PINN is defined over a continuous region and can be estimated using Monte Carlo. To this end, three sets of collection points, i.e., $\mathcal{D}_{\Omega^+}$, $\mathcal{D}_{\Gamma_u}$, and $\mathcal{D}_{\Gamma^+_f}$, should be sampled in Ω^+, Γ_u and Γ^+_f, respectively. Note that, since Ω^+ is assumed as a regular geometric space, sampling $\mathcal{D}_{\Omega^+}$ and $\mathcal{D}_{\Gamma^+_f}$ is much simpler and convenient than the conventional PINNs which need to sample in the complex geometry Ω and both complex traction-free and traction force boundaries Γ_f.

The structure of the PD-PINN is summarized in Fig. 1, and the training process of the PD-PINN is described as follows:

1) Sample three datasets $\mathcal{D}_{\Omega^+}$, $\mathcal{D}_{\Gamma_u}$, and $\mathcal{D}_{\Gamma^+_f}$ by using mesh grid, and generate $\mathcal{D}^{\rho}_{\Omega^+}$ based on $\mathcal{D}_{\Omega^+}$ based on the calculating geometry.
2) Train the pseudo-density embedding baseline model $\hat{\rho}(\mathbf{x})$ based on $\mathcal{D}^{\rho}_{\Omega^+}$.
3) Train the PD-PINN based on the loss defined in (10).

IV. Experiments and Discussion

To validate the effectiveness of the proposed PD-PINN, we perform a comparative analysis with a conventional PINN using the example depicted in Fig. 2. This illustration presents a complex geometry featuring linear elastic material, where a force is applied to the right boundary, and zero displacements are applied to the left boundary. The material properties are specified as Young's modulus of $E = 1$ and a Poisson's ratio of $\nu = 0.3$, yielding Lamé parameters λ and μ calculated as $\lambda = \frac{E\nu}{(1+\nu)(1-\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$ in 2-D cases. We assume f_i is uniformly set to 0 for simplicity. The ground truth of the example is obtained using a commercial finite element solver, specifically Abaqus.

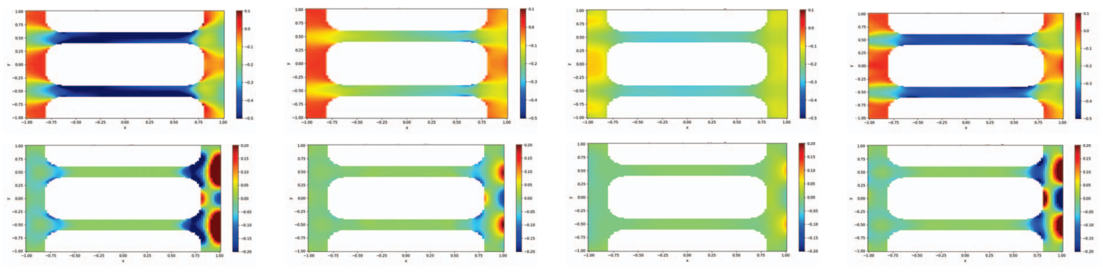
TABLE I
Error comparison between conventional PINN and PD-PINN.

Method	ϵ_{11}	ϵ_{12}	ϵ_{22}	σ_{11}	σ_{12}	σ_{22}
Conventional PINN	5.54%	1.33%	3.20%	5.77%	1.14%	0.47%
PD-PINN C=0.1	0.89%	0.54%	0.75%	0.16%	0.54%	0.12%
PD-PINN C=1.0	**0.83%**	**0.43%**	**0.39%**	**0.09%**	**0.39%**	**0.06%**

For PD-PINN, both the pseudo-density embedding baseline model and the displacement approximation network are implemented as multilayer perceptrons, with structures set to "($\mathbf{x}$)-**30**-30-$\hat{\rho}$" and "($\mathbf{x}$)-**30**-30-30-($\hat{u}_1, \hat{u}_2$)", respectively. The conventional PINN also adopts a multilayer perceptron architecture, configured as "($\mathbf{x}$)-**30**-30-30-($\hat{u}_1, \hat{u}_2$)". We adopted the same as [13] with the first layers of all the aforementioned neural networks incorporating Fourier feature mappings. Concerning the loss function of PD-PINN, both w_{pde^+} and w_{bc^+} are set to 1. Regarding the conventional PINN, w_{pde} and w_{bc} are set to 0.5 and 1, respectively, as suggested that the boundary condition loss should be assigned a larger weight [5]. We employed a cosine scheduler and the Adam optimizer with a learning rate ranging from 5e-3 to 1e-4 for both PD-PINN and conventional PINN. All methods are trained for 500K epochs to observe convergence and ensure training stability.

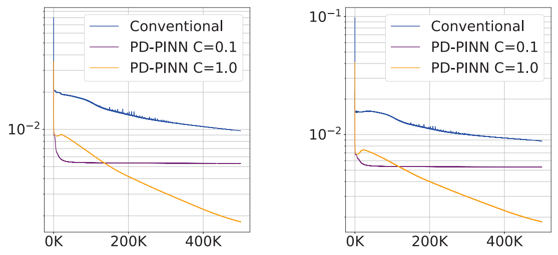
In the initial demonstration, we highlight the streamlining effect that PD-PINN introduces to the preprocessing phase. Illustrated in Fig.3(a), the conventional PINN necessitates the sampling of points along the boundaries of the geometry (indicated by the green lines). Additionally, it requires the computation of norm vectors corresponding to these sampled points. Notably, the geometry's traction-free boundaries exhibit irregularities, making point sampling and norm vector computation intricate. In contrast, as depicted in Fig.3(b), PD-PINN simplifies these procedures by focusing solely on the boundaries of the design space. As a result, we have reduced the complexity of the sampling process.

Table I presents the errors in the predicted values and the ground truth after 500K training steps. Specifically, we provide the errors for components of the strain tensor and the Cauchy stress tensor, namely ϵ_{11}, ϵ_{12}, ϵ_{22}, σ_{11}, σ_{12}, and σ_{22}. To demonstrate the influence of the hyperparameter C, the results obtained by PD-PINN under the conditions of both $C = 0.1$ and $C = 1.0$ are summarized. It can be observed from Table I that PD-PINN can achieve a 3% to 8% reduction (on average) in the error rates for various strain and stress components. These results exhibit a substantial improvement over conventional PINN. To further illustrate the effectiveness of PD-PINN, we present the predicted distribution map of ϵ_{11} in Fig. 4. It can be observed that, compared with conventional PINN, the predictions provided by PD-PINN with $C = 1.0$ closely align with the ground truth obtained from finite element analysis. Fig. 5 also depicts the convergence trends of



(a) FEM (b) Conventional PINN (c) PD-PINN with C=0.1 (d) PD-PINN with C=1.0

Fig. 4. The distribution map of the stress ϵ_{11} (top-row) and the strain σ_{12} (bottom-row) provided by the finite element method (i.e., the ground truth), conventional PINN, the PD-PINN with $C = 0.1$, the PD-PINN with $C = 1$, and .



(a) Strain Error ϵ_{12} (b) Stress Error σ_{12}

Fig. 5. The convergence trends of the conventional PINN, the PD-PINN with $C = 0.1$, and the PD-PINN with $C = 1$ on ϵ_{12} and σ_{12}.

the errors on ϵ_{12} and σ_{12} for conventional PINNs, PD-PINN with $C = 0.1$, and PD-PINN with $C = 1.0$. It is evident that, in the early stage, PD-PINN with $C = 0.1$ exhibits the fastest convergence. Although PD-PINN with $C = 1.0$ initially demonstrates slower convergence than $C = 0.1$, it achieves lower errors than conventional PINNs after approximately 130K epochs. These results underscore the effectiveness of the proposed PD-PINN.

V. Conclusion and Future Direction

The paper introduces PD-PINN, a new approach for linear elastic solid mechanics in the context of PINNs. It utilizes a pseudo-density embedding baseline model to efficiently represent boundary features and incorporate global geometric information, speeding up training. PD-PINN removes the need for explicit loss weight settings for traction-free boundaries, simplifying boundary condition specification. Notably, it allows the use of regular geometric space and traction boundaries for sampling collection points, eliminating the need for irregular boundary sampling. This improves preprocessing and demonstrates superior convergence compared to traditional PINNs in a linear elastic solid mechanics example.

Acknowledgment

The authors would like to express our deep gratitude to individuals, especially Professor Yew-Soon Ong, Dr. Jian Cheng Wong, Dr. Abhishek Gupta, and Dr. Chi Chun Ooi, for their expertise and useful critiques in this research work. This research is partly supported by the Distributed Smart Value Chain programme which is funded under the Singapore RIE2025 Manufacturing, Trade and Connectivity (MTC) Industry Alignment Fund-Pre-Positioning (Award No: M23L4a0001), and the College of Computing & Data Science, Nanyang Technological University (NTU).

References

[1] D. W. Heermann and D. W. Heermann, *Computer-simulation methods*. Springer, 1990.

[2] M. Raissi, “Deep hidden physics models: Deep learning of nonlinear partial differential equations,” *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 932–955, 2018.

[3] M. Raissi, 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*, vol. 378, pp. 686–707, 2019.

[4] J. C. Wong, P.-H. Chiu, C. Ooi, M. H. Dao, and Y. Ong, “Lsa-pinn: Linear boundary connectivity loss for solving pdes on complex geometry,” *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:256598217

[5] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, “An expert's guide to training physics-informed neural networks,” *arXiv preprint arXiv:2308.08468*, 2023.

[6] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient flow pathologies in physics-informed neural networks,” *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[7] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[8] M. A. Nabian, R. J. Gladstone, and H. Meidani, “Efficient training of physics-informed neural networks via importance sampling,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 8, pp. 962–977, 2021.

[9] A. Daw, J. Bu, S. Wang, P. Perdikaris, and A. Karpatne, “Rethinking the importance of sampling in physics-informed neural networks,” *arXiv preprint arXiv:2207.02338*, 2022.

[10] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, “Characterizing possible failure modes in physics-informed neural networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 548–26 560, 2021.

[11] J. Yin, Z. Wen, S. Li, Y. Zhanga, and H. Wang, “Dynamically configured physics-informed neural network in topology optimization applications,” *arXiv preprint arXiv:2312.06993*, 2023.

[12] P. L. Gould, *Introduction to linear elasticity*. New York: Springer, 1994, vol. 2. [Online]. Available: https://doi.org/10.1007/978-1-4614-4833-4

[13] J. Cheng Wong, C. Ooi, A. Gupta, and Y.-S. Ong, “Learning in sinusoidal spaces with physics-informed neural networks,” *IEEE Transactions on Artificial Intelligence*, pp. 1–15, 2022.