GAMMAS 2026, Vol. 1, No. 1, p. 10–22 DOI: 10.14464/gammas.v8i1.962

International Association
of Applied Mathematics and Mechanics
– Archive for Students –

PI-GEFN: A Physics-Informed Graph Edge Filter Network inspired by finite element formulation

A. Potnis[a,⋆], D. Anton[a], H. Wessels[a]

[a] Institute of Applied Mechanics, Division Data-Driven Modeling of Mechanical Systems, Technische Universität Braunschweig, Pockelsstr. 3, 38106 Braunschweig, Germany

received 20.12.2025, accepted 06.04.2026, published 10.04.2026

⋆ corresponding author: a.potnis@tu-braunschweig.de

Abstract: *This work introduces **PI-GEFN**, a Physics-Informed Graph Edge Filter Network inspired by finite element methods, designed to solve both forward and inverse problems in linear elasticity. Unlike conventional graph neural networks that abstract physical relationships, PI-GEFN integrates physically meaningful stiffness contributions as edge features and incorporates residual-based loss functions derived from the finite element system. This architecture enables the surrogate model to preserve the equilibrium characteristics of the physical system while remaining scalable and data-efficient. We demonstrate the model's effectiveness across various parametric regimes, including geometry variations, material properties, and Neumann boundary conditions, while achieving high accuracy in displacement predictions and parameter identification. The model performs robustly even in a limited data regime and demonstrates competitive accuracy and convergence with physics-informed neural network methods. These qualities position PI-GEFN as a versatile and physically grounded surrogate modeling tool for real-time simulation, design, and inverse identification tasks in computational mechanics.*

Keywords: Graph neural network, finite element method, inverse problem, linear elasticity

1 Introduction

Graph Neural Networks (GNNs) have emerged as a transformative approach for addressing complex problems in mechanics, owing to their inherent ability to handle irregular, mesh-based data and incorporate physical inductive biases. The review paper [22] presents a comprehensive overview of recent advancements in this domain, highlighting the shift from conventional fully connected neural network surrogates to the discrete value-based neural network surrogates. Notably, developments such as message passing neural networks [10], graph attention networks [20], and equivariant GNNs [12] have significantly improved the ability to predict mechanical responses under varying loading conditions and geometries. The incorporation of domain knowledge, such as conservation laws and boundary conditions, into GNN training has led to improved generalization, robustness, and data efficiency. Furthermore, the paper [22] discusses applications across elasticity, plasticity, fracture mechanics, and meta-material design, illustrating the versatility and growing maturity of GNN-based models in computational mechanics. These advancements set a strong foundation for future research into real-time simulation, inverse problems, and generalized solvers for diverse

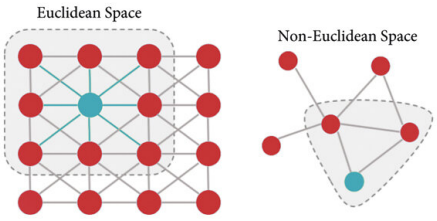

Figure 1 – Structured grid versus graph-based (unstructured) data representations [2]

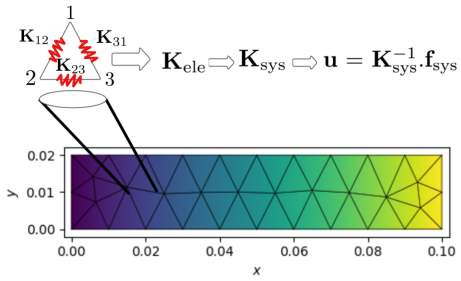

Figure 2 – FE motivation visualization

mechanical systems.

Recent literature has explored diverse applications of GNNs across solid mechanics, ranging from elasticity and fracture to topology optimization and design of meta-materials. For instance, in [9], the Physics-Informed Graph Neural Galerkin Network framework was proposed, which unifies forward and inverse problem-solving by incorporating Galerkin principles directly into the GNN loss formulation, achieving high accuracy in solving partial differential equation (PDE)-governed problems. In [6], the Multichannel Aggregation Network, a GNN surrogate model using a graph U-Net architecture, was introduced to predict large deformation behavior in hyperelastic solids, outperforming traditional finite element method (FEM) approaches in computational efficiency. Similarly, in [13], a multiscale GNN with adaptive mesh refinement was developed for phase-field fracture modeling, mimicking multigrid solvers to capture both global and local damage evolution. In [23], the use of a graph-based variational autoencoder for both linear and nonlinear inverse design of 3D truss meta-materials was demonstrated, achieving high-fidelity reconstructions and property interpolation in latent space. Across these works, a common trend is the integration of FEM-based priors ([9, 14]), multi-resolution graph structures ([13]), and physics-informed losses ([9]) to enhance generalizability, data efficiency, and solution accuracy, underscoring the increasing maturity of GNNs in computational mechanics. Graph-based learning offers a powerful alternative to conventional neural networks like convolutional neural networks, which are primarily designed for data arranged on regular, grid-like structures commonly referred to as structured grids (e.g., images, where pixels are evenly spaced). However, many real-world mechanics problems, such as those involving finite element meshes or material microstructures, do not conform to such regular grids. Instead, their data reside in irregular or unstructured grids. These unstructured grids can be more naturally represented as graphs, where each node might correspond to a mesh point or physical location, and edges capture relationships such as connectivity, stiffness, or proximity. As shown in Figure 1, the left side represents a structured grid, whereas the right side illustrates an unstructured, graph-based representation often encountered in finite element meshes. This shift in data structure is one of the key reasons why GNNs have become increasingly relevant for modeling physical systems. An important architectural advancement that enables physics-aware message passing is the concept of edge-conditioned filters, where the message-passing weights are dynamically generated from edge features using a small neural network. This idea was first introduced in [18] in the context of point cloud processing and has since been adopted in various domains. However, its application to mechanical systems governed by PDEs, especially in conjunction with finite element stiffness information, remains relatively unexplored. Our proposed architecture builds on this idea, but tailors it specifically for FEM-inspired edge features, bridging the gap between classical stiffness assembly and graph-based learning. The design of our proposed framework, PI-GEFN, is rooted in a fundamental motivation: to preserve the physical meaning of stiffness as known in the FEM and embed it directly into a learnable edge-conditioned Graph Convolutional Neural Network (GCN), i.e., a specific class of graph neural network architecture. Although GCNs are powerful for learning over arbitrary meshes and graph domains, they often lose direct contact with the mechanical interpretation of interactions between nodes. PI-GEFN is an attempt to bring back this missing link by treating graph edges not just as abstract connections but as physical entities with stiffness values derived from FEM principles.

In classical FEM, the stiffness of an element, such as a triangle in a 2D mesh, contributes to the global stiffness matrix $\mathbf{K}_{sys}$, which governs the relationship between nodal displacements $\mathbf{u}$ and external forces $\mathbf{f}$ as $\mathbf{K}_{sys}\mathbf{u} = \mathbf{f}$. In our approach, we revisit this stiffness assembly and treat each element's contribution as a localized spring system. As illustrated in Figure 2, the elemental stiffness contributions (e.g., $\mathbf{K}_{12}, \mathbf{K}_{23}, \mathbf{K}_{31}$) can be conceptually

viewed as springs connecting adjacent nodes, thereby forming a direct analogy to edge features in a GCN.

This motivates the core idea of using the elemental stiffness entries derived from FEM integration as edge features in our GCN. When converting a FEM mesh into a graph, each triangle is treated as a localized system contributing to edges between its vertices. Instead of relying on learned attention or geometric distances alone, we encode the stiffness contributions $\mathbf{K}_{ij}$ as physically meaningful attributes on each edge (i, j), consistent with the assembly process of the global matrix.

By doing this, we bridge the theoretical gap between the physics-based discretization of PDEs and the data-driven learning capability of GCNs. Each edge now carries a stiffness-informed weight, and each node updates its state in a way that mimics the equilibrium propagation seen in FEM. The learning is further guided by a residual-based loss term of the form $\|\mathbf{K}_{\text{sys}}\mathbf{u} - \mathbf{f}\|_2$, ensuring that the model not only fits data but also respects underlying mechanical laws.

The proposed PI-GEFN framework offers several distinct advantages over traditional fully connected neural networks (FC-NNs) and many physics-informed or data-driven models discussed above. These advantages are both computational and architectural, making the method particularly suitable for high-fidelity mechanics problems involving complex domains. A few of the most prominent benefits are summarized below:

- **Structure-aware representation:** Unlike FC-NNs, which treat each input as an independent point, our method exploits the underlying mesh structure via graph connectivity. This enables the model to preserve geometric and topological fidelity of the problem domain, leading to more physically accurate learning and generalization.
- **No need for explicit spatial derivatives of the displacement field:** In contrast to methods that require first- or second-order spatial derivatives of the displacement field with respect to the physical coordinates (e.g., Physics-Informed Neural Networks (PINNs) or variational graph neural networks), our framework avoids computing such derivatives explicitly. This reduces computational complexity and avoids errors associated with finite differencing, symbolic differentiation, or automatic differentiation of spatial operators. Standard gradients with respect to the trainable network parameters are still required during optimization.
- **Precomputed integration with standard shape functions:** Element-wise matrices are built using classical finite element routines with isoparametric shape functions and Gauss quadrature. These quadrature points are fixed and precomputed, and hence the stiffness matrix $\mathbf{K}$ and force vector $\mathbf{f}$ can be assembled once and reused. This eliminates the need for repetitive numerical integration during training, which can be a significant computational bottleneck in weak-form or variational physics-informed approaches and related meshless methods.
- **Physics-informed edge and node features:** By embedding FEM-based stiffness components into edge and node features (e.g., using Mandel representation of $\mathbf{K}_{ij}$ and $\mathbf{K}_{ii}$), we incorporate physical inductive bias directly into the GCN architecture. This improves convergence and reduces the number of learnable parameters, leading to faster and more stable training.
- **Recovery of continuous displacement fields and resultants:** Once the discrete displacement values at the nodes are predicted by the GCN, they can be seamlessly extended to a continuous displacement field using the same shape functions used during FEM assembly. This provides a physically consistent and smooth ansatz for the displacement field. Moreover, element-wise stresses (i.e., resultants) can be computed directly from these discrete values using the standard post-processing pipeline namely, by applying the constitutive relation $\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}$. Here, $\mathbb{C}$ denotes the fourth-order elasticity tensor. In the numerical implementation, symmetry is exploited and the tensor is represented in reduced Mandel notation, resulting in a matrix representation consistent with standard finite element formulations. This eliminates the need for explicit derivative computation during training, while still allowing full-field stress recovery in post-processing.
- **Efficient handling of Dirichlet boundary conditions:** We employ a hard-enforcement scheme for essential (Dirichlet) boundary conditions by using a boundary mask applied at the output layer. This avoids penalty-based soft constraints [16, 24] and ensures that the displacement boundary conditions are exactly satisfied.
- **Minimized need for collocation points:** Since our model operates on a mesh-based graph with encoded physical properties, there is no requirement for dense sampling of collocation points across the domain. A single graph per simulation suffices, unlike PINNs which often require thousands of collocation points to achieve acceptable accuracy.

- **Parametric generalization:** Our training is conducted over a dataset of graphs generated from varying material parameters and boundary conditions, allowing the model to generalize across different stiffness and loading regimes. This parametric training approach is missing in many works (e.g., [3, 5, 9]), which focus on single-scenario prediction.
- **Improved data efficiency and scalability:** Compared to FC-NNs that require retraining or interpolation for new geometries or material conditions, our graph-based method is inherently scalable and data-efficient. The node-wise and edge-wise modeling aligns naturally with real-world FEM meshes.
- **Reusability and modularity:** Since the quadrature weights, Jacobians, and shape function gradients are computed once per element, they can be stored and reused across multiple training epochs. This modularity makes the pipeline both memory-efficient and consistent with standard FEM software.

2 Methodology

2.1 Forward Problem

This section describes the formulation of PI-GEFN for solving the forward problem in linear elasticity. For uniformity, we define the state equation consisting of the equilibrium equation and the associated boundary conditions under parameterization $\boldsymbol{\kappa}$. The displacement field is denoted by $\mathbf{u} : \Omega \to \mathbb{R}^2$, with Cartesian components u_x and u_y. The physical domain is given by $\Omega \subset \mathbb{R}^2$, with Dirichlet and Neumann boundaries $\Gamma_\mathrm{D}, \Gamma_\mathrm{N} \subset \partial\Omega$. The vector $\mathbf{b}$ denotes the body force per unit volume, $\mathbf{t}$ the prescribed traction on Γ_N, and $\mathbf{n}$ the outward unit normal on Γ_N. The stress tensor is written as $\boldsymbol{\sigma}(\mathbf{u}, \boldsymbol{\kappa})$, and $\mathbf{u}_\mathrm{D}$ refers to prescribed displacements on Γ_D.

$$\mathbf{F}^*(\mathbf{u}, \boldsymbol{\kappa}) = \begin{Bmatrix} \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, \boldsymbol{\kappa}) + \mathbf{b} & \text{in } \Omega \\ \mathbf{u}_\mathrm{D} - \mathbf{u} & \text{on } \Gamma_\mathrm{D} \\ \boldsymbol{\sigma}(\mathbf{u}, \boldsymbol{\kappa})\mathbf{n} - \mathbf{t} & \text{on } \Gamma_\mathrm{N} \end{Bmatrix} = \mathbf{0}. \tag{1}$$

The objective of PI-GEFN is to train the parameters $\boldsymbol{\theta}$ of the EdgeNet components such that the network approximates the displacement field. In the discrete setting, this reads

$$\begin{aligned} u_x(\tilde{x}, \tilde{y}, \boldsymbol{\kappa}) &\approx u_x^{\mathrm{pred}}(\tilde{x}, \tilde{y}, \boldsymbol{\kappa}; \boldsymbol{\theta}), \\ u_y(\tilde{x}, \tilde{y}, \boldsymbol{\kappa}) &\approx u_y^{\mathrm{pred}}(\tilde{x}, \tilde{y}, \boldsymbol{\kappa}; \boldsymbol{\theta}), \end{aligned} \tag{2}$$

where $(\tilde{x}, \tilde{y})$ denote nodal coordinates of the mesh, $\boldsymbol{\kappa}$ collects parametric inputs (e.g., material parameters and boundary conditions), and $\boldsymbol{\theta}$ are trainable network parameters.

After finite element discretization, the displacement field is represented by the algebraic degree-of-freedom vector

$$\mathbf{u} \in \mathbb{R}^{2N},$$

collecting the two displacement components at each of the N mesh nodes. With slight abuse of notation, we use the same symbol $\mathbf{u}$ for the discrete vector. The associated finite element system reads

$$\mathbf{K}_{\mathrm{sys}}\mathbf{u} = \mathbf{f}.$$

The assembled finite element system also defines the graph inputs used by PI-GEFN: for each node i, the diagonal stiffness block $\mathbf{K}_{ii}$ together with the nodal load entry $\mathbf{f}_i$ forms the node feature vector, while each off-diagonal block $\mathbf{K}_{ij}$ is used as the edge attribute for a connected node pair (i, j).

The forward-training objective is defined as

$$\begin{aligned} \mathscr{L}_{\mathrm{fwd}}(\boldsymbol{\theta}) = \lambda_{\mathrm{sup}} \left\| \mathbf{D}_{\mathrm{sup}} \left(\mathbf{S}\mathbf{u}_{\mathrm{pred}}(\boldsymbol{\theta}) - \mathbf{u}_{\mathrm{true}} \right) \right\|_2^2 \\ + \lambda_{\mathrm{res}} \left\| \mathbf{D} \left(\mathbf{K}_{\mathrm{sys}}\mathbf{u}_{\mathrm{pred}}(\boldsymbol{\theta}) - \mathbf{f} \right) \right\|_2, \end{aligned} \tag{3}$$

where $\mathbf{u}_{\mathrm{pred}}(\boldsymbol{\theta}) \in \mathbb{R}^{2N}$ denotes the predicted global displacement vector. The matrix $\mathbf{S} \in \mathbb{R}^{m \times 2N}$ is a Boolean selection operator that extracts degrees of freedom associated with a prescribed fraction $p \in (0, 1]$ of mesh nodes for which displacement labels are available. If N denotes the total number of nodes, then $m \approx 2pN$. If pN is not an integer, the number of supervised nodes is rounded up to the next integer in practice. In this work, we set $p = 0.2$. Accordingly, the supervised reference displacement vector $\mathbf{u}_{\mathrm{true}} := \mathbf{S}\mathbf{u}_{\mathrm{FEM}} \in \mathbb{R}^m$ is obtained by restricting the FEM reference solution $\mathbf{u}_{\mathrm{FEM}} \in \mathbb{R}^{2N}$ to the supervised degrees of freedom.

For better readability, we define

$$\begin{aligned} \mathscr{L}_{\mathrm{sup}} &= \left\| \mathbf{D}_{\mathrm{sup}} \left(\mathbf{S}\mathbf{u}_{\mathrm{pred}} - \mathbf{u}_{\mathrm{true}} \right) \right\|_2^2, \\ \mathscr{L}_{\mathrm{res}} &= \left\| \mathbf{D} \left(\mathbf{K}_{\mathrm{sys}}\mathbf{u}_{\mathrm{pred}} - \mathbf{f} \right) \right\|_2. \end{aligned}$$

We intentionally use the unsquared L2 norm in $\mathscr{L}_{\mathrm{res}}$. Compared with a squared residual norm, this choice reduces the dominance of occasional large residual entries during training and led to more stable optimization in our experiments. Since the residual is not expected to vanish exactly during training, the resulting objective remains differentiable almost everywhere and did

not cause difficulties in practice. To balance the contributions of displacement components with different magnitudes, we introduce a diagonal scaling matrix

$$\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_{2N}),$$

with

$$d_{2i-1} = 1, \qquad d_{2i} = \alpha, \qquad i = 1, \ldots, N.$$

In this work, we set $\alpha = 10$. The matrix $\mathbf{D}_{\mathrm{sup}} = \mathbf{SDS}^T$ denotes the restriction of $\mathbf{D}$ to supervised degrees of freedom.

This scaling is introduced purely for numerical balancing and does not represent statistical weighting. It is applied consistently to both supervised and residual terms.

Although Equation (1) is stated in strong form, all operators in the loss function are derived from the corresponding weak formulation via standard finite element discretization. The stiffness matrix $\mathbf{K}_{\mathrm{sys}}$, force vector $\mathbf{f}$, and reference solution $\mathbf{u}_{\mathrm{true}}$ are obtained from a high-fidelity finite element solver.

The final optimization problem reads

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathscr{L}_{\mathrm{fwd}}(\boldsymbol{\theta}). \tag{4}$$

2.2 Message Passing with Edge-Conditioned Filters

For each GCN layer $\ell = 1, \ldots, L$, let $\mathbf{h}_j^{(\ell-1)} \in \mathbb{R}^{d_{\ell-1}}$ denote the node feature vector at node j produced by the previous layer (or the input feature vector for $\ell = 1$). Here, $d_{\ell-1}$ and d_ℓ denote the input and output feature dimensions of layer ℓ, respectively, and $\mathbf{e}_{ij} \in \mathbb{R}^{f_e}$ denotes the edge feature vector associated with the edge connecting nodes i and j, where f_e is the dimension of the edge feature vector. An edge-conditioned filter is then computed using an EdgeNet, a lightweight multilayer perceptron (MLP) that dynamically generates filter weights conditioned on edge features [18]:

$$\boldsymbol{\phi}_{ij}^{(\ell)} = \mathrm{EdgeNet}^{(\ell)}(\mathbf{e}_{ij}) \in \mathbb{R}^{d_\ell d_{\ell-1}}$$

The flattened vector $\boldsymbol{\phi}_{ij}^{(\ell)}$ is reshaped to a weight matrix:

$$\mathbf{W}_{ij}^{(\ell)} = \mathrm{reshape}\left(\boldsymbol{\phi}_{ij}^{(\ell)}\right) \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$$

The message from node j to node i is computed as:

$$\mathbf{m}_{ij}^{(\ell)} = \mathbf{W}_{ij}^{(\ell)} \mathbf{h}_j^{(\ell-1)}$$

The aggregated message at node i is given by:

$$\mathbf{h}_i^{(\ell)} = \tanh\left(\frac{1}{|\mathscr{N}(i)|} \sum_{j \in \mathscr{N}(i)} \mathbf{m}_{ij}^{(\ell)}\right).$$

where $\mathscr{N}(i)$ denotes the set of neighboring nodes of node i.

At the final layer, $\mathbf{h}_i^{(L)}$ is mapped to the normalized displacement prediction $\hat{\mathbf{u}}_{\mathrm{pred},i} \in \mathbb{R}^2$.

Finally, Dirichlet boundary conditions are enforced using diagonal binary mask matrices $\mathbf{M}_i \in \{0,1\}^{2\times 2}$, where each diagonal entry indicates whether the corresponding degree of freedom at node i is constrained (1) or free (0). Here, $\hat{\mathbf{u}}_{\mathrm{pred},i}^{\mathrm{BC}}$ denotes the normalized prescribed Dirichlet displacement at node i.

The corrected normalized displacement vector at node i, denoted by $\hat{\mathbf{u}}_{\mathrm{pred},i}^{\mathrm{corr}} \in \mathbb{R}^2$, is defined as

$$\hat{\mathbf{u}}_{\mathrm{pred},i}^{\mathrm{corr}} = (\mathbf{I} - \mathbf{M}_i)\, \hat{\mathbf{u}}_{\mathrm{pred},i} + \mathbf{M}_i\, \hat{\mathbf{u}}_{\mathrm{pred},i}^{\mathrm{BC}}.$$

The corrected normalized predictions are then denormalized element-wise:

$$u_{\mathrm{pred},i}^{(k)} = \tfrac{1}{2}\left(\hat{u}_{\mathrm{pred},i}^{(k)} + 1\right)\left(u_{\max}^{(k)} - u_{\min}^{(k)}\right) + u_{\min}^{(k)}, \quad k \in \{x, y\}.$$

The global displacement vector $\mathbf{u}_{\mathrm{pred}}(\boldsymbol{\theta}) \in \mathbb{R}^{2N}$ used in the loss formulation (Section 2.1) is obtained by assembling the denormalized nodal vectors $\mathbf{u}_{\mathrm{pred},i} = [u_{x,i}, u_{y,i}]^T$ according to the standard finite element degree-of-freedom ordering.

2.3 Error Metrics

Model accuracy is assessed using the relative L2 norm (rL2) and the mean absolute error (MAE):

$$\begin{aligned} \mathrm{rL2} &= \frac{\left\|\mathbf{u}_{\mathrm{pred}} - \mathbf{u}_{\mathrm{true}}\right\|_2}{\left\|\mathbf{u}_{\mathrm{true}}\right\|_2}, \\ \mathrm{MAE} &= \frac{1}{2N} \sum_{i=1}^{N} \left\|\mathbf{u}_{\mathrm{pred},i} - \mathbf{u}_{\mathrm{true},i}\right\|_1. \end{aligned} \tag{5}$$

rL2 is computed over all global degrees of freedom. The MAE averages the absolute error over the $2N$ displacement components. All error metrics are computed after denormalization in physical units.

2.4 Parametric Generalization

Parameterization of PI-GEFN can be done through $\boldsymbol{\kappa}$ as follows:

Geometry Parameters: Simulations vary geometry-specific parameters such as the radius of a hole, resulting in different structural configurations. Training graphs are generated for varying radii.

Material Parameters: Simulations vary Young's modulus E and Poisson's ratio ν, generating different stiffness matrices $\mathbf{K}_{\mathrm{sys}}$.

Material Parameters and Neumann Boundary Conditions: In this joint parameterization setting, Young's

modulus E, Poisson's ratio ν, and the external force vector $\mathbf{f}$ are varied simultaneously. This enables the PI-GEFN to learn the coupled dependence of the displacement response on both material stiffness and external loading.

2.5 Inverse Problem

The inverse problem aims to identify the parameter vector $\boldsymbol{\kappa}$ such that the predicted displacement field $\mathbf{u}(\boldsymbol{\kappa})$ reproduces observed displacement data.

For the formulation of the reduced approach to model calibration, we apply the implicit function theorem, see, e.g., [11]. Here, $\mathbf{O} \in \mathbb{R}^{n_{\text{obs}} \times 2N}$ is a linear observation operator extracting the measured degrees of freedom and $\mathbf{d} \in \mathbb{R}^{n_{\text{obs}}}$ denotes the observed displacement data. In practice, the measurement data $\mathbf{d}$ is interpolated from sensor locations to the numerical discretization points of the FEM mesh used during PI-GEFN training. The interpolated data is denoted by $\mathbf{d}^{\text{int}}$. This interpolation is performed once prior to optimization. For more details on the reduced approach, the reader is referred to [17].

To improve numerical conditioning, the parameter vector $\boldsymbol{\kappa}$ is normalized element-wise to the unit interval $[0,1]$ using min–max scaling. Optimization is carried out in the normalized space, and physical parameters are recovered via denormalization before surrogate evaluation.

To ensure balanced contributions of displacement components with different magnitudes, the same diagonal scaling strategy introduced in the forward problem is applied in observation space. Let

$$\mathbf{D}_{\text{obs}} = \mathbf{O}\mathbf{D}\mathbf{O}^T \in \mathbb{R}^{n_{\text{obs}} \times n_{\text{obs}}},$$

denote the restriction of the global scaling matrix $\mathbf{D}$ to the observed degrees of freedom. The inverse loss is then defined as

$$\mathscr{L}_{\text{inv}}(\boldsymbol{\kappa}) = \left\| \mathbf{D}_{\text{obs}} \left(\mathbf{O}\mathbf{u}_{\text{pred}}(\boldsymbol{\kappa}) - \mathbf{d}^{\text{int}} \right) \right\|_2^2. \tag{6}$$

The reduced identification problem then reads

$$\boldsymbol{\kappa}^* = \arg\min_{\boldsymbol{\kappa}} \mathscr{L}_{\text{inv}}(\boldsymbol{\kappa}). \tag{7}$$

The optimization is performed using the L-BFGS-B algorithm, which operates under box constraints in $[0,1]^{n_{\boldsymbol{\kappa}}}$, ensuring physically admissible parameter values [4]. During each iteration, the pre-trained PI-GEFN model serves as a differentiable surrogate that maps the current parameter estimate to a displacement prediction.

3 Results

This section presents the results of our PI-GEFN model on both forward and inverse problems in linear elasticity. In the forward problem, the model predicts the displacement field $\mathbf{u}$ for a given parameter vector $\boldsymbol{\kappa}$. In the inverse problem, the pre-trained PI-GEFN model is used within an inverse identification procedure to identify unknown material parameters and boundary conditions from displacement measurements. For all comparisons presented in this section, FEM reference results—used as inverse targets where applicable—are generated using the numerical setup described in Section 2.5. Prior to applying the method to real-world experimental data, a sanity check is conducted using synthetic data for each case.
All two-dimensional numerical analyses presented in Sections 3.1, 3.2, and 3.3 are performed under a plane stress assumption, which is appropriate for the considered thin plate geometries.

3.1 Parameterization of Geometric Radius

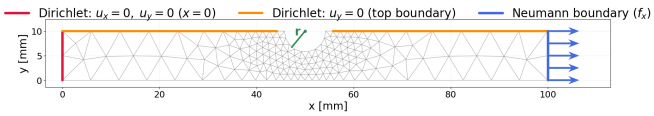

Figure 3 – Symmetric geometry with varying hole radius (all dimensions are in mm)

In this parameterization variant (see Figure 3), we isolate the influence of geometric change, specifically the radius r of the central hole, on the displacement field. The material parameters, namely Young's modulus $E = 185\,\text{GPa}$ and Poisson's ratio $\nu = 0.25$, are kept fixed, while only the geometric radius is varied. This allows PI-GEFN to focus solely on learning the structural response induced by geometric variation. Due to the geometric and loading symmetry of the configuration shown in Figure 3, symmetry conditions are exploited in the numerical setup, allowing the analysis to be restricted to a symmetric portion of the domain without loss of generality.
The selected range for the radius parameter is:

- Radius: $r \in [2.0, 6.0]\,\text{mm}$

To ensure that the model effectively captures the influence of radius variation, we explicitly append the normalized radius value as a node-wise feature for every graph in the training dataset. Adding a radius explicitly as an input dimension significantly improves model performance and interpretability.

Table 1 – PI-GEFN summary for the forward problem for $\boldsymbol{\kappa} = [r]$.

Metric	Value	Unit
Training data points	512	samples
Hidden feature dimensions	[16,16,16,16]	–
EdgeNet size	$[f_e, 64, 128, d_\ell d_{\ell-1}]$	–
Observed rL2 error	6.0×10^{-4}	–
Observed MAE	5.10×10^{-6}	mm
Training time	14	h

As observed in our experiments, the PI-GEFN model trained with **explicit radius input** achieves a significantly lower mean rL2 error of 6.0×10^{-4} on 100 randomly selected unseen test graphs sampled from the interior of the training range, but not included in training. In contrast, when the radius was encoded only through the graph geometry, without being explicitly provided as a node feature, the model exhibited a much higher rL2 error of approximately 4.0×10^{-2} on the same set of graphs. This stark difference highlights the advantage of explicitly incorporating key geometrical parameters into the model input rather than relying on information implicitly encoded in the graph topology. Please refer to Figure 4 for a visual comparison of the rL2 error obtained with implicit and explicit radius encoding. The numerical setup and training configuration are summarized in Table 1. The training dataset consists of 512 graph instances generated by sampling the radius parameter r from the interval $r \in [2.0, 6.0]$ mm using a Sobol' sequence. The EdgeNet sizes reported in Table 1 correspond to the layer-wise dimensions of the MLP used for edge-conditioned message passing. Reported rL2 and MAE values are computed on 100 randomly selected unseen test graphs sampled from the same radius range, but not included in training. Training is performed on an NVIDIA A100 GPU with 80 GB memory.

Furthermore, to evaluate the effectiveness of the explicitly parameterized PI-GEFN model in an inverse setting, as briefly discussed in Section 2.5, we construct a set of synthetic datasets with prescribed hole radii $r \in \{2.0, 2.75, 3.5, 4.0, 5.25, 6.0\}$ mm. These datasets are used to assess the model's ability to recover geometric parameters from displacement observations. Specifically, the inverse task consists of identifying the geometric radius r while keeping the material parameters fixed. As summarized in Table 2, the model trained with explicit radius input demonstrates accurate inverse identification performance, highlighting its robustness even in challenging identification scenarios.

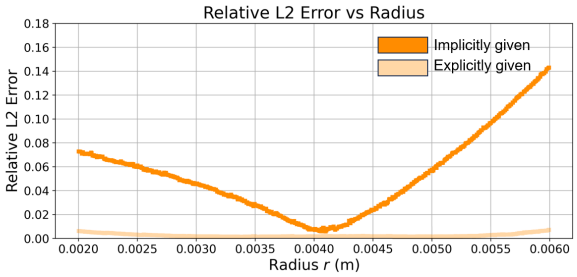

Figure 4 – Comparison of prediction accuracy with and without explicit radius encoding in the graph. Explicit encoding shows significantly better generalization.

Table 2 – Inverse identification of geometric radius from displacement field using PI-GEFN for various test cases.

Metric	Radius r
Ground truth	2.00 mm
PI-GEFN prediction	2.11 mm
Relative error	5.50 %
Ground truth	2.75 mm
PI-GEFN prediction	2.63 mm
Relative error	−4.36 %
Ground truth	3.50 mm
PI-GEFN prediction	3.63 mm
Relative error	3.71 %
Ground truth	4.00 mm
PI-GEFN prediction	4.05 mm
Relative error	1.25 %
Ground truth	5.25 mm
PI-GEFN prediction	5.09 mm
Relative error	−3.05 %
Ground truth	6.00 mm
PI-GEFN prediction	6.30 mm
Relative error	5.00 %

3.2 Parameterization of Material Parameters

To enable the model to generalize across different material behaviors, we generate a set of stiffness matrices corresponding to varying material parameters using a Sobol sequence. Specifically, the Young's modulus is varied within $E \in [150, 258]$ GPa and the Poisson's ratio within $\nu \in [0.125, 0.3636]$, while the geometry and boundary conditions are kept fixed (uniaxial tensile force $f_x = 6498$ N), see Figure 5. For each sampled parameter set, a FEM simulation is performed, and the

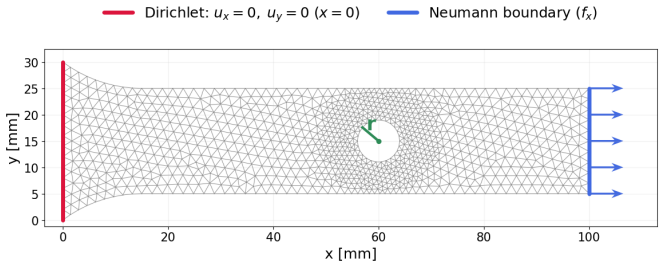

Figure 5 – Geometry under consideration (all dimensions are in mm)

resulting displacement field is converted into a graph representation suitable for learning, as described in Section 2.2.

These ranges and the considered geometry (see Figure 5) are consistent with those used in the reference study by [1], which also involves inverse modeling on the same digital dataset provided by [19]. The employed dataset is based on experimental full-field digital image correlation (DIC) measurements and therefore inherently contains measurement noise and possible specimen misalignment effects. Adopting these parameter bounds and data sources ensures a fair and meaningful comparison with prior work in the literature, while simultaneously highlighting the applicability of the proposed method to noisy real-world experimental data.

Table 3 provides all details used to train the model over the material parameter space, and Table 4a summarizes the performance of the inverse problem. A total of 1024 training graphs are generated using Sobol sampling over the material parameter space. The EdgeNet sizes listed in Table 3 define the MLP structure used to generate edge-conditioned filter matrices. Error metrics are evaluated on 100 randomly selected unseen test cases sampled from the same material-parameter space, but not included in training. Reference errors are taken from [1]. Training is performed on an NVIDIA A100 GPU with 80 GB memory. The FEM reference solutions used for comparison are computed using 2D three-node triangular elements with linear shape functions. For the experimental DIC data [19], we report inverse identification results using both raw and interpolated displacement fields. The raw data correspond to displacement measurements evaluated directly at the sensor locations and therefore retain measurement noise and possible specimen misalignment effects inherent to real experiments. In contrast, the interpolated data are obtained by projecting the noisy DIC measurements onto the finite element mesh used in the inverse parameter identification. Such interpolation is required for FEM-based inverse parameter identification, as FEM solutions are defined at mesh nodes, and does not remove the measurement noise entirely, but typically smooths the field and can reduce its apparent level. It is additionally considered for PI-GEFN to enable a consistent comparison. Reporting results for both cases follows the methodology of [1] and allows assessing the robustness of the proposed approach with respect to measurement noise, while ensuring a fair comparison with FEM-based inverse identification.

Table 3 – PI-GEFN summary for the forward problem for $\boldsymbol{\kappa} = [E, \nu]$.

Metric	Value	Unit
Training data points	1024	samples
Hidden feature dimensions	[16, 16, 16, 16]	–
EdgeNet size	$[f_e, 64, 128, 256, d_\ell d_{\ell-1}]$	–
Observed rL2 error	7.15×10^{-4}	–
Observed MAE	1.41×10^{-5}	mm
Reference rL2 error	6.32×10^{-5}	–
Reference MAE	1.08×10^{-6}	mm
Training time	26	h
Reference training time	32	h

Upon comparing the inverse-modeling results (i.e., the identified material parameters), we observed that the reference FEM (interpolated) values obtained in this work differ slightly from those reported in [1]. This discrepancy arises due to mesh resolution, element type, boundary condition enforcement strategies, numerical integration schemes, and interpolation methods applied to the simulation output.

However, it is important to emphasize that the core objective here is not to replicate the exact FEM reference values used by [1], but rather to evaluate the accuracy and generalization capability of the surrogate model within a consistent pipeline. When compared against the high-fidelity FEM results generated from our own solver, the proposed surrogate model (PI-GEFN) demonstrates strong inverse identification capabilities. This mirrors the performance observed for PINNs in [1], where their model closely matched the FEM results produced within their own framework. Hence, although the reference FEM values differ slightly across works due to solver-level differences, each surrogate model is shown to perform well within its own setup, thereby validating the effectiveness of the data-driven inverse estimation approach.

Table 4 – Comparison of inverse material parameter identification results obtained with PI-GEFN for (a) material-only parameterization, $\boldsymbol{\kappa} = [E, \nu]$, and (b) joint parameterization including Neumann boundary conditions, $\boldsymbol{\kappa} = [E, \nu, f_x]$. PINN reference results from [1] are shown once for consistency. Relative error is defined as $(\kappa_i^{\text{PI-GEFN}} - \kappa_i^{\text{true}})/\kappa_i^{\text{true}}$. In the experimental case, the "true" value denotes the FEM estimate based on interpolated DIC data.

	Young's modulus E	**Poisson's ratio** ν
PINN results [1]		
FEM (interpolated data)	185.182 GPa	0.2590
PINN (raw data)	175.354 GPa	0.2327
Relative error (raw)	−5.31 %	−10.14 %
PINN (interpolated data)	184.648 GPa	0.2571
Relative error (interpolated)	−0.29 %	−0.76 %

(a) Material-only parameterization ($\boldsymbol{\kappa} = [E, \nu]$).

	E	ν
Sanity check (synthetic data)		
Ground truth	200.000 GPa	0.2000
PI-GEFN prediction	199.679 GPa	0.2130
Relative error	−0.16 %	6.50 %
PI-GEFN results using experimental data [19]		
FEM (interpolated data)	173.000 GPa	0.2333
PI-GEFN (raw data)	165.000 GPa	0.2504
Relative error (raw)	−4.62 %	7.33 %
PI-GEFN (interpolated data)	172.560 GPa	0.2384
Relative error (interpolated)	−0.25 %	2.19 %

(b) Joint parameterization with fixed f_x ($\boldsymbol{\kappa} = [E, \nu, f_x]$).

	E	ν
Sanity check (synthetic data)		
Applied force $f_x = 3300$ N *(fixed).*		
Ground truth	200.000 GPa	0.2000
PI-GEFN prediction	201.000 GPa	0.1998
Relative error	0.50 %	−0.10 %
PI-GEFN results using experimental data [19]		
Applied force $f_x = 3249$ N *(fixed).*		
FEM (interpolated data)	173.000 GPa	0.2333
PI-GEFN (raw data)	167.000 GPa	0.2484
Relative error (raw)	−3.47 %	6.47 %
PI-GEFN (interpolated data)	173.560 GPa	0.2338
Relative error (interpolated)	0.32 %	0.21 %

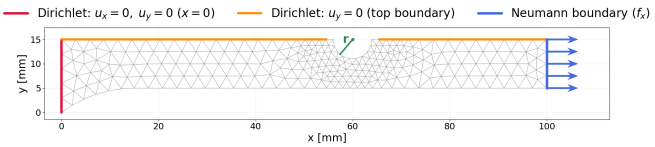

Figure 6 – Symmetric geometry under consideration (all dimensions are in mm)

3.3 Parameterization of Material Parameters with Neumann Boundary Conditions

Building on the material-only parameterization discussed in Section 3.2, we now extend PI-GEFN to incorporate variable external tractions by varying Neumann boundary conditions. In addition to varying material parameters (i.e., varying the stiffness matrix $\mathbf{K}$), external force vectors $\mathbf{f}$ are also systematically changed. By introducing this additional dimension of variability, PI-GEFN is trained to capture how displacements evolve not just from material stiffness, but also from the magnitude of loading; see Figure 6.

The selected ranges for the material parameters are:

- Young's modulus: $E \in [150, 258]$ GPa
- Poisson's ratio: $\nu \in [0.125, 0.3636]$
- Force: The applied force f_x is sampled from the interval $f_x \in [2988, 3654.15]$ N, corresponding to a 10 % variation around $\mu_{f_x} = 3321.08$ N.

Table 5 provides all details used to train the model over the material parameter space, and Table 4b provides a performance summary for the inverse problem. The extended parameterization uses 4096 training graphs generated via Sobol sampling over $\boldsymbol{\kappa} = [E, \nu, f_x]$. Reported forward error metrics are evaluated on unseen test graphs sampled from the same parameter space, but not included in training. EdgeNet sizes in Table 5 reflect the deeper MLP used for edge-conditioned message passing. The same error metrics as in previous sections are employed. Training is performed on an NVIDIA A100 GPU with 80 GB memory. For the inverse problem in this configuration, the applied force f_x is kept fixed while the material parameters are identified from displacement observations. Inverse identification results are compared against FEM reference values obtained within the same numerical setup, as well as PINN results reported in [1].

The extended PI-GEFN model presented here achieves performance comparable to that of the earlier version, which only parameterized material properties.

Table 5 – PI-GEFN summary for the forward problem for $\boldsymbol{\kappa} = [E, \nu, f_x]$.

Metric	Value	Unit
Training data points	4096	samples
Hidden feature dimensions	[16,16,16,16,16]	–
EdgeNet size	$[f_e, 64, 128, 128, 256, d_\ell d_{\ell-1}]$	–
Observed rL2 error	3.8×10^{-4}	–
Observed MAE	1.09×10^{-5}	mm
Training time	48	h

First, the model now incorporates the external force magnitude f_x as part of the parameter vector $\boldsymbol{\kappa} = [E, \nu, f_x]$, allowing it to generalize across varying Neumann boundary conditions in addition to material variability. This enables the model to learn how displacement fields evolve not just due to changes in material stiffness, but also in response to loading magnitude.

Second, unlike the previous setup which used a full geometry, the current approach adopts a symmetric half-body configuration. Leveraging this symmetry reduces the problem domain, simplifies the solution space, and allows the model to learn more efficiently. Consequently, the applied force value f_x is halved compared to the full-body setup, but due to the symmetric formulation, the physical behavior and predictive capacity remain consistent. This modeling choice not only improves computational efficiency but also leads to smoother learning dynamics during training.

Compared to the material-only parameterization presented in Section 3.2, the present configuration extends the parameter space from two to three dimensions, $\boldsymbol{\kappa} = [E, \nu, f_x]$. Despite this increased dimensionality, the forward prediction accuracy (Table 5) remains comparable to the material-only case (Table 3), and shows a slight improvement in the reported error metrics. This behavior is attributed to the larger training dataset and the symmetric problem formulation adopted here.

For the inverse problem, the interpolated results in Table 4b are consistent with the identification accuracy observed in the material-only setting (Table 4a). The surrogate therefore maintains stability and accuracy even when trained over an extended parameter space.

Although the forward model is trained jointly in $[E, \nu, f_x]$, inverse identification in this configuration is performed with the applied force f_x fixed. This is motivated by identifiability considerations. In linear elasticity, displacement magnitudes scale proportionally with the ratio f_x/E, which can introduce correlations between loading magnitude and stiffness if both are treated as unknowns. Fixing f_x during inverse identification isolates the material parameters and results in a well-posed and numerically stable estimation problem. These observations indicate that the proposed PI-GEFN framework can accommodate combined material and loading variability in the forward setting, while preserving reliable inverse identification performance.

4 Conclusion and Outlook

This work presented the Physics-Informed Graph Edge Filter Network (PI-GEFN), a physics-informed graph neural network framework inspired by the finite element method, with capabilities for both forward and inverse problems in linear elasticity. The model integrates finite-element-based structural information as edge features and enforces physical consistency using residual-based loss terms. It is designed to be extensible, interpretable, and generalizable across varying material parameters and boundary conditions, allowing the same framework to accommodate different parameterizations without changing its core architecture.

Key conclusions drawn from this work include:

1. **Explicit parameter input improves convergence and accuracy:** Explicitly incorporating parameters such as geometric radius as node-wise features leads to significantly lower prediction errors and faster convergence compared to relying on implicit encoding through mesh geometry.
2. **Comparable accuracy to PINN-based approaches in inverse elasticity problems:** The proposed PI-GEFN model achieves identification accuracy on par with the PINN-based method presented by [1] for material parameter identification, as demonstrated in Section 3.2. Both approaches exhibit strong agreement with FEM-based ground truth within their respective solver frameworks.
3. **Robust inverse modeling with sparse supervision:** Even when the forward model is trained with labels on only 20% of mesh nodes, PI-GEFN accurately recovers physical parameters in inverse tasks from full-field displacement data.
4. **Flexible extension to boundary condition parameterization:** The model accommodates Neumann boundary conditions (f_x) alongside material parameters (E, ν) in the parameter vector

$\boldsymbol{\kappa}$, allowing for broader applicability across varied loading scenarios.

5. **Extensible, physics-informed surrogate architecture:** The surrogate integrates physical structure via the stiffness matrix $\mathbf{K}$ and force vector $\mathbf{f}$, maintaining interpretability while allowing the same framework to accommodate different parameterizations across parameter spaces.
6. **Agreement with high-fidelity FEM results:** Across both synthetic and real-world tests, PI-GEFN reliably matches the results obtained from FEM simulations, with inverse identification errors often below 1.5%, validating the model's accuracy and reliability.
7. **Built upon established FEM foundations:** The proposed PI-GEFN framework can naturally build on decades of advancements in FEM, such as adaptive refinement, multiscale modeling, and error estimation. This compatibility positions it as a promising bridge between classical simulation-based modeling and emerging neural network-based approaches.

Outlook: Based on current findings, several promising research directions emerge for advancing PI-GEFN in both capability and applicability:

1. **Joint parameterization of geometry and physics:** While the current work separately handles geometric parameters (e.g., the hole radius) and physical parameters (E, ν, f_x), a logical extension is to unify them within a single parameter space $\boldsymbol{\kappa}$. This would enable the model to identify multiple interdependent geometric and physical parameters from observed displacement fields, making it suitable for more complex inverse design tasks in engineering and biomechanics.
2. **Learnable surrogate for stiffness and force assembly:** A known bottleneck in the inverse problem is the need to reassemble the FEM matrices $\mathbf{K}(\boldsymbol{\kappa})$ and $\mathbf{f}(\boldsymbol{\kappa})$ during every forward pass. Although still faster than solving the full PDE system, this introduces a computational overhead. Future work could explore training a secondary graph neural network (or MLP) to directly predict $\mathbf{K}$ and $\mathbf{f}$ from the parameter vector $\boldsymbol{\kappa}$, effectively bypassing FEM assembly and enabling real-time inverse identification.
3. **Robustness to sparse sensor data:** While PI-GEFN shows strong performance on synthetically generated graphs with rich node coverage, performance slightly drops when applied to real-world settings with limited sensor locations. Enhancing robustness under sparse observations can be achieved by training the model on graph samples with varying sparsity levels, node distributions, and noise conditions. This would align the surrogate more closely with experimental deployment.
4. **Noise-aware inverse modeling:** Real experimental data is inherently noisy due to sensor inaccuracies or environmental disturbances. To account for this, future versions of PI-GEFN could incorporate probabilistic modeling techniques such as Bayesian neural networks or Monte Carlo dropout for uncertainty quantification. For example, [8] demonstrated that dropout at test time can serve as an efficient approximate Bayesian inference method, enabling neural networks to output predictive uncertainty estimates with minimal modification to the model architecture. More advanced Bayesian formulations for noisy inverse problems have been explored in the context of physics-informed learning, notably the Bayesian physics-informed neural network, which integrates physical laws and noisy observations to estimate posterior distributions over parameters using Hamiltonian Monte Carlo or variational inference for uncertainty quantification [21]. Incorporating such probabilistic methods into PI-GEFN would allow it to not only identify mechanical parameters but also determine confidence bounds, significantly enhancing its reliability in safety-critical engineering applications.
5. **Extension to hyperelasticity:** The proposed PI-GEFN can be extended to finite-strain hyperelasticity while retaining the central concept of stiffness-informed edge features. The architecture will adopt a two-pass predictor–corrector strategy, where in the first pass (predictor) the model takes as input node features (coordinates, boundary flags, material parameters) and base edge features (geometry, length, orientation, area/volume, aspect ratio) to predict an initial displacement field $\mathbf{u}^{(0)}$ using EdgeNet-U. A differentiable physics layer will then compute element strains and hyperelastic stresses (e.g., St. Venant–Kirchhoff or Neo-Hookean models), assemble the internal force vector $\mathbf{f}_{\text{int}}(\mathbf{u}^{(0)})$, and extract stiffness-like edge proxies (e.g., projected axial stiffness or $\mathbf{B}^{\top}\mathbf{E}\mathbf{B}$ summaries). Here, $\mathbf{B}$ denotes the strain–displacement matrix and $\mathbf{E}$ the constitutive matrix, which depends on the material parameters (e.g., Young's modulus E

and Poisson's ratio ν for isotropic linear elasticity) following standard FEM formulations as described in [7]. In the second pass (corrector), these proxies will be concatenated to the base edge features and reprocessed by EdgeNet-U (with shared weights) to obtain a refined displacement field $\mathbf{u}^{(1)}$. The physics layer will be applied again to recompute internal forces and form the residual $\mathbf{R} = \mathbf{f}_{\text{ext}} - \mathbf{f}_{\text{int}}(\mathbf{u}^{(1)})$. The model is trained by minimizing $\|\mathbf{R}\|_2^2$ with additional penalties for boundary condition satisfaction and, if necessary, energy regularization to stabilize hyperelastic behavior. The training protocol will progress from linear elastic and small-strain hyperelastic phases to full finite-strain cases with pseudo-load increments $\eta \in [0,1]$, reusing the two-pass block at each increment with initialization from the previous step. At inference, small-load cases will use a single two-pass application, whereas large-load cases will advance in pseudo-load steps, carrying forward displacements and updated stiffness proxies, thus preserving the physics-informed edge feature philosophy while ensuring stability and accuracy in nonlinear regimes.

6. **Modeling non-homogeneous and spatially varying materials.** An important extension of PI-GEFN is the treatment of structures with spatially varying material properties, such as composites or functionally graded materials. In this setting, different elements (or regions) possess distinct material parameters, e.g., Young's modulus and Poisson's ratio. From a weak-form perspective, the governing formulation remains unchanged, as the local material parameters are directly incorporated into the element-specific constitutive matrix $\mathbf{E}_e$ during the computation of stiffness-like quantities. Consequently, the same PI-GEFN architecture can be employed without structural modification. The only adaptation concerns the construction of physics-informed edge features: when assembling stiffness proxies (e.g., projected axial stiffness or $\mathbf{B}^\top \mathbf{E}_e \mathbf{B}$ summaries), the element-dependent $\mathbf{E}_e$ is used. This preserves physical consistency in the message-passing process while enabling accurate modeling of heterogeneous materials and spatially resolved inverse identification of material parameters.

Code Availability: The complete implementation of the PI-GEFN pipeline used in this study is publicly available. The repository includes the data generation scripts, model architecture definitions, training and evaluation routines for both forward and inverse problems, as well as post-processing utilities required to visualize the results presented in this work.

The source code is hosted on GitHub and is available in the PI-GEFN repository [15]. The experimental DIC dataset used in this work is available in the Zenodo repository [19].

All experiments reported in this paper were conducted using this codebase. The repository further contains documentation describing the software dependencies, execution workflow, and example configurations for reproducing the numerical experiments.

CRediT authorship statement: A. Potnis: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **D. Anton**: Writing - review & editing, Supervision. **H. Wessels**: Writing - review & editing, Supervision, Funding acquisition.

References

[1] David Anton, Jendrik-Alexander Tröger, Henning Wessels, Ulrich Römer, Alexander Henkes, and Stefan Hartmann. Deterministic and statistical calibration of constitutive models from full-field data with parametric physics-informed neural networks. *Advanced Modeling and Simulation in Engineering Sciences*, 12:12, 2025.

[2] Uzair Bhatti, Hao Tang, Guilu Wu, Shah Marjan, and Aamir Hussain. Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *International Journal of Intelligent Systems*, 2023, 02 2023.

[3] Nolan Black and Ahmad Najafi. Learning finite element convergence with the multi-fidelity graph neural network. *Computer Methods in Applied Mechanics and Engineering*, 397:115120, 07 2022.

[4] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[5] Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

[6] Saurabh Deshpande, Stéphane PA Bordas, and Jakub Lengiewicz. MAgNET: A graph U-Net architecture for mesh-based simulations. *Engineering Applications of Artificial Intelligence*, 133:108055, 2024.

[7] Dieter Dinkler and Ursula Kowalsky. *Introduction to finite element methods*. Springer Vieweg Wiesbaden, 1 edition, September 2023. ISBN 978-3-658-42742-9.

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

[9] Han Gao, Matthew J. Zahra, and Jian-Xun Wang. Physics-informed graph neural galerkin networks: A unified framework for solving PDE-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390: 114502, 2022.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[11] S. G. Krantz and H. R. Parks. *The implicit function theorem: History, theory, and applications*. Modern Birkhauser Classics. Birkhauser, New York, 1 edition, 2013. ISBN 978-1-4614-5981-1.

[12] Ravi Ghanshyam Patel, Cosmin Safta, and Reese E. Jones. Equivariant graph convolutional neural networks for the representation of homogenized anisotropic microstructural mechanical response. *Computer Methods in Applied Mechanics and Engineering*, 432(A), 11 2024. ISSN ISSN 0045-7825.

[13] Roberto Perera and Vinamra Agrawal. Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations. *Computer Methods in Applied Mechanics and Engineering*, 429:117152, 2024.

[14] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2021.

[15] Atharva Potnis. PI-GEFN-pipeline. `https://github.com/Atharvapotnis/PI-GEFN-pipeline`, 2026. GitHub repository, accessed 2026-03-25.

[16] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[17] Ulrich Römer, Stefan Hartmann, Jendrik-Alexander Tröger, David Anton, Henning Wessels, Moritz Flaschel, and Laura De Lorenzis. Reduced and all-at-once approaches for model calibration and discovery in computational solid mechanics. *Applied Mechanics Reviews*, 77(4):040801, 05 2025. ISSN 0003-6900.

[18] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29–38, 2017.

[19] J.-A. Tröger, S. Hartmann, D. Anton, and H. Wessels. Digital image correlation measurement of linear elastic steel specimen, 2024. Dataset available on Zenodo.

[20] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2017.

[21] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021. ISSN 0021-9991.

[22] Yingxue Zhao, Haoran Li, Haosu Zhou, Hamid Reza Attar, Tobias Pfaff, and Nan Li. A review of graph neural network applications in mechanics-related domains. *Artificial Intelligence Review*, 57:315, 2024.

[23] Zhaoxuan Zheng, Hae Young Noh, Daeho Jang, Jinkyu Yang, Craig E Maloney, and Eunho Kim. Unifying the design space and optimizing linear and nonlinear truss metamaterials by generative modeling. *Nature Communications*, 14(1):6841, 2023.

[24] Olek C. Zienkiewicz and Robert L. Taylor. *The finite element method for solid and structural mechanics*. Elsevier Butterworth-Heinemann, 6th edition, 2005.