Analysis of three dimensional potential problems in non-homogeneous media with deep learning based collocation method

Hongwei Guo⁶,†, Xiaoying Zhuang⁶,⁎, Xiaoyu Meng⁶, Timon Rabczuk⁵,⁎

⁎Division of Computational Mechanics, Ton Duc Thang University, Ho Chi Minh City, Vietnam
†Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam
⑥Computational Science and Simulation Technology, Leibniz Universität Hannover, Hannover, Germany
⑦Department of Geotechnical Engineering, Tongji University, Shanghai, China.
⑧Key Laboratory of Geotechnical and Underground Engineering of Ministry of Education, Tongji University, Shanghai, China.

Abstract

A deep learning based collocation method is presented in this paper to solve the three dimensional potential problems in non-homogeneous media. Based on the universal approximation theorem, the neural network can be utilized to approximate solutions for different PDEs in different geometries. The performance of deep learning based method depends on the configurations of the network and other hyper-parameter settings. This makes the choice of neural network configurations extremely important. The configuration of this deep collocation method is setup by comparing different schemes of smooth activation functions, sampling methods for collocation points generation, combined optimizers. Besides, a convergence proof of this deep collocation method in solving non-homogeneous potential problems is performed. Then the deep collocation method is applied to the analysis of different material variations, and it can be concluded that the deep collocation method predicts the temperature and flux accurately for different material variations, especially the exponential material variations. As a result, the deep learning based collocation method shows a great potential in approximating solutions to PDEs.

Keywords: deep learning, collocation method, potential problem, PDEs, sampling method, activation function, non-homogeneous

1. Introduction

Recent years have witnessed the vast growing application of neural networks in physics, this is partly due to the fact that by training the neural network, high-dimensional raw data can be converted to low-dimensional codes [1], and thus the high-dimensional PDEs can be directly solved using a ‘meshfree’ deep learning algorithm, which improves computing efficiency and reduces the complexity of problems. The
deep learning method deploys a deep neural network architecture with nonlinear activation functions, which introduces the nonlinearity that the system as a whole needs for learning nonlinear patterns. This lends some credence to the application of a physics informed machine learning method in discovering the physical behind the potential problems in non-homogeneous media.

The problems of potential are a category of physical problems. For some physical parameters for potential problems, for example, heat conductivity, permeability, permittivity, resistivity, magnetic permeability, have a spatial distribution, and they can change in one or more coordinates. In order to solve these problems, we can translate the non-homogeneous problems into homogeneous problems in some classes of material variations. The steady state heat conduction analysis of FGMs analysis is a typical representative of this category of physical problem. By solving the heat conduction problems for FGMs, we can next better solve this category of physical problems. Due to the inherent mathematical difficulties, closed-form solutions exist in a few simple cases. Some traditional powerful methods, such as the finite element method(FEM) and the boundary element method(BEM), are used to solve the engineering problems by meshing [2]. In recent years the MFS-DRM approach has become popular, because it has lead to a mesh-free scheme. But only with a certain mathematical skill this approach can be achieved [3]. With the neural network the computational complexity and cost can be reduced. It can obtain the random sampling points by stochastic gradient descent, and then the neural network can be trained, in order to satisfy the differential operator, initial condition, and boundary conditions. In the process meshing is not required, and the PDE problem can be translated into a machine learning problem [4]. This method can be shown in this paper is very suitable for solving of these cases. And a series of numerical examples are tested to verify the implementation of this method.

Deep learning was proposed in 2006 [5, 6]. It is an unsupervised feature learning method with neural network architectures including multiple hidden layers with a lot of nodes [7]. Equipped with this hierarchical structure, it can extract information from complicated raw input data with multiple levels of abstraction through a layer-by-layer process [8]. Various variants such as multilayer perceptron (MLP), convolutional neural networks (CNN) and recurrent/recursive neural networks (RNN) [9] have been developed and applied to e.g. image processing [10, 11], object detection [12, 13], speech recognition [14, 15], biology [16, 17] and even finance [18, 19]. Deep learning can learn features from data automatically, and the features can be used to get the approximation of solutions to differential equations [20]. Over the past decade it has been widely used in physics applications due to high performance demonstrated.

Artificial neural networks (ANN) stands at the center of the deep learning revolution, it can be traced back to the 1940’s [21] but they became especially popular in the past few decades due to the vast development in computer science and computational science such as backpropagation technique and advances in deep neural networks. Due to the simplicity and feasibility of ANNs to deal with nonlinear and multi-dimensional problems, they were applied in inference and identification by data scientists [22]. They were also adopted to solve partial differential equations (PDEs) [23–25] but shallow ANNs are unable to learn the complex nonlinear patterns effectively. With improved theories incorporating unsupervised pre-training, stacks of auto-encoder variants, and deep belief nets, deep learning has become also an interesting alternative to classical methods such as FEM.

According to the universal approximation theorem [26, 27], any continuous function can be approximated by a feedforward neural network with one single hidden layer. However, the number of neurons of the hidden layer tends to increase exponentially with increasing complexity and non-linearity of a model. Recent studies show that DNNs render better approximations for nonlinear functions [28]. Some researchers employed deep learning for the solution of PDEs. E et al. developed a deep learning-based numerical method for high-dimensional parabolic PDEs and back-forward stochastic differential equations [29, 30]. Raissi et al. [31] introduced physics-informed neural networks for supervised learning of nonlinear partial differential equations. Beck et al. [32] employed deep learning to solve nonlinear stochastic differential equations and Kolmogorov equations. Sirignano and Spiliopoulos [3] provided a theoretical proof for deep neural networks as PDE approximators, and concluded that it converged as the number of hidden layers tend to infinity. Anitescu et al. [33], Guo et al. [34], and Nguyen-Thanh et al. [35] applied deep neural networks for finding the solutions for second and fourth order boundary value problems.

In practice the learning ability of deep neural networks can strongly rely on the using of the configurations
of the neural network and the optimization algorithms. Such as the activation functions, optimization algorithms, with the activation functions the neural networks can be applied to many nonlinear models, and optimization algorithms can update model parameters and minimize the value of the loss function. In this paper, we therefore compare the different parameters to select the optimal configuration of the neural network. In this context, we use deep learning method to get the optimal solution of the three dimensional problems.

The paper is organised as follows: First, we describe the three dimensional problems of potential inhomogeneous media. Then we introduce the deep learning based collocation method, which includes the neural network architecture, activation function, sampling method and a convergence proof. Subsequently, the results of numerical tests are presented. Finally, we demonstrate the effectiveness of the deep learning method for solving problems of potential in non-homogeneous media.

2. The governing equation for 3D problems of potential

The general differential equation for potential function \( \phi \) defined on a region \( \Omega \) bounded by surface \( \tau \), with an outward normal \( n \), can be written as:

\[
(k(x)\phi)_{,i} = k(x)\phi_{,ii} + k_i(x)\phi_{,i} = 0
\]

where \( k \) is a position-dependent material function. Equation (1) is the field equation for a wide range of problems in physics and engineering such as heat transfer, incompressible flow, gravity field, shaft torsion, electrostatics and magnetostatics, some of which are shown in Table 1 presenting a list of \( k(x) \) for different circumstance [36].

Table 1: Problems belong to the category of problems of potential

| Problems          | Scalar function \( \phi \) | \( k(x) \)                                   | Dirichlet | Neumann               |
|-------------------|-----------------------------|----------------------------------------------|-----------|-----------------------|
| Heat transfer     | Temperature \( T \)         | Thermal conductivity \( (k) \)              | \( T = \bar{T} \) | Heat flow \( q = -k\frac{dT}{dn} \) |
| Ground water flow | Hydraulic head \( H \)     | Permeability \( (k) \)                      | \( H = \bar{H} \) | Velocity flow \( q = -k\frac{dH}{dn} \) |
| Electrostatic flow| Field potential \( V \)    | Permittivity \( (\varepsilon) \)             | \( V = \bar{V} \) | Electric flow \( q = -k\frac{dV}{dn} \) |
| Electric conduction | Electropotential \( E \) | Resistivity \( (k) \)                       | \( E = \bar{E} \) | Electric current \( q = -k\frac{dE}{dn} \) |
| Magnetostatic     | Magnetic potential \( M \) | Magnetic permeabilit \( (\mu) \)            | \( M = \bar{M} \) | Magnetic flux density \( q = -k\frac{dM}{dn} \) |

The Dirichlet \( \tau_D \) and Neumann boundary \( \tau_N \) conditions are represented respectively as:

\[
\phi(x, t) = \bar{\phi}, x \in \tau_D,
q(x, t) = -k(x)\frac{\partial\phi(x, t)}{\partial n} = \bar{q}, x \in \tau_N
\]

where \( n \) is the unit outward normal to \( \tau_N \). Mixed boundary conditions are also frequently encountered: flux is prescribed over some portion of the boundary and potential is prescribed over the complementary portion of the boundary.

For non-homogeneous media, \( k(x) \) can be the function of coordinates. Functionally graded materials (FGMs) is one of those non-homogeneous media, which has been widely applied in practise. The composition and volume fraction of the FGM constituents vary gradually, giving a non-uniform microstructure with continuously graded macroproperties such as thermal conductivity, elasticity, hardness, etc. Generally, the spatially dependent material property \( k(x) \) can be generally expressed as: \( k(x) = k_0 f(x) \), where \( k_0 \) is a reference value for \( k \). \( f(x) \) is the material property variation function, and three typical material property variation functions (quadratic, exponential and trigonometric), namely:
Parabolic: \( f(x) = (a_1 + a_2x)^2 \)
Exponential: \( f(x) = (a_1e^{\beta x} + a_2e^{-\beta x})^2 \)  \( (3) \)
Trigonometric: \( f(x) = (a_1\cos\beta x + a_2\sin\beta x)^2 \)

Apparently, the solving of potential problems in non-homogeneous media can be boiled down to solving Equation (1) subjected to boundary conditions in Equation (3).
Thus the governing equations for different material variations and considering that the conductivity varies in z coordinate, obtaining:

\[
\begin{align*}
k_0(a_1 + a_2z)^2 & \quad (a_1 + a_2z)\nabla^2 \phi + 2a_2\phi_z = 0 \\
k_0(a_1e^{\beta z} + a_2e^{-\beta z})^2 & \quad (a_1e^{\beta z} + a_2e^{-\beta z})^2\nabla^2 \phi + 2\beta(a_1e^{2\beta z} + a_2e^{-2\beta z})\phi_z = 0 \\
k_0(a_1\cos\beta z + a_2\sin\beta z)^2 & \quad (a_1\cos\beta z + a_2\sin\beta z)^2\nabla^2 \phi + 2\beta(0.5(a_2^2 - a_1^2)\sin2\beta z + a_1a_2\cos2\beta z)\phi_z = 0
\end{align*}
\]

3. The deep learning based collocation method

3.1. Feed forward neural network

The basic architecture of a fully connected feedforward neural network is shown in Figure 1. It comprises multiple layers: input layer, one or more hidden layers and output layer. Each layer consists of one or more nodes called neurons, shown in Figure 1 by small colored circles, which is the basic unit of computation. For an interconnected structure, every two neurons in neighboring layers have a connection, which is represented by a connection weight, see Figure 1 where the weight between neuron \( k \) in hidden layer \( l-1 \) and neuron \( j \) in hidden layer \( l \) is denoted by \( w_{lk}^j \). No connection exists among neurons in the same layer as well as in the non-neighboring layers. Input data, defined from \( x_1 \) to \( x_N \), flows through this neural network via connections between neurons, starting from the input layer, through hidden layer \( l-1 \), \( l \), to the output layer, which eventually outputs data from \( y_1 \) to \( y_M \).

![Forward Propagation of Activation Values](image)

![Back Propagation of Errors](image)

Figure 1: Architecture of a fully connected feedforward back-propagation neural network.
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 $b^l_j$ for bias of neuron $j$ in layer $l$. Furthermore, the activation function is defined for an 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 $\sigma$ here.

There are many activation functions $\sigma$ proposed for inference and identification with neural networks such as sigmoid function \cite{37}, hyperbolic tangent function ($Tanh$) \cite{37}, Rectified linear units ($Relu$), to name a few. And some recent smooth activation functions, such as Swish \cite{38}, LeCun's Tanh \cite{37}, Bipolar sigmoid \cite{37}, Mish \cite{38}, Arctan \cite{39}, listed in Table 3 have been studied and compared in the numerical example section. All selected activation functions must be smooth enough in order to avoid gradient vanishing during backpropagation, since the governing equation is introduced in the loss which is the second order derivatives of the field variable.
| Activation function | Explicit function form | Function figure | Derivatives of function figure |
|---------------------|-----------------------|----------------|-----------------------------|
| Tanh                | $f(x) = \frac{e^{x} - 1}{e^{x} + 1}$ | ![Tanh Functions](image1) | ![Derivatives of Tanh Functions](image2) |
| Sigmoid             | $f(x) = \frac{1}{1 + e^{-x}}$ | ![Sigmoid Functions](image3) | ![Derivatives of Sigmoid Functions](image4) |
| Swish               | $f(x) = \frac{x}{1 + e^{-x}}$ | ![Swish Functions](image5) | ![Derivatives of Swish Functions](image6) |
| LeCuns Tanh         | $f(x) = 1.7159 \times \tanh(\frac{2}{3}x)$ | ![LeCuns Tanh Functions](image7) | ![Derivatives of LeCuns Tanh Functions](image8) |
| Bipolar sigmoid     | $f(x) = \frac{1}{\cosh(x)} - \frac{1}{\cosh(x)}$ | ![Bipolar sigmoid Functions](image9) | ![Derivatives of Bipolar sigmoid Functions](image10) |
| Mish                | $f(x) = x \times \tanh(\ln(1 + e^x))$ | ![Mish Functions](image11) | ![Derivatives of Mish Functions](image12) |
| Arctan              | $f(x) = \tan^{-1}(x)$ | ![Arctan Functions](image13) | ![Derivatives of Arctan Functions](image14) |
| Silu                | $f(x) = x \times \text{sigmoid}(x)$ | ![Silu Functions](image15) | ![Derivatives of Silu Functions](image16) |
Afterward, the value on each neuron in the hidden layers and output layer can be yielded by adding the weighted sum of values of output values from previous layer to basis. An intermediate quantity for neuron \( j \) on hidden layer \( l \) is defined as
\[
a_l^j = \sum_k w_{jk}^l y_{l-1}^k + b_l^j,
\]
and its output is given by the activation of the above weighted input
\[
y_l^j = \sigma(a_l^j) = \sigma\left(\sum_k w_{jk}^l y_{l-1}^k + b_l^j\right),
\]
where \( y_{l-1}^k \) is the output from previous layer.

When Equation (5) is applied to compute \( y_l^j \), the intermediate quantity \( a_l^j \) is 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 (4) can be written in a compact matrix form, which calculates the weighted inputs for all neurons on certain layers efficiently, obtaining:
\[
a = W^l y_{l-1} + b^l,
\]
Accordingly, from Equation (5), \( y = \sigma(a) \), where activation functions are applied elementwise.

Based on the previous derivation and description, we can draw a definition which will be used in Section 3.3:

**Definition 3.1.** (Feedforward Neural Network) A generalized neural networks with activation can be written in a tuple form \((f_1, \sigma_1), \ldots, (f_n, \sigma_n)\), with \( f_i \) an affine-line function \((f_i = W_i x + b_i)\) that maps \( R^{i-1} \to R^i \) and activation \( \sigma_i \) the mapping \( R^i \to R^i \), which in all defines a continuous bounded function mapping \( R^D \to R^n \):
\[
FNN : R^d \to R^n, \text{ with } F^n(x; \theta) = \sigma_n \circ f_n \circ \cdots \circ \sigma_1 \circ f_1
\]
where \( d \) the dimension of the inputs, \( n \) the number of field variables, \( \theta \) consisting of hyperparameters such as weights and biases and \( \circ \) denotes the element-wise operator.

The universal approximation theorem [26, 27] reveals that this continuous bounded function \( F \) with nonlinear activation \( \sigma \) can be adopted to capture the nonlinear property of the system, which is just the case of potential problems. With this definition, a theorem follows as [40]:

**Theorem 1.** If \( \sigma^i \in C^m(R^i) \) is non-constant and bounded, then \( F^n \) is uniformly m-dense in \( C^m(R^n) \).

### 3.2. Backpropagation

Backpropagation (backward propagation) comes in as a handy, useful mathematical tool, which can be used to train multilayer feed-forward networks by calculating the gradient of a loss function and finding the minimum value of the loss function. The backward (output-to-input) flow of computation to determine how much to adjust each weight is shown in Figure [2] which makes efficient reuse of intermediate values that were computed by the forward pass.
The chain rule of calculus is the basic of Backpropagation, and the chain rule is used to calculate the derivative of cost with regard to the weight in the network. With the chain rule we can recognize the contribution of each weight to overall error, so each weight will be updated and the error will be reduced, then We can produce good predictions.

In our problem, the governing equation contains partial derivatives of potential function $\phi(x)$ approximated by the deep neural networks $f(x; \theta)$. For the approximation defined by $f(x; \theta)$, in order to find the weights and biases, a loss function $L(f, w)$ is defined to be minimised. The backpropagation algorithm for computing the gradient of this loss function $L(f, w)$, the weight coefficients $w$ and thresholds of neurons $b$ can be calculated as follow.

- The output $Y$ of the $l$-th layer $j$-th neuron is calculated as:
  \[ Y^l_j = F(\sum w^l_{i,j} Y^{l-1}_i - b^l_j); \] (8)

- The output value of the $j$-th neuron of the output layer is calculated as:
  \[ Y_j = F(\sum w_{i,j} Y^{n-1}_i - b_j) \] (9)

- The error function $L$ of the network is:
  \[ L = \frac{1}{2} \sum (Y_j - d_j)^2 \] (10)

  Where $L$ is the error function, and $d$ is a target output of a neuron.

- The error $\varepsilon$ of the $j$-th neuron is defined as:
  \[ \varepsilon_j = Y_j - d_j \] (11)
• Error of the $j$-th element of the $l$-th hidden layer is:

$$
\varepsilon_j^k = \frac{\partial L}{\partial Y_j^l} = \sum_j \frac{\partial L}{\partial Y_j^l} \frac{\partial S_j^l}{\partial Y_j^l} w_{i,j}^l \quad (12)
$$

$$
\epsilon_j^l = \sum_j (Y_j - d_j) F'(S_j^l) w_{i,j}^l = \sum_j \varepsilon_j^l F'(S_j^l) w_{i,j}^l \quad (13)
$$

• The partial derivatives of error function by weight coefficients for hidden layers is:

$$
\frac{\partial L}{\partial w_{i,j}^l} = \sum_j \frac{\partial L}{\partial Y_j^l} \frac{\partial Y_j^l}{\partial S_j^l} \frac{\partial S_j^l}{\partial S_j^l} \frac{\partial S_j^l}{\partial Y_j^l} \frac{\partial Y_j^l}{\partial w_{i,j}^l} = \varepsilon_j^l F'(S_j^l) w_{i,j}^l \quad (14)
$$

• The partial derivatives of error function by weight coefficients for output layers is equal to:

$$
\frac{\partial L}{\partial w_{i,j}^l} = \frac{\partial L}{\partial Y_j^l} \frac{\partial Y_j^l}{\partial S_j^l} \frac{\partial S_j^l}{\partial S_j^l} \frac{\partial S_j^l}{\partial Y_j^l} \frac{\partial Y_j^l}{\partial w_{i,j}^l} = \varepsilon_j^l F'(S_j^l) Y_j^l \quad (15)
$$

• The partial derivatives of error function by thresholds is equal to:

$$
\frac{\partial L}{\partial b_j^l} = \frac{\partial L}{\partial Y_j^l} \frac{\partial Y_j^l}{\partial S_j^l} \frac{\partial S_j^l}{\partial S_j^l} \frac{\partial S_j^l}{\partial b_j^l} = \gamma_j^l F'(S_j^l) \quad (16)
$$

• Weight coefficients and thresholds of neurons are defined as:

$$
w_{i,j}^l(t + 1) = w_{i,j}^l(t) - \alpha \varepsilon_j^l F'(S_j^l) Y_j^l \quad (17)
$$

$$
b_{j}^l(t + 1) = b_{j}^l(t) - \alpha \varepsilon_j^l F'(S_j^l) \quad (18)
$$

where $\alpha$ is learning rate.

In this application, the whole model is built upon the tensorflow python library, which computes the derivatives of a symbolic variable, then stores the derivative operations into new nodes added to the graph for later use. Obviously, this is rather advantageous in computing partial derivatives, which can be computed from its extended graph by running backpropagation repeatedly.

### 3.3. Formulation of deep collocation method

We can retrieve from Section 2, Table 2 the governing equations for different material variations. Then, the formulation of a deep collocation in solving the three dimensional problems of potential is introduced in this section. Collocation method is a widely used method seeking numerical solutions for ordinary, partial differential and integral equations [41]. 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 method tends to be relatively insensitive to instabilities (such as blowing/vanishing gradients with neural networks) and is a viable way to train the deep neural networks.

Taking the parabolic material variation for example, the governing equation can be boiled down to the solution of a second order equations with boundary constraints. Thus we first discretize the physical domain and boundaries with collocation points denoted by $x_Ω = (x_1, ..., x_{N_Ω})^T$ and $x_Γ(x_1, ..., x_{N_Γ})^T$. There is a wide variety of sampling method available for analysis, it is thoroughly studied and compared in the numerical example section. Then the potential function $\phi$ is approximated with the aforementioned deep feedforward neural network $\phi^h(x; \theta)$. A loss function can thus be constructed to find the approximate solution by minimizing of governing equation with boundary conditions approximated by $\phi^h(x; \theta)$. 

9
Substituting $\phi^h(\mathbf{x}; \theta)$ into governing equation, we obtain
\[ G(\mathbf{x}; \theta) = k(\mathbf{x})\phi^h_{\theta_i}(\mathbf{x}; \theta) + k_i(\mathbf{x})\phi^h_{\theta_j}(\mathbf{x}; \theta), \] (19)
which results in a physical informed deep neural network $G(\mathbf{x}; \theta)$.

The boundary conditions illustrated in Section 2 can also be expressed by the neural network approximation $\phi^h(\mathbf{x}_\Gamma; \theta)$ as:

On $\Gamma_D$, we have
\[ \phi^h(\mathbf{x}_\Gamma; \theta) = \bar{\phi}, \] (20)
On $\Gamma_N$, 
\[ q^h(\mathbf{x}_\Gamma; \theta) = \bar{q}. \] (21)

where $\phi^h(\mathbf{x}_\Gamma; \theta)$ can be obtained from Equation (3) by combing $\phi^h(\mathbf{x}_\Gamma; \theta)$.

It should be noted that $\mathbf{n}$ here refer to the normal directions along the boundaries. Note the induced physical informed neural network $G(\mathbf{x}; \theta)$, $q(\mathbf{x}; \theta)$ share the same parameters as $\phi^h(\mathbf{x}; \theta)$. Considering the generated collocation points in domain and on boundaries, they can all be learned by minimizing the mean square error loss function [42]:
\[ L(\theta) = MSE = MSE_G + MSE_{\Gamma_D} + MSE_{\Gamma_N}, \] (22)
with
\[ MSE_G = \frac{1}{N_d} \sum_{i=1}^{N_d} \left\| G(\mathbf{x}_i; \theta) \right\|^2, \]
\[ MSE_{\Gamma_D} = \frac{1}{N_{\Gamma_D}} \sum_{i=1}^{N_{\Gamma_D}} \left\| \phi^h(\mathbf{x}_\Gamma; \theta) - \bar{\phi} \right\|^2, \] (23)
\[ MSE_{\Gamma_N} = \frac{1}{N_{\Gamma_N}} \sum_{i=1}^{N_{\Gamma_N}} \left\| q(\mathbf{x}_\Gamma; \theta) - \bar{q} \right\|^2 = \frac{1}{N_{\Gamma_N}} \sum_{i=1}^{N_{\Gamma_N}} \left\| -k(\mathbf{x}_\Gamma) \frac{\partial \phi^h(\mathbf{x}_\Gamma; \theta)}{\partial n} - \bar{q} \right\|^2. \]

where $\mathbf{x}_\Omega \in \mathbb{R}^N$, $\theta \in \mathbb{R}^K$ are the neural network parameters. $L(\theta) = 0$, $\phi^h(\mathbf{x}; \theta)$ is then a solution to potential function. Here, the defined loss function measures how well the approximation satisfies the physics law (governing equation), boundary conditions. Our goal is to find a set of parameters $\theta$ that the approximated potential $\phi^h(\mathbf{x}; \theta)$ minimizes the loss $L$. If $L$ is a very small value, the approximation $\phi^h(\mathbf{x}; \theta)$ is very closely satisfying governing equations and boundary conditions, namely
\[ \phi^h = \arg \min_{\theta \in \mathbb{R}^K} L(\theta). \] (24)

The solution of heat conduction problems by deep collocation method can be reduced to an optimization problem. In the deep learning Tensorflow framework, a variety of optimizers are available. One of the most widely used optimization methods is the Adam optimization algorithm, which is also adopted in the numerical study in this paper. The idea is to take a descent step at collocation point $\mathbf{x}_i$ with Adam-based learning rates $\alpha_i$,
\[ \theta_{i+1} = \theta_i + \alpha_i \nabla_{\theta} L(\mathbf{x}_i; \theta_i) \] (25)
and then the process in Equation (25) is repeated until a convergence criterion is satisfied.

3.4. Convergence of deep collocation method for potential problems

With the universal approximation theorem of neural networks, a feedforward neural network is used to approximate the potential function as $\phi^h(\mathbf{x}; \theta)$. Further, the approximation ability of neural networks for the potential problems needs to be proved. The approximation power of neural networks for a quasilinear
parabolic PDEs has been proved by Sirignano et al. [4]. For potential problems, whose governing equation is an elliptic partial equation, the proof can be boiled down to:

\[ \exists \phi^h \in F^n, \text{ s.t. as } n \to \infty, \ L(\theta) \to 0, \ \phi^h \to \phi \]  

(26)

potential problem has a unique solution, s.t. \( \phi \in C^2(\Omega) \) with its derivatives uniformly bounded. Also, the conductivity function \( k(x) \) is assumed to be \( C^{1,1} \) (\( C^1 \) with Lipschitz continuous derivative).

**Theorem 2.** With assumption that \( \Omega \) is compact and considering measures \( \ell_1, \ell_2, \) and \( \ell_3 \) whose supports are constrained in \( \Omega, \Gamma_D, \) and \( \Gamma_N. \) Also, the governing Equation (1) subject to 2 is assumed to have a unique classical solution and conductivity function \( k(x) \) is assumed to be \( C^{1,1} \) (\( C^1 \) with Lipschitz continuous derivative). Then, \( \forall \ v > 0, \exists \ K > 0, \) which may depend on \( \sup\|\phi_i\| \) and \( \sup\|\phi_i\|, \) s.t. \( \exists \ \phi^h \in F^n, \) that satisfies \( L(\theta) \leq K \varepsilon \)

**Proof.** For governing Equation (1) subject to 2 according to Theorem 1 \( \forall \ v > 0, \exists \ \phi^h \in F^n, \) s.t.

\[ \sup_{x \in \Omega} \|\phi_{\cdot i} (x_\Omega) - \phi_{\cdot i}^h (x_\Omega)\|^2 + \sup_{x \in \Omega} \|\phi_{\cdot ii} (x_\Omega) - \phi_{\cdot ii}^h (x_\Omega)\|^2 \leq \varepsilon \]  

(27)

Recalling that the Loss is constructed in the form shown in Equation (22), for \( MSE_G, \) applying triangle inequality, and obtains:

\[ \|G(x_\Omega; \theta)\|^2 \leq \|k(x_\Omega)\phi_{\cdot i}^h (x_\Omega; \theta)\|^2 + \|k_{\cdot i}(x_\Omega)\phi_{\cdot i}^h (x_\Omega; \theta)\|^2 \]  

(28)

Also, considering the \( C^{1,1} \) conductivity function \( k(x), \exists \ M_1 > 0, \ M_2 > 0, \exists \ x \in \Omega, \|k(x)\| \leq M_1, \|k_{\cdot i}(x)\| \leq M_2. \) From Equation (27), it can be obtained that:

\[ \int_{\Omega} k_{\cdot i}^2(x_\Omega) (\phi_{\cdot i}^h - \phi_{\cdot i})^2 \, d\ell_1 \leq M_2^2 \varepsilon^2 \ell_1(\Omega) \]

\[ \int_{\Omega} k_{\cdot i}^2(x_\Omega) (\phi_{\cdot ii}^h - \phi_{\cdot ii})^2 \, d\ell_1 \leq M_1^2 \varepsilon^2 \ell_1(\Omega) \]  

(29)

On boundaries \( \Gamma_D \) and \( \Gamma_N, \)

\[ \int_{\Gamma_D} (k^h (x_{\Gamma_D}; \theta) - \phi (x_{\Gamma_D}; \theta))^2 \, d\ell_2 \leq \varepsilon^2 \ell_2(\Gamma_D) \]

\[ \int_{\Gamma_N} k_{\cdot i}^2(x_{\Gamma_N}) (\phi_{\cdot i}^h (x_{\Gamma_N}; \theta) - \phi_{\cdot i}^h (x_{\Gamma_N}; \theta))^2 \, d\ell_3 \leq M_1^2 \varepsilon^2 \ell_3(\Gamma_N) \]  

(30)

Therefore, using Equations (29 and 30) as \( n \to \infty \)

\[ L(\theta) = \frac{1}{N_D} \sum_{i=1}^{N_D} \|k(x_{\Omega})\phi_{\cdot i}^h (x_{\Omega}; \theta) + k_{\cdot i}(x_{\Omega})\phi_{\cdot i}^h (x_{\Omega}; \theta)\|^2 + \]

\[ \frac{1}{N_D} \sum_{i=1}^{N_D} \|\phi_{\cdot i}^h (x_{\Gamma_D}; \theta) - \phi_{\cdot i}^h \| + \frac{1}{N_D} \sum_{i=1}^{N_D} \|\phi_{\cdot ii}^h (x_{\Gamma_D}; \theta) - \phi_{\cdot ii}^h \|^2 \]

\[ \leq \frac{1}{N_D} \sum_{i=1}^{N_D} \|\phi_{\cdot i}^h (x_{\Omega}; \theta)\|^2 + \frac{1}{N_D} \sum_{i=1}^{N_D} \|\phi_{\cdot ii}^h (x_{\Omega}; \theta)\|^2 \]

\[ \leq (M_2^2 + M_1^2) \varepsilon^2 \ell_1(\Omega) + \varepsilon^2 \ell_2(\Gamma_D) + M_1^2 \varepsilon^2 \ell_3(\Gamma_N) = K \varepsilon \]  

Finally, the proof is completed.

\( \square \)
With the hold of Theorem \[2\] and conditions that \( \Omega \) is a bounded open subset of \( \mathbb{R} \), \( \forall n \in N_+ \), \( \phi^h \in F^n \in L^2(\Omega) \), it can be concluded from Sirignano et al. \[4\] that:

**Theorem 3.** \( \forall p < 2 \), \( \phi^h \in F^n \) converges to \( \phi \) strongly in \( L^p(\Omega) \) as \( n \to \infty \) with \( \phi \) being the unique solution to the potential problems.

To summary, for feedforward neural networks \( F^n \in L^p \) space (\( p < 2 \)), the approximated solution \( \phi^h \in F^n \) will converge to the solution to this PDE. This will justify the application of physics-informed and data-driven deep learning method in solving potential problems.

### 3.5. Collocation points generation

Training the model is an important process in machine learning algorithms. The deep collocation method utilizes the neural networks to solving PDEs with randomly generated training points in the physical domain. The optimal training points will generate more stable and accurate results. There are many sampling methods to generate collocation points for deep collocation method. The Halton and Hammersley sequences get the points by a construction of the radical inverse \[43\], and they are both low discrepancy sequences, the difference is that one dimension of Hammersley sequence is replaced with a regular grid. The method of Korobov Lattice creates samples from a Korobov lattice point sets \[44\]. Sobol Sequenc is a quasi-random low-discrepancy sequences to generate points \[45\]. Two closely related low-discrepancy point sets. Latin hypercube sampling (LHS) is a statistical method, and a near-random sample of parameter values is generated from a multidimensional distribution \[46\]. Monte Carlo methods can create points by using the process of repeated random sampling \[47\]. For Random sampling the probability of each point being selected in a cubic is equal \[48\]. The distribution of points in a cubic by the sampling methods will be shown in Table \[4\].
4. Numerical examples

In this section, several numerical tests are studied to give guidance to the further application of this method, such as a suitable configurations of neural networks, sampling method, optimizers for training. Also, different material variations in one or more coordinates of this inhomogeneous media are studied to unveil the patterns of potential function with deep neural networks. The relative error between the predicted potential function and the analytical solution for different neural network configurations are computed as the evaluation metric:

\[ e = \frac{\|E_{\text{pred}}\| - \|E_a\|}{\|E_a\|} \]  \hspace{1cm} (32)

where \( \|E_a\| \) is the analytical solution and \( \|E_{\text{pred}}\| \) is the predicted solution.

Table 4: Sampling method

| Sampling method          | points figure | Sampling method          | points figure |
|--------------------------|---------------|--------------------------|---------------|
| Latin hypercube          | ![Points Figure](image1.png) | Monte Carlo              | ![Points Figure](image2.png) |
| Random                   | ![Points Figure](image3.png) | Halton Sequences         | ![Points Figure](image4.png) |
| Hammersley Sequence      | ![Points Figure](image5.png) | Korobov Lattice          | ![Points Figure](image6.png) |
| Sobol Sequence           | ![Points Figure](image7.png) |                          |               |

The sampling methods are used to generate sequence of points within a cube. The purpose of sampling method is to make the network better trained and proper sampling will help to avoid that the neural networks are only trained on fixed points, and it has a better prediction for the new data. All sampling methods introduced here are studied and compared in the numerical example and hoping to give some insight for later application.
4.1. Case 1. Cube with material gradation along the z-axis

A unit cube (L=1) with prescribed constant temperature on two sides is considered. The top surface of the cube at z=1 is maintained at a temperature of T = 100 while the bottom at z=0 is zero. The remaining four faces are insulated (zero normal flux). Three different classes of variations shown in Table 5 are considered. The profiles of the thermal conductivity k(z) of the three cases are illustrated in Figure 3 and the boundary conditions of the unit cube are illustrated in Figure 4. Moreover, for each thermal conductivity, the analytical solution is available and each is shown in Table 5.

Figure 3: Thermal conductivity variation along the z direction
Table 5: Analytical solutions for various forms of thermal conductivity $k(x)$

| $k(x)$               | Analytical solution for potential function |
|----------------------|-------------------------------------------|
| $5(1 + 2z)^2$        | $\phi = \frac{300z}{1 + 2z}$             |
| $5e^{2z}$            | $\phi = 100\frac{1 - e^{-2z}}{1 - e^{-2x}}$ |
| $5(cosz + 2sinz)^2$  | $\phi = 100\frac{(\cot(L)+2)*sinz}{(cosz+2sinz)}$ |

In what follows, the DCM with different configurations is applied for this model and the results are compared. Figure 5 shows the relative error corresponding to the DCM with different activation functions with varying layers. It can be concluded that the arctan function yields very stable and accurate results. Both arctan and Tanh function outperformed other activation functions in the prediction of potential function. According to Figure 6 for Korobov, Hammersley, Random sampling method, as hidden layer increases, the results get worse, but for LatinHypercube sampling, the results improve as hidden layer increases. LatinHypercube sampling method is chosen for the rest numerical examples in this application with deep neural networks. Figure 7 shows the predicted error corresponding to the DCM with different material variations with varying hidden layers. The DCM can best predict the potential function with the exponential conductivity. Moreover, Figure 5-7 shows the DCM with 2 hidden layers has a better performance for the problem.
Figure 5: Comparison of results predicted by DCM with different activation functions
Figure 6: Comparison of results predicted by DCM with different sampling methods.
Figure 7: Comparison of predicted results for different material variations
Figure 8: Comparison of predicted results for different collocation points in cube
To further investigate the requirements of collocation points for the deep collocation method, we study the relative errors obtained for different numbers of collocation points, which include the number of points in cube and the number of points in surfaces, the relative error of temperature is compared in Figure 8 and 9. It is obvious, that the number of points in cube has a greater influence on the results. For this problem when the number of collocation points in the cube is 3000 and the number of points on the boundaries is 300, the result can reach the most desired accuracy.

The temperature profile along the z-axis for the three material variations are plotted and compared with the corresponding analytical solutions in Figure 10, the predicted temperature and flux and analytical results are in excellent agreement.
The predicted temperature and flux contour for three material variations inside the cube is shown in Figure 11-13. For all three cases as shown, it can be observed that the heat distribution varies with graded variation in the z coordinates which is consistent with the material property of the FGMs.
Figure 12: Predicted (a) temperature for trigonometric variation of $k$; and (b) flux for the functionally graded unit cubic

Figure 13: Predicted (a) temperature for quadratic variation of $k$; and (b) flux for the functionally graded unit cubic

An optimization algorithm can minimize a loss function, the algorithm includes first- and second-order based methods. First-order methods minimize the function using its gradient. Second-order method minimize the function using the second derivative (Hessian). In this application a combination of those two optimizers is deployed, with Adam algorithm chosen for the first-order method, L-BFGS the second-order method. The combined optimization method is proved better performance in the model.

The convergence history of different optimizers can be seen in Figure 14, it is obvious, Adam converges faster but over the long term L-BFGS achieves lower loss, that compared with the Adam or L-BFGS, the combination of Adam and L-BFGS can greatly improve the accuracy of the results, and the combination can maintain efficiency.
In the following the problem is solved using network with 1, 2, 3, 4, 5 and 6 hidden layer, and the results are illustrated in Figure 15, it can be clearly found that when the network with 2 hidden layers, the results can reach a desired level of accuracy in a short time.
4.2. Case 2: Cube with a 3D material gradation

The three-dimensional thermal conductivity variation is

\[ k(x, y, z) = (5 + 0.2x + 0.4y + 0.6z + 0.1xy + 0.2yz + 0.3zx + 0.7xyz)^2 \]  

the iso-surfaces of 3D variation of the thermal conductivity is illustrated in Figure 16 and the analytical solution is

\[ \phi(x, y, z) = \frac{xyz}{(5 + 0.2x + 0.4y + 0.6z + 0.1xy + 0.2yz + 0.3zx + 0.7xyz)} \]
The boundary conditions at the six faces of the cube are listed in Table 5.

Table 6: the boundary conditions of cube with a 3D material gradation

| Boundary condition | Dirichlet | Neumann |
|--------------------|-----------|---------|
| $\phi(0, y, z) = 0$ | $q(1, y, z) = -0.2zy(25 + 2y + 3z + zy)$ |
| $\phi(x, 0, z) = 0$ | $q(x, 1, z) = -0.1xz(50 + 2x + 6z + 3xz)$ |
| $\phi(x, y, 0) = 0$ | $q(1, y, z) = -0.1xy(50 + 2x + 4y + xy)$ |

With the network configurations studied in Case 1 such as 3000 collocation points inside the cube and 300 collocation points on the boundaries, the results for predicted temperature and flux contour inside the cube are shown in Figure 17, which are in exact agreement with the analytical solution. It can also be observed that the temperature varies in the direction of material variation.
The temperature along the diagonal line for the cube with a 3D material gradation is plotted and compared with the analytical results in Figure 18; the results of the predicted are consistent with the analytical results.

4.3. Case 3: Irregular shape geometry

Last but not the least, an FGM in irregular shape geometry is studied, with inner radius is 0.3, outer radius 0.5, the top surface is \(Z=0.1\), and the thermal conductivity for the geometry varies as an exponential
function according to

\[ k(z) = 5e^{3z} \]  \hspace{1cm} \text{(35)}

The profile of the thermal conductivity \( k(z) \) of the variation is illustrated in Figure 19.

The profile of thermal conductivity in the direction. The exponential variation of the conductivity is \( k(z) = 5e^{3z} \)

The temperature is specified along the inner radius as \( T_{\text{inner}} = 0 \), and outer radius as \( T_{\text{outer}} = 100 \), and all other surfaces are insulated. The boundary conditions of the geometry are shown in Figure 20.
The results of predicted temperature are shown in Figure and the solution of the problem is verified using software ABAQUS in Figure 21-22. It can be seen that the results of predicted solution by neural network and the verified solution by ABAQUS are in very good agreement. It can be proved that neural network is suitable for solving this irregular shaped functionally graded material. Also, it can be concluded that the
The temperature along the radial direction at the edge is plotted and compared with the verified results by ABAQUS in Figure 23, the results of the predicted and verified Temperature are in excellent agreement.

4.4. Conclusion

This paper presents the prediction for solving the problems of potential in non-homogeneous media with different geometries using deep learning based collocation method. This deep collocation method combines the classical collocation method and the deep learning method in one framework and reduce the solving
of partial differential equations into an optimization problem. Because nonlinear activation function is adopted to introduce nonlinearity into the neural network, it enables us to discover the nonlinear pattern under certain mechanical problems. For detailed implementation, the governing equation is incorporated in the loss function which makes the deep collocation driven by not merely pure data but also the physic law. For the training of this deep collocation method, sampling inside the physic domain is necessary, there is a variety of favorable sampling methods, but not all of them suitable for our model. With numerical experiments, it is verified that the Latin Hypercube sampling is very suitable for our model. In this sense, our deep collocation method is truly "meshfree". Also, this method is semi-supervised learning, no labels are attached to the solution except for the meet of boundary conditions. Moreover, in this implementation, we seek to find the favorable configuration of neural networks and offer guidance for the later application.

Various numerical examples concerning the heat transfer, typical potential problems, in functionally graded materials are studied to verify the effective of deep collocation method and offer advice for the neural network configuration including the choice of activation function, sampling method, optimizers, and numbers of collocation points inside domain and on the boundaries. In conclusion, it is verified that the deep collocation method is very suitable for solving problems of potential in nonhomogeneous media and unveils the temperature and flux field accurately. So it has the potential to become a classical numerical method, and the problems involve obtaining a general training set and building a general model will be widely used in the future works.
References

[1] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[2] Wenzhen Qu, Wen Chen, and Zhouchi Fu. Solutions of 2d and 3d non-homogeneous potential problems by using a boundary element-collocation method. *Engineering Analysis with Boundary Elements*, 60:2–9, 2015.

[3] CJS Alves and CS Chen. A new method of fundamental solutions applied to nonhomogeneous elliptic problems. *Advances in Computational Mathematics*, 23(1-2):125–142, 2005.

[4] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1359–1364, 2018.

[5] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

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

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Liping Yang, Alan MacEachren, Praenjit 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.

[10] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentin, Huining Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaodong Wu, Fangheng Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[11] Wanli Ouyang, Xiaogang Wang, Xingyu Zeng, Shi Qi, Ping Luo, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Chen-Change Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2015.

[12] 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*, 2015.

[13] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[14] Ali Bou Nassif, Ismail Shahin, Imtitan Attili, Mohammad Azzez, and Khaled Shaalan. Speech recognition using deep neural networks: a systematic review. *IEEE Access*, 7:1084344–1084351, 2019.

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

[16] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *Engineering Analysis with Boundary Elements*, 23(1):35–48, 1998.

[17] Isaac E Lagaris, Aristidis C 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.

[18] 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.

[19] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *International conference on computer vision*, pages 173–182, 2016.

[20] Vitaliy Gyrya, Mikhail Jurievich Shashkov, Alexei N Skurikhin, and Svetlana Tokareva. Machine learning approaches for deep neural networks: a systematic review. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2015.

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

[22] Anna Cicalese, Maria Elena Baldi, and Alessandro Ricci. Pedagogical use of deep learning for biology and medicine. *Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[23] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[24] Vitaliy Gyrya, Mikhail Jurievich Shashkov, Alexei N Skurikhin, and Svetlana Tokareva. Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study.

[25] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Biology and medicine*. In *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[26] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving differential equations. *Advances in neural information processing systems*, pages 2403–2412, 2018.

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

[28] Wenzhen Qu, Wen Chen, and Zhouchi Fu. Solutions of 2d and 3d non-homogeneous potential problems by using a boundary element-collocation method. *Engineering Analysis with Boundary Elements*, 60:2–9, 2015.

[29] 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.

[30] 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.

[31] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[32] Vitaliy Gyrya, Mikhail Jurievich Shashkov, Alexei N Skurikhin, and Svetlana Tokareva. Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study.

[33] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[34] Vitaliy Gyrya, Mikhail Jurievich Shashkov, Alexei N Skurikhin, and Svetlana Tokareva. Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study.

[35] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[36] Vitaliy Gyrya, Mikhail Jurievich Shashkov, Alexei N Skurikhin, and Svetlana Tokareva. Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study.

[37] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[38] Vitaliy Gyrya, Mikhail Jurievich Shashkov, Alexei N Skurikhin, and Svetlana Tokareva. Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study.

[39] Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.
forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[32] 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.

[33] Cosmin Anitescu, Elena Atroshchenko, Naif Alajlan, and Timon Rabczuk. Artificial neural network methods for the solution of second order boundary value problems. *Computers, Materials & Continua*, 59(1):345–359, 2019.

[34] Hongwei Guo, Xiaoying Zhuang, and Timon Rabczuk. A deep collocation method for the bending analysis of kirchhoff plate. *CMC-COMPUTERS MATERIALS & CONTINUA*, 59(2):433–456, 2019.

[35] Vien Minh Nguyen-Thanh, Xiaoying Zhuang, and Timon Rabczuk. A deep energy method for finite deformation hyperelasticity. *European Journal of Mechanics-A/Solids*, page 103874, 2019.

[36] Federico Paris and Jose Canas. *Boundary element method: fundamentals and applications*, volume 1. Oxford University Press, USA, 1997.

[37] Anshul Dhingra. Activation functions in neural networks.

[38] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[39] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pages 4939–4948, 2018.

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

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

[42] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[43] Ewaryst Rafajlowicz and Rainer Schwabe. Halton and hammersley sequences in multivariate nonparametric regression. *Statistics & Probability Letters*, 76(8):803–812, 2006.

[44] Xiaoqun Wang, Ian H Sloan, and Josef Dick. On korobov lattice rules in weighted spaces. *SIAM journal on numerical analysis*, 42(4):1760–1779, 2004.

[45] Josef Dick, Friedrich Pillichshammer, and Benjamin J Waterhouse. The construction of good extensible korobov rules. *Computing*, 79(1):79–91, 2007.

[46] Michael D Shields and Jiaxin Zhang. The generalization of latin hypercube sampling. *Reliability Engineering & System Safety*, 148:96–108, 2016.

[47] Alexander Shapiro. Monte carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

[48] Frank Olken. *Random sampling from databases*. PhD thesis, University of California, Berkeley, 1993.

[49] Alok Sutradhar and Glaucio H Paulino. A simple boundary element method for problems of potential in non-homogeneous media. *International Journal for Numerical Methods in Engineering*, 60(13):2203–2230, 2004.