XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

DeepONet Surrogate Modeling for Continuum Mechanics in Unbounded Domains: Bypassing BEM Scalability via Operator Learning

Leonardo Santiago[1], George Em Karniadakis[2], Josué Labaki[1]

[1]*Universidade Estadual de Campinas (UNICAMP), School of Mechanical Engineering*
R. Mendeleyev, 200, 13083-860, Campinas, SP, Brazil
l201292@dac.unicamp.br, labaki@unicamp.br
[2]*Division of Applied Mathematics, Brown University*
182 George St., 02912, Providence, RI, USA
george_karniadakis@brown.edu

Abstract. Continuum mechanics problems with unbounded domains require the use of sophisticated mathematical models due to the nature of these systems being governed by partial differential equations. One such tool is the Boundary Element Method (BEM), which reformulates a given boundary-value problem into equivalent integral equations posed solely on the domain's boundary. However, classical BEM approaches involve computing a large number of numerical integrals, leading to significant computational cost. This is particularly true for large-scale problems, where the method's total computing time scales poorly due to dense kernel evaluations. This work presents a data-driven approach using DeepONet architectures trained on synthetic data from numerical simulations. After training, DeepONet performs fixed-cost inference independent of problem scale, enabling the evaluation of solutions in milliseconds. We achieved an error of less than 5% compared with the analytical solution for the problem of a point load applied to an infinite, isotropic, linear-elastic full-space.

Keywords: Surrogate Modeling, DeepONet, Boundary Element Method (BEM), Fixed-Cost Inference, Continuum Mechanics

1 Introduction

Boundary value problems (BVPs) with unbounded domains are prevalent in various fields of science and engineering [1]. These problems are notoriously challenging to model due to their infinite domain and need to enforce far-field conditions (e.g., the Sommerfeld radiation condition [2]). Kelvin's problem in elasticity, which describes the response of an infinite elastic medium to a point force, is a rare example of such problems with an analytical solution [3]. Classical approaches often rely on the boundary element method (BEM), which uses fundamental solutions to reduce the problem to boundary integrals [4]. While BEM is well-suited for infinite domains, it becomes computationally expensive for large-scale problems [5], due to the presence of linear systems with dense and possibly non-symmetric matrices, yielding $\mathcal{O}(N^2)$ operations where N is the number of equations in the resulting linear system, and kernel evaluation via numerical integration is required for each operation. Various strategies have been explored to accelerate the solution of these problems including hardware-based schemes such as General-Purpose Graphics Processing Units (GPGPUs [6]). Nevertheless, evaluating these kernels remains a hindrance that can render BEM impractical for some large-scale analyses.

This bottleneck has motivated research into surrogate models that can rapidly predict solutions for new scenarios without solving the full problem each time. Deep operator networks (DeepONets) were introduced in 2019 by Lu [7] as universal approximators of operators. Unlike standard neural networks that map finite-dimensional inputs to outputs, the DeepONet can take an entire function (e.g., a boundary condition or material distribution) as input and output the solution function, allowing one trained model to solve many instances of a BVP at fixed cost. This makes them attractive for surrogate modeling of PDEs where the domain or boundary conditions vary. The success of DeepONet in related contexts, such as structural mechanics [8], suggests that extending them to BEM-type problems is feasible. Our main contribution is the exploration of training strategies, including a basic vanilla architecture and one that is enhanced by Proper Orthogonal Decomposition techniques, that achieve sub-5% relative L_2 error across a broad material range in the Kelvin problem.

2 Methodology

In this section, we present the mathematical formulation, model architecture, data generation process, and training procedures used to implement the DeepONet surrogate for elasticity problems in unbounded domains.

2.1 Problem setup

Consider Kelvin's problem in elasticity, which models the equilibrium state of an infinite, homogeneous, linearly elastic, isotropic full-space subject to a point load (Fig. 1). The state of the system, $\mathbf{u} = \mathbf{u}(\mathbf{x})$, is described by the Navier-Cauchy equation

$$(\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) + \mu\nabla^2\mathbf{u} + \mathbf{f} = \mathbf{0}, \tag{1}$$

in which λ and μ are the Lamé parameters and $\mathbf{f}$ is the vector of body forces. As the body is only subject to a static point load, the body force term $\mathbf{f}$ can be modeled as

$$\mathbf{f} = \mathbf{P}\delta(\mathbf{x}). \tag{2}$$

Here, $\mathbf{P} = [f, 0, 0]^{\mathrm{T}}$ acts on the origin and $\delta(\mathbf{x})$ is the 3D Dirac delta generalized function. Kelvin's proposed solution for the displacement $\mathbf{u} = [u_1, u_2, u_3]^{\mathrm{T}}$ in Eq. 1 yields [9]:

$$\begin{aligned} u_1 &= \frac{f}{16\pi\mu(1-\nu)}\left(\frac{2(1-2\nu)}{||\mathbf{x}||} + \frac{1}{||\mathbf{x}||} + \frac{x_1^2}{||\mathbf{x}||^3}\right) \\ u_2 &= \frac{f x_1 x_2}{16\pi\mu(1-\nu)||\mathbf{x}||^3} \\ u_3 &= \frac{f x_1 x_3}{16\pi\mu(1-\nu)||\mathbf{x}||^3}. \end{aligned} \tag{3}$$

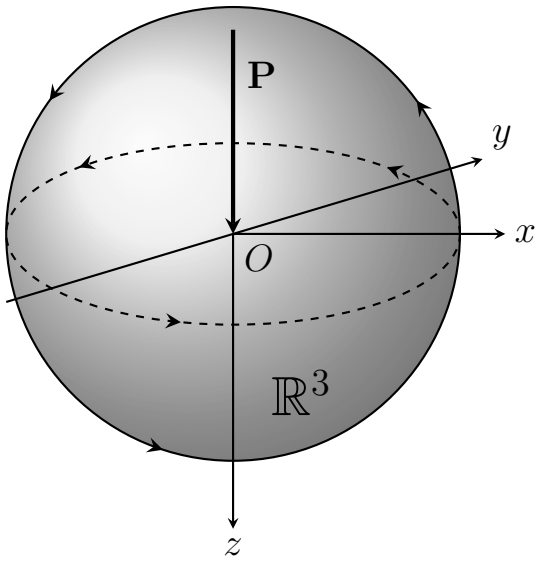


Figure 1. The Kelvin problem in elasticity.

Without loss of generality, we model the partial differential equation in Eq. 1 as a member of the family of PDEs taking the form

$$(L_{\mu,\lambda}\mathbf{u})(\mathbf{x}) = \mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{4}$$

with the operator $L_{\mu,\lambda} : \mathcal{A}_\mu \times \mathcal{A}_\lambda \to L(\mathcal{U} : \mathcal{U}^*)$ mapping from the combined parameter space (in this case, $\mathbb{R}^2$) to the space of linear operators mapping $\mathcal{U}$ to its dual $\mathcal{U}^*$. Here, $\mathcal{U}$ is a Banach space, $\mathcal{U} \ni \mathbf{u} : \Omega \to \mathbb{R}^{d_\Omega}$, $\mathbf{f} \in \mathcal{U}^*$, $\mathcal{A}_\mu, \mathcal{A}_\lambda \subset \mathbb{R}$ and $\Omega \subset \mathbb{R}^{d_\Omega}$ is a manifold of dimension d_Ω. In this case, $d_\Omega = 3$.

From this formulation, we define the operator

$$\mathcal{G} := L_{\mu,\lambda}\mathbf{f} : \mathcal{A}_\mu \times \mathcal{A}_\lambda \to \mathcal{U} \tag{5}$$

as a map from the parameters to the solution: $(\mu \times \lambda) \mapsto \mathbf{u}$. Note that using both Lamé parameters and Poisson's ratio in the problem formulation introduces redundancy to this parameter-based approach, as ν is deterministically

derived from λ and μ. It may be interesting, however, to select only Poisson's ratio and the second Lamé parameter, μ, for our modeling. We argue for this strategy for three main reasons:

1. ν is a parameter that material scientists and engineers typically use to characterize materials, adding interpretability to data-driven approaches;
2. ν translates directly to a measurable physical quantity, as it represents transverse-to-axial strain ratio. The same is true for μ, as it corresponds directly to the material's shear modulus. λ, however, despite its relationship with the material's bulk modulus, encloses a less straightforward physical interpretation;
3. ν values are bounded within the interval $[-1, 0.5)$, while λ values are unbounded above (the value of λ can be indefinitely large as the material approaches incompressibility).

Hence, from now on we will use parameter sets A_ν and A_μ as the elastic medium's material parameters. We then define the set encoding the operator's input as: $\mathcal{E} := \mathcal{A}_\mu \times \mathcal{A}_\nu$ and the mapping defined in Eq. 5 is thus modified to

$$\begin{aligned} \mathcal{G} := L_{\mu,\nu} : \mathcal{E} &\to \mathcal{U} \\ (\mu \times \nu) &\mapsto \mathbf{u}. \end{aligned} \tag{6}$$

Essentially, our goal is now to learn this particular mapping between a finite-dimensional space to an infinite-dimensional space. This is accomplished by analyzing a finite number of observed input-output pairs from this mapping.

Consider a dataset composed of observations $\mathcal{D} = \{(\boldsymbol{\xi_i}, \mathbf{x}_j, \mathbf{u}_{ij})\}_{i=1,j=1}^{N_\mathcal{E}, N_\Omega}$, where each $\xi := (\mu, \nu) \in \mathcal{E} \subset \mathbb{R}^{d_\mathcal{E}}$ ($d_\mathcal{E}$ is the parameter manifold's dimension) is sampled from a probability distribution. Here, $\mathbf{u}_{ij} = \mathcal{G}(\boldsymbol{\xi}_i)(\mathbf{x}_j)$, which represents the true value of the output function $\mathcal{G}(\boldsymbol{\xi}_i)$ evaluated at point $\mathbf{x}_j$. In other words, our objective is to approximate $\mathcal{G}$ using a parametric map

$$\mathcal{G}_{\boldsymbol{\theta}} : \mathcal{E} \to \mathcal{U}, \quad \boldsymbol{\theta} \in \mathbb{R}^p \tag{7}$$

such that we find the set of parameters $\boldsymbol{\theta}^*$ that approximates the original PDE operator, $\mathcal{G}_{\boldsymbol{\theta}^*} \approx \mathcal{G}$. Considering the loss function

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{N_\mathcal{E} N_\Omega} \sum_{i=1}^{N_\mathcal{E}} \sum_{j=1}^{N_\Omega} ||\mathbf{u}_{ij} - \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{\xi}_i)(\mathbf{x}_j)||_2^2, \tag{8}$$

with $||\cdot||_2$ denoting the L_2 norm, this can then be formulated as an optimization problem for

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{9}$$

2.2 DeepONet Architecture

The universal approximation theorem for operators [10] allows one to approximate continuous operators using DeepONets [7]. The DeepONet as defined originally in [7] introduces inductive bias by considering two separate neural networks: one taking the operator's input function $\boldsymbol{\xi}$ as input and another one taking the operator's output locations $\mathbf{x}$. This allows one to define two distinct neural networks

$$\begin{aligned} \mathcal{B}(\,\cdot\,; \boldsymbol{\theta}_\mathcal{B}) : \mathcal{E} &\to \mathbb{R}^p \\ \boldsymbol{\xi} &\mapsto \mathbf{b}(\boldsymbol{\xi}), \end{aligned} \tag{10}$$

and

$$\begin{aligned} \mathcal{T}(\,\cdot\,; \boldsymbol{\theta}_\mathcal{T}) : \Omega &\to \mathbb{R}^p \\ \mathbf{x} &\mapsto \mathbf{t}(\mathbf{x}). \end{aligned} \tag{11}$$

We denote $\mathbf{b}(\boldsymbol{\xi}) = [b_1(\boldsymbol{\xi}), \ldots, b_p(\boldsymbol{\xi})]^\mathrm{T}$ and $\mathbf{t}(\mathbf{x}) = [t_1(\mathbf{x}), \ldots, t_p(\mathbf{x})]^\mathrm{T}$ as vectors from an embedding of dimensionality p, while $\boldsymbol{\theta}_\mathcal{B}$ and $\boldsymbol{\theta}_\mathcal{T}$ are the training parameters of each neural network. Next, the architecture that models the output of parametric operator described in Eq. 7 can be defined as

$$(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{\xi}))(\mathbf{x}) = \sum_{k=1}^{p} b_k(\boldsymbol{\xi}; \boldsymbol{\theta}_\mathcal{B}) t_k(\mathbf{x}; \boldsymbol{\theta}_\mathcal{T}), \tag{12}$$

with $\boldsymbol{\theta} = \{\boldsymbol{\theta}_\mathcal{B}, \boldsymbol{\theta}_\mathcal{T}\}$. The $\mathcal{B}$ and $\mathcal{T}$ neural networks are named the branch and trunk networks, respectively. They provide the capacity of the DeepONet model to approximate a multitude of operators commonly encountered

in physical and engineering applications [7]. We must note, however, that effective neural network training is essential to ensure model reliability, maximize predictive performance, and prevent detrimental outcomes caused by model overfitting or underfitting. Therefore, an effective training mechanism must be available to leverage the expressivity of DeepONets. The architecture as originally defined in [7] will henceforth be referred to as the *vanilla* DeepONet.

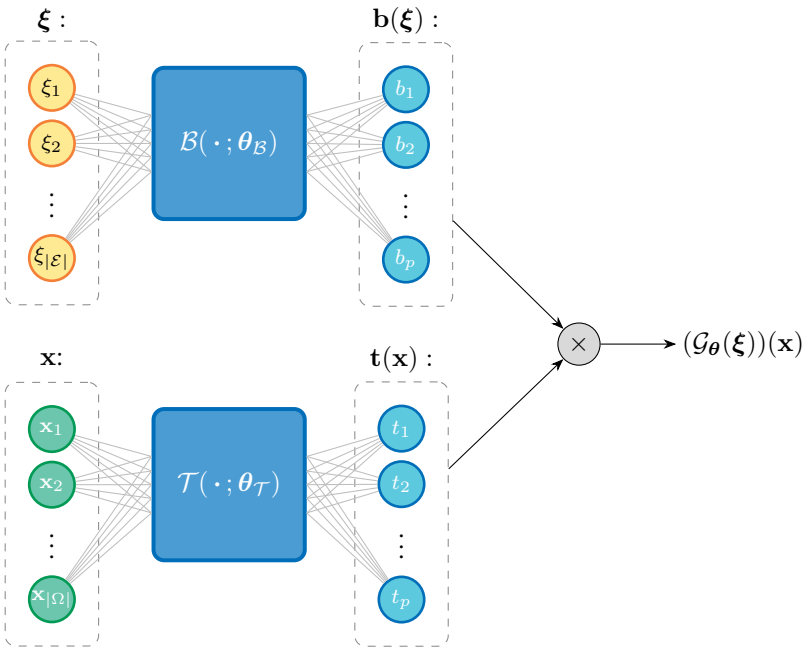


Figure 2. Architecture of the DeepONet.

In the trunk network, the input consists of the spatial dimensions $\mathbf{x}$, while the branch network encodes the different input functions corresponding to the operator's input function (e.g., medium's material parameters μ and ν). In this case, the DeepONet is trained using a dataset $\mathcal{D}$ of input-output data, which is divided into 80% training, 10% validation, and 10% testing splits. This training aims to solve the optimization problem defined in Eq. 13, which, by incorporating Eq. 12, can be rewritten as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbb{R}^p}{\arg\min}\, \frac{1}{N_\mathcal{E} N_\Omega} \sum_{i=1}^{N_\mathcal{E}} \sum_{j=1}^{N_\Omega} ||\mathbf{u}_{ij} - \sum_{k=1}^{p} b_k(\boldsymbol{\xi}_i;\boldsymbol{\theta}_\mathcal{B}) t_k(\mathbf{x}_j;\boldsymbol{\theta}_\mathcal{T})||_2. \tag{13}$$

Let $\mathbf{U} = (\mathbf{u}_{ij}) \in \mathbb{R}^{N_\mathcal{E}\times N_\Omega}$, $\mathbf{B}(\boldsymbol{\theta}_\mathcal{B}) = \left[\mathbf{b}(\boldsymbol{\xi}_1)^\mathrm{T}, \mathbf{b}(\boldsymbol{\xi}_2)^\mathrm{T}, \ldots, \mathbf{b}(\boldsymbol{\xi}_{N_\mathcal{E}})^\mathrm{T}\right]^\mathrm{T} \in \mathbb{R}^{N_\mathcal{E}\times p}$ and $\mathbf{T}(\boldsymbol{\theta}_\mathcal{T}) = \left[\mathbf{t}(\mathbf{x}_1)^\mathrm{T}, \mathbf{t}(\mathbf{x}_2)^\mathrm{T}, \ldots, \mathbf{t}(\mathbf{x}_{N_\Omega})^\mathrm{T}\right]^\mathrm{T} \in \mathbb{R}^{N_\Omega\times p}$. The loss function in Eq. 8 can then be redefined as

$$\mathcal{L}(\boldsymbol{\theta}) := ||\mathbf{U} - \mathbf{B}(\boldsymbol{\theta}_\mathcal{B})\mathbf{T}(\boldsymbol{\theta}_\mathcal{T})^\mathrm{T}||_F^2, \tag{14}$$

in which now $||\cdot||_F$ denotes the Frobenius norm. The DeepONet is then trained with Eq. 9 using the loss function defined in Eq. 14.

While the vanilla DeepONet employs the trunk network to learn the basis of the output function from the data, an alternative approach uses basis functions created beforehand through Proper Orthogonal Decomposition (POD) of the training data, which then serve as a substitute for the trunk net. A neural network is then employed as the branch network to learn the POD basis' coefficients. Let $\phi_0(\mathbf{x})$ be the dataset's mean and $\Phi = \{\phi_1(\mathbf{x}), \ldots, \phi_p(\mathbf{x})\}$ the set of precomputed basis functions, Eq. 12 can be rewritten as

$$(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{\xi}))(\mathbf{x}) = \sum_{k=1}^{p} b_k(\boldsymbol{\xi};\boldsymbol{\theta}_\mathcal{B})\phi_k(\mathbf{x}) + \phi_0(\mathbf{x}). \tag{15}$$

2.3 Data Generation

From the formulation previously defined, we sample $N_\mathcal{E} = 300$ sets of each of the parameters $\mathcal{E}_i$ ($i = 1, 2, \ldots, d_\mathcal{E}$) according to the following rules:

1. The applied load f is set to 100 kN, which encompasses the order of magnitude of large structures and machines (e.g., large-scale wind turbines). f is not included as a training parameter because the problem is simply linearly dependent on the loading magnitude;
2. The logarithm of the shear modulus μ is sampled from a uniform distribution $\mathcal{U}_{[6,8]}$, yielding an interval of $[10^6, 10^8]$ Pa or 1 MPa to 100 MPa, including a wide range of soil materials, including some of practical engineering applications such as sand and clay;
3. Poisson's ratio ν is sampled from a uniform distribution $\mathcal{U}_{[0,0.5)}$, which encompasses most materials used in engineering applications. Here, we excluded idealized incompressible solids possessing a ν value of 0.5.

From these distributions we sample the parameters of the PDE's definition, forming a dataset $\boldsymbol{\Xi}$ with shape $(N_{\mathcal{E}} \times d_{\mathcal{E}})$.

Since the problem's domain is unbounded, we must address the challenge of sampling the trunk network's input $\mathbf{x} = [x_1, x_2, x_3]^{\mathrm{T}}$ within a finite region. The problem's operator must satisfy the Sommerfeld radiation condition, which implies solution decay as $|\mathbf{x}| \to \infty$. Since we know from Eq. 3 that $\mathbf{u}$ decays at $\mathcal{O}\left(||\mathbf{x}||^{-1}\right)$, an initial approach consists of defining a cutoff radius r. To keep grid resolution low and save computational resources, $r = 1$ was initially chosen. We then sample $N_\Omega = 8000$ points in this uniform 3D grid. The result is a dataset $\mathbf{X}$ with shape $(N_\Omega \times d_\Omega)$, where each row is a collocation point consisting of a vector $[\mathbf{x}_1, \ldots, \mathbf{x}_{d_\Omega}]$.

Finally, the solution is obtained by evaluating Eq. 3 at each point of the $\mathbf{X}$ grid for each sample in $\boldsymbol{\Xi}$, yielding a $(N_{\mathcal{E}} \times N_\Omega)$ training dataset.

2.4 Optimization

All models were implemented using PyTorch and trained locally on a Linux machine without Graphical Processing Unit (GPU) hardware acceleration. This significantly limited training efficiency. This is particularly important in the case of three-dimensional problems.

Three neural network architectures were tested for both branch and trunk networks: *Multi-Layer Perceptrons* (MLPs), *ResNets* [11], and *Kolmogorov–Arnold Networks* (KANs) [12] utilizing *SiLU* activation functions and Chebyshev polynomials, respectively. Among these, ResNets achieved the best accuracy and shortest training time. The ADAM optimizer with step-scheduling was employed for parameter updates and the mean squared error was used as the objective function.

3 Results and Discussion

We now evaluate the performance of the proposed DeepONet models, comparing training strategies, assessing accuracy across displacement components, and analyzing inference efficiency and error distribution.

Table 1 presents the best-performing models for the two implemented training strategies. These results are presented in terms of the relative error between the solution obtained with the DeepONet with respect to the analytical solution of the Kelvin problem, considering displacements in the $i = x, y, z$–directions.

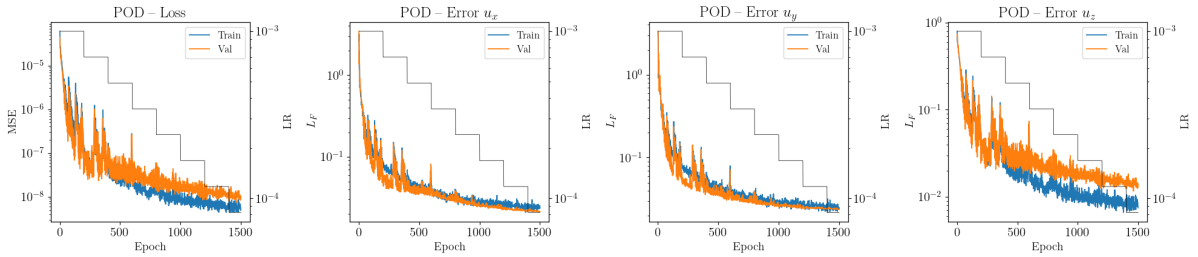
Training Strategy	Test error (u_x) [%]	Test error (u_y) [%]	Test error (u_z) [%]
Vanilla	3.49	3.44	3.89
POD	3.53	3.60	1.29

Table 1. Testing relative error [%] for best performing DeepONet setups.

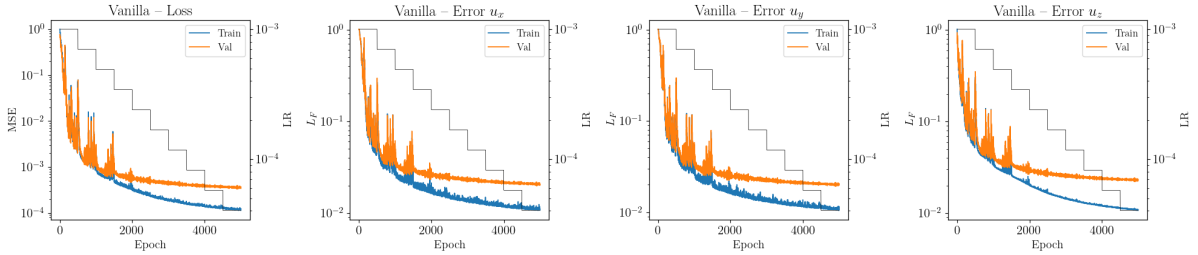
These results show that the two strategies perform similarly, with relative errors inferior to 4% across the board. It is interesting to note, however, that the POD strategy outperforms to some degree the vanilla strategy for the component that is parallel to the load vector $\mathbf{P}$ (u_z). Our leading hypothesis to explain this phenomenon is that the problem's axisymmetry (Fig. 1) induces a larger share of the displacement's variance in the load direction. This is a direct consequence of applying Proper Orthogonal Decomposition in the operator's basis, which chooses dominant eigenvectors from the covariant matrix for model reduction. For this reason, early gradients would tend to favor the direction of steepest decrease for u_z during optimization. Additionally, the results for the POD strategy are achieved in a shorter training time (60 s) than the vanilla strategy (2950 s). The POD-DeepONet model also exhibited a remarkably low inference latency of 3 ms on the testing set, while the vanilla model took approximately 169 ms to process all testing samples.

Figure 3 presents the training history for both models. The first column shows the mean squared error value for the training and validation sets during optimization, while the remaining columns show the relative error of

the euclidean norm for each component of the solution vector. Despite the presence of noise, particularly in the beginning of training, we can observe that the models achieved satisfactory convergence for all performance metrics.



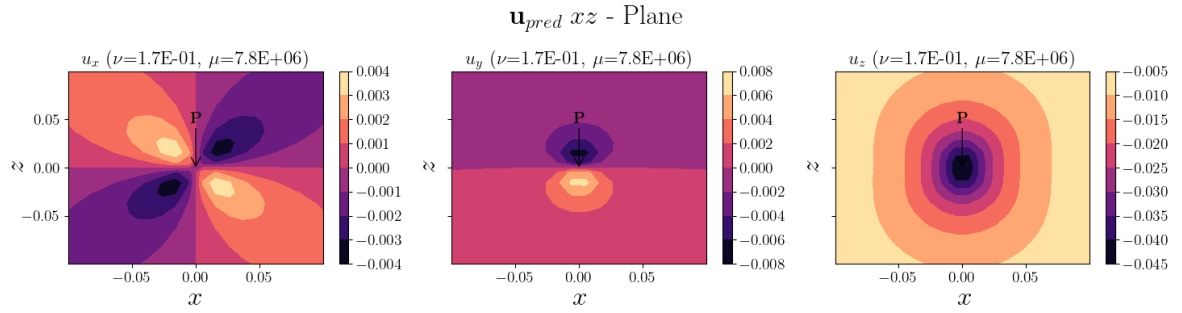
(a) Training metrics for POD-DeepONet.



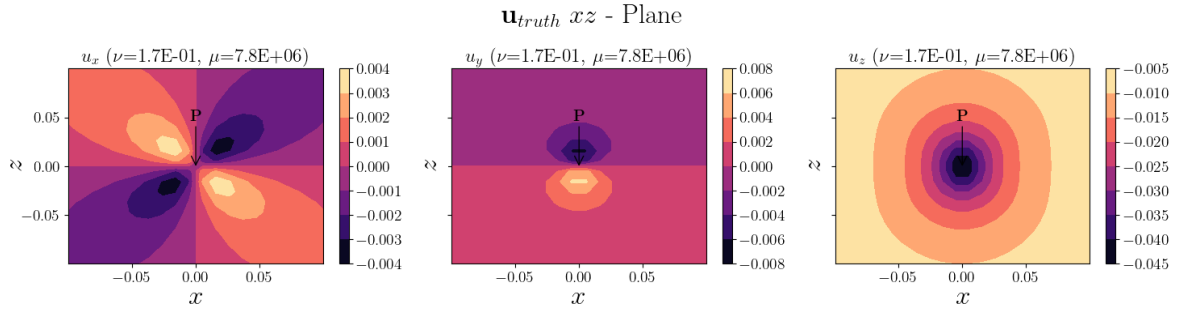
(b) Training metrics for Vanilla DeepONet.

Figure 3. Comparison of training metrics for POD-DeepOnet and Vanilla DeepOnet. The secondary axis on the right of each plot represents the step-varying learning rate.

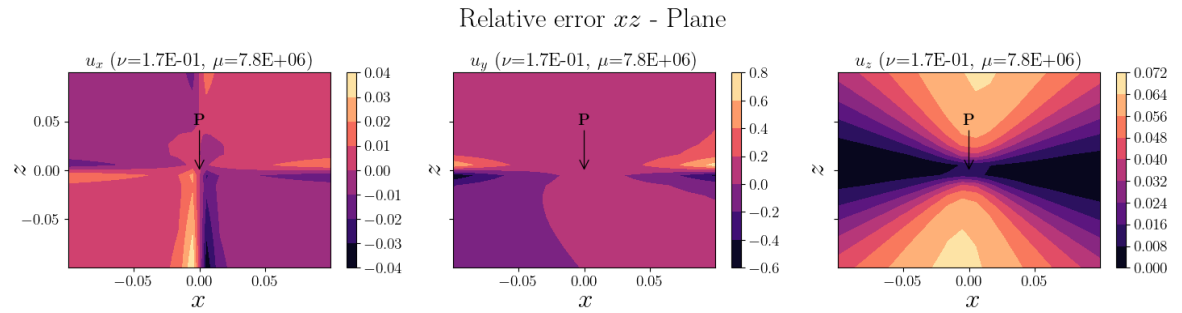
Finally, Fig. 4 presents a sample of the errors of the best performing model for each component on the XZ-plane (which is parallel to the source load). Each column represents the displacement field's components in the $i = x, y, x$ - directions, respectively. The first row is the model's predicted value when given a sample input function (in this case, $\nu = 0.17, \mu = 7.8 \times 10^6$ MPa, the second row is the actual value for these samples, computed from Eq. 3, and the third row is the relative error between the true and predicted values. The close alignment between the predicted and actual values, coupled with the consistently low relative errors across each component (u_x, u_y, u_z), demonstrates the model's capacity to effectively learn the underlying operator behavior for this multi-dimensional output.



(a) Prediction for $\nu = 0.17$, $\mu = 7.8 \times 10^6$ MPa.



(b) Ground truth for $\nu = 0.17$, $\mu = 7.8 \times 10^6$ MPa.



(c) Relative error for $\nu = 0.17$, $\mu = 7.8 \times 10^6$ MPa.

Figure 4. Comparison of predictions, ground truths, and relative errors on the XZ-plane for $\nu = 0.17$, $\mu = 7.8 \times 10^6$ MPa. The distances are in meters.

4 Conclusions

This work addressed the challenge of solving continuum mechanics problems in unbounded domains, where traditional methods like the Boundary Element Method (BEM) face severe scalability limitations due to dense kernel evaluations. To overcome this, we proposed using DeepONet architectures as efficient, data-driven surrogates.

Our results demonstrate that DeepONet models show promising performance as surrogate models in problems where BEM is classically employed. All models achieved sub-5% relative L_2 error across all displacement components (u_x, u_y, u_z), with the POD-enhanced model displaying particular accuracy computing displacements in the load direction (u_z). Furthermore, the POD-enhanced strategy led to satisfactory inference latency (3 ms), highlighting its superior efficiency.

Acknowledgements. The research leading to this paper has been funded in part by the São Paulo Research Foundation (Fapesp) through grants number 2022/20753-5, 2023/16399-1 and 2024/05277-5. The support of Unicamp is also gratefully acknowledged.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] F. M. Minhós and H. A. S. Carrasco. *Higher order boundary value problems on unbounded domains: types of solutions, functional problems and applications*, volume 5. World Scientific, 2017.

[2] A. Sommerfeld. *Partial differential equations in physics*, volume 1. Academic press, 1949.

[3] W. Thomson. Note on the integration of the equations of equilibrium of an elastic solid. *Cambridge and Dublin Mathematical Journal*, vol. 3, pp. 87–89, 1848.

[4] S. C. Dutta and R. Roy. A critical review on idealization and modeling for interaction among soil–foundation–structure system. *Computers & structures*, vol. 80, n. 20-21, pp. 1579–1594, 2002.

[5] N. A. Gumerov and R. Duraiswami. Fmm accelerated bem for 3d laplace & helmholtz equations. In *Proc. Int. Conf. on Boundary Element Techniques*, volume 7, pp. 79–84, 2006.

[6] J. Labaki, L. O. S. Ferreira, and E. Mesquita. Constant boundary elements on graphics hardware: a gpu-cpu complementary implementation. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 33, pp. 475–482, 2011.

[7] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, vol. 3, n. 3, pp. 218–229, 2021.

[8] B. Ahmed, Y. Qiu, D. W. Abueidda, W. El-Sekelly, de B. G. Soto, T. Abdoun, and M. E. Mobasher. Physics-informed deeponet with stiffness-based loss functions for structural response prediction. *arXiv preprint arXiv:2409.00994*, vol. -, 2024.

[9] A. Favata. On the kelvin problem. *Journal of Elasticity*, vol. 109, pp. 189–204, 2012.

[10] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, vol. 6, n. 4, pp. 911–917, 1995.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. -, pp. 770–778, 2016.

[12] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, vol. -, 2024.