An Energy Approach to the Solution of Partial Differential Equations in Computational Mechanics via Machine Learning: Concepts, Implementation and Applications

Preprint · August 2019

CITATIONS
0

READS
2

8 authors, including:

Cosmin Anitescu
Bauhaus Universität Weimar
26 PUBLICATIONS **319** CITATIONS

SEE PROFILE

An Energy Approach to the Solution of Partial Differential Equations in Computational Mechanics via Machine Learning: Concepts, Implementation and Applications

E. Samaniego[a], C. Anitescu[b], S. Goswami[b], V.M. Nguyen-Thanh[c], H. Guo[c], K. Hamdia[c], T. Rabczuk[b,*], X. Zhuang[c]

[a]*School of Engineering and Departamento de Recursos Hídricos y Ciencias Ambientales, Universidad de Cuenca, Av. 12 de Abril s/n., Cuenca, Ecuador*
[b]*Institute of Structural Mechanics, Bauhaus-Universität Weimar, Marienstraße 15 99423 Weimar*
[c]*Institute of Continuum Mechanics, Leibniz Universität Hannover, Appelstraße 11, 30167 Hannover, Germany*

Abstract

Partial Differential Equations (PDE) are fundamental to model different phenomena in science and engineering mathematically. Solving them is a crucial step towards a precise knowledge of the behaviour of natural and engineered systems. In general, in order to solve PDEs that represent real systems to an acceptable degree, analytical methods are usually not enough. One has to resort to discretization methods. For engineering problems, probably the best known option is the finite element method (FEM). However, powerful alternatives such as mesh-free methods and Isogeometric Analysis (IGA) are also available. The fundamental idea is to approximate the solution of the PDE by means of functions specifically built to have some desirable properties. In this contribution, we explore Deep Neural Networks (DNNs) as an option for approximation. They have shown impressive results in areas such as visual recognition. DNNs are regarded here as function approximation machines. There is great flexibility to define their structure and important advances in the architecture and the efficiency of the algorithms to implement them make DNNs a very interesting alternative to approximate the solution of a PDE. We concentrate in applications that have an interest for Computational Mechanics. Most contributions that have decided to explore this possibility have adopted a collocation strategy. In this contribution, we concentrate in mechanical problems and analyze the energetic format of the PDE. The energy of a mechanical system seems to be the natural loss function for a machine learning method to approach a mechanical problem. As proofs of concept, we deal with several problems and explore the capabilities of the method for applications in engineering.

Keywords: Physics informed, Deep neural networks, Energy approach

*Corresponding author.
Email address: timon.rabczuk@uni-weimar.de (T. Rabczuk)

1. Introduction

Computational mechanics aims at solving mechanical problems using computer methods. These mechanical problems can originate from the study of either natural or engineered systems. In order to describe their behaviour in a precise manner, mathematical models have to be devised. In engineering applications, these mathematical models are often based on partial differential equations (PDEs). When realistic models are considered, one has to resort to numerical methods to solve them. The idea is to look for an approximate solution for the problem, in a finite-dimensional space. Then, the problem reduces to find a finite set of parameters that define this approximate solution. Conventional ways to tackle the solution of PDEs are the finite element method (FEM) [1], mesh-free methods [2], and isogeometric analysis [3].

In recent times, the use of deep neural networks (DNNs) have led to outstanding achievements in several areas, such as visual recognition [4]. Feed-forward neural networks are devised to approximate the target functions. The approximating functions depend on certain parameters (the weights and the biases) that have to be "learned" by means of a "training" process. Then, it is conceivable that Neural Networks may be used to approximate the solution of a PDE. This is the perspective that we are going to adopt with respect to DNNs in this article: they are function approximation machines.

The success of DNNs in important learning tasks can be related to the development of very powerful computational tools. Libraries such as Tensorflow [5] and PyTorch [6] provide the building blocks to devise learning machines for very different problems. Interfaces that allow to code in a readable languages like Python and the availability of optimized numerical algorithms leads to achieving a near-mathematical notation, reminiscent of computing plataforms like FEniCS [7].

There are some works in which the idea of using DNNs to solve PDEs has been pursued [8]. However, in general, they have dealt with the strong form of the PDE, leading to an approach based on collocation [9]. Then, the training process is based in devising an objective function, the empirical loss function, whose minimization leads to the fulfilment of the governing equations. Also, the problems dealt with are not in general problems directly related to engineering applications.

A way to approach the solution of a PDE is to write it as a variational problem [10]. From the mechanical point of view, the corresponding functional has the meaning of an energy. Given the fact that the training process in machine learning can be regarded as a process of minimizing the loss function, it seems natural to regard the energy of the system as a very good candidate for this loss function. In addition, the near-mathematical syntax achieved by the platforms associated to Tensorflow and Pytorch imply a high degree of readability and ease of implementation of the PDE solver. Moreover, once the variational problem is approximated in a finite-dimensional space, it becomes an optimization problem, which is very convenient given that machine learning libraries are specially oriented to optimization techniques.

It is worth mentioning that our approach points at solving the PDEs by means of DNNs as an approximation strategy. In that respect, what we propose is different from approaches such as [11]. They use labeled data from numerical simulations (although it could be obtained from experiments, in principle) to help the solution of a Boundary Value Problem in some

specific aspect, where detailed knowledge of the phenomena that is being modelled is lacking. For instance, [12] replaces the constitutive model for a data-driven model. In summary, they use machine learning to build a surrogate model, while we use it to build the approximation space.

In this work, we explore the possibility of using a DNN based solver for PDEs. The paper is organized as follows. In Section 2, we introduce the reader to the generalized problem setup. Section 3 provides a brief introduction to DNN. The strategy for solving PDEs with DNNs based on collocation method and deep energy method is explained in Section 4. The implementation technique is explained in Section 5. To explain the implementation, we have used snippets from the code. Some representative applications in computational mechanics are tackled in Section 6, to explore the possibilities of this approach. Finally, Section 7 concludes the study by summarizing the key results of the present work.

2. Mathematical modeling of continuous physical systems

The general aim of this work is to set the foundations for a new paradigm in the field of computational mechanics that enriches deep learning with the long standing developments in mathematical physics. To that end, we first consider a generalized PDE and later briefly introduce the energy approach.

2.1. Partial Differential Equations

Consider a generalized PDE expressed as:

$$\mathcal{N}[\boldsymbol{u}] = 0 \quad \text{on} \quad \Omega, \tag{1}$$

where $\mathcal{N}$ stands for a (possibly non-linear) differential operator acting upon a (possibly vector-valued) function, $\boldsymbol{u}$. In addition, $\boldsymbol{u}$ must satisfy both the Dirichlet and the Neumann-boundary conditions. The solution of this PDE subjected to appropriate boundary conditions is referred to a boundary value problem (BVP). Many problems in science and engineering can be written as specific cases of this generalized format. Sometimes, this version of the corresponding BVP is called the strong form.

Let us consider how this general setting can be applied to the case of a linear elastic body. We will have the displacement, $\boldsymbol{u}$ as the primal variable. Then, the second order total strain tensor,$\boldsymbol{\epsilon}(\boldsymbol{u})$ is defined as follows:

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}\Big(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\Big). \tag{2}$$

For the sake of simplicity, in Eq. (2) we have dropped the explicit dependence of the involved fields on position, $\boldsymbol{x}$ and time, t. The Cauchy stress tensor, $\boldsymbol{\sigma}(\boldsymbol{\epsilon})$ can be computed as:

$$\boldsymbol{\sigma}(\boldsymbol{\epsilon}) = \mathbb{C} : \boldsymbol{\epsilon}, \tag{3}$$

where $\mathbb{C}$ is the elasticity tensor. The momentum balance equation, neglecting the inertial effects and considering zero body forces reads as:

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}. \tag{4}$$

Hence, we define Eq. (1) as:

$$\mathcal{N}[\boldsymbol{u}] = \nabla \cdot \boldsymbol{\sigma}\big(\boldsymbol{\epsilon}(\boldsymbol{u})\big), \tag{5}$$

which for sufficiently smooth $\boldsymbol{u}$ can be solved by a collocation-type method as detailed in Section 4.

2.2. Energy Approach

For the energy approach to a BVP, consider the following problem

$$\min_{\boldsymbol{u}} \mathcal{E}[\boldsymbol{u}], \tag{6}$$

where $\boldsymbol{u}$ is constrained by Dirichlet boundary conditions. Here, we will assume that the total variational energy of the system, $\mathcal{E}$ is such that there exists a unique solution of the problem defined in Eq. (6) and is the same as the solution of Eq. (1). In that case, Eq. (1) is called the Euler-Lagrange equation of the variational problem defined in Eq. (6).

The first step to go from the variational energy formulation to the corresponding strong form is to find a stationary state of $\mathcal{E}$ by equating its first variation to zero:

$$\delta\mathcal{E}[\boldsymbol{u}, \delta\boldsymbol{u}] = \frac{\mathrm{d}}{\mathrm{d}\tau}\mathcal{E}[\boldsymbol{u} + \tau\delta\boldsymbol{u}]\bigg|_{\tau=0} = 0, \tag{7}$$

which has to hold for any admissible $\delta\boldsymbol{u}$ (i.e., for any smooth enough function having homogeneous boundary conditions on the Dirichlet part of the boundary). The resulting expression is the so-called weak form of the problem defined by Eq. (1) plus appropriate boundary conditions. In mechanical problems, this corresponds to the principle of virtual work. The weak form is the point of departure of important discretization methods such as the FEM. A characteristic feature of the approach used in this contribution is that it deals directly with the minimization of the variational energy, $\mathcal{E}$ circumventing the need to derive the weak form from the energy of the system explicitly.

In order to illustrate this approach, let us again consider a linear elastic body. One can define the stored elastic strain energy, $\Psi(\boldsymbol{\epsilon})$ as:

$$\Psi(\boldsymbol{\epsilon}) = \frac{1}{2}\boldsymbol{\epsilon} : \mathbb{C} : \boldsymbol{\epsilon}. \tag{8}$$

Notice that the Cauchy stress tensor, $\boldsymbol{\sigma}(\boldsymbol{\epsilon})$ can be computed as follows:

$$\boldsymbol{\sigma}(\boldsymbol{\epsilon}) = \frac{\partial\Psi}{\partial\boldsymbol{\epsilon}}. \tag{9}$$

Then, for the case of zero body forces and homogeneous Neumann boundary conditions, we can define the energy of the solid as

$$\mathcal{E}[\boldsymbol{u}] = \int_{\Omega} \Psi\big(\boldsymbol{\epsilon}(\boldsymbol{u})\big) d\Omega. \tag{10}$$

3. Deep Neural Networks for PDE discretization

Artificial Neural Networks (ANNs) are learning machines loosely inspired by the biological neural networks. The general idea is to transform some given input into an output. For this, there are several units, called neurons, whose state can change depending on the inputs and the functional relations between these states. ANNs are usually represented by a graph, whose nodes are the neurons.

Given an input $\boldsymbol{x}$, an ANN would map it to an output $\boldsymbol{u}_p(\boldsymbol{x})$. The functional structure of the ANN is defined by a set of parameters (which can be collected in a vector, $\boldsymbol{p}$). Deep Neural Networks are ANNs obtained by means of the composition of simple functions:

$$\boldsymbol{u}_p(\boldsymbol{x}) = A_L(\sigma(A_{L-1}(\sigma(\dots\sigma(A_1(\boldsymbol{x}))\dots)))), \tag{11}$$

where A_l (with $l = 1, 2, \dots, L$) are affine mappings and σ is a the activation function applied element-wise. It is important to note that the non-linearity of the DNNs come from the activation function. Due to the graphical representation of the composition, each of these functions is called a layer, where L denotes the number of layers of the network. So, loosely speaking, we say that an ANN is a DNN with one layer, barring the input and the output layers..

Each affine mapping A_l can be defined by a (in general not square) matrix and a vector. The elements of the matrix are called *weights* and the elements of the vector are called *biases*. They together parametrize the function $\boldsymbol{u}_p(\boldsymbol{x})$, and can be grouped in a vector $\boldsymbol{p}$. In the following, we consider feed-forward fully-connected DNNs, where each neuron is connected to all the neurons in the adjacent layers. Other types of neural networks can be considered as well, such as convolutional neural networks [13], where the layers and the connections between them are organized in an hierarchical structure.

Once the architecture of the network has been decided, that is, once the number of layers, the number of neurons per layer, and the activation functions have been frozen, defining $\boldsymbol{u}_p(\boldsymbol{x})$ boils down to determining $\boldsymbol{p}$. The process of determining these parameters (the weights and the biases) is called *training* the network. Usually, when solving problems with machine learning, this entails having a large amount of input and output data. In the approach used here, data for training is generated from evaluations of the PDEs at given point in the domain, as well as from certain points on the boundary. This issue is explained more in detail below.

4. Solution strategies

4.1. Collocation Methods

Collocation methods are based on the solution of the BVP in strong form. Given a set of points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$ in the domain Ω, the idea is to evaluate $\mathcal{N}[\boldsymbol{u}_p]$ at each point $\boldsymbol{x}_i$ and construct a loss function whose optimization tends to impose the condition $\mathcal{N}[\boldsymbol{u}_p](\boldsymbol{x}_i) = 0$. For instance, loss functions based on the mean square error of $\mathcal{N}[\boldsymbol{u}_p](\boldsymbol{x}_i)$ can be used.

4.2. Deep Energy Method

The main idea of the method advocated in this contribution is to take advantage of the variational (energetic) structure of some BVPs. To that end, the energy of the system is used as the loss function for the DNN, as proposed by [14]. Due to its mechanical flavor, we name it the Deep Energy Method (DEM) here. One of the key ingredients is to approximate the energy of the body by a weighted sum of the energy density at integration points. Then, the following form for the loss function, $\mathcal{L}(\boldsymbol{p})$ is obtained:

$$\mathcal{E}[\boldsymbol{u}_p] \approx \mathcal{L}(\boldsymbol{p}) = \sum_i \Psi\big(\boldsymbol{\epsilon}(\boldsymbol{u}_p(\boldsymbol{x}_i))\big) w_i, \tag{12}$$

where $\boldsymbol{u}_p(\boldsymbol{x}_i)$ is the displacement approximation function evaluated at the integration point, $\boldsymbol{x}_i$ and its corresponding weight, w_i. A key issue is the prescription of the boundary conditions. In an variational energy formulation, Neumann boundary conditions can in principle be imposed in a natural way. However, Dirichlet boundary conditions have to be imposed in a somehow more involved way, since DNN approximation functions do not fulfill the Kronecker delta property. We will address these issues in Section 6.

4.2.1. Optimization in machine learning

Once the architecture of a DNN and the empirical loss function have been selected, finding the parameters $\boldsymbol{p}$ (weights and biases) that define the approximation $\boldsymbol{u}_p$ of the the unknown $\boldsymbol{u}$ is the main task. Then, having in mind that the loss function can be expressed as

$$\mathcal{L}(\boldsymbol{p}) = \sum_i f_i(\boldsymbol{p}), \tag{13}$$

we end up facing a non-convex finite-sum optimization problem [15]:

$$\min_{\boldsymbol{p}} \sum_i f_i(\boldsymbol{p}). \tag{14}$$

In machine learning, first order methods are almost universal. In fact, one may say that most optimization strategies in machine learning are variants of the so-called gradient descent method. A typical iteration of this method is the following:

$$\boldsymbol{p}^{k+1} = \boldsymbol{p}^k + \gamma_k \nabla_p \left(\sum_i f_i(\boldsymbol{p}) \right), \tag{15}$$

where $\boldsymbol{u}_p^k$ is the DNN approximation of the unknown $\boldsymbol{u}$ parametrized by vector $\boldsymbol{p}^k$, which groups all weights and biases at iteration k. In addition, γ_k stands for a parameter called learning rate. Several variations of this algorithm can be considered. First of all, not all the points are always considered. In fact, stochasticity in the selection of the points considered in the finite sum is a general practice, resulting in the so-called stochastic gradient descent (SGD) algorithm. Acceleration strategies are also very common. Another very common variant implies the addition of a penalty term to avoid unbounded values of the parameters. Sometimes, even some approximation of second order information is included, for example, by means of the L-BFGS method, a quasi-Newton method.

One important issue related to the optimization problem is the structure of the search space. Recent mathematical results describe the presence of highly unfavorable properties of the topological structure of DNN spaces. This means that, although the space of functions generated by a DNN can be very expressive, the structure of the search space may cause difficulties for optimization algorithms [16]. However, some unexpected properties of the gradient-descent-based algorithms seem to be able to overcome these difficulties [17], at least for some applications. This an issue that is the object of intense research currently.

5. Implementation

In this section, we illustrate how the proposed DEM approach can be used for solving problems in the field of continuum mechanics. The primary goal of the approach is to

obtain the field variables by solving the governing partial differential equation, which in turn requires to compute the integral Eq. (10). We do this by using machine learning tools available in open-source libraries like TensorFlow and PyTorch. This allows us to use collective knowledge accumulated in a very vibrant community.

We describe the typical steps that one would have to follow for a TensorFlow implementation. The idea is to illustrate the general procedure. Specific details for each application are given below. The full source code will be published on GitHub or can be obtained by contacting the authors.

First, the geometry is generated using uniformly spaced points in the whole domain and obtain the training points, $\boldsymbol{X}_f$ and prediction points, $\boldsymbol{x}_{pred}$. Before we start to train the network, we initialize the weights of the network randomly from a Gaussian distribution using the Xavier initialization technique [18]. The network is initialized in the following manner:

```
    def initialize_NN(self,layers):
        weights = []
        biases = []
        num_layers = len(layers)
        for l in range(0, num_layers - 1):
            W = self.xavier_init(size=[layers[l], layers[l + 1]])
            b = tf.Variable(tf.zeros([1,layers[l + 1]]),
                dtype=tf.float32)
            weights.append(W)
            biases.append(b)
        return weights, biases

    def xavier_init(self, size):
        in_dim = size[0]
        out_dim = size[1]
        xavier_stddev = np.sqrt(2.0 / (in_dim + out_dim))
        w_init = tf.Variable(tf.truncated_normal([in_dim, out_dim],
                stddev=xavier_stddev))
        return w_init
```

where the network is initialized with *layers*. Once the weights are initialized, we begin training the neural network.

Next, we modify the neural network outputs in such a way so that the Dirichlet boundary conditions are exactly satisfied. This can be done as in the following listing, where uNN and vNN denote the output from the neural network.

```
    def net_uv(self,x,y,vdelta):
        X = tf.concat([x,y],1)
        uv = self.neural_net(X,self.weights,self.biases)
        uNN = uv[:,0:1]
        vNN = uv[:,1:2]
        u = (1-x)*x*uNN
        v = y*(y-1)*vNN
        return u, v
```

The neural network subroutine can be defined according to the type of activation function chosen between subsequent layers as in the following example:

```
    def neural_net(self,X,weights,biases):
```

```
        num_layers = len(weights) + 1

        H = 2.0 * (X - self.lb) / (self.ub - self.lb) - 1.0
        for l in range(0, num_layers - 2):
            W = weights[l]
            b = biases[l]
            H = tf.nn.relu(tf.add(tf.matmul(H, W), b))**2
        W = weights[-1]
        b = biases[-1]
        Y = tf.add(tf.matmul(H, W), b)
        return Y
```

Here $self.lb$ and $self.ub$ denote the lower bound and the upper bound of the inputs to the neural network, which implies the spatial co-ordinates.

In the next step, automatic differentiation is used to compute the displacement gradients and compute the variational energy at any point in the domain.

```
    def net_energy(self,x,y):
        u, v = self.net_uv(x,y)
        u_x = tf.gradients(u,x)[0]
        v_y = tf.gradients(v,y)[0]
        u_y = tf.gradients(u,y)[0]
        v_x = tf.gradients(v,x)[0]
        u_xy = (u_y + v_x)
        sigmaX = self.c11*u_x + self.c12*v_y
        sigmaY = self.c21*u_x + self.c22*v_y
        tauXY = self.c33*u_xy
        energy = 0.5*(sigmaX*u_x + sigmaY*v_y + tauXY*u_xy)
        return energy
```

Finally, we minimize the mean error of the energy losses computed in the previous step. For optimization, we use the Adam (adaptive momentum) optimizer followed by a quasi-Newton method (L-BFGS). Once the network is optimized, we predict the values of the field variables at x_{pred} points.

6. Applications

In this section, we explore the application of DEM to solve PDEs in various domains of continuum mechanics. The first section considers a one-dimensional problem and the results obtained using DEM is compared to the available analytical solution. Later, we discuss the application of DEM on linear elasticity, hyperelasticity. In the latter part of the section, we solve PDEs for coupled problems. For coupled problems, we address phase-field modeling of fracture, piezoelectricity and lastly the bending of a Kirchhoff plate.

The implementation has been carried out using the `TensorFlow` framework [19]. Details on the network architecture, such as number of layers, number of neurons in each layer, the activation function used etc. have been provided with each example.

6.1. DNN with ReLU activation functions and FE in 1D

ANNs are known to be universal approximators. Moreover, recent results in approximation theory show that DNNs using rectified linear units (ReLU) as activation functions can reproduce linear finite element spaces [20]. ReLU functions are defined by $ReLU(x) = max(0, x)$.

These results have been extended to more general Finite Element spaces using Sobolev norms in [21]. Here we illustrate the relationship between FE approximations and DNNs by means of a one-dimensional example, as done in [20].

Let us consider a discretization of the interval $[0, l]$ in the real line: $x_0 < x_1 < \ldots < x_n$, where $x_0 = 0$ and $x_n = l$. Then, consider a piecewise linear FE approximation of a function $u(x)$ taking values on $[0, l]$:

$$u^h(x) = \sum_{i=0}^{n} u_i N_i(x) \tag{16}$$

where $N_i(x)$ is a linear hat function with compact support in the subinterval $[x_{i-1}, x_{i+1}]$, such that $N_i(x_i) = 1$. For $i = 0$ and $i = n$, $N_i(x)$ has compact support in $[x_0, x_1]$ and $[x_{n-1}, x_n]$, respectively. Function $N_i(x)$, for intermediate nodes, can be expressed in terms of rectified linear unis $ReLU$ as follows:

$$N_i(x) = \frac{ReLU(x - x_{i-1}) - ReLU(x - x_i)}{h_i} - \frac{ReLU(x - x_i) - ReLU(x - x_{i+1})}{h_{i+1}} \tag{17}$$

where $h_i = x_i - x_{i-1}$

Rearranging terms, we get

$$N_i(x) = \frac{1}{h_i} ReLU(x - x_{i-1}) - \left(\frac{1}{h_i} + \frac{1}{h_{i+1}}\right) ReLU(x - x_i) + \frac{1}{h_{i+1}} ReLU(x - x_{i+1}) \tag{18}$$

Using Eq. 18 and Eq. 16, the following expression can be obtained:

$$u^h(x) = u_0 + \sum_{i=0}^{n-1} \left(\frac{\Delta_{i+1}}{h_{i+1}} - \frac{\Delta_i}{h_i}\right) ReLU(x - x_i) \tag{19}$$

where $\Delta_i = u_i - u_{i-1}$, for $i = 1, 2, \ldots, n$, and $\Delta_i = 0$, for $i = 0$. Then, u^h can be expressed as an ANN with one hidden layer, having x as input, $u^h(x)$ as output, and $ReLU$ as the activation function:

$$u^h(x) = u_0 + \sum_{i=0}^{n-1} w_i ReLU(x - x_i) \tag{20}$$

with

$$w_i = \left(\frac{\Delta_{i+1}}{h_{i+1}} - \frac{\Delta_i}{h_i}\right) \tag{21}$$

Notice that u^h is parametrized by the weights w_i, which can be grouped in a vector $\boldsymbol{w}$, and by the position of the nodes x_i (the biases of the hidden layer), which can be grouped in the vector $\boldsymbol{x}_p$. For fixed $\boldsymbol{x}_p$, and prescribing u_0, the vector value of the elements of $\boldsymbol{w}$ is defined by the parameters u_i, i.e., the values of u^h evaluated at the nodes x_i.

Consider the approximation in Eq. (20). Then, if $\Psi\big(\epsilon\big)$ is the one-dimensional linear elastic energy density, we can define an energy

$$\mathcal{E}(\boldsymbol{w}, \boldsymbol{x}_p) = \int_0^l \Psi\big(\epsilon(u^h(x; \boldsymbol{w}, \boldsymbol{x}_p))\big) dx. \tag{22}$$

in terms of the weights vector $\boldsymbol{w}$ and a vector containing the position of the nodes, $\boldsymbol{x}_p$:

$$\min_{\boldsymbol{w},\boldsymbol{x}_p} \mathcal{E}(\boldsymbol{w}, \boldsymbol{x}_p) \tag{23}$$

For fixed $\boldsymbol{x}_p$, the problem would be equivalent to solving a standard FE problem (remember the relationship between $\boldsymbol{w}$ and the nodal FE unknowns). When $\boldsymbol{x}_p$ is not fixed, from the mathematical point of view, we would be solving an adaptive FE problem. However, since the problem is now non-linear (and probably non-convex), there may be issues related to the use of typical machine learning algorithms to optimize the loss function defined in 23.

Although this 1D problem is illustrative of the meaning of the parameters of a DNN approximation, it is also clear that, for general situations, assigning physical meaning to these parameters would be very difficult. This is may be a consequence of the mesh-free character of DNN approximations for the solution of PDEs. This has several implications. Collocation points, for instance, are not to be confused with nodes in a Finite Element mesh. Points in a FE mesh are tightly related to the parameters of the approximation space, while collocation points are the inputs for the training process in the DNN approximation.

6.2. Linear Elasticity problem

In this section, the physics informed neural network is developed to solve the governing partial differential equation for linear-elastic problems using DEM approach. The solution of the displacement field is obtained by minimizing the stored elastic strain energy of the system. The problem statement is written as:

$$\begin{aligned} \text{Minimize:} \quad & \mathcal{E} = \Psi(\epsilon), \\ \text{where:} \quad & \Psi(\epsilon) = \int_{\Omega} f(\boldsymbol{x}) dx, \\ \text{and} \quad & f(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{\epsilon} : \mathbb{C} : \boldsymbol{\epsilon}, \\ \text{subject to:} \quad & \boldsymbol{u} = \overline{\boldsymbol{u}} \text{ on } \partial\Omega_D, \\ \text{and} \quad & \boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{t}_N \text{ on } \partial\Omega_N \end{aligned} \tag{24}$$

where $\Psi(\epsilon)$ denotes the stored elastic strain energy of the system expressed in terms of the strain tensor, $\epsilon(\boldsymbol{u})$, $\mathbb{C}$ represents the constitutive elastic matrix, $\overline{\boldsymbol{u}}$ is the prescribed displacement on the Dirichlet boundary, $\partial\Omega_D$ and $\boldsymbol{t}_N$ is the prescribed boundary forces on the Neumann boundary, $\partial\Omega_N$. Using the DEM approach, the homogeneous Neumann boundary conditions are automatically satisfied. The field variables are obtained by minimizing the mean error of the energy functional at the integration points. Additionally, the non-homogeneous Neumann boundary conditions are satisfied by minimizing the mean error at the integration points on the boundary. For optimization, the neural network is trained by using a combination of Adam optimizer and second-order quasi-Newton method (L-BFGS).

We shall now present the results obtained using DEM for two and three-dimensional linear elastic problems. Analytical solution is available for all these problems, hence the accuracy of the method is measured by computing the $\mathcal{L}_2$ error for the displacement field and the strain energy.

6.2.1. Pressurized thick-cylinder

The first example considers the benchmark problem of of a thick cylinder subjected to internal pressure under plane stress condition [22]. Due to symmetry in the geometry and boundary conditions, only a quarter section represented by an annulus is analyzed. The cylinder is subjected to internal pressure applied on the inner circular edge. The geometrical setup and the boundary conditions are shown in Fig. 1(a). In this example, we have considered $E = 1 \times 10^5$ and $\nu = 0.3$. The analytical solution, in terms of the stress components, is [23]:

$$\begin{aligned}
\sigma_{rr} &= \frac{R_a^2 P}{R_a^2 - R_b^2}\left(1 - \frac{R_a^2}{r^2}\right), \\
\sigma_{\theta\theta} &= \frac{R_a^2 P}{R_a^2 - R_b^2}\left(1 + \frac{R_a^2}{r^2}\right), \\
\sigma_{r\theta} &= 0, \\
u_{rad} &= \frac{R_a^2 P r}{E(R_b^2 - R_a^2)}\left(1 - \nu + \left(\frac{R_b}{r}\right)^2 (1+\nu)\right), \\
u_{exact} &= u_{rad}\cos\theta, \\
v_{exact} &= u_{rad}\cos\theta,
\end{aligned} \tag{25}$$

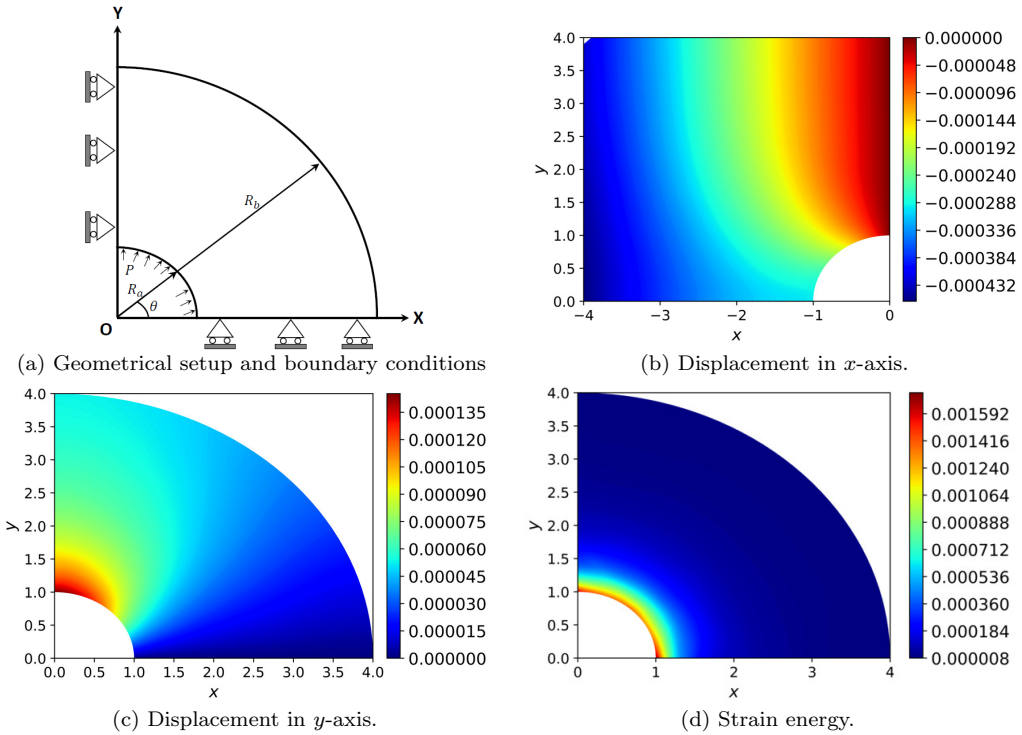
where R_a and R_b represents the inner and the outer radius of the cylinder, respectively with $R_a = 1$ and $R_b = 4$. P is the pressure exerted along the inner circular edge. In Eq. (25), u_{exact} and v_{exact} are the analytical solution of the displacement field in x and y-axis, respectively. The minimization problem reads as stated in Eq. (65). The Dirichlet and Neumann boundary conditions for the thick cylinder under internal pressure are:

$$\begin{aligned}
u(0, y) = v(x, 0) &= 0, \\
t_{N,x}(R_a, \theta) &= Pu, \\
t_{N,y}(R_a, \theta) &= Pv,
\end{aligned} \tag{26}$$

where u and v are the solutions of the elastic field in x and y-axis, respectively and $t_N(R_a, \theta)$ denotes the traction force which is expressed in terms of the applied internal pressure and the displacement field. To find the solution of the displacement field, the trial solution is defined as:

$$\begin{aligned}
u &= x\hat{u}, \\
v &= y\hat{v},
\end{aligned} \tag{27}$$

where $\hat{u}$ and $\hat{v}$ are obtained from the neural network. The solution of u and v are chosen in such a way that it satisfies all the boundary conditions as in [24, 25]. As a consequence, no component corresponding to the boundary loss is needed in the loss function. This significantly simplifies the objective function to be minimized. We have used a network with 3 hidden layers of 30 neurons each and $N_x \times N_y$ uniformly spaced points in the interior of the annulus, with $N_x = N_y = 80$. The inner circular edge is discretized using N_{Bound} uniformly spaced points to apply the internal pressure in terms of traction force, with $N_{Bound} = 80$. For the input layer and the first two hidden layers, we have considered rectified linear units, $\texttt{ReLU}^2$ activation function; whereas for the last layer connected to the output layer, linear



(a) Geometrical setup and boundary conditions

(b) Displacement in x-axis.

(c) Displacement in y-axis.

(d) Strain energy.

Figure 1: Setup and numerical results for the pressurized thick cylinder example

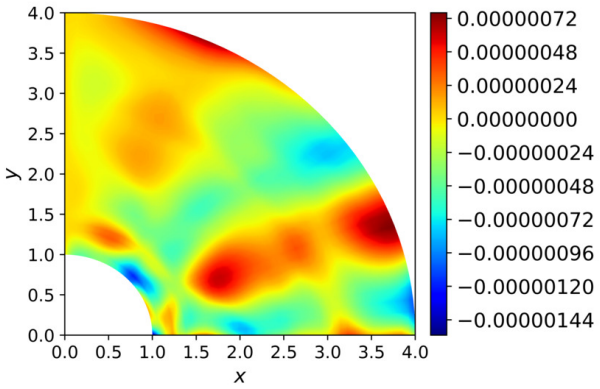
activation function has been considered. The loss function is computed as

$$\begin{aligned}
\mathcal{L}_{elastic} &= \mathcal{L}_{int} - \mathcal{L}_{neu}, \\
\mathcal{L}_{int} &= \frac{A_\Omega}{N_x \times N_y} \sum_{i=1}^{N_x \times N_y} f(\boldsymbol{x}_i), \\
\mathcal{L}_{neu} &= \frac{A_\Omega}{N_{bound}} \sum_{i=1}^{N_{Bound}} f_{neu}(\boldsymbol{x}_i), \\
f(\boldsymbol{x}) &= \frac{1}{2}\boldsymbol{\epsilon} : \mathbb{C} : \boldsymbol{\epsilon}, \\
f_{neu}(\boldsymbol{x}) &= t_{N,x}(\boldsymbol{x}) + t_{N,y}(\boldsymbol{x}),
\end{aligned} \tag{28}$$

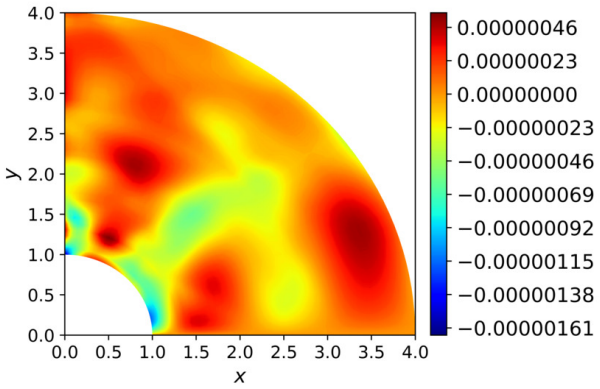
where $\mathcal{L}_{int}$ and $\mathcal{L}_{neu}$ defines the losses in the elastic strain energy and the Neumann boundary loss, respectively and A_Ω is the area of the analyzed domain. The predicted displacement field using the DEM approach is shown in Fig. 1(b)-(c). The strain energy obtained using the neural network is presented in Fig. 1(d). In order to quantify the accuracy of the results obtained using the proposed approach, we compute the relative $\mathcal{L}_2$ error as

$$\mathcal{L}_2^{rel} = \frac{\sqrt{\sum_{i=1}^{N_{pred}} \mathcal{V}_{err}^2(x_i) dx}}{\sqrt{\sum_{i=1}^{N_{pred}} \mathcal{V}_{exact}^2(x_i) dx}}, \tag{29}$$

where $\mathcal{V}_{exact}$ corresponds to the results obtains using the analytical solution of the field variable, $\mathcal{V}$ and $\mathcal{V}_{err} = \mathcal{V}_{exact} - \mathcal{V}_{pred}$, where $\mathcal{V}_{pred}$ is obtained from the neural network.



(a) Error in the displacement field in x-axis.



(b) Error in the displacement field in y-axis.

Figure 2: Error plots for pressurized thick-cylinder.

Corresponding to the displacement and the energy norms, relative prediction errors of 0.5% and 3.7%, respectively have been observed. Fig. 2 shows the errors in the displacement field.

6.2.2. Plate with a circular hole

The second problem is a benchmark problem of an infinite plate with a hole, in plane stress. To solve the problem numerically, we consider a finite domain. Exploiting the symmetry of the problem, only one quarter of the plate is analyzed. The geometrical setup and the boundary conditions for the analyzed domain are depicted in Fig. 3. The material properties considered are $E = 1 \times 10^5$ and $\nu = 0.3$. A traction force, $T_x = 10$ is applied on the plate as shown in Fig. 3. The analytical solution of the problem, in terms of polar coordinates is given by [26]:

$$\begin{aligned}
\sigma_{rr} &= \frac{T_x}{2}\left(1 - \frac{R^2}{r^2}\right) + \frac{T_x}{2}\left(1 + 3\frac{R^4}{r^4} - 4\frac{R^2}{r^2}\right)\cos 2\theta, \\
\sigma_{\theta\theta} &= \frac{T_x}{2}\left(1 + \frac{R^2}{r^2}\right) - \frac{T_x}{2}\left(1 + 3\frac{R^4}{r^4}\right)\cos 2\theta, \\
\sigma_{r\theta} &= -\frac{T_x}{2}\left(1 + 2\frac{R^2}{r^2} - 3\frac{R^4}{r^4}\right)\sin 2\theta,
\end{aligned} \tag{30}$$

where R denotes the radius of the circular hole and (r, θ) is used to represent the radial distance and the angle for locating any point on the plate. Representing the stresses in the Cartesian co-ordinate system, we obtain

$$\begin{pmatrix} \sigma_{xx}(x,y) \\ \sigma_{yy}(x,y) \\ \sigma_{xy}(x,y) \end{pmatrix} = A^{-1} \begin{pmatrix} \sigma_{rr}(r,\theta) \\ \sigma_{\theta\theta}(r,\theta) \\ \sigma_{r\theta}(r,\theta) \end{pmatrix}, \tag{31}$$

where the transformation matrix A is expressed as

$$A = \begin{pmatrix} \cos^2\theta & \sin^2\theta & 2\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -2\sin\theta\cos\theta \\ -\sin\theta\cos\theta & \sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{pmatrix}. \tag{32}$$

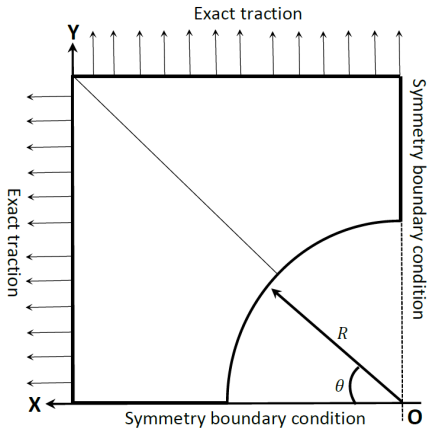


Figure 3: Problem setup for the plate with a circular hole example

The minimization problem reads as stated in Eq. (65). The Dirichlet and Neumann boundary conditions are:

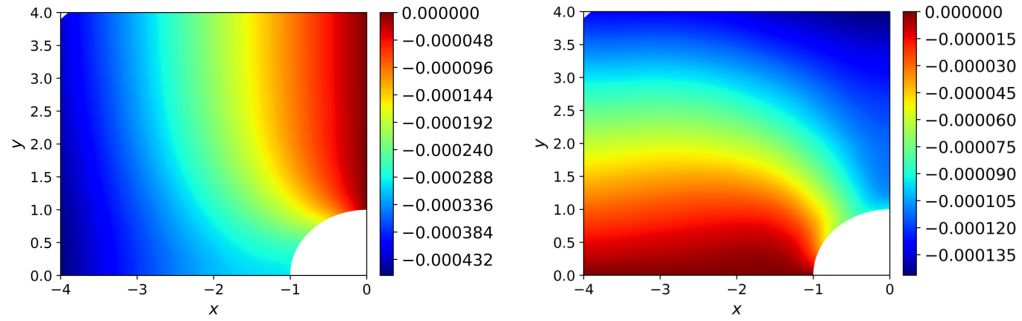
$$\begin{aligned} u(0,y) = v(x,0) = 0, \\ t_{N,x} = T_x, \end{aligned} \tag{33}$$

where u and v are the solutions of the elastic field in x and y-axis, respectively and $t_{N,x}$ denotes the traction force applied on the plate along the x-axis. To find the solution of the displacement field, the trial solution is defined as in Eq. (27). Similar to the previous example, we have used a network with 3 hidden layers of 30 neurons each. For the first two layers, we have considered rectified linear units, $\texttt{ReLU}^2$ activation function; whereas for the last layer, linear activation function has been considered. The domain is discretized into $N_x \times N_y$ uniformly spaced points in the interior of the plate and N_{Bound} uniformly spaced points on the edge for applying the traction force, with $N_{Bound} = 80$. We use Eq. (28) to compute the loss function to optimize the network. The predicted displacement field for the plate with a circular hole, using the DEM approach, is shown in Fig. 4 (a)-(b).

To quantify the accuracy of the neural network, the relative $\mathcal{L}_2$ error corresponding to u and strain energy has been computed using Eq. (29). Corresponding to u and strain energy, a prediction error of 1.8% and 3.19%, respectively have been observed. Fig. 4(c)-(d) shows the errors in the displacement field.

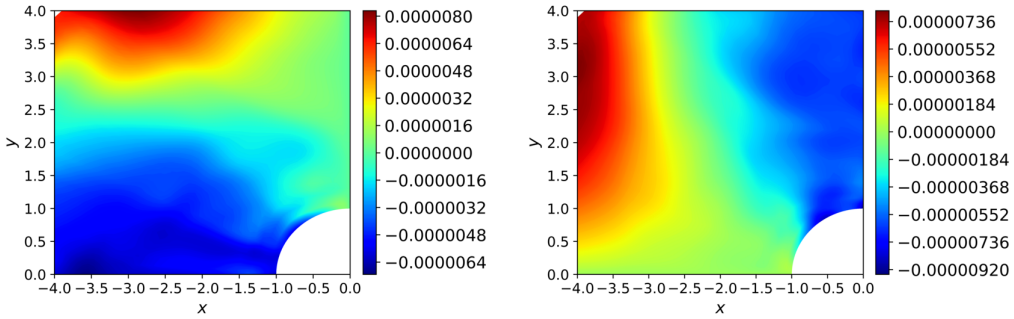
6.2.3. Hollow sphere under internal pressure

In this example, a hollow sphere is subjected to internal pressure is considered. Owing to the symmetrical structure, we analyze only one-eighth of the hollow sphere. The problem domain and its corresponding boundary conditions are shown in Fig. 5(a). The analytical



(a) Displacement in x-axis.

(b) Displacement in y-axis.



(c) Error in the displacement field in x-axis.

(d) Error in the displacement field in y-axis.

Figure 4: Computed solution and errors for plate with hole.

solution of the problem is given by [27]

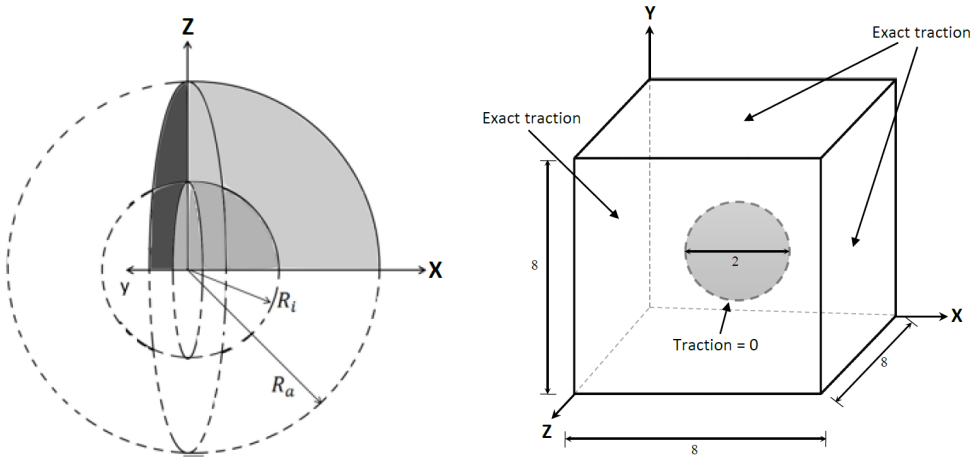
$$
\begin{aligned}
u_r &= \frac{PR_b^3 r}{E(R_a^3 - R_b^3)}\left(1 - 2\nu + \frac{(1+\nu)R_a^3}{2r^3}\right), \\
\sigma_r &= \frac{PR_i^3(R_a^3 - r^3)}{r^3(R_a^3 - R_b^3)}, \\
\sigma_\phi = \sigma_\theta &= \frac{PR_i^3(R_a^3 + 2r^3)}{2r^3(R_a^3 - R_b^3)},
\end{aligned} \tag{34}
$$

where r and θ denote the radial distance and the angle with respect to the origin to locate any co-ordinatewithin the domain and ϕ represents the azimuthal angular in the $x-y$ plane from the x-axis. In Eq. (34), R_a and R_b denote the outer and the inner radius of the hollow sphere, respectively, where $R_a = 4$ and $R_b = 1$. For this problem, the material properties considered are $E = 1 \times 10^3$ and $\nu = 0.3$. The hollow sphere is subjected to an internal pressure, $P = 1$. The Dirichlet boundary conditions are:

$$
u(x, y, 0) = v(x, y, 0) = w(x, y, 0) = 0, \tag{35}
$$

where u, v and w are the solutions of the elastic field in x, y, and z-axis, respectively. To ensure that the boundary conditions are exactly satisfied, we have set

$$
\begin{aligned}
u &= x\hat{u}, \\
v &= y\hat{v}, \\
w &= z\hat{w},
\end{aligned} \tag{36}
$$



(a) Hollow sphere under internal pressure. (b) Cube with a spherical hole.

Figure 5: Geometrical Setup and boundary condition for the application of linear elasticity on three-dimensional structure.

where $\hat{u}$, $\hat{v}$ and $\hat{w}$ are obtained from the neural network. To obtain the displacement field, we have used a network with 3 hidden layers of 50 neurons each and $N_x \times N_y \times N_z$ uniformly spaced points in the interior of the domain, $N_x = N_y = 80$ and $N_z = 10$. The boundary face on which the internal pressure, in terms of traction force, is applied is discretized into N_{Bound} points, with $N_{Bound} = 1600$. For the first two layers, we have considered rectified linear units, $\texttt{ReLU}^2$ activation function; whereas for the last layer, linear activation function has been considered. For a three-dimensional problem, the loss function is computed as

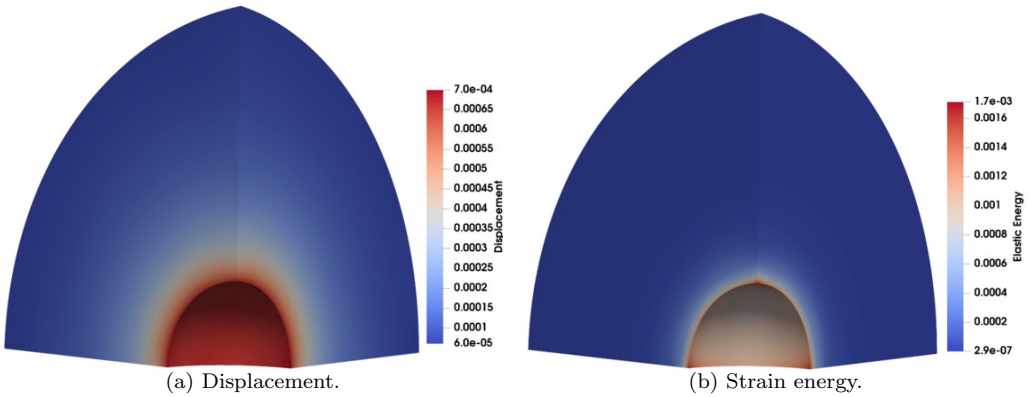
$$\begin{aligned}
\mathcal{L}_{elastic} &= \mathcal{L}_{int} - \mathcal{L}_{neu}, \\
\mathcal{L}_{int} &= \frac{V_\Omega}{N_x \times N_y \times N_z} \sum_{i=1}^{N_x \times N_y \times N_z} f(\boldsymbol{x}_i), \\
\mathcal{L}_{neu} &= \frac{V_{\partial\Omega}}{N_{bound}} \sum_{i=1}^{N_{Bound}} f_{neu}(\boldsymbol{x}_i), \\
f(\boldsymbol{x}) &= \frac{1}{2}\boldsymbol{\epsilon} : \mathbb{C} : \boldsymbol{\epsilon}, \\
f_{neu}(\boldsymbol{x}) &= t_{N,x}(\boldsymbol{x}) + t_{N,y}(\boldsymbol{x}) + t_{N,z}(\boldsymbol{x}),
\end{aligned} \tag{37}$$

where $\mathcal{L}_{int}$ and $\mathcal{L}_{neu}$ defines the losses in the elastic strain energy and the Neumann boundary loss, respectively and V_Ω is the volume of the analyzed domain. The predicted displacement field and the strain energy, using the DEM approach is shown in Fig. 6.

Similar to the previous examples, the $\mathcal{L}_2$ error is computed to check the accuracy of the solution. Corresponding to u and strain energy, a prediction error of 1.05% and 4.44%, respectively have been observed.

6.2.4. Cube with a spherical hole subject to uniform tension

As the last example of the linear elasticity section, we consider a spherical hole in a cube subjected to uniform tensile loading. Owing to symmetry, only one-eight of the spherical hole is analyzed as shown in Fig. 5(b). The analytical stresses in spherical coordinates for



(a) Displacement. (b) Strain energy.

Figure 6: Computed solution for the hollow sphere subjected to internal pressure

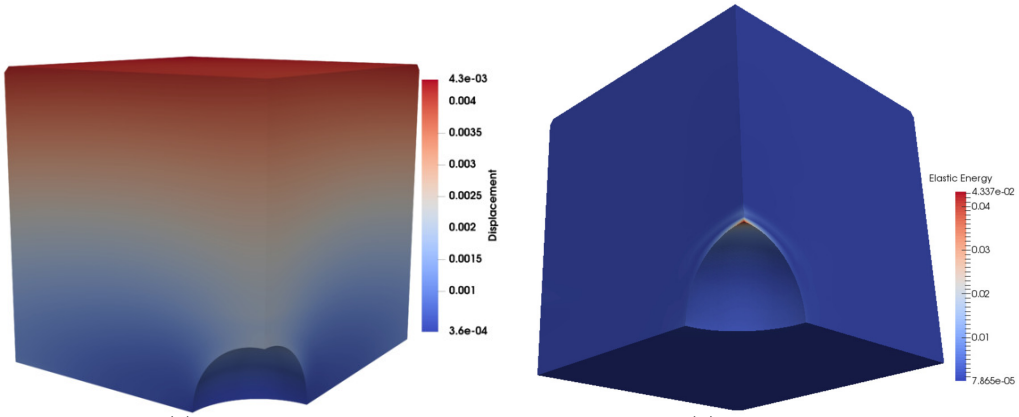
this are [28]:

$$
\begin{aligned}
\sigma_{rr} &= S\cos^2\theta + \frac{S}{7-5\nu}\left(\frac{a^3}{r^3}(6-5\cos^2\theta(5-\nu)) + \frac{6a^5}{r^5}(3\cos^2\theta-1)\right),\\
\sigma_{\phi\phi} &= \frac{3S}{2(7-5\nu)}\left(\frac{a^3}{r^3}(5\nu-2+5\cos^2\theta(1-2\nu)) + \frac{a^5}{r^5}(1-5\cos^2\theta)\right),\\
\sigma_{\theta\theta} &= S\sin^2\theta + \frac{S}{2(7-5\nu)}\left(\frac{a^3}{r^3}\left(4-5\nu+5\cos^2\theta\,(1-2\nu)\right) + \frac{3a^5}{r^5}(3-7\cos^2\theta)\right),\\
\sigma_{r\theta} &= S\left(-1+\frac{1}{7-5\nu}\left(\frac{12a^5}{r^5} - \frac{5a^3(1+\nu)}{r^3}\right)\right),
\end{aligned}
\tag{38}
$$

where S denotes the applied uniaxial tension and a is the radius of the spherical hole. The material properties considered for this example are $E = 1\times10^3$ and $\nu = 0.3$. The Dirichlet boundary conditions are:

$$u(x,y,0) = v(x,y,0) = w(x,y,0) = 0, \tag{39}$$

where u, v and w are the solutions of the elastic field in x, y, and z-axis, respectively. To ensure that the boundary conditions are exactly satisfied, we use the same trial function for the displacement field as stated in Eq. (36). For obtaining the displacement, we have used a fully connected neural network with 3 hidden layers of 50 neurons each. Similar to previous example, for the first two layers $\texttt{ReLU}^2$ activation function and for the last layer linear activation function have been used. The domain is discretized into N_{Int} points in the interior of the domain, with $N_{Int} = 32000$. The boundary face on which the internal pressure in terms of traction force is applied is discretized into N_{Bound} points, with $N_{Bound} = 1600$.



(a) Displacement (b) Strain energy

Figure 7: Computed solution for the cube with a hole example

For a three-dimensional problem, the loss function is computed as

$$\begin{aligned}
\mathcal{L}_{elastic} &= \mathcal{L}_{int} - \mathcal{L}_{neu}, \\
\mathcal{L}_{int} &= \frac{V_\Omega}{N_{Int}} \sum_{i=1}^{N_{Int}} f(\boldsymbol{x}_i), \\
\mathcal{L}_{neu} &= \frac{V_{\partial\Omega}}{N_{bound}} \sum_{i=1}^{N_{Bound}} f_{neu}(\boldsymbol{x}_i), \\
f(\boldsymbol{x}) &= \frac{1}{2}\boldsymbol{\epsilon} : \mathbb{C} : \boldsymbol{\epsilon}, \\
f_{neu}(\boldsymbol{x}) &= t_{N,x}(\boldsymbol{x}) + t_{N,y}(\boldsymbol{x}) + t_{N,z}(\boldsymbol{x}),
\end{aligned} \tag{40}$$

where $\mathcal{L}_{int}$ and $\mathcal{L}_{neu}$ defines the losses in the elastic strain energy and the Neumann boundary loss, respectively and V_Ω is the volume of the analyzed domain. The predicted displacement field and the strain energy are shown in Fig. 7.

Similar to the previous examples, the $\mathcal{L}_2$ error is computed for the strain energy to check thee accuracy of the solution, and a prediction error of 5.3% is observed. In the next section, we would analyze the PDEs involved in solving problems of hyperelasticity.

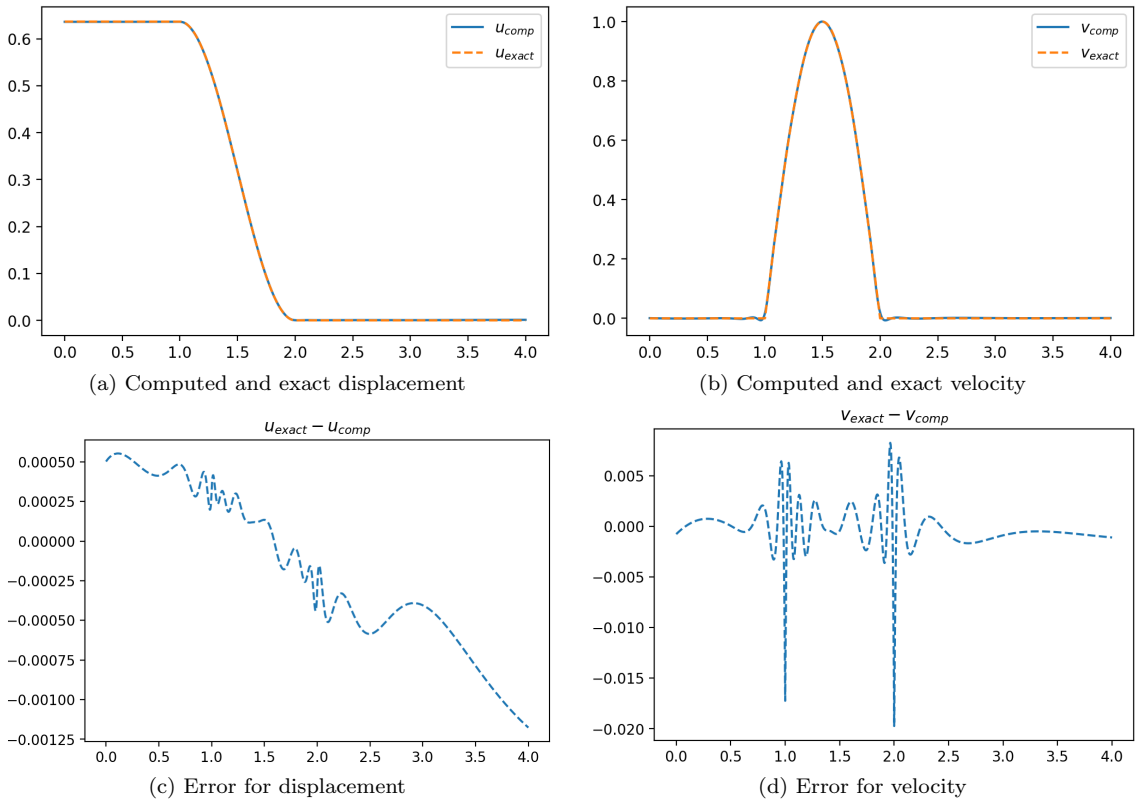
6.3. Elastodynamics

To investigate the approximation properties of neural networks for time-dependent problems, we consider a wave propagation example. In one dimension, the governing (equilibrium) equation is of the form:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \text{ for } x \in \Omega := (a, b) \text{ and } t \in (0, T) \tag{41}$$

together with the initial conditions

$$u(x, 0) = u_0(x) \text{ and } u_t(x) = v_0(x) \text{ for } x \in \Omega \tag{42}$$



(a) Computed and exact displacement

(b) Computed and exact velocity

(c) Error for displacement

(d) Error for velocity

Figure 8: Comparison between the computed and exact solutions for the 1D wave propagation example

and boundary conditions

$$u(x,t) = \bar{u}(t) \text{ for } x \in \Gamma_D \text{ and} \tag{43}$$

$$u_x(x,t) = g(t) \text{ for } x \in \Gamma_N. \tag{44}$$

Here $u(x,t) : \Omega \times [0,T] \to \mathbb{R}$ is the displacement at point x and time t, c is a real positive constant representing the wave propagation speed, u_0 is the initial displacement, v_0 is the initial velocity, and $\bar{u}(t)$ and $g(t)$ are the prescribed data for the Dirichlet and Neumann boundaries Γ_D and respectively Γ_N.

In this example, we assume that $c = 1$ and the left side of the domain is subject to a sinusoidal load while the right side is fixed:

$$u_x(0,t) = \begin{cases} -\sin(\pi t), & \text{for } 0 \le t \le 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{and } u(L,t) = 0, \tag{45}$$

where $L = 4$. The initial displacement and velocity are set to zero:

$$u_0(x) = v_0(x) = 0. \tag{46}$$

We compute the displacement up to time $T = 2$ with a space-time discretization neural network with 3 hidden layers of 50 neurons each. The input layer has 2 neurons (for the space and time coordinates), while the output layer has 1 neuron for outputting the displacement. The loss function is defined based on a collocation approach which incorporates terms corresponding to the equilibrium equation together with the initial and boundary conditions

as:

$$\mathcal{L}(\boldsymbol{p}) = \mathcal{L}_{resid}(\boldsymbol{p}) + \mathcal{L}_{init}(\boldsymbol{p}) + \mathcal{L}_{bnd}(\boldsymbol{p}), \tag{47}$$

where

$$\mathcal{L}_{resid}(\boldsymbol{p}) = \frac{1}{N_{int}} \sum_{i=1}^{N_{int}} (\hat{u}_{tt}(x_i^{int}, t_i^{int}; \boldsymbol{p}) - c^2 \hat{u}_{xx}(x_i^{int}, t_i^{int}; \boldsymbol{p}))^2 \tag{48}$$

$$\mathcal{L}_{init}(\boldsymbol{p}) = \frac{1}{N_{init}} \sum_{j=1}^{N_{init}} \left[(\hat{u}(x_j^{init}, 0; \boldsymbol{p}) - u_0(x_j^{init}))^2 + (\hat{u}_t(x_j^{init}, 0; \boldsymbol{p}) - v_0(x_j^{init}))^2 \right] \tag{49}$$

$$\mathcal{L}_{bnd}(\boldsymbol{p}) = \frac{1}{N_{bnd}} \sum_{k=1}^{N_{bnd}} \left[(\hat{u}_x(0, t_k^{bnd}; \boldsymbol{p}) - \bar{u}(t_k^{bnd}))^2 + (\hat{u}(L, t_k^{bnd}; \boldsymbol{p}) - g(t_k^{bnd}))^2 \right]. \tag{50}$$

Here $\hat{u}(x, t; \boldsymbol{p})$ is the neural network approximation to $u(x, t)$ which is determined by the parameters $\boldsymbol{p}$ obtained by minimizing the loss function. Moreover, (x_i^{int}, t_i^{int}) for $i = 1, \ldots, N_{int}$ are interior collocation points, x_j^{init} with $j = 1, \ldots, N_{init}$ are the points in the space domain and t_k^{bnd}, $k = 1, \ldots, N_{bnd}$ are points in time corresponding to the initial and boundary conditions respectively.

The approximation obtained with $N_{int} = 199^2$,and $N_{init} = 199$, and $N_{bnd} = 200$ collocation points as well as the errors in the computed displacement and velocity are shown in Figure 8. The relative errors in L^2 norm for the displacement and velocity are $1.422382 \cdot 10^{-3}$ and $5.954601 \cdot 10^{-3}$ respectively.

6.4. Hyperelasticity

In the context of the elastostatics at finite deformation (shown in Figure 9), the equilibrium equation along with the boundary conditions, for an initial configuration, is given as:

$$\text{Equilibrium:} \quad \nabla \cdot \boldsymbol{P} + \boldsymbol{f}_b = \boldsymbol{0}, \tag{51}$$

$$\text{Dirichlet boundary} : \boldsymbol{u} = \overline{\boldsymbol{u}} \quad \text{on } \partial\Omega_D, \tag{52}$$

$$\text{Neumann boundary} : \boldsymbol{P} \cdot n = \bar{\boldsymbol{t}} \quad \text{on } \partial\Omega_N, \tag{53}$$

in which $\overline{\boldsymbol{u}}$ and $\bar{\boldsymbol{t}}$ are prescribed values on Dirichlet boundary and Neumann boundary, respectively. Therein, the boundaries have to fulfill $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$, $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. The outward normal unit vector is denoted by n. The 1^{st} Piola-Kirchhoff stress tensor $\boldsymbol{P}$ is related to its power conjugate $\boldsymbol{F}$, so-called deformation gradient tensor, by a constitutive equation $\boldsymbol{P} = \partial\Psi/\partial\boldsymbol{F}$. The deformation gradient is defined as follows

$$\boldsymbol{F} = \text{Grad } \boldsymbol{\varphi}(\boldsymbol{X}), \tag{54}$$

where $\boldsymbol{\varphi}$ denotes the mapping of material points in the initial configuration to the current configuration. It is defined as:

$$\boldsymbol{\varphi}(\boldsymbol{X}) := \boldsymbol{x} = \boldsymbol{X} + \boldsymbol{u}.$$

Since the strain energy density Ψ describing the elastic energy stored in the body is explicitly known, it is more convenient to find the possible deformations in a form of potential energy

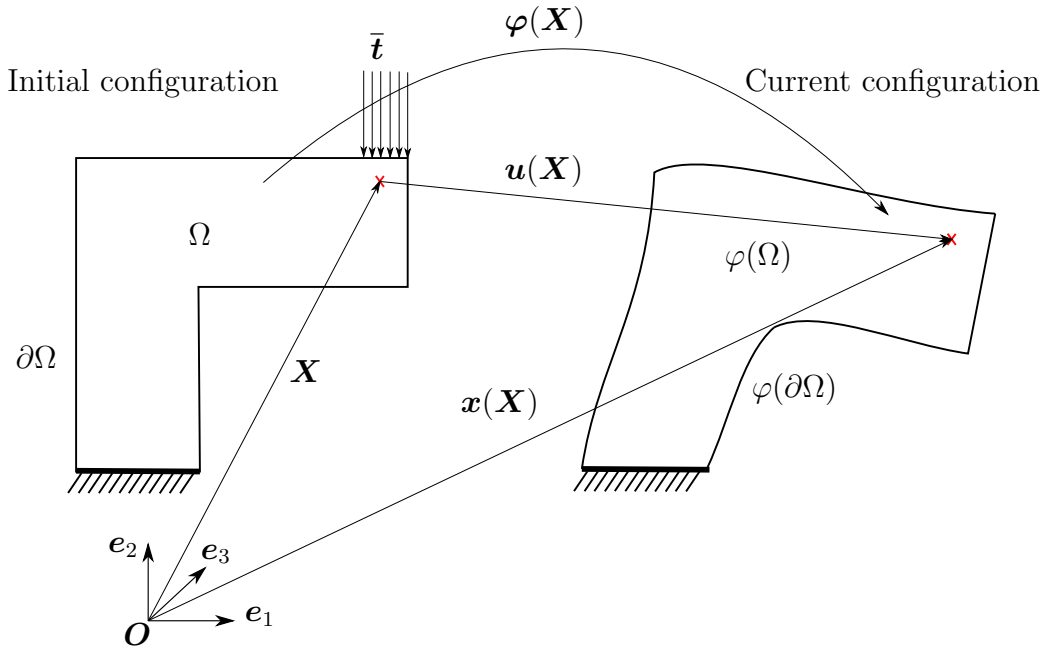


Figure 9: Motion of body B.

rather than the strong formulation. The potential functional is written as

$$\mathcal{E}(\boldsymbol{\varphi}) = \int_\Omega \Psi \, dV - \int_\Omega \boldsymbol{f}_b \cdot \boldsymbol{\varphi} \, dV - \int_{\partial\Omega_N} \bar{\boldsymbol{t}} \cdot \boldsymbol{\varphi} \, dA. \tag{55}$$

In order to obtain the solution we minimize the potential energy

$$\min_{\boldsymbol{\varphi} \in H} \; \mathcal{E}(\boldsymbol{\varphi}), \tag{56}$$

where H is the set of admissible functions (trial functions). The method is named *the principle of minimum total potential energy.*

Now that our objective is to minimize the potential energy with the aid of neural networks, we need to cast the minimization into the optimization problem in the context of machine learning. Therefore, the definition of a loss function is essential. In this case we exploit the potential energy as the loss function. It reads

$$\mathcal{L}(\boldsymbol{p}) = \int_\Omega \Psi(\boldsymbol{\varphi}(\boldsymbol{X};\boldsymbol{p})) \, dV - \int_\Omega \boldsymbol{f}_b \cdot \boldsymbol{\varphi}(\boldsymbol{X};\boldsymbol{p}) \, dV - \int_{\partial\Omega_N} \bar{\boldsymbol{t}} \cdot \boldsymbol{\varphi}(\boldsymbol{X};\boldsymbol{p}) \, dA, \tag{57}$$

where the trial function now reads

$$\boldsymbol{\varphi}_p(\boldsymbol{X}) = \boldsymbol{u}_p(\boldsymbol{X}) + \boldsymbol{X}, \tag{58}$$

and $\boldsymbol{u}_p(\boldsymbol{X})$ has to fulfill boundary conditions prior to being applied any operators. We use a short notation $f_p(\boldsymbol{X})$ instead of $f(\boldsymbol{X};\boldsymbol{p})$. Here, a feedforward neural network constructed by hyperparameters $\boldsymbol{p}$ correspond to the weights and biases of the neural architecture is employed to establish the trial solution. In this context the loss function now is written by

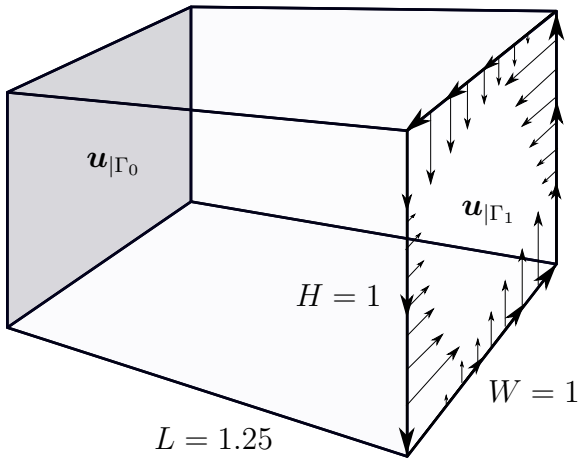


Figure 10: A hyperelastic 3D Cuboid is twisted an angle of 60^o.

means of an approximation as follows

$$\mathcal{L}(\boldsymbol{p}) \approx \frac{V_\Omega}{N_\Omega} \sum_{i=1}^{N_\Omega} \Psi((\boldsymbol{\varphi}_p)_i) - \frac{V_\Omega}{N_\Omega} \sum_{i=1}^{N_\Omega} (\mathbf{f}_b)_i \cdot (\boldsymbol{\varphi}_p)_i - \frac{A_{\partial\Omega_N}}{N_{\partial\Omega_N}} \sum_{i=1}^{N_{\partial\Omega_N}} \bar{\mathbf{t}}_i \cdot (\boldsymbol{\varphi}_p)_i, \tag{59}$$

where N_Ω is the total number of data of the solid; $N_{\partial\Omega_N}$ is the number of data of the surface having force. V_Ω is the volume of the solid and $A_{\partial\Omega_N}$ is the area of the surface. If a function f is evaluated at a material point $\mathbf{X}_i$, we denote this by f_i.

An example about torsion inspired from a Fenics application [29] is chosen to examine in order to verify the robustness of DEM in hyperelasticity problems. Let us consider a 3D Cuboid made of an isotropic, homogeneous, hyperelastic material with length $L = 1.25$, width $W = 1.0$ and depth $L = 1.0$ (we drop out all units for the sake of simplicity). The solid is clamped at the left surface and is twisted an angle of 60^o counterclockwise by prescribed displacement boundary conditions $\boldsymbol{u}_{|\Gamma_1}$ applied at the right-end surface. In addition, the Cuboid is subjected to a body force $\boldsymbol{f}_b = [0, -0.5, 0]^T$ and traction forces $\bar{\boldsymbol{t}} = [1, 0, 0]^T$ at the bottom, back, top and front surfaces (Figure 10). The Dirichlet boundary conditions are prescribed as follows

$$\begin{aligned} \boldsymbol{u}_{|\Gamma_0} &= [0, 0, 0]^T, \\ \boldsymbol{u}_{|\Gamma_1} &= \begin{bmatrix} 0 \\ 0.5\,[0.5 + (X_2 - 0.5)\cos(\pi/3) - (X_3 - 0.5)\sin(\pi/3) - X_2] \\ 0.5\,[0.5 + (X_2 - 0.5)\sin(\pi/3) + (X_3 - 0.5)\cos(\pi/3) - X_3] \end{bmatrix}. \end{aligned} \tag{60}$$

In this problem, the Neo-Hookean model is considered and it has the form

$$\Psi(I_1, J) = \frac{1}{2}\lambda[\log(J)]^2 - \mu \log(J) + \frac{1}{2}\mu(I_1 - 3), \tag{61}$$

where two invariants are given by

$$I_1 = \text{trace}(\boldsymbol{C}), \qquad J = \det(\boldsymbol{F}). \tag{62}$$

The right Cauchy-Green tensor is expressed as $\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F}$. In this example, material properties are prescribed in Table 1,

Description	Value
E - Young modulus	10^6
ν - Poisson ratio	0.3
μ - Lame' parameter	$\frac{E}{2(1+\nu)}$
λ - Lame' parameter	$\frac{E\nu}{(1+\nu)(1-2\nu)}$

Table 1: Material parameters for 3D hyperelastic Cuboid.

With the prescribed boundary conditions, the displacement has the form

$$\boldsymbol{u}_p(\boldsymbol{X}) = \mathbf{u}_{|\Gamma_1} \frac{X_1}{1.25} + (X_1 - 1.25)\, X_1\, \widehat{\boldsymbol{u}}(\mathbf{X}; \boldsymbol{p}). \tag{63}$$

Setup. We generate 64000 equidistant grid points ($N_1 = 40, N_2 = 40, N_3 = 40$) over the entire domain as the feeding data to train our network (Figure 11). A 5-layer network $(3 - 30 - 30 - 30 - 3)$ is constructed. Therein, we use 3 neurons in the input layer for the nodal coordinates and 3 neurons in the output layer for the unconstrained solution. Moreover, in each hidden layer 30 neurons are used to enforce the learning behavior. We use *tanh* activation function to evaluate neural values in the hidden layers. The learning rate $r = 0.5$ is used in the L-BFGS optimizer. The neural network is strained for 50 steps.

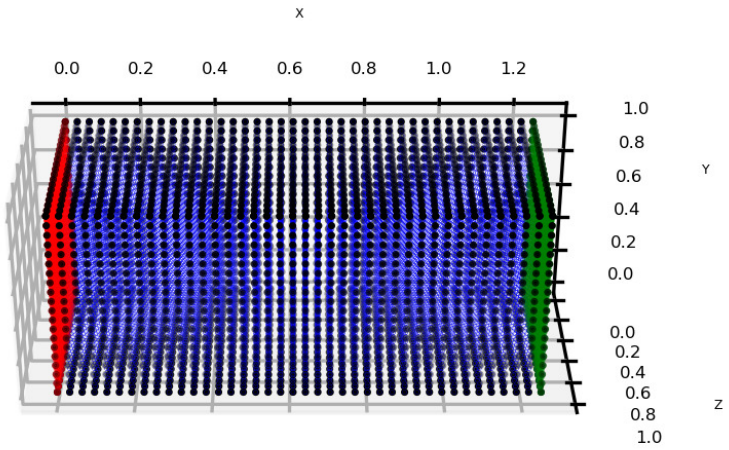
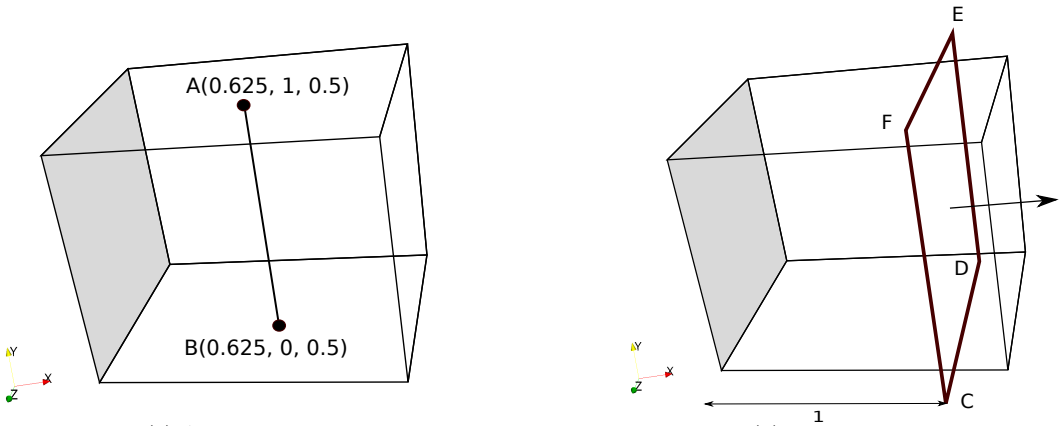


Figure 11: The training point distribution of a 3D hyperelastic cuboid in DEM. The red points correspond to the first Dirichlet boundary condition. The points data are used to enforce/train the second Dirichlet boundary condition. The black points are used to train the surface force. The blue points are used for the interior domain integration and the body force.

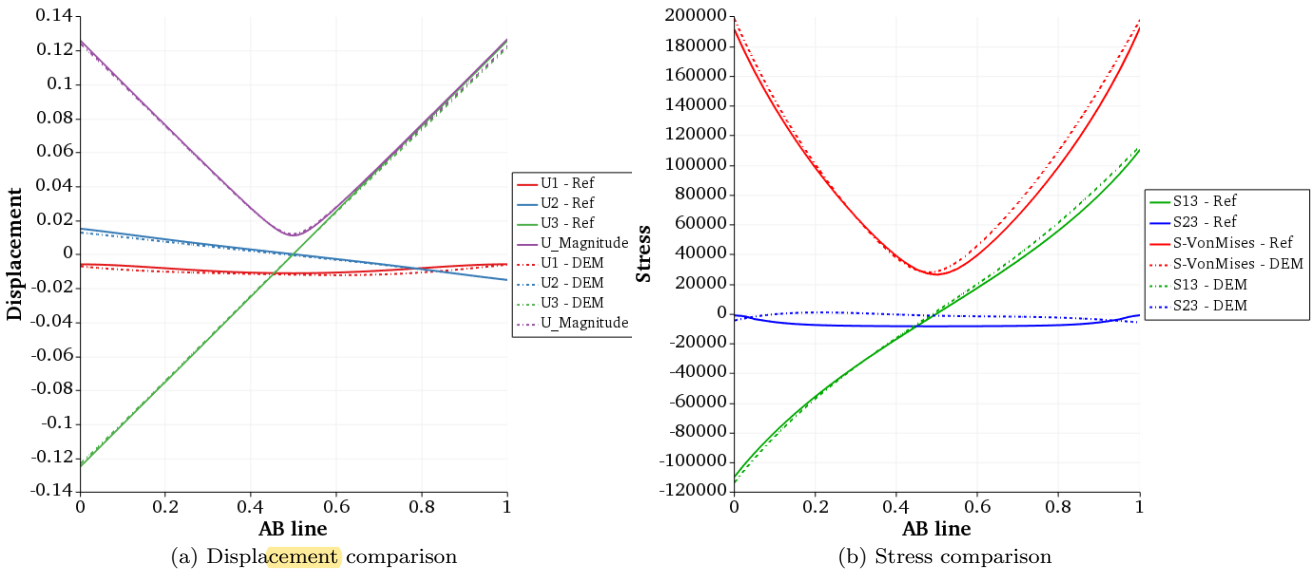
Result. Since the analytical solutions are not available in this example, we use a finite element program to simulate the Cuboid with a fine mesh to obtain the reference solution. We

first measure the displacements and stresses along the AB line depicted in the Figure 12a. The result in general reveals a good agreement between DEM solution and the reference solution in Figure 13. It is found that the deep energy method exhibits good solutions in terms of displacements. However, there has small difference in terms of stress because the outputs of our neural network are the displacements and the gradient fields are then given in the post-processing procedure based on the displacement field.



(a) AB line (b) CDEF plane

Figure 12: Positions of AB line and CDEF plane.

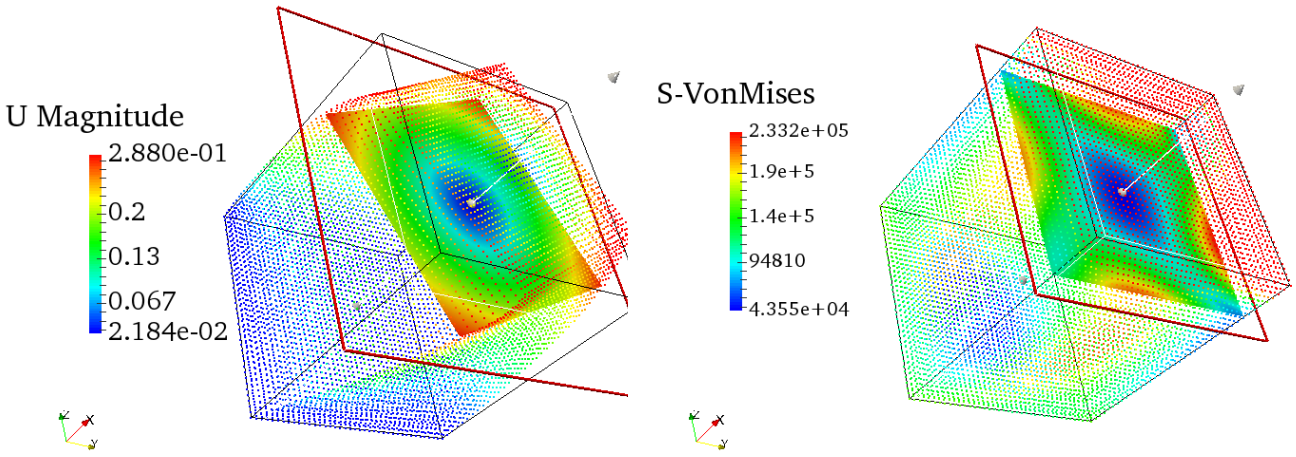


(a) Displacement comparison (b) Stress comparison

Figure 13: The displacement and stress results of DEM measured at the AB line compared to the reference solution.

We observe the surface displacement and surface stress at the CDEF plane as sketched in Figure 12b. Figure 14 shows the predicted displacement in terms of magnitude and the

predicted stress in terms of VonMises. Then we compare the norms of the DEM solution and reference solution in terms of L^2 norm and H^1 seminorm in two setups. Setup 1: We use 8000 equidistant data points ($N_1 = 20, N_2 = 20, N_3 = 20$) as the feeding data to train our network; the network is trained for 50 steps. Setup 2: We use 64000 equidistant data points ($N_1 = 40, N_2 = 40, N_3 = 40$) as the feeding data to train our network; the neural network is trained for 25 steps. The L^2 norm and H^1 seminorm of the reference solution are given by a Fenics program with a fine mesh as: $||u||^{FE}_{L^2} = 0.13275$ and $||u||^{FE}_{H^1} = 0.51407$. As shown in Table 2, the DEM obtain good results in terms of L^2 norm and H^1 seminorm.



(a) Displacement

(b) VonMises stress

Figure 14: Displacement and VonMises stress at the CDEF plane in DEM of a twisted Neo-Hookean 3D Cuboid.

Data	**DEMM**	
	$\|\|u\|\|_{L^2}$	$\|\|u\|\|_{H^1}$
20x20x20 (50 steps)	0.12921	0.49929
40x40x40 (25 steps)	0.13210	0.51001

Table 2: The L^2 norm and H^1 seminorm of DEM in 2 setups. The corresponding norms of the reference solution are given by Fenics program with a fine mesh as: $||u||^{FE}_{L^2} = 0.13275$, $||u||^{FE}_{H^1} = 0.51407$.

For the next three sections, we apply DEM on problems involving more than one field. We start with the modeling of fracture using phase field approach.

6.5. Phase field modeling of fracture

The prevention of fracture-induced failure is a major constraint in engineering designs. The phase field model for fracture is an effective way to model fracture by assuming the process zone has a finite width which is controlled by a length scale parameter (l_0). A sharp crack topology is recovered in the limit as $l_0 \to 0$ [30]. In this approach, the effects associated with crack formation such as stress release are incorporated into the constitutive

model. A continuous scalar parameter (ϕ) is used to track the fracture pattern. The cracked region is represented by $\phi = 1$ while the undamaged portion is given by $\phi = 0$. The phase field approach aims to simultaneously solve for the displacement field and fracture region by minimizing the total potential energy of system, as postulated by the Griffith theory for brittle fracture [31].

In this section, physics informed neural networks are developed to solve the governing partial differential equations for fracture analysis using DEM. The neural networks are trained to approximate the displacement field and the damage fields which satisfy the governing equations used to describe the physical phenomena. Modeling fracture using the phase field method involves the solving for the vector-valued elastic field, $\boldsymbol{u}$ and the scalar-valued phase field, ϕ. In DEM, the solution is obtained by minimization of the total energy of the system, $\mathcal{E}$ [32]. The problem statement can be written as:

$$\begin{aligned} \text{Minimize:} \quad & \mathcal{E} = \mathcal{E}_e + \mathcal{E}_c, \\ \text{subject to:} \quad & \boldsymbol{u} = \overline{\boldsymbol{u}} \text{ on } \partial\Omega_D, \end{aligned} \tag{64}$$

where $\mathcal{E}_e$ is the stored elastic strain energy, $\mathcal{E}_c$ is the fracture energy and $\overline{\boldsymbol{u}}$ is the prescribed displacement on the Dirichlet boundary, $\partial\Omega_D$. In Eq. (64), $\mathcal{E}_e$ and $\mathcal{E}_c$ are defined as:

$$\begin{aligned} \mathcal{E}_e &= \int_\Omega g(\phi)\Psi_0(\boldsymbol{\epsilon})d\Omega, \\ \mathcal{E}_c &= \frac{G_c}{2l_0}\int_\Omega \left(\phi^2 + l_0^2|\nabla\phi|^2\right)d\Omega + \int_\Omega g(\phi)H(\boldsymbol{x},t)d\Omega, \end{aligned} \tag{65}$$

where $g(\phi)$ represents the monotonically decreasing stress-degradation function, G_c is the critical energy release rate, $\Psi_0(\boldsymbol{\epsilon}, \phi)$ is the initial strain energy density functional expressed in terms of linearised strain tensor, $\boldsymbol{\epsilon(u)}$. A common form of the degradation function, as used in literature, for isotropic solids is [33]:

$$g(\phi) = (1-\phi)^2. \tag{66}$$

$H(\boldsymbol{x}, t)$ contains the maximum positive tensile energy (Ψ_0^+) in the history of deformation of the system and is defined as:

$$H(\boldsymbol{x},t) = \max_{s\in[0,t]} \Psi_0^+(\boldsymbol{\epsilon}(\boldsymbol{x},s)), \tag{67}$$

where $\boldsymbol{x}$ is a point in the domain. The strain-history functional enforces irreversibility condition on the crack [33]. Ψ_0^+ is obtained as:

$$\Psi_0^+(\boldsymbol{\epsilon}) = \frac{\lambda}{2}\langle \text{tr}(\boldsymbol{\epsilon})\rangle_+^2 + \mu \text{tr}(\boldsymbol{\epsilon}^2)_+, \tag{68}$$

where λ and μ are Lamé constants. The tensile strain, $\boldsymbol{\epsilon}_+$ is computed using the spectral decomposition of the strain tensor. The strain-history functional can be used to define initial cracks in the system [34]. The initial strain history function ($H(\boldsymbol{x}, 0)$) could be defined as a function of the closest distance from $\boldsymbol{x}$ to the line (l), which represents the discrete crack [35]. In particular, we set

$$H(\boldsymbol{x},0) = \begin{cases} \frac{BG_c}{2l_0}\left(1 - \frac{2d(\boldsymbol{x},l)}{l_0}\right) & d(\boldsymbol{x},l) \leqslant \frac{l_0}{2} \\ 0 & d(\boldsymbol{x},l) > \frac{l_0}{2} \end{cases}, \tag{69}$$

where B is a scalar parameter that controls the magnitude of the scalar history field and is calculated as:

$$B = \frac{1}{1-\phi} \quad \text{for} \quad \phi < 1. \tag{70}$$

We shall now present the results obtained using the physics informed neural networks to study phase field modeling of fracture.

6.5.1. One-dimensional phase field model

The first example is to approximate the phase field in a one-dimensional bar of length, $L = 50$ units and has a crack located at 25 units. An analytical solution for a one-dimensional phase field model is available in [33]. Hence, it is possible to validate the results obtained from the proposed approach. The analytical solution for $\phi(x)$ is:

$$\phi_{ex}(x) = \exp\left(\frac{-|x-a|}{l_0}\right), \tag{71}$$

where the crack is located at a units. We consider $l_0 = 1$ in this example, which is solved using DCM and DEM to compare the accuracy of both methods with the analytical solution.

The formulation of the collocation method is to minimize the function f, defined by

$$f(x) = \phi''(x) - \frac{1}{l_0^2}\phi(x) \text{ in } \Omega, \tag{72}$$

at a set of chosen collocation points, x subject to the Dirichlet-type boundary conditions:

$$\begin{aligned} \phi(a) = 1 \ , \ \phi'(a) = 0 \ , \\ \lim_{x\to\infty} \phi(x) = \lim_{x\to-\infty} \phi(x) = 0, \text{ and} \\ \lim_{x\to\infty} \phi'(x) = \lim_{x\to-\infty} \phi'(x) = 0. \end{aligned} \tag{73}$$

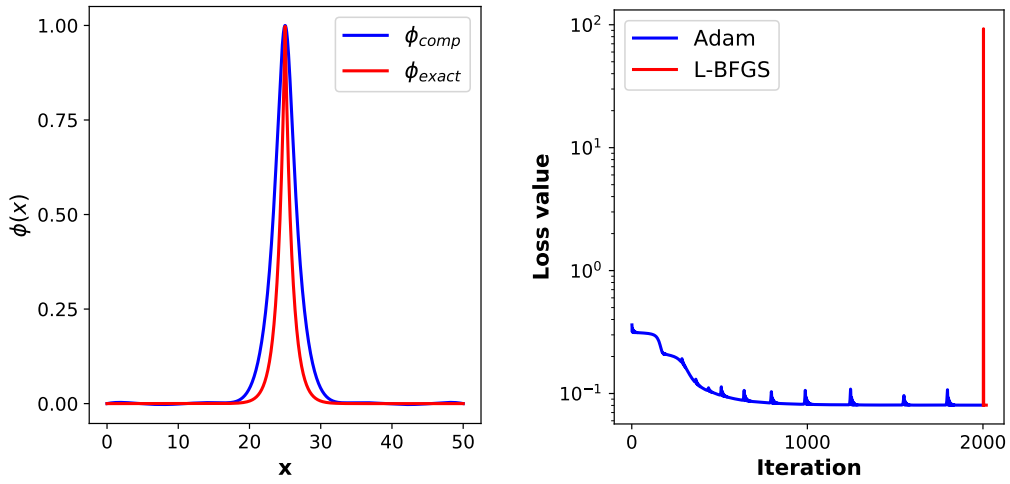
We have used a network with 3 hidden layers of 50 neurons each and $N_{Col} = 8000$ uniformly spaced points within the domain. To find the solution of ϕ in the domain $[0, 50]$, the trial solution is defined as

$$\phi(x) = -\frac{x(x-50)}{625}\left[(x-25)\hat{\phi} + 1\right], \tag{74}$$

where $\hat{\phi}$ is given by the neural network. The solution of ϕ is chosen in such a way that it satisfies all the boundary conditions as in [24, 25]. This ensures that the boundary conditions are satisfied during the training of the network. In the collocation method, the mean squared error of the function, f at the collocation points is minimized. The loss function, $\mathcal{L}_{Col}$ for the collocation method is defined as:

$$\mathcal{L}_{Col} = \frac{1}{N_{Col}} \sum_{i=1}^{N_{Col}} |f(x_i)|^2. \tag{75}$$

For optimization, we use the Adam (adaptive momentum) optimizer followed by a quasi-Newton method (L-BFGS). The boundary conditions are satisfied exactly as could be seen in Fig. 15(a), where ϕ_{comp} represents the value of ϕ obtained using the neural network and ϕ_{exact} is obtained using Eq. (71). Fig. 15(b) shows the convergence of the loss function first with the Adam optimizer and later using the L-BFGS optimizer. To measure the accuracy



(a) Comparison of ϕ_{exact} and ϕ_{comp}.

(b) Convergence of the loss function.

Figure 15: One-dimensional phase field model using the collocation method.

of the method, the relative $\mathcal{L}_2$ error, $\mathcal{L}_2^{rel}$ is computed using using the formula:

$$\mathcal{L}_2^{rel} = \frac{\sqrt{\sum_{i=1}^{N_{pred}} (\phi_{comp} - \phi_{ex})^2 dx}}{\sqrt{\sum_{i=1}^{N_{pred}} \phi_{ex}^2 dx}}. \tag{76}$$

For the collocation method, $\mathcal{L}_2^{rel} = 70.6\%$ due to the inability of the collocation to capture the discontinuity in the derivative at $x = 25$.

Using the same network architecture and the same number of integration points, $N_{Int} = 8000$ in the interior domain, we solve the one-dimensional problem using DEM. For DEM, the problem is written as:

$$\text{Minimize:} \quad I(\phi) = \frac{1}{2} \int_{\Omega} \left(\phi(x) - l_0^2 |\nabla \phi|^2 \right) dx, \tag{77}$$

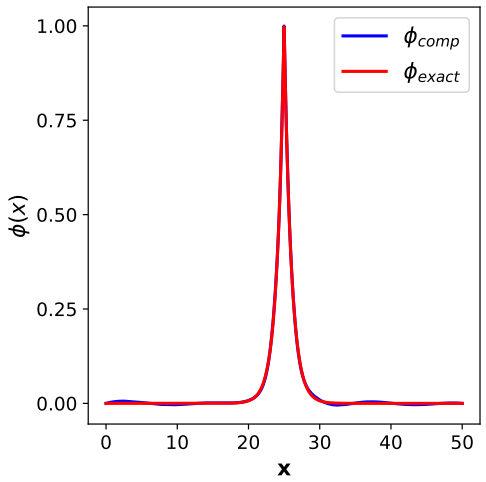
subject to the same boundary conditions as in Eq. (73). The form of the solution used for DEM is as stated in Eq. (74). The loss function, $\mathcal{L}_{Ener}$ for DEM is defined as

$$\mathcal{L}_{Ener} = \frac{L}{N_{Int}} \sum_{i=1}^{N_{Int}} \left(\phi(x_i) - l_0^2 |\nabla \phi(x_i)|^2 \right). \tag{78}$$

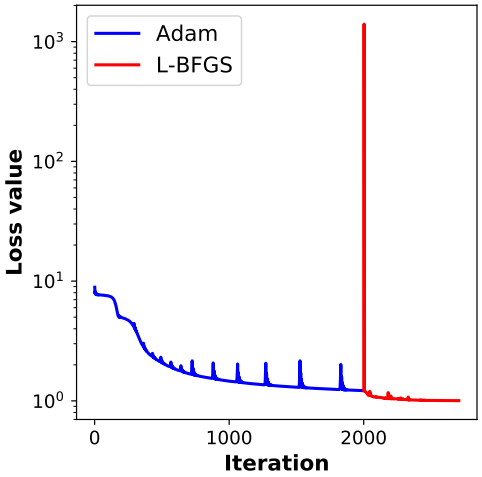
Fig. 16(a) presents the solution of ϕ obtained using DEM, while Fig. 15(b) shows the convergence of the loss function. For DEM, $\mathcal{L}_2^{rel} = 2.88\%$.

6.5.2. Single-edge notched tension example

In this example, we consider a unit square plate with a horizontal crack from the midpoint of the left outer edge to the center of the plate. The geometric setup and boundary conditions of the problem are shown in Fig. 17(a). In this example, we consider $l_0 = 0.0125$ and $G_c = 2.7 \times 10^{-3}$ kN/mm. The material properties of the plate are expressed in terms of Láme's constants, $\lambda = 121.15$ kN/mm^2 and $\mu = 80.77$ kN/mm^2. The domain is subdivided into 3 sections along the y-axis; the crack zone, the bottom and the top of the crack. The

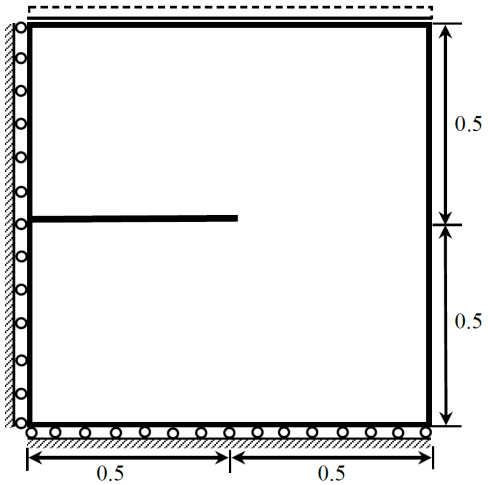


(a) Comparison of ϕ_{exact} and ϕ_{comp}.

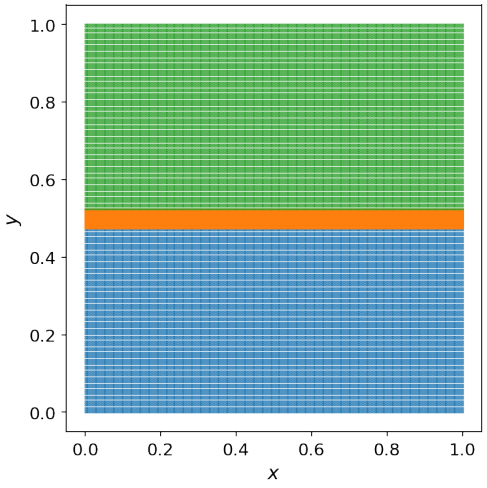


(b) Convergence of the loss function.

Figure 16: One-dimensional phase field model using the energy method.



(a) Geometrical setup and boundary conditions.



(b) Training grid.

Figure 17: Single-edge notch tension example.

crack zone is between $[(0.5-2l_0),\ (0.5+2l_0)]$ on the y-axis. The training grid is shown in Fig. 17(b). The strain history function is used to initialize the crack in the domain as in Eq. (67). We have used a network with 3 hidden layers of 20 neurons each and $N_x \times N_y$ uniformly spaced points for each section in the interior of the plate with $N_x = 300$ and $N_y = 81$. The problem statement is defined as:

$$\begin{aligned} \text{Minimize:} \quad & I(\phi) = \int_{\Omega} f(\phi(\boldsymbol{x}))dx, \\ \text{where} \quad & f(\boldsymbol{x}) = \frac{G_c}{2l_0}\left(\phi(\boldsymbol{x})^2 + l_0^2|\nabla\phi(\boldsymbol{x})|^2 + g(\phi(\boldsymbol{x}))H(\boldsymbol{x},0)\right). \end{aligned} \tag{79}$$

The loss function, $\mathcal{L}_0$ is defined as:

$$\begin{aligned} &A_{tc} = A_{bc} = (0.5-2l_0), \\ &A_c = 4l_0, \\ &\mathcal{L}_0 = \frac{1}{N_x \times N_y}\left(A_{tc}\sum_{i=1}^{N_x\times N_y} f(\boldsymbol{x}_i^{tc}) + A_c\sum_{i=1}^{N_x\times N_y} f(\boldsymbol{x}_i^{c}) + A_{bc}\sum_{i=1}^{N_x\times N_y} f(\boldsymbol{x}_i^{bc})\right), \end{aligned} \tag{80}$$

where A_{tc}, A_{bc}, A_c denote the areas of section on top and bottom of the crack and the crack zone, respectively. $\boldsymbol{x_i^{tc}}$, $\boldsymbol{x_i^{c}}$ and $\boldsymbol{x_i^{bc}}$ in Eq. (80) represent points in the three sub domains in the interior of the plate. To compare the performance of DEM, we also initialize the crack using the collocation method. The formulation of the collocation method is to minimize the function f, defined by

$$\begin{aligned} & f(x) = G_c l_0 \Delta\phi(\boldsymbol{x}) - \frac{G_c}{l_0}\phi(\boldsymbol{x}) - g'(\phi(\boldsymbol{x}))H(\boldsymbol{x},0) \text{ in } \Omega, \\ \text{and} \quad & g'(\phi(\boldsymbol{x})) = -2(1-\phi(\boldsymbol{x})) \end{aligned} \tag{81}$$

subjected to homogeneous Neumann-type boundary conditions on the entire boundary defined as:

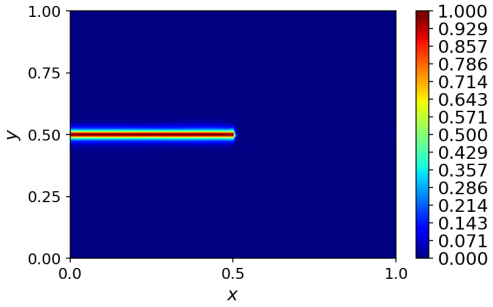
$$\nabla\phi \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega. \tag{82}$$

We have used a network with 5 hidden layers of 20 neurons each and $N_x \times N_y$ uniformly spaced points for each section in the interior of the plate and N_{Bound} uniformly spaced points on each edge of the plate, with $N_x = 300$ $N_y = 81$ and $N_{Bound} = 200$. The loss function, $\mathcal{L}_{Col}$ for the collocation method is defined as:

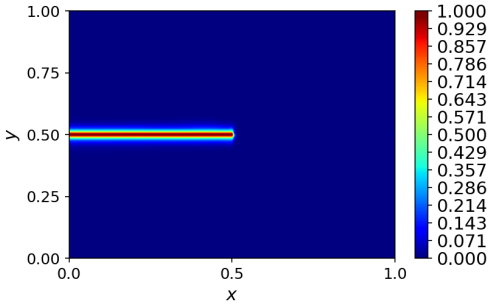
$$\begin{aligned} \mathcal{L}_{Col} = \frac{1}{N_x \times N_y}\left(A_{tc}\sum_{i=1}^{N_x\times N_y} |f(\boldsymbol{x}_i^{tc})|^2 + A_c\sum_{i=1}^{N_x\times N_y} |f(\boldsymbol{x}_i^{c})|^2 + A_{bc}\sum_{i=1}^{N_x\times N_y} |f(\boldsymbol{x}_i^{bc})|^2\right) + \\ \frac{1}{N_{Bound}}\sum_{i=1}^{N_{Bound}}\left(\left[\frac{\partial\phi(\boldsymbol{x}_i^L)}{\partial x}\right]^2 + \left[\frac{\partial\phi(\boldsymbol{x}_i^R)}{\partial x}\right]^2 + \left[\frac{\partial\phi(\boldsymbol{x}_i^T)}{\partial y}\right]^2 + \left[\frac{\partial\phi(\boldsymbol{x}_i^B)}{\partial y}\right]^2\right), \end{aligned} \tag{83}$$

where $\boldsymbol{x}_i^L, \boldsymbol{x}_i^R, \boldsymbol{x}_i^T, \boldsymbol{x}_i^B$ are the points on the left, right, top and bottom edges of the plate, respectively. Fig. 18(a) and (b) show the initial crack in the plate using DEM and DCM, respectively. Fig. 18(c) and (d) present the one-dimensional approximation plots and Fig. 18(e) and (f) show the convergence of the loss function for both the methods.

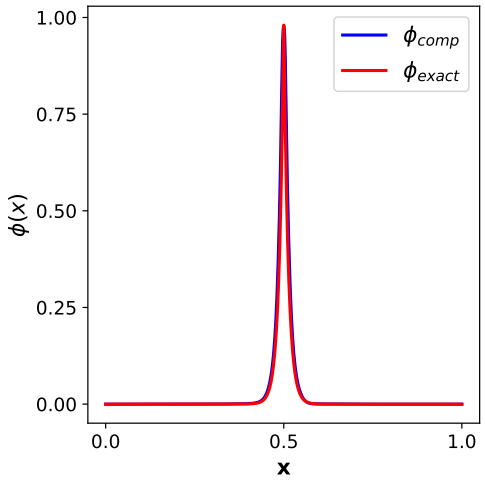
In subsubsection 6.5.1, we observed that DEM performs better than DCM for the same



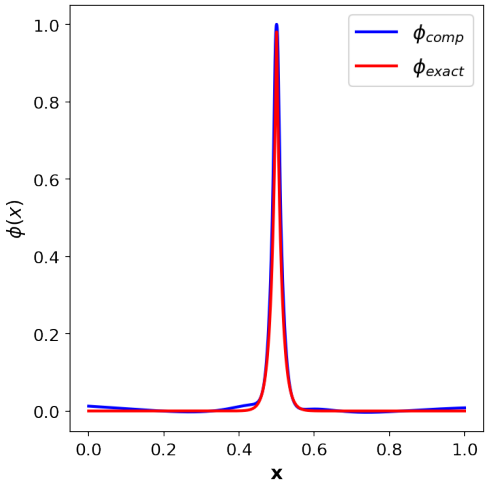
(a) Initial crack using DEM.



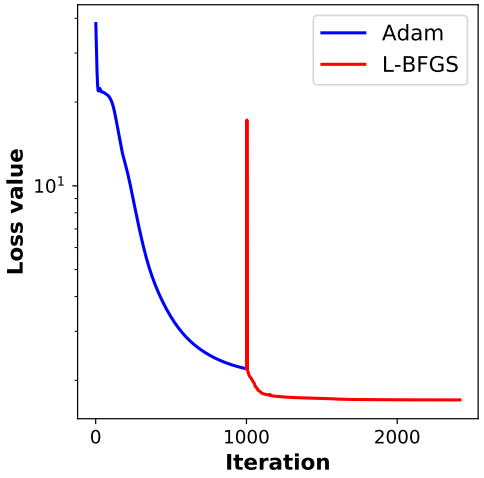
(b) Initial crack using DCM.



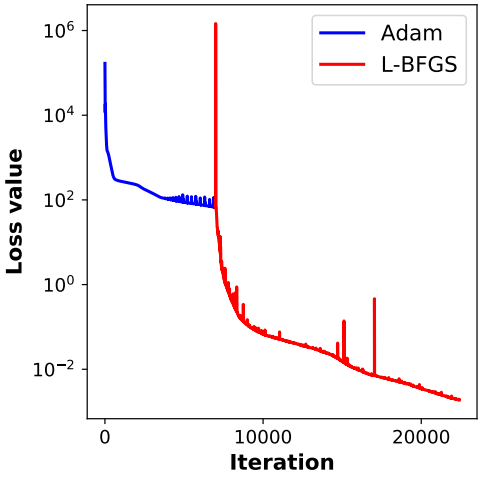
(c) Comparison of ϕ_{exact} and ϕ_{comp} using DEM.



(d) Comparison of ϕ_{exact} and ϕ_{comp} using DCM.



(e) Convergence of the loss function using DEM.



(f) Convergence of the loss function using DCM.

Figure 18: Initialization of crack in the plate using the strain-history function.

network parameters. From the results of the initilization of the crack, we can conclude that DEM requires a smaller network to predict results accurately. Moreover, it is also seen that DCM needs an explicit treatment of the Neumann boundary conditions even for traction-free boundaries, which requires the use of more collocation points. The convergence of the loss function is also slow, requiring 10 times more iterations. Hence, we use DEM to study the growth of crack in a plate under tensile loading.

To observe the propagation of crack in the plate, we simultaneously solve the elastic field and the phase field using a monolithic solver. The minimization problem reads as stated in Eq. (65). The Dirichlet boundary conditions are:

$$u(0,y) = v(x,0) = 0, \quad v(x,1) = \Delta v, \tag{84}$$

where u and v are the solutions of the elastic field in x and y-axis-axis. The computation is performed by applying constant displacement increments of $\Delta v = 1 \times 10^{-3}$ mm. The model for the Dirichlet problem is:

$$\begin{aligned} u &= [x(1-x)]\hat{u}, \\ v &= [y(y-1)]\hat{v} + y\Delta v, \end{aligned} \tag{85}$$

where $\hat{u}$ and $\hat{v}$ are obtained from the neural network. We have used a network with 5 hidden layers of 50 neurons each. The loss function, $\mathcal{L}_{\mathcal{E}}$ is defined as:

$$\begin{aligned} \mathcal{L}_{\mathcal{E}} &= \mathcal{L}_{Elas} + \mathcal{L}_{PF}, \\ \mathcal{L}_{Elas} &= \frac{1}{N_x \times N_y} \left(A_{tc} \sum_{i=1}^{N_x \times N_y} f_e(\boldsymbol{x}_i^{tc}) + A_c \sum_{i=1}^{N_x \times N_y} f_e(\boldsymbol{x}_i^{c}) + A_{bc} \sum_{i=1}^{N_x \times N_y} f_e(\boldsymbol{x}_i^{bc}) \right), \\ \mathcal{L}_{PF} &= \frac{1}{N_x \times N_y} \left(A_{tc} \sum_{i=1}^{N_x \times N_y} f_c(\boldsymbol{x}_i^{tc}) + A_c \sum_{i=1}^{N_x \times N_y} f_c(\boldsymbol{x}_i^{c}) + A_{bc} \sum_{i=1}^{N_x \times N_y} f_c(\boldsymbol{x}_i^{bc}) \right), \\ f_e(\boldsymbol{x}) &= g(\phi(\boldsymbol{x}))\Psi_0(\boldsymbol{\epsilon}(\boldsymbol{x})), \\ f_c(\boldsymbol{x}) &= \frac{G_c}{2l_0} \left(\phi(\boldsymbol{x})^2 + l_0^2 |\nabla \phi(\boldsymbol{x})|^2 \right) + g'(\phi(\boldsymbol{x}))H(\boldsymbol{x},t), \end{aligned} \tag{86}$$

where $\mathcal{L}_{Elas}$ and $\mathcal{L}_{PF}$ define the losses in the elastic strain energy and the fracture energy, respectively. The scatter plots of the deformed configuration and the corresponding phase field plots for the simulation are shown in Fig. 19 and Fig. 20, respectively.

In the next section, we apply DEM on an electro-mechanically coupled systems.

6.6. Piezoelectricity

The electro-mechanical coupling is generated in some crystal materials where an electrical polarization, $\mathbf{P}$ is induced due to the existence of the mechanical strain or the existence of an electrical field can cause deformation of the geometry. The linear dependence between the electric polarization and the associated mechanical strain refers to the piezoelectric effect which is common in non-centrosymmetric crystals. The direct and the indirect piezoelectric effect can be expressed in mathematical form of as

$$\begin{aligned} \mathbf{P} &= \mathbf{p} : \boldsymbol{\epsilon}, \\ \boldsymbol{\sigma} &= \mathbf{p} \cdot \boldsymbol{E}, \end{aligned} \tag{87}$$

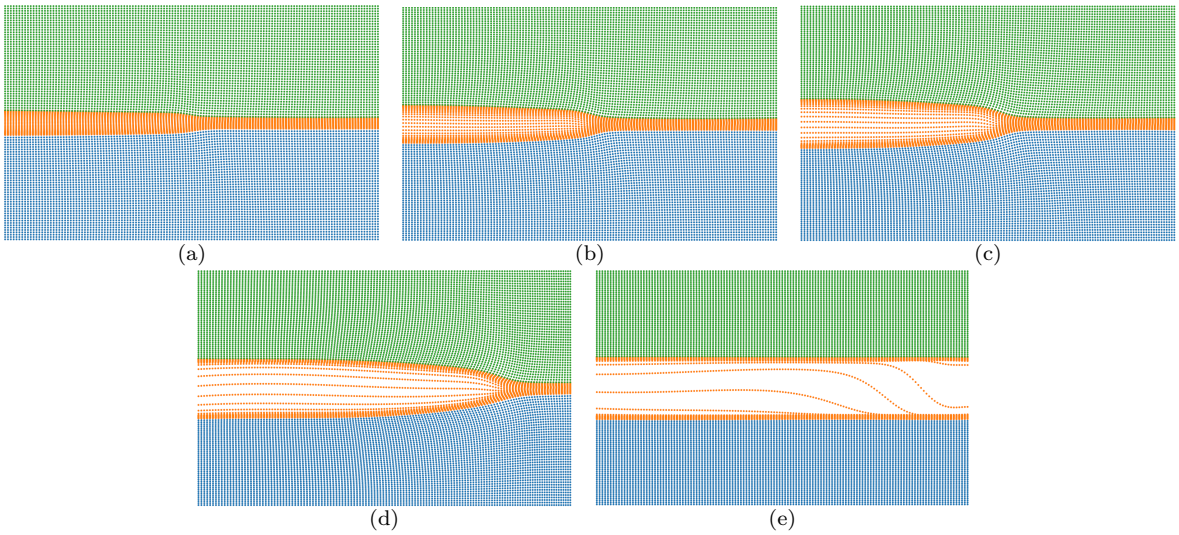


Figure 19: Scatter plots of the deformed configuration for prescribed displacement of (a) 1×10^{-3}, (b) 2×10^{-3}, (c) 3×10^{-3}, (d) 4×10^{-3} and (e) 5×10^{-3}.

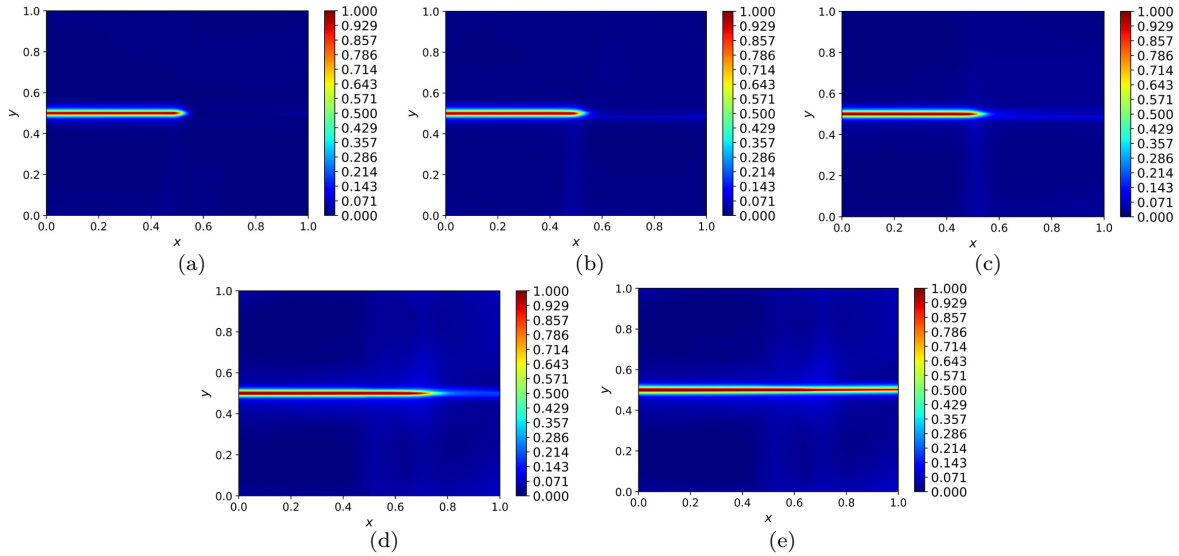


Figure 20: Crack pattern for prescribed displacement of (a) 1×10^{-3}, (b) 2×10^{-3}, (c) 3×10^{-3}, (d) 4×10^{-3} and (e) 5×10^{-3}.

where $\mathbf{p}$ refers to the third order piezoelectric tensor, $\boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}$ are the second order strain and stress tensors, and $\boldsymbol{E}$ is the electric field vector with $E_i = -\theta_{,i}$ and θ is the electric potential. It can be concluded that the piezoelectricity is a cross coupling between the elastic and the dielectric variables Hence, the constitutive equations to identify the coupling between the mechanical stress and the strain with the related electric field and displacement are given by

$$\begin{aligned} \boldsymbol{D} &= \boldsymbol{e} : \boldsymbol{\epsilon} + \boldsymbol{\kappa} \cdot \boldsymbol{E}, \\ \boldsymbol{\sigma} &= \mathbb{C} : \boldsymbol{\epsilon} - \boldsymbol{e} \cdot \boldsymbol{E}, \end{aligned} \tag{88}$$

where $\boldsymbol{D}$ refers to the electric displacement that causes the electrical polarization, while $\boldsymbol{e}$, $\boldsymbol{\kappa}$, $\mathbb{C}$ are the piezoelectric, the dielectric, and the elastic tensors, respectively.

Energy minimization is adopted in the DEM to obtain the solution of the field variables. The problem statement reads as

$$\begin{aligned} \text{Minimize:} \quad & \mathcal{E} = \mathcal{E}_i + W_{ext}, \\ \text{subject to:} \quad & \boldsymbol{u} = \overline{\boldsymbol{u}} \text{ on } \partial\Omega_{D_u}, \\ \text{and} \quad & \theta = \overline{\theta} \text{ on } \partial\Omega_{D_\theta}, \\ \text{and} \quad & \boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{t}_N \text{ on } \partial\Omega_N, \end{aligned} \tag{89}$$

where $\mathcal{E}$ represents the total energy of the system, $\mathcal{E}_i$ is the bulk internal energy, whose density is Ψ, and W_{ext} stands for the work of external forces. The internal energy density, Ψ, is expressed as

$$\Psi = \frac{1}{2}\boldsymbol{\epsilon} : \mathbb{C} : \boldsymbol{\epsilon} - \boldsymbol{E} \cdot \boldsymbol{e} \cdot \boldsymbol{E} - \frac{1}{2}\boldsymbol{E} \cdot \boldsymbol{\kappa} \cdot \boldsymbol{E}. \tag{90}$$

We shall now present the example of a cantilever beam to show the application of DEm to solve PDEs involving electromechanical coupling.

6.6.1. Cantilever beam

In this example, a cantilever beam configuration (the most common problem in evaluating the effect of piezoelectricity) is solved under plane strain conditions. The material is assumed to be isotropic and linearly elastic. The geometrical setup and the boundary conditions are shown in Fig. 21. A fully connected network with three hidden layers of 150 neurons each is used to solve the electro-mechanically coupled problem. For the first two layers, we have considered hyperbolic tangent, `tanh` activation function; whereas for the last layer, linear activation function has been considered. To optimize the hyper-parameters, the loss function is calculated similar to the examples discussed in the previous sections. For training, we have used the ADAM optimizer followed by L-BFGS.

In this problem, we investigate two loading conditions:

1. pure mechanical loading
2. pure electrical loading

In the case of pure mechanical loading, a point load is applied as shown in Fig. 21. Due to the electro-mechanical coupling, an electrical potential, θ is induced by the mechanical load. Assuming closed circuit configuration, θ is fixed to zero on the bottom surface while the electrode placed on the top surface undergoes a difference in electric potential as a result of the deformation. Fig. 22b shows the plots of the deformed shape and the electric potential obtained using the DEM approach.

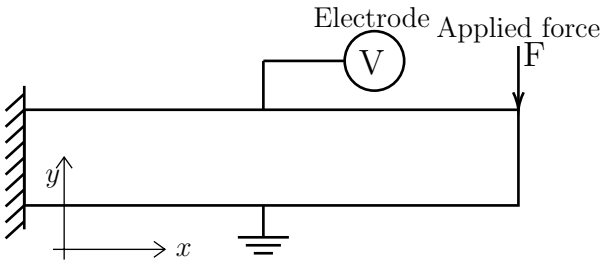


Figure 21: Schematic diagram showing the electrical and the mechanical boundary conditions for the cantilever nanobeam under study.

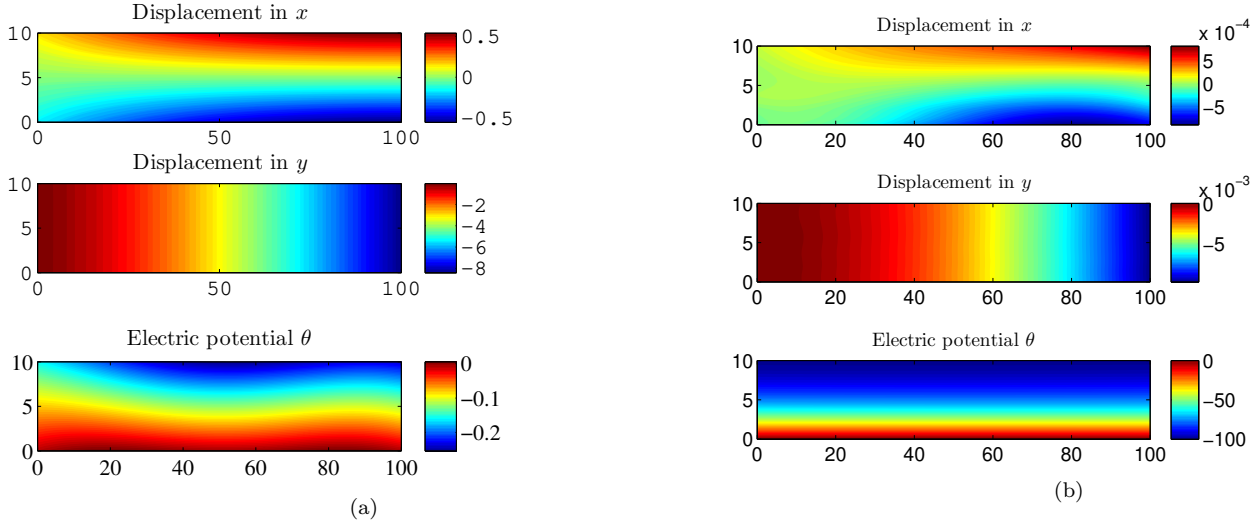


Figure 22: The predicted values of three outputs of the system using DEM: (a) applying mechanical loading, and (b) applying electrical loading.

In the next step, we consider pure electrical loading. In this case, the electric potential on the top surface is fixed to $V = -100$ MV. The defomed configuration is presented in Fig. 22b. Both the cases considered in this example, demonstrate the ability of the DEM to solve electro-mechanically coupled system without the aid of the classical numerical methods.

All the examples discussed previously involved second-order PDEs. To show the robustness of DEM, we take up the study of a fourth-order PDE in the next section.

6.7. Kirchhoff Plate bending

The Kirchhoff plate bending problem is a classical problem governed by a fourth-order PDE. The weak formulation of the problem involves the second-order derivatives of the transversal deflection. In traditional mesh-based methods, the essential requirement of C^1 continuity poses significant challenges. This, however, can be easily solved by the proposed deep collocation and the deep energy methods with a set of DNNs approximating the transversal deflection, which are proven to be effective in the bending analysis of Kirchhoff plate of various geometries and even with cut-outs.

The governing equation of Kirchhoff bending problem, expressed in terms of transversal

deflection, is

$$\bigtriangledown^2 \left(\bigtriangledown^2 w\right) = \bigtriangledown^4 w = \frac{p}{D}, \tag{91}$$

where $\bigtriangledown^4 (\cdot) = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4}$ is commonly referred to as biharmonic operator. Similar to the previous examples, the problem statement in strong form is equivalent the minimization of the total potential energy of the system, $\mathcal{E}$, where

$$\mathcal{E} = \iint_{\Omega} \left(\frac{1}{2}\boldsymbol{k}^T \boldsymbol{M} - qw\right) d\Omega - \int_{S_3} \bar{V}_n w dS + \int_{S_2+S_3} \bar{M}_n \frac{\partial w}{\partial n} dS \tag{92}$$

under uniformly distributed load. The problem is stated as:

$$w = \underset{v \in \ddot{H(D)}}{argmin}\, \mathcal{E}(v) \tag{93}$$

Additionally, the boundary conditions of the Kirchhoff plate taken into consideration can be classified into three types; simply-supported, clamped and free boundary conditions and is expressed as

$$\partial\Omega = \Gamma_1 + \Gamma_2 + \Gamma_3. \tag{94}$$

To solve Kirchhoff bending problems with boundary constraints, the physical domain is first discretized with randomly distributed collocation points denoted by $\boldsymbol{x}_\Omega = (x_1, \ldots, x_{N_\Omega})^T$. Another set of collocation points are added to discretize the boundaries denoted by $\boldsymbol{x}_\Gamma (x_1, \ldots, x_{N_\Gamma})^T$. The transversal deflection, w can thus be approximated with the aforementioned deep feed-forward neural network $w^h(\boldsymbol{x}; \boldsymbol{p})$. The loss function is constructed to find the approximate solution by minimizing either the mean squared error of the governing equation in residual form or the total energy, along with boundary conditions approximated by $w^h(\boldsymbol{x}; \boldsymbol{p})$.

Substituting $w^h(\boldsymbol{x}_\Omega; \boldsymbol{p})$ into either Equation 91 or into 92, a physical informed deep neural network is obtained: $G(\boldsymbol{x}_\Omega; \boldsymbol{p})$ for the strong form of the problem, and $\mathcal{E}_p(\boldsymbol{x}_\Omega; \boldsymbol{p})$ the energy approach, respectively.

The boundary conditions can also be expressed by the physical informed neural network approximation $w^h(\boldsymbol{x}_\Gamma; \boldsymbol{p})$:
On clamped boundaries Γ_1, we have

$$w^h(\boldsymbol{x}_{\Gamma_1}; \boldsymbol{p}) = \tilde{w}, \;\; \frac{\partial w^h(\boldsymbol{x}_{\Gamma_1}; \boldsymbol{p})}{\partial n} = \tilde{\boldsymbol{p}}_n. \tag{95}$$

On simply-supported boundaries Γ_2,

$$w^h(\boldsymbol{x}_{\Gamma_2}; \boldsymbol{p}) = \tilde{w}, \;\; \tilde{M}_n(\boldsymbol{x}_{\Gamma_2}; \boldsymbol{p}) = \tilde{M}_n, \tag{96}$$

On free boundaries Γ_3,

$$M_n(\boldsymbol{x}_{\Gamma_3}; \boldsymbol{p}) = \tilde{M}_n, \;\; \frac{\partial M_{ns}(\boldsymbol{x}_{\Gamma_3}; \boldsymbol{p})}{\partial s} + Q_n(\boldsymbol{x}_{\Gamma_3}; \boldsymbol{p}) = \tilde{q}, \tag{97}$$

It should be noted that $\boldsymbol{n}, \boldsymbol{s}$ here refer to the normal and tangent directions along each boundary.

It is clear that all induced physical informed neural network in Kirchhoff bending analysis $G(\boldsymbol{x}; \boldsymbol{p})$, $\mathcal{E}(\boldsymbol{x}_\Omega; \boldsymbol{p})$, $M_n(\boldsymbol{x}; \boldsymbol{p})$, $M_{ns}(\boldsymbol{x}; \boldsymbol{p})$, $Q_n(\boldsymbol{x}; \boldsymbol{p})$ share the same parameters as $w^h(\boldsymbol{x}; \boldsymbol{p})$. Considering the generated collocation points in domain and on boundaries, they can all be

learned by minimizing the mean square error loss function:

$$L(\boldsymbol{p}) = MSE = MSE_G + MSE_{\Gamma_1} + MSE_{\Gamma_2} + MSE_{\Gamma_3}, \tag{98}$$

with

$$\begin{aligned}
MSE_G &= \frac{1}{N_d}\sum_{i=1}^{N_d}\left\|G\left(\boldsymbol{x}_\Omega;\boldsymbol{p}\right)\right\|^2 = \frac{1}{N_\Omega}\sum_{i=1}^{N_\Omega}\left\|\bigtriangledown^4 w^h\left(\boldsymbol{x}_\Omega;\boldsymbol{p}\right) - \tfrac{p}{D}\right\|^2, \\
MSE_{\Gamma_1} &= \frac{1}{N_{\Gamma_1}}\sum_{i=1}^{N_{\Gamma_1}}\left\|w^h\left(\boldsymbol{x}_{\Gamma_1};\boldsymbol{p}\right) - \tilde{w}\right\|^2 + \frac{1}{N_{\Gamma_1}}\sum_{i=1}^{N_{\Gamma_1}}\left\|\tfrac{\partial w^h\left(\boldsymbol{x}_{\Gamma_1};\boldsymbol{p}\right)}{\partial n} - \tilde{\boldsymbol{p}}_n\right\|^2, \\
MSE_{\Gamma_2} &= \frac{1}{N_{\Gamma_2}}\sum_{i=1}^{N_{\Gamma_2}}\left\|w^h\left(\boldsymbol{x}_{\Gamma_2};\boldsymbol{p}\right) - \tilde{w}\right\|^2 + \frac{1}{N_{\Gamma_2}}\sum_{i=1}^{N_{\Gamma_2}}\left\|\tilde{M}_n\left(\boldsymbol{x}_{\Gamma_2};\boldsymbol{p}\right) - \tilde{M}_n\right\|^2, \\
MSE_{\Gamma_3} &= \frac{1}{N_{\Gamma_3}}\sum_{i=1}^{N_{\Gamma_3}}\left\|\tilde{M}_n\left(\boldsymbol{x}_{\Gamma_3};\boldsymbol{p}\right) - \tilde{M}_n\right\|^2 + \frac{1}{N_{\Gamma_3}}\sum_{i=1}^{N_{\Gamma_3}}\left\|\tfrac{\partial M_{ns}\left(\boldsymbol{x}_{\Gamma_3};\boldsymbol{p}\right)}{\partial s} + Q_n\left(\boldsymbol{x}_{\Gamma_3};\boldsymbol{p}\right) - \tilde{q}\right\|^2,
\end{aligned} \tag{99}$$

where $x_{\,\Omega} \in \mathbb{R}^N$, $\boldsymbol{p} \in \mathbb{R}^K$ are the neural network parameters. $L(\boldsymbol{p}) = 0$, $w^h(\boldsymbol{x};\boldsymbol{p})$ is then a solution to transversal deflection. Our goal is to find a set of parameters $\boldsymbol{p}$ such that the approximated deflection $w^h(\boldsymbol{x};\boldsymbol{p})$ minimizes the loss $L(\boldsymbol{p})$. If $L(\boldsymbol{p})$ is a very small value, the approximation $w^h(\boldsymbol{x};\boldsymbol{p})$ is very closely satisfying governing equations and boundary conditions, namely

$$\boldsymbol{p} = \underset{\hat{\boldsymbol{p}}\in\mathbb{R}^K}{\operatorname{argmin}} L(\hat{\boldsymbol{p}}) \tag{100}$$

However, in order to solve the problem of a thin plate with the deep energy method, it is necessary to construct the loss function in a different way:

$$\mathcal{L}(\boldsymbol{p}) = MAE_{\mathcal{E}} + MSE_{\Gamma_1} + MSE_{\Gamma_2}, \tag{101}$$

Here, only due to the fact that energy method is adopted, only Dirichlet boundary conditions among three boundary types need to be taken into account. This makes it easier than the deep collocation method. The latter needs a physical discovery of Neumann boundary conditions with DNN. In this case, as shown above, the twist moment and shear force along the boundaries need to be approximated with the physical informed deep neural networks.

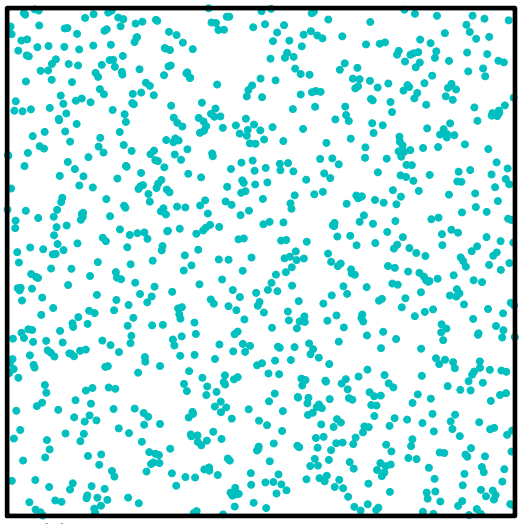
The plate bending problems with DNN based method can be accordingly reduced to an optimization problem. In the deep learning Tensorflow framework, a variety of optimizers are available to help to gain an optimal solution, and the Adam and LBFGS optimizers are mainly adopted in numerical examples.

Next, a series of numerical examples are tested to verify the feasibility of deep collocation method and deep energy method in the analysis of Kirchhoff plate bending problems.

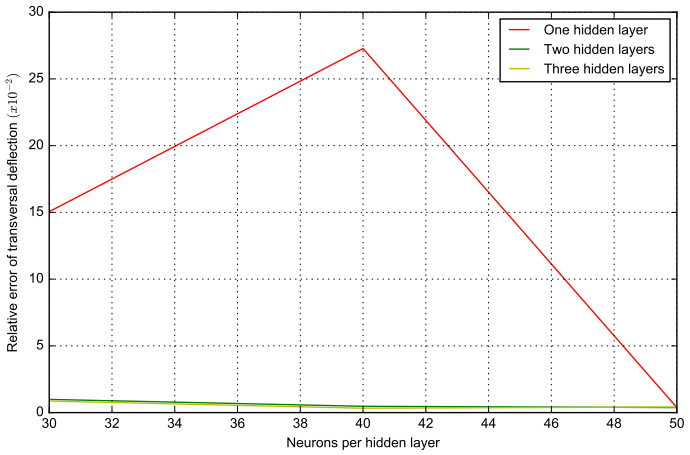
6.7.1. Simply-supported square plate on Winkler foundation

The deep collocation method is first applied to study the bending problems of various plates, which are chosen with analytical or exact solutions as comparisons.

To begin with, we consider a simply-supported square plate resting on Winkler foundation, which assumes that the foundation's reaction $p(x,y)$ can be described by $p(x,y) = kw$, k being a constant called *foundation modulus*. For a plate on a continuous Winkler foundation,



(a) Collocation points in the square domain.



(b) The relative error of deflection with varying hidden layers and neurons.

Figure 23

Table 1: Maximum deflection predicted by deep collocation method.

Square Plate on Winkler foundation	Predicted Maximum Deflection	Analytical solution
1 hidden layer, 30 neurons	0.33999	0.32137
1 hidden layer, 40 neurons	0.35689	
1 hidden layer, 50 neurons	0.32168	
2 hidden layers, 30 neurons	0.32248	
2 hidden layers, 40 neurons	0.32176	
2 hidden layers, 50 neurons	0.32168	
3 hidden layers, 30 neurons	0.32216	
3 hidden layers, 40 neurons	0.32172	
3 hidden layers, 50 neurons	0.32181	

the governing Equation 91 can be written as

$$\bigtriangledown^2\left(\bigtriangledown^2 w\right) = \bigtriangledown^4 w = \frac{(p-q)}{D} = \frac{(p-kw)}{D} \tag{102}$$

Here, D denotes the flexural stiffness of the plate depending on the plate thickness and material properties.

The analytical solution for this numerical example is [36]:

$$w = \frac{16p}{ab}\sum_{m=1,3,5,\cdots}^{\infty}\sum_{n=1,3,5,\cdots}^{\infty}\frac{\sin\frac{m\pi x}{a}\sin\frac{n\pi y}{b}}{mn\left[\pi^4 D\left(\frac{m^2}{a^2}+\frac{n^2}{b^2}\right)^2+k\right]} \tag{103}$$

For this numerical example, the arrangement of collocation points is depicted in Figure 23(a). Neural networks with different neurons and depth are applied in the calculation to study the feasibility of deep collocation in solving Kirchhoff plate bending problems. Table 1 lists the maximum deflection at the central point with varying hidden layers and neurons. It is clear that the results predicted by more hidden layers are more desirable and as hidden layer and neuron number grows, the maximum deflection becomes more accurate approaching the analytical serial solution for even two hidden layers. The relative error shown in

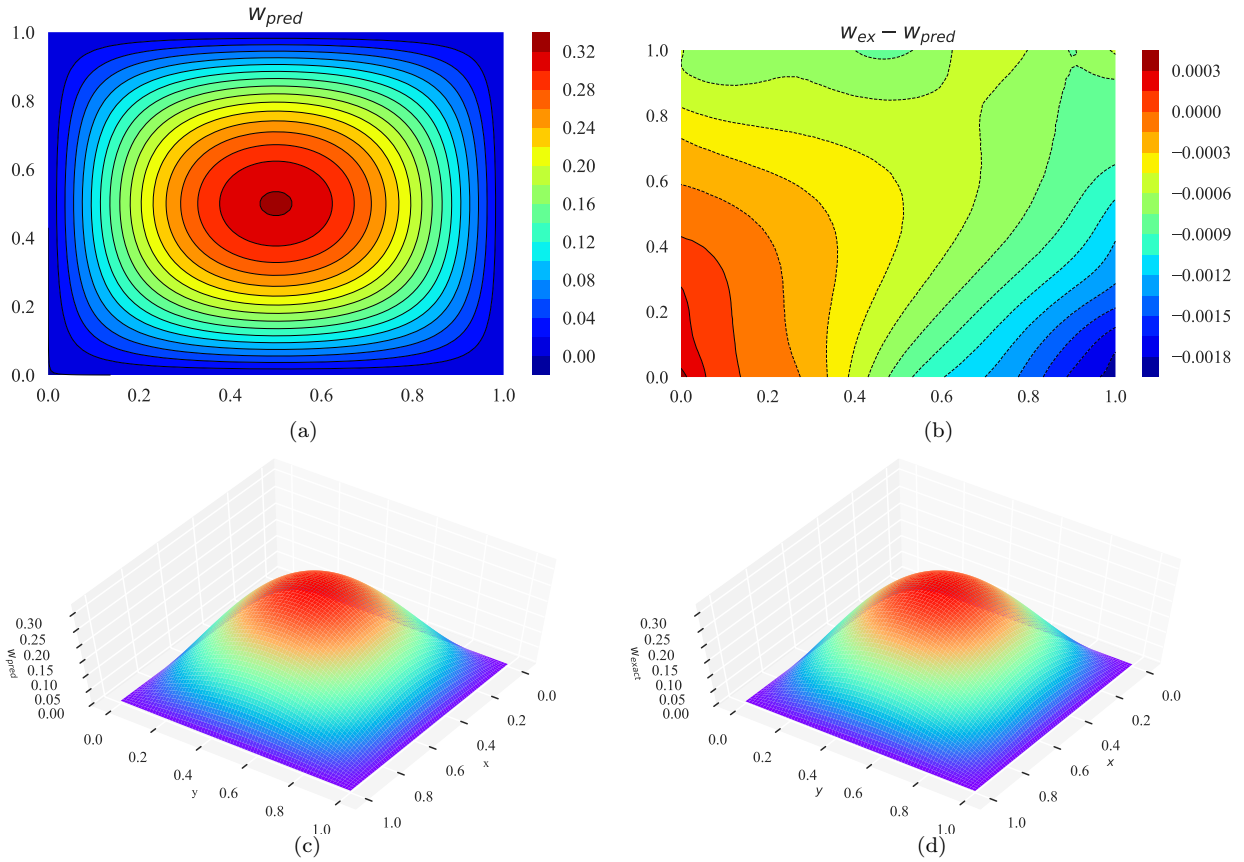


Figure 24: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Analytical deflection of the simply-supported plate on Winkler foundation with 3 hidden layers and 50 neurons.

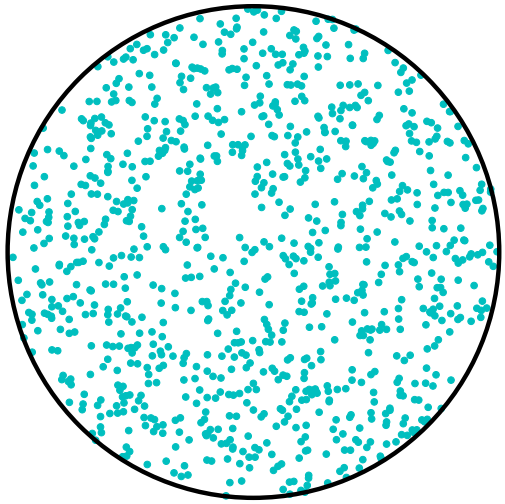
Figure 23(b) better depicts the advantages of deep neural network than shallow wide neural network. More hidden layers, with more neurons yield flatting of the relative error. Various contour plots are shown in Figure 24 and compared with the analytical solution. It is obvious that the deep collocation method predict the deflection of the whole plate in good accordance with the analytical solution.

6.7.2. Clamped circular plate

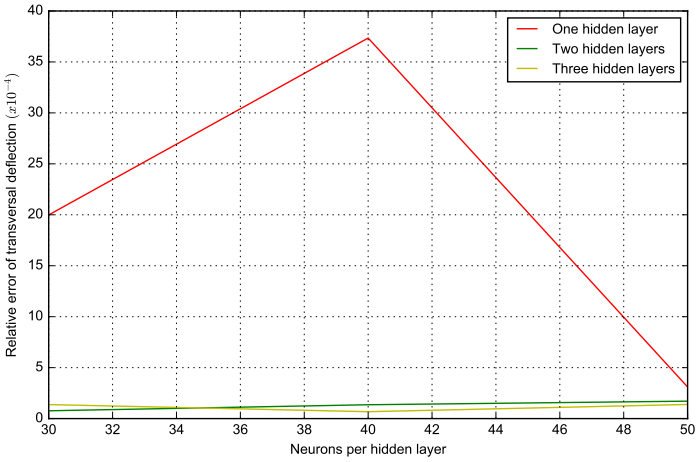
Also, in this section, we study the feasibility of deep collocation method in the analysis of Kirchhoff plate bending problems. We first apply the deep collocation method in studying a clamped circular plate with radius R under a uniform load p. 1000 collocation points shown in Figure 25(a) are employed in the domain of the circular plate. The exact solution of this problem can be referred in [36]:

$$w = \frac{p\left(R^2 - (x^2 + y^2)\right)^2}{64D}, \tag{104}$$

D denoting the flexural stiffness. The maximum deflection at the central of the circular plate with varying hidden layers and neurons is summarized in Table 3 and compared with the exact solution. The predicted maximum deflection become more accurate with increasing number of neurons and hidden layers. The relative error for deflection of clamped circular



(a) Collocation points in the circular domain



(b) The relative error of deflection with varying hidden layers and neurons

Figure 25

Table 2: Maximum deflection predicted by deep collocation method.

Clamped Circular Plate	Predicted Maximum Deflection	Exact solution
1 hidden layers, 30 neurons	15.5958	15.6250
1 hidden layers, 40 neurons	15.5685	
1 hidden layers, 50 neurons	15.6201	
2 hidden layers, 30 neurons	15.6251	
2 hidden layers, 40 neurons	15.6264	
2 hidden layers, 50 neurons	15.6224	
3 hidden layers, 30 neurons	15.6269	
3 hidden layers, 40 neurons	15.6247	
3 hidden layers, 50 neurons	15.6229	

plate with increasing hidden layers and neurons is depcited in Figure 25(b). As hidden layer number increases, the relative error curves flatten out and converges to the exact solution. All neural networks perform well with a relative error magnitude of 1×10^{-4}.

The deformation contour, deflection error contour, predicted and exact deformation figures are displayed in Figure 26. It is clear that the predicted deflection of this circular plate agrees well with the exact solution.

6.7.3. Simply-supported square plate under a sinusoidally distributed load

For the next two numerical examples, deep energy method is applied in the calculation. For deep energy method, we use a different DNN scheme based on Pytorch, and to further improve the primary results, different strategies have been proposed, including an autoencoder and tailored activation function, which has been proved to be feasible and efficient during the numerical experiment. The LBFGS optimizer is applied during the training of the deep neural network. The numerical experiments are executed on a 64-bit macOS Mojave server with Intel(R) Core(TM) i7-8850H CPU, 32GB memory.

First, a simply-supported square plate under a sinusoidal distribution transverse loading

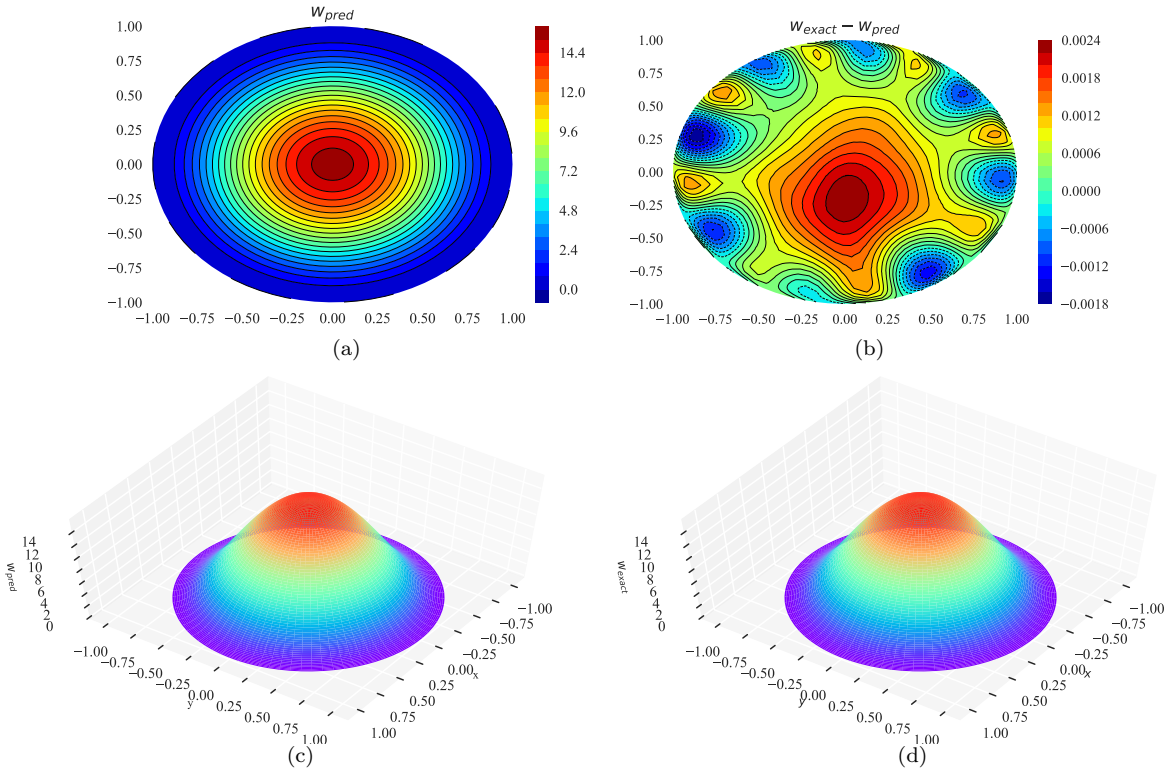


Figure 26: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Exact deflection of the clamped circular plate with 3 hidden layers and 50 neurons

is studied with deep energy method. The distributed load is given by

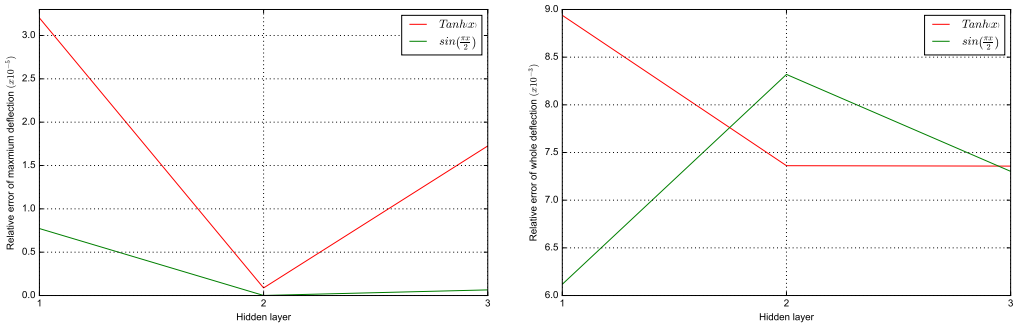
$$p = \frac{p_0}{D}\sin\left(\frac{\pi x}{a}\right)\sin\left(\frac{\pi y}{b}\right). \tag{105}$$

where a,b are the length of the plate.The analytical solution for this problem is given by

$$w = \frac{p_0}{\pi^4 D\left(\frac{1}{a^2}+\frac{1}{b^2}\right)^2}\sin\left(\frac{\pi x}{a}\right)\sin\left(\frac{\pi y}{b}\right). \tag{106}$$

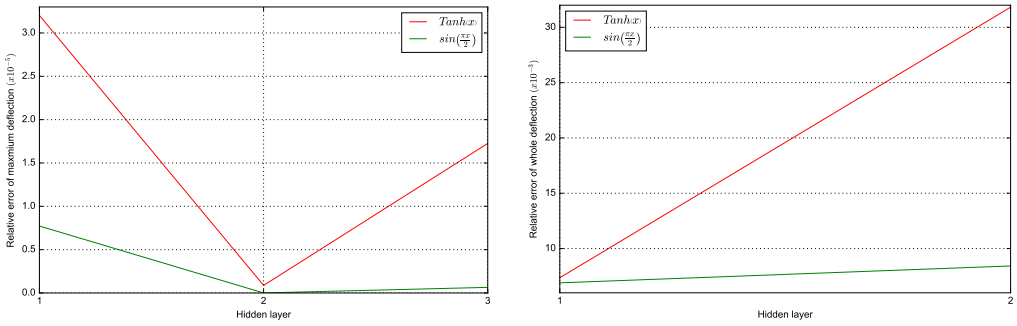
For the detailed configuration of this numerical example, a $31x31$ random distributed collocation point set shown in Figure 23 is first generated in the plate domain. Based on this, the Monte Carlo integration scheme is adopted in the calculation of total potential energy. For this problem with sinusoidal load, the activation function was tailored accordingly as $f(x) = sin\left(\frac{\pi x}{2}\right)$, which improves the accuracy of the results. A neural network with 50 neurons each hidden layer is deployed to study the relative error of maximum deflection at central points and the whole deflection by the proposed activation function and classical Tanh activation function. And for the DNN, three hidden layers is deployed. As for DNN with an autoencoder, two encoding layer configurations $[2,1]*H$ and $[3]*H$ are considered, with H the number of neurons on each encoding layer. It is shown clearly in Figure 27, Figure 28 that the numericals gained by proposed activation function outweigh classical Tanh activation function for both two neural network schemes.

Moreover, we carry on studying an accurate and efficient configuration of the proposed DNN with an autoencoder, in hope for a better discovery of its physical patterns. For this numerical example, ten encoding layer types are studied and shown in Figure 29. We



(a) (b)

Figure 27: Relative error of (a) maximum deflection and (b) whole deflection predicted by Tanh and proposed activation function of a DNN for the simply-supported plate



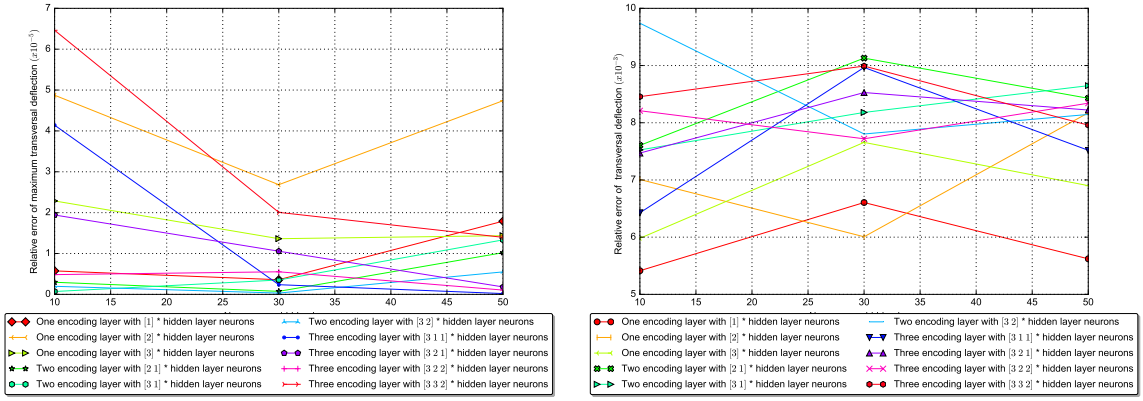
(a) (b)

Figure 28: Relative error of (a) maximum deflection and (b) whole deflection predicted by Tanh and proposed activation function of a DNN with an autoencoder for the simply-supported plate

begin by studying the influence of different encoding layer with different neurons per layer on the accuracy and efficiency of this problem. We can see that with more encoding layer and neurons per layer, it will obtain a more stable and accurate results. Moreover, the corresponding computational cost of those ten schemes are listed in Figure 30. It is clear that a two encoding layer autoencoder can meet both accuracy and efficiency ends. And the neurons on each scheme increase, it is clearly that the numerical solution converges to the analytical solution.

Finally, the deep energy scheme is then compared with an open source IGA-FEM code from on this problem [37], which is also based on the kirchhoff plate theory. The accuracy and computational time of both scheme are compared. For DEM, a single encoding layer with 30 neurons is applied with DNN including increasing collocation points. And the computational time for DEM means the training time. It is clear that IGA-FEM is clearly faster and more accurate. But, the performance of DEM is still in the acceptance range. Once the DNN with an autoencoder is trained, it can be very fast used to predict deflection and stress in the whole plate.

For deep energy method configuration, an auto-encoder is added to the deep neural networks to better capture the physical patterns of this problem. We also studied the optimal configuration of this auto encoder with different layers and neuron numbers, which will be beneficial to the further application of this deep energy method.



(a) (b)

Figure 29: Relative error of (a) maximum deflection and (b) whole deflection predicted by different encoding layer configurations of a DNN with an autoencoder for the simply-supported plate.

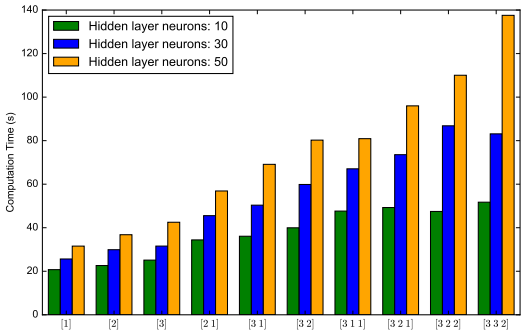
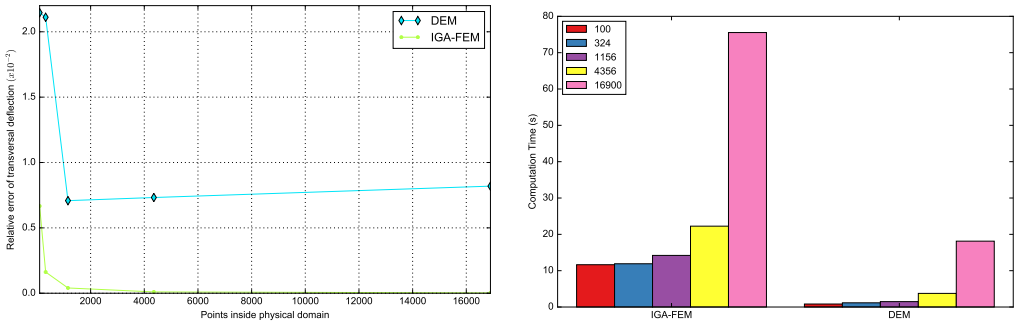


Figure 30: The computational time of ten encoding layer configurations for the autoencoder



(a) (b)

Figure 31: Relative error of (a) deflection and (b) predicted by the DNN with an autoencoder and IGA for the simply-supported plate.

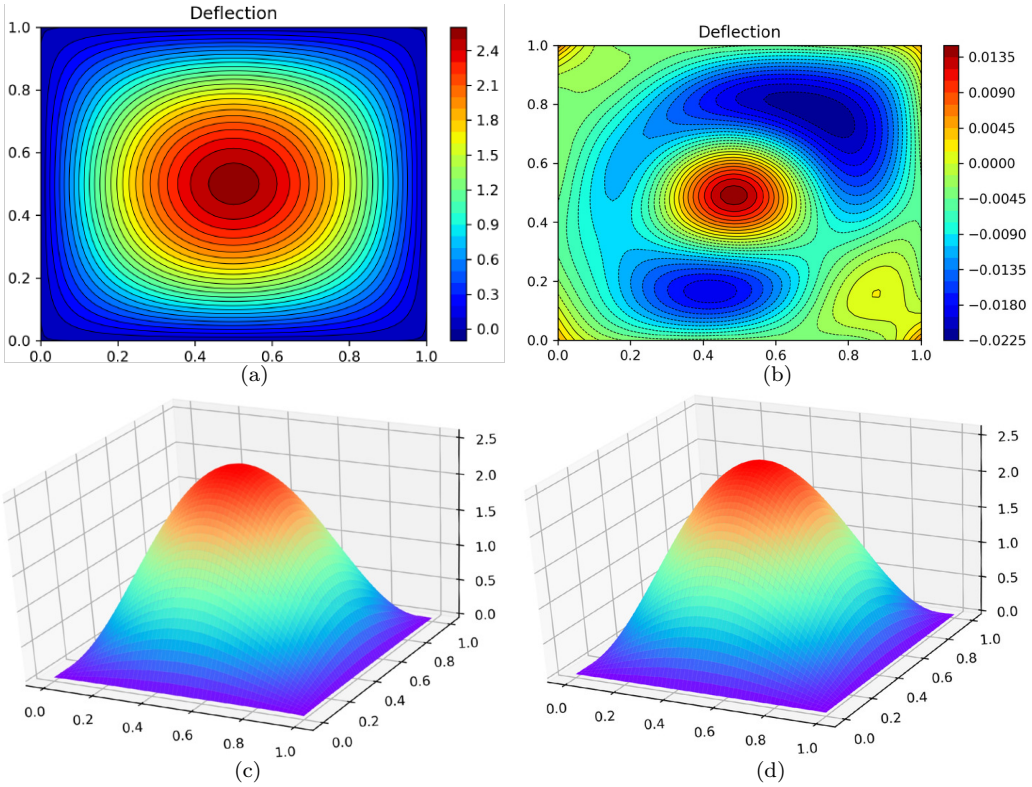


Figure 32: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Analytical deflection of the simply-supported square plate.

To better reflect the deflection vector in the whole physical domain, the contour plot, contour error plot of deflection predicted by two encoding layers with [150,100] neurons are shown in Figure 32, which agree well with deflection obtained from analytical solution.

6.7.4. Simply-supported annular plate

In this section, an annular plate, which is simply-supported on the outer circle and free on the inner circle, is studied with deep energy method, for this problem, deep collocation method performs poorly for a feed-forward DNN, when considering free boundary conditions. However, for energy based method, only the essential boundary conditions needs to be taken into consideration, which simplifies the problem. The numerical results predicted by the deep energy method lend credence to the feasibility of deep energy method in analysis of plates with cut-out.

The analytical solution of this problem is [36]:

$$w = \frac{qa^4}{64D}\left\{-\left[1-\left(\frac{r}{a}^4\right)\right]+\frac{2\alpha_1}{1+\nu}\left[1-\left(\frac{r}{a}\right)^2\right]-\frac{4\alpha_2\beta^2}{1-\nu}\log\left(\frac{r}{a}\right)\right\}, \tag{107}$$

where $\alpha_1 = (3+\nu)(1-\beta^2) - 4(1+\nu)\beta^2\kappa$, $\alpha_2 = (3+\nu) + 4(1+\nu)\kappa$, $\beta = \frac{b}{a}$, $\kappa = \frac{\beta^2}{1-\beta^2}\log\beta$, with a, b the outer and inner radius of the annular plate respectively.

Likewise, we studied the deflection at a certain point to study the convergency of deflection with the varying encoding layers, which is chosen for annular plate as $(\frac{a+b}{2}, 0)$. In this numerical example, 1000 collocation points are generated in the physical domain. A DNN

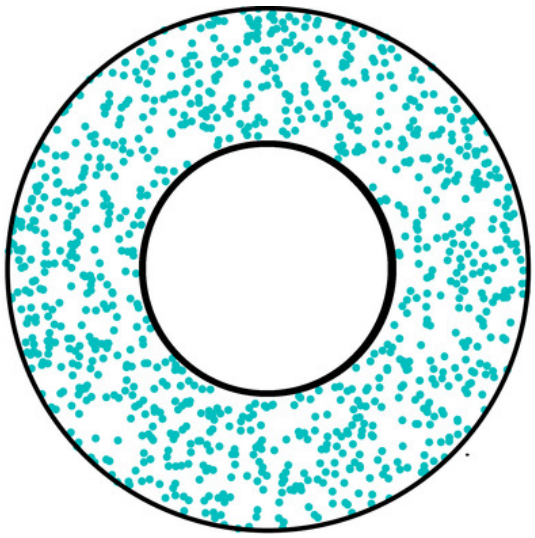
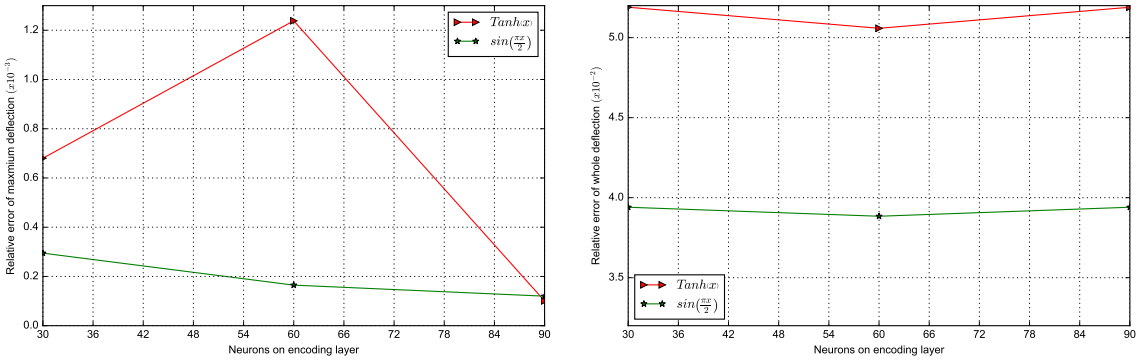


Figure 33: Collocation points in the annular domain.



(a) (b)

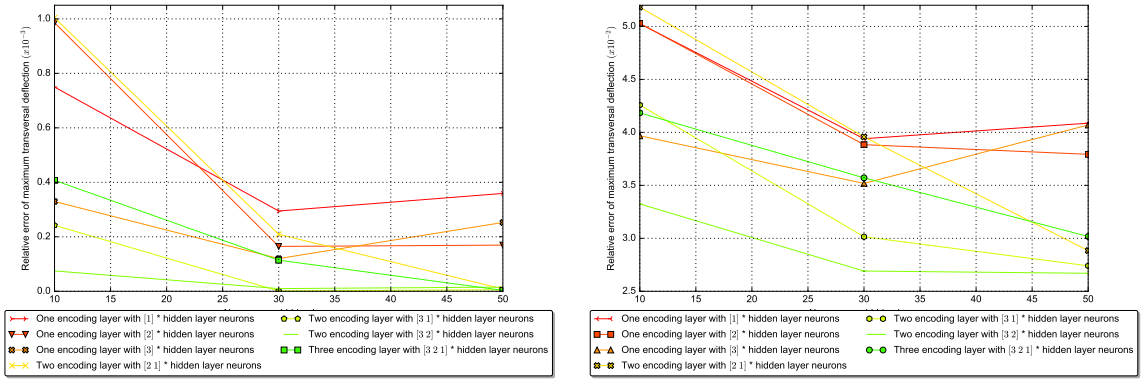
Figure 34: Relative error of (a) maximum deflection and (b) whole deflection predicted by *tanh* and proposed activation function of a DNN for the annular plate.

with an autoencoder is also constructed which will help better predict the physical pattern of this problem. During our numerical example, it is discovered that the proper point sampling inside physical domain affects the stability and accurate of this deep energy method. And compared with some other sampling method, the collocation points generated in Figure 33 can be suitable for the numerical analysis of this problem.

As for the numerical analysis, we first test the deep neural network with an autocoder with different activation function, namely the mostly used $tanh\,(x)$ and the proposed $sin\left(\frac{\pi x}{2}\right)$. Shown in Figure 34 is very clear that the proposed activation based DNN better predicts the deflection for the whole plate.

Further, we studied some chosen encoding layer configurations in affecting the deflection of the annular plate and its corresponding computational cost. Shown in Figure 37, as the number of neurons and encoding layers increase, the deflection converges to the analytical solution. Combined with computational cost demonstrated in Figure 36, it can be concluded that two encoding layers can be sufficient for the bending analysis of Kirchhoff plate.

Additionally, the deflection contour and the error contour are also shown in Figure 37. We



(a) (b)

Figure 35: Relative error of (a) maximum deflection and (b) whole deflection predicted by different encoding layer configurations of a DNN with an autoencoder for the simply-supported plate.

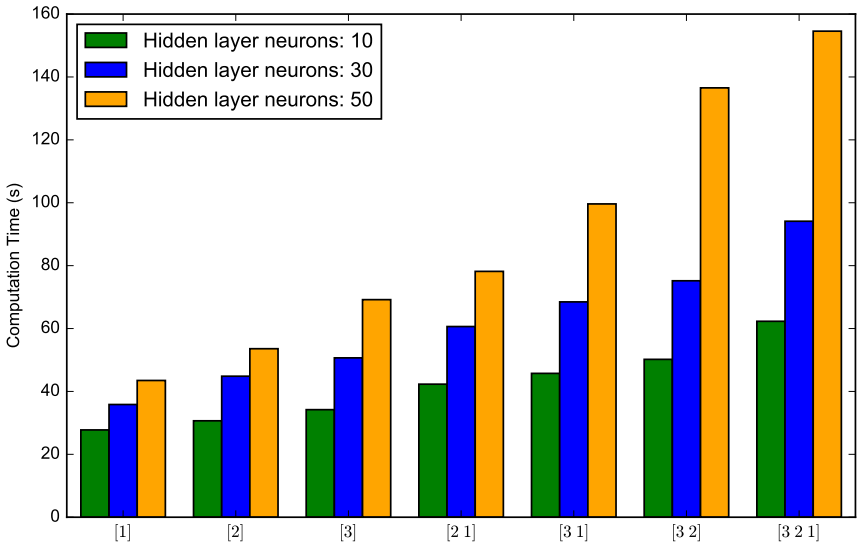


Figure 36: The computational time of seven encoding layer configurations for the autoencoder.

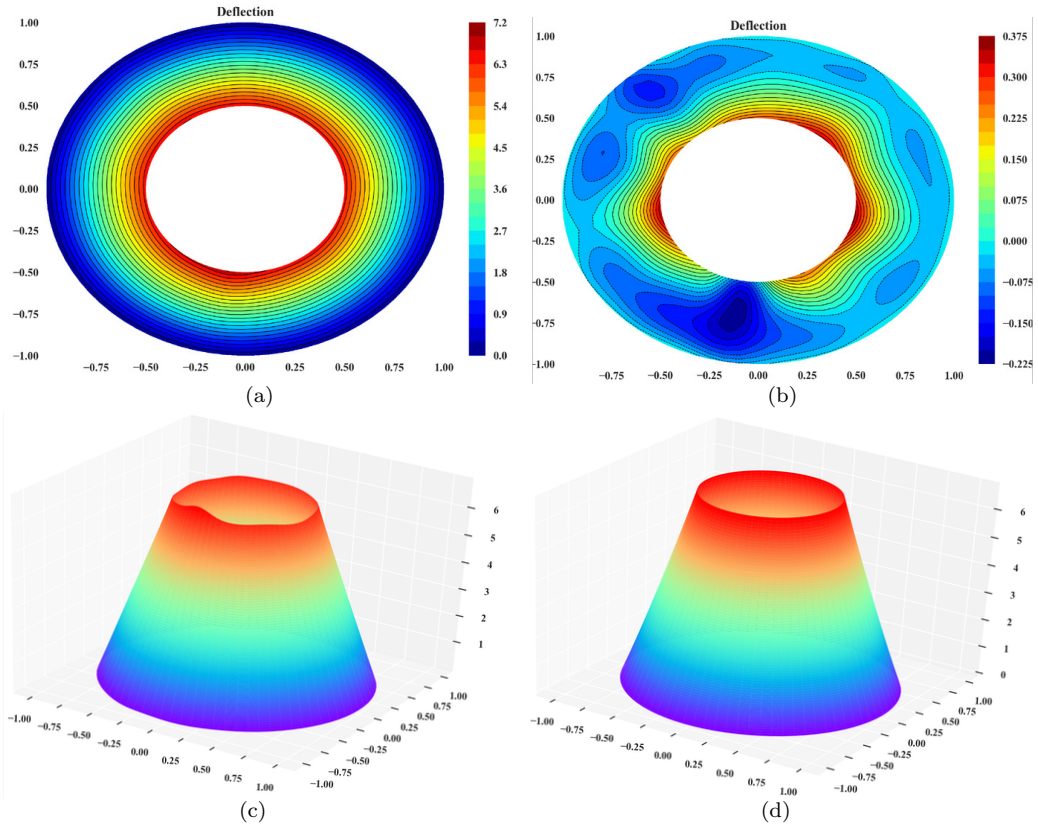


Figure 37: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Analytical deflection of the simply-supported annular plate with 3 hidden layers and 50 neurons.

can conclude from Figure 37 that the transversal deflection of Kirchhoff plate predicted by our deep energy method agrees well with the analytical solution, even for plate with hole.

7. Concluding remarks

The use of DNNs to solve boundary value problems has been explored. Several very relevant examples from computational mechanics have been solved using DNNs to build the approximation space. This constitutes a proof of concept for the possibility of approximating the solution of BVPs using concepts and tools coming from deep learning.

The first pillar of our approach is to use an energy characterizing the behaviour of the physical body under study. This energy is the basis for the construction of the loss function. The second fundamental idea is that the approximation space is defined by the architecture of the neural network. This two ideas lead to a finite sum nonconvex problem. In order to solve this problem, we remain faithful to the machine learning philosophy of using gradient based optimization approaches. This allows us to use standard libraries from the machine learning community such as TensorFlow.

One of the advantages of the approach proposed here is that a wealth of concepts and tools developed by a very active community can be used. This also leads to ease of implementation. Near-mathematical notation can be used to define the loss function, which has the meaning of energy. Conceptually, this opens a door to the possibility of trying different

mathematical models by defining the corresponding energies and implementing them in a very straightforward and transparent manner.

However, there are still some important caveats. The optimization problems that have to be solved are non-convex. The approximation spaces, although being very expressive, can be very difficult to analyze. Even linear problems lead to non-linear, non-convex discrete problems. This poses several challenges that have to be explored by the computational mechanics community.

References

[1] T. J. Hughes, The finite element method: linear static and dynamic finite element analysis, Courier Corporation, 2012.

[2] A. Huerta, T. Belytschko, S. Fernández-Méndez, T. Rabczuk, X. Zhuang, M. Arroyo, Meshfree methods, Encyclopedia of Computational Mechanics Second Edition (2018) 1–38.

[3] T. J. Hughes, G. Sangalli, M. Tani, Isogeometric analysis: Mathematical and implementational aspects, with applications, in: Splines and PDEs: From Approximation Theory to Numerical Linear Algebra, Springer, 2018, pp. 237–315.

[4] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[5] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[6] N. Ketkar, Introduction to pytorch, in: Deep learning with python, Springer, 2017, pp. 195–208.

[7] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, G. N. Wells, Unified form language: A domain-specific language for weak formulations of partial differential equations, ACM Transactions on Mathematical Software (TOMS) 40 (2) (2014) 9.

[8] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, The Journal of Machine Learning Research 19 (1) (2018) 932–955.

[9] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561.

[10] B. Dacorogna, Introduction to the Calculus of Variations, World Scientific Publishing Company, 2014.

[11] J. Baiges, R. Codina, I. Castanar, E. Castillo, A finite element reduced order model based on adaptive mesh refinement and artificial neural networks.

[12] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Computer Methods in Applied Mechanics and Engineering 304 (2016) 81–101.

[13] L. Ruthotto, E. Haber, Deep Neural Networks Motivated by Partial Differential Equations, arXiv e-print 1804.04272.

[14] E. Weinan, B. Yu, The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, Communications in Mathematics and Statistics 6 (1) (2018) 1–12.

[15] L. Lei, C. Ju, J. Chen, M. I. Jordan, Non-convex finite-sum optimization via scsg methods, in: Advances in Neural Information Processing Systems, 2017, pp. 2348–2358.

[16] P. Petersen, M. Raslan, F. Voigtlaender, Topological properties of the set of functions generated by neural networks of fixed size, arXiv preprint arXiv:1806.08459.

[17] M. I. Jordan, Dynamical, symplectic and stochastic perspectives on gradient-based optimization, University of California, Berkeley.

[18] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467.

[20] J. He, L. Li, J. Xu, C. Zheng, Relu deep neural networks and linear finite elements, arXiv preprint arXiv:1807.03973.

[21] J. Opschoor, P. Petersen, C. Schwab, Deep relu networks and high-order finite element methods, SAM, ETH Zürich.

[22] Y. Jia, C. Anitescu, Y. J. Zhang, T. Rabczuk, An adaptive isogeometric analysis collocation method with a recovery-based error estimator, Computer Methods in Applied Mechanics and Engineering 345 (2019) 52–74.

[23] S. P. Timoshenko, J. Goodier, Theory of elasticity, McGraw-hill, 1970.

[24] V. Avrutskiy, Enhancing approximation abilities of neural networks by training derivatives, arXiv preprint arXiv:1712.04473.

[25] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.

[26] P. L. Gould, Y. Feng, Introduction to linear elasticity, Springer, 1994.

[27] D. Schillinger, J. A. Evans, F. Frischmann, R. R. Hiemstra, M.-C. Hsu, T. J. Hughes, A collocated c0 finite element method: Reduced quadrature perspective, cost comparison with standard finite elements, and explicit structural dynamics, International Journal for Numerical Methods in Engineering 102 (3-4) (2015) 576–631.

[28] M. A. Scott, R. N. Simpson, J. A. Evans, S. Lipton, S. P. Bordas, T. J. Hughes, T. W. Sederberg, Isogeometric boundary element analysis using unstructured t-splines, Computer Methods in Applied Mechanics and Engineering 254 (2013) 197–221.

[29] A. Logg, K.-A. Mardal, G. Wells, Automated solution of differential equations by the finite element method: The FEniCS book, Vol. 84, Springer Science & Business Media, 2012.

[30] B. Bourdin, G. Francfort, J.-J. Marigo, Numerical experiments in revisited brittle fracture, Journal of the Mechanics and Physics of Solids 48 (4) (2000) 797–826. doi:10.1016/S0022-5096(99)00028-9.

[31] A. Griffith, The Phenomena of Rupture and Flow in Solids, Philisophical Transactions of the Royal Society of London 221 (Series A) (1921) 163–198.

[32] M. J. Borden, T. J. Hughes, C. M. Landis, C. V. Verhoosel, A higher-order phase-field model for brittle fracture: Formulation and analysis within the isogeometric analysis framework, Computer Methods in Applied Mechanics and Engineering 273 (2014) 100–118. doi:10.1016/J.CMA.2014.01.016.

[33] C. Miehe, F. Welschinger, M. Hofacker, Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations, International Journal for Numerical Methods in Engineering 83 (10) (2010) 1273–1311. doi:10.1002/nme.2861.

[34] C. Miehe, M. Hofacker, F. Welschinger, A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits, Computer Methods in Applied Mechanics and Engineering 199 (45-48) (2010) 2765–2778. doi:10.1016/J.CMA.2010.04.011.

[35] M. J. Borden, C. V. Verhoosel, M. A. Scott, T. J. Hughes, C. M. Landis, A phase-field description of dynamic brittle fracture, Computer Methods in Applied Mechanics and Engineering 217-220 (2012) 77–95. doi:10.1016/J.CMA.2012.01.008.

[36] S. P. Timoshenko, S. Woinowsky-Krieger, Theory of plates and shells, McGraw-hill, 1959.

[37] V. P. Nguyen, C. Anitescu, S. P. Bordas, T. Rabczuk, Isogeometric analysis: an overview and computer implementation aspects, Mathematics and Computers in Simulation 117 (2015) 89–116.