Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 389 (2022) 114414
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

# HiDeNN-TD: Reduced-order hierarchical deep learning neural networks

Lei Zhang[a,c,1], Ye Lu[b], Shaoqiang Tang[a,*], Wing Kam Liu[b,*]

[a] *HEDPS and LTCS, College of Engineering, Peking University, Beijing 100871, China*
[b] *Department of Mechanical Engineering, Northwestern University, Evanston, USA*
[c] *Visiting student at Department of Mechanical Engineering, Northwestern University, USA*

Received 22 August 2021; received in revised form 23 November 2021; accepted 25 November 2021
Available online 18 December 2021

**Abstract**

This paper presents a tensor decomposition (TD) based reduced-order model of the hierarchical deep-learning neural networks (HiDeNN). The proposed HiDeNN-TD method keeps advantages of both HiDeNN and TD methods. The automatic mesh adaptivity makes the HiDeNN-TD more accurate than the finite element method (FEM) and conventional proper generalized decomposition (PGD) and TD, using a fraction of the FEM degrees of freedom. This work focuses on the theoretical foundation of the method. Hence, the accuracy and convergence of the method have been studied theoretically and numerically, with a comparison to different methods, including FEM, PGD, TD, HiDeNN and Deep Neural Networks. In addition, we have theoretically shown that the PGD/TD converges to FEM at increasing modes, and the PGD/TD solution error is a summation of the mesh discretization error and the mode reduction error. The proposed HiDeNN-TD shows a high accuracy with orders of magnitude fewer degrees of freedom than FEM, and hence a high potential to achieve fast computations with a high level of accuracy for large-size engineering and scientific problems. As a trade-off between accuracy and efficiency, we propose a highly efficient solution strategy called HiDeNN-PGD. Although the solution is less accurate than HiDeNN-TD, HiDeNN-PGD still provides a higher accuracy than PGD/TD and FEM with only a small amount of additional cost to PGD.


*Keywords:* Hierarchical deep-learning neural networks; Proper generalized decomposition; Canonical tensor decomposition; Reduced order finite element method; Convergence study and error bound

## 1. Introduction

Despite the constantly increasing computer power, numerical simulations of physical systems with numerous degrees of freedom (DoFs) remain computationally prohibitive. These kinds of problems arise usually in simulation-based engineering and scientific applications, and the repetitive manipulation (or modification) of the mesh system has been identified as a key time-costly issue in standard finite element method (FEM) [1]. This has been a motivation for developing the isogeometric approaches [2] and meshfree particle methods [3–6].

* Corresponding authors.
*E-mail addresses:* maotang@pku.edu.cn (S. Tang), w-liu@northwestern.edu (W.K. Liu).
[1] Current address: The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Science, Beijing 100190, China.

https://doi.org/10.1016/j.cma.2021.114414
0045-7825/

In recent years, the deep neural network (DNN) has shown some interesting features in handling the solution of physics constrained systems. The universal approximation theorem [7,8] and the natural scalability of DNN have been the foundation of its superior performance for large systems. This has thus motivated the use of DNN to approximate the solution of partial differential equations (PDEs) [9–11]. A recently developed Hierarchical Deep-learning Neural Network (HiDeNN) method [12,13] falls within this perspective. The so-called HiDeNN is developed by constraining the weights and biases of DNN to mesh coordinates to build multiple dimensions finite element, meshfree, isogeometric, B-spline, and NURBS interpolation functions. HiDeNN allowed the automatic mesh adaptivity and showed a good potential to prevent large mesh systems and the standard time-consuming mesh refinement procedure. In order to further enhance the efficiency of HiDeNN, this work proposed HiDeNN-TD, a reduced-order model of HiDeNN using the tensor decomposition (TD), in particular the canonical tensor decomposition which can give rise to the optimal proper generalized decomposition (PGD).

The PGD-based model reduction methods rely on the idea of separation of variables, and are usually written in the format of canonical decomposition. We remark that Song discovered an equivalent form of PGD when being used to function approximation [14]. He actually constructed the optimal decomposition of any square integrable function into minimal number of modes, each to be the product of single variable functions, and proved exponential convergence rate for such construction. This kind of method was originally proposed in an *a priori* setting [15–18], in which the separated functions are computed on-the-fly by solving the PDEs. It has gained increased popularity in recent years. For overcoming the intrusiveness and extending the applicability of the method, *a posteriori* data-driven PGD [19–22] has also been developed more recently. In contrast to *a priori* PGD, *a posteriori* method uses a database to learn the separated functions and thus can be used as regression for constructing reduced order surrogate models.

The conventional PGD methods usually use an incremental solution scheme to compute the separated functions mode by mode. Hence, the optimality of decomposition cannot be guaranteed. Some attempts have been made to improve the optimality of PGD [23]. The optimal PGD can be defined as a Galerkin projection with a fixed number of modes [24], which requires solving all the PGD modes together. This type of solution scheme is also adopted in the canonical tensor decomposition (see e.g. [25]). In order to clearly distinguish the two types of solution schemes, we define the one with a fixed number of modes as the general TD in this paper.

In our work, we adopted the same idea of separation of variables and the solution scheme of TD for solving PDEs in the HiDeNN framework, leading to the so-called HiDeNN-TD method, which is expected to have reduced degrees of freedom with high accuracy compared with other traditional methods such as the FEM. Indeed, the space separated PGD, leading to lower dimensional space functions, is usually considered for reducing the computational complexity of 3D separable domains (see e.g. [26]). However, the convergence aspect with respect to the mesh refinement and number of modes has been less studied.

We investigated the convergence aspect of the PGD/TD approach in this paper. Based on the approximation function spaces, we analyzed the numerical error and convergence associated with different approaches and compared their error bounds. It can be shown that the HiDeNN-TD is more accurate than both FEM and conventional PGD/TD, thanks to the adaptivity achieved by HiDeNN. Furthermore, we suggested fixing the number of modes firstly and solving them together. Hence, *HiDeNN-TD can require fewer modes than PGD*. This is advantageous for high-dimensional problems where the optimality of modes is crucial. The numerical examples have confirmed our theoretical analysis. In addition, we numerically investigated the relationship between the approximation error and the modes, and proposed a strategy to select the prescribed mode number in HiDeNN-TD. The proposed HiDeNN-TD has shown a high potential to achieve high performance computing with high accuracy.

We remark that this paper focuses on the theoretical foundation of various approximation methods based on HiDeNN. Theoretically, HiDeNN can provide the most accurate solutions, but the computational cost is higher than HiDeNN-TD. We seek to make a trade-off between accuracy and speed by combining HiDeNN and TD in this work. We shall demonstrate out finding that HiDeNN-TD gives better results than PGD, TD, and FEM by taking advantages of both HiDeNN and TD. The currently used gradient based optimization scheme can converge quickly to a well-accepted tolerance with a better accuracy and more efficiency than that of the FEM. HiDeNN-TD has slightly more degrees of freedom than TD/PGD but provides the capability to improve the accuracy of the results. The added degrees of freedom compared to TD is controllable to achieve a high efficiency. For example, one efficient way to solve HiDeNN-TD solution is to start from PGD to get a first estimate of the number of modes then switch to HiDeNN-TD to gain more accurate results. The estimation of the number of modes can be based

on low-accurate PGD solutions, rather than high-accurate PGD solutions which require a larger number of modes. HiDeNN-TD provides a way to optimize the accuracy of the decomposition and sacrifices only slightly in speed. Furthermore, our work uses gradient based optimization scheme with a tight tolerance to get the converged results for demonstrating the theoretical proof. But in practical cases, a trade-off between accuracy and efficiency is usually made. To further prove this point, we have provided a variant solution method called HiDeNN-PGD in this paper, which combines directly the PGD type solution scheme with HiDeNN in the previously mentioned hybrid manner. As expected, the HiDeNN-PGD is only slightly more costly than PGD and provides more accurate solutions. Based on this scheme, a more efficient HiDeNN-TD solution method can be further developed to improve the accuracy and is expected to give the highest accuracy for solutions. The HiDeNN-PGD can be seen as a more efficient variant by making the trade-off between accuracy and efficiency.

The rest of the paper is organized as follows. Section 2 gives a brief overview of different numerical methods for PDEs, in which the approximation function spaces are described. The error analysis based on a class of PDEs is given in Section 3. Section 4 presents the proposed HiDeNN-TD method. Section 5 provides some numerical examples and discussions. Finally, the paper closes with some concluding remarks.

## 2. DNN, HiDeNN, FEM and TD based function approximation

Function approximation is a key component in numerical solutions of PDEs. In this section, we briefly review how such an approximation can be performed in terms of FEM, DNN, HiDeNN and TD. We present their approximation function sets, which will be used in the theoretical analysis in Subsection 2.2. We restrict the discussion to a scalar-valued function $u(\boldsymbol{x}) : \mathbb{R}^3 \to \mathbb{R}$. The conclusions should be straightforwardly extended to vector functions.

### 2.1. Overview of the approximation function sets

**Deep neural network-based method**

According to the universal approximation theorem, a DNN can be designed to approximate any given continuous function to desired accuracy [7,8,27]. Thus it can be a candidate to approximate solutions for solving PDEs [9,10, 28–33], i.e.,

$$u^h(\boldsymbol{x}) = \mathcal{F}^{NN}(\boldsymbol{x}), \tag{1}$$

where $\mathcal{F}^{NN}$ represents the neural network with $\boldsymbol{x}$ as input and $u^h$ as output. Note that $u^h$ can be a multidimensional vector. For instance, in a classical feedforward neural network (FFNN) [34–36] with $N_L$ layers, recursive relations among neurons are as follows

$$a^l_{j=1} = x, a^l_{j=2} = y, a^l_{j=3} = z, \text{ if } l = 1 \text{ (input layer)}; \tag{2}$$

$$a^l_j = \mathcal{A}(\sum_{i=1}^{N_N^{l-1}} W^l_{ij} a^{l-1}_i + b^l_j), \text{ if } l \in \{2, \ldots, N_L - 1\} \text{ (hidden layer)}. \tag{3}$$

Hence, the output layer can be defined as

$$\mathcal{F}^{NN}_j = a^{N_L}_j = \sum_{i=1}^{N_N^{N_L-1}} W^{N_L}_{ij} a^{N_L-1}_i + b^{N_L}_j, \text{ if } l = N_L \text{ (output layer)}, \tag{4}$$

with the detailed definition of the notations in Table 1. Therefore, once the weights $\boldsymbol{W}$, biases $\boldsymbol{b}$ and activation functions $\boldsymbol{\mathcal{A}}$ have been chosen, $\mathcal{F}^{NN}$ can serve as an approximation function with the input variable as $\boldsymbol{x} = (x, y, z)$.

The approximation function set forward by a general DNN is

$$\mathcal{N}^h = \left\{ u^h(\boldsymbol{x}) \middle| u^h = \mathcal{F}^{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\mathcal{A}}), W^l_{ij} \in \mathbb{R}, b^l_j \in \mathbb{R} \right\}, \tag{5}$$

where $\mathcal{F}^{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\mathcal{A}})$ denotes a DNN with the input $\boldsymbol{x}$, and depends on weights $\boldsymbol{W}$, biases $\boldsymbol{b}$ and activation functions $\boldsymbol{\mathcal{A}}$.

**Table 1**
Notation table of variables used in the feed forward neural network.

| | |
|---|---|
| $\boldsymbol{x} = (x, y, z)$ | Space coordinates |
| $l$ | Counting index for number of layers |
| $i$ | Counting index for neurons in layer $l-1$ |
| $j$ | Counting index for neurons in layer $l$ |
| $N_L$ | Number of layers in the neural network |
| $N_N^l$ | Number of neurons in layer $l$ |
| $W_{ij}^l$ | Weight connecting the $i$th neuron in layer $l-1$ to the $j$th in layer $l$ |
| $b_j^l$ | Bias of the $j$th neuron in layer $l$ |
| $a_j^l$ | Neuron value for $j$th neuron in $l$th layer |
| $\mathcal{A}$ | Activation function |
| $\mathcal{F}^{NN}$ | Feedforward neural network function |

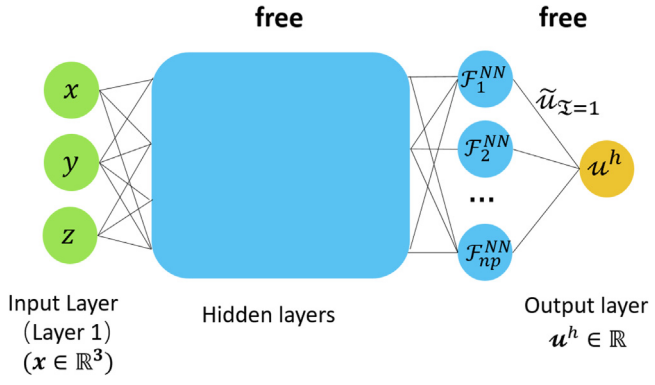



**Fig. 1.** Illustration for the DNN interpolation functions with $\boldsymbol{x} = (x, y, z)$ as input and $u^h$ as output.

For interpretation, DNN can be somehow rewritten in the form of shape functions associated to nodal values of $u^h$. In this way, the function approximation reads

$$u^h = \sum_{\mathcal{I}=1}^{np} \mathcal{F}_{\mathcal{I}}^{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\mathcal{A}}) u_{\mathcal{I}}, \tag{6}$$

as illustrated in Fig. 1, where $np$ is the number of nodes. $\mathcal{F}_{\mathcal{I}}^{NN}$ represents the value of the $\mathcal{I}$-th neuron in the last hidden layer, i.e., the output of the previous hidden layers. $u_{\mathcal{I}}$ is the corresponding weight connecting the output layer with the last hidden layer. This interpolation form may provide a more interpretable structure for DNN.

In multidimensional cases, such as for 3D mechanical problems, the above equation can be straightforwardly applied to each component of displacement as follows

$$u_x^h = \sum_{\mathcal{I}=1}^{np} \mathcal{F}_{\mathcal{I}}^{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\mathcal{A}}) u_{x\mathcal{I}}, \tag{7}$$

$$u_y^h = \sum_{\mathcal{I}=1}^{np} \mathcal{F}_{\mathcal{I}}^{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\mathcal{A}}) u_{y\mathcal{I}}, \tag{8}$$

$$u_z^h = \sum_{\mathcal{I}=1}^{np} \mathcal{F}_{\mathcal{I}}^{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\mathcal{A}}) u_{z\mathcal{I}}. \tag{9}$$

**HiDeNN**

The recently developed HiDeNN method [12] uses a similar DNN structure of (6) with additional constraints to build a family of function approximations. Similar to the FEM, the domain $\Omega$ is discretized by a mesh with $np$ nodes $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{np}$. Then the finite element shape functions $N_{\mathcal{I}}$ can be constructed by the neural network block,

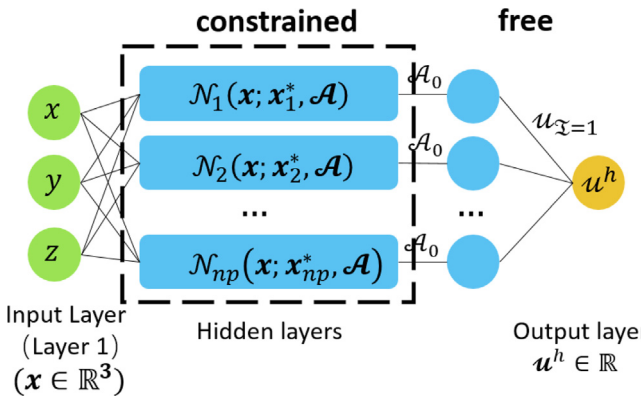



**Fig. 2.** Illustration for the HiDeNN interpolation functions with $\boldsymbol{x} = (x, y, z)$ as input and $u^h$ as output. Weights and biases inside dashed line-box are constrained. $\mathcal{A}_0$ is the identity activation function defined by $\mathcal{A}_0(x) = x$.

namely,

$$\mathcal{N}_{\mathcal{I}}(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}}) \tag{10}$$

where $\mathcal{I} = 1, 2, \ldots, np$. Different from $\mathcal{F}_{\mathcal{I}}^{NN}(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}})$ in (6), $\mathcal{N}_{\mathcal{I}}(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}})$ precisely equals the finite element shape function $N_{\mathcal{I}}(\boldsymbol{x})$ with inputs $\boldsymbol{x}$ and an output $N_{\mathcal{I}}$, satisfying the following constraints for shape functions automatically,

$$\sum_{\mathcal{I}=1}^{np}\mathcal{N}_{\mathcal{I}}(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}}) = 1, \mathcal{N}_{\mathcal{I}}(\boldsymbol{x}_{\mathcal{J}};\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}}) = \delta_{\mathcal{I}\mathcal{J}}. \tag{11}$$

With Kronecker Delta constraints, we can apply Dirichlet boundary conditions directly similar to that of the finite element method, so that all the weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ are functions of nodal coordinates $\boldsymbol{x}_I$. Thus we can rewrite the shape function explicitly in terms of $\boldsymbol{x}_I$ as

$$\mathcal{N}_{\mathcal{I}}(\boldsymbol{x};\boldsymbol{x}_{\mathcal{I}}^*,\boldsymbol{\mathcal{A}}) = \mathcal{N}_{\mathcal{I}}(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}}), \tag{12}$$

where $\boldsymbol{x}_{\mathcal{I}}^*$ denotes the support of $N_{\mathcal{I}}(\boldsymbol{x})$, e.g. in linear 1D cases $\boldsymbol{x}_{\mathcal{I}}^* = \{x_{\mathcal{I}-1}, x_{\mathcal{I}}, x_{\mathcal{I}+1}\}$.

Combining such neural network blocks for the entire mesh gives the final form of HiDeNN, as shown in Fig. 2. This results in the approximation function set

$$\mathcal{H}^h = \left\{ u^h(\boldsymbol{x}) \middle| u^h = \sum_{\mathcal{I}=1}^{np}\mathcal{N}_{\mathcal{I}}(\boldsymbol{x};\boldsymbol{x}_{\mathcal{I}}^*,\boldsymbol{\mathcal{A}})u_{\mathcal{I}}, u_{\mathcal{I}} \in \mathbb{R} \right\}. \tag{13}$$

The parametric expression with nodal positions $\boldsymbol{x}_{\mathcal{I}}^*$ allows automatic r-adaptivity, and accordingly improves the local and global accuracy of the interpolant. Remark that there are two different solution schemes for HiDeNN shown in Appendix B. In the following numerical examples, we adopt Scheme 1, i.e., optimize nodal values and nodal positions by Adam algorithm simultaneously.

For regular mesh, we propose a constrained version of HiDeNN, which requires that the nodes merely move along each direction. The shape function at the node $\boldsymbol{x}_{\mathcal{I}} = (x_I, y_J, z_K)$ is the product of 1D HiDeNN shape functions $\mathcal{N}_I(x;\boldsymbol{x}_I^*,\boldsymbol{\mathcal{A}}), \mathcal{N}_J(y;\boldsymbol{y}_J^*,\boldsymbol{\mathcal{A}}), \mathcal{N}_K(z;\boldsymbol{z}_K^*,\boldsymbol{\mathcal{A}})$. The approximation function set of the regulated HiDeNN reads

$$\mathcal{HR}^h = \left\{ u^h(\boldsymbol{x}) \middle| u^h = \sum_{I=1}^{n_1}\sum_{J=1}^{n_2}\sum_{K=1}^{n_3}\mathcal{N}_I(x;\boldsymbol{x}_I^*,\boldsymbol{\mathcal{A}})\mathcal{N}_J(y;\boldsymbol{y}_J^*,\boldsymbol{\mathcal{A}})\mathcal{N}_K(z;\boldsymbol{z}_K^*,\boldsymbol{\mathcal{A}})u_{(I,J,K)}, u_{(I,J,K)} \in \mathbb{R} \right\}. \tag{14}$$

Here, $n_1, n_2, n_3$ are the number of nodes in $x, y, z$ directions, respectively. This might lose accuracy but avoids mesh distortion and reduces DoFs.

**Finite element method**

The approximation function set $\mathcal{H}^h$ degenerates to the FE approximation function set $\mathcal{V}^h$ [1] when the nodal position is fixed, which reads

$$\mathcal{V}^h = \left\{ u^h(\boldsymbol{x}) \middle| u^h = \sum_{\mathcal{I}=1}^{np} N_{\mathcal{I}}(\boldsymbol{x}) u_{\mathcal{I}},\, u_{\mathcal{I}} \in \mathbb{R} \right\}. \tag{15}$$

From the DNN viewpoint, this corresponds to fixing the weights and biases that are functions of $\boldsymbol{x}^*_{\mathcal{I}}$, as shown in Fig. 3.

**Tensor decomposition**

Under the assumption of separation of variables, the function $\boldsymbol{u}$ may be approximated by the sum of the products of multiple 1D functions, i.e.,

$$u^h(\boldsymbol{x}) = u^h(x, y, z) = \sum_{q=1}^{Q} X^{(q)}(x) Y^{(q)}(y) Z^{(q)}(z), \tag{16}$$

where $Q$ is the number of modes, and the product of $X^{(q)}$, $Y^{(q)}$, $Z^{(q)}$ provides a mode for the interpolation function. This form or concept is known as canonical tensor decomposition [25]. The so-called PGD method has adopted this concept for solving PDEs [15,37] and for data learning [19,20,38].

Thanks to the separation of variables, only shape functions of reduced dimension are needed. In (16), 1D FE shape functions can be used for a 3D problem, namely,

$$X^{(q)}(x) = \sum_{I=1}^{n_1} N_I(x) \beta_I^{(q)}, \tag{17}$$

$$Y^{(q)}(y) = \sum_{J=1}^{n_2} N_J(y) \gamma_J^{(q)}, \tag{18}$$

$$Z^{(q)}(z) = \sum_{K=1}^{n_3} N_K(z) \theta_K^{(q)}. \tag{19}$$

Here, $n_1, n_2, n_3$ are the number of nodes in $x, y, z$ directions, respectively. Thus the corresponding approximation function set for $Q$ modes with a prescribed $Q$ is

$$\mathcal{M}_Q^h = \left\{ u^h(\boldsymbol{x}) \middle| u^h = \sum_{q=1}^{Q} \left( \sum_{I=1}^{n_1} N_I(x) \beta_I^{(q)} \right) \left( \sum_{J=1}^{n_2} N_J(y) \gamma_J^{(q)} \right) \left( \sum_{K=1}^{n_3} N_K(z) \theta_K^{(q)} \right), \right.$$
$$\left. \beta_I^{(q)}, \gamma_J^{(q)}, \theta_K^{(q)} \in \mathbb{R} \right\}. \tag{20}$$

Fig. 4 illustrates a DNN format of the TD interpolation function with $Q$ modes. The symbol $\boxed{\mathscr{M}}$ in the figure is used to represent the multiplication in a DNN format and is given in [12]. When a few modes may represent the function $u^h(x, y, z)$, this method is advantageous in terms of lower integration complexity and less DoFs [15,39]. The DoFs are of the order $O((n_1 + n_2 + n_3)Q)$, which are linear with the spatial dimension and far smaller than traditional methods (e.g., FEM).

Note that the interpolation function in (16) can be rearranged as

$$u^h(x, y, z) = \sum_{I=1}^{n_1} \sum_{J=1}^{n_2} \sum_{K=1}^{n_3} N_I(x) N_J(y) N_K(z) \left( \sum_{q=1}^{Q} \beta_I^{(q)} \gamma_J^{(q)} \theta_K^{(q)} \right), \tag{21}$$

which is regarded as a finite element interpolation function with $N_I(x)N_J(y)N_K(z)$ as shape functions and $\left(\sum_{q=1}^{Q} \beta_I^{(q)} \gamma_J^{(q)} \theta_K^{(q)}\right)$ as coefficients. Eq. (21) is the equivalent form of PGD/TD interpolation function in (20). We rewrite it in the FEM form to show the relationship with FEM, but the nodal value is in the form of separation of variables, so DoFs are still linear with dimensions. Fig. 5 illustrates a DNN format of (21), which will be the basis for the proposed HiDeNN-TD method.

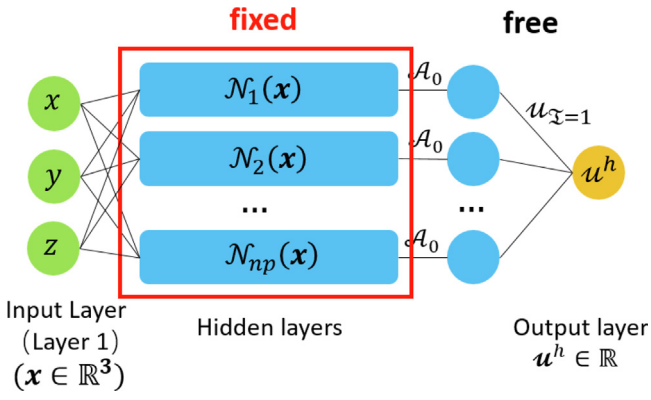



**Fig. 3.** Illustration for the FEM interpolation functions in the form of DNN with $\boldsymbol{x} = (x, y, z)$ as input and $u^h$ as output. Weights and biases inside red solid line-box are fixed. $\mathcal{A}_0$ is the identity activation function defined by $\mathcal{A}_0(x) = x$.

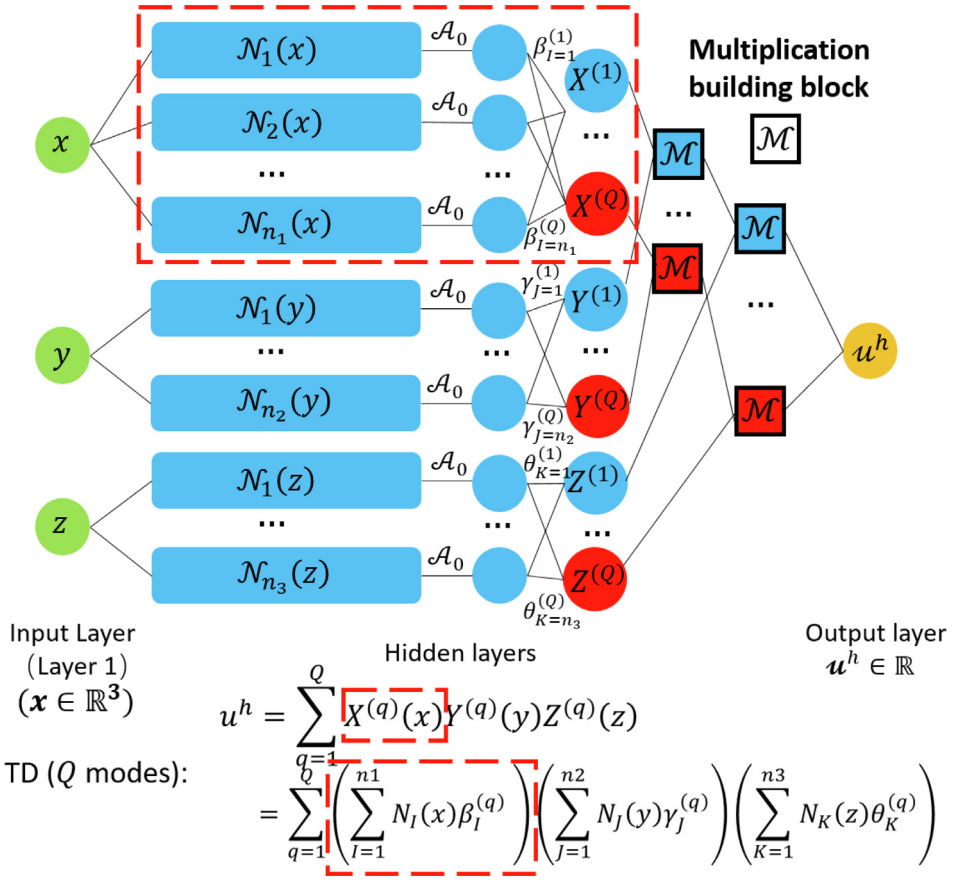



**Fig. 4.** Illustration for the TD interpolation function with $Q$ modes in the form of DNN. The symbol $\boxed{\mathcal{M}}$ is used to represent the multiplication in a DNN format.

Multidimensional shape functions of TD, i.e., the products of 1D shape functions, are fixed and determined by nodal positions along each direction. In addition, the nodal values in the last layer are constraint in the form of the

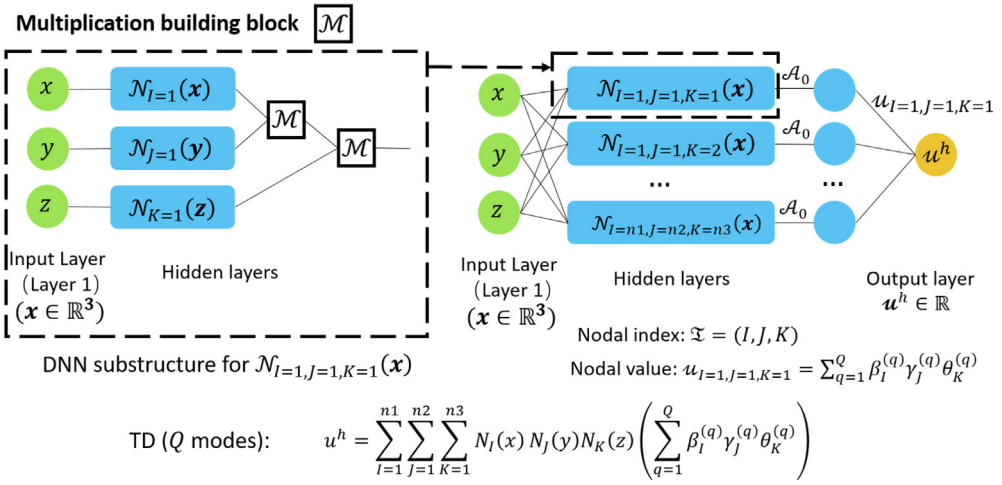



**Fig. 5.** Illustration for the expansion (21) of TD interpolation function with $Q$ modes in the form of DNN. The symbol $\boxed{\mathcal{M}}$ is used to represent the multiplication in a DNN format.

tensor product, i.e.,

$$u_{(I,J,K)} = \sum_{q=1}^{Q} \beta_I^{(q)} \gamma_J^{(q)} \theta_K^{(q)}. \tag{22}$$

### 2.2. HiDeNN-TD: reduced order HiDeNN via TD

HiDeNN gets better accuracy compared with classical FEM due to the adaptivity of nodal positions. More DoFs might result in heavier cost. On the other hand, representation of separated variables provides a reduced order model to improve the efficiency but might lose accuracy. Here, we propose HiDeNN-TD, a reduced-order model of HiDeNN via TD, which seeks to leverage accuracy and computational cost.

In HiDeNN-TD, the shape functions in each direction are written in the DNN format, namely, (17)–(19) are replaced by 1D HiDeNN interpolants (refer to Appendix A),

$$X^{(q)}(x) = \sum_{I=1}^{n_1} \mathcal{N}_I(x; \boldsymbol{x}_I^*, \boldsymbol{\mathcal{A}}) \beta_I^{(q)}, \tag{23}$$

$$Y^{(q)}(y) = \sum_{J=1}^{n_2} \mathcal{N}_J(y; \boldsymbol{y}_J^*, \boldsymbol{\mathcal{A}}) \gamma_J^{(q)}, \tag{24}$$

$$Z^{(q)}(z) = \sum_{K=1}^{n_3} \mathcal{N}_K(z; \boldsymbol{z}_K^*, \boldsymbol{\mathcal{A}}) \theta_K^{(q)}. \tag{25}$$

Thus the interpolation function set is defined by

$$\mathcal{G}_Q^h = \left\{ u^h \middle| u^h = \sum_{q=1}^{Q} \left( \sum_{I=1}^{n_1} \mathcal{N}_I(x; \boldsymbol{x}_I^*, \boldsymbol{\mathcal{A}}) \beta_I^{(q)} \right) \left( \sum_{J=1}^{n_2} \mathcal{N}_J(y; \boldsymbol{y}_J^*, \boldsymbol{\mathcal{A}}) \gamma_J^{(q)} \right) \left( \sum_{K=1}^{n_3} \mathcal{N}_K(z; \boldsymbol{z}_K^*, \boldsymbol{\mathcal{A}}) \theta_K^{(q)} \right) \right\}. \tag{26}$$

The corresponding DNN format of the HiDeNN-TD interpolation function with $Q$ modes is shown in Fig. 6. Since adaptivity occurs only in each direction, the mesh is always regular.

### 2.3. Relationship among the DNN, HiDeNN, FEM, TD and HiDeNN-TD approximation function sets

In this subsection, we explore the relationship among FEM, DNN, HiDeNN, TD, and HiDeNN-TD.

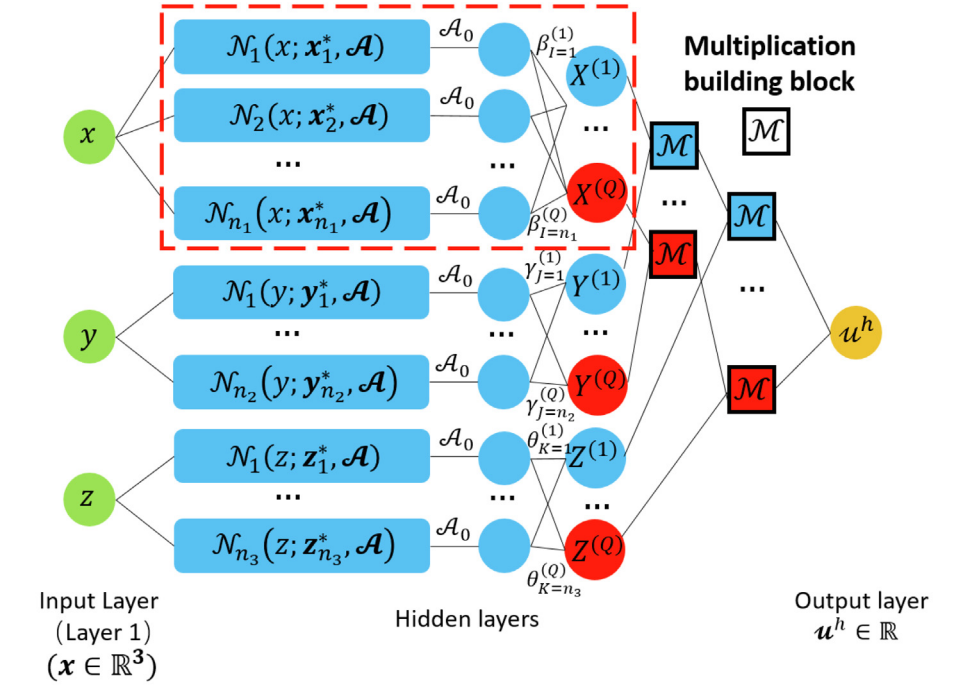



**Fig. 6.** Illustration for the HiDeNN-TD interpolation function with $Q$ modes in the form of DNN.

Assume that TD and FEM are based on the same regular mesh with $n_1, n_2, n_3$ nodes along $x, y, z$ directions. This mesh also serves as an initial guess of $\boldsymbol{x}_I^*, \boldsymbol{y}_J^*, \boldsymbol{z}_K^*$ in HiDeNN and HiDeNN-TD. The shape functions of FEM are the product of 1D shape functions, i.e., the shape function associated with the node $(x_I, y_J, z_K)$ is $N_{(I,J,K)}(x, y, z) = N_I(x)N_J(y)N_K(z)$. DNN has a more general structure than HiDeNN, and might be fully-connected.

By definition, we have the following relationship among approximation function sets of DNN, HiDeNN, constrained HiDeNN with regular mesh, FEM and TD:

$$\mathcal{M}_Q^h \subset \mathcal{V}^h \subset \mathcal{HR}^h \subset \mathcal{H}^h \subset \mathcal{N}^h. \tag{27}$$

Especially when $Q$ is big enough ($Q \geq \min\{n_1, n_2\}$ for 2D and $Q \geq \min\{n_1n_2, n_2n_3, n_3n_1\}$ for 3D), we have

$$\mathcal{M}_Q^h = \mathcal{V}^h \subset \mathcal{G}_Q^h = \mathcal{HR}^h \subset \mathcal{H}^h \subset \mathcal{N}^h, \tag{28}$$

as illustrated in Fig. 7.

The above conclusions (27)–(28) are based on the following observations:

- According to (21), the TD interpolation functions can be regarded as finite element interpolation functions, which belong to $\mathcal{V}^h$, so $\mathcal{M}_Q^h \subset \mathcal{V}^h$. Especially, when $Q$ is big enough ($Q \geq \min\{n_1, n_2\}$ for 2D and $Q \geq \min\{n_1n_2, n_2n_3, n_3n_1\}$ for 3D), $\mathcal{M}_Q^h$ approaches to $\mathcal{V}^h$, i.e., $\mathcal{M}_Q^h = \mathcal{V}^h$. Detailed numerical results will be shown in Section 5. Proofs can be found in Appendix D.

- In HiDeNN, an optimization of the nodal positions is performed. Thus FEM may be regarded as a specific case in HiDeNN with nodal coordinates fixed.

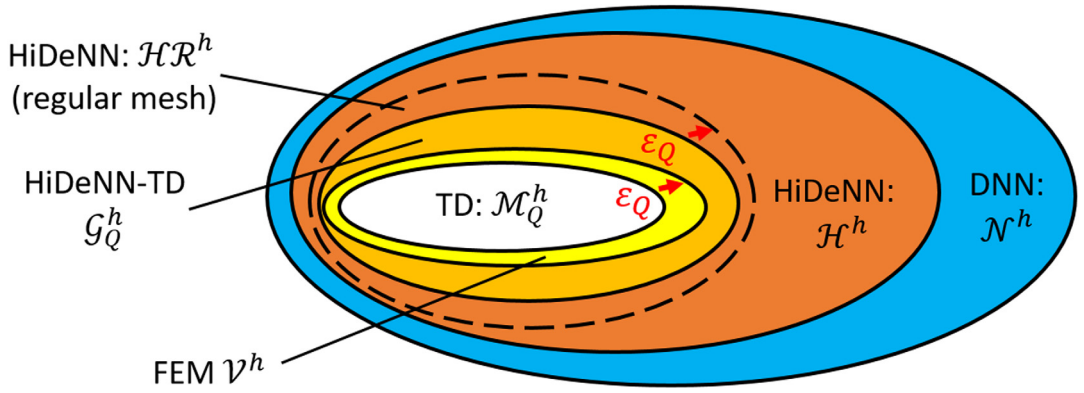



**Fig. 7.** Illustration for relationship among interpolation function sets of TD, FEM, HiDeNN-TD, HiDeNN and DNN. Especially, when $Q$ tends to infinity, $\mathcal{M}_Q^h$ approaches to $\mathcal{V}^h$. When $Q$ is big enough, FEM interpolation set $\mathcal{V}^h$ is the subset of HiDeNN-TD one $\mathcal{G}_Q^h$.

**Table 2**
Comparison of DoFs for different methods on a 3D mesh.

| | FEM | PGD/TD | HiDeNN-TD | HiDeNN (see Remark) |
|---|---|---|---|---|
| DoFs | $n_1 \times n_2 \times n_3$ | $(n_1+n_2+n_3) \times Q$ | $(n_1+n_2+n_3) \times Q + (n_1+n_2+n_3)$ | $n_1 \times n_2 \times n_3 + n_1 \times n_2 \times n_3 \times 3$ |

- HiDeNN is a class of structured DNN with weights and biases as functions of nodal values and nodal positions.
- HiDeNN-TD requires the nodal values in the form of tensor product, which is a reduced-order model of the constrained HiDeNN with regular mesh, so $\mathcal{G}_Q^h \subset \mathcal{HR}^h, \forall Q \in \mathbb{N}$. Especially, when $Q$ is big enough, $\mathcal{G}_Q^h$ approaches to $\mathcal{HR}^h$, i.e., $\mathcal{G}_Q^h = \mathcal{HR}^h$. For details please refer to Appendix E. On the other hand, HiDeNN-TD has slightly more DoFs than TD under the same number of modes, so $\mathcal{M}_Q^h \subset \mathcal{G}_Q^h, \forall Q \in \mathbb{N}$. When $Q$ is small, $\mathcal{M}_Q^h$ is the subset of the intersection of $\mathcal{G}_Q^h$ and $\mathcal{V}^h$.

We can also summarize the DoFs for different methods in Table 2. It is shown that HiDeNN-TD and TD have only a linear growth in terms of DoFs, whereas the DoFs of FEM and HiDeNN may grow in a polynomial manner.

**Remark**: In HiDeNN, the efficiency depends on how the $\boldsymbol{x}^*$ are computed. For most situation, the optimization loop of $\boldsymbol{x}^*$ can be set outside the solution iteration loop that the usual finite element solution schemes such as those available in commercial software can be employed for the solution $u^h$. In this paper, we solve all DoFs together.

## 3. Error analysis of FEM, TD, HiDeNN, DNN-based solutions and HiDeNN-TD for PDEs

We consider a partial differential equation with homogeneous boundary conditions

$$\begin{cases} \mathcal{L}\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{b}(\boldsymbol{x}) = 0 \text{ in } \Omega \subset \mathbb{R}^d, \\ \boldsymbol{u}|_{\partial\Omega} = \boldsymbol{0}, \end{cases} \tag{29}$$

where $\boldsymbol{u}$ denotes an $m$-dimensional vector-valued function in a certain Hilbert space $H(\mathbb{R}^d, \mathbb{R}^m)$, $\boldsymbol{b}$ the source term, $\boldsymbol{x} \in \mathbb{R}^d$ the $d$-dimensional space coordinates, and $\mathcal{L}$ a second-order differential operator.

We assume that an energy potential $\Pi[u]$ exists, formulated in the following form

$$\Pi[\boldsymbol{u}] = \frac{1}{2}a(\boldsymbol{u}, \boldsymbol{u}) - (\boldsymbol{b}, \boldsymbol{u}), \tag{30}$$

where $a(\cdot, \cdot)$ is the symmetric bilinear form corresponding to the second-order differential operator $\mathcal{L}$, and $(\boldsymbol{f}, \boldsymbol{g}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{g} \mathrm{d}\boldsymbol{x}$ denotes the inner product. For example, let $a(\boldsymbol{f}, \boldsymbol{g}) = \int_\Omega \nabla \boldsymbol{f} \cdot \nabla \boldsymbol{g} \mathrm{d}\boldsymbol{x}$ for Poisson equation. The minimization of $\Pi[u]$ gives the solution to (29), which reads

$$\boldsymbol{u} = \underset{\boldsymbol{u}^* \in H(\mathbb{R}^d, \mathbb{R}^m)}{\arg\min} \Pi[\boldsymbol{u}^*]. \tag{31}$$

Such a weak form is commonly adopted in interpolation theory based numerical approaches, such as the methods shown in Section 2. Denoted by $\mathcal{S}^h \subset H(\mathbb{R}^d, \mathbb{R}^m)$ the discretized approximation solution set with a characteristic mesh size $h$, the approximate solution based on this given interpolation function set is then

$$\boldsymbol{u}^h = \underset{\boldsymbol{u}^{h*} \in \mathcal{S}^h}{\arg\min}\, \Pi[\boldsymbol{u}^{h*}]. \tag{32}$$

In the following, we shall take $\mathcal{S}^h$ to be $\mathcal{M}^h_{0,Q}, \mathcal{V}^h_0, \mathcal{HR}^h_0, \mathcal{H}^h_0, \mathcal{N}^h_0, \mathcal{G}^h_{0,Q}$, which are the subsets of $\mathcal{M}^h_Q, \mathcal{V}^h, \mathcal{HR}^h, \mathcal{H}^h, \mathcal{N}^h, \mathcal{G}^h_Q$ under homogeneous boundary conditions, respectively.

### 3.1. Error analysis

For simplicity, we confine ourselves with scalar $u \in H(\mathbb{R}^d, \mathbb{R})$. We assert the following relations among error bounds for FEM, DNN, HiDeNN, constrained HiDeNN with regular mesh and TD

$$\begin{aligned} &\left\|u^{TD} - u^{exact}\right\|_E \geq \left\|u^{FEM} - u^{exact}\right\|_E \geq \left\|u^{HiDeNN-Regular} - u^{exact}\right\|_E \\ &\geq \left\|u^{HiDeNN} - u^{exact}\right\|_E \geq \left\|u^{DNN} - u^{exact}\right\|_E. \end{aligned} \tag{33}$$

Here, $\|\cdot\|_E = \sqrt{a(\cdot,\cdot)}$ is called as the energy norm, and $u^{exact}$ is the exact solution of the problem, i.e.,

$$u^{exact} = \underset{u^* \in H(\mathbb{R}^d, \mathbb{R})}{\arg\min}\ \Pi[u^*]. \tag{34}$$

In the numerical tests, it can be analytical or a very fine mesh FEM solution.

Especially when $Q$ is big enough ($Q \geq \min\{n_1, n_2\}$ for 2D and $Q \geq \min\{n_1n_2, n_2n_3, n_3n_1\}$ for 3D), the error bounds for six methods become

$$\begin{aligned} &\left\|u^{TD} - u^{exact}\right\|_E \geq \left\|u^{FEM} - u^{exact}\right\|_E \geq \left\|u^{HiDeNN-TD} - u^{exact}\right\|_E \\ &\geq \left\|u^{HiDeNN-Regular} - u^{exact}\right\|_E \geq \left\|u^{HiDeNN} - u^{exact}\right\|_E \geq \left\|u^{DNN} - u^{exact}\right\|_E. \end{aligned} \tag{35}$$

(33) and (35) can be derived theoretically as below. The relationship (27) is inherited by $\mathcal{M}^h_{0,Q}, \mathcal{V}^h_0, \mathcal{HR}^h_0, \mathcal{H}^h_0, \mathcal{N}^h_0$, i.e.,

$$\mathcal{M}^h_{0,Q} \subset \mathcal{V}^h_0 \subset \mathcal{HR}^h_0 \subset \mathcal{H}^h_0 \subset \mathcal{N}^h_0. \tag{36}$$

These five methods are all based on the minimal energy principle,

$$u = \underset{u^{h*} \in \mathcal{S}^h}{\arg\min}\, \Pi[u^{h*}], \tag{37}$$

where $\mathcal{S}^h$ is selected as $\mathcal{M}^h_{0,Q}, \mathcal{V}^h_0, \mathcal{HR}^h_0, \mathcal{H}^h_0, \mathcal{N}^h_0$ for TD, FEM, constrained HiDeNN with regular mesh, HiDeNN, and DNN, respectively. Thus due to the relationship (36) among them, we have

$$\Pi[u^{TD}] \geq \Pi[u^{FEM}] \geq \Pi[u^{HiDeNN-Regular}] \geq \Pi[u^{HiDeNN}] \geq \Pi[u^{DNN}]. \tag{38}$$

This leads to the error bounds (33). In the same manner, the relationship (28) leads to the error bounds (35). (35) shows that HiDeNN-TD might reach better accuracy than FEM with regular mesh at increasing number of modes. Especially, considering the known proof of the convergence of FEM, HiDeNN-TD is able to reach the same convergence with enough number of modes, i.e., converges to the exact solution with refining mesh.

We remark that the above theoretical analysis does not account for numerical aspects, such as solution schemes to determine the global minimizer of the energy potential. Some methods such as DNN, HiDeNN, and HiDeNN-TD might lose some accuracy due to optimization algorithms. For practical numerical verifications in Section 5, we use a very tight tolerance to ensure the accuracy of the results, and confirm our theoretical analysis inequalities.

### 3.2. Proof of the error decomposition in the TD method

The TD based model reduction induces two kinds of errors: mesh discretization error and mode reduction error.

**Theorem 1.** *Let $u^{exact}$, $u^{TD}$ and $u^{FEM}$ be the exact solution, the numerical solution of TD and FEM, respectively. TD and FEM take the same regular mesh, and FEM takes the shape functions as the product of 1D shape functions of TD in each dimension. Then the following error decomposition holds:*

$$\left(\|u^{TD}-u^{exact}\|_E\right)^2=\left(\|u^{FEM}-u^{exact}\|_E\right)^2+\left(\|u^{TD}-u^{FEM}\|_E\right)^2. \tag{39}$$

**Proof.** We calculate

$$\begin{aligned}&\left(\|u^{TD}-u^{exact}\|_E\right)^2 &(40)\\ =&\left(\|u^{TD}-u^{FEM}+u^{FEM}-u^{exact}\|_E\right)^2 &(41)\\ =&\left(\|u^{FEM}-u^{exact}\|_E\right)^2+\left(\|u^{TD}-u^{FEM}\|_E\right)^2+2a\left(u^{TD}-u^{FEM},u^{FEM}-u^{exact}\right).\end{aligned}$$

By the Gauss theorem, we obtain

$$\begin{aligned}&a\left(u^{TD}-u^{FEM},u^{FEM}-u^{exact}\right) &(42)\\ =&a\left(u^{TD}-u^{FEM},u^{FEM}\right)-a\left(u^{TD}-u^{FEM},u^{exact}\right)\\ =&a\left(u^{TD}-u^{FEM},u^{FEM}\right)+\left(u^{TD}-u^{FEM},\mathcal{L}u^{exact}\right)\\ =&a\left(u^{TD}-u^{FEM},u^{FEM}\right)-\left(u^{TD}-u^{FEM},b\right).\end{aligned}$$

Since TD and FEM share the same mesh and shape functions, $v=u^{TD}-u^{FEM}$ belongs to the test function space of FEM. By the weak form of the FEM problem, we have

$$a\left(v,u^{FEM}\right)-(v,b)=0. \tag{43}$$

That is to say, (42) vanishes and hence (39) holds. □

This theorem asserts that the TD error is a direct sum of the mesh discretization error (the FEM error) and the mode reduction error (the difference between FEM and TD). The same conclusion also applies to PGD.

## 4. The formulation of HiDeNN-TD: the 2D Poisson problem as illustration

In Subsection 2.2, we defined HiDeNN-TD in terms of the approximation function space. Here, we give the detailed formulation of the method.

For the sake of simplicity and without loss of generality, we consider 2D Poisson problem,

$$\begin{cases}\Delta u(x,y)+b(x,y)=0 \text{ in } \Omega_{(x,y)}\subset\mathbb{R}^2,\\ u|_{\partial\Omega}=0.\end{cases} \tag{44}$$

(44) is solved in the regular domain $\Omega_{(x,y)}=[a,b]\times[c,d]$ with homogeneous boundary conditions. Note that if inhomogeneous boundary conditions are under consideration, we can separate the solution into two parts,

$$u=u^0+\tilde{u}, \tag{45}$$

where $u^0$ is an arbitrary function satisfying boundary conditions, and $\tilde{u}$ is the solution to the new equation with homogeneous boundary condition.

The variational form of (44) is

$$\Pi[u]=\frac{1}{2}\int_{\Omega_{(x,y)}}|\nabla u|^2\,\mathrm{d}x\mathrm{d}y-\int_{\Omega_{(x,y)}}u(x,y)b(x,y)\mathrm{d}x\mathrm{d}y. \tag{46}$$

Substituting HiDeNN-TD interpolation function into (46), we obtain

$$\begin{aligned}\Pi[u^h]=&\frac{1}{2}\sum_{p=1}^{Q}\sum_{q=1}^{Q}\left(\int_{x_1}^{x_{n_1}}\frac{d}{dx}X^{(p)}(x)\frac{d}{dx}X^{(q)}(x)\mathrm{d}x\right)\left(\int_{y_1}^{y_{n_2}}Y^{(p)}(y)Y^{(q)}(y)\mathrm{d}y\right) &(47)\\ &+\frac{1}{2}\sum_{p=1}^{Q}\sum_{q=1}^{Q}\left(\int_{x_1}^{x_{n_1}}X^{(p)}(x)X^{(q)}(x)\mathrm{d}x\right)\left(\int_{y_1}^{y_{n_2}}\frac{d}{dy}Y^{(p)}(y)\frac{d}{dy}Y^{(q)}(y)\mathrm{d}y\right)\end{aligned}$$

$$- \sum_{q=1}^{Q} \int_{\Omega_{(x,y)}} X^{(q)}(x) Y^{(q)}(y) b(x,y) \mathrm{d}x\mathrm{d}y,$$

with the discrete mesh $[x_1 = a, x_2, \ldots, x_{n_1} = b] \times [y_1 = c, y_2, \ldots, y_{n_2} = d]$. Notice that there exist cross terms in (47). For convenience and considering the difficulties to do exact integration between different discrete meshes, all the modes share the same mesh $[x_1, x_2, \ldots, x_{n_1}] \times [y_1, y_2, \ldots, y_{n_2}]$ and the same shape functions. We use Gauss quadrature for the source term.

Once the interpolation function set (26) is obtained, variational principle gives the approximate solution. The process in HiDeNN-TD is formulated as

$$\begin{aligned}
&\text{find } \beta_I^{(1)}, \gamma_J^{(1)}, \ldots, \beta_I^{(Q)}, \gamma_J^{(Q)}, x_I, y_J, I = 1, \ldots, n_1, J = 1, \ldots, n_2 \\
&\min\ \Pi[u^h] = \frac{1}{2} \int_{\Omega_{(x,y)}} \left|\nabla u^h\right|^2 \mathrm{d}x\mathrm{d}y - \int_{\Omega_{(x,y)}} u^h(x,y) b(x,y) \mathrm{d}x\mathrm{d}y \\
&\quad u^h = \sum_{q=1}^{Q} \left( \sum_{I=1}^{n_1} \mathcal{N}_I(x; \boldsymbol{x}_I^*, \boldsymbol{\mathcal{A}}) \beta_I^{(q)} \right) \left( \sum_{J=1}^{n_2} \mathcal{N}_J(y; \boldsymbol{y}_J^*, \boldsymbol{\mathcal{A}}) \gamma_J^{(q)} \right) \\
&\text{and } \sum_{I=1}^{n_1} \mathcal{N}_I(x; \boldsymbol{x}_I^*, \boldsymbol{\mathcal{A}}) = 1, \sum_{J=1}^{n_2} \mathcal{N}_J(y; \boldsymbol{y}_J^*, \boldsymbol{\mathcal{A}}) = 1.
\end{aligned} \tag{48}$$

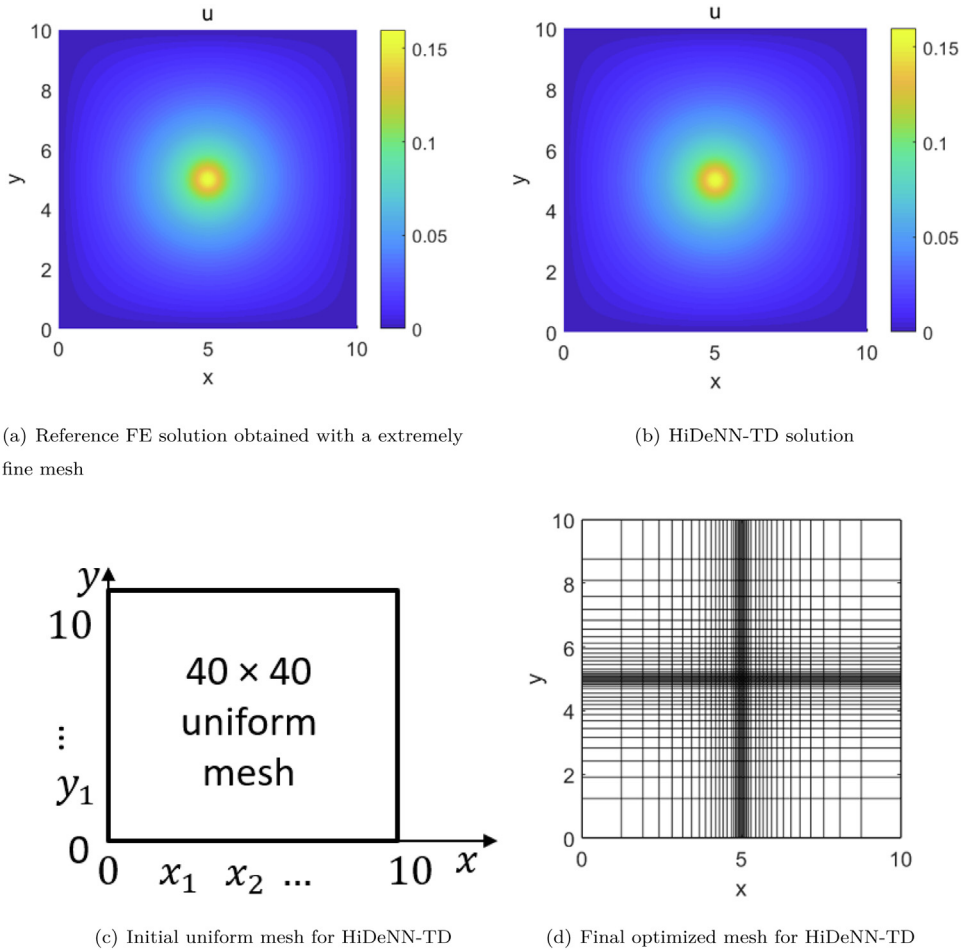
The gradient descent method is applied to iteratively minimize $\Pi[u^h]$ and solve for all parameters together. In the following numerical examples, we choose Adam algorithm [40], i.e.,

1. Initialization: Set number of modes $Q$, initial nodal positions $x_I, y_J, I = 1, 2, \ldots, n_1, J = 1, 2, \ldots, n_2$, initial coefficients $\beta_I^{(q)}, \gamma_J^{(q)}, q = 1, 2, \ldots, Q$ and maximal iteration step $M$
2. Algorithm:
   While $k \leq M$ do
   (a) Compute gradient $\frac{\partial}{\partial x_I}, \frac{\partial}{\partial y_J}, \frac{\partial}{\partial \beta_I^{(q)}}, \frac{\partial}{\partial \gamma_J^{(q)}}, I = 1, \ldots, n_1, J = 1, \ldots, n_2, q = 1, \ldots, Q$;
   (b) Update $x_I, y_J, \beta_I^{(q)}, \gamma_J^{(q)}, I = 1, \ldots, n_1, J = 1, \ldots, n_2, q = 1, \ldots, Q$ by using Adam algorithm (keep the order of the nodal coordinates in each dimension, i.e., $a = x_1 < x_2 < \cdots < x_{n_1} = b, c = y_1 < y_2 < \cdots < y_{n_2} = d$, in the updating process);
   End while.

Remark that the solution scheme to solve (48) is flexible. HiDeNN-TD reduces to TD with fixed nodal coordinates. The nodal coordinates do not need to be optimized at all times during the computations. In most cases, the coordinates converge more quickly than the nodal solutions. Hence, depending on the problem and the accuracy we need, we can design the most efficient algorithm by switching between HiDeNN-TD and TD/PGD during the solution steps.

## 5. Numerical examples

In this section, we study the performance of HiDeNN-TD with comparison to FEM, HiDeNN and PGD/TD methods. This study mainly focuses on the accuracy comparison and the convergence behavior, as the computational cost can be strongly affected by the under-optimized implementation. Remark that in the following numerical examples, we set a very tight tolerance $10^{-11}$ to ensure the convergence of these methods to verify our theoretical studies, which might result in the increase in numerical costs. In practice, this tolerance can be relaxed for better performance. The computational efficiency of the proposed HiDeNN-TD method will be investigated more in our future work on the basis of GPU-computing.




(a) Reference FE solution obtained with a extremely fine mesh

(b) HiDeNN-TD solution

(c) Initial uniform mesh for HiDeNN-TD

(d) Final optimized mesh for HiDeNN-TD

**Fig. 8.** FE solution versus HiDeNN-TD solution.

### *5.1. 2D case*

The HiDeNN-TD is applied to the Poisson problem with a concentrated load

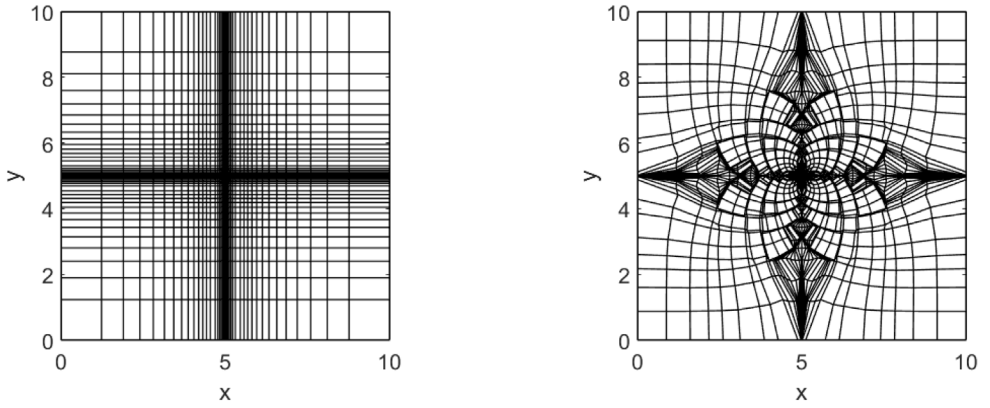
$$b(x, y) = \exp\left(-10(x-5)^2 - 10(y-5)^2\right) \tag{49}$$

in the domain $\Omega = [0, 10] \times [0, 10]$. To demonstrate the capability of the method, the domain analyzed by HiDeNN-TD is initialized with a uniform mesh of 40 by 40 elements. As shown in Fig. 8, the final solution agrees with the reference FE solution. This reference solution is obtained with a very fine mesh containing $4,000 \times 4,000$ elements.

To measure the accuracy, we define the relative energy norm error for Poisson problem,

$$error = \frac{\|u^h - u^{ref}\|_E}{\|u^{ref}\|_E} = \frac{\sqrt{\int_\Omega |\nabla(u^h - u^{ref})|^2 \mathrm{d}\boldsymbol{x}}}{\sqrt{\int_\Omega |\nabla u^{ref}|^2 \mathrm{d}\boldsymbol{x}}} \tag{50}$$

Table 3 illustrates the evolution of accuracy with an increasing number of modes. For comparison purposes, we also applied the PGD, TD, FEM and HiDeNN for the same problem on the uniform mesh of 40 by 40. As expected, HiDeNN is the most accurate one but leads to a larger number of DoFs. The proposed HiDeNN-TD method can have the same level of accuracy and only requires a small number of modes. Compared to PGD and TD, the HiDeNN-TD is much more accurate at a limited number of modes. Taking a closer look at PGD and TD, these two methods converge overall to the FEM on the coarse mesh. HiDeNN-TD and HiDeNN can overcome this limitation imposed




(a) Inital mesh for HiDeNN (b) Final optimized mesh for HiDeNN

**Fig. 9.** Mesh optimization in HiDeNN.

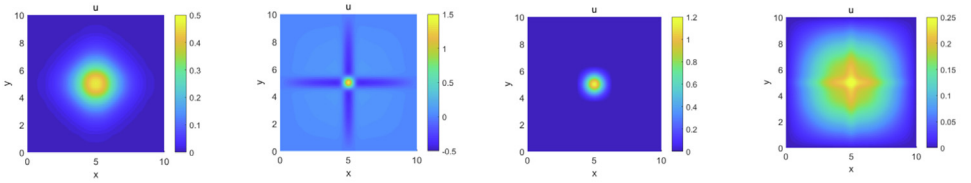
**Table 3**
Accuracy comparison for different methods on the coarse 40 by 40 mesh.

| Mode number | PGD | | TD | | FEM | | HiDeNN-TD | | HiDeNN (Regular mesh) | | HiDeNN | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error |
| 1 | 78 | 38.167% | 78 | 38.167% | **1521** | **11.659%** | 156 | 37.357% | **1599** | **3.660%** | **4719** | **2.102%** |
| 2 | 156 | 16.500% | 156 | 14.422% | | – | **234** | **9.293%** | | – | | – |
| 3 | 234 | 13.188% | 234 | 11.789% | | – | 312 | 3.674% | | – | | – |
| 4 | 312 | 11.811% | 312 | 11.664% | | – | 390 | 3.662% | | – | | – |
| 5 | 390 | 11.685% | **390** | **11.659%** | | – | 468 | 3.661% | | – | | – |
| 6 | 468 | 11.666% | 468 | 11.659% | | – | 546 | 3.661% | | – | | – |
| 8 | **624** | **11.659%** | | – | | – | | – | | – | | – |
| 20 | 1560 | 11.659% | | – | | – | | – | | – | | – |

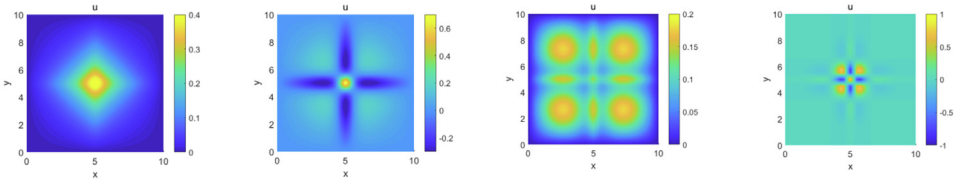
by the mesh size with the adaptivity. This observation is consistent with our theoretical analysis. HiDeNN-TD and TD take fewer modes than PGD to converge. When number of modes is bigger than 4, the error of HiDeNN-TD reduces less than 0.002% with an external mode added. Meanwhile, the difference of the errors of PGD with 5 and 6 modes is 0.019%. This is due to different solution schemes of PGD, TD and HiDeNN-TD. PGD adopts a greedy manner to solve modes, while TD and HiDeNN-TD realize the global optimization with a given number of modes. Moreover, it should be noticed that, unlike HiDeNN, the HiDeNN-TD only increases slightly the DoFs when compared with PGD and TD. This attributes to the separation of variables. Indeed, the mesh adaptation is performed only in the two separated axes, as shown in Fig. 8(d). For comparison purposes, the final optimized mesh of HiDeNN is illustrated in Fig. 9. It is shown that the HiDeNN enables a full adaptivity for the entire mesh, which leads to a significantly different and more accurate final result. Note that we use Jacobian to control the mesh quality. Nevertheless, the HiDeNN-TD shows attractive advantages in terms of DoFs.

Fig. 10 illustrates the first four modes of HiDeNN-TD, PGD, and TD after scaling. The PGD modes remain similar to TD in this example. However, the modes of HiDeNN-TD seem to be more concentrated in the region of interest. This difference mainly comes from the mesh adaptivity.

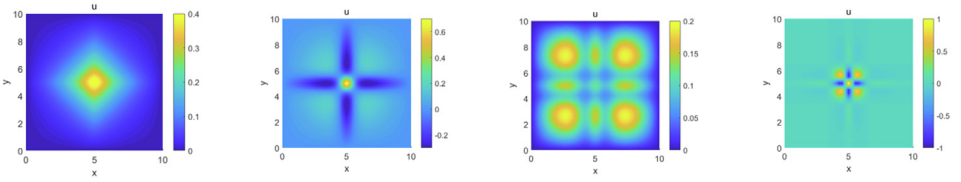
To further confirm the performance of HiDeNN-TD, we compare the accuracy of different methods on different meshes. In Table 4, the PGD, TD and HiDeNN-TD results are obtained from the final converged mode. It is shown that the HiDeNN-TD always gives more accurate results with fewer degrees of freedom. This confirms the previous observation. From the point of view of mesh refinement, the error of all methods decreases. We shall discuss the convergence rates of different methods with respect to mesh size in the following subsubsection.



(a) First four modes in HiDeNN-TD



(b) First four modes in PGD



(c) First four modes in TD

**Fig. 10.** Mode comparison for HiDeNN-TD, PGD and TD.

**Table 4**
Accuracy comparison for different methods with different meshes.

| Mesh | PGD | | TD | | FEM | | HiDeNN-TD | | HiDeNN (Regular Mesh) | | HiDeNN | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error |
| 40 × 40 | 624(8) | 11.659% | 390(5) | 11.659% | **1,521** | **11.659%** | **468**(5) | **3.661%** | **1,599** | **3.660%** | **4,719** | **2.102%** |
| 80 × 80 | 1,422(9) | 5.887% | 790(5) | 5.887% | 6,241 | 5.887% | 1,106(6) | 1.851% | 6,399 | 1.847% | 19,039 | 1.406% |
| 160 × 160 | 2,862(9) | 2.948% | 1,908(6) | 2.948% | 25,281 | 2.948% | 2,226(6) | 1.174% | 25,599 | 1.173% | 76,479 | 1.081% |
| 320 × 320 | 7,018(11) | 1.469% | 5,104(8) | 1.469% | 101,761 | 1.469% | 3,828(5) | 0.896% | 102,399 | 0.894% | 306,559 | 0.889% |
| 640 × 640 | 11,502(9) | 0.724% | 7,668(6) | 0.724% | 408,321 | 0.724% | 10,224(7) | 0.606% | 409,599 | 0.606% | 1,227,519 | 0.597% |

Note that the number in the parentheses indicates the number of modes.

### *5.1.1. Convergence studies*

In the HiDeNN-TD method, the mode number has to be prescribed. In general, this is unknown for a given problem and can vary significantly from one to another. Thus, we want to study the convergence property of this method and the PGD to get a general idea about how to choose the mode number.

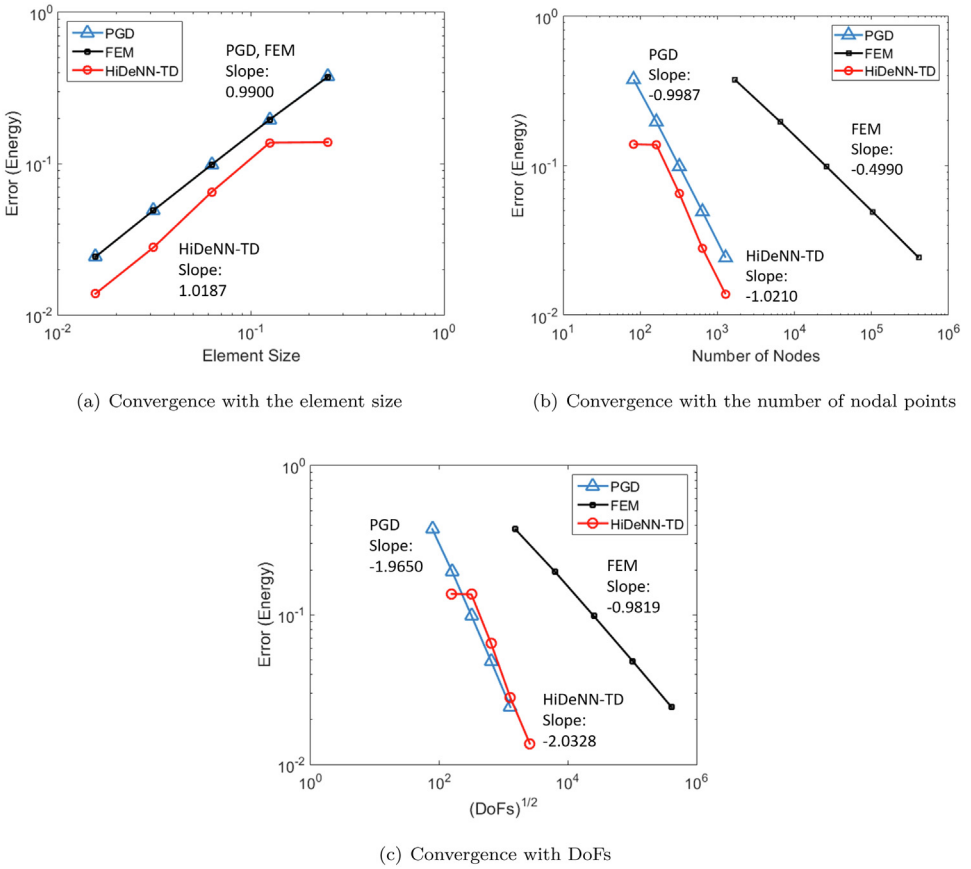
As a first attempt, we restrict ourselves to a one-mode solution problem. Let the body force

$$\begin{aligned} b(x,y) = &\left(20-400(x-5)^2\right)\exp\left(-10(x-5)^2\right)\left(\exp\left(-10(y-5)^2\right)-\exp(-250)\right) \\ &+\left(\exp\left(-10(x-5)^2\right)-\exp(-250)\right)\left(20-400(y-5)^2\right)\exp\left(-10(y-5)^2\right), \end{aligned} \tag{51}$$

and the analytical solution

$$u(x,y) = \left(\exp\left(-10(x-5)^2\right)-\exp(-250)\right)\left(\exp\left(-10(y-5)^2\right)-\exp(-250)\right). \tag{52}$$

This eliminates the effect of the number of modes, and allows us to study the convergence rate of the methods with respect to mesh refinement. This kind of error is usually known as discretization error in FEM. To do so, the body force term is manufactured so that the final solution is analytically known in a separated form. As shown in Fig. 11,




(a) Convergence with the element size

(b) Convergence with the number of nodal points

(c) Convergence with DoFs

**Fig. 11.** Convergence rate of FEM, PGD, HiDeNN-TD with respect to mesh refinement.

the PGD and HiDeNN-TD results converge with respect to the element size at a rate similar to FEM. However, if we consider the DoFs or number of nodal points, the convergence is much faster for PGD and HiDeNN-TD. This confirms that doing separation of variables for PGD and HiDeNN-TD does not degrade the convergence rate in terms of mesh refinement.

The PGD based model reduction induces two kinds of errors: mesh discretization error and mode reduction error. In particular, we have theoretically showed this decomposition in Subsection 3.2 and numerically observed that the latter one seems independent of the mesh. In order to show this numerically, we use a manufactured load (49) to compute the multi-modes PGD solution illustrated in the previous examples in different meshes. The results are shown in Fig. 12. The convergence rates on the number of modes remain similar regardless of the mesh size. It seems that the log error is linearly proportional to the mode number, consistent with the exponential convergence rate proof in [14]. The decreasing slope remained unchanged from the very coarse mesh to the fine one. This implies the coarse mesh has the same mode reduction error as a fine mesh. Furthermore, we studied this by curve fitting, and obtained the following formulas,

$$e^{PGD-FEM} = \frac{\|u^{PGD} - u^{FEM}\|_E}{\|u^{ref}\|_E} \approx c_1 \exp\left(-Q/\tau\right), e^{FEM} = \frac{\|u^{FEM} - u^{ref}\|_E}{\|u^{ref}\|_E} \approx c_2 h, \tag{53}$$

where $h$ is the element size, and parameters $c_1 \approx 0.6893, c_2 \approx 0.4714, \tau \approx 1.1793$ are obtained by fitting. Combining with error decomposition equation (39), we have

$$e^{PGD} = \frac{\|u^{PGD} - u^{ref}\|_E}{\|u^{ref}\|_E} = \sqrt{(e^{PGD-FEM})^2 + (e^{FEM})^2} \approx \sqrt{(c_1 \exp\left(-Q/\tau\right))^2 + (c_2 h)^2} \tag{54}$$

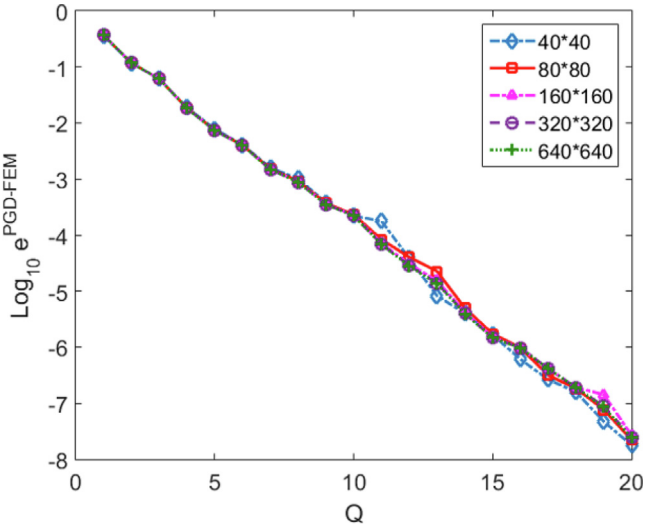



**Fig. 12.** Convergence of PGD with respect to the increasing number of modes for different meshes.

This can be used to choose the mode number. Note that this formula is obtained by curve fitting rather than any theoretical proof, and might be problem-dependent. We are exploring a general function relating the error to the number of modes and mesh size, and seeking for the theoretical support.

From the above observation, we may consider using a coarse mesh PGD, which is very cheap, to study the mode reduction error and choose an appropriate mode number for HiDeNN-TD. Since the HiDeNN-TD is always more accurate than the usual PGD, the selected prescribed number should be large enough.

### 5.2. 3D Case

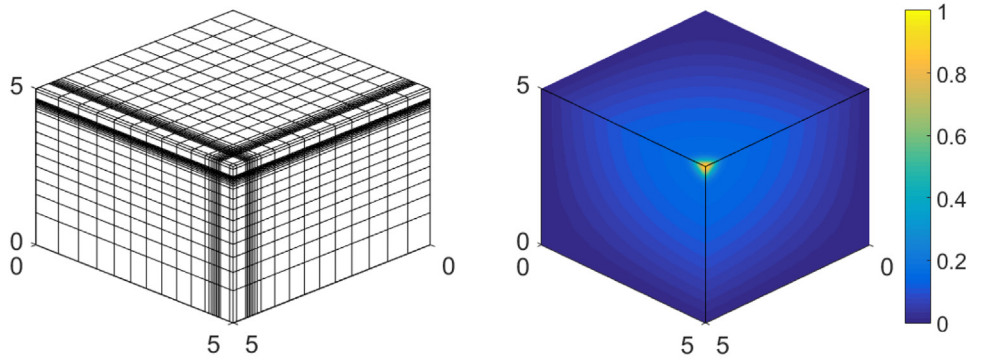
The proposed HiDeNN-TD method has been tested in three dimensional cases. We take the body force as

$$
\begin{aligned}
&b(x,y,z) \\
&= \left(20-400(x-5)^2\right)\exp\left(-10(x-5)^2\right)\left(\exp\left(-10(y-5)^2\right)-\exp(-250)\right) \\
&\quad\times\left(\exp\left(-10(z-5)^2\right)-\exp(-250)\right) \\
&\quad+\left(\exp\left(-10(x-5)^2\right)-\exp(-250)\right)\left(20-400(y-5)^2\right) \\
&\quad\times\exp\left(-10(y-5)^2\right)\left(\exp\left(-10(z-5)^2\right)-\exp(-250)\right) \\
&\quad+\left(\exp\left(-10(x-5)^2\right)-\exp(-250)\right)\left(\exp\left(-10(y-5)^2\right)-\exp(-250)\right) \\
&\quad\times\left(20-400(z-5)^2\right)\exp\left(-10(z-5)^2\right) \\
&\quad+\frac{3\pi^2}{1000}\sin(\frac{\pi x}{10})\sin(\frac{\pi y}{10})\sin(\frac{\pi z}{10})-\frac{1}{3125}x(x-10)\sin(\frac{\pi z}{10})-\frac{1}{3125}y(y-10)\sin(\frac{\pi z}{10}) \\
&\quad+\frac{\pi^2}{625000}x(x-10)y(y-10)\sin(\frac{\pi z}{10})
\end{aligned}
\tag{55}
$$

and the analytical solution as

$$
\begin{aligned}
&u(x,y,z) \\
&= \left(\exp\left(-10(x-5)^2\right)-\exp(-250)\right)\left(\exp\left(-10(y-5)^2\right)-\exp(-250)\right)\left(\exp\left(-10(z-5)^2\right)-\exp(-250)\right) \\
&\quad+\frac{1}{10}\sin(\frac{\pi x}{10})\sin(\frac{\pi y}{10})\sin(\frac{\pi z}{10})+\frac{1}{6250}x(x-10)y(y-10)\sin(\frac{\pi z}{10}).
\end{aligned}
\tag{56}
$$

Fig. 13 presents the final optimized mesh with $40\times40\times40$ elements and solution for HiDeNN-TD. Similar to the previous two-dimensional example, Table 5 reports the evolution of error at an increasing mode number for




(a) Final optimized mesh for HiDeNN-TD (b) HiDeNN-TD solution

**Fig. 13.** Final optimized mesh and solution for HiDeNN-TD in one-eighth part of the 3D computational domain.

**Table 5**
Accuracy comparison for different methods on the coarse $40 \times 40 \times 40$ mesh.

| Mode number | PGD | | TD | | FEM | | HiDeNN-TD | | HiDeNN (Regular Mesh) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error |
| 1 | 117 | 72.474% | 117 | 72.474% | **59,319** | **27.870%** | 234 | 72.349% | **59,436** | **10.780%** |
| 2 | 234 | 29.883% | 234 | 27.927% | | – | **351** | **10.797%** | | – |
| 3 | 351 | 28.026% | 351 | 27.923% | | – | 468 | 10.797% | | – |
| 4 | 468 | 27.931% | 468 | 27.920% | | – | 585 | 10.797% | | – |
| 5 | 585 | 27.895% | 585 | 27.872% | | – | 702 | 10.796% | | – |
| 6 | 702 | 27.881% | 702 | 27.871% | | – | – | – | | – |
| 7 | 819 | 27.877% | 819 | 27.871% | | – | – | – | | – |
| 8 | 936 | 27.874% | **936** | **27.870%** | | – | – | – | | – |
| 17 | **1,989** | **27.870%** | | – | | – | | – | | – |
| 20 | 2,340 | 27.870% | | – | | – | – | – | | – |

**Table 6**
Accuracy comparison for different methods with different meshes for 3D problem.

| Mesh | PGD | | TD | | FEM | | HiDeNN-TD | | HiDeNN (Regular Mesh) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error | DoFs | Error |
| $40 \times 40 \times 40$ | 1,989(17) | 27.870% | 936(8) | 27.870% | **59,319** | **27.870%** | **702**(5) | **10.796%** | **59,436** | **10.780%** |
| $80 \times 80 \times 80$ | 2,607(11) | 14.416% | 1,185(5) | 14.416% | 493,039 | 14.416% | 1,422(5) | 6.770% | 493,276 | 6.770% |
| $160 \times 160 \times 160$ | 4,770(10) | 7.247% | 2,385(5) | 7.247% | 4,019,679 | 7.247% | 2,862(5) | 4.036% | 4,020,156 | 3.984% |
| $320 \times 320 \times 320$ | 9,570(10) | 3.628% | 5,742(6) | 3.628% | 32,461,759 | 3.628% | 5,742(5) | 1.831% | 32,462,716 | 1.830% |

different methods on a coarse mesh. Again, the HiDeNN-TD outperforms the other methods in terms of accuracy and DoFs. The same conclusion can be drawn on finer meshes, as reported in Table 6.

### *5.3. Discussions*

This paper focuses on the theoretical aspect of HiDeNN-TD. We set a very small tolerance ($10^{-11}$) to get the converged solutions in previous examples (see Tables 3 to 6), and thus the solution time may not be optimal. In practical cases, a trade-off between accuracy and efficiency is usually made.

Table 7 shows the computational time of different methods, including HiDeNN-PGD. The basic idea is to start from PGD solutions and then use HiDeNN to further improve the solution accuracy by optimizing the mesh coordinates. The details will be explained below. The computations are performed in Matlab on an Intel Xeon e5-2650 v2 processor. We use 6 Gaussian points in each dimension of elements to ensure the quadrature accuracy.

**Table 7**
Computational cost comparison for different methods with different meshes for 3D problem (for PGD and TD, the number of modes (converged modes) needed to get the same accuracy of the FEM is reported next to the CPU time). Here, HiDeNN-PGD adopts a PGD and Newton–Raphson method hybrid strategy.

| Mesh | PGD (ADS, Tol $10^{-3}$ ) | | TD (ADS, Tol $10^{-3}$ ) | | FEM (PCG solver) | | HiDeNN-PGD (Hybrid) $\epsilon = 10^{-1}$ | | HiDeNN-PGD (Hybrid) $\epsilon = 10^{-2}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Time (s) | Error | Time (s) | Error | Time (s) | Error | Time (s) | Error | Time (s) | Error |
| $40 \times 40 \times 40$ | **1.7346**(17) | **27.870%** | **0.65338**(8) | **27.870%** | **444.15** | **27.870%** | **1.5210**(5) | **12.263%** | 4.8016(5) | 11.425% |
| $80 \times 80 \times 80$ | 0.57950(11) | 14.416% | 0.40189(5) | 14.417% | 3,542.3 | 14.416% | 1.0106(5) | 10.997% | 9.2530(5) | 7.336% |
| $160 \times 160 \times 160$ | 0.55331(10) | 7.247% | 0.47499(5) | 7.247% | 36,503 | 7.247% | 1.8125(5) | 6.512% | 36.063(5) | 4.425% |
| $320 \times 320 \times 320$ | 1.2607(10) | 3.628% | 5.9524(6) | 3.628% | 223,607 | 3.628% | 6.1171(5) | 3.492% | 6.2287(5) | 3.492% |

As for solution schemes, both PGD and TD adopt a standard alternating direction strategy (ADS) that computes one coefficient at a time while fixing the others. The difference between PGD and TD is that PGD solves one mode at a time whereas TD solves all the modes simultaneously with a given number of modes, and the PGD solutions as the initial guess. The convergence criterion for each mode of PGD is

$$\left( \frac{1}{n_1 - 2} \sum_{I=2}^{n_1-1} (\beta_I^{(q)(k)} - \beta_I^{(q)(k-1)})^2 + \frac{1}{n_2 - 2} \sum_{J=2}^{n_2-1} (\gamma_J^{(q)(k)} - \gamma_J^{(q)(k-1)})^2 + \frac{1}{n_3 - 2} \sum_{K=2}^{n_3-1} (\theta_K^{(q)(k)} - \theta_K^{(q)(k-1)})^2 \right)^{\frac{1}{2}} < 10^{-3} \tag{57}$$

with $q$ counting index for modes and $k$ the iteration step, and that for TD is

$$\begin{aligned} &\Big( \frac{1}{(n_1 - 2)Q} \sum_{q=1}^{Q} \sum_{I=2}^{n_1-1} (\beta_I^{(q)(k)} - \beta_I^{(q)(k-1)})^2 + \frac{1}{(n_2 - 2)Q} \sum_{q=1}^{Q} \sum_{J=2}^{n_2-1} (\gamma_J^{(q)(k)} - \gamma_J^{(q)(k-1)})^2 \\ &+ \frac{1}{(n_3 - 2)Q} \sum_{q=1}^{Q} \sum_{K=2}^{n_3-1} (\theta_K^{(q)(k)} - \theta_K^{(q)(k-1)})^2 \Big)^{\frac{1}{2}} < 10^{-3}. \end{aligned} \tag{58}$$

FEM solutions are obtained by the preconditioned conjugate gradients method (PCG) as it is not possible to solve a very large problem with a direct matrix solver with limited computational memory.

For improving the efficiency of HiDeNN-TD, we developed a specific solution strategy, called HiDeNN-PGD, which can be seen as a further trade-off between PGD and HiDeNN-TD. The key idea is to separately solve the modes using PGD and then optimize the mesh coordinates using Newton–Raphson method. This hybrid strategy shows good performance in terms of accuracy and efficiency. We set the following convergence criterion for HiDeNN-PGD strategy,

$$\left( \sum_{I=2}^{n_1-1} \left( \frac{\partial \Pi^{(k)}}{\partial x_I} \right)^2 + \sum_{J=2}^{n_2-1} \left( \frac{\partial \Pi^{(k)}}{\partial y_J} \right)^2 + \sum_{K=2}^{n_3-1} \left( \frac{\partial \Pi^{(k)}}{\partial z_K} \right)^2 \right)^{\frac{1}{2}} < \epsilon, \tag{59}$$

where the superscript $k$ is the iteration step, and $\epsilon$ is the tolerance.

The computational cost of HiDeNN-TD/HiDeNN-PGD is influenced by several factors including the optimization scheme, initial guess and required accuracy. Note that we present two results with different tolerances for HiDeNN-PGD in the table. As shown in Table 7, when taking a large tolerance ($\epsilon = 10^{-1}$), HiDeNN-PGD strategy helps to achieve a good accuracy with a comparable efficiency to PGD/TD, faster and a lot more accurate than FEM. Note that Table 7 reports also the results with the converged modes for PGD, TD and HiDeNN-PGD. As we can see, PGD takes more modes than TD and HiDeNN-PGD. When taking a tighter tolerance ($\epsilon = 10^{-2}$) for HiDeNN-PGD, we further improve the accuracy with additional computational cost. All the computations are under the framework of serial algorithms.

Based on the HiDeNN-PGD solution, we can add an additional loop to optimize the solution using TD while fixing the mode number, i.e., HiDeNN-TD in Table 8. From a theoretical point of view, as expected, the HiDeNN-TD further improves the solution accuracy as shown in Table 8. It turns out that the current solution accuracy is improved by HiDeNN-TD in the first or second significant digit with 5 modes. When considering fewer modes (2 modes), HiDeNN-TD shows a much better accuracy compared to HiDeNN-PGD, due to the fact that TD requires fewer

**Table 8**
Computational cost comparison for HiDeNN-PGD and HiDeNN-TD with different meshes and modes for 3D problem. Here, HiDeNN-PGD adopts a PGD and Newton–Raphson method hybrid strategy, and HiDeNN-TD refines the HiDeNN-PGD results by a TD step.

| Mesh | HiDeNN-PGD (Hybrid) | | | | HiDeNN-TD (Hybrid) | | | |
|---|---|---|---|---|---|---|---|---|
| | $\epsilon = 10^{-1}$ | | $\epsilon = 10^{-1}$ | | $\epsilon = 10^{-1}$ | | $\epsilon = 10^{-1}$ | |
| | Time (s) | Error | Time (s) | Error | Time (s) | Error | Time (s) | Error |
| $40 \times 40 \times 40$ | **0.58879**(2) | **17.139%** | **1.5210**(5) | **12.263%** | **0.60405**(2) | **12.784%** | **1.7562**(5) | **12.256%** |
| $80 \times 80 \times 80$ | 0.35486(2) | 16.709% | 1.0106(5) | 10.997% | 0.37804(2) | 12.189% | 1.3804(5) | 10.988% |
| $160 \times 160 \times 160$ | 0.68524(2) | 13.772% | 1.8125(5) | 6.512% | 0.73674(2) | 7.523% | 3.3137(5) | 6.468% |
| $320 \times 320 \times 320$ | 0.66899(2) | 4.046% | 6.1171(5) | 3.492% | 0.80433(2) | 3.532% | 8.8504(5) | 3.476% |

**Table 9**
Accuracy and computational cost for HiDeNN-TD with PSO method for 3D problem.

| Mesh | HiDeNN-TD (PSO) | |
|---|---|---|
| | Time (s) | Error |
| $40 \times 40 \times 40$ | **4,242.1**(5) | **6.026%** |
| $80 \times 80 \times 80$ | 4,562.8(5) | 3.404% |
| $160 \times 160 \times 160$ | 5,548.0(5) | 2.100% |
| $320 \times 320 \times 320$ | 12,145(5) | 1.102% |

**Table 10**
Accuracy comparison for different methods with different meshes for 3D problem.

| Mesh | HiDeNN-TD (Adam) Error | HiDeNN (Regular Mesh, Adam) Error | HiDeNN-TD (PSO) Error |
|---|---|---|---|
| $40 \times 40 \times 40$ | **10.796%** | **10.780%** | **6.026%** |
| $80 \times 80 \times 80$ | 6.770% | 6.770% | 3.404% |
| $160 \times 160 \times 160$ | 4.036% | 3.984% | 2.100% |
| $320 \times 320 \times 320$ | 1.831% | 1.830% | 1.102% |

modes than PGD for a given accuracy. This is consistent with our theoretical analysis. In addition, since this is a high-dimensional optimization problem, looking for a globally optimal solution requires more computational effort. Remark that the above gradient based algorithms cannot guarantee the global minimum when facing nonconvex optimization problems. Seeking a good solution scheme to further improve the efficiency of HiDeNN-TD will be our future work. Some other global optimization methods such as particle swarm optimization (PSO) [41] can be considered. Our experience showed that the PSO can help us to achieve accurate results, since PSO is designed to search for the global minima from many different starting points. As shown in Table 10, the PSO can consistently achieve a higher accuracy for all the meshes compared to Adam method. For example, we can achieve about 6% error for the $40 \times 40 \times 40$ mesh, whereas the solution obtained with Adam is about 11% in error. Yet this method might not be as efficient as the HiDeNN-PGD strategy, as shown in Tables 7 and 9.

The HiDeNN-TD needs *a priori* assumption on the number of modes. The alternative hybrid solution scheme can help estimate the number of modes using PGD and improve the computational speed while providing solutions more accurate than those by PGD/TD and FEM (see Tables 7 and 8). This confirms our theoretical analysis. Another way is to estimate the necessary number of modes (numerically or mathematically). For this reason, we explored the error of TD method. We decompose the error into mode reduction error and discretization error theoretically, and obtain (39) (the proof is given in Subsection 3.2). Based on this, we have tested by establishing a function by curve fitting, relating the error to the number of modes and mesh size.

Concerning potential applications, the proposed HiDeNN-TD straightforwardly applies to problems with regular shapes and meshes. For problems with irregular shapes or meshes, a mesh transformation is required. This point

may limit the application of the method to arbitrary domains, although some techniques have been proposed to overcome this issue. We proposed one mapping strategy in Appendix F.

## 6. Conclusion

A reduced-order hierarchical deep learning network has been proposed. The so-called HiDeNN-TD is a combination of HiDeNN and TD with separated spatial variables. This combined method presents several advantages over HiDeNN and TD methods. First, it allows leveraging the automatic mesh adaptivity of the HiDeNN method for reducing the number of modes in TD approximation. Second, combining TD with HiDeNN reduces significantly the number of degrees of freedom for HiDeNN and potentially leads to a much higher computational efficiency. Furthermore, we have demonstrated that both HiDeNN and HiDeNN-TD can provide more accurate solutions than FEM and PGD, through an error analysis with the help of analyzing the approximation function sets.

The numerical results have confirmed the mathematical analysis. These examples have been performed based on 2D and 3D Poisson problems. It is shown that the proposed HiDeNN-TD method can provide accurate solutions with the least degrees of freedom. In order to have an idea for the prescribed number of modes in HiDeNN-TD, we have studied numerically the convergence rate on PGD approximation. It has been found that the convergence rate on the mode number is insensitive to the mesh size. Therefore, we can expect to use PGD and a very coarse mesh to compute the necessary number of modes for HiDeNN-TD. This finding is interesting and provides a useful guideline on the choice of the number of modes for HiDeNN-TD or other PGD-based methods that may require a better optimality in terms of basis.

Theoretical results of convergence studies need to be derived through a rigorous mathematical analysis. The numerical results provided in this paper can serve as the first evidence for demonstrating the capabilities of the method.

In addition, we proposed an efficient variant of the method, called HiDeNN-PGD. The method leverages the PGD type solution scheme for computing the modes and the HiDeNN method for optimizing the mesh coordinates. Consequently, this method makes the solution time comparable to PGD method and results more accurate than PGD/TD and FEM. The method has made a trade-off between the solution accuracy and efficiency, compared to HiDeNN-TD. Further improvement of the method can be employing some global optimization algorithms, such as PSO, as discussed previously. The key point of the future development is to find a good solution scheme that is able to achieve the global optimum while maintaining a high efficiency.

The proposed HiDeNN-TD and its variant HiDeNN-PGD have shown good potentials to achieve the high accuracy and efficiency for solving PDEs. They can provide a powerful tool for problems that require very large mesh systems such as the additive manufacturing simulations and topology optimization problems.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

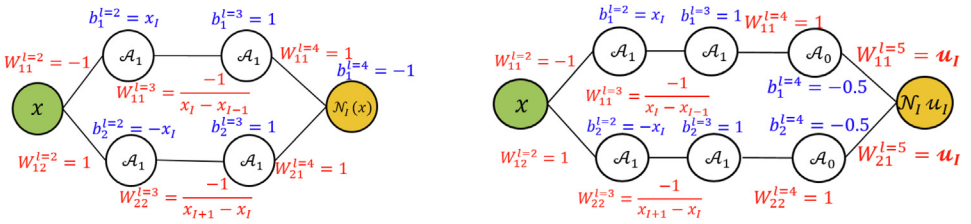
## Acknowledgments

L. Zhang and S. Tang are supported by National Natural Science Foundation of China Grant Number 11890681, 11832001, 11521202 and 11988102. W.K. Liu and Y. Lu are supported by National Science Foundation, USA Grant Numbers CMMI-1934367 and CMMI-1762035.

## Appendix A. 1D HiDeNN formulation

In standard 1D FEM, the computational domain $\Omega$ is discretized by a grid with $np$ nodes and the shape function associated with an internal node $x_I$ is

$$N_I(x) = \begin{cases} \dfrac{x - x_{I-1}}{x_I - x_{I-1}}, & x_{I-1} \le x \le x_I, \\ \dfrac{x_{I+1} - x}{x_{I+1} - x_I}, & x_I \le x \le x_{I+1}, \\ 0, & elsewhere, \end{cases} \tag{A.1}$$




(a) DNN-based 1D shape function (b) DNN-based 1D interpolation function

**Fig. A.14.** Deep neural network (DNN) representation of the 1D global shape function and interpolation function.

where $x_{I-1}$ and $x_{I+1}$ are the two neighbor points of the node $x_I$ from the left side and right side, respectively.

We rewrite $N_I(x)$ in a DNN format consisting of weights, biases, and activation functions. Considering the shape function in a piecewise linear form, we choose the ReLU function, $\mathcal{A}_1 = \max(0, x)$, as the activation function. Fig. A.14(a) shows the DNN representation of the linear shape function. The corresponding formula is

$$\begin{aligned}\mathcal{N}_I(x;\boldsymbol{W},\boldsymbol{b},\boldsymbol{\mathcal{A}}) &= W_{11}^{l=4}\mathcal{A}_1\left(W_{11}^{l=3}\mathcal{A}_1\left(W_{11}^{l=2}x + b_1^{l=2}\right) + b_1^{l=3}\right) \\ &\quad + W_{21}^{l=4}\mathcal{A}_1\left(W_{22}^{l=3}\mathcal{A}_1\left(W_{12}^{l=2}x + b_2^{l=2}\right) + b_2^{l=3}\right) + b_1^{l=4} \\ &= \mathcal{A}_1\left(\frac{-1}{x_I - x_{I-1}}\mathcal{A}_1\left(-x + x_I\right) + 1\right) + \mathcal{A}_1\left(\frac{-1}{x_{I+1} - x_I}\mathcal{A}_1\left(x - x_I\right) + 1\right) - 1,\end{aligned} \tag{A.2}$$

where $\boldsymbol{W} = [W_{11}^{l=2}, W_{12}^{l=2}, W_{11}^{l=3}, W_{22}^{l=3}, W_{11}^{l=4}, W_{21}^{l=4}]$, and $\boldsymbol{b} = [b_1^{l=2}, b_2^{l=2}, b_1^{l=3}, b_2^{l=3}, b_1^{l=4}]$ are the weights and biases of the connected neurons. Detailed definitions of the notations are provided in Table 1. Note that all the weights and biases are functions of nodal coordinates. The formula can be rewritten as the form of

$$\mathcal{N}_I(\boldsymbol{x};\boldsymbol{x}_I^*,\boldsymbol{\mathcal{A}}), \tag{A.3}$$

where $\boldsymbol{x}_I^*$ denotes the vector that represents the neighbor nodes of node $\boldsymbol{x}_I$ involved in $N_I(\boldsymbol{x})$. For 1D linear shape function, it should be $\boldsymbol{x}_I^* = \{x_{I-1}, x_I, x_{I+1}\}$. For the sake of clarity, one more layer is added to introduce the nodal value $u_I$, i.e., the formula becomes

$$\begin{aligned}u_I^h &= \mathcal{N}_I(x;\ \boldsymbol{W},\boldsymbol{b},\ \boldsymbol{\mathcal{A}})u_I = \mathcal{N}_I(x;\ \boldsymbol{x}_I^*,\ \boldsymbol{\mathcal{A}})u_I;\ \text{no summation on } I \\ &= \mathcal{A}_0\left(\mathcal{A}_1\left(\frac{-1}{x_I - x_{I-1}}\mathcal{A}_1\left(-x + x_I\right) + 1\right) - 0.5\right)u_I \\ &\quad + \mathcal{A}_0\left(\mathcal{A}_1\left(\frac{-1}{x_{I+1} - x_I}\mathcal{A}_1\left(x - x_I\right) + 1\right) - 0.5\right)u_I,\end{aligned} \tag{A.4}$$

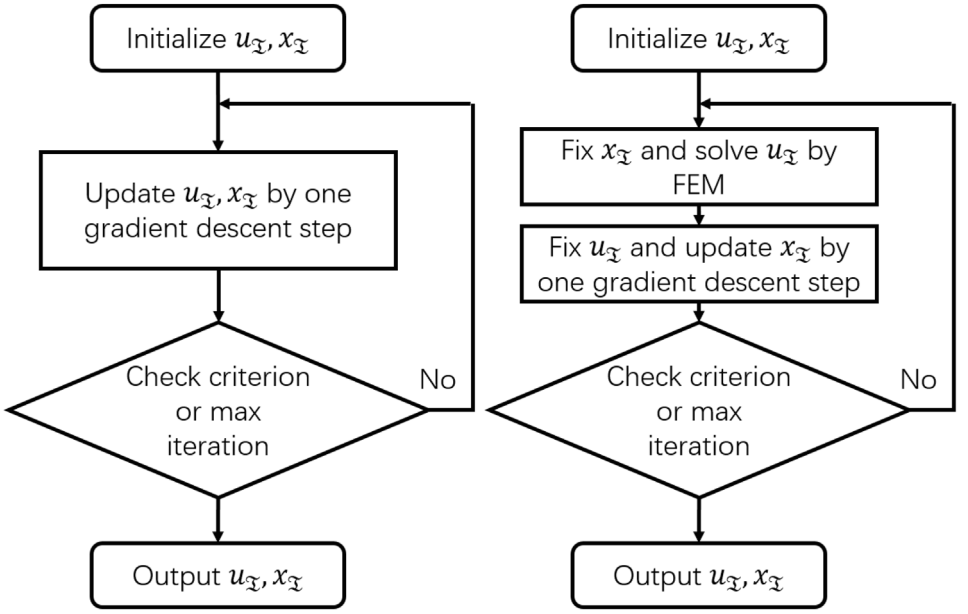
where $u_I^h$ and $u_I$ are the interpolated displacement and nodal displacement at node $x_I$, $\boldsymbol{\mathcal{A}} = [\mathcal{A}_0, \mathcal{A}_1]$ are the activation functions used for the construction of the DNN approximation. $\mathcal{A}_0(x) = x$ is an identical function. Fig. A.14(b) gives the DNN representation of the interpolation of the nodal displacement at node $x_I$.

Once the shape function with nodal value for an arbitrary node $x_I$ is constructed, the interpolation is obtained by assembling all DNNs, i.e.,

$$u^h(x) = \sum_{I=1}^{np} \mathcal{N}_I(x;\ \boldsymbol{x}_I^*,\ \boldsymbol{\mathcal{A}})u_I. \tag{A.5}$$

Compared with classical FEM, nodal positions are introduced as additional DoFs in the optimization for HiDeNN, which increases both the local and global accuracy of the interpolants.

Ref. [12] also presented the DNN representation of various rational functions including Lagrange polynomials, B-spline, Reproducing Kernel Particle Method (RKPM), NURBS, Isogeometric analysis (IGA), etc., and multidimensional shape functions.




(a) Scheme 1: optimize $u_{\mathcal{I}}, x_{\mathcal{I}}$ simultaneously. (b) Scheme 2: optimize $u_{\mathcal{I}}, x_{\mathcal{I}}$ alternately.

**Fig. B.15.** Two different solution schemes for HiDeNN.

## Appendix B. Solution schemes for HiDeNN

We present two different solution schemes for HiDeNN as illustrated in Fig. B.15. The first way is the classical optimization process, i.e., to optimize nodal values and nodal positions by gradient descent algorithm such as Adam algorithm simultaneously. The second way is to split the updating process into two parts: a FEM solver to update nodal values with fixed nodal positions; and then a gradient descent step to update nodal positions with fixed nodal values. Although this may be more expensive than the previous scheme, the latter one can be programmed as a post-processor for an existing finite element algorithm.

## Appendix C. Convergence for tensor decomposition method at increasing number of modes

In this section, we discuss the convergence of the canonical decomposition method at increasing $Q$ (number of modes). In Section 2, we have shown that $\mathcal{M}_Q^h \subset \mathcal{V}^h$, provided that their interpolations are based on the same basis functions. Here, we make some further discussions.

For 2D case, we compare

$$\mathcal{M}_Q^h = \left\{ u^h \middle| u^h = \sum_{q=1}^{Q} \left( \sum_{I=1}^{n_1} N_I(x) \beta_I^{(q)} \right) \left( \sum_{J=1}^{n_2} N_J(y) \gamma_J^{(q)} \right), \beta_I^{(q)}, \gamma_J^{(q)} \in \mathbb{R}, I = 1, \ldots, n_1, J = 1, \ldots, n_2 \right\} \tag{C.1}$$

with

$$\mathcal{V}^h = \left\{ u^h \middle| u^h = \sum_{I=1}^{n_1} \sum_{J=1}^{n_2} N_I(x) N_J(y) u_{(I,J)}, u_{(I,J)} \in \mathbb{R}, I = 1, \ldots, n_1, J = 1, \ldots, n_2 \right\}. \tag{C.2}$$

As the same basis functions are used, it follows that:

**1.** $\forall Q \in \mathbb{N}, \mathcal{M}_Q^h \subset \mathcal{V}^h$;

**2.** If $Q_1 \leq Q_2$, $Q_1, Q_2 \in \mathbb{N}$, $\mathcal{M}_{Q_1}^h \subset \mathcal{M}_{Q_2}^h$;

**3.** $\forall Q \geq \min\{n_1, n_2\}$, $Q \in \mathbb{N}$, $\mathcal{M}_Q^h = \mathcal{V}^h$.

The first two statements are straightforward. The last property is proved in Appendix D.

We then conclude the following relationship:

$$\mathcal{M}^h_{Q=1} \subset \mathcal{M}^h_{Q=2} \subset \cdots \subset \mathcal{M}^h_{Q=\min\{n_1,n_2\}} = \mathcal{M}^h_{Q=\min\{n_1,n_2\}+1} = \cdots = \mathcal{V}^h. \tag{C.3}$$

In other words, $\mathcal{M}^h_Q$ is always a subset of $\mathcal{V}^h$, and it approaches to $\mathcal{V}^h$ when $Q$ increases. Consequently, when enough number of modes are taken, the canonical decomposition result reaches the same accuracy as the FEM solution. We remark that $\min\{n_1, n_2\}$ is precisely the minimal number of modes to ensure $\mathcal{M}^h_Q = \mathcal{V}^h$.

The above discussions extend to 3D readily.

**1.** $\forall Q \in \mathbb{N}, \mathcal{M}^h_Q \subset \mathcal{V}^h$;

**2.** If $Q_1 \leq Q_2$, $Q_1, Q_2 \in \mathbb{N}, \mathcal{M}^h_{Q_1} \subset \mathcal{M}^h_{Q_2}$;

**3.** $\forall Q \geq \min\{n_1n_2, n_2n_3, n_1n_3\}$, $Q \in \mathbb{N}, \mathcal{M}^h_Q = \mathcal{V}^h$.

We note that the minimal number of modes to ensure $\mathcal{M}^h_Q = \mathcal{V}^h$ in 3D is essentially to find a best rank-$r$ approximation to order-3 tensor, which is an open mathematical problem. An upper bound is given in property 3.

## Appendix D. Convergence for 2D tensor decomposition method at increasing number of modes

**Theorem 2.** *For 2D case, $\mathcal{M}^h_Q$ and $\mathcal{V}^h$ are defined in (C.1) and (C.2), respectively. We have*

$$\forall Q \geq \min(n_1, n_2),\ Q \in \mathbb{N},\ \mathcal{M}^h_Q = \mathcal{V}^h. \tag{D.1}$$

**Proof.** For any interpolation function

$$u^{h,FEM} = \sum_{I=1}^{n_1}\sum_{J=1}^{n_2} N_I(x)N_J(y)u_{(I,J)} \tag{D.2}$$

in the set $\mathcal{V}^h$, we write the nodal values in the form of matrix,

$$\boldsymbol{U} = \begin{bmatrix} u_{(1,1)} & u_{(1,2)} & \cdots & u_{(1,n_2)} \\ u_{(2,1)} & u_{(2,2)} & \cdots & u_{(2,n_2)} \\ \vdots & \vdots & & \vdots \\ u_{(n_1,1)} & u_{(n_1,2)} & \cdots & u_{(n_1,n_2)} \end{bmatrix}. \tag{D.3}$$

According to the singular value decomposition (SVD), $\boldsymbol{U}$ is represented by

$$\boldsymbol{U} = \sum_{q=1}^{rank(\boldsymbol{U})} \sigma^{(q)}\boldsymbol{w}^{(q)} \otimes \boldsymbol{v}^{(q)}, \sigma^{(1)} \geq \sigma^{(2)} \geq \cdots \geq \sigma^{(rank(\boldsymbol{U}))} > 0, \tag{D.4}$$

where $\boldsymbol{w}^{(q)}$ is the $n_1$-dimensional vector, and $\boldsymbol{v}^{(q)}$ is the $n_2$-dimensional vector. Thus $u^{h,FEM}$ is rewritten in the form of the separation of variables, i.e.,

$$u^{h,FEM} = \sum_{I=1}^{n_1}\sum_{J=1}^{n_2} N_I(x)N_J(y)\left(\sum_{q=1}^{rank(\boldsymbol{U})} \sigma^{(q)}w_I^{(q)}v_J^{(q)}\right). \tag{D.5}$$

So we have $u^{h,FEM} \in \mathcal{M}^h_Q$, if $Q \geq \min\{n_1, n_2\} \geq rank(\boldsymbol{U})$. Combining with $\mathcal{M}^h_Q \subset \mathcal{V}^h$, we obtain (D.1). □

We remark that SVD tells us the minimal number of modes to reproduce the FE solution, i.e. $\min\{n_1, n_2\}$.

## Appendix E. Convergence for HiDeNN-TD at increasing number of modes

FEM and TD can be regarded as specific cases in the regulated HiDeNN and HiDeNN-TD with nodal coordinates fixed, respectively, i.e.,

$$\mathcal{HR}^h = \bigcup_{x_I^*, y_J^*, z_K^*} \mathcal{V}^h(x_I^*, y_J^*, z_K^*), \quad \mathcal{G}^h_Q = \bigcup_{x_I^*, y_J^*, z_K^*} \mathcal{M}^h_Q(x_I^*, y_J^*, z_K^*). \tag{E.1}$$

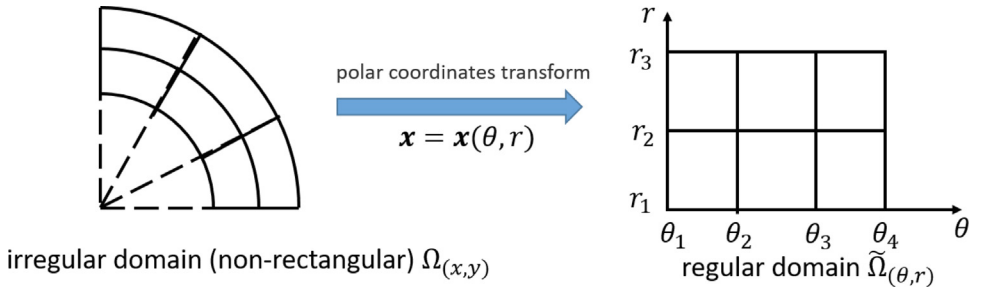



**Fig. F.16.** A quarter of ring is transformed in to a rectangular by the polar coordinates transformation.

In Appendices C and D, we show that $\mathcal{M}_Q^h$ approaches to $\mathcal{V}^h$ when $Q$ is big enough. When $Q \geq \min\{n_1, n_2\}$ for 2D and $Q \geq \min\{n_1 n_2, n_2 n_3, n_3 n_1\}$ for 3D, we have $\mathcal{M}_Q^h = \mathcal{V}^h$, so $\mathcal{G}_Q^h = \mathcal{H}\mathcal{R}^h$.

## Appendix F. Space separated PGD

When the domain is not intrinsically separable, fully separated representation cannot be applied directly. [37] immersed the non-separable domain onto a fully separable one. [42] used geometrical mapping to deal with layered domain, where interfaces are not planar. Now we combine representation of separated variables with FE mapping to deal with more complex geometrical cases.

### *F.1. Mesh mapping and recovering for irregular domains*

The main idea is to map original irregular domain $\Omega_{(\boldsymbol{x})}$ to a regular one $\tilde{\Omega}_{(\tilde{\boldsymbol{x}})}$, and then apply the separated representation. Fig. F.16 illustrates a simple example. A quarter of ring becomes a rectangular through polar transformation. Then the new representation of separated variables is

$$u^h = \sum_{q=1}^{Q} \tilde{X}^{(q)}(\theta)\tilde{Y}^{(q)}(r), \tag{F.1}$$

where $\theta$ and $r$ are the functions of space coordinates $x, y$. By virtue of parametric transformation in FEM, we propose a general way to define this mapping as illustrated in Fig. F.17. We present a FE mesh over the 2D irregular computational domain $\Omega_{\boldsymbol{x}}$ first with nodes $\boldsymbol{x}_{i,j}, i = 1, 2, \ldots, n_1, j = 1, 2, \ldots, n_2$. Then we define a mapping to its corresponding lattice $(i, j)$. $\tilde{\boldsymbol{x}}$ is the coordinates of the transformed domain $\tilde{\Omega}_{\tilde{\boldsymbol{x}}}$. The mapping consists of two steps:

1. Mapping each element to a square/cubic

The first mapping is the classical parametric mapping in FEM. We make a change of coordinates which maps the 4-node element into a square $[-1, 1]^2$ for 2D or maps the 8-node element into a cubic $[-1, 1]^3$ for 3D. The coordinates of a point $\boldsymbol{\xi}$ in the square are related to the physical coordinates of a point $\boldsymbol{x}$ in the element by mappings of the form

$$\boldsymbol{x} = \sum_{a=1}^{n_e} N_a^e(\boldsymbol{\xi})\boldsymbol{x}_a^e \tag{F.2}$$

where $n_e$ is the number of nodes of the element ($n_e = 4$ for 2D and $n_e = 8$ for 3D), $\boldsymbol{x}_a^e$ is the coordinates of the $a$th node of the element, and $N_a^e(\boldsymbol{\xi})$ is the corresponding shape function. $\boldsymbol{\xi}$ is called as natural coordinates.

2. Mapping the square/cubic to a lattice

For the sake of separated representation, we define the second mapping to assemble the square/cubic into a lattice. The transformed formula is

$$\tilde{\boldsymbol{x}} = \sum_{a=1}^{n_e} N_a^e(\boldsymbol{\xi})\tilde{\boldsymbol{x}}_a^e \tag{F.3}$$

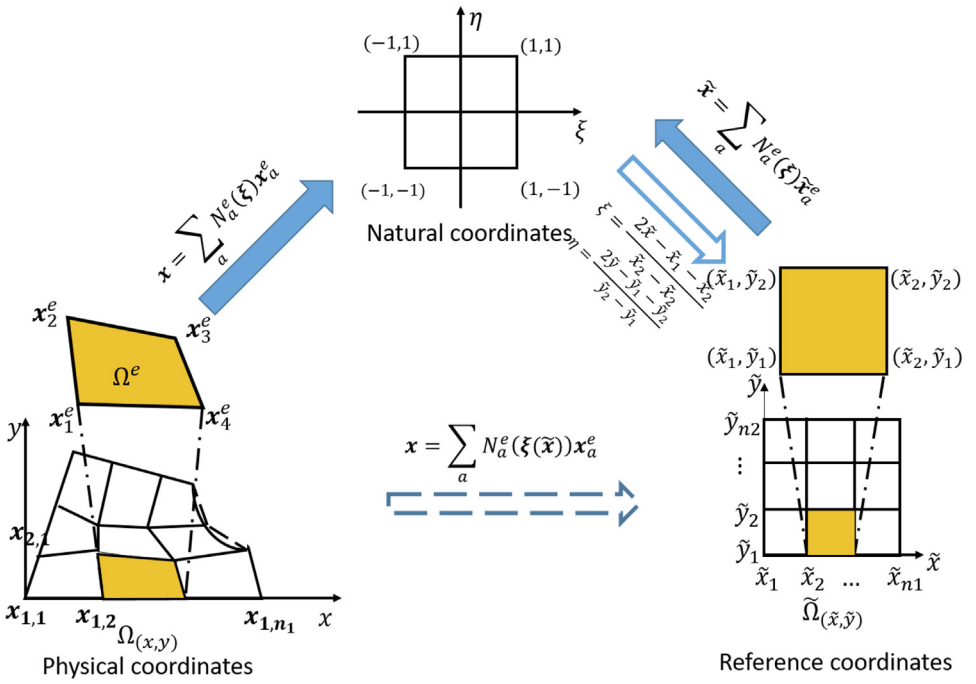



**Fig. F.17.** Illustration for the geometrical mapping. The irregular domain with irregular mesh is related to a regular domain with regular mesh by a 2-step mapping.

and the inverse transformation is

$$\xi = \frac{2\tilde{x} - \tilde{x}_1^e - \tilde{x}_2^e}{\tilde{x}_2^e - \tilde{x}_1^e} \tag{F.4}$$

$$\eta = \frac{2\tilde{y} - \tilde{y}_1^e - \tilde{y}_2^e}{\tilde{y}_2^e - \tilde{y}_1^e} \tag{F.5}$$

$$\zeta = \frac{2\tilde{z} - \tilde{z}_1^e - \tilde{z}_2^e}{\tilde{z}_2^e - \tilde{z}_1^e}. \tag{F.6}$$

The final transformed domain $\tilde{\Omega}_{\tilde{x}}$ is called as reference domain, which is a regular domain with a regular mesh $[\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{n_1}] \times [\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{n_2}] \times [\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{n_3}]$. $\tilde{x}_1^e, \tilde{x}_2^e, \tilde{y}_1^e, \tilde{y}_2^e, \tilde{z}_1^e, \tilde{z}_2^e$ are the coordinates of the element $[\tilde{x}_1^e, \tilde{x}_2^e] \times [\tilde{y}_1^e, \tilde{y}_2^e] \times [\tilde{z}_1^e, \tilde{z}_2^e]$ in the reference domain. For convenience, we might take the mesh in the reference domain as a lattice corresponding to the index of the nodes in the physical domain, i.e., $\tilde{x}_i = i, i = 1, 2, \ldots, n_1$, $\tilde{y}_j = j, j = 1, 2, \ldots, n_2$, $\tilde{z}_k = k, k = 1, 2, \ldots, n_3$.

The whole mapping is defined as below,

$$\boldsymbol{x} = \sum_{a=1}^{n_e} N_a^e(\boldsymbol{\xi}(\tilde{\boldsymbol{x}}))\boldsymbol{x}_a^e. \tag{F.7}$$

Then separation of spatial variables is applicable to the reference domain, i.e., the interpolation function set is

$$\tilde{\mathcal{M}}_Q^h = \left\{ u^h \middle| u^h = \sum_{q=1}^{Q} \tilde{X}(\tilde{x})\tilde{Y}(\tilde{y})\tilde{Z}(\tilde{z}) = \sum_{q=1}^{Q} \left( \sum_{i=1}^{n_1} N_i(\tilde{x})\beta_i^{(q)} \right) \left( \sum_{j=1}^{n_2} N_j(\tilde{y})\gamma_j^{(q)} \right) \left( \sum_{k=1}^{n_3} N_k(\tilde{z})\theta_k^{(q)} \right) \right\}, \tag{F.8}$$

where $N_i(\tilde{x})$, $N_j(\tilde{y})$, $N_i(\tilde{z})$ are shape functions, and $\beta_i^{(q)}$, $\gamma_j^{(q)}$, $\theta_k^{(q)}$ are the corresponding coefficients of the $q$th mode. Note that $\tilde{x}$, $\tilde{y}$, $\tilde{z}$ are the functions of physical coordinates $x$, $y$, $z$.

### F.2. Solution schemes

There are different solution schemes to solve the problem. A straight way is to find the solution by minimizing the variational formula with the given number of modes $Q^*$ directly, i.e.,

$$u^h = \underset{u^h \in \tilde{\mathcal{M}}^h_{Q=Q^*}}{\arg\min} \ \Pi[u^h(\boldsymbol{x}; \boldsymbol{\beta}^{(q)}, \boldsymbol{\gamma}^{(q)}, \boldsymbol{\theta}^{(q)})]. \tag{F.9}$$

All the parameters are solved at the same time.

Yet this global optimization might be expensive, so we might borrow the idea from PGD [15,39], i.e., incremental solution scheme. More precisely, the solution scheme is

$$\begin{aligned} &\text{The first mode:} && u^{PGD,(1)} = \underset{\tilde{u} \in \tilde{\mathcal{M}}^h_{Q=1}}{\arg\min}\, \Pi[u^0 + \tilde{u}]; \\ &\text{For } m > 1, \text{ the } m\text{th mode:} && \tilde{u}^{(m)} = \underset{\tilde{u} \in \tilde{\mathcal{M}}^h_{Q=1}}{\arg\min}\, \Pi[u^{PGD,(m-1)} + \tilde{u}], \\ &\text{The PGD solution with } m \text{ modes:} && u^{PGD,(m)} = u^{PGD,(m-1)} + \tilde{u}^{(m)}, \end{aligned} \tag{F.10}$$

We remarked that it is also possible to solve several modes simultaneously in one incremental step.

In general, the initial guess $u_0$ is set to be zero. When dealing with the boundary conditions, $u^0$ can be arbitrary continuous functions satisfying boundary conditions. We can also set an appropriate initial guess to improve the efficiency.

These two solution schemes have their advantages and disadvantages. With the same number of modes $Q^*$, we solve the modes one by one using the latter solution scheme (PGD) (F.10) while optimize all the modes together using the previous one (TD) (F.9). Thus, we have $\Pi[u^{TD}] < \Pi[u^{PGD}]$ and then obtain

$$\|u^{TD} - u^{exact}\|_E \leq \|u^{PGD} - u^{exact}\|_E. \tag{F.11}$$

This indicates the solution of the previous solution scheme (F.9) might be better than that of the incremental way. However, this TD solution scheme (F.9) might cost more.

### F.3. Illustrating the solution procedure: the 2D Poisson problem

For the sake of simplicity and without loss of generality, we consider 2D Poisson problem with incremental solution scheme for illustration,

$$\begin{cases} \Delta u(x, y) + b(x, y) = 0 \text{ in } \Omega_{(x,y)} \subset \mathbb{R}^2, \\ u|_{\partial\Omega} = 0. \end{cases} \tag{F.12}$$

(F.12) is solved in the irregular domain $\Omega_{(x,y)}$ with homogeneous boundary conditions.

After mapping, the solution is represented in the form of (F.8). Then we solve it with incremental solution scheme. Let previous $q-1$ modes solved. The $q$th mode is obtained by

$$\tilde{u}^{(q)} = \underset{\tilde{u} \in \tilde{\mathcal{M}}^h_{Q=1}}{\arg\min}\, \Pi[u^{PGD,(q-1)} + \tilde{u}], \tag{F.13}$$

where $u^{PGD,(q-1)}$ is the sum of the previous $q-1$ modes. We rewrite the interpolation function in the following matrix form

$$u^{PGD,(q)} = u^{PGD,(q-1)} + \tilde{u}^{(q)}, \tilde{u}^{(q)} = \left((\boldsymbol{\beta}^{(q)})^T \boldsymbol{N}^{\beta}(\tilde{x})\right)\left((\boldsymbol{\gamma}^{(q)})^T \boldsymbol{N}^{\gamma}(\tilde{y})\right), \tag{F.14}$$

where $\boldsymbol{\beta}^{(q)}, \boldsymbol{\gamma}^{(q)}$ are the coefficient vectors, and $\boldsymbol{N}^{\beta}(\tilde{x}), \boldsymbol{N}^{\gamma}(\tilde{y})$ denotes the vector containing shape functions.

Substituting (F.14) into the variational formula (46), we have

$$\begin{aligned} \Pi[u^{PGD,(q)}] = &\frac{1}{2}\int_{\Omega_{(x,y)}} \left(\nabla(\tilde{u}^{(q)})\right)^2 \mathrm{d}x\mathrm{d}y + \int_{\Omega_{(x,y)}} \left(\nabla(\tilde{u}^{(q)})\right) \cdot \left(\nabla(u^{PGD,(q-1)})\right) \mathrm{d}x\mathrm{d}y \\ &- \int_{\Omega_{(x,y)}} \tilde{u}^{(q)}(x, y) b(x, y) \mathrm{d}x\mathrm{d}y + \Pi[u^{PGD,(q-1)}]. \end{aligned} \tag{F.15}$$

The quadratic term with respect to $\boldsymbol{\beta}^{(q)}$, $\boldsymbol{\gamma}^{(q)}$ in the variational formula is given by

$$\int_{\Omega_{(x,y)}} \left(\nabla(\tilde{u}^{(q)})\right)^2 \mathrm{d}x\mathrm{d}y = \int_{\tilde{\Omega}_{(\tilde{x},\tilde{y})}} \left(\nabla_{(\tilde{x},\tilde{y})}(\tilde{u}^{(q)})\right)^T \boldsymbol{J}^{-T}\boldsymbol{J}^{-1}\nabla_{(\tilde{x},\tilde{y})}(\tilde{u}^{(q)})\det(\boldsymbol{J})\mathrm{d}\tilde{x}\mathrm{d}\tilde{y}. \tag{F.16}$$

with the Jacobi matrix

$$\boldsymbol{J} = \frac{\partial(x,\, y)}{\partial(\tilde{x},\, \tilde{y})}. \tag{F.17}$$

The gradient of $\tilde{u}^{(q)}$ with respect to $(\tilde{x},\, \tilde{y})$ is

$$\nabla_{(\tilde{x},\tilde{y})}\tilde{u}^{(q)} = \begin{bmatrix} \dfrac{\partial}{\partial\tilde{x}} \\ \dfrac{\partial}{\partial\tilde{y}} \end{bmatrix} \tilde{u}^{(q)} = \begin{bmatrix} (\boldsymbol{\beta}^{(q)})^T \dfrac{d\boldsymbol{N}^{\beta}(\tilde{x})}{d\tilde{x}} (\boldsymbol{N}^{\gamma}(\tilde{y}))^T\gamma^{(q)} \\ (\boldsymbol{\beta}^{(q)})^T \boldsymbol{N}^{\beta}(\tilde{x})(\dfrac{d\boldsymbol{N}^{\gamma}(\tilde{y})}{d\tilde{y}})^T\gamma^{(q)} \end{bmatrix}. \tag{F.18}$$

Note that the expression is a nonlinear algebraic system. It is hard to solve it directly, so we use the alternating direction strategy as below:

In each iteration step,

1. Fix $\gamma$ and solve $\beta$

The quadratic term becomes

$$\int_{\Omega_{(x,y)}} \left(\nabla(\tilde{u}^{(q)})\right)^2 \mathrm{d}x\mathrm{d}y = \int_{\tilde{\Omega}_{(\tilde{x},\tilde{y})}} \beta^{(q),T}\boldsymbol{B}^{\beta,T}(\tilde{x},\, \tilde{y})\boldsymbol{B}^{\beta}(\tilde{x},\, \tilde{y})\beta^{(q)}\det(\boldsymbol{J})\mathrm{d}\tilde{x}\mathrm{d}\tilde{y} \tag{F.19}$$

with

$$\boldsymbol{B}^{\beta}(\tilde{x},\, \tilde{y}) = \boldsymbol{J}^{-1}\begin{bmatrix} (\boldsymbol{\gamma}^{(q)})^T\boldsymbol{N}^{\gamma}(\tilde{y})(\dfrac{d\boldsymbol{N}^{\beta}(\tilde{x})}{d\tilde{x}})^T \\ (\boldsymbol{\gamma}^{(q)})^T\dfrac{d\boldsymbol{N}^{\gamma}(\tilde{y})}{d\tilde{y}}(\boldsymbol{N}^{\beta}(\tilde{x}))^T \end{bmatrix}. \tag{F.20}$$

The stiffness matrix for $\beta^{(q)}$ is

$$\boldsymbol{K}^{\beta} = \int_{\tilde{\Omega}_{(\tilde{x},\tilde{y})}} \boldsymbol{B}^{\beta,T}(\tilde{x},\, \tilde{y})\boldsymbol{B}^{\beta}(\tilde{x},\, \tilde{y})\det(\boldsymbol{J})\mathrm{d}\tilde{x}\mathrm{d}\tilde{y}. \tag{F.21}$$

2. Fix $\beta$ and solve $\gamma$

The quadratic term becomes

$$\int_{\Omega_{(x,y)}} \left(\nabla(\tilde{u}^{(q)})\right)^2 \mathrm{d}x\mathrm{d}y = \int_{\tilde{\Omega}_{(\tilde{x},\tilde{y})}} \gamma^{(q),T}\boldsymbol{B}^{\gamma,T}(\tilde{x},\, \tilde{y})\boldsymbol{B}^{\gamma}(\tilde{x},\, \tilde{y})\gamma^{(q)}\det(\boldsymbol{J})\mathrm{d}\tilde{x}\mathrm{d}\tilde{y} \tag{F.22}$$

with

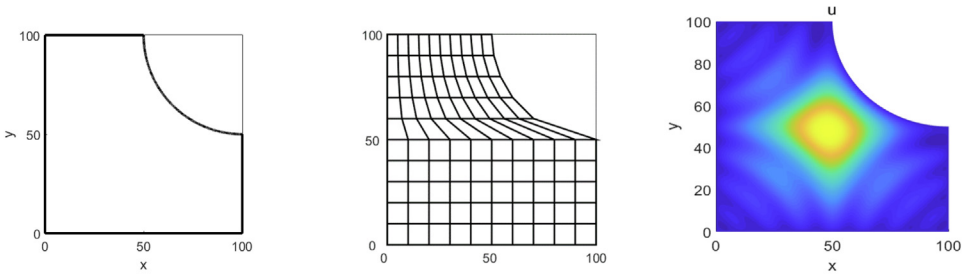
$$\boldsymbol{B}^{\gamma}(\tilde{x},\, \tilde{y}) = \boldsymbol{J}^{-1}\begin{bmatrix} (\boldsymbol{\beta}^{(q)})^T\dfrac{d\boldsymbol{N}^{\beta}(\tilde{x})}{d\tilde{x}}(\boldsymbol{N}^{\gamma}(\tilde{y}))^T \\ (\boldsymbol{\beta}^{(q)})^T\boldsymbol{N}^{\beta}(\tilde{x})(\dfrac{d\boldsymbol{N}^{\gamma}(\tilde{y})}{d\tilde{y}})^T \end{bmatrix}. \tag{F.23}$$

The stiffness matrix for $\gamma^{(q)}$ is

$$\boldsymbol{K}^{\gamma} = \int_{\tilde{\Omega}_{(\tilde{x},\tilde{y})}} \boldsymbol{B}^{\gamma,T}(\tilde{x},\, \tilde{y})\boldsymbol{B}^{\gamma}(\tilde{x},\, \tilde{y})\det(\boldsymbol{J})\mathrm{d}\tilde{x}\mathrm{d}\tilde{y}. \tag{F.24}$$

If we present a regular mesh over a regular domain $\Omega_{(x,y)}$, the mapping is a linear transformation for coordinates. Let one element of the regular mesh $[x_1^e,\, x_2^e] \times [y_1^e,\, y_2^e]$. The mapping (F.7) reduces to

$$x = \frac{x_2^e - x_1^e}{\tilde{x}_2^e - \tilde{x}_1^e}(\tilde{x} - \tilde{x}_1^e) + x_1^e,\; y = \frac{y_2^e - y_1^e}{\tilde{y}_2^e - \tilde{y}_1^e}(\tilde{y} - \tilde{y}_1^e) + y_1^e. \tag{F.25}$$




(a) Physical domain (b) FE mesh (c) FEM solution

**Fig. F.18.** Geometry and mesh for the plate. (a) Model of the plate. (b) 100 elements mesh as an illustration. (c) FEM solution with $4.90 \times 10^7$ elements.

$\boldsymbol{J}$ reduces to a diagonal matrix $\text{diag}(\frac{x_2^e - x_1^e}{\tilde{x}_2^e - \tilde{x}_1^e}, \frac{y_2^e - y_1^e}{\tilde{y}_2^e - \tilde{y}_1^e})$. Thus we have

$$\boldsymbol{J}^{-T}\boldsymbol{J}^{-1}\det(\boldsymbol{J}) = \frac{x_2^e - x_1^e}{\tilde{x}_2^e - \tilde{x}_1^e}\frac{y_2^e - y_1^e}{\tilde{y}_2^e - \tilde{y}_1^e}, \tag{F.26}$$

which is constant in each element and separated representation. Thus this method degenerates to the classical PGD. For irregular mesh, $\boldsymbol{J}^{-T}\boldsymbol{J}^{-1}\det(\boldsymbol{J})$ in (F.16) is the function of $\boldsymbol{x}$ and mostly non-separated representation. In the numerical implementation, we usually approximate it by a separated form using SVD technique, i.e.,

$$\boldsymbol{J}^{-T}\boldsymbol{J}^{-1}\det(\boldsymbol{J}) \approx \begin{bmatrix} \sum_a \phi_{11}^{(a)}(\tilde{x})\psi_{11}^{(a)}(\tilde{y}) & \sum_a \phi_{12}^{(a)}(\tilde{x})\psi_{12}^{(a)}(\tilde{y}) \\ \sum_a \phi_{12}^{(a)}(\tilde{x})\psi_{12}^{(a)}(\tilde{y}) & \sum_a \phi_{22}^{(a)}(\tilde{x})\psi_{22}^{(a)}(\tilde{y}) \end{bmatrix}. \tag{F.27}$$

This converts the 2D integral (F.16) to the product of 1D integrals along different directions ($\tilde{x}$ and $\tilde{y}$ directions in the reference domain), which might reduce the computational cost.

### *F.4. Numerical examples*

In this section, we study the performance of PGD using the above mapping technique with comparison to FEM. The first problem is the Poisson problem with a body force

$$b(x, y) = \sin\left(\frac{(x-50)(y-50)}{100} + \frac{\pi}{2}\right) \tag{F.28}$$

in an irregular plate as shown in Fig. F.18(a). We present a FE mesh with $7000 \times 7000$ elements first. Note that the meshing is not unique, and we adopt a simple way for convenience, as illustrated in Fig. F.18(b). Based on the same mesh, FEM and PGD with mapping technique are adopted to solve this problem. To study the performance of PGD, we define the relative $H^1$ norm error

$$difference_{H^1}^{PGD-FEM} = \frac{\|u^{PGD} - u^{FEM}\|_{H^1}}{\|u^{FEM}\|_{H^1}} = \frac{\sqrt{\int_\Omega (u^{PGD} - u^{FEM})^2 + (\nabla u^{PGD} - \nabla u^{FEM})^2 \mathrm{d}x\mathrm{d}y}}{\sqrt{\int_\Omega (u^{FEM})^2 + (\nabla u^{FEM})^2 \mathrm{d}x\mathrm{d}y}} \tag{F.29}$$

to measure the difference between FEM and PGD. The $difference_{H^1}^{PGD-FEM}$ vs. $Q$ curve is plotted in Fig. F.19, where $Q$ is the number of modes. It is clearly shown that the difference decreases to zero with increasing number of modes. When $Q > 34$, PGD reaches a good accuracy with less than a 3.00% difference from the FEM. The DoFs of FEM are $4.90 \times 10^7$, while those of PGD are merely $4.76 \times 10^5$ with $Q = 34$ modes.

The geometry of the second problem is a square plate with one hole, as shown in Fig. F.20(a). The body force is

$$b(x, y) = \sin\left(\frac{(x-5)(x-10)}{100} + \frac{\pi}{2}\right). \tag{F.30}$$

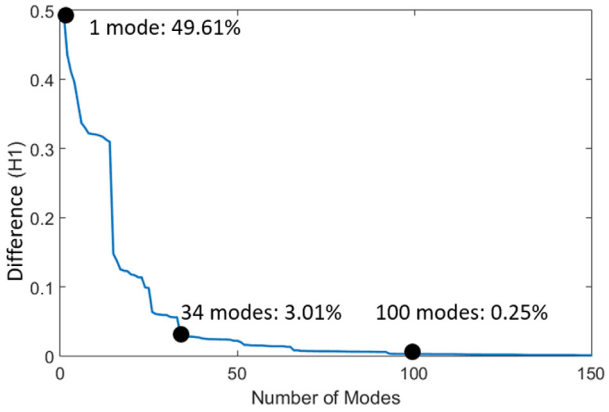
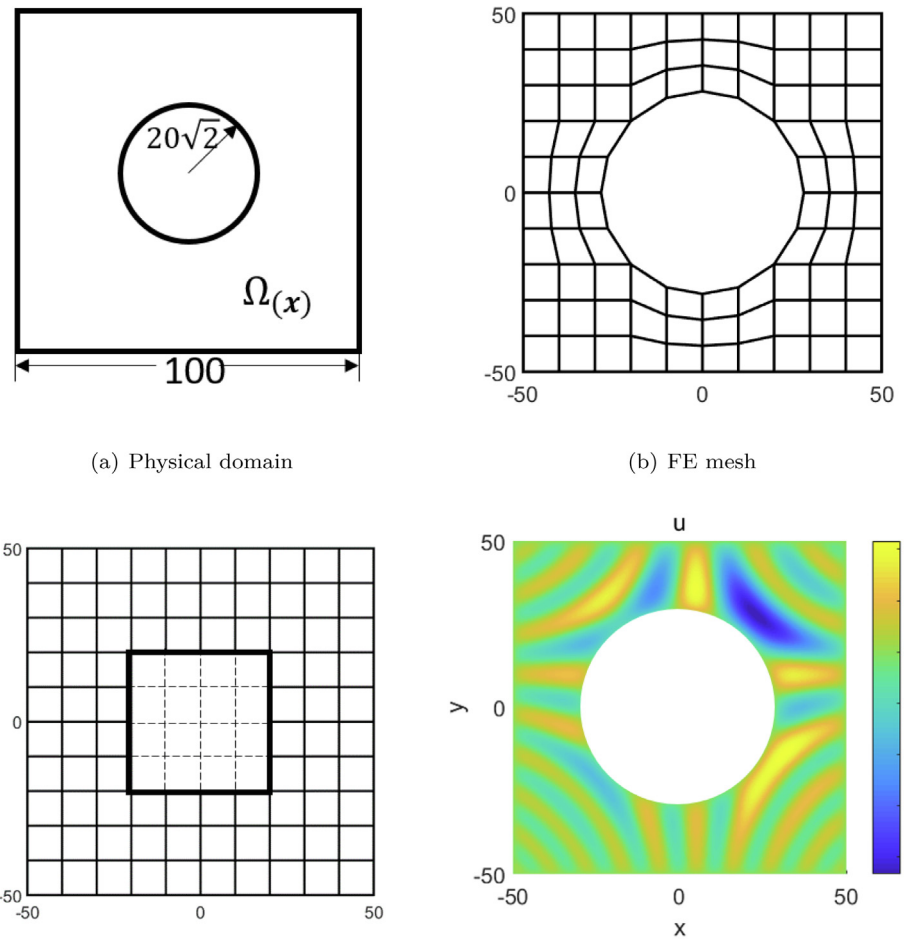



**Fig. F.19.** The $difference_{H^1}^{PGD-FEM}$ vs. $Q$ curve. $difference_{H^1}^{PGD-FEM}$ denotes the difference between FEM and PGD with mapping technique. $Q$ is the number of modes.




(a) Physical domain

(b) FE mesh

(c) Reference mesh

(d) FEM solution

**Fig. F.20.** Geometry and mesh for the plate with one hole. (a) Model of the plate with one hole. (b) 84 elements mesh as an illustration. (c) FEM mesh corresponding to 84 elements.. (d) FEM solution with $4.116 \times 10^7$ elements.

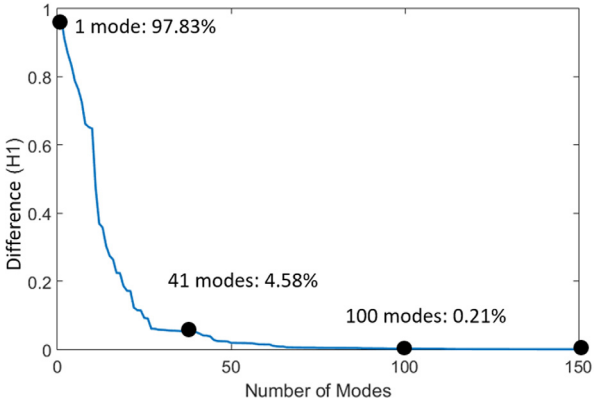



**Fig. F.21.** The $difference_{H^1}^{PGD-FEM}$ vs. $Q$ curve. $difference_{H^1}^{PGD-FEM}$ denotes the difference between FEM and PGD with mapping technique. $Q$ is the number of modes.

Fig. F.20(b) illustrates a FE meshing with 84 elements, and the transformed reference domain is a square plate with a square hole as shown in Fig. F.20(c). A fine mesh with $4.116 \times 10^7$ elements is used to solve this problem. The FEM solution is shown in Fig. F.20(d), and the difference between PGD and FEM is illustrated in Fig. F.21. The PGD result converges to FEM one when increasing number of modes. The difference maintains less than 5% when $Q > 41$.

These two examples verify the effectiveness of the mapping technique for space separated PGD to deal with the irregular domain.

## References

[1] T. Belytschko, W.K. Liu, B. Moran, K. Elkhodary, Nonlinear Finite Elements for Continua and Structures, John Wiley & Sons, 2013.
[2] T.J. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.
[3] W.K. Liu, S. Jun, Y.F. Zhang, Reproducing kernel particle methods, Internat. J. Numer. Methods Fluids 20 (8–9) (1995) 1081–1106.
[4] W.K. Liu, Y. Chen, S. Jun, J. Chen, T. Belytschko, C. Pan, R. Uras, C. Chang, Overview and applications of the reproducing kernel particle methods, Arch. Comput. Methods Eng. 3 (1) (1996) 3–80.
[5] S. Li, D. Qian, W.K. Liu, T. Belytschko, A meshfree contact-detection algorithm, Comput. Methods Appl. Mech. Engrg. 190 (24–25) (2001) 3271–3292.
[6] S. Li, W.K. Liu, Meshfree Particle Methods, Springer Publishing Company, Incorporated, 2007.
[7] K. Hornik, M. Stinchcombe, H. White, et al., Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[8] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314.
[9] W. E, J. Han, A. 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) (2017) 349–380.
[10] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[11] H. Li, O.L. Kafka, J. Gao, C. Yu, Y. Nie, L. Zhang, M. Tajdari, S. Tang, X. Guo, G. Li, S. Tang, G. Cheng, W.K. Liu, Clustering discretization methods for generation of material performance databases in machine learning and design optimization, Comput. Mech. 64 (2) (2019) 281–305.
[12] L. Zhang, L. Cheng, H. Li, J. Gao, C. Yu, R. Domel, Y. Yang, S. Tang, W.K. Liu, Hierarchical deep-learning neural networks: finite elements and beyond, Comput. Mech. 67 (1) (2021) 207–230.
[13] S. Saha, Z. Gan, L. Cheng, J. Gao, O.L. Kafka, X. Xie, H. Li, M. Tajdari, H.A. Kim, W.K. Liu, Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113452.
[14] J. Song, Optimal representation of high-dimensional functions and manifolds in low-dimensional visual space (in Chinese), Chin. Sci. Bull. 46 (12) (2001) 977–984.
[15] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids, J. Non-Newton. Fluid Mech. 139 (3) (2006) 153–176.
[16] F. Chinesta, P. Ladeveze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Arch. Comput. Methods Eng. 18 (4) (2011) 395–404.

[17] M. Bhattacharyya, A. Fau, U. Nackenhorst, D. Néron, P. Ladevèze, A multi-temporal scale model reduction approach for the computation of fatigue damage, Comput. Methods Appl. Mech. Engrg. 340 (2018) 630–656.
[18] D. Modesto, S. Zlotnik, A. Huerta, Proper generalized decomposition for parameterized helmholtz problems in heterogeneous and unbounded domains: application to harbor agitation, Comput. Methods Appl. Mech. Engrg. 295 (2015) 127–149.
[19] Y. Lu, N. Blal, A. Gravouil, Adaptive sparse grid based HOPGD: Toward a nonintrusive strategy for constructing space-time welding computational vademecum, Internat. J. Numer. Methods Engrg. 114 (13) (2018) 1438–1461.
[20] Y. Lu, N. Blal, A. Gravouil, Datadriven HOPGD based computational vademecum for welding parameter identification, Comput. Mech. 64 (1) (2019) 47–62.
[21] Y. Lu, N. Blal, A. Gravouil, Multi-parametric space-time computational vademecum for parametric studies: Application to real time welding simulations, Finite Elem. Anal. Des. 139 (2018) 62–72.
[22] P. Díez, S. Zlotnik, A. García-González, A. Huerta, Algebraic PGD for tensor separation and compression: an algorithmic approach, C. R. Méc. 346 (7) (2018) 501–514.
[23] A. Giacoma, D. Dureisseix, A. Gravouil, An efficient quasi-optimal space-time PGD application to frictional contact mechanics, Adv. Model. Simul. Eng. Sci. 3 (1) (2016) 1–17.
[24] A. Nouy, A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations, Comput. Methods Appl. Mech. Engrg. 199 (23–24) (2010) 1603–1626.
[25] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[26] B. Bognet, A. Leygue, F. Chinesta, Separated representations of 3D elastic solutions in shell geometries, Adv. Model. Simul. Eng. Sci. 1 (1) (2014) 1–34.
[27] S. Tang, Y. Yang, Why neural networks apply to scientific computing? Theor. Appl. Mech. Lett. 11 (3) (2021) 100242.
[28] J.-L. Wu, J.-X. Wang, H. Xiao, J. Ling, A priori assessment of prediction confidence for data-driven turbulence modeling, Flow Turbul. Combust. 99 (1) (2017) 25–46.
[29] H. Xiao, J.L. Wu, J.X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed Bayesian approach, J. Comput. Phys. 324 (2016) 115–136.
[30] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[31] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017, arXiv preprint arXiv:1711.10561.
[32] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[33] W. E, B. Yu, The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[34] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[35] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall PTR, 1994.
[36] A. Oishi, G. Yagawa, Computational mechanics enhanced by deep learning, Comput. Methods Appl. Mech. Engrg. 327 (2017) 327–351.
[37] D. González, A. Ammar, F. Chinesta, E. Cueto, Recent advances on the use of separated representations, Internat. J. Numer. Methods Engrg. 81 (5) (2010) 637–659.
[38] N. Blal, A. Gravouil, Non-intrusive data learning based computational homogenization of materials with uncertainties, Comput. Mech. 64 (3) (2019) 807–828.
[39] F. Chinesta, R. Keunings, A. Leygue, The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer, Springer Science & Business Media, 2014.
[40] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[41] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, IEEE, 1995, pp. 1942–1948.
[42] C. Ghnatios, E. Abisset, A. Ammar, E. Cueto, J.-L. Duval, F. Chinesta, Advanced separated spatial representations for hardly separable domains, Comput. Methods Appl. Mech. Engrg. 354 (2019) 802–819.