A Deep Collocation Method for the Bending Analysis of Kirchhoff Plate

Hongwei Guo[a], Timon Rabczuk[b], and Xiaoying Zhuang [*a,c,d]

[a]Institute of Continuum Mechanics, Leibniz Universität Hannover, Appelstraße 11, 30157 Hannover, Germany
[b]Institute of Structural Mechanics, Bauhaus-Universität Weimar, Marienstr.15 D-99423 Weimar, Germany
[c]Department of Geotechnical Engineering, Tongji University, Siping Road 1239, 200092 Shanghai, P.R.China
[d]Key Laboratory of Geotechnical and Underground Engineering of Ministry of Education, Tongji University, 200092 Shanghai, P.R.China

Abstract

In this paper, a deep collocation method (DCM) for thin plate bending problems is proposed. This method takes advantage of computational graphs and backpropagation algorithms involved in deep learning. Besides, the proposed DCM is based on a feedforward deep neural network (DNN) and differs from most previous applications of deep learning for mechanical problems. First, batches of randomly distributed collocation points are initially generated inside the domain and along the boundaries. A loss function is built with the aim that the governing partial differential equations (PDEs) of Kirchhoff plate bending problems, and the boundary/initial conditions are minimised at those collocation points. A combination of optimizers is adopted to in the backpropagation process to minimize the loss function so as to obtain the optimal hyperparameters. In Kirchhoff plate bending problems, the C1 continuity requirement poses significant difficulties in traditional mesh-based methods. This can be solved by

*Corresponding authors:
Xiaoying Zhuang, +49 511 762-19589, Xiaoying.Zhuang@gmail.com

the proposed DCM, which uses a deep neural network to approximate the continuous transversal deflection, and is proved to be suitable to the bending analysis of Kirchhoff plate of various geometries.

Keywords: Deep learning, Collocation method, Kirchhoff plate, Higher-order PDEs.

1 Introduction

Thin plates are widely employed as basic structural components in engineering fields [1], which combines light weight, efficient load-carrying capacity, economy with technological effectiveness. Their mechanical behaviours have long been studied by various methods such as finite element method [2, 3], boundary element method [4, 5], meshfree method [6], isogeometric analysis [7], and numerical manifold method [8–10]. The Kirchhoff bending problem is a classical fourth-order problem, its mechanical behaviour is described by fourth-order partial differential equation as it is pretty difficult to construct a shape function to be globally C^1 continuous but piecewise C^2 continuous, namely, H^2 regular, for those mesh-based numerical method. However, according to the universal approximation theorem, see Cybenko [11] and Hornic [12], any continuous function can be approximated arbitrarily well by a feedforward neural network, even with a single hidden layer, which offers a new possibility of analysing Kirchhoff plate bending problems. We will first give a brief introduction of deep learning.

Deep learning was first brought up as a new branch of machine learning in the realm of artificial intelligence in 2006 [13], which uses deep neural networks to learn features of data with high-level of abstractions [14]. The deep neural networks adopt artificial neural network architectures with various hidden layers, which exponentially reduce the computational cost and amount of training data in some applications [15]. The major two desirable traits of deep learning lie in the nonlinear processing in multiple hidden layers in supervised or unsupervised learning [13]. Several types of deep neural networks such as convolutional neural networks (CNN) and recurrent/recursive neural networks (RNN) [16] have been created, which further boost the application of deep learning in image processing [17], object detection [18], speech recognition [19] and many other domains including genomics [20] and even finance [21].

As a matter of fact, artificial neural networks (ANN) which are main tools in deep learning have been around since the 1940's [22] but have not performed well until recently. They only become a major part of machine learning in the last several decades due to strides in computing techniques and explosive growth in date collection and availability, especially the arrival of backpropagation technique and advance in deep neural networks. However, based on the function approximation capabilities of feed forward neural networks, ANN were adopted to solving partial differential equations (PDEs) [23–25], which results in a solution that can be described by a closed analyt-

ical form. Basically, ANN methods can be suitable for solving PDEs in that they are smooth enough, solutions in analytical forms can be evaluated at arbitrary points in or outside the problem domain. Yadav et al. elaborately introduced the network methods for differential equations [26]. In the past, when neural networks with many hidden layers were tried to solve nonlinear PDEs in order to get a better results, it usually took a long time for training, which is due to a vanishing gradient problem. However, the proposal of pretraining, which sets the initial values of connection weights and biases, with the back propagation algorithm are now proposed to solve this problem efficiently. More recently, with improved theory incorporating unsupervised pre-training, stacks of auto-encoder variants, and deep belief nets, deep learning has become a central and popular hotspot in research and applications.

Also, some researchers studied the application of deep learning in solving PDEs. Mills et al. deployed a deep conventional neural network to solve Schrödinger equation, which directly learned the mapping between potential and energy [27]. E et al. applied deep learning-based numerical methods for high-dimensional parabolic PDEs and back-forward stochastic differential equations, which was proven to be efficient and accurate even for 100-dimensional nonlinear PDEs [28, 29]. Also, E and Yu proposed a Deep Ritz method for solving variational problems arising from partial differential equations [30]. Raissi et al. however solves PDEs in a different way and has made a series of contribution to this field. They first applied the probabilistic machine learning in solving linear and nonlinear differential equations using Gaussian Processes and later introduced a data-driven Numerical Gaussian Processes to solve time-dependent and nonlinear PDEs, which circumvented the need for spatial discretization [31–33]. Later, Raissi et al. [34–36] introduced a physical informed neural networks for supervised learning of nonlinear partial differential equations from Burger's equations to Navier-Stokes equations. Two distinct models were tailored for spatio-temporal datasets: continuous time and discrete time models. Raissi later employed a deep learning approach for discovering nonlinear PDEs from noisy observations in space and time with two deep neural networks, one for the representation of nonlinear-dynamic PDEs and one for a prior on the unknown solution [37]. Raissi applied a deep neural networks in solving coupled forward-backward stochastic differential equations and their corresponding high-dimensional PDEs [38]. Beck et al. [39, 40] studied the deep learning in solving stochastic differential equations and Kolmogorov equations, and validated the accuracy and speed proposed method, especially in high dimensions. Nabian and Meidani studied the presentation of high-dimensional random partial differential equations with a feed-forward fully-connected deep neural networks [41, 42]. Based on the physics informed deep neural networks, Tartakovsky et al. studied the estimation of parameters and unknown physics in PDE models [43]. Qin et al. applied the deep residual network and observation data to approximate unknown governing differential equations [44]. Sirignano and Spiliopoulos [45] gave a theoretic motivation of using deep neural networks as PDE approximators, which converges as the

3

number of hidden layers tend to infinity. Based on this, a deep Galerkin method was tested to solve PDEs including high-dimensional ones. Berg and Nyström [46] proposed a unified deep neural network approach to approximate solutions to PDEs and then used deep learning to discover PDEs hidden in complex data sets from measurement data [47]. In general, a deep feed-forward neural networks can well-sever as a suitable solution approximators, especially for high-dimensional PDEs with complex domains.

Meanwhile, some researchers study the surrogate of FEM by deep learning, which mainly trains the deep neural networks from datasets obtained from FEM. From work done by Liang et al. [48,49], a machine learning approach was first used to investigate the relationship between geometric features of aorta and FEM-predicted ascending aortic aneurysm rupture risk and then a deep learning was used to estimate the stress distribution of the aorta, which will be beneficial to real-time patient-specific computational simulations. Lee et al. introduced the background information involved in using deep learning for structural engineering [50]. Later, Wang et al. [51] applied deep learning in calculating U* index for the high efficient load paths analysis, with training data obtained from ANSYS results.

However, in this research, we will not confine deep learning application within FEM datasets. Rather, the deflection of Kirchhoff plate is first approximated with deep physical informed feedforward neural networks with hyperbolic tangent activation functions and trained by minimizing loss function related to governing equation of Kirchhoff bending problems and related boundary conditions. The training data for deep neural networks are obtained by randomly distributed collocation points from the physical domain of the plate. And clearly, this deep collocation method is a truly mesh-free method without the need of background grids. In this study, the method is established and applied to enrich deep learning with longstanding developments in engineering mechanics.

The paper is organised as follows: First a brief introduction of Kirchhoff plate bending strong form with typical boundary conditions is given. Then we introduce a basic knowledge of the deep learning technique and algorithms, which be helpful for later application. For numerical analysis, the deep collocation method with varying hidden layers and neurons are adopted for plates with various shapes, boundary and load conditions, hoping to manifest the favourable numerical features such as high accuracy and robustness of the proposed method.

2   Kirchhoff plate bending

Based on Kirchhoff plate bending theory [1], the relation between lateral deflection $w(x, y)$ of the middle surface $(z = 0)$ and rotations about the x,y-axis can be given
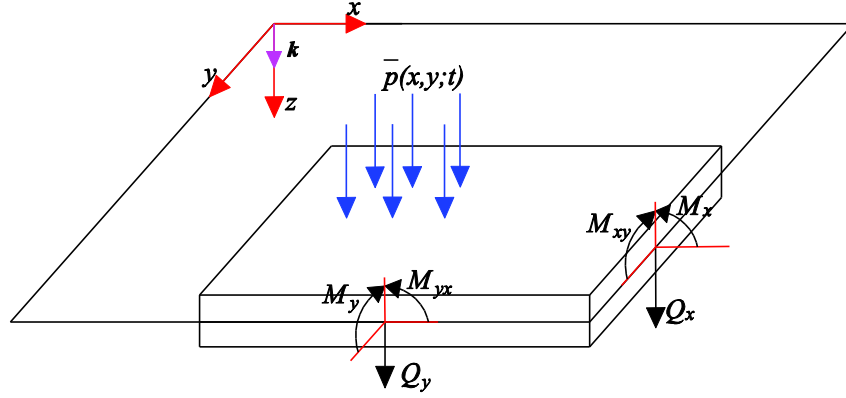
Figure 1: Kirchhoff plate in the coordinate system.

by

$$\theta_x = \frac{\partial w}{\partial x}, \qquad \theta_y = \frac{\partial w}{\partial y}. \tag{1}$$

Under the coordinate system shown in Figure 1, the displacement field in a thin plate can be expressed as:

$$
\begin{aligned}
u\left(x, y, z\right) &= -z\frac{\partial w}{\partial x}, \\
v\left(x, y, z\right) &= -z\frac{\partial w}{\partial y}, \\
w\left(x, y, z\right) &= w\left(x, y\right).
\end{aligned}
\tag{2}
$$

It is obviously that the transversal deflection of the middle plane of the thin plate can be regard as the field variables of the bending problem of thin plates. The corresponding bending and twist curvatures are the generalized strains:

$$k_x = -\frac{\partial^2 w}{\partial x^2}, \ k_y = -\frac{\partial^2 w}{\partial y^2}, \ k_{xy} = -2\frac{\partial^2 w}{\partial x \partial y}. \tag{3}$$

Therefore, the geometric equations of Kirchhoff bending can be expressed as:

$$
\boldsymbol{k} = \left\{ \begin{matrix} k_{xx} \\ k_{yy} \\ k_{xy} \end{matrix} \right\} = - \left\{ \begin{matrix} \frac{\partial^2 w}{\partial x^2} \\ \frac{\partial^2 w}{\partial y^2} \\ 2\frac{\partial^2 w}{\partial x \partial y} \end{matrix} \right\} = \boldsymbol{L}w, \tag{4}
$$

with L being the differential operator defined as $L = -\left(\frac{\partial^2}{\partial x^2} \quad \frac{\partial^2}{\partial y^2} \quad 2\frac{\partial^2}{\partial x \partial y}\right)^T$. According to the bending and twisting moments, shown in Figure 1 can be obtained as:

$$M_x = -D_0\left(\frac{\partial^2 w}{\partial x^2} + \nu\frac{\partial^2 w}{\partial y^2}\right),$$

$$M_y = -D_0\left(\frac{\partial^2 w}{\partial y^2} + \nu\frac{\partial^2 w}{\partial x^2}\right), \tag{5}$$

$$M_{xy} = M_{yx} = -D_0\left(1 - \nu\right)\frac{\partial^2 w}{\partial xy}.$$

Here $D_0 = \frac{Eh^3}{12(1-\nu^2)}$ is the bending rigidity, where E and ν are the Young's modulus and Poisson ratio, and h is the thickness of the thin plate. For isotropic thin plate, the constitutive equation can be expressed in Matrix form

$$M=Dk \tag{6}$$

with $D = D_0\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \left(1-\nu\right)/2 \end{bmatrix}$. The shear forces can be obtained in terms of the generalizsed stress components

$$Q_x = \frac{\partial M_x}{\partial x} + \frac{\partial M_{xy}}{\partial y}, \quad Q_y = \frac{\partial M_{xy}}{\partial x} + \frac{\partial M_y}{\partial y} \tag{7}$$

The differential equation for the deflections for thin plate based on Kirchhoff's assumptions can be expressed by transversal deflection as

$$\bigtriangledown^2\left(\bigtriangledown^2 w\right) = \bigtriangledown^4 w = \frac{p}{D} \tag{8}$$

where $\bigtriangledown^4\left(\right) = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4}$ is commonly called biharmonic operator.

Consequently, the Kirchhoff plate bending problems can be boiled down to a fourth order PDE problem, which pose difficulty for tradition mesh-based method in constructing a shape function to be H^2 regular. Moreover, the boundary conditions of Kirchhoff plate taken into consideration in this paper can be generally classified into three parts, namely,

$$\partial\Omega = \Gamma_1 + \Gamma_2 + \Gamma_3. \tag{9}$$

For clamped edge boundary, $\Gamma_1 : w = \tilde{w}, \frac{\partial w}{\partial n} = \tilde{\theta}_n, w = \tilde{w}, \tilde{\theta}_n$ are functions of arc length along this boundary.

For simply supported edge boundary, $\Gamma_2 : w = \tilde{w}, M_n = \tilde{M}_n, \tilde{M}_n$ is also a function of arc length along this boundary.

For free boundary conditions, $\Gamma_3 : M_n = \tilde{M}_n, \frac{\partial M_{ns}}{\partial s} + Q_n = \tilde{q}$, where $\tilde{q}$ is the load exerted along this boundary.

It should be noted that n, s here refer to the normal and tangent directions along the boundaries.

3 Deep Collocation Method for solving Kirchhoff plate bending

In this section, we will begin with introducing some preliminaries on deep learning, including the feed forward neural network architectures, some useful algorithms involved in deep learning. Then based on those basis, the formulation of deep collocation method is elucidated.

3.1 Feed forward neural network

The basic architecture of a fully connected feedforward neural network is shown in Figure 2, which comprises of multiple layers: input layer, one or more hidden layers and output layer. Each layer consists of one or more nodes called neurons, shown in the Figure 2 by small coloured circles, which is the basic unit of computation. For an interconnected structure, every two neurons in neighbouring layers have a connection, which is represented by a connection weight. Depicted in Figure 2, the weight between neuron k in hidden layer $l-1$ and neuron j in hidden layer l is denoted by w_{jk}^l. No connection exists among neurons in the same layer as well as in the non-neighbouring layers. Input data, defined from x_1 to x_N, flow through this neural network via connections between neurons, starting from input layer, through hidden layer $l-1$, l, to output layer, which eventually output data from y_1 to y_M. The feedforward neural work defines a mapping $FNN : \mathbb{R}^N \to \mathbb{R}^M$.

However, it should be noted that the number of neurons on each hidden layers and number of hidden layers can be any number and are invariably determined through a trial and error procedure. It has also been concluded that any continuous function can be approximated with any desired precision by a feed forward with even a single hidden layer [52, 53].

On each neuron in the feed forward neural network, a bias is supplied including neurons in the output layer except the neurons in the input layer, which is defined by b_j^l for bias of neuron j in layer l. Besides, the activation function is defined for output of each neuron in order to introduce a non-linearity into the neural network and make the back-propagation possible where gradients are supplied along with an error to update weights and biases. The activation function in layer l will be denoted by σ here. There are many activation functions can be used such as sigmoids function, hyperbolic tangent function $(Tanh)$, Rectified linear units $(Relu)$, and so on. Some suggestions upon the choice of activation function can be referred in [54]. Hence, for the value on each neuron in the hidden layers and output layer adds the weighted sum of values of output values from the previous layer with corresponding connection weights to basis on the neuron. A intermediate quantity for neuron j on hidden layer
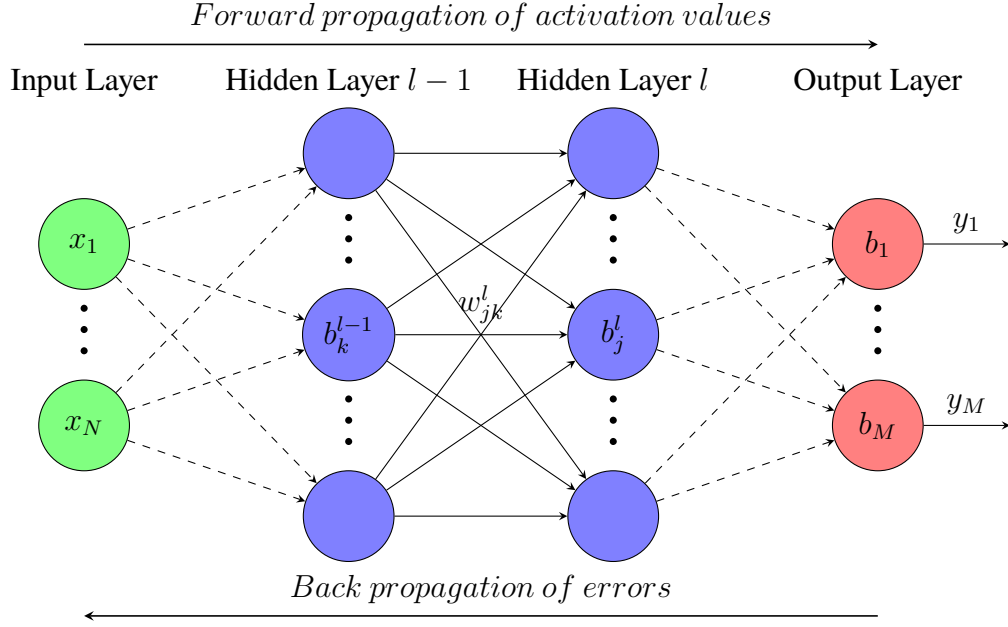
7

Figure 2: Architecture of a fully connected feedforward back-propagation neural network.

l is defined as

$$a_j^l = \sum_k w_{jk}^l y_k^{l-1} + b_j^l, \tag{10}$$

and its output is given by the activation of the above weighted input

$$y_j^l = \sigma\left(a_j^l\right) = \sigma\left(\sum_k w_{jk}^l y_k^{l-1} + b_j^l\right), \tag{11}$$

where y_k^{l-1} is the output from previous layer.

So, basically, when Equation 11 is applied to compute y_j^l, the intermediate quantity a_j^l was calculated along the way. This quantity turns out to be useful and named here as weighted input to neuron j on hidden layer l. Equation 10 can be written in a compact matrix form, which calculate weighted inputs for all neurons on certain layer efficiently, obtaining:

$$\boldsymbol{a} = \boldsymbol{W}^l \boldsymbol{y}^{l-1} + \boldsymbol{b}^l, \tag{12}$$

and accordingly, from Equation 12, $\boldsymbol{y}^l = \sigma(\boldsymbol{a})$, where activation function is applied elementwise. A feedforward network thus defines a function $f(\boldsymbol{x};\theta)$ depending on input data $\boldsymbol{x}$ and parametrised by θ consisting of weights and biases in each layer. The defined function provides an efficient way to approximate unknown field variables.

3.2 Backpropagation

Backpropagation ($backward\ propagation$) is an important and computationally effi-
cient mathematical tool to compute gradients in deep learning [55]. Essentially, back-
propagation is based on recursively applying the chain rule and decides which compu-
tations can be run in parallel from computational graphs. In our problem, the govern-
ing equation is the fourth order partial derivatives of field variable $w\,(\boldsymbol{x})$ approximated
by the deep neural networks $f\,(\boldsymbol{x};\theta)$, so this makes backpropagation a critical role. For
the approximation defined by $f\,(\boldsymbol{x};\theta)$, in order to find the weights and biases, a loss
function $\boldsymbol{L}\,(f,w)$ is defined to be minimised [56]. The backpropagation algorithm for
computing the gradient of loss function $\boldsymbol{L}\,(\boldsymbol{f},w)$ can be defined as follows [55]:

- **Input**: Input dataset $x^1,...,x^n$, prepare activation y^1 for input layer;

- **Feedforward**: For each layer $xl = 2,3,...,L$, compute $a^l = \sum_k W^l y^{l-1} + b^l$, and $\sigma\left(a^l\right)$;

- **Output error**: Compute the error $\delta^L = \nabla_{y^L}\boldsymbol{L} \odot \sigma'_L(a^L)$

- **Backpropagation error**: For each $l = L-1, L-2,...,2$, compute $\delta^l = \left((W^{l+1})^T \delta^{l+1}\right) \odot \sigma'_l(a^l)$;

- **Output**: The gradient of the loss function is given by $\frac{\partial \boldsymbol{L}}{\partial w^l_{jk}} = y^{l-1}_k \delta^l_j$ and $\frac{\partial \boldsymbol{L}}{\partial b^l_j} = \delta^l_j$.

Here, $\odot$ denotes the Hadamard product.

Now, there are a lists of deep learning frameworks for us to choose to setup a train-
ing. The main two approaches, Pytorch and Tensorflow, however computing deriva-
tives in the computational graphs distinctly. The former inputs a numerical value and
then compute the derivatives at this node, while the latter computers the derivatives
of a symbolic variable, then store the derivative operations into new nodes added
to the graph for later use. Obviously, the latter is more advantageous in computing
higher-order derivatives, which can be computed from its extended graph by running
backpropagation repeatedly. In this paper, since the fourth-order derivatives of field
variables is needed to be computed, the Tensorflow framework is thus adopted for
calculation [57].

3.3 Formulation of deep collocation method

The formulation of a deep collocation in solving Kirchhoff plate bending problems
is introduction in this section. Collocation method is a widely used method seeking
numerical solutions for ordinary, partial differential and integral equations [58]. It
is a popular method for trajectory optimization in control theory. A set of randomly
distributed points (also known as collocation points) is often deployed to represent a

desired trajectory that minimizes the loss function while satisfying a set of constraints. The collocation methods tend to be relatively insensitive to instability of system (such as blowing/vanishing gradients with neural networks), then it can be a viable way to train the deep neural networks in this paper [59].

Recalled form Section 2, Equation 8,9, the solving of Kirchhoff plate bending problems can be boiled down to the solving of a fourth order biharmonic equations with the type of boundary constraints. Thus we first discretize the physical domain with collocation points denoted by $\boldsymbol{x}_\Omega = (x_1, ..., x_{N_\Omega})^T$. Another set of collocation points are deployed to discretize boundary conditions denoted by $\boldsymbol{x}_\Gamma(x_1, ..., x_{N_\Gamma})^T$. Then the transversal deflection w is approximated with the aforementioned deep feed-forward neural network $w^h(\boldsymbol{x}; \theta)$. A loss function can thus be constructed to find the approximate solution by considering the minimizing of governing equation with boundary conditions approximated by $w^h(\boldsymbol{x}; \theta)$. The mean squared error loss form is adopted here.

Substituting $w^h(\boldsymbol{x}_\Omega; \theta)$ into Equation 8, we can get:

$$G(\boldsymbol{x}_\Omega; \theta) = \triangledown^4 w^h(\boldsymbol{x}_\Omega; \theta) - \frac{p}{D}, \tag{13}$$

which results in a physical informed deep neural network $G(\boldsymbol{x}_\Omega; \theta)$.

For boundary conditions illustrated in Section 2, considering all three boundaries, they can also be expressed by the neural network approximation $w^h(\boldsymbol{x}_\Gamma; \theta)$ as:
On Γ_1, we have

$$w^h(\boldsymbol{x}_{\Gamma_1}; \theta) = \tilde{w}, \ \frac{\partial w^h(\boldsymbol{x}_{\Gamma_1}; \theta)}{\partial n} = \tilde{\theta}_n. \tag{14}$$

On Γ_2,

$$w^h(\boldsymbol{x}_{\Gamma_2}; \theta) = \tilde{w}, \ \tilde{M}_n(\boldsymbol{x}_{\Gamma_2}; \theta) = \tilde{M}_n, \tag{15}$$

where $\tilde{M}_n(\boldsymbol{x}_{\Gamma_2}; \theta)$ can be obtained from Equation 5 by combing $w^h(\boldsymbol{x}_{\Gamma_2}; \theta)$.
On Γ_3,

$$M_n(\boldsymbol{x}_{\Gamma_3}; \theta) = \tilde{M}_n, \ \frac{\partial M_{ns}(\boldsymbol{x}_{\Gamma_3}; \theta)}{\partial s} + Q_n(\boldsymbol{x}_{\Gamma_3}; \theta) = \tilde{q}, \tag{16}$$

where $M_{ns}(\boldsymbol{x}_{\Gamma_3}; \theta)$ can be obtained from Equation 5 and $Q_n(\boldsymbol{x}_{\Gamma_3}; \theta)$ can be obtained from Equation 7 by combing $w^h(\boldsymbol{x}_{\Gamma_3}; \theta)$.

It should be noted that $\boldsymbol{n}, \boldsymbol{s}$ here refer to the normal and tangent directions along the boundaries. As induced physical informed neural network $G(\boldsymbol{x}; \theta)$, $M_n(\boldsymbol{x}; \theta)$, $M_{ns}(\boldsymbol{x}; \theta)$, $Q_n(\boldsymbol{x}; \theta)$ share the same parameters as $w^h(\boldsymbol{x}; \theta)$. Considering the generated collocation points in domain and on boundaries, they can all be learned by minimizing the mean square error loss function:

$$L(\theta) = MSE = MSE_G + MSE_{\Gamma_1} + MSE_{\Gamma_2} + MSE_{\Gamma_3}, \tag{17}$$

with

$$MSE_G = \frac{1}{N_d} \sum_{i=1}^{N_d} \left\| G\left(\boldsymbol{x}_{\,\Omega}; \theta\right) \right\|^2 = \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} \left\| \nabla^4 w^h\left(\boldsymbol{x}_{\,\Omega}; \theta\right) - \frac{p}{D} \right\|^2,$$

$$MSE_{\Gamma_1} = \frac{1}{N_{\Gamma_1}} \sum_{i=1}^{N_{\Gamma_1}} \left\| w^h\left(\boldsymbol{x}_{\,\Gamma_1}; \theta\right) - \tilde{w} \right\|^2 + \frac{1}{N_{\Gamma_1}} \sum_{i=1}^{N_{\Gamma_1}} \left\| \frac{\partial w^h\left(\boldsymbol{x}_{\,\Gamma_1}; \theta\right)}{\partial n} - \tilde{\theta}_n \right\|^2,$$

$$MSE_{\Gamma_2} = \frac{1}{N_{\Gamma_2}} \sum_{i=1}^{N_{\Gamma_2}} \left\| w^h\left(\boldsymbol{x}_{\,\Gamma_2}; \theta\right) - \tilde{w} \right\|^2 + \frac{1}{N_{\Gamma_2}} \sum_{i=1}^{N_{\Gamma_2}} \left\| \tilde{M}_n\left(\boldsymbol{x}_{\,\Gamma_2}; \theta\right) - \tilde{M}_n \right\|^2,$$

$$MSE_{\Gamma_3} = \frac{1}{N_{\Gamma_3}} \sum_{i=1}^{N_{\Gamma_3}} \left\| \tilde{M}_n\left(\boldsymbol{x}_{\,\Gamma_3}; \theta\right) - \tilde{M}_n \right\|^2 + \frac{1}{N_{\Gamma_3}} \sum_{i=1}^{N_{\Gamma_3}} \left\| \frac{\partial M_{ns}\left(\boldsymbol{x}_{\,\Gamma_3}; \theta\right)}{\partial s} + Q_n\left(\boldsymbol{x}_{\,\Gamma_3}; \theta\right) - \tilde{q} \right\|^2,$$

$$(18)$$

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

$$w^h = \underset{\theta \in R^K}{\arg\min}\, L\left(\theta\right) \tag{19}$$

Then, the solving of thin plate bending problems by deep collocation method can be reduced to an optimization problem. In deep learning Tensorflow/Pytorch framework, there are a variety available optimizers. One of the most widely used optimization method can be gradient descent based method is the Adam optimization algorithm [60], which is also adopted in the numerical study in this paper. Take a descent step at collocation point $\boldsymbol{x}_i$ with Adam-based learning rates α_i,

$$\theta_{i+1} = \theta_i + \alpha_i \nabla_\theta L\left(\boldsymbol{x}_i; \theta_i\right) \tag{20}$$

And then the process in Equation 20 is repeated until convergence criterion is satisfied.

4 Numerical examples

In this section, several numerical examples on plate bending problems with various shapes and boundary conditions is studied. And for implementation, a combined optimizer suggested by Berg et al. in [46] is adopted using L-BFGS optimizer [61] first and in linear search where BFGS may fail, a Adam optimizer is then applied with a very small learning rate. For all numerical examples, predicted maximum transverse with increasing layers are studied in order to show a convergence of deep collocation method in solving the plate bending problem.

4.1 Simply-supported square plate

A simply-supported square plate under a sinusoidal distribution of transverse loading is studied. The distributed load is given by

$$p = \frac{p_0}{D} \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right). \tag{21}$$

Here, a,b the length of the plate. D denotes the flexural stiffness of the plate and depends on the plate thickness and material properties. The exact solution for this problem is given by

$$w = \frac{p_0}{\pi^4 D\left(\frac{1}{a^2} + \frac{1}{b^2}\right)^2} \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right). \tag{22}$$

Here, w represents the transverse plate deflection. For this numerical example, we first generate 1000 randomly distributed collocation points in the physical domain depicted in Figure 3. And we thoroughly studied the influence of deep neural network with a varying number of hidden layer and neurons on the maximum deflection at the centre of the plate, which is then shown in Table 1. The numerical results are compared with the exact solution. It is clear that the results predicted by more hidden layers are more desirable, especially for neural networks with three hidden layers. To better reflect the deflection vector in the whole physical domain, the contour plot, contour error plot of deflection for increasing hidden layers with 50 neurons are shown in Figure 5, Figure 6, Figure 7
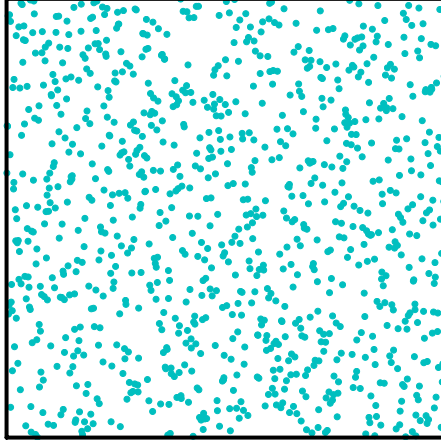

Figure 3: Collocation points discretize the square domain.

Table 1: Maximum deflection predicted by deep collocation method.

| Simply-supported Square Plate	Predicted Maximum Deflection	Exact Maximum deflection
1 hidden layer, 20 neurons	2.566529	
1 hidden layer, 30 neurons	2.566556	
1 hidden layer, 40 neurons	2.566541	
1 hidden layers, 50 neurons	2.566576	
1 hidden layers, 60 neurons	2.566509	
2 hidden layers, 20 neurons	2.566501	
2 hidden layers, 30 neurons	2.566429	
2 hidden layers, 40 neurons	2.566329	2.566496
2 hidden layers, 50 neurons	2.566442	
2 hidden layers, 60 neurons	2.566420	
3 hidden layers, 20 neurons	2.566366	
3 hidden layers, 30 neurons	2.566518	
3 hidden layers, 40 neurons	2.566434	
3 hidden layers, 50 neurons	2.566468	
3 hidden layers, 60 neurons	2.566549	
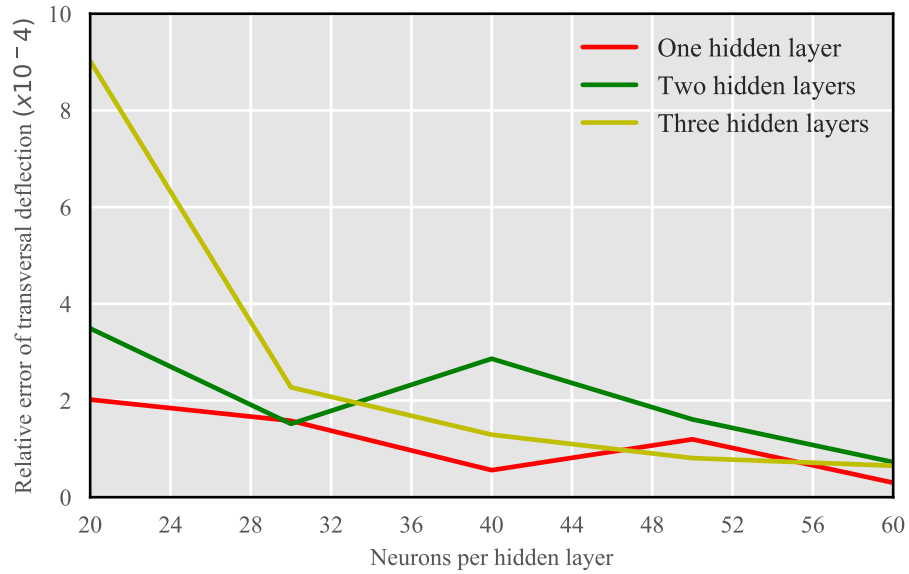
Figure 4: The relative error of deflection with varying hidden layers and neurons.

13

In Table 1, we employed a varying number of hidden layers from 1 to 4 and in each layer and the number of neurons varies from 20 to 60. We calculated the corresponding maximum transversal deflection at the centre of the square plate. From the L_2 relative error of deflection vector at all predicted points is shown in Figure 4 for each case. And it is very clear for even the neural network with only one single hidden layer with 20 neurons, the results is already very accurate and favourable. For most cases, with increasing neurons and hidden layers, the results converge to the exact solution and the results are very accurate even with a few neurons and a single hidden layer. In Figure 4, all three hidden layer types get very accurate results. Though the single layer with 20 neurons is the most accurate in all three types with 20 neurons, the magnitude of all is 1×10^{-4} and the other two results are also very accurate. And as the number of hidden layer and neurons increases, the relative error curves become flat and obtain results around exact solutions.

From Figure 5, Figure 6, Figure 7, we can observe that the deflection is accurately predicted by the deep collocation method, which agree well with the exact solutions. And as the hidden layer number increases, the numerical results converge to the exact solutions in the whole square plate. The predicted plate deformation agrees well with the exact deformation. All these lend some credence to the suitable application of this deep learning based method. The advantageous of neural networks with hidden layers is not conspicuously reflected in this numerical example, as the next numerical example shows more clearly.

(a) Predicted deflection contour

(b) Deflection error contour
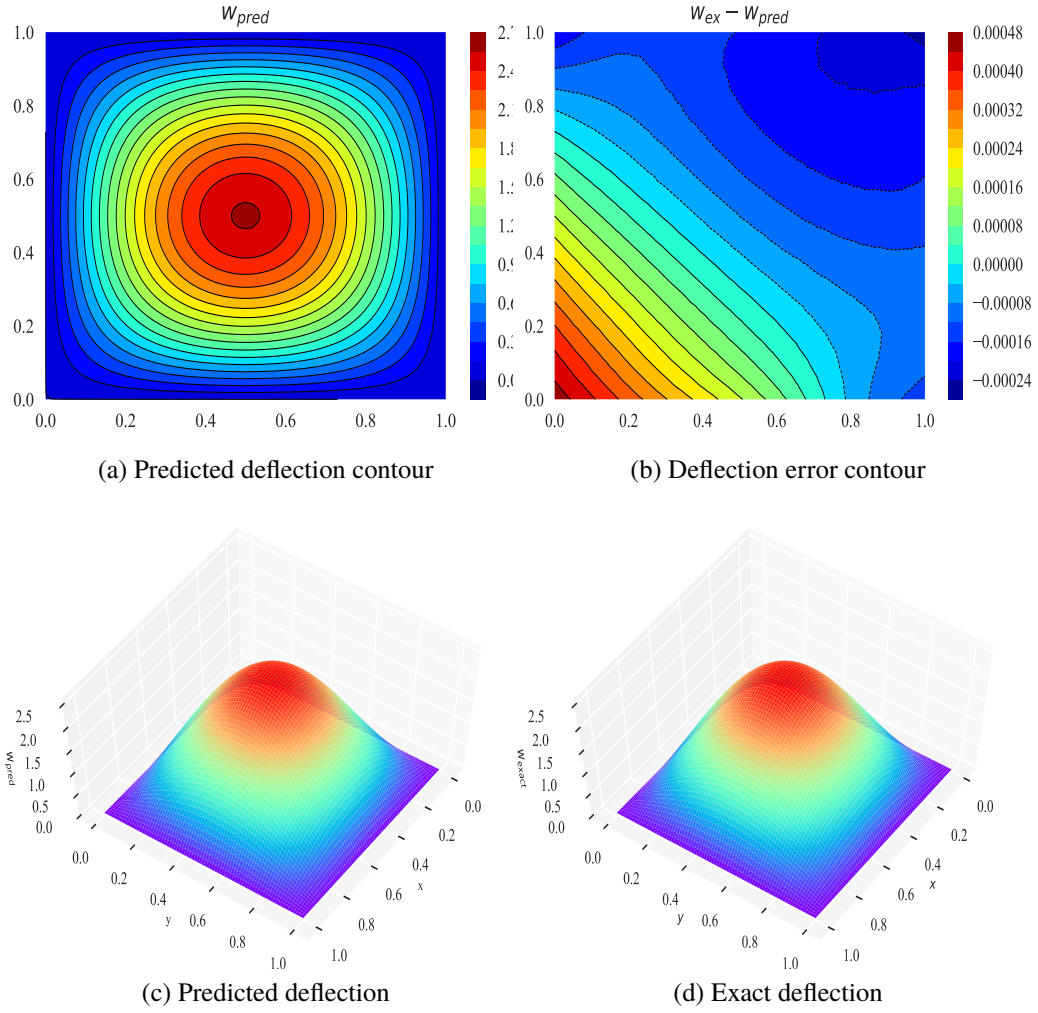
(c) Predicted deflection

(d) Exact deflection

Figure 5: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Exact deflection of the simply-supported square plate with 1 hidden layers and 50 neurons.
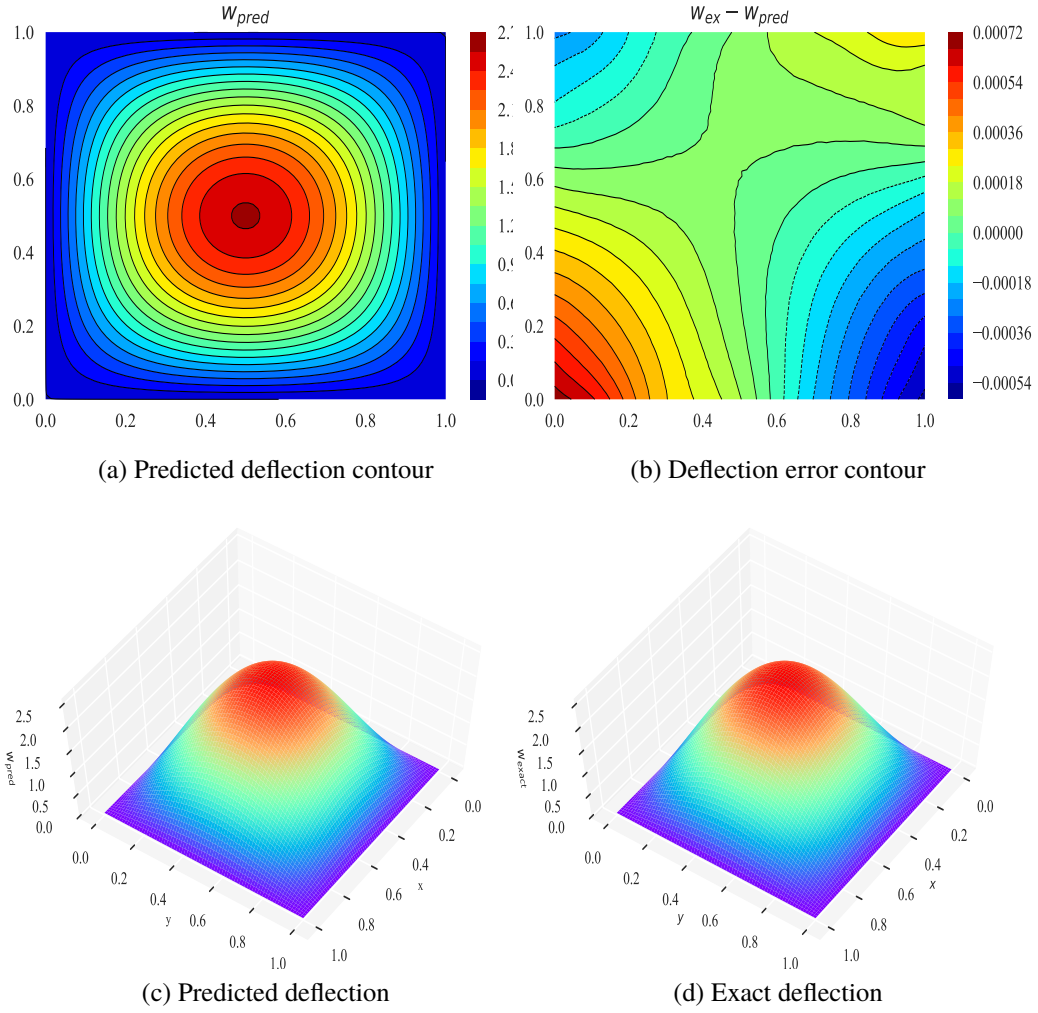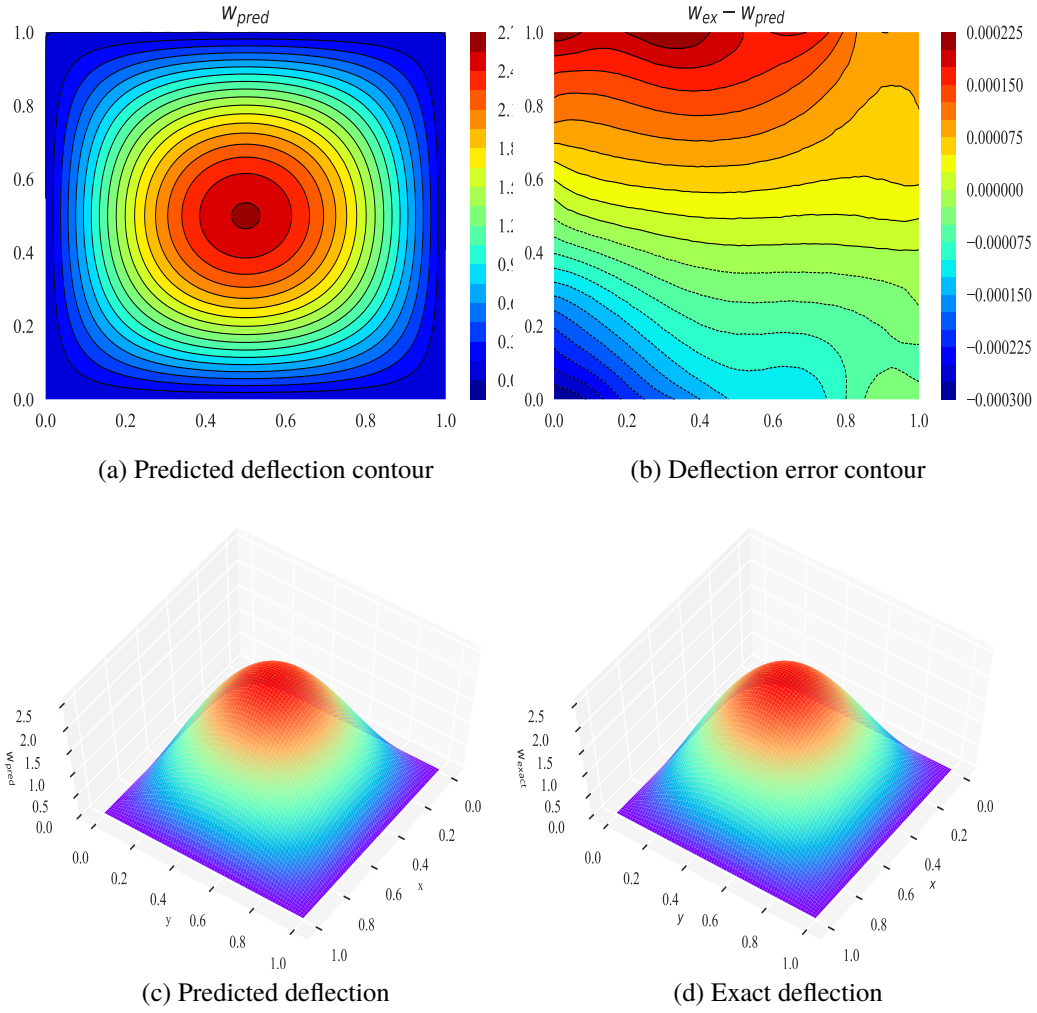
(a) Predicted deflection contour

(b) Deflection error contour

(c) Predicted deflection

(d) Exact deflection

Figure 6: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Exact deflection of the simply-supported square plate with 2 hidden layers and 50 neurons.

(a) Predicted deflection contour   (b) Deflection error contour

(c) Predicted deflection   (d) Exact deflection

Figure 7: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Exact deflection of the simply-supported square plate with 3 hidden layers and 50 neurons.

4.2   Clamped square plate

A clamped square plate under a uniformly distributed transverse loading is also analyzed with deep collocation method in this section. There is no available explicit form exact solution for deflection of among the whole plate. And to better illustration the accuracy of this method, the analytical solution obtained by the Galerkin method

referred in [62] is adopted as a comparison:

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} = \frac{b^4 p}{D} \begin{bmatrix} 0.318682766 \\ 0.038459815 \\ 0.038459815 \\ 0.008281438 \end{bmatrix}, \tag{23}$$

$$
\begin{aligned}
w &= \frac{b^4 q}{D} \left\{ a_{11} \left(1 - \frac{x}{a}\right)^2 \left(1 - \frac{y}{b}\right)^2 \left(\frac{x}{a}\right)^2 \left(\frac{y}{b}\right)^2 + a_{12} \left(1 - \frac{x}{a}\right)^2 \left(\frac{y}{b} - \frac{y^2}{b^2}\right)^2 \left(\frac{x}{a}\right)^2 \left(\frac{y}{b}\right)^2 \right\} + \\
&\quad \frac{b^4 q}{D} \left\{ a_{21} \left(\frac{x}{a} - \frac{x^2}{a^2}\right)^2 \left(1 - \frac{y}{b}\right)^2 \left(\frac{x}{a}\right)^2 \left(\frac{y}{b}\right)^2 + a_{22} \left(\frac{x}{a} - \frac{x^2}{a^2}\right)^2 \left(\frac{y}{b} - \frac{y^2}{b^2}\right)^2 \left(\frac{x}{a}\right)^2 \left(\frac{y}{b}\right)^2 \right\}
\end{aligned}
\tag{24}
$$

For the maximum transversal deflection at the centre of an isotropic square plate, Ritz method gives the maximum deflection at the centre as $w_{max} = 0.00133 \frac{qa^4}{D}$ [62],and Timoshenko and Krieger [63] gave a exact solution $w_{max} = 0.00126 \frac{qa^4}{D}$. Here, D denotes the flexural stiffness of the plate and depends on the plate thickness and material properties. a,b the length dimension of the plate. 1000 randomly generated collocation points as in Figure 3 are also used to discritize the clamped square plate here.

For this clamped case, a deep feedforward neural network with increasing layers and neurons is also studied in order to validate the convergence of this scheme. First, the maximum central deflection shown in Table 2 is also calculated for changing layers and neurons and are compared with aforementioned Ritz method, Galerkin method and exact solution by Timoshenko. It is demonstrated that our deep collocation method give most agreeable results with the exact solution. However, for neural networks with single hidden layer, the results are not that accurate even with 60 neurons. But as the neuron number increases, the results are indeed more accurate for the neural network with single hidden layer. This can be observed for the other two hidden layer types. Additionally, as the number of hidden layer increases, the results are much more accurate than the single hidden layer neural network results.

The relative error with the analytical solution with different hidden layers and different neurons is shown in Figure 8. Although the magnitude of relative error of deflection for this numerical example is 1×10^{-4}, this dose not mean that our deep collocation method is not that accurate for this problem. For it is mentioned that the deflection vector as a comparison to calculate the relative error is gained from Galerkin method, and we have gotten from Table 2, that our method gives more accurate maximum deflection than Galerkin method. As hidden layers increase, the two flat relative error curves nearly coincide and converge to the exact solution.

18

Table 2: Maximum deflection predicted by deep collocation method.

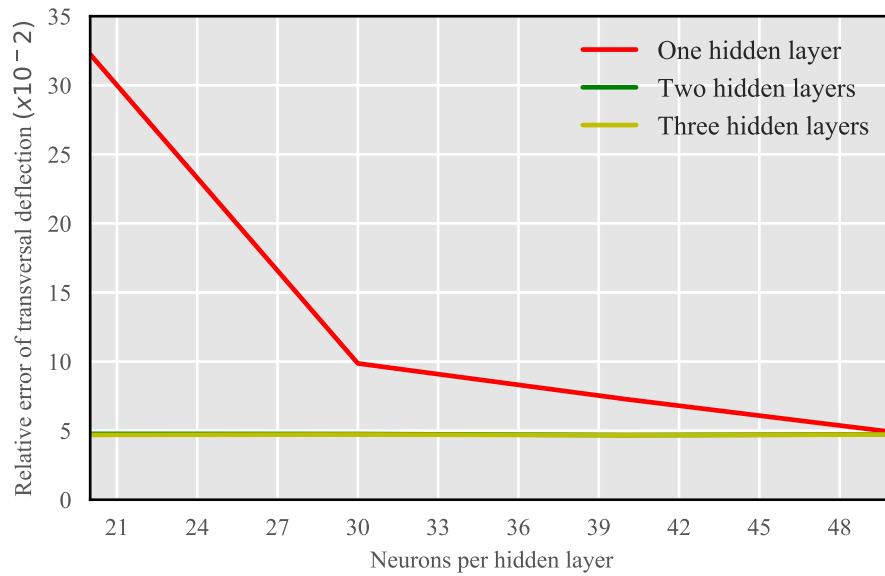| Clamped Square Plate	Predicted Maximum Deflection	Galerkin method	Ritz method	Exact solution
1 hidden layer, 20 neurons	0.860568			
1 hidden layer, 30 neurons	1.152175			
1 hidden layer, 40 neurons	1.195843			
1 hidden layers, 50 neurons	1.249870			
2 hidden layers, 20 neurons	1.257226			
2 hidden layers, 30 neurons	1.257759	1.321993	1.330000	1.260000
2 hidden layers, 40 neurons	1.265145			
2 hidden layers, 50 neurons	1.261974			
3 hidden layers, 20 neurons	1.261780			
3 hidden layers, 30 neurons	1.260374			
3 hidden layers, 40 neurons	1.261743			
3 hidden layers, 50 neurons	1.258893			

Figure 8: The relative error of deflection with varying hidden layers and neurons.

(a) Predicted deflection contour   (b) Deflection error contour

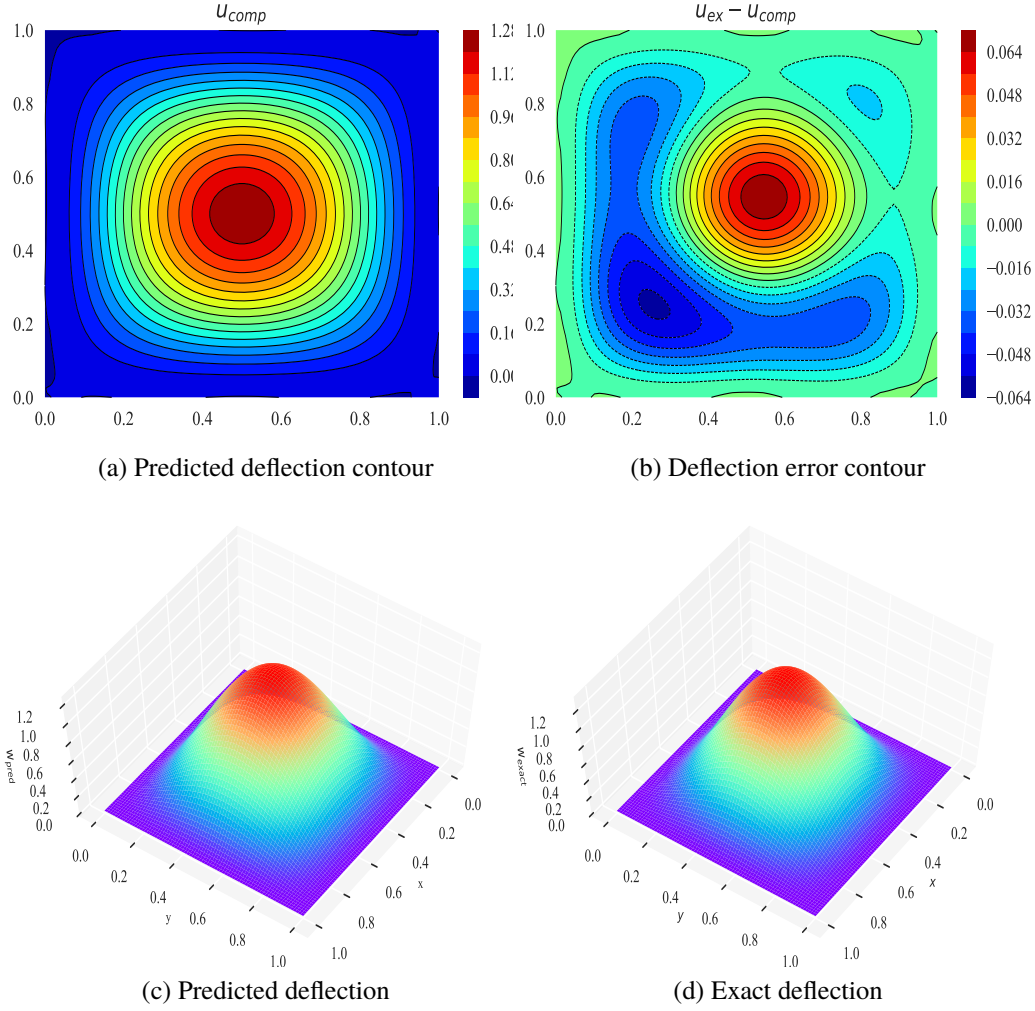(c) Predicted deflection   (d) Exact deflection

Figure 9: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Exact deflection of the clamped square plate with 3 hidden layers and 50 neurons.

Finally, to better depict the favourable of our method, the deflection contour, relative error contour and deformed deflection of the middle surface are also listed for the deep neural network with three layers and 50 neurons in Figure 9. It is clear that the deep collocation method yields results agrees well with the analytical solution.

20

4.3   Clamped circular plate

A clamped circular plate with radius R under a uniform load p is studied here. 1000 collocation points shown in Figure 10 are deployed among the circular plate first. Then, we applied deep collocation method to study the deformation of this circular plate. This problem has a exact solution, which can be referred in [63]:

$$w = \frac{p\left(R^2 - \left(x^2 + y^2\right)\right)^2}{64D}, \tag{25}$$

with D denotes the flexural stiffness of the plate and depends on the plate thickness and material properties.

The maximum deflection at the central of the circular plate with varying hidden layers and neurons in Table 3 and compared with exact solution. It is obvious that the predicted maximum deflection is very accurate, and as the neuron and hidden number increase, the maximum deflection are more and more close to the exact solution.

The relative error for deflection of clamped circular plate with increasing hidden layers and neurons is shown in Figure 11 in order to show the convergent of this method. From this figure, we can get that the as hidden layer number increases, the relative error curves become flat and converge very well to the exact solution. However, all neural networks perform well with a relative error magnitude of 1×10^{-4}.
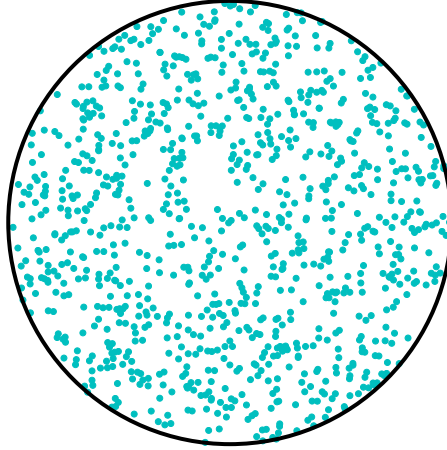

Figure 10: Collocation points discretize the circular domain.

Table 3: Maximum deflection predicted by deep collocation method.

Clamped Circular Plate	Predicted Maximum Deflection	Exact solution
1 hidden layer, 30 neurons	15.5958	
1 hidden layer, 40 neurons	15.5685	
1 hidden layer, 50 neurons	15.6201	
2 hidden layers, 30 neurons	15.6251	
2 hidden layers, 40 neurons	15.6264	15.6250
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	

Finally, the deformation contour, deflection error contour, predicted and exact deformation figure are displayed in Figure 12. The deflection of this circular plate agrees well with the exact solution. The accuracy of this collocation method is again shown here, which also illustrates that this deep collocation method can be easily and agreeably applied to simulate deformation of plates of various shapes.
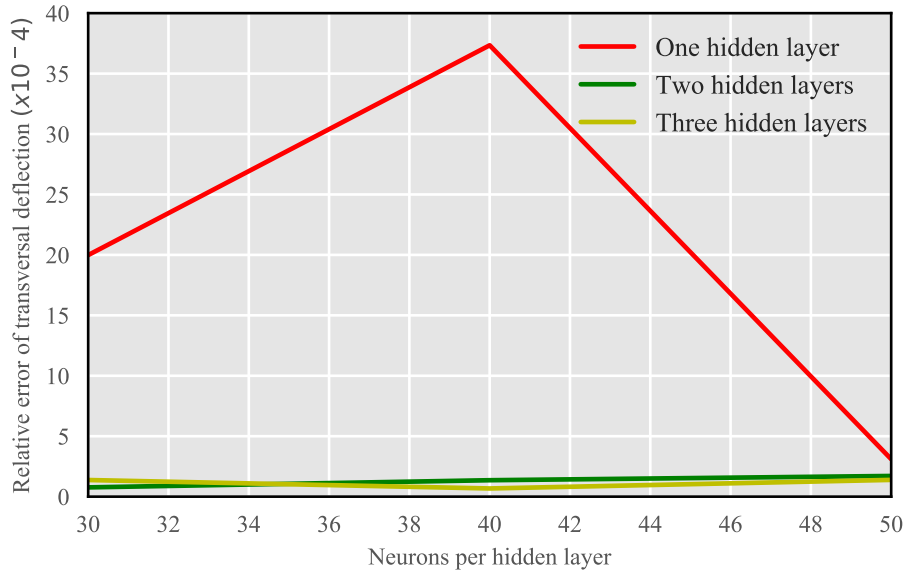

Figure 11: The relative error of deflection with varying hidden layers and neurons.
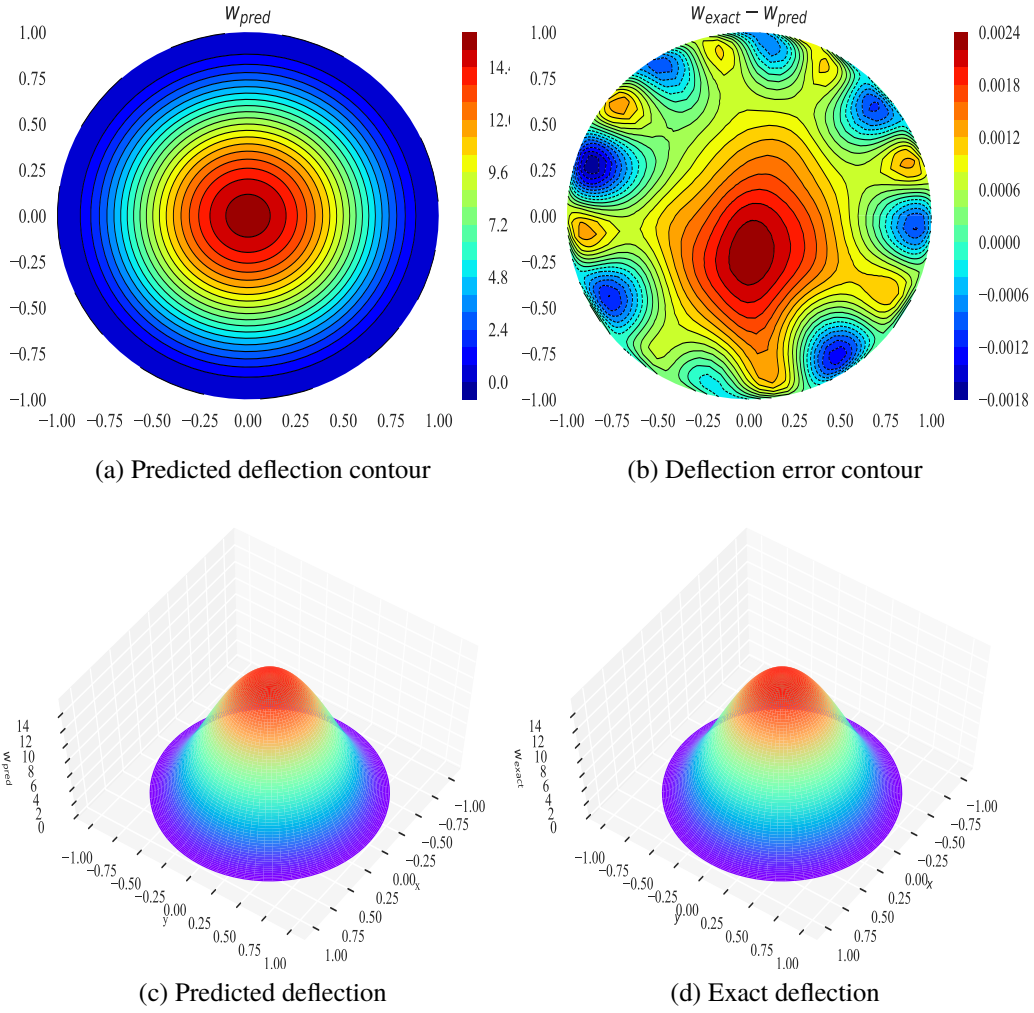
W_{pred}

$W_{exact} - W_{pred}$

(a) Predicted deflection contour

(b) Deflection error contour

(c) Predicted deflection

(d) Exact deflection

Figure 12: (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.

4.4 Simply-supported square plate on Winkler foundation

The simply-supported square plate resting on Winkler foundation is studied in this section, which assumes that the foundation's reaction $p(x, y)$ can be described by $p(x, y) = kw$, with k a constant called $foundation\ modulus$. Considering a plate

on a continuous Winkler foundation, the governing Equation 8 can be written as

$$\bigtriangledown^2 \left(\bigtriangledown^2 w\right) = \bigtriangledown^4 w = \frac{(p-q)}{D} = \frac{(p-kw)}{D} \tag{26}$$

The analytical solution for this numerical example is given as [63]:

$$w = \frac{16p}{ab} \sum_{m=1,3,5,\cdots}^{\infty} \sum_{n=1,3,5,\cdots}^{\infty} \frac{\sin\frac{m\pi x}{a}\sin\frac{n\pi y}{b}}{mn\left[\pi^4 D\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)^2 + k\right]} \tag{27}$$

For this numerical example, the arrangement of collocation points are the same as that in Figure 3. For the detail implementation, neural networks with different neurons and deepth are applied in the calculation. Also, maximum deflections shown in Table 4 at the central point in all those cases are first studied in order to unveil the accuracy of the deep collocation method.

Good agreement can be obseeved in this numerical example as well. From Table 4, we can obseeved as hidden layer and neuron number grows, the maximum deflection becomes more accurate and close to the analytical serial solution for even two hidden layers. The relative error shown in Figure 13 better depicts the advantages of deep neural network than shallow wide neural network. And with more hidden layers, with neurons increase, the relative error cure becomes flat and very close to zero, which shows that the deep collocation method with only two hidden layers can well approximate the deflection.

To better illustrate the deflection distribution around the whole plate, deflection contour, deflection error contour, deformation contour on deformed figure are shown in Figure 14 and compared with the analytical solution. It is demonstrated that the proposed method agrees well with the analytical solution.

Table 4: Maximum deflection predicted by deep collocation method.

| Square Plate on Winkler foundation	Predicted Maximum Deflection	Exact solution
1 hidden layer, 30 neurons	0.33999	
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	0.32137
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	
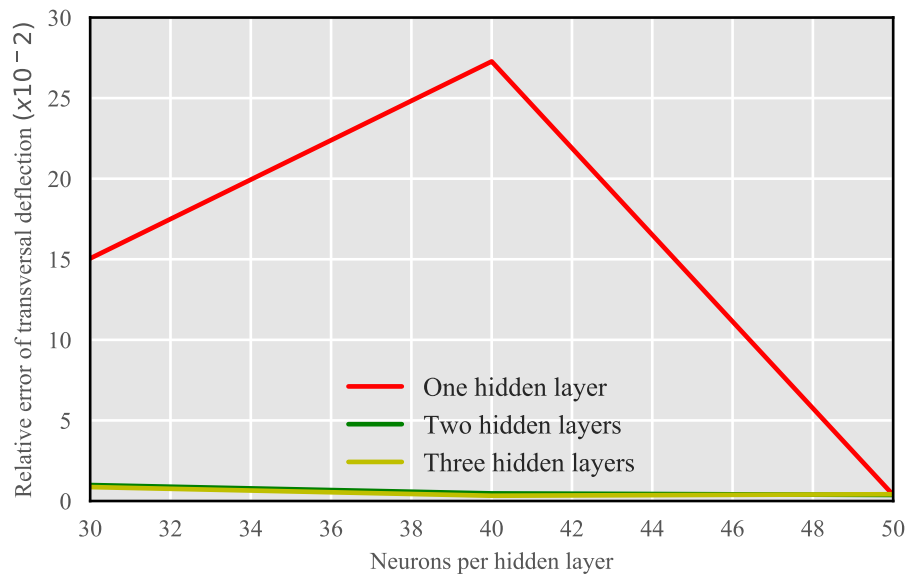
Figure 13: The relative error of deflection with varying hidden layers and neurons.

(a) Predicted deflection contour

(b) Deflection error contour

(c) Predicted deflection
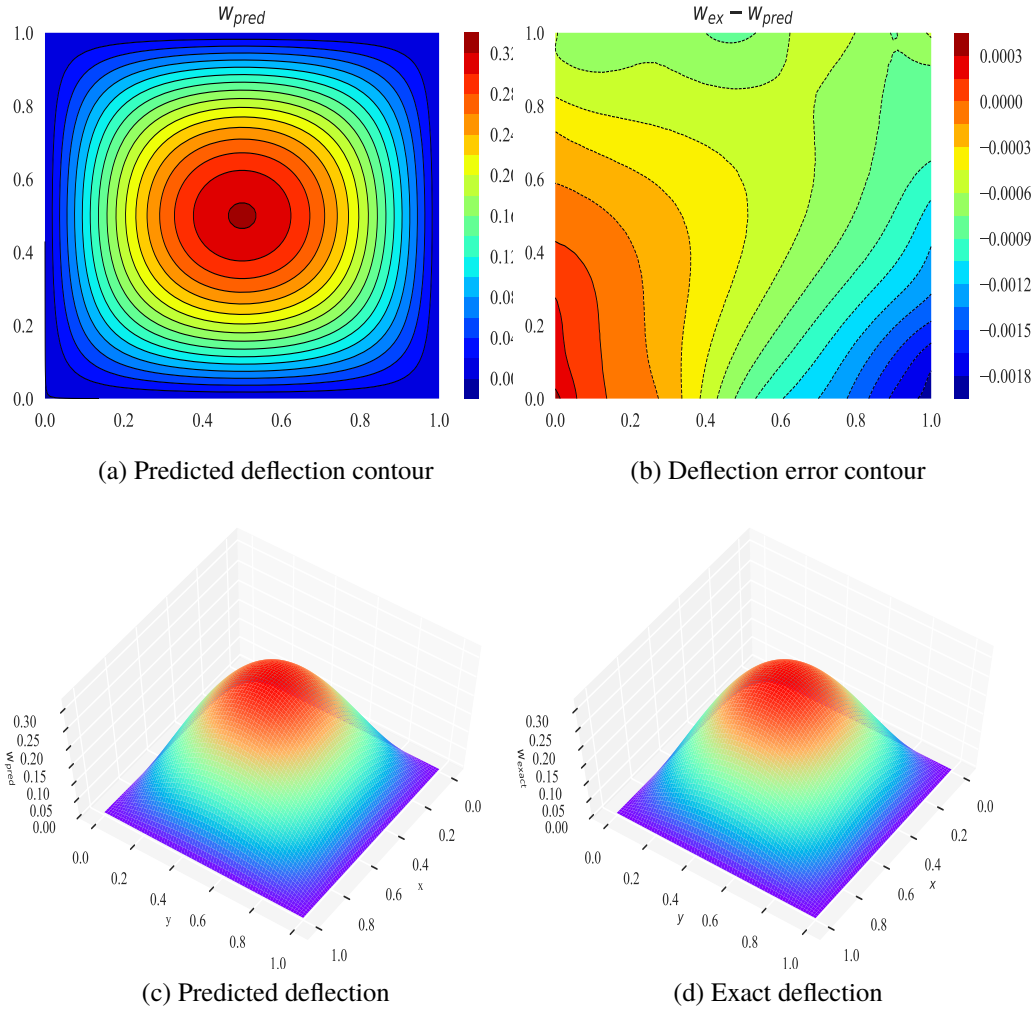
(d) Exact deflection

Figure 14: (a) Predicted deflection contour (b) Deflection error contour (c) Predicted deflection (d) Exact deflection of the simply-supported plate on Winkler foundation with 3 hidden layers and 50 neurons.

5   Conclusions

In this study,study the bending analysis of Kirchhoff plates of various shapes, loads and boundary conditions. The governing equation of this problem is a fourth order partial differential equation (biharmonic equation), which is an important kind of PDEs in engineering mechanics. The proposed deep collocation method is a truly "mesh-

free" method, and can be used to approximate any continuous function, which is very suitable for the analysis of thin plate bending problems. The deep collocation method is very simple in implementation, which can be further applied in a wide variety of engineering problems.

Moreover, the deep collocation method with randomly distributed collocations and deep neural networks perform very well with a MSE loss function minimized by the combined L-BFGS and Adam optimizer. An accurate result can even be gotten for the single layer and 20 neurons case. However, as the increase of hidden layers and neurons on each layer, most results become more accurate and converge to the exact and analytical solution. For circular plates, this method become extremely efficient and accurate, and accurate results can be obtained with only a few layers and neurons. More importantly, once those deep neural networks are trained, they can be used to evaluate the solution at any desired points with minimal additional computation time.

However, there are still some intriguing issues remained to be studied for the deep neural network based method such as the influence of choosing other neural network types, activation functions, loss function forms, weight/bias initialization, and optimizers on the accuracy and efficiency of this deep collocation method, which will be studied in our future research.

Acknoledgement:

References

[1] Eduard Ventsel and Theodor Krauthammer. *Thin plates and shells: theory: analysis, and applications*. CRC press, 2001.

[2] Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[3] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[4] John T Katsikadelis. *The boundary element method for engineers and scientists: theory and applications*. Academic Press, 2016.

[5] Carlos Alberto Brebbia and Stephen Walker. *Boundary element techniques in engineering*. Elsevier, 2016.

[6] Gui-Rong Liu. *Meshfree methods: moving beyond the finite element method*. CRC press, 2009.

[7] Vinh Phu Nguyen, Cosmin Anitescu, Stéphane PA Bordas, and Timon Rabczuk. Isogeometric analysis: an overview and computer implementation aspects. *Mathematics and Computers in Simulation*, 117:89–116, 2015.

[8] Hong Zheng, Zhijun Liu, and Xiurun Ge. Numerical manifold space of hermitian form and application to kirchhoff's thin plate problems. *International Journal for Numerical Methods in Engineering*, 95(9):721–739, 2013.

[9] Hongwei Guo and Hong Zheng. The linear analysis of thin shell problems using the numerical manifold method. *Thin-Walled Structures*, 124:366–383, 2018.

[10] Hongwei Guo, Hong Zheng, and Xiaoying Zhuang. Numerical manifold method for vibration analysis of kirchhoff's plates of arbitrary geometry. *Applied Mathematical Modelling*, 66:695–727, 2019.

[11] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[12] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[13] Rocio Vargas, Amir Mosavi, and Ramon Ruiz. Deep learning: A review. Oct 2018.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[15] Ali Al-Aradi, Adolfo Correia, Danilo Naiff, Gabriel Jardim, and Yuri Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning. *arXiv preprint arXiv:1811.08782*, 2018.

[16] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[17] Liping Yang, Alan MacEachren, Prasenjit Mitra, and Teresa Onorati. Visually-enabled active deep learning for (geo) text and image classification: a review. *ISPRS International Journal of Geo-Information*, 7(2):65, 2018.

[18] Zhong-Qiu Zhao, Peng Zheng, Shoutao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 2019.

[19] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: a systematic review. *IEEE Access*, 2019.

[20] Tianwei Yue and Haohan Wang. Deep learning for genomics: A concise overview. *arXiv preprint arXiv:1802.00810*, 2018.

[21] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[22] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[23] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[24] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neuralnetwork methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

[25] Kevin Stanley McFall and James Robert Mahan. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Transactions on Neural Networks*, 20(8):1221–1233, 2009.

[26] Neha Yadav, Anupam Yadav, Manoj Kumar, et al. *An introduction to neural network methods for differential equations*. Springer, 2015.

[27] Kyle Mills. Deep learning and the schrödinger equation. *Physical Review A*, 96(4), 2017.

[28] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, Nov 2017.

[29] +Jiequn Han, +Arnulf Jentzen, and +Weinan E. Solving high-dimensional partial differential equations using deep learning. *PNAS; Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

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

[31] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683 – 693, 2017.

[32] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125 – 141, 2018.

[33] M. Raissi, P. Perdikaris, and G. Karniadakis. Numerical gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, 2018.

[34] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. 11 2017.

[35] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. 11 2017.

[36] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019.

[37] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *J. Mach. Learn. Res.*, 19(1):932–955, January 2018.

[38] Maziar Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. 04 2018.

[39] Christian Beck, Sebastian Becker, Philipp Grohs, Nor Jaafari, and Arnulf Jentzen. Solving stochastic differential equations and kolmogorov equations by means of deep learning. 06 2018.

[40] Christian Beck, Weinan E, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, Jan 2019.

[41] Mohammad Amin Nabian and Hadi Meidani. A deep neural network surrogate for high-dimensional random partial differential equations. 06 2018.

[42] Mohammad Amin Nabian and Hadi Meidani. Physics-informed regularization of deep neural networks. 10 2018.

[43] Alexandre M. Tartakovsky, Carlos Ortiz Marrero, Paris Perdikaris, Guzel D. Tartakovsky, and David Barajas-Solano. Learning parameters and constitutive relationships with physics informed deep neural networks. 08 2018.

[44] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. 11 2018.

[45] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018.

[46] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28 – 41, 2018.

30

[47] Jens Berg and Kaj Nyström. Data-driven discovery of pdes in complex datasets. *Journal of Computational Physics*, 384:239 – 252, 2019.

[48] Liang Liang, Minliang Liu, Caitlin Martin, John A. Elefteriades, and Wei Sun. A machine learning approach to investigate the relationship between shape features and numerically predicted risk of ascending aortic aneurysm. *Biomechanics and Modeling in Mechanobiology*, 16(5):1519–1533, Apr 2017.

[49] Liang Liang, Minliang Liu, Caitlin Martin, and Wei Sun. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. *Journal of The Royal Society Interface*, 15(138):20170844, Jan 2018.

[50] Seunghye Lee, Jingwan Ha, Mehriniso Zokhirova, Hyeonjoon Moon, and Jaehong Lee. Background information of deep learning for structural engineering. *Archives of Computational Methods in Engineering*, 25(1):121–129, Jul 2017.

[51] Qingguo Wang, Geng Zhang, Chenchen Sun, and Nan Wu. High efficient load paths analysis with u* index generated by deep learning. *Computer Methods in Applied Mechanics and Engineering*, 344:499 – 511, 2019.

[52] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183 – 192, 1989.

[53] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[54] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the selection of initialization and activation function for deep neural networks. 05 2018.

[55] Michael A. Nielsen. Neural networks and deep learning, 2018.

[56] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *Schedae Informaticae*, 1/2016, 2017.

[57] Ali Al-Aradi, Adolfo Correia, Danilo Naiff, Gabriel Jardim, and Yuri Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning. 11 2018.

[58] Satya N Atluri. *Methods of computer modeling in engineering & the sciences*, volume 1. Tech Science Press Palmdale, 2005.

[59] Pulkit Agrawal. Collocation based approach for training recurrent neural networks.

[60] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[61] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, Aug 1989.

[62] Y. Khan, P. Tiwari, and R. Ali. Application of variational methods to a rectangular clamped plate problem. *Computers & Mathematics with Applications*, 63(4):862 – 869, 2012.

[63] Stephen P Timoshenko and Sergius Woinowsky-Krieger. *Theory of plates and shells*. McGraw-hill, 1959.