PHYSICS-INFORMED NEURAL NETWORKS FOR SOLVING
THERMO-MECHANICS PROBLEMS OF FUNCTIONALLY GRADED
MATERIAL

A PREPRINT

Mayank Raj
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai-600036
me16b147@smail.iitm.ac.in

Pramod Kumbhar
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai-600036
pramodkumbhar789@gmail.com

Ratna Kumar Annabattula*
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai-600036
ratna@iitm.ac.in

January 7, 2022

ABSTRACT

Differential equations are indispensable to engineering and hence to innovation. In recent years, physics-informed neural networks (PINN) have emerged as a novel method for solving differential equations. PINN method has the advantage of being meshless, scalable, and can potentially be intelligent in terms of transferring the knowledge learned from solving one differential equation to the other. The exploration in this field has majorly been limited to solving linear-elasticity problems, crack propagation problems. This study uses PINNs to solve coupled thermo-mechanics problems.

*Corresponding author
of materials with functionally graded properties. An in-depth analysis of the PINN framework has been carried out by understanding the training datasets, model architecture, and loss functions. The efficacy of the PINN models in solving thermo-mechanics differential equations has been measured by comparing the obtained solutions either with analytical solutions or finite element method-based solutions. While $R^2$ score of more than 99% has been achieved in predicting primary variables such as displacement and temperature fields, achieving the same for secondary variables such as stress turns out to be more challenging. This study is the first to implement the PINN framework for solving coupled thermo-mechanics problems on composite materials. This study is expected to enhance the understanding of the novel PINN framework and will be seminal for further research on PINNs.

**Keywords**  Physics-informed neural networks · Deep learning · Differential equations · Functionally graded material

1 Introduction

The term “neural network” encompasses a wide range of machine learning models with one similarity, i.e., so-called “neurons” as the basic building block [1, 2]. Neural networks have seen many applications in fields not limited to computer vision [3], banking and finance [4], bio-informatics [5, 6], renewable energy systems [7, 8] and material science [9, 10, 11]. In other words, if there is a function approximation task, neural networks are indisputable models to use. The use cases are not limited to just function approximation tasks. Rather, neural networks dominate in image captioning and language translation [12], which are one-to-many or many-to-many mapping tasks. So what is behind the success of neural networks? Certainly, the flexibility they offer in designing composite parametric functions and the ease of optimizing those parameters [13] lends to their success. Often called universal function approximators [14], these neural networks can, in theory, fit most of the relational mappings. However, the challenge lies in designing a neural network architecture so that at least one set of optimum weights fits the desired function. The so-called training of a neural network, i.e., estimation of optimal parameters using an optimization algorithm, can also be challenging. However, after going through those steps, one gets a trained neural network model that can predict/estimate the desired target variable on a given set of inputs in seconds. Alongside all the success stories, neural networks also have their own set of drawbacks. Training often requires considerable data, careful data sampling to cover variance in the domain and target space, difficult parallelization of the training process, and dedicated hardware such as GPUs [15] for fast training. Differential equations are indispensable to computational engineering. Irrespective of engineering subdomains, differential equations appear in one form or another. Often one needs to solve such equations on the domain of interest
numerically. The state of the art of technique for numerically solving such differential equations is the finite element method. While the finite element method has its pros and cons, the physics-informed neural network (PINN) is a promising alternative to solving differential equations.

The family of a neural network is vast, and variety can be in various forms, such as their architecture (arrangement of neurons and layers), learning process (supervised or unsupervised), the purpose of the model (regression or classification). Physics-informed neural network (PINN) is yet another variant in this family. Presently in its infancy, PINN can potentially disrupt how differential equations are solved. PINN has an apparent advantage of being a meshless method over FEM. Unlike conventional supervised learning Deep Neural Networks (DNNs), PINN models are trained without the target samples. While PINN in its current form can take more time than commercial finite element solvers such as Abaqus® to solve simple elasticity problems, it can do better when simulations get complex and the number of simulations increases, such as in an optimization scenario. PINN can use the neural network transfer learning concept [16] to train on new samples in marginal time.

In an abstract sense, a physics-informed neural network tries to optimize the parameters of a parametric function that approximates the solution of the differential equation on a given domain. The network does the optimization by minimizing an objective called the loss function in the machine learning community. The prefix “Physics-informed" comes from the fact that the loss function is obtained using the underlying physics of the problem represented by the differential equations. Often, the loss is the functional corresponding to the differential equation.

The physics-informed neural network was first introduced by Raissi et al. [17] as an alternate method to solve partial differential equations. PINNs can be classified into two categories based on the implementation of the loss function. One of the approaches uses the collocation technique, where simply the residual of the differential equation is minimized by training the neural network [18, 19]. While on the other hand, there are energy-based loss functions where the functional of the differential equation is minimized [20]. While the collocation approach is straightforward, it involves computing higher-order derivatives, resulting in difficult training. On the other hand, the energy-based method [20] requires lower order derivatives due to the functional’s weak form of the differential equation. Hence, energy-based methods have become more popular with numerous applications in the last few years. It is used in the prediction of crack propagation using the phase-field method [21, 22]. Further, the ease with which adaptive meshing can be done in PINN was shown by Goswami et al. [23]. Li et al. [24] compared the effectiveness of the PINN approach with other ML-based techniques in terms of various metrics such as coefficient of determination. Recently, Dwivedi et al. [25, 26] introduced Physics Informed Extreme Learning Machine, which improves the efficiency of Physics Informed Neural
Network by utilizing ideas of Extreme Learning Machine. Dwivedi et al. also introduced Distributed Physics Informed Neural Network [27, 28, 29] where the domain can be split into multiple sub-domains, and separate PINNs can be trained over them with ensuring continuity of variables on the interface of sub-domains.

Functionally graded materials (FGMs) are a unique class of composites incorporating the material properties of two or more different homogeneous materials [30]. The material property variation is usually continuous and smooth such that it can be represented as explicit smooth mathematical functions. Usually, these materials trump the conventional homogeneous components in terms of superior thermal and mechanical capabilities. Hence, it is not surprising that the FGMs find ubiquitous applications ranging from mechanical to bio-medical applications [31].

While there are many papers published in PINNs in the last year, none of them presents a comprehensive discussion on the pros and cons of the methodology. Moreover, the application has been limited to isotropic material, and discussions have mostly been centered around the primary variables such as displacement fields and the damage parameter in phase-field modeling of fracture. There has not been any discussion of secondary variables such as strain and stress fields that depend on primary variables’ derivatives. Moreover, there has been a knowledge gap in understanding the convergence of energy-based PINN neural networks.

This paper starts with a discussion on using PINN models to solve one-dimensional differential equations. We specifically solve coupled thermo-mechanics problems in solid mechanics for the first time using PINN. We consider materials with spatially varying properties and show the robustness of PINN to arrive at the desired solution. We further discuss how and how not to implement different kinds of boundary conditions. Further on, we point out that the PINN model converges to a physically meaningful loss function value. We then demonstrate the efficacy of PINN in solving the two-dimensional thermo-mechanics problem with functionally graded properties. We achieve remarkable PINN prediction accuracy of displacement and temperature fields. However, we show that excellent accuracy in primary variables such as displacement fields does not imply the same for secondary variables such as stress fields. We conclude the paper by discussing further avenues of research on PINN models.

2 Methods

2.1 Governing differential equations for thermo-elasticity

Let us consider a solid body occupying spatial domain $\Omega \subset \mathcal{R}^d$. The solid body is composed of material with thermo-elastic properties (see Figure 1). The $d$ dimensional domain is bounded by a surface $\Gamma$. The bounding surface is
Physics-informed neural network

Figure 1: Schematic of a solid object with spatially varying properties lying in a two-dimensional space.

defined locally in terms of the outward unit normal vector \( n \). Let the elastic modulus and thermal conductivity of body vary as \( E(x) \) and \( k(x) \) where \( x \) is a point in the domain \( \Omega \). The static thermo-elastic response of the body, displacement field \( u : \Omega \to R^d \) and temperature field \( T : \Omega \to R \) is governed by the following differential equations:

\[
\nabla \cdot \sigma(x) + f(x) = 0 \quad \forall x \in \Omega \tag{1}
\]

\[
\nabla \cdot (k(x) \nabla T(x)) = 0 \quad \forall x \in \Omega \tag{2}
\]

under the boundary conditions:

\[
\begin{align*}
\sigma(x) \cdot n &= t_{bc}(x) \quad \forall x \in \Gamma_t \\
T(x) &= T_{bc}(x) \quad \forall x \in \Gamma_T
\end{align*}
\]

\[
\begin{align*}
u(x) &= u_{bc}(x) \quad \forall x \in \Gamma_u
\end{align*}
\]
\( \sigma(x) \) and \( f(x) \) represents stress field and body force respectively. Moreover, the prescribed boundary conditions \( u_{bc}, t_{bc} \) and \( T_{bc} \) are defined such that \( (\Gamma_u \cup \Gamma_t \cup \Gamma_T) \subset \Gamma \) and \( \Gamma_u \cap \Gamma_t = \Phi \). The stress field is related to strain field which in turn is related to displacement and temperature fields as follows:

\[ \sigma(x) = C(x) : \varepsilon(x) \]

\[ \varepsilon(x) = \varepsilon^{el}(x) - \alpha \Delta T(x)I \]

\[ \varepsilon^{el}_{ij} = \frac{u_{i,j} + u_{j,i}}{2} \]

where \( \varepsilon^{el} \) is the elastic strain while \( \alpha \Delta T \) is the thermal strain. \( \alpha \) is the coefficient of thermal expansion, \( I \) is the identity matrix and \( \Delta T(x) = T(x) - T_0 \) where \( T_0 \) is the temperature of the body in the reference state. The elastic material constitutive matrix \( C(x) \) is written in terms of elastic modulus and Poisson’s ratio \( \nu \) as:

\[ C(x) = E(x) \begin{bmatrix} \frac{1}{1 - \nu^2} & \frac{\nu}{1 - \nu^2} & 0 \\ \frac{\nu}{1 - \nu^2} & \frac{1}{1 - \nu^2} & 0 \\ 0 & 0 & \frac{1}{(1 + \nu)} \end{bmatrix} \]

for two-dimensional object with plane-stress assumption.

The functional corresponding to differential equations governing the thermo-elastic response of solid body is given as:

\[ F = W_{int} - W_{ext} \]

\[ = \int \frac{1}{2} \varepsilon^{el}(x) : C(x) : \varepsilon^{el}(x) d\Omega + \int \frac{1}{2} k(x) \nabla T(x) \cdot \nabla T(x) d\Omega - \left( \int f(x) \cdot n d\Omega + \int t(x) \cdot n d\Gamma \right) \]

The functional \( F \) can be viewed as internal energy \( W_{int} \) less work done by the external forces \( W_{ext} \).

In this study, there are a few examples where we have used PINN to predict only the elastic response of the object under consideration. In that scenario, only Equation (1) governs the response of object. Moreover, the coefficient of thermal expansion \( \alpha \) is set to be zero, and the internal energy corresponding to temperature gradient is removed from Equation (2). After setting up the differential equations and boundary conditions governing the problem at hand, PINN is used to minimize the functional, thereby obtaining the desired response fields \( u(x) \) and \( T(x) \). The details of the PINN framework are discussed in the following subsection.
2.2 Physics informed neural network

Neural networks are composite parametric functions designed to learn any relational mapping. While the traditional neural networks used to have only one hidden layer, the abundance of computational resources has made it possible to design and train models with numerous hidden layers. Hence the term deep neural network (DNN) has become synonymous with neural networks. The process of training a DNN is called deep learning. A DNN is built up using the following building blocks: (1) neurons, (2) input layer, (3) output layer, (4) hidden layers, (5) activation function/s, and (6) weights and biases. Mathematically, a fully-connected deep neural network with straight-forward architecture can be described as:

$$
\begin{align*}
  h^1 &= \psi(W^1 x + b^1) \\
  h^2 &= \psi(W^1 h^1 + b^1) \\
  &\vdots \\
  h^i &= \psi(W^i h^{i-1} + b^i) \\
  &\vdots \\
  \tilde{v} &= W^o h^{o-1} + b^o
\end{align*}
$$

where $x$ and $\tilde{v}$ are respectively the input and output vectors. $W^i$ are the weight matrices, $b^i$ are the bias parameters, $h^i$ are hidden layer vectors and $\psi(.)$ is the activation function used in different layers. Moreover, a further transformation, $\zeta : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is applied on $\tilde{v}$ in order to satisfy Dirichlet boundary conditions:

$$\tilde{z}(x) = \zeta(x, \tilde{v}(x))$$

More on the transformation $\zeta(.)$ is discussed in Section 3. Hence, the sequence of operations in Equations (7) and (8) can be represented as a DNN $\tilde{z}(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n$, which maps an $m$-dimensional real vector $x$ to $n$-dimensional real vector $\tilde{z}$. In the Results and Discussion Section 3 $\tilde{u}$ and $\tilde{T}$ will be often used in place of $\tilde{z}$. The vector $\tilde{z}$ will In order to optimize parameters of a conventional DNN, a loss function is defined as $\mathcal{L}(\tilde{z}, z) : \mathbb{R}^{m \times m} \rightarrow \mathbb{R}$, where $z$ is true value of the target variable and $\mathcal{L}(.)$ can represent a loss function such as mean squared error. However, in case of PINN, the loss function is defined only in terms of $\tilde{u}$ and its derivatives with respect to $x$, for example: $\mathcal{L}(\tilde{z}, \tilde{z}_x) : \mathbb{R}^{m \times m \times n} \rightarrow \mathbb{R}$. 

7
Physics-informed neural network

In order to solve the ODEs given in Equations (1) and (2) under the boundary conditions in Equation (3), the functional given is Equation (6) must be discretized to obtain the loss function $\mathcal{L}$ for training the PINN. Hence, the discretized functional can be written as:

$$
\mathcal{L} = \sum_{x_i \in \Omega} \frac{1}{2} w(x_i) \varepsilon(t)(x_i) : C(x_i) : \varepsilon(t)(x_i) \\
+ \sum_{x_i \in \Omega} \frac{1}{2} w(x_i) k(x_i) \nabla \tilde{T}(x_i) \cdot \nabla \tilde{T}(x_i) \\
- \left( \sum_{x_i \in \Omega} w(x_i) f(x_i) \cdot n + \sum_{x_i \in d\Gamma} w'(x_i) t(x_i) \cdot n \right)
$$

(9)

where $w(x)$ and $w'(x)$ represent the weights for numerical integration on domain and boundary nodes respectively. The PINN predicted solutions $\hat{u}(x)$ and $\hat{T}(x)$ is obtained by minimizing $\mathcal{L}$ such that:

$$
\begin{bmatrix}
\hat{u}(x) \\
\hat{T}(x)
\end{bmatrix} = \hat{z} = \arg\inf_{\hat{z}} \mathcal{L}(\hat{z}, \hat{z})
$$

(10)

2.3 Training physics informed neural network

The parameters of DNN: weights and biases are optimized using the backpropagation [13] algorithm. The algorithm computes the derivative of the loss function with respect to all parameters and subsequently updates them using the gradient descent rule. The weight update process is repeated for a number of epochs until the convergence criteria are satisfied. The parameters of the PINN model are updated using the Adam optimizer [32], which is an enhanced version of gradient descent. Computing derivatives in the backpropagation algorithm is at the core of the training process. The technique called automatic differentiation [33] is used for computing derivatives and hence is at the core of any deep learning library such as Tensorflow [34]. To understand automatic differentiation, let’s take a step back to reflect on how neural networks (or any such parametric function) are built in Tensorflow. To build a neural network, one uses the so-called building block parametric operators in Tensorflow. Examples of such operators are nothing but matrix multiplication, addition, and an activation function. It is to be noted that the gradients to all such operators are already defined in Tensorflow. For example, let’s look at the matrix multiplication operation. Let us say there is a matrix $W$, which when multiplied with vector $x$ of compatible size, returns the matrix multiplication result from $y = Wx$. Hence, whenever one needs to compute the gradient of this operation i.e, $dy_i/dx_j$, the result is the matrix $W$. In a nutshell, the parametric PINN model is built of such fundamental operations whose derivative is well defined in terms of parameters.
themsevles. Hence, one can compute the derivative of the whole PINN function by applying chain rule and using the
derivatives of the basic operations.

2.4 Measure of PINN accuracy - coefficient of determination

The accuracy of PINN prediction is measured in terms of coefficient of determination, popularly known as $R^2$ score. It
is a measure of the amount of variance in the true target variable captured by the predicted values. Mathematically, it
can be defined as:

\[ R^2 \text{ score} = 1 - \frac{\sum_{i \in \Omega} (u_i - \hat{u}_i)^2}{\sum_{i \in \Omega} (u_i - \bar{u})^2} \] (11)

where $\Omega$ represents the domain. $u_i$, $\bar{u}$ and $\hat{u}_i$ are scalar values and represent true displacement values, mean of true
displacement values, and predicted displacement values. The coefficient of determination can similarly be obtained
for any scalar variable of interest such as a particular strain component. While $R^2$ score is in general captures the
performance of PINN, it may not be a suitable measure in some cases. More discussion on $R^2$ score is provided in
Section 3.

3 Results and Discussion

3.1 1D-FGM-ELAS-DIRCH

Consider a bar with uniform cross-section and unit length lying along the $x$-axis from $x = 0$ to $x = 1$. The bar is
assumed to be functionally graded, i.e. the elastic modulus varies spatially as $E(x) = 1/(1 + x)$. Moreover, the bar
is assumed to be fixed at one end, and a load is applied on the other to observe the elastic response. Hence, for a
displacement controlled analysis, the boundary conditions are given as $u(0) = 0$, $u(1) = 1$, where $u(x)$ represents the
displacement field of the bar as a function of $x$ under the given load. The differential equation governing the elastic
response of object for given properties and boundary conditions is given in Table 1. The solution $u(x)$ to the differential
equation can be analytically derived to be:

\[ u(x) = \frac{x^2 + 2x}{3} \] (12)

It is to be noted that throughout this study, consistent units of various physical quantities can be assumed. Hence,
we do not explicitly mention units in various figures and tables. Now, to solve the above differential equation using
a PINN, one needs to understand the three fundamental elements of the PINN framework: (1) training dataset, (2)
Physics-informed neural network

A P

REPRINT

Figure 2: Schematic of one-dimensional bar with spatially varying stiffness and displacement node.

Table 1: Differential equations and loss functions used in training PINNs for solving different problems.

| Problem Code           | Differential Equation                                                                 | Loss function                                                                 |
|------------------------|----------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| 1D-FGM-ELAS-DIRCH      | \( \frac{d}{dx} \left( \frac{1}{1+x} \frac{du}{dx} \right) = 0 \)                   | \( \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{2} \frac{du}{dx} \right)^2 \)       |
| 1D-FGM-ELAS-NEU        | \( \frac{d}{dx} \left( \frac{1}{1+x} \frac{du}{dx} \right) = 0 \)                   | \( \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{2} \frac{du}{dx} \right)^2 - t(1)u(1) \) |
| 1D-ELAS-BF             | \( \frac{d^2 u}{dx^2} + 1 = 0 \)                                                   | \( \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{2} \frac{du}{dx} \right)^2 - u_i - t(1)u(1) \) |
| 1D-FGM-ELAS-THERMO     | \( \frac{d}{dx} \left( \frac{1}{1+x} \frac{du}{dx} \right) \left( \alpha(T - T_0) \right) = 0 \) | \( \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{2} \frac{du}{dx} \right)^2 - T(x_i) + \frac{10}{(1+x_1)} \left( \frac{dT}{dx} \right)^2 \) |
| Kirsch’s Problem       | \( \nabla \cdot \sigma(x) = 0 \)                                                     | \( \sum_{k \in \Omega} w(x_k) \epsilon(x_k) : C : \epsilon(x_k) - \sum_{k \in \Gamma} w(x_k) t(x_k) \cdot n(x_k) \) |
| 2D-FGM-ELAS-NEU        | \( \nabla \cdot \sigma(x) = 0 \)                                                     | \( \sum_{k \in \Omega} \frac{1}{2} w(x_k) \epsilon(x_k) : C : \epsilon(x_k) - \sum_{k \in \Gamma} w(x_k) t(x_k) \cdot n(x_k) \) |
| 2D-FGM-ELAS-DIRCH      | \( \nabla \cdot \sigma(x) = 0 \)                                                     | \( \sum_{k \in \Omega} \frac{1}{2} w(x_k) \epsilon(x_k) : C : \epsilon(x_k) - \sum_{k \in \Gamma} w(x_k) t(x_k) \cdot n(x_k) \) |
| 2D-FGM-THERMO-ELAS     | \( \nabla \cdot (k(x) \nabla T) = 0 \)                                              | \( \sum_{k \in \Omega} \left( \frac{1}{2} w(x_k) \epsilon^{el}(x_k) : C : \epsilon^{el}(x_k) + \frac{10}{(1+x_1)} T(x_k) \cdot \nabla T(x_k) \right) \) |

choice of neural network architecture, (3) loss function. The training dataset or the input to the PINN will be the spatial coordinate of nodes in the domain. Hence, the domain has to be discretized into a set of nodes; however, unlike FEM, there is no notion of elements in the PINN framework. While there is the freedom to non-uniformly discretize the domain into nodes, for simplicity, the nodes are chosen to be uniformly distributed in this example. It is to be noted that there is no target dataset in training PINN models.

Figure 3: Representative PINN architecture for purely displacement controlled analysis of 1D material.
The next step is to decide the neural network architecture. While the PINN framework offers a lot of flexibility in determining the model architecture, a few of such parameters get determined by the task at hand. For example, the number of nodes in the input and output layers of the model must respectively be equal to the dimensionality of the problem. In the case of the one-dimensional solid mechanics problem, both the input and output layer has one neuron each. The input neuron corresponds to the spatial position of nodes, and the output neuron corresponds to the displacement. Moreover, in the case of purely Dirichlet boundary conditions, the PINN model architecture can be modified to satisfy displacement boundary conditions implicitly (see Equation (8)).

Let us assume that there is a blackbox neural network function \( \tilde{v}(x) : \Omega \rightarrow \mathbb{R} \), as defined in Equation (7), which maps points in the spatial domain \( \Omega \) to a real number. Now, the Dirichlet boundary conditions as assumed in this example can be implemented by the transformation \( \tilde{u}(x) := x + x(1 - x)\tilde{v}(x) \) thereby ensuring \( \tilde{u}(0) = 0 \) and \( \tilde{u}(1) = 1 \). In Figure 3, connections in black colour correspond to \( \tilde{u}(x) \), while the connection in blue is the transformation applied over \( \tilde{v}(x) \) to get \( \tilde{u}(x) \). The details of neural network \( \tilde{v}(x) \) is given in Table 2. It is to be noted that one cannot do any transformation to implement Dirichlet BC such as \( \tilde{u}(x) := x^2 + x^2(1 - x^2)\tilde{v}(x) \). This transformation will not only ensure \( \tilde{u}(0) = 0 \) and \( \tilde{u}(1) = 1 \), but it will also simultaneously put constraints on the displacement gradient which is \( \tilde{u}'(0) = 0 \) for the given case. Hence, a transformation which puts undesired constraints on gradients of primary variables must be avoided.

| Problem Code        | No. of input nodes | No. of output nodes | No. of hidden layers | No. of nodes in hidden layers | Activation function |
|---------------------|--------------------|---------------------|----------------------|-------------------------------|--------------------|
| 1D-FGM-ELAS-DIRCH   | 1                  | 1                   | 2                    | [5, 5]                        | tanh(.)            |
| 1D-FGM-ELAS-NEU     | 1                  | 1                   | 2                    | [5, 5]                        | tanh(.)            |
| 1D-ELAS-BF          | 1                  | 1                   | 2                    | [5, 5]                        | tanh(.)            |
| 1D-FGM-THERMO-ELAS  | 1                  | (1,1)               | (2,1)                | [(5, 5), (10, 10), (5,5)]     | tanh(.)            |
| Kirch’s Problem     | 2                  | 2                   | 3                    | [100, 100, 100]               | tanh²(.)           |
| 2D-FGM-ELAS-NEU     | 2                  | 2                   | 3                    | [100, 100, 100]               | tanh²(.)           |
| 2D-FGM-ELAS-DIRCH   | 2                  | 2                   | 3                    | [100, 100, 100]               | tanh²(.)           |
| 2D-FGM-THERMO-ELAS  | 2                  | 3                   | 2                    | [300, 300]                    | elu²(.)            |

The last and the most crucial element of the PINN framework is the loss function used in training such models. As highlighted earlier, the loss used sets the PINN model apart from the rest of the neural network family. The PINN loss function is discretized form of the functional corresponding to the underlying differential equation. Hence, the loss function will also change from one differential equation to the other. It is to be noted that discretization is done to numerically evaluate the functional in Equation (6) for training the PINN. The neural network model trains itself (fits the differential equation) by minimizing the loss function and thereby finding a function that minimizes the functional corresponding to the differential equation. However, the tricky part is that the functional and hence the loss
function often depends not only on the primary target variables but also on their derivatives. Differential equation and corresponding loss function can be found in Table 1.

In the context of solid mechanics, the loss function is the internal strain energy minus work done by external forces on the object under consideration. It is evident from Table 1, the loss involves computing strain values which depends on gradient of \( u(x) \) with respect to \( x \). Since the neural network is a parametric composite function, it is possible to compute the derivative of the target variable \( \tilde{u}(x) \) with respect to the input variable \( x \) in terms of parameters of the neural network. While computing such derivatives can be a tedious task given the complexity of the neural network, the state-of-the-art deep learning libraries turn out to be the savior. The derivative can be easily computed using the automatic differentiation technique (discussed in Section 2.3) implemented using the library Tensorflow. The complete pipeline of the PINN framework is summarized in Figure 4.

![Flowchart of PINN framework](image)

After training the neural network with physics-informed loss, the given one-dimensional differential equation is solved to obtain \( \hat{u}(x) \) which is the same as \( \tilde{u}(x) \) at the end of training PINN. The comparison between \( \hat{u}(x) \), and analytical solution is shown in Figure 5. It is worthwhile to reiterate that the physics-informed loss must converge to the minimum strain energy. The training of neural networks is an optimization task where the global minima should ideally correspond to the unique minimum energy solution of the system. \( R^2 \) for different variables in given in Table 3.
Figure 5: Decline in physics-informed loss (left) and comparison between analytical and predicted solution obtained at the end of training process (right) for one-dimensional elasticity problem with displacement controlled load (Section 3.1).

Table 3: Coefficient of determination (R² score) for various variables of interest in different solid mechanics problems.

| Problem Code               | \( u_1 \) | \( u_2 \) | \( \sigma_{11} \) | \( \sigma_{12} \) | \( \varepsilon_{11}^{el} \) | \( \varepsilon_{12}^{el} \) | \( \varepsilon_{22}^{el} \) | \( T \) |
|----------------------------|-----------|-----------|-------------------|-------------------|------------------|------------------|------------------|------|
| 1D-FGM-ELAS-DIRCH          | 99.94%    | -         | < 0               | -                 | 95.14%           | -                | 95.14%           | -    |
| 1D-FGM-ELAS-NEU            | 99.99%    | -         | < 0               | -                 | 99.99%           | -                | 99.99%           | -    |
| 1D-ELAS-BF                 | 99.98%    | -         | 99.65%            | -                 | 99.65%           | -                | 99.65%           | -    |
| 1D-ELAS-THERMO-ELAS        | 99.99%    | -         | < 0               | -                 | 97.61%           | -                | 97.61%           | -    |
| Kirsch’s Problem           | 99.89%    | 99.02%    | 97.52%            | 99.29%            | 89.75%           | 97.32%           | 99.29%           | 88.05%|
| 2D-FGM-ELAS-NEU            | 99.83%    | 99.99%    | 83.01%            | 80%               | 89.75%           | 97.32%           | 99.29%           | 88.05%|
| 2D-FGM-ELAS-DIRCH          | 99.92%    | 99.99%    | 81.78%            | 90.81%            | 89.75%           | 97.32%           | 99.29%           | 88.05%|
| 2D-FGM-THERMO-ELAS         | 99.99%    | 99.99%    | 88.84%            | 70.91%            | 73.92%           | 99.93%           | 99.95%           | 99.97%|

3.2 1D-FGM-ELAS-NEU

The domain is again assumed to be spanning from \( x = 0 \) to \( x = 1 \), while the cross-section is uniform. With the same functionally graded elastic modulus, the differential equation governing the elastic response remains the same as the last problem (see Table 1). While the left end of the bar is fixed, a unit traction \( t \) is applied on the right end such that \( u(0) = 0, \ t(1) = 1 \). The analytical solution for this case is given as:

\[
  u(x) = \frac{2x + x^2}{2}
\]  

The training dataset remains the same, i.e., the set of nodes on the domain. While the neural network architecture implicitly ensured the Dirichlet boundary conditions, implementation of Neumann boundary condition is ensured in the definition of loss function given in Table 1. Since the loss function is internal energy minus the work done by external forces, unlike the other use cases of a neural network, the loss function does have a physical significance in the case of
Figure 6: Decline in physics-informed loss (left) and comparison between analytical and predicted solution obtained at the end of training process (right) for one-dimensional elasticity problem with force controlled load (Section 3.2).

PINN. The loss can be expected to converge to zero only in situations where the internal energy stored is equal to the work done. Otherwise, it can very well either saturate to a negative value if work done by the external force is more than the energy stored or can saturate to a positive value if the loss is only composed of internal energy. It is also to be noted that loss function is not always internal energy minus the work done by external force even when the later can be a non-zero quantity. For example, in previous example the loss function was only composed of internal energy even though work done by external force would be finite. So whenever, the Dirichlet boundary condition is satisfied by a transformation abstracted in Equation (8), the work done by the corresponding reaction force must not be subtracted in the loss function.

The comparison between analytical and PINN solutions for this case is shown in Figure 6. It can be observed that the loss converges to a negative value as the work done by an external force is more than that of strain energy stored in the bar. Moreover, a good match is observed between the analytical and PINN predicted solution. The $R^2$ score for different variables is given in Table 3.

3.3 1D-ELAS-BF

In the last two examples, the use of PINN to implement Dirichlet and Neumann boundary conditions was discussed. We consider one more example with body force to demonstrate the robustness of the PINN framework. In this case, we will assume the elastic modulus to be unity over the domain. Other properties such as area of cross-section, mass density are assumed to be unity. The governing equations are given in Table 1. Boundary conditions are applied such
that $u(0) = 0$, $t(1) = 1$. The analytical solution for this case is given by:

$$u(x) = \frac{4x - x^2}{2}$$

Unsurprisingly, the PINN model can match the analytical solution in this case as well. The comparison is shown in Figure 7 and $R^2$ score was given in Table 3.

![Figure 7: Decline in physics-informed loss (left) and comparison between analytical and predicted solution obtained at the training process (right) for 1D mechanics problem with force controlled analysis and body force (Section 3.3).](image)

It is quite remarkable that the PINN model can perfectly fit solutions to differential equations under different boundary conditions. However, one cannot just expect the black-box PINN model to fit the solution of all possible differential equations. For example, a complex PINN model that is a non-linear function can never fit a differential equation where the solution $u$ is a constant value or linear in $x$. A practical example would be estimating the displacement field in a coarsely discretized multi-phase composite microstructure. Due to the coarse discretization, the true displacement would be piece-wise linear functions. Moreover, the gradient at the interface of two adjacent phases can be very high depending on the difference in their stiffness. Hence, it will be difficult for PINN to fit that displacement field with a high displacement gradient at the interface of phases. However, this inability of PINN need not necessarily be a disadvantage because coarse discretization is only a problem pertaining to the modeling of materials. In reality, displacement fields in materials will be reasonably smooth unless there is a discontinuity in the material due to cracks or manufacturing defects.
It is to be noted that all the discussed examples were selected such that the solution is quadratic in nature. While it is often challenging to comment on the nature of the black-box PINN function, it can undoubtedly be said from the excellent match that the models were just aptly parameterized to fit solutions of quadratic nature.

### 3.4 1D-FGM-THERMO-ELAS

In this section, we demonstrate the ability of PINN to estimate thermo-mechanical response of the functionally graded material. Let’s again consider a 1D bar $\Omega$ from $x = 0$ to $x = 1$. The thermo-elastic response of the object is governed by the differential equations given in Table 1. Let the elastic modulus of object vary as $E(x) = 1 / (1 + x)$ and conductivity to vary as $k(x) = 10 / (1 + x)$. The mechanical and thermal boundary conditions are imposed such that $u(0) = 0; \ u(1) = 1; \ T(0) = 0; \ T(1) = 1$ where $u(x)$ and $T(x)$ are respectively the displacement and temperature fields. The coefficient of thermal expansion $\alpha$ and reference temperature $T_0$ are assumed to be unity and zero respectively. The analytical solution for the purpose of comparison can be derived to be:

$$u(x) = \frac{x^3}{9} + \frac{14}{27}x^2 + \frac{10}{27}x$$  \hspace{1cm} (15)

and

$$T(x) = \frac{2x + x^2}{3}$$  \hspace{1cm} (16)

The usual steps described in last section are followed in order to solve the pair of differential equations using PINN. The domain is discretized into a set of uniformly spaced $N = 50$ nodes. The neural network $\tilde{v}(x) = (\tilde{v}_1(x) \ \tilde{v}_2(x))^T$, is transformed as follows to ensure satisfaction of Dirichlet boundary conditions:

$$\begin{bmatrix}
\tilde{u}(x) \\
\tilde{T}(x)
\end{bmatrix} = \begin{bmatrix}
x + (1-x)\tilde{v}_1(x) \\
x + (1-x)\tilde{v}_2(x)
\end{bmatrix}$$  \hspace{1cm} (17)

The neural network architecture of $v(x)$ is given in Table 2. Further, the loss for training the PINN is given in Table 1. The model is sufficiently trained and comparison between analytical and PINN predicted solutions can be observed in Figure 8. The $R^2$ score for primary and secondary variables are given in Table 3.

### 3.5 Kirsch’s Problem

Let us consider an infinite rectangular plate with a circular hole. Let us assume that the circular hole has a radius of 0.1 and is centered at $(0, 0)$. Moreover, the plate’s thickness and the material’s elastic modulus are assumed to be unity.
Physics-informed neural network

Figure 8: Reduction is loss with epoch (left), comparison of analytical solutions with predicted displacement field (middle) and temperature field (right) obtained at the end of training process (Section 3.4).

with appropriate units. Uniaxial traction \( t = [1 \ 0]^T \) is applied in x-direction on both ends of the plate. The problem can be solved by considering a quarter of the infinite plate and applying Dirichlet boundary conditions as per symmetry. Moreover, for the purpose of simulation, the domain is assumed to be finite, as shown in Figure 9.

Figure 9: Domain, boundary conditions and applied load for Kirsch’s problem (Section 3.5).

Let us say that \( \tilde{v}(x) : \Omega \rightarrow \mathbb{R}^2 \) is the neural network model which maps spatial coordinates vector \( (x) \) to a real-valued vector in \( \mathbb{R}^2 \). \( \Omega \) represents the set of points that constitute the plate while \( \Gamma \) is the set of points on the boundary of the plate. The details of neural network \( \tilde{v}(x) : \Omega \rightarrow \mathbb{R}^2 \) is given in Table 2. To ensure the satisfaction of Dirichlet boundary conditions, the following transformation is applied \( \tilde{u}(x) = x \odot \tilde{v}(x) \), where \( \tilde{u}(x) \) gives displacement vector and \( \odot \) represents Hadamard product. The governing equations and corresponding loss function is given in Table 1. Under the plane stress conditions, the material constitutive matrix is given in Equation (5). Distribution of nodes and corresponding weights \( (w(x)) \) are shown in the Figure 10. It is to be noted that weights are required for numerical
integration in case nodes are non-uniformly distributed over the domain. In the case of uniform distribution of nodes, as in the case of other examples, the weights would be a constant value.

The PINN model is trained for 2500 epochs in this case. The loss, which is the internal strain energy - work done decreased rapidly at the beginning of the training process and then saturated to a value of approximately \(-0.512\). The reduction in the loss vs. epochs is shown in Figure 11. In order to better understand the training process, it is worthwhile to look at the distribution of activation functions, weights and biases, and gradients for weight updates throughout the training process. They are given in Appendix: Figures 26 to 28. The comparison between predicted and analytical solutions is shown in Figure 12, while the same for stress fields can be found in Figure 13. It is observed that the predicted fields are in good agreement with the analytical solution. The R\(^2\) score for various variables can found in Table 3.

### 3.6 2D-FGM-ELAS-NEU

Let us consider a rectangular domain \(\Omega\) as shown in the Figure 14. As shown in the figure, both the degrees of freedom of nodes at the bottom edge are constrained to move. The elastic modulus of the plate is varying along \(x_2\) direction and is given as \(E(x_1, x_2) = 1 / (1 + x_2)\).

A uniform traction \(t\) is applied on the top boundary. Let us say that \(\tilde{v}(x) : \Omega \rightarrow \mathbb{R}^2\) is the neural network model which maps spatial coordinates vector \(x\) in domain \(\Omega\) to a real-valued vector in \(\mathbb{R}^2\). \(\Omega\) represents the set of points that
constitute the plate, while $\Gamma$ is the set of points on the boundary of the plate. In order to satisfy Dirichlet boundary conditions, the following transformation is applied:

$$
\tilde{u}(x) = \begin{bmatrix}
\tilde{u}_1 \\
\tilde{u}_2
\end{bmatrix} = \begin{bmatrix}
x_2 \\
x_2
\end{bmatrix} \circ \begin{bmatrix}
\tilde{v}_1 \\
\tilde{v}_2
\end{bmatrix}
$$

(18)

where $u(x)$ gives displacement vector and $\circ$ represents Hadamard product. Decrease in loss (internal strain energy - work done by an external force) with epoch count as the PINN is trained shown in Figure 15. Comparisons between displacement and stress fields predicted by PINN model and the Abaqus FEM simulation are shown in Figure 15 and Figure 16, and 17. A good match between the Abaqus and the PINN predicted solution is observed. The $R^2$ score, for this case is given in Table 3. It is to be noted that for $\sigma_{22}$ the $R^2$ scores are negative, which usually means quite a bad prediction except when the variance in the target variable is very close to zero. The true value of $\sigma_{22}$ is more or less constant over the domain where $R^2$ score is negative. This results in small variance in true target variable and hence a small value in denominator in calculation of $R^2$ score (see Equation (11)). Therefore, even a small absolute error would result in a negative $R^2$ score.
3.7 2D-FGM-ELAS-DIRCH

While we have seen how to train the PINN model on functionally graded material with Neumann boundary conditions, let’s consider the same domain and material properties with a different boundary condition. A uniform displacement boundary condition is applied on the top edge as shown in the Figure 18. Since there is no traction, in this case, the loss function will only have the internal strain energy component (see Table 1).

\[
\tilde{\mathbf{u}}(x) = \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} = \begin{bmatrix} x_2 \tilde{v}_1 \\ x_2/3L + (3L - x_2)x_2 \tilde{v}_2 \end{bmatrix}
\]

Decrease in loss (internal strain energy) with epoch count as the PINN is trained is shown in Figure 19. Comparisons between stress and displacement fields obtained from PINN and using Abaqus simulation are done in Figures 20 and 21. \( R^2 \) score for this case also was given in Table 3.

3.8 2D-FGM-THERMO-ELAS

Finally, let us consider a thermo-mechanics problem on 2D domain. The domain and boundary conditions are shown in the Figure 22. Both elastic modulus \( E(x) \) and conductivity \( k(x) \) varies along y-axis such that \( E(x_1, x_2) = 1 / (1 + x_2) \)
and \( k(x_1, x_2) = 10 / (1 + x_2) \). The coefficient of thermal expansion \( \alpha \) is assumed to be unity and the reference temperature \( T_0 \) is assumed to be zero.

The differential equations governing the thermo-elastic response of object is given in Table 1. All primary variables \( u_1, u_2 \) and \( T \) are set to zero on the bottom edge of the plate. The top edge is subjected to vertical displacement of one unit i.e., \( u_2(x_2 = 3L) = 1 \), while temperature of top edge is fixed at one unit \( T(x_2 = 3L) = 1 \). In order to implement boundary conditions in PINN framework, the blackbox function \( \tilde{v}(x) \) is modified as:

\[
\begin{bmatrix}
\tilde{u}_1(x) \\
\tilde{u}_2(x) \\
\tilde{T}(x)
\end{bmatrix}
= \begin{bmatrix}
x_2 \tilde{u}_1(x) \\
x_2/(3L) + x_2(3L - x_2) \tilde{v}_2(x) \\
x_2/(3L) + x_2(3L - x_2) \tilde{v}_3(x)
\end{bmatrix}
\]

Figure 13: Comparison between PINN predicted stress fields and corresponding analytical solutions (Section 3.5).
Figure 14: Domain and boundary conditions for functionally graded material object under force load defined in Section 3.6.

The architecture of neural network $v(x)$ is given in Table 2. The loss function used in training the model is given in Table 1. The decline in loss as the PINN model is trained can be seen in Figure 23. After training the model, the PINN predictions are compared with solutions obtained using Abaqus simulation. The comparison between primary fields $u_1(x_1), u_2(x_1), T(x)$ and stress fields $\sigma_{11}(x), \sigma_{22}(x)$ and $\sigma_{12}(x)$ can be observed in Figure 24 and Figure 25, respectively. The $R^2$ score, for different variables is given in Table 3.

4 Conclusions

This study has used PINN to solve solid mechanics and coupled thermo-mechanics problems of objects with functionally graded properties. Through various one and two-dimensional examples, the efficacy of PINNs in accurately predicting primary target variables has been established. On the other hand, it has also been found that PINNs currently do not perform very well in predicting secondary variables such as stress fields. When compared to the finite element
Figure 15: Reduction in loss as the PINN is trained to predict elastic response of two-dimensional functionally graded material object defined in Section 3.6.

simulation method, PINN is advantageous in terms of being meshless and flexible enough that various changes in the form of material properties, domain, and boundary conditions can be incorporated with much ease.

At the same time, there are a few drawbacks of the PINN-based method. There is no knowledge on how to design the neural network given a set of differential equations. The choice of activation function used in PINN influences the kind of functions it can fit. Hence, activation functions play a much significant role in PINN as compared to other neural networks. The PINN framework is also very sensitive to convergence criteria of losses. It is because a small change in loss can sometimes result in a drastic change in the obtained solutions.

The current challenges in using PINN only open the door to further avenues of research. Studying how to design PINN architectures, improving the accuracy of secondary variables, and consistent convergence criteria will be beneficial. Research along the lines of transfer learning using PINN and parallelized training of distributed PINNs will be of great significance.

5 Appendix

A neural network is often debugged by looking at various variables such as gradients, activation values, weights, and biases. The evolution of those variables while training the model gives us about model architecture’s efficacy. While training PINN, a slow training speed was observed. We consider the Kirsch’s problem discussed earlier in the Section 3
and plot the distribution of various variables to get insight into training. It is worthwhile to mention that there are a few open-source libraries such as Tensorboard (https://www.tensorflow.org/tensorboard), which can help in generating such plots with much ease. Though, in this example, all the plots are generated manually in python.

In Figure 26, we first look at the distribution of activation values in different hidden layers as the model is trained. The distribution of activation values at the start of the training process would be random due to the random initialization of weights. However, as the training happens, the distribution should be saturated. Moreover, mean activation close to zero with a low standard deviation is not desired as it makes the gradients zero and hampers the training process. In
Figure 26, it is observed that mean of activation values are not zero, which does not impede training. Moreover, the activation values for Layer 2 and Layer 3 have not saturated till 2000 epochs.

In Figure 27, we also look at the means and standard deviations of weights and biases in different layers. We observe that distributions of weights in different layers have not saturated in 2000 epochs as the standard deviation keeps on increasing.

We finally look at Figure 28, which shows the distribution of gradients in different layers. We see that the gradients are non-zero at the very start of the training process. Hence, the PINN model trains rapidly for the initial few epochs. However, after a few epochs, the gradients become very close to zero results in slow training of the model.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, ISBN 0262035618, 2016.

[2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, ISSN 1476-4687.

[3] Y. Huang, Y. Chen, Survey of State-of-Art Autonomous Driving Technologies with Deep Learning, in: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 221–228, 2020.

[4] J. Huang, J. Chai, S. Cho, Deep learning in finance and banking: A literature review and classification, Front. Bus. Res. China 14 (1) (2020) 13, ISSN 1673-7431.

[5] B. Tang, Z. Pan, K. Yin, A. Khateeb, Recent Advances of Deep Learning in Bioinformatics and Computational Biology, Front. Genet. 10 (2019) 214, ISSN 1664-8021.

[6] S. F. Ardabili, A. Mosavi, P. Ghamisi, F. Ferdinand, A. R. Varkonyi-Koczy, U. Reuter, T. Rabczuk, P. M. Atkinson, Covid-19 outbreak prediction with machine learning, Algorithms 13 (10) (2020) 249.

[7] A. R. Peeketi, R. K. Desu, P. Kumbhar, R. K. Annabattula, Thermal analysis of large granular assemblies using a hierarchical approach coupling the macro-scale finite element method and micro-scale discrete element method through artificial neural networks, Comput. Part. Mech. 6 (4) (2019) 811–822.

[8] A. Mosavi, M. Salimi, S. Faizollahzadeh Ardabili, T. Rabczuk, S. Shamshirband, A. R. Varkonyi-Koczy, State of the art of machine learning models in energy systems, a systematic review, Energies 12 (7) (2019) 1301.

[9] S. Masubuchi, E. Watanabe, Y. Seo, S. Okazaki, T. Sasagawa, K. Watanabe, T. Taniguchi, T. Machida, Deep-learning-based image segmentation integrated with optical microscopy for automatically searching for two-dimensional materials, NPJ 2D Mater. Appl. 4 (1) (2020) 3, ISSN 2397-7132.
[10] M. Raj, S. Thakre, R. K. Annabattula, A. K. Kanjarla, Estimation of Local Strain Fields in Two-Phase Elastic Composite Materials Using UNet-Based Deep Learning, Integr. Mater. Manuf. Innov. 10 (3) (2021) 444–460, ISSN 2193-9772.

[11] B. Liu, N. Vu-Bac, T. Rabczuk, A stochastic multiscale method for the prediction of the thermal conductivity of Polymer nanocomposites through hybrid machine learning algorithms, Compos. Struct. 273 (2021) 114269.

[12] S. Merity, N. S. Keskar, R. Socher, Regularizing and Optimizing LSTM Language Models, CoRR abs/1708.02182.

[13] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536, ISSN 1476-4687.

[14] K. Hornik, M. B. Stinchcombe, H. L. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[15] K.-S. Oh, K. Jung, GPU implementation of neural networks, Pattern Recognition 37 (6) (2004) 1311–1314, ISSN 0031-3203.

[16] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A Survey on Deep Transfer Learning, in: V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, I. Maglogiannis (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2018, Springer International Publishing, Cham, ISBN 978-3-030-01424-7, 270–279, 2018.

[17] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, ISSN 0021-9991.

[18] E. Weinan, Y. Bing, The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems, Commun. Math. Stat. 6 (1) (2018) 1–12, ISSN 2194-671X.

[19] C. Anitescu, E. Atroshchenko, N. Alajlan, T. Rabczuk, Artificial neural network methods for the solution of second order boundary value problems, Comput. Mater. Contin. 59 (1) (2019) 345–359.

[20] E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Comput. Methods Appl. Mech. Eng. 362 (2020) 112790.

[21] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theor. Appl. Fract. Mech. 106 (2020) 102447.
[22] S. Goswami, C. Anitescu, T. Rabczuk, Adaptive fourth-order phase field analysis for brittle fracture, Comput. Methods Appl. Mech. Eng. 361 (2020) 112808, ISSN 0045-7825.

[23] S. Goswami, C. Anitescu, T. Rabczuk, Adaptive fourth-order phase field analysis using deep energy minimization, Theor. Appl. Fract. Mech. 107 (2020) 102527.

[24] W. Li, M. Z. Bazant, J. Zhu, A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches, Comput. Methods Appl. Mech. Eng. 383 (2021) 113933.

[25] V. Dwivedi, B. Srinivasan, Solution of Biharmonic Equation in Complicated Geometries With Physics Informed Extreme Learning Machine, Journal of Computing and Information Science in Engineering 20 (6), ISSN 1530-9827.

[26] V. Dwivedi, B. Srinivasan, Physics Informed Extreme Learning Machine (PIELM)—A rapid method for the numerical solution of partial differential equations, Neurocomputing 391 (2020) 96–118, ISSN 0925-2312.

[27] V. Dwivedi, N. Parashar, B. Srinivasan, Distributed physics informed neural network for data-efficient solution to partial differential equations, 2019.

[28] V. Dwivedi, N. Parashar, B. Srinivasan, Distributed learning machines for solving forward and inverse problems in partial differential equations, Neurocomputing 420 (2021) 299–316, ISSN 0925-2312.

[29] G. K. Yadav, S. Natarajan, B. Srinivasan, Distributed PINN for linear elasticity - a unified approach for smooth, singular, compressible and incompressible media, Int. J. Comput. Methods 2021 (ja) (0) null.

[30] M. Koizumi, The concept of FGM, Ceram. Trans. 34 (1993) 3–10.

[31] M.-J. Pindera, J. Aboudi, A. M. Glæser, S. M. Arnold, Use of composites in multi-phased and functionally graded materials, Compos. B. Eng. 28 (1-2).

[32] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2017.

[33] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic Differentiation in Machine Learning: a Survey, J Mach Learn Res. 18 (153) (2018) 1–43.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu,
X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org, 2015.
Figure 17: Comparison of PINN predicted stress fields and corresponding Abaqus solution for problem defined in section Section 3.6.
Figure 18: Domain and boundary conditions for functionally graded material object under displacement controlled load defined in Section 3.7.
Figure 19: Reduction in loss vs epoch as the PINN is trained to predict elastic response of two-dimensional functionally graded material object with displacement load defined in Section 3.7.
Figure 20: Comparison between PINN prediction displacement fields and corresponding Abaqus solution for problem defined in Section 3.7.
Figure 21: Comparison between PINN predicted stress fields and corresponding Abaqus solution for problem defined in Section 3.7.
Figure 22: Domain and boundary conditions for two-dimensional functionally graded material object under displacement and temperature load defined in Section 3.8.

\[ k(x_1, x_2) = \frac{1}{1 + x_2} \]

\[ E(x_1, x_2) = \frac{1}{1 + x_2} \]
Figure 23: Reduction in internal strain and thermal energy as the model is trained to predict thermo-mechanical response of two-dimensional functionally graded material object defined in Section 3.8.
Figure 24: Comparison between PINN predicted displacement and temperature fields and corresponding Abaqus simulation for problem defined in Section 3.8.
Figure 25: Comparision between PINN predicted stress fields and corresponding Aabqus solution for problem defined in Section 3.8.
Figure 26: Distribution of activation function values for different layers for PINN trained on problem defined in Section 3.5. The thick line represents the mean while the faint bars represents the spread about the mean of activation values at various epochs.
Figure 27: Mean and standard deviation of weights and biases as the PINN is trained for problem defined in Section 3.5.
Figure 28: Mean and standard deviation of gradients as the PINN trained for problem defined in Section 3.5. It is to be noted that zero gradients would result in no further weight updates.