

Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0)

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

<https://doi.org/10.1016/j.cma.2023.116356>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

**Please provide feedback**

Please support the ScholarWorks@UMBC repository by emailing [scholarworks-group@umbc.edu](mailto:scholarworks-group@umbc.edu) and telling us what having access to this work means to you and why it's important to you. Thank you.



# Isogeometric Convolution Hierarchical Deep-learning Neural Network: Isogeometric analysis with versatile adaptivity

Lei Zhang<sup>a,b,1</sup>, Chanwook Park<sup>c,1</sup>, Ye Lu<sup>d</sup>, Hengyang Li<sup>c</sup>, Satyajit Mojumder<sup>e</sup>,  
Sourav Saha<sup>e</sup>, Jiachen Guo<sup>e</sup>, Yangfan Li<sup>c</sup>, Trevor Abbott<sup>c</sup>, Gregory J. Wagner<sup>c</sup>,  
Shaoqiang Tang<sup>f</sup>, Wing Kam Liu<sup>c,\*</sup>

<sup>a</sup> The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China

<sup>b</sup> School of Engineering Science, University of Chinese Academy of Sciences, Beijing 100049, China

<sup>c</sup> Department of Mechanical Engineering, Northwestern University, Evanston, USA

<sup>d</sup> Department of Mechanical Engineering, University of Maryland Baltimore County, Baltimore 21250, USA

<sup>e</sup> Theoretical and Applied Mechanics Program, Northwestern University, Evanston, USA

<sup>f</sup> HEDPS and LTCS, College of Engineering, Peking University, Beijing 100871, China

Available online xxx

## Abstract

We are witnessing a rapid transition from Software 1.0 to 2.0. Software 1.0 focuses on manually designed algorithms, while Software 2.0 leverages data and machine learning algorithms (or artificial intelligence) for optimized, fast, and accurate solutions. For the past few years, we have been developing Convolution Hierarchical Deep-learning Neural Network Artificial Intelligence (C-HiDeNN-AI), which enables the realization of *Engineering Software 2.0* by opening the next-generation neural network-based computational tools that can simultaneously train data and solve mechanistic equations. This paper focuses on solving partial differential equations with C-HiDeNN. Still, the same neural network can be used for training and calibration with experimental data, which will be discussed in a separate paper. This paper presents a computational framework combining the C-HiDeNN theory with isogeometric analysis (IGA), called Convolution IGA (C-IGA). C-IGA has five key features that advance IGA: (1) arbitrarily high-order smoothness and convergence rates without increasing degrees of freedom; (2) a Kronecker delta property that enables direct imposition of Dirichlet boundary conditions; (3) automatic and flexible global/local mesh-adaptivity with built-in length scale control and adjustable radial basis functions; (4) ability to handle irregular meshes and triangular/tetrahedral elements; and (5) GPU implementation that speeds up the program as fast as finite element method (FEM). Mathematically, we prove that both IGA and C-IGA mappings are equivalent, and by taking a special design and modified anchors as nodes, C-IGA degenerates to IGA. We demonstrate the accuracy, convergence rates, mesh-adaptivity, and performance of C-IGA with several 1D, 2D, and 3D numerical examples. The future applications of C-IGA from topology optimization to product manufacturing with multi-GPU programming are discussed.

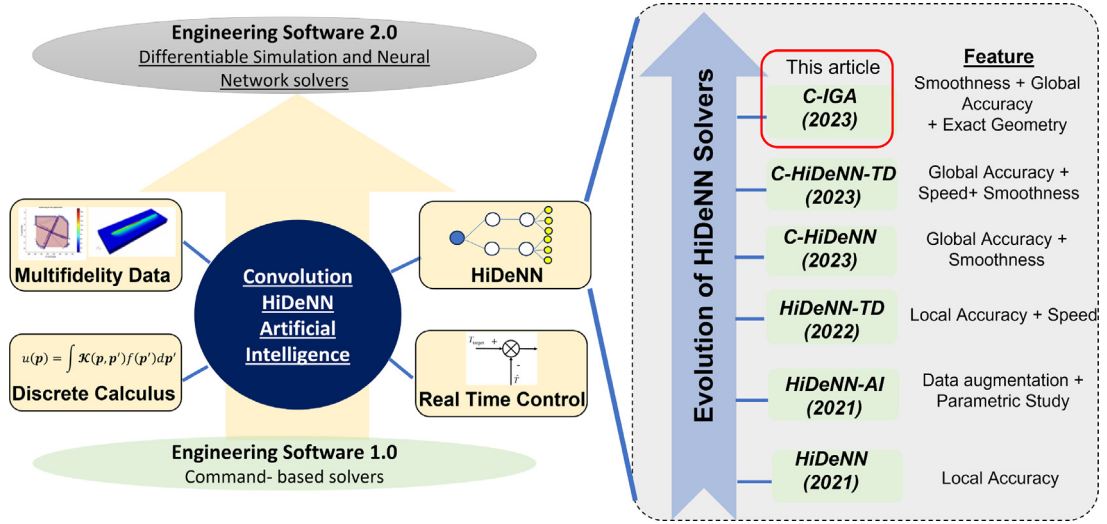
© 2023 Elsevier B.V. All rights reserved.

**Keywords:** Convolution isogeometric analysis (C-IGA); Convolution hierarchical deep-learning neural network (C-hiDeNN); Software 2.0; r-h-p-s-a adaptive finite element method (FEM); High-order smoothness and convergence

\* Corresponding author.

E-mail address: [w-liu@northwestern.edu](mailto:w-liu@northwestern.edu) (W.K. Liu).

<sup>1</sup> These authors contributed equally to this work.



**Fig. 1.** The components of Convolution Hierarchical Deep-learning Neural Network Artificial Intelligence system for engineering software 2.0. The enlarged figure on the right shows the evolution of HiDeNN solvers with key features [9–13].

## 1. Introduction

The field of computational science and engineering is witnessing a paradigm shift in both algorithms and software. Computational scientists are developing methods that rely on training neural networks from experimental or computational data. In a famous blog post [1], Andrej Karpathy defined this new generation of software as *software 2.0*. Later, Erik Meijer elaborated on the idea [2] by explaining two drastically different approaches of software development: *software 1.0* where the developers need to specify the problem, design the algorithms, and break down a complex task into multiple smaller components, and *software 2.0* where the developers use data, design an optimization problem, and train a neural network to solve any problem. In this article, the authors adopt the latter vision of next-generation software and propose a *theory* called *Convolution Hierarchical Deep-learning Neural Network Artificial Intelligence (C-HiDeNN-AI)* that will transform the current generation of *Engineering Software 1.0* to *Engineering Software 2.0*. One of the biggest advantages of using *software 2.0* is the use of *automatic differentiation* [3] to minimize the Lagrangian form of any governing equation. This is defined as *differentiable simulation* [4]. For real-time control and optimization of the manufacturing processes, one needs both a fast and differentiable tool, as it involves computing derivatives of the Lagrangian many times [5]. Currently, many works rely on reduced-order method to avoid computing a more accurate but computationally intensive solution of the governing equations [6–8]. However, the HiDeNN solvers construct a general neural network structure that can embed the parametric space into the fast solvers with increased accuracy and either learn or train with or without data.

The main components of the C-HiDeNN-AI are shown in Fig. 1.

The first component of C-HiDeNN-AI is multifidelity data. This data is required for an AI to learn the appropriate physics of the system. For example, in the case of designing a feedback control system for metal additive manufacturing, one needs to monitor the temperature data coming from the specimen or the melt pool. However, the data can come from multiple sources and can be used to train the entire network or some part of it while the rest of the network solves partial differential equations.

The second building block is discrete calculus. The discrete calculus formulations, especially kernel and convolution-based equations, form the basis of the HiDeNN solvers. Kernel-based operator learning methods [14] and deep learning interpretation of discrete calculus algorithms [15] have demonstrated that neural networks can essentially degenerate into integral equations with each layer representing one particular integral transform. This approach is already established in the literature of partial differential equation solvers [16–25]. Based on this method, any functional relationship between an input  $f(\mathbf{p})$  and an output  $u(\mathbf{p})$  can be expressed as

$$u(\mathbf{p}) = \int \mathcal{K}(\mathbf{p}, \mathbf{p}') f(\mathbf{p}') d\mathbf{p}', \quad (1)$$

**Table 1**

A Summary of the HiDeNN Solvers based on Eq. (2).

Parameters	Physical quantity	Method(s) that consider the parameter in the kernel
$p_1$	Spatial $X$ -direction	HiDeNN, HiDeNN-AI, HiDeNN-TD, C-HiDeNN, C-HiDeNN-TD
$p_2$	Spatial $Y$ -direction	HiDeNN, HiDeNN-AI, HiDeNN-TD, C-HiDeNN, C-HiDeNN-TD
$p_3$	Spatial $Z$ -direction	HiDeNN, HiDeNN-AI, HiDeNN-TD, C-HiDeNN, C-HiDeNN-TD
$p_4$	Time	HiDeNN-AI, HiDeNN-TD, C-HiDeNN, C-HiDeNN-TD
$p_5$	Manufacturing Parameter(s)	HiDeNN-AI, HiDeNN-TD, C-HiDeNN-TD
$p_6$	Physical Parameter(s)	HiDeNN-AI, HiDeNN-TD, C-HiDeNN-TD
$p_7$	Geometric Boundary Conditions	<b>C-IGA</b>

where the vector  $\mathbf{p} = [p_1, p_2, \dots, p_P]$  represents the parameter vector and  $P$  is the total number of parameters.  $\mathcal{K}(\cdot, \cdot)$  is the kernel function for interpolation. The HiDeNN solvers convert this kernel into custom neural networks. The parameter vector can include spatial variables such as  $p_1 = x$ ,  $p_2 = y$ ,  $p_3 = z$ ; temporal variables such as  $p_4 = t$ ; and system parameters such as scan speed for an additive manufacturing process  $p_5 = v$ . For large-scale engineering problems,  $\mathbf{p}$  can be extended to even higher dimensions. To counter the curse of dimensionality and improve the speed of calculations, the kernel can be split into multiple directional convolution kernels by the following formulation:

$$u(\mathbf{p}) = \int \mathcal{K}_1(p_1, p'_1) \int \mathcal{K}_2(p_2, p'_2) \cdots \int \mathcal{K}_P(p_P, p'_P) f(\mathbf{p}) dp'_1 dp'_2 \cdots dp'_P, \quad (2)$$

where elements of  $\mathbf{p}$  are independent of each other.

Based on this idea, a family of HiDeNN solvers has been developed over recent years (see, the third building block in Fig. 1). One feature of these HiDeNN solvers is that they are designed to leverage modern hardware such as a Graphics Processing Unit (GPU) and a Tensor Processing Unit (TPU). The TPUs are specifically designed to train sufficiently deep neural networks [26]. The C-HiDeNN-AI has already demonstrated its superior efficiency over serial programming when implemented with a single GPU [12]. However, multi-GPU programming has become a standard for extremely large-scale scientific computing and generative AI training. For example, ChatGPT is known to be trained with over 10K NVIDIA H100 GPUs [27]. Therefore, the development of C-HiDeNN-AI software that can fully leverage the unprecedentedly fast-growing GPU technology is now more essential than ever. For the convenience of readers, a summary of this evolution is given below (also please see Table 1):

- **HiDeNN:** HiDeNN consists of a custom neural network where a group of neurons from each layer mimics the interpolation of finite element shape functions [9] that can be either **trained or solved**. If a HiDeNN is trained, a single dataset of independent variables is required and nodal adaptivity ( $r$ -adaptivity in FEM) is achieved through training by minimizing the loss function. When solved, no initial dataset is required. The HiDeNN minimizes the energy of the physical system with nodal adaptivity. Applications of HiDeNN-FEM to nonlinear problems have been discussed in [28].
- **HiDeNN-AI:** Saha et al. [29] proposed an artificial intelligence system that can enrich the HiDeNN with computational or experimental data and perform a *parametric* study. The article proposed three general classes of problems: *type 1* for which a lot of data is available with little physical insight; *type 2* where some physics is missing that can be recovered from data, and *type 3* where a fast reduced order model is required to solve large scale problems.
- **HiDeNN-Tensor Decomposition (TD):** To improve the speed of the HiDeNN, a tensor decomposition method is introduced [10,30] where the kernels of Eq. (2) are approximated by tensor decomposition of the space-time-parameter matrix and superposition of the modes.
- **Convolution HiDeNN (C-HiDeNN):** The HiDeNN method is extended to convolution HiDeNN (C-HiDeNN) to increase the smoothness and accuracy of the solution without increasing the degrees of freedom (DoFs) to solve [11,12]. This smoothness is achieved by designing an interpolation that accounts for additional layers of neighborhood elements.



- **Convolution HiDeNN (C-HiDeNN)-Tensor Decomposition (TD):** To improve the speed of C-HiDeNN and solve problems spanning multiple length scales, C-HiDeNN-TD is developed [11,13]. With this method, an unprecedented speed up is achieved for multi-scale and multi-material topology optimization [13]. A demonstration will be presented in Section 6. Combining topology optimization with multi-scale structures holds promise for multi-functional design with superior performance [31–33].
- **Isogeometric Convolution HiDeNN (C-IGA) (Current Article):** In the previous methods, the spatial, temporal, and parameter spaces were integrated successfully. However, geometric accuracy can be an important parameter for manufacturing. Modern 3D printers can materialize virtually any design if supplied with a Computer-Aided Design (CAD) geometry. The quality of the final build depends largely on how smooth and geometrically accurate the supplied CAD geometry is. It has been demonstrated in the literature that isogeometric analysis (IGA) can capture the complex geometry of a problem [34–36], which inspired the current development of C-IGA that integrates isogeometric shape functions with C-HiDeNN. With the flexibility of the shape functions used in C-IGA, our method can achieve more versatile adaptivity in the solution, which is currently not available in FEM or IGA.

**Remark 1.** The HiDeNN-TD solvers use Eq. (2) to counter the *curse of dimensionality*. In *software 1.0*, for three space dimensions, one time dimension, and  $(P-4)$  parameters where  $P \geq 4$ , the total number of degrees of freedom (DoFs) increase exponentially with problem dimensions:  $(n^3 \times n \times n^{(P-4)}) \approx O(n^P)$ , assuming  $n$  DoFs for each dimension. In HiDeNN-TD, with  $Q$  modes, the total DoFs only increase linearly:  $Q \times (n \times 3 + n + n \times (P-4)) \approx O(Q \times n \times P)$ .

In all of these solvers, the physical equations, such as the minimum of potential energy (when it exists), are converted to an optimization problem via loss function and solved for the solutions. This loss function can be designed to match a desired control response from the system in real-time. Thus, the C-HiDeNN-AI aims to combine all the elements shown in Fig. 1 to produce a software environment that can reduce the design to the final product cycle in the manufacturing plant with orders of magnitude speed up.

The final building block in Fig. 1 is the real-time control. To ensure real-time control, we need a fast, reliable, and accurate solver that can be integrated directly in manufacturing hardware. The proposed C-HiDeNN-AI provides the flexibility of training a neural network from data while keeping the structure of the network similar as solving. By generating the data using the *solver*, and training the network with the data, we can achieve the real-time control of manufacturing system for better product.

The remainder of the article is organized as follows: Section 2 will review the latest C-HiDeNN solvers based on which the current method is developed, and perform an overview of versatile adaptivity for C-HiDeNN framework. Section 3 will present the building blocks of the C-IGA method with explanatory sketches and present a comparative discussion between the IGA and C-IGA including how C-IGA achieves versatile adaptivity. Section 4 will present an analysis of the continuity and convergence of the method. Section 5 will discuss some numerical examples applying the C-IGA method. Finally, Section 6 will propose some future extensions of the proposed solver.

## 2. Review of the C-HiDeNN framework

(\* Starting from this section, all mathematical notations and symbols are defined in Table 2.)

### 2.1. C-HiDeNN approximation

C-HiDeNN borrows the same concept as HiDeNN that constructs a general approximation using neural networks with predefined coefficients and activation functions, but it has one additional convolution layer to enable nonlinear activation and versatile mesh adaptivity. The HiDeNN concept and its relation with the C-HiDeNN can be found in Appendix A. Considering an element-wise approximation, the C-HiDeNN approximation can be written as

$$\mathbf{u}^{C-HiDeNN}(\tilde{\xi}) = \sum_{i \in A^e} \mathcal{N}_i(\tilde{\xi}) \sum_{j \in A_s^{\tilde{\xi}_i}} \mathcal{W}_{a,j}^{\tilde{\xi}_i}(\tilde{\xi}) \mathbf{u}_j = \sum_{k \in A_s^e} \tilde{N}_k(\tilde{\xi}) \mathbf{u}_k. \quad (3)$$

where  $\tilde{\xi}$  refers to the parent coordinates with  $\tilde{\xi} \in \tilde{\Omega}^e = [-1, 1]^d$  with  $d$  being the problem dimension.  $\mathbf{u}_j$  is the nodal coefficient. The mapping between the parent domain and the physical domain can be defined as isoparametric

**Table 2**

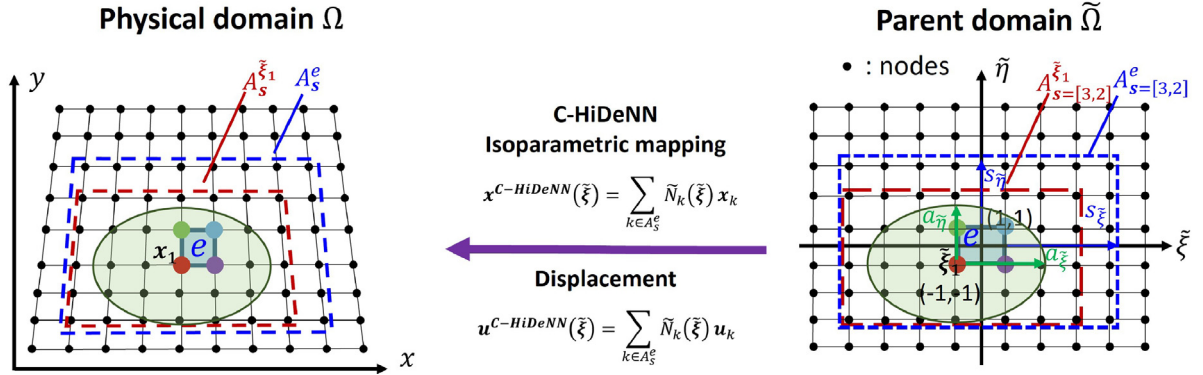
Nomenclature.

Symbol	Description
$\Omega, \hat{\Omega}, \tilde{\Omega}$	Physical, parametric, parent domain
$X, x_i$	Global nodal coordinates and nodal coordinates of node $i$ in physical domain
$\chi, \xi_i$	Global nodal coordinates and nodal coordinates of node $i$ in parametric domain
$d$	Dimension of problem
$s, a, p$	Patch size, dilation parameter, polynomial order
$A^e$	Set of nodes in element $e$
$A_s^{\xi_i}, A_s^{\xi_i}$	Set of nodes in a patch domain of node $i$ for C-HiDeNN and C-IGA
$A_s^e$	Set of nodes in a patch domain of element $e$ , $A_s^e = \bigcup_{i \in A^e} A_s^{\xi_i}$
$x_{GP}^e$	Gauss point of element $e$
$\tilde{\xi}_{GP}, w_{GP}$	Gauss points and weights in parent domain
$U, u_i$	Global nodal variables and nodal variable of node $i$
$U_s^i$	Column vector of nodal variables corresponding to $A_s^{\xi_i}$ or $A_s^{\xi_i}$
$\mathcal{N}_i(\tilde{\xi}), \mathcal{N}_i(\xi)$	Compact supported interpolation function at node $i$
$\mathcal{W}_{a,j}^{\xi_i}(\tilde{\xi}), \mathcal{W}_{a,j}^{\xi_i}(\xi)$	Convolution patch function of node $j$ , constructed for $A_s^{\xi_i}$ or $A_s^{\xi_i}$ with a dilation parameter of $a$
$\mathcal{W}_{a,s}^{\xi_i}(\tilde{\xi}), \mathcal{W}_{a,s}^{\xi_i}(\xi)$	Vector of convolution patch functions constructed for $A_s^{\xi_i}$ or $A_s^{\xi_i}$ with a dilation parameter of $a$
$\tilde{N}_k(\tilde{\xi}), \tilde{N}_k(\xi)$	C-HiDeNN and C-IGA shape functions
$P(\tilde{\xi}), P(\xi)$	Vector of basis functions to be reproduced
$\Psi_a(\tilde{\xi}), \Psi_{a,j}(\tilde{\xi})$	Vector of radial basis functions and its one component for node $j$
$J(\tilde{\xi}), J(\xi)$	Jacobian matrix for physical domain in C-HiDeNN and C-IGA
$\tilde{J}(\tilde{\xi})$	Jacobian matrix for parametric domain in C-IGA
$\Pi, \Pi^e$	Variational energy of global domain and element $e$
$D$	Elasticity matrix
$\tilde{B}_s^e$	Strain–displacement matrix of elemental patch $A_s^e$
$\tilde{N}_s^e$	C-HiDeNN or C-IGA shape functions for elemental patch $A_s^e$
$R_l(\xi), B_l$	Basis functions and coefficients for general geometric mapping
$np$	Number of global nodes
$ns$	Number of nodes in the nodal patch $A_s^{\xi_i}$
$h, h_{\xi}$	Element size in physical, parametric domain
$u^{Ext}, u^h$	Exact solution, numerical solution with a element size $h$

mapping, as illustrated in Fig. 2.  $\mathcal{N}_i(\tilde{\xi})$  is a function with compact support associated with node  $\tilde{\xi}_i$  of an element, and  $\mathcal{W}_{a,j}^{\xi_i}(\tilde{\xi})$  is a local convolution patch function with a dilation parameter  $a$  that interpolates the neighborhood ( $A_s^{\xi_i}$ , called *nodal patch*) of the node  $\tilde{\xi}_i$ . For example, in Fig. 2, element  $e$  has four nodes,  $A^e = \{\tilde{\xi}_1, \tilde{\xi}_2, \tilde{\xi}_3, \tilde{\xi}_4\}$ . The function  $\mathcal{N}_1(\tilde{\xi})$  is the interpolant of node  $\tilde{\xi}_1$  among the four nodes. The function  $\mathcal{W}_{a,j}^{\xi_i}(\tilde{\xi})$  interpolates the patch domain of node  $\tilde{\xi}_1$ , which is  $A_s^{\xi_1}$ , where  $s$  is the size of the nodal patch and indicates the number of layers of nodes surrounding the center node. In general,  $s$  can be different in different directions (see Fig. 2 where  $s = [3, 2]$  in two different directions respectively), similarly for the dilation parameter  $a = [a_{\xi}, a_{\eta}]$ . In the current work, we restrict the discussion to a uniform case with the same scalar  $s$  and  $a$  for all directions. The *elemental patch* domain of element  $e$  is defined by  $A_s^e = \bigcup_{i \in A^e} A_s^{\xi_i}$  (blue dashed box in Fig. 2). The final approximation can be written as a product of the convolution shape functions and the nodal coefficients in the *elemental patch*.

The convolution patch function  $\mathcal{W}_{a,j}^{\xi_i}$  plays a similar role as the kernel function in convolution neural networks, which connects the neighboring nodes and is expected to improve the solution smoothness. The function  $\mathcal{N}_i(\tilde{\xi})$  provides a compact support for the final interpolant  $\tilde{N}_k(\tilde{\xi})$ . Various choices of functions  $\mathcal{N}_i$  and  $\mathcal{W}_{a,j}^{\xi_i}$  have been discussed in [11]. In general,  $\mathcal{N}_i$  can be any function with compact support, such as finite element [37] and isogeometric [34] basis functions. Meshfree type approximations (e.g., [16,38–40]) can be used for the local convolution patch function  $\mathcal{W}_{a,j}^{\xi_i}$ . We can design the C-HiDeNN approximation to satisfy the Kronecker delta and reproducing properties, as discussed in [11,12].

Among all possible combinations of the two interpolants ( $\mathcal{N}_i$  and  $\mathcal{W}_{a,j}^{\xi_i}$ ) of C-HiDeNN, we consider a special case, C-FEM [11], where FEM shape functions are used for  $\mathcal{N}_i$ . The radial basis function [41,42] is chosen for



**Fig. 2.** 2D C-HiDeNN isoparametric mapping between the parent domain and the physical domain. An element  $e$  is drawn in the parent domain such that the element lies in  $[-1, 1]^{d=2}$ . The nodal patch  $A_s^{\xi_i}$  and the elemental patch  $A_s^e$  are shown. In general, the patch size  $s$  can be a vector, meaning that it can have different patch sizes for each direction.  $s = [3, 2]$  means that the patch domain comprises three layers of elements in  $\xi$ -direction and two layers of elements in  $\eta$ -direction. The green ellipse represents the dilation domain with a dilation parameter  $a$ . Likewise,  $a$  can have different quantities for each direction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$\mathcal{W}_{a,j}^{\xi_i}$ , defined by

$$\mathcal{W}_{a,j}^{\xi_i} = \Psi_a(\xi) \alpha_j + P(\xi) \kappa_j, \quad (4)$$

where  $\Psi_a(\xi)$  is a vector composed of  $\Psi_{a,j}(\xi)$ , a radial basis function with a dilation of  $a$ . Here,  $a$  normalizes the radial distance between the point of interest  $\xi$  and the center node  $\xi_i$ . For example,  $\Psi_{a,j}(\xi)$  can be written as

$$\Psi_{a,j}(\xi) = \psi \left( \frac{|\xi - \xi_j|}{a} \right) \quad (5)$$

with a cubic spline function

$$\psi(z) = \begin{cases} \frac{2}{3} - 4z^2 + 4z^3 & \forall z \in [0, \frac{1}{2}] \\ \frac{4}{3} - 4z + 4z^2 - \frac{4}{3}z^3 & \forall z \in [\frac{1}{2}, 1] \\ 0 & \forall z \in (1, +\infty) \end{cases} \quad (6)$$

or a Gaussian function

$$\psi(z) = e^{-z^2}, \quad (7)$$

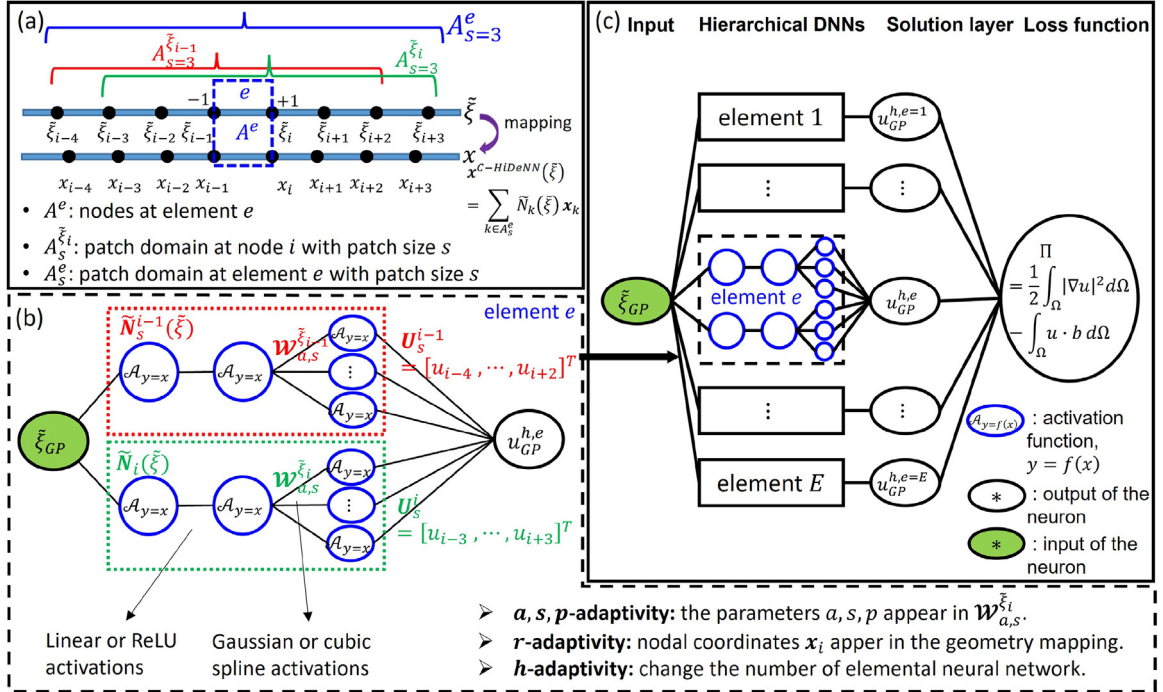
where  $z = |\xi - \xi_j|/a$ .  $P(\xi)$  is a vector of functions that are expected to be reproduced, such as  $p$ th order polynomials.  $\alpha_j$  and  $\kappa_j$  are coefficients given in [Appendix B](#).

The neural network structure of C-HiDeNN approximation is shown in [Fig. 3](#). The solution field  $u^{C-HiDeNN}$  is a function of nodal solutions  $U$ , nodal coordinates  $X$ , and variable parameters  $a, s, p$ . The nodal solutions of the partial differential equation can be found by solving the following optimization problem:

$$\text{Solve } U = \arg \min_{u_i \in \mathbb{R}} \Pi(u^{C-HiDeNN}(x; U, X, a, s, p)), \quad (8)$$

where  $u_i$  is the  $i$ th component of  $U$  and the function  $\Pi(\ast)$  can be viewed as an operator that computes the variational energy. Here, we fix all parameters  $X, a, s, p$  except the nodal solution  $U$ . This is mathematically equivalent to solving the variational equation like standard FEM using the matrix form, which is elucidated in [Appendix C](#).

$$\text{Train } X, a, s, p = \arg \min_{x_i \in \Omega \setminus \partial \Omega, a \in \mathbb{R}^+, s \in \mathbb{N}, p \in \mathbb{N}} \Pi(u^{C-HiDeNN}(x; U, X, a, s, p)) \quad (9)$$



**Fig. 3.** Neural network structure of C-HiDeNN for a 1D scientific problem. (a) Nodal coordinates in the parent domain and physical domain whose mapping is defined by the C-HiDeNN interpolation in an isoparametric manner:  $x^{C-HiDeNN}(\xi) = \sum_{k \in A_s^e} \tilde{N}_k(\xi) x_k$ . For each node in element  $e$ , the nodal patch domain  $A_s^{i-1}$  is defined with a patch size  $s$ . For example,  $A_{s=3}^{i-1} = \{\xi_{i-4}, \dots, \xi_{i+2}\}$  and  $A_{s=3}^e = \{\xi_{i-3}, \dots, \xi_{i+3}\}$ . The elemental patch domain is the union of all nodal patches of an element:  $A_s^e = \bigcup_{i \in A^e} A_s^{i-1}$ . (b) Elemental neural network of C-HiDeNN. The first two hidden layers have linear or ReLU activation functions while the third hidden layer (whose weights are called convolution patch function,  $\mathcal{W}_{a,s}^{\xi_i}$ ) has Gaussian or cubic spline activation functions that enable nonlinear interpolation. Due to this third hidden layer, the C-HiDeNN interpolation can have versatile refinements:  $r, h, p, a, s$ -adaptivity (see Section 5.1 for numerical results). Unlike HiDeNN or FEM, the nodal shape function of C-HiDeNN ( $\tilde{N}_i^j(\xi)$ ) and the nodal variable  $U_s^i$  are column vectors of length  $n(A_s^{i-1})$ , which is elucidated in Appendix C. (c) The C-HiDeNN structure for the entire problem. The variational energy is used as a loss function of the problem. For more details about the neural network structure of C-HiDeNN, readers may refer to [12].

where  $\partial\Omega$  is the domain boundary, respectively, and  $\mathbb{R}^+$  and  $\mathbb{N}$  are the set of positive real numbers and natural numbers, respectively. The optimization of  $X$  is referred to  $r$ -adaptivity in FEM literature, but the other parameters do not have counterparts in standard FEM. There exists a difference in the optimization of parameters  $a, s, p$ : they are optimized patch by patch, while nodal coordinates  $X$  are optimized node by node. That is to say, all convolution patch functions  $\mathcal{W}_{a,s}^{\xi_i}$  in the nodal patch  $A_s^{i-1}$  always share the same parameters  $a, s, p$  in the optimization, and these parameters can be different in different nodal patches. In addition, unlike most adaptive FEM exploits a posteriori error estimator, HiDeNN theory only utilizes automatic differentiation and objective function (i.e., variational energy) for various adaptivity. Detailed implementations are discussed in the next section.

## 2.2. Discussions on parameters: $p, s$ and $a$

There are three controlling parameters ( $p, s, a$ ) in C-HiDeNN interpolation, which can be optimized to enhance solution efficiency for a specific problem.

- Positive integer  $p$  refers to the order of  $P(\xi)$  in the convolution patch functions in Eq. (4), which directly relates to the order of reproducing properties. The reproducing property will directly affect the convergence rate [11,22,43].
- Positive integer  $s$  is the size of the nodal patch  $A_s^{i-1}$  that is used as an influence domain for computing the convolution patch functions  $\mathcal{W}_{a,s}^{\xi_i}$ . The nodal patch  $A_s^{i-1}$  in  $d$ -dimensional space comprises  $(2s+1)^d$  nodes centered

**Table 3**

Conditions for parameters  $p, s, a$  in C-HiDeNN.

Parameters	$p$	$s$	$a$
Conditions	$p \in \mathbb{N}^+$	$s \geq p/2, s \in \mathbb{N}^+$	$a \in \mathbb{R}^+$

at node  $\tilde{\xi}_i$  if the mesh is uniform. Note that we need enough nodes (at least  $(p + 1)$  nodes along one dimension) to guarantee the  $p$ th order reproducing property, which leads to:  $s \geq p/2$ . Increasing  $s$  means enhancing the nodal connectivity, but this also leads to a larger bandwidth of the global stiffness matrix and computational cost.

- Positive real number  $a$  is the radial dilation of the radial basis function  $\Psi_a(\xi)$  in the convolution patch functions in Eq. (4). It controls the characteristic length scale of the radial function, and further influences the shape of the convolution patch function. We can adjust this parameter to change the shape and characteristic length scale of shape functions without changing the mesh and the reproducing property. The selection of  $a$  is independent of  $s$  and  $p$ .

We summarize conditions for the three parameters in Table 3.

The extension of C-HiDeNN to isogeometric analysis will be presented in the next section. The adaptivity of the Isogeometric C-HiDeNN (C-IGA) with respect to different controlling parameters will be discussed in detail.

### 3. Isogeometric C-HiDeNN (C-IGA) with two mappings

#### 3.1. Isoparametric mapping with exact geometry

Like traditional FEM, the FE mesh used in C-HiDeNN approximation does not guarantee the geometric accuracy in the numerical analysis. We borrow the idea of IGA [34] and propose the so-called isogeometric C-HiDeNN, shortly C-IGA, to link directly from CAD (Computer Aided Design) models to numerical analysis. Compared to C-HiDeNN, there is an additional intermediate domain called the parametric domain between the physical and parent domains in C-IGA, which is similar to that of IGA. The three domains of IGA and C-IGA are illustrated in Fig. 4 (IGA) and Fig. 5 (C-IGA). The domains of element  $e$  in the physical, parametric, and parent domains are denoted by  $\Omega^e$ ,  $\hat{\Omega}^e$ ,  $\tilde{\Omega}^e$  respectively, and corresponding coordinate systems are denoted by  $\mathbf{x}$ ,  $\xi$ , and  $\tilde{\xi}$ , respectively.

The key ideas of C-IGA are: (1) construct the isoparametric C-HiDeNN interpolation in the parametric domain, and (2) reproduce the geometric mapping  $\mathbf{x}^{Geo}(\xi)$  from CAD/IGA by mapping discrete nodal points through the geometric mapping  $\xi_k \xrightarrow{\mathbf{x}^{Geo}(\xi)} \mathbf{x}_k$ .

Eqs. (10) and (11) shown below are the isoparametric C-HiDeNN interpolation in the parametric domain, which are the same as those of C-HiDeNN, except that the parent coordinate  $\tilde{\xi}$  is changed to the parametric coordinates  $\xi$ .

$$\mathbf{x}^{C-IGA}(\xi) = \sum_{i \in A^e} \mathcal{N}_i(\xi) \sum_{j \in A_s^{\xi_i}} \mathcal{W}_{a,j}^{\xi_i}(\xi) \mathbf{x}_j = \sum_{k \in A_s^e} \tilde{N}_k(\xi) \mathbf{x}_k \quad (10)$$

and

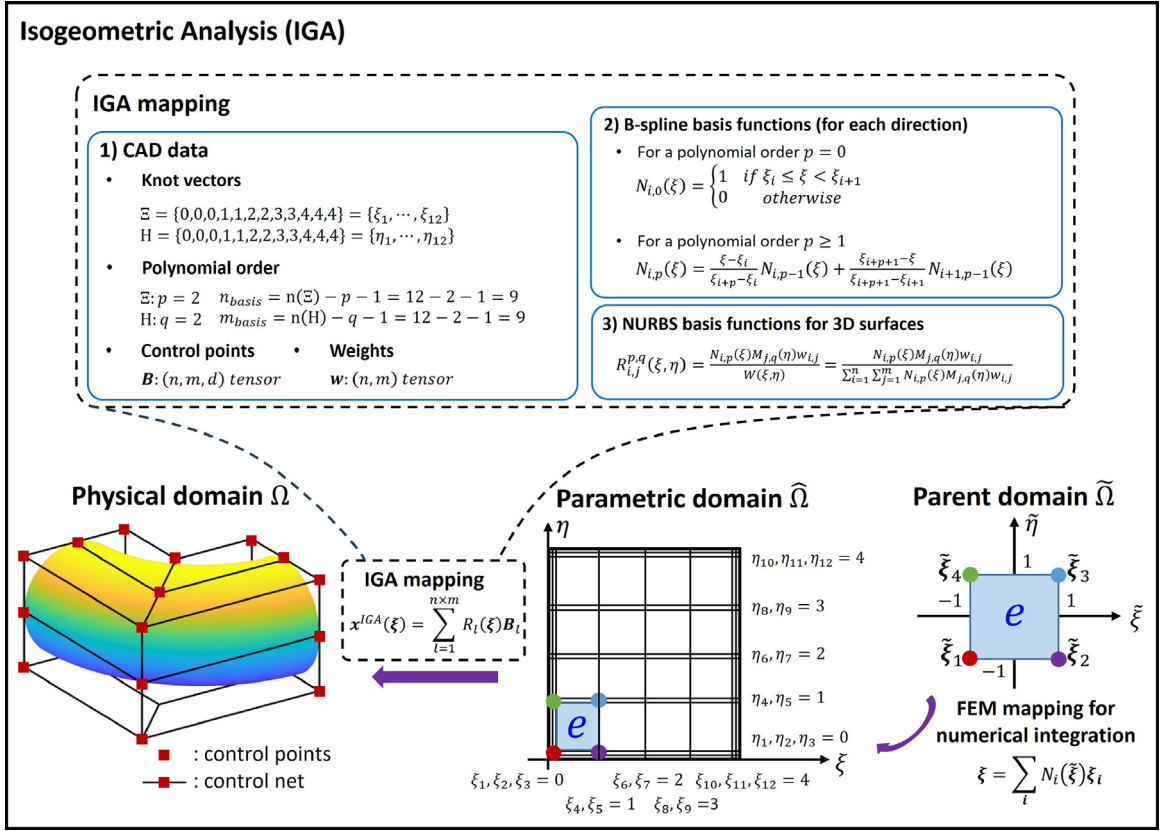
$$\mathbf{u}^{C-IGA}(\xi) = \sum_{i \in A^e} \mathcal{N}_i(\xi) \sum_{j \in A_s^{\xi_i}} \mathcal{W}_{a,j}^{\xi_i}(\xi) \mathbf{u}_j = \sum_{k \in A_s^e} \tilde{N}_k(\xi) \mathbf{u}_k. \quad (11)$$

We expect that the C-IGA coordinate mapping in Eq. (10) equals the geometric mapping  $\mathbf{x}^{Geo}(\xi)$  from CAD/IGA. To achieve this, there are two points to consider:

- The physical coordinates  $\mathbf{x}_k$  should come from the geometric mapping,  $\mathbf{x}_k = \mathbf{x}^{Geo}(\xi_k)$ , with given parametric coordinates  $\xi_k$  in the parametric domain. Substituting  $\mathbf{x}_k = \mathbf{x}^{Geo}(\xi_k)$  into Eq. (10), we can rewrite the coordinate mapping as:

$$\mathbf{x}^{C-IGA}(\xi) = \sum_{k \in A_s^e} \tilde{N}_k(\xi) \mathbf{x}^{Geo}(\xi_k). \quad (12)$$





**Fig. 4.** (a) An overview of Isogeometric Analysis (IGA) for a 3D toroidal surface. The black dashed box illustrates the procedure of building IGA mapping from the parametric domain to the physical domain. (1) CAD data used for constructing IGA mapping consist of knot vectors, polynomial orders for each parametric direction, control points, and weights for Non-Uniform Rational B-Splines (NURBS) basis functions. The number of basis functions is determined by  $n_{basis} = n(\Xi) - p - 1$  where  $n(\Xi)$  is the number of knots in  $\Xi$  and  $p$  is the polynomial order. The same rule applies to  $\eta$ -direction. For simplicity,  $n, m$  will be used to refer to the number of basis functions in  $\xi$ - and  $\eta$ - directions, respectively. The control points and weights are not shown in this figure. Readers may refer to the Appendix A.2. of [34] paper. (2) With knot vectors and polynomial orders, B-spline basis functions are constructed in each direction with the recursive relation. (3) NURBS basis functions are constructed as a weighted average of the B-spline functions. Note that for a 3D surface, the 2D parametric domain is mapped into the 3D physical domain because control points are defined in the 3D physical domain. The mapping from the parent domain to the parametric domain follows the standard FEM mapping. This mapping is required for the numerical integration. Because of the two mappings, two Jacobian determinants should be considered for the numerical integration.

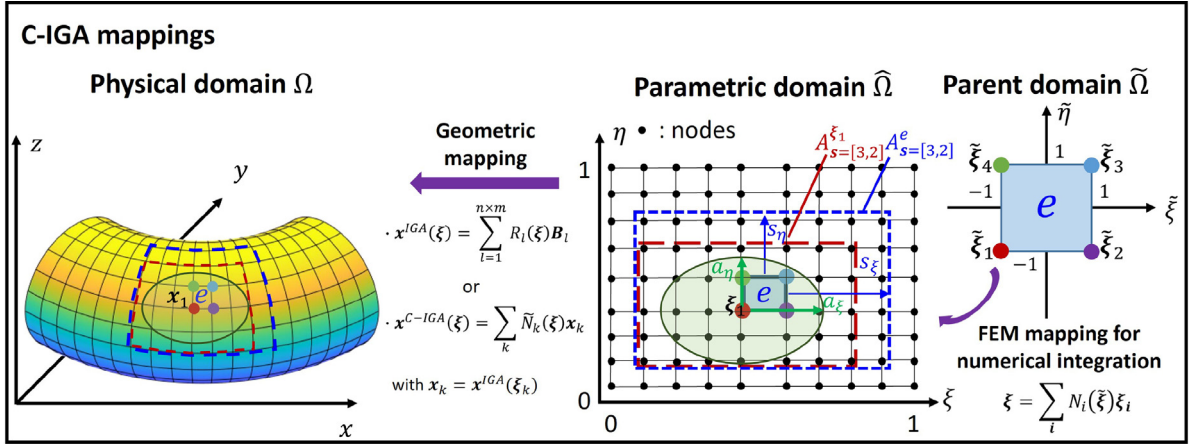
• The C-IGA shape functions  $\tilde{N}_k(\xi)$  should be able to reproduce the basis functions of the geometric mapping  $x^{Geo}(\xi)$ . In general, the IGA geometric mapping is written in the form:

$$x^{Geo}(\xi) = \sum_l R_l(\xi) B_l, \quad (13)$$

where  $R_l(\xi)$  are basis functions, and  $B_l$  coefficients. For example,  $R_l(\xi)$  are B-spline or NURBS basis functions, and  $B_l$  are control points in the IGA mapping  $x^{IGA}(\xi)$  as illustrated in Fig. 4. To reproduce  $x^{Geo}(\xi)$ , we start from the reproducing condition for  $R_l(\xi)$ , that is,

$$\sum_{k \in A_5^e} \tilde{N}_k(\xi) R_l(\xi_k) = R_l(\xi). \quad (14)$$

This property of C-IGA interpolation is realized by specifically designing the polynomial basis  $P(\xi)$  in Eq. (4). Since IGA can use either B-spline basis or NURBS basis function for  $x^{IGA}(\xi)$ , there are two choices for  $P(\xi)$ : (1) general polynomial basis functions, or (2) polynomials over the weighting function (weighted polynomials). Details of designing  $P(\xi)$  are discussed in Appendix D.



**Fig. 5.** C-IGA mappings between the physical domain, parametric domain, and the parent domain. The mapping from the parent domain to the parametric domain is the standard FEM mapping used for numerical integration like IGA. The parametric domain shows the knot space defined in IGA framework. Note that the parametric domain shown in Fig. 4 is normalized to  $[0, 1]^2$  for simplicity. Unlike IGA where the elements in the parametric domain are defined by the knot vectors, C-IGA can build any mesh in the  $[0, 1]^2$  space. Based on this mesh, we define all C-HiDeNN-related quantities (i.e.,  $s$ ,  $a$ , and  $p$ ) in the parametric domain. Then the geometric mapping from the parametric domain to the physical domain is placed to recover the exact geometry. In C-IGA, we can use either of the two geometric mappings (IGA or C-IGA), which are mathematically identical. As the nodal variables are also interpolated with C-IGA shape functions,  $\mathbf{u}(\xi) = \sum_k \tilde{N}_k(\xi) \mathbf{u}_k$ , C-IGA mapping is isoparametric.

Based on the above two points, we can prove that the C-IGA mapping is precisely the  $\mathbf{x}^{Geo}(\xi)$  and can reproduce the exact geometry of CAD models. The proof is given in the next subsection.

Similar to IGA, for the convenience of numerical integration, we need a second mapping from the parent domain to the parametric domain in one element, defined by

$$\xi(\tilde{\xi}) = \sum_{i \in A^e} N_i(\tilde{\xi}) \xi_i. \quad (15)$$

The deep neural network (DNN) form of C-IGA for a 1D problem is illustrated in Fig. 6. Like C-HiDeNN, the elemental neural network of C-IGA (Fig. 6(b)) also has multiple neurons in the third hidden layer that connect neighboring nodes in the nodal patch domain  $A_s^{\xi_i}$  with convolution patch functions  $\mathcal{W}_{a,s}^{\xi_i}$  as weights. The difference between C-HiDeNN and C-IGA in the neural network form is that C-IGA has two mappings (the geometric mapping and the FEM mapping) and the main interpolation takes place in the parametric domain.

With the above two mappings, the variational energy for an elastic system in one element  $e$  is given:

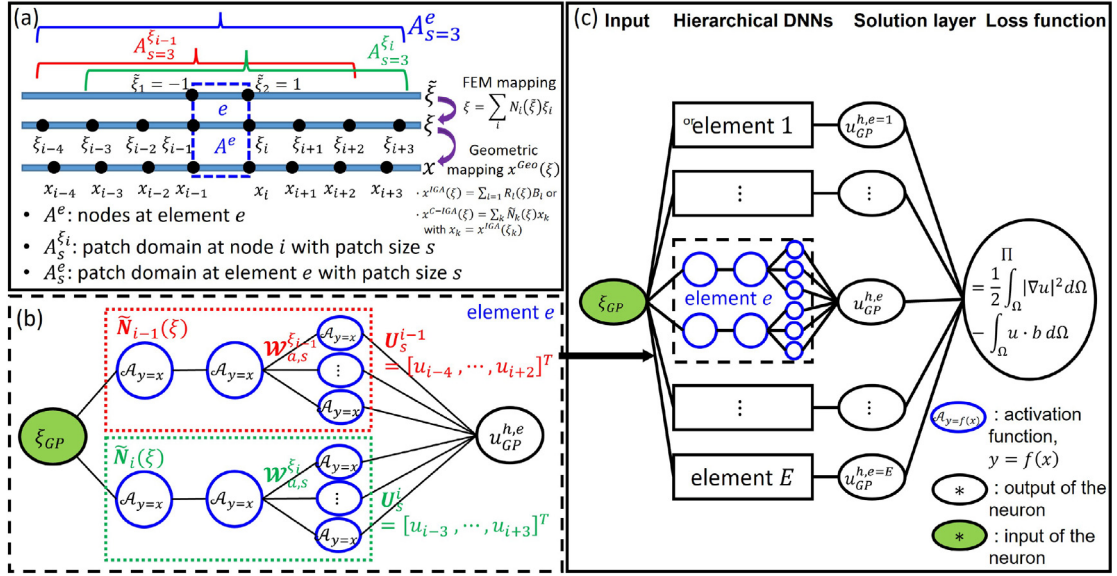
$$\begin{aligned} \Pi^e[\mathbf{u}^h(\mathbf{x})] &= \frac{1}{2} \int_{\Omega^e} \boldsymbol{\sigma} : \boldsymbol{\varepsilon} dV(\mathbf{x}) - \int_{\Omega^e} \mathbf{b}(\mathbf{x}) \mathbf{u}^h dV(\mathbf{x}) \\ &= \frac{1}{2} \int_{\tilde{\Omega}^e} (\mathbf{U}_s^e)^T (\tilde{\mathbf{B}}_s^e)^T \mathbf{D} \tilde{\mathbf{B}}_s^e \mathbf{U}_s^e \det(\mathbf{J}) \det(\tilde{\mathbf{J}}) dV(\tilde{\xi}) \\ &\quad - \int_{\tilde{\Omega}^e} \mathbf{b}(\tilde{\xi}) (\tilde{\mathbf{N}}_s^e)^T \mathbf{U}_s^e \det(\mathbf{J}) \det(\tilde{\mathbf{J}}) dV(\tilde{\xi}), \end{aligned} \quad (16)$$

where  $\boldsymbol{\sigma}$  and  $\boldsymbol{\varepsilon}$  are the stress and strain, respectively, and  $\mathbf{b}(\tilde{\xi})$  is the body force.  $\mathbf{U}_s^e$  is the unknown nodal displacement vector with a length of  $n(A_s^e) \times d$ .  $\tilde{\mathbf{B}}_s^e$  is the strain-displacement matrix whose size is  $[3, n(A_s^e) \times d]$  for a 2D element.  $\mathbf{D}$  is the modulus matrix. Unlike C-HiDeNN (see Appendix C), C-IGA has two Jacobian matrices corresponding to the two mappings.  $\mathbf{J}$  is for the first mapping  $\mathbf{x}(\xi) : \xi \xrightarrow{\mathbf{x}^{Geo}(\xi)} \mathbf{x}$  defined by

$$\mathbf{J}(\xi) = \frac{\partial \mathbf{x}}{\partial \xi}. \quad (17)$$

Because of the identity of the geometric mapping  $\mathbf{x}(\xi) = \mathbf{x}^{Geo}(\xi)$  and the C-IGA mapping in Eq. (12),  $\mathbf{J}(\xi)$  can be obtained from either of them. The second Jacobian matrix  $\tilde{\mathbf{J}}$  corresponds to the second mapping defined in





**Fig. 6.** Neural network structure of C-IGA for a 1D problem. (a) Nodal coordinates in the parent domain, parametric domain, and physical domain. (b) The elemental neural network of C-IGA and (c) the global neural network of C-IGA. The structure is similar to C-HiDeNN shown in Fig. 3 except for the geometric mapping illustrated in (a).

Eq. (15):

$$\tilde{J}(\tilde{\xi}) = \frac{\partial \tilde{\xi}}{\partial \xi}. \quad (18)$$

### 3.2. Proof of exact geometry

We assume that the geometric mapping consists of basis functions  $R_l(\xi)$  and coefficients  $B_l$  as shown in Eq. (13). By designing  $P(\xi)$ , e.g., taking  $P(\xi) = R_l(\xi)$ , the convolution patch functions in the nodal patch can reproduce each basis  $R_l(\xi)$  exactly, i.e.,

$$\sum_{j \in A_s^{i,i}} W_{a,j}^{i,i}(\xi) R_l(\xi_j) = R_l(\xi). \quad (19)$$

Combining with the partition of unity for  $\tilde{N}_i(\xi)$ , C-IGA interpolations have the same reproducing property Eq. (14). The proof is

$$\sum_k \tilde{N}_k(\xi) R_l(\xi_k) = \sum_i \tilde{N}_i(\xi) \sum_{j \in A_s^{i,i}} W_{a,j}^{i,i}(\xi) R_l(\xi_j) = R_l(\xi) \sum_i \tilde{N}_i(\xi) = R_l(\xi). \quad (20)$$

Thus we have the following proof of identity between the C-IGA mapping (Eq. (12)) and the geometric mapping  $x^{Geo}(\xi)$ :

$$\begin{aligned} x^{C-IGA}(\xi) &= \sum_k \tilde{N}_k(\xi) x^{Geo}(\xi_k) \\ &= \sum_k \tilde{N}_k(\xi) \sum_l R_l(\xi_k) B_l \\ &= \sum_l B_l \sum_k \tilde{N}_k(\xi) R_l(\xi_k) \\ &= \sum_l B_l R_l(\xi) = x^{Geo}(\xi). \end{aligned} \quad (21)$$

Remark that we mainly talk about B-spline and NURBS basis functions in this paper. The proof is still valid for T-spline if the convolution patch functions in the nodal patch can reproduce T-spline basis functions. Complex CAD geometries with extraordinary nodes and arbitrary genus number [44,45] can also be handled by C-HiDeNN/C-IGA interpolation (Eqs. (3) or (11)). The C-HiDeNN/C-IGA formulation uses both finite element shape functions and meshfree type basis functions, which are not influenced by extraordinary nodes. More details on the implementation will be investigated in our future work.

### 3.3. Degeneration of C-IGA to IGA

Although C-IGA can reproduce the IGA mapping, the present C-IGA cannot degenerate to IGA directly, as C-IGA has less DoFs than IGA. However, if we take the modified anchors  $\{t_i\}$  of IGA as the supporting nodes instead of non-repeating knots  $\{\xi_i\}$  to construct C-IGA interpolation, and use IGA basis functions as the convolution patch functions, the new C-IGA will degenerate to IGA (they are equal in fact).

$$\mathbf{u}^{C-IGA}(\xi) = \sum_i N_i(\xi; \mathbf{t}_i^*) \sum_{j \in A_s^{t_i}} N_{j,p}(\xi) \mathbf{d}_j, \quad (22)$$

where  $\mathbf{t}_i^*$  represents the anchors associated with FEM shape function  $N_i$ , such as  $\mathbf{t}_i^* = [t_{i-1}, t_i, t_{i+1}]$  in 1D linear case. The nodal patch  $A_s^{t_i}$  is the set of nodes (anchors) in a patch domain of node (anchor)  $t_i$ .  $N_{j,p}(\xi)$  is the  $p$ th order basis function for node (anchor)  $t_j$ , and  $\mathbf{d}_j$  is the corresponding nodal coefficient. Note that  $\mathbf{d}_j$  is not a nodal value of the solution like  $\mathbf{u}_j$ , for there is no Kronecker delta property in the new formulation. Details of the derivation can be found in [Appendix G](#).

### 3.4. $r$ -adaptivity

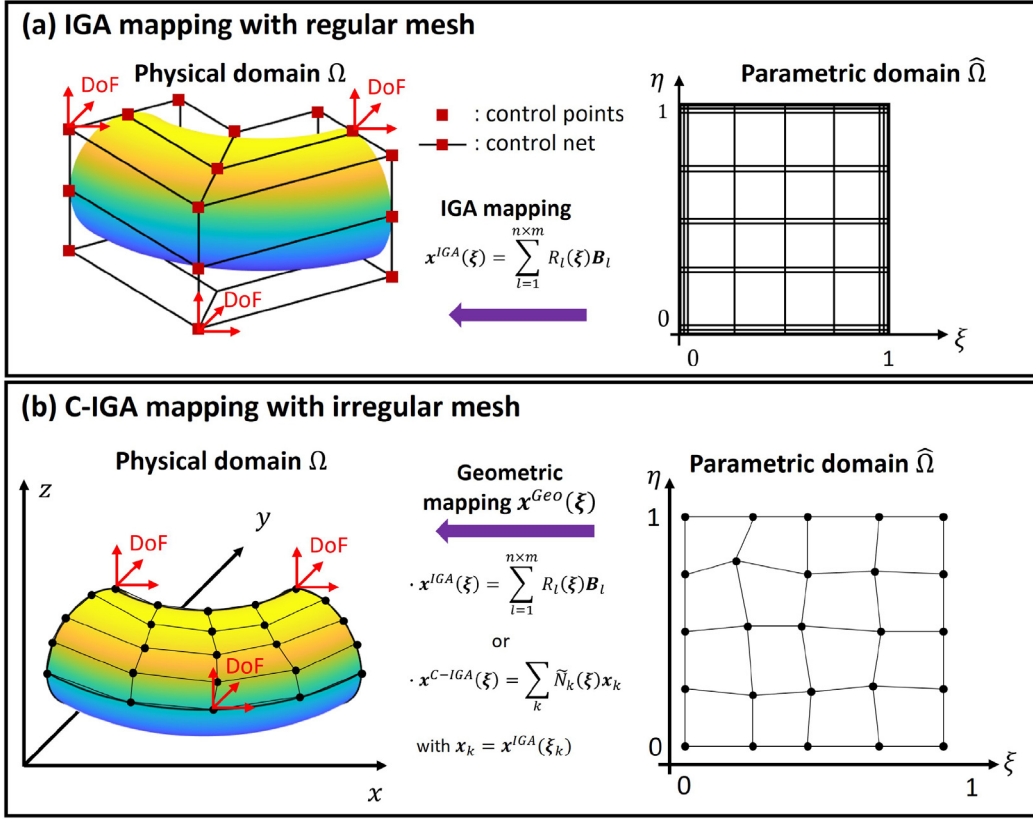
In IGA, the multivariate B-spline and NURBS basis functions are constructed by tensor product, which requires a regular mesh in the parametric domain. The T-spline [35] enables local mesh refinement but it is still limited by regular meshes; IGA cannot handle arbitrary irregular meshes. On the other hand, both C-IGA and C-HiDeNN allow an irregular mesh because they can generate any type of mesh in the parametric (C-IGA) or parent (C-HiDeNN) domain. The ability to handle an irregular mesh is one of the key advantages of C-IGA over IGA, as illustrated in [Fig. 7](#). As discussed in Section 2.1,  $r$ -adaptivity is achieved by minimizing the variational energy with respect to the nodal parametric coordinates, that is,

$$\text{Train } \chi = \arg \min_{\xi_i \in \hat{\Omega} \setminus \partial \hat{\Omega}} \Pi(\mathbf{u}^{C-IGA}(\mathbf{x}; \mathbf{U}, \chi, a, s, p)). \quad (23)$$

Note that  $\chi = [\xi_1, \xi_2, \dots, \xi_{np}]$  denotes a vector of all nodal parametric coordinates with  $np$  number of global nodes. Nodal solution  $\mathbf{U}$  is obtained after given  $\chi, a, s, p$ , which can be interpreted that  $\mathbf{U}$  is the function of  $\chi$  in the optimization process. We limit the movement of boundary nodes to keep the exact geometry unchanged. The difference from the  $r$ -adaptivity of C-HiDeNN is to optimize nodal parametric coordinates instead of nodal physical coordinates. The physical coordinates  $\mathbf{x}_k$  will also follow the migration of parametric coordinates  $\xi_k$  according to the mapping  $\mathbf{x}_k = \mathbf{x}^{Geo}(\xi_k)$ .

There are several points to be considered here.

- We can leverage automatic gradient functionalities for gradient-based optimization in many popular scientific computation packages such as PyTorch [46] and JAX [47].
- The convexity of optimization is not guaranteed, meaning that we might not be able to find the optimal nodal coordinates even after prolonged iteration. Furthermore, it may result in an unphysical mesh having a negative Jacobian determinant. This issue has been discussed in [28]. A careful selection of learning rate and early stopping criteria is required.
- The global mesh update may suffer from an immense cost of optimization. A more efficient strategy might be to update locally. In-depth studies on this local refinement are needed.



**Fig. 7.** Comparison between IGA and C-IGA mappings. (a) IGA mapping is built on a regular mesh because the B-splines and NURBS basis functions are constructed by tensor product. In IGA, control points have DoFs. (b) C-IGA mapping can be built on an irregular mesh, enabling the automatic  $r$ -adaptivity. Like conventional FEM or C-HiDeNN, the DoFs of C-IGA are attached to nodes.

### 3.5. $a$ -adaptivity

In addition to  $r$ -adaptivity, both C-HiDeNN and C-IGA can adjust the dilation parameter  $a$  to get better accuracy. The so-called  $a$ -adaptivity refers to the dilation parameter  $a$  varying from one nodal patch to another nodal patch. As discussed in Section 2.2, the parameter  $a$  controls the characteristic length scale of the radial function, and further influences the shape of the convolution patch function. For a complex solution, the optimal characteristic length should vary with spatial positions. We seek the optimal spatial distribution of the dilation parameter  $a$  by minimizing the variational energy:

$$\text{Train } \mathbf{a} = \arg \min_{\mathbf{a}^{\xi_i} \in \mathbb{R}^+} \Pi(u^{C-IGA}(\mathbf{x}; \mathbf{U}, \boldsymbol{\chi}, \mathbf{a}, s, p)). \quad (24)$$

The vector  $\mathbf{a}$  represents a patch-wise dilation parameter, denoted by  $\mathbf{a} = [a^{\xi_1}, a^{\xi_2}, \dots, a^{\xi_{np}}]$  with  $a^{\xi_i}$  representing the dilation parameter of convolution patch functions  $\mathcal{W}_{\mathbf{a}^{\xi_i}, s}^{\xi_i}$  in the nodal patch  $A_s^{\xi_i}$ . Unlike the  $r$ -adaptivity where the global nodal coordinates are updated,  $a$ -adaptivity can be performed patch-wise, rendering a wide choice of optimization strategies.

Unlike  $r$ -adaptivity which might result in unphysical meshes,  $a$ -adaptivity is free from the same issue because the adjustment of  $a$  does not influence the mesh. The only constraint for  $a$  is to be a positive real number, as given in Table 3. Additionally, although we adopt an isoparametric mapping, the mapping is precisely the geometrical mapping  $\mathbf{x}^{Geo}(\xi)$  according to the equivalence Eq. (21). That is, the mapping is independent of the dilation parameter  $a$ . This leads to a simpler derivation of analytical derivatives with respect to  $a$  (derivation is given in Appendix E) in  $a$ -adaptivity compared to  $r$ -adaptivity.

**Table 4**

Comparison between FEM, IGA, C-HiDeNN, and C-IGA.

	FEM	IGA	C-HiDeNN	C-IGA
DoFs	Attached to nodes	Attached to control points	Attached to nodes	Attached to nodes
Interpolation	Interpolates nodal points	<i>Does not</i> interpolate control points	Interpolates nodal points	Interpolates nodal points
Reproducing property	Exact order	Exact geometry	Exact order	Exact geometry
Adaptivity	r-, h-, p- adaptivity	h-, k- refinement. Need to update control points.	r-, h-, p-, s-, a- adaptivity. Versatile	r-, h-, p-, s-, a- adaptivity. Versatile

### 3.6. $s$ - $p$ -adaptivity

According to Table 3,  $s \geq p/2$ . However, our preliminary numerical tests suggest  $s \geq p$  for good accuracy, efficiency, and numerical stability. Small  $s$  compared to  $p$  might cause stability issues when we impose the Kronecker delta property on the convolution patch functions. A larger  $p$  leads to a higher order convergence rate while a larger  $s$  leads to a more accurate solution and a higher computational cost. Spatially varying  $p$  and  $s$  can be considered similarly to  $a$ -adaptivity, called  $s$ - $p$ -adaptivity. That is, like  $a$ -adaptivity, the  $s$ - $p$ -adaptivity can be achieved nodal patch-wise. For efficiency, we can adopt a hybrid strategy, i.e., use large  $p$  or  $s$  for important regions and small ones for the other regions. This idea is verified in the following 1D numerical examples. More studies will be investigated in our future work.

### 3.7. Discussions

C-IGA is an extension of C-HiDeNN that realizes exact geometry like IGA. We compare different aspects of FEM, IGA, C-HiDeNN, and C-IGA in Table 4.

First, unlike IGA where the control points have DoFs (as known as control variables according to the original paper [34]), all FEM, C-HiDeNN, and C-IGA have DoFs attached to nodal points. This is the reason why C-IGA can handle an irregular mesh, as discussed in the Section 3.4 and Fig. 7. The three methods (FEM, C-HiDeNN, and C-IGA) are also able to exactly interpolate nodal points because the convolution patch functions have a Kronecker delta property, whereas IGA cannot interpolate control points except for repeating knots. In terms of the reproducing property, FEM and C-HiDeNN can reproduce polynomial orders up to a given  $p$  because they use polynomial basis functions. On the other hand, both IGA and C-IGA use NURBS basis which are rational functions (neglecting the case when the NURBS weights are all equal to 1, i.e., B-splines with polynomials). With NURBS basis functions, they can reproduce exact geometry, but not the polynomials like FEM or C-HiDeNN. Additionally, C-HiDeNN uses a parent domain  $[-1, 1]^d$  (with some extended neighborhood depending on the patch size  $s$ ) like FEM which focuses on one element (see Fig. 2). In contrast, C-IGA uses the parametric domain which is defined globally, as shown in Fig. 5.

In terms of adaptivity, C-HiDeNN and C-IGA have versatile  $r$ -,  $h$ -,  $p$ -,  $s$ -,  $a$ - adaptivities in contrast with FEM or IGA. Detailed comparison of adaptivities between IGA and C-IGA can be found in Table 5. Some key points from the table are listed below.

- C-IGA follows the same framework of C-HiDeNN or FEM. There is no control net in the interpolations, although the geometric mapping may come from the IGA mapping with a coarse control net. B-spline, NURBS, and T-spline in IGA are all developed based on a regular mesh, since their basis functions are obtained by tensor product. Compared with IGA, C-IGA allows an irregular mesh and nodal adaptivity in all directions except for the boundary nodes. In short, the  $r$ -adaptivity of C-IGA can improve accuracy without additional DoFs.

- The  $r$ -adaptivity of C-HiDeNN is realized in the physical domain. In contrast, C-IGA introduces an additional intermediate domain, parametric domain, to realize the exact geometry. The  $r$ -adaptivity is realized in such a parametric domain, and the physical coordinates move accordingly.

- C-IGA and IGA both have the parameter  $p$  to denote the order. The IGA can elevate the order  $p$  by  $p$ -refinement or  $k$ -refinement. The former adopts the order elevation element by element after knot insertion, which is analogous

**Table 5**

Comparison of adaptivity between IGA and C-IGA.

Parameters	IGA	C-IGA
Mesh (r)	————	• Automatic r-adaptivity due to HiDeNN
Mesh (h)	• Control net • Allows nonuniform but regular mesh *	• No control net • Allows irregular mesh.
Order (p)	• $p$ -refinement: knot insertion followed by order elevation • $k$ -refinement: order elevation followed by knot insertion • Both cases need extra DoFs and re-computation of control points and knot vectors.	• Elevates $p$ without increasing DoFs.
Patch size (s)	————	• Controls solution accuracy through modifying connectivity and bandwidth of the stiffness matrix
Dilation (a)	————	• Controls characteristic length scales. • Does not change the mesh.

\* B-spline, NURBS, and T-spline are all developed based on the regular mesh, because their basis functions are obtained by tensor product.

to  $p$ -refinement in FEM. The latter adopts the order elevation followed by the knot insertion. The  $k$ -refinement significantly lowers the DoFs increment compared with the  $p$ -refinement, but still requires some additional DoFs and basis functions. In contrast, the order elevation of C-IGA occurs in the convolution patch function, specifically with  $P(\xi)$  in Eq. (4). Hence, we can elevate the order  $p$  without changing the mesh and DoFs.

• C-IGA has two additional parameters compared with IGA: the patch size  $s$  and the dilation parameter  $a$ . Patch size  $s$  denotes the size of the patch domain involved in building shape functions. It controls the nodal connectivity and affects the bandwidth of the global stiffness matrix. As discussed previously, the dilation parameter  $a$  controls the characteristic length scale of the radial function. C-IGA allows the  $a$ -adaptivity to seek an optimal distribution of patch-wise  $a$  for improved solution accuracy without changing the mesh and DoFs.

## 4. Continuity and convergence rate

### 4.1. High-order continuity

In this Subsection, we analyze the continuity of shape functions  $\tilde{N}_k(\xi)$  of C-HiDeNN/C-IGA interpolations. The C-HiDeNN/C-IGA convolution shape function  $\tilde{N}_k(\xi)$  is defined by

$$\tilde{N}_k(\xi) = \sum_{i|k \in A_s^{\xi_i}} \mathcal{N}_i(\xi) \mathcal{W}_{a,k}^{\xi_i}. \quad (25)$$

According to the chain rule, the  $N$ th order derivative of  $\tilde{N}_k(\xi)$  is

$$D^N \tilde{N}_k(\xi) = \sum_{i|k \in A_s^{\xi_i}} \left[ (D^N \mathcal{N}_i(\xi)) \mathcal{W}_{a,k}^{\xi_i} + N (D^{N-1} \mathcal{N}_i(\xi)) (D \mathcal{W}_{a,k}^{\xi_i}) + \cdots + \mathcal{N}_i(\xi) (D^N \mathcal{W}_{a,k}^{\xi_i}) \right], \quad (26)$$

where  $D^N$  denotes the  $N$ th order derivative. It clearly shows that the continuity of  $\tilde{N}_k(\xi)$  depends on the continuities of  $\mathcal{N}_i(\xi)$  and  $\mathcal{W}_{a,k}^{\xi_i}$ . Assuming  $\mathcal{N}_i(\xi)$  is  $C^n$  continuous and  $\mathcal{W}_{a,k}^{\xi_i}$  is  $C^m$  continuous,  $D^N \tilde{N}_k(\xi)$  is continuous up to  $N = \min(n, m)$ .

There exists a special case for 1D interpolation:  $\tilde{N}_k(\xi)$  has  $C^N$  continuity with  $N = \min(n + 1, m)$ , if the  $D^{n+1} \mathcal{N}_i(\xi)$  is discontinuous across nodes, e.g., FEM shape functions. That is due to the Kronecker delta property of  $\mathcal{W}_{a,k}^{\xi_i}$  and partition of unity property of  $\mathcal{N}_i(\xi)$ . The first term in Eq. (26) at the left and right limits of  $\xi_j$  is

$$\sum_{i|k \in A_s^{\xi_j^+}} \left( \frac{d^N}{d\xi^N} \mathcal{N}_i(\xi_j^+) \right) \mathcal{W}_{a,k}^{\xi_j^+} = \sum_{i|k \in A_s^{\xi_j^-}} \left( \frac{d^N}{d\xi^N} \mathcal{N}_i(\xi_j^-) \right) \delta_{jk} = \delta_{jk} \frac{d^N}{d\xi^N} \sum_{i|k \in A_s^{\xi_j}} \mathcal{N}_i(\xi_j) = 0, \quad (27)$$

$$\sum_{i|k \in A_s^{\xi_i}} \left( \frac{d^N}{d\xi^N} \mathcal{N}_i(\xi_j^-) \right) \mathcal{W}_{a,k}^{\xi_i}(\xi_j^-) = \sum_{i|k \in A_s^{\xi_i}} \left( \frac{d^N}{d\xi^N} \mathcal{N}_i(\xi_j^-) \right) \delta_{jk} = \delta_{jk} \frac{d^N}{d\xi^N} \sum_{i|k \in A_s^{\xi_i}} \mathcal{N}_i(\xi_j^-) = 0. \quad (28)$$

The first term vanishes at nodes, so the continuity of (26) depends on the second term and the final term.

If adopting the radial basis functions (Eq. (4)) as convolution patch function  $\mathcal{W}_{a,j}^{\xi_i}$ , the continuity of  $\mathcal{W}_{a,j}^{\xi_i}$  depends on two parts: the radial function  $\Psi_a(\xi)$  and reproduced basis functions  $\mathbf{P}(\xi)$ . Generally,  $\mathbf{P}(\xi)$  is  $C^\infty$ , such as polynomials.  $\Psi_a(\xi)$  can also be  $C^\infty$  when taking the Gaussian radial functions Eq. (7). As a result, the overall continuity is depending on the selection of  $\mathcal{N}_i(\xi)$  in general. Careful design of  $\mathcal{N}_i(\xi)$  and  $\mathcal{W}_{a,j}^{\xi_i}$  can lead to arbitrary high order  $C^N$  continuity of C-HiDeNN/C-IGA interpolants, which is difficult for traditional FEM. Note that we only require partition of unity and compact support for  $\mathcal{N}_i(\xi)$  [11]. Thus B-spline basis functions and meshfree basis functions satisfying partition of unity can be good selections for  $\mathcal{N}_i(\xi)$ .

## 4.2. Theoretical convergence rate

If taking  $P(\xi)$  as  $p$ th order polynomials in the reproducing conditions, C-HiDeNN/C-IGA interpolants satisfy the reproducing property as well, e.g., the following reproducing conditions for 1D  $p$ th order polynomials

$$\sum_{k \in A_s^e} \tilde{N}_k(\xi)(\xi_k)^q = \xi^q, q = 0, 1, 2, \dots, p. \quad (29)$$

Our previous study [11] gives the below theoretical results for C-HiDeNN interpolants on the upper bound of the error of C-HiDeNN for elliptic equations

$$\|u^{\text{C-HiDeNN}}(\mathbf{x}) - u^{\text{Ext}}(\mathbf{x})\|_{L^2} \leq \tilde{C}_0 h^{p+1} \|u^{\text{Ext}}\|_{H^{p+1}} \quad (30)$$

and

$$\|u^{\text{C-HiDeNN}}(\mathbf{x}) - u^{\text{Ext}}(\mathbf{x})\|_{H^1} \leq \tilde{C}_1 h^p \|u^{\text{Ext}}\|_{H^{p+1}} \quad (31)$$

where  $\tilde{C}_0, \tilde{C}_1$  are constants independent of  $h$ ,  $h$  is the mesh size,  $p$  is the reproduced polynomial order, and  $u^{\text{Ext}}$  denotes the exact solution. It indicates that C-HiDeNN/C-IGA achieves the  $(p+1)$ th order convergence rate in terms of  $L^2$ -norm and the  $p$ th order convergence rate in terms of  $H^1$ -norm when the interpolation functions satisfy the  $p$ th order reproducing conditions in Eq. (29). For detailed proof and numerical verifications, refer to the Ref. [11]. In this work, we also verify this conclusion for C-IGA in numerical examples, and further numerically study convergence rates with adaptivities of various parameters.

## 5. Numerical examples

### 5.1. 1D example and implementations

We first use a 1D example to illustrate C-IGA and study the convergence rate as well as all kinds of parameter adaptivity. For illustrative purposes, we consider a rod fixed at both ends under a body force  $b(x)$  as shown in Fig. 9. The strong form reads

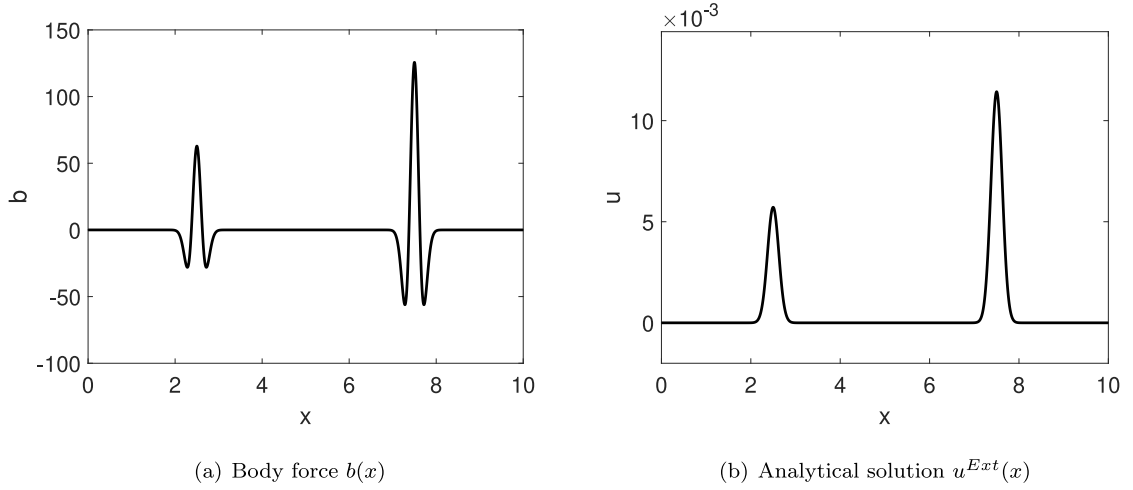
$$\frac{d}{dx} \left( AE \frac{du}{dx} \right) + b(x) = 0, x \in [0, L] \quad (32)$$

with Dirichlet boundary conditions

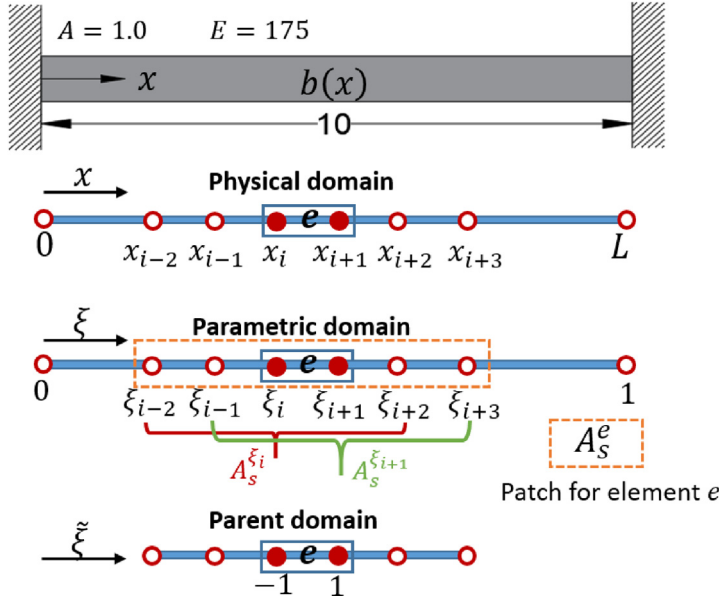
$$u(0) = 0, u(L) = 0. \quad (33)$$

Here,  $u(x)$  is the displacement field,  $E$  the stiffness of the rod defined as  $E = 175$ ,  $A$  the section area set to be  $A = 1$ , and  $L$  the length of the rod set to be  $L = 10$ . All parameters are unitless. The body force  $b(x)$  is given by

$$b(x) = -\frac{400\pi^2(x-2.5)^2 - 20\pi}{e^{10\pi(x-2.5)^2}} - \frac{800\pi^2(x-7.5)^2 - 40\pi}{e^{10\pi(x-7.5)^2}}. \quad (34)$$



**Fig. 8.** The body force  $b(x)$  and analytical solution  $u^{Ext}(x)$  for 1D numerical model.



**Fig. 9.** 1D numerical model and illustrations for C-IGA interpolants.

Substituting the body force into Eq. (32), one obtain the analytical solution

$$u^{Ext}(x) = \frac{1}{AE}(e^{-10\pi(x-2.5)^2} - e^{-62.5\pi}) + \frac{2}{AE}(e^{-10\pi(x-7.5)^2} - e^{-560.25\pi}) - \frac{e^{-562.5\pi} - e^{-62.5\pi} + 2e^{-62.5\pi} - 2e^{-560.25\pi}}{10AE}x. \quad (35)$$

Fig. 8 plots the body force and analytical solution.

Fig. 9 illustrates three domains of C-IGA: physical, parametric, and parent domain. 1D C-IGA interpolation is defined in the parametric domain:

$$u^h(\xi) = \sum_{i=1}^{np} \mathcal{N}_i(\xi) \sum_{j \in A_s^{\xi_i}} \mathcal{W}_{a,j}^{\xi_i}(\xi) u_j = \sum_{k=1}^{np} \tilde{N}_k(\xi) u_k \quad (36)$$



where  $np$  is the number of nodes. Here, we adopt the cubic spline function in Eq. (6) as the radial function,  $p$ th order polynomials  $[1, \xi, \xi^2, \dots, \xi^p]$  as  $\mathbf{P}(\xi)$  in the convolution patch function  $\mathcal{W}_{a,j}^{\xi_i}$  (Eq. (4)).

For IGA, the mapping is defined by a knot vector  $\Xi = [0, 0, 1, 1]$  with two ends as control points  $B_1 = 0, B_2 = L$ . This gives the simplest geometric mapping, an affine, i.e.,

$$x^{IGA}(\xi) = \sum_{l=1}^2 R_l(\xi) B_l = L\xi \quad (37)$$

with

$$R_1(\xi) = 1 - \xi, R_2(\xi) = \xi. \quad (38)$$

We adopt the same geometric mapping  $x^{Geo}(\xi) = L\xi$  as the first mapping between the parametric and the physical domains. Note that it is also an isoparametric mapping

$$x^{C-IGA}(\xi) = \sum_k \tilde{N}_k(\xi) x_k = \sum_k \tilde{N}_k(\xi) \cdot L\xi_k \quad (39)$$

by taking  $x_k = x^{Geo}(\xi_k) = L\xi_k$  and  $\mathbf{P}(\xi) = [1, \xi, \xi^2, \dots, \xi^p]$  in the reproducing condition. Since the C-HiDeNN interpolation reproduces  $p$ th order polynomials including the linear basis, we have the following equivalence

$$x^{C-IGA}(\xi) = \sum_k \tilde{N}_k(\xi) x_k = \sum_k \tilde{N}_k(\xi) \cdot L\xi_k = L \cdot \sum_k \tilde{N}_k(\xi) \xi_k = L\xi. \quad (40)$$

The second mapping is also an affine in the element  $e$ , i.e.,

$$\xi(\tilde{\xi}) = \frac{1 - \tilde{\xi}}{2} \xi_i + \frac{1 + \tilde{\xi}}{2} \xi_{i+1}, \tilde{\xi} \in [-1, 1]. \quad (41)$$

The variational energy of Eq. (32) in one element  $e$  is

$$\begin{aligned} \Pi^e[u^h(x; \xi_k)] &= \frac{1}{2} \int_e AE \left( \frac{d}{dx} u^h \right)^2 dx - \int_e b(x) u^h dx \\ &= \frac{1}{2} \int_{-1}^1 (\mathbf{U}^e)^T (\tilde{\mathbf{B}}^e)^T AE \tilde{\mathbf{B}}^e \mathbf{U}^e \left| \frac{dx}{d\tilde{\xi}} \right| \left| \frac{d\tilde{\xi}}{d\xi} \right| d\tilde{\xi} - \int_{-1}^1 b(\tilde{\xi}) \tilde{\mathbf{N}}^e \mathbf{U}^e \left| \frac{dx}{d\tilde{\xi}} \right| \left| \frac{d\tilde{\xi}}{d\xi} \right| d\tilde{\xi} \end{aligned} \quad (42)$$

with 1D interpolants  $u^h$  defined in Eq. (36). According to above simple mappings Eqs. (40) and (41), we have

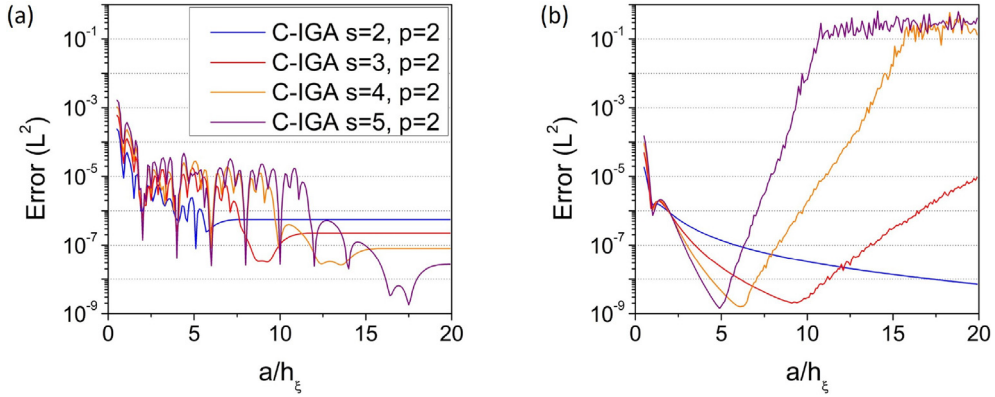
$$\left| \frac{dx}{d\tilde{\xi}} \right| = L, \left| \frac{d\tilde{\xi}}{d\xi} \right| = \frac{\xi_{i+1} - \xi_i}{2}. \quad (43)$$

### 5.1.1. Convergence study

We start with a uniform mesh to study convergence rates. The element sizes in the physical domain and the parametric domain are denoted by  $h = \frac{L}{np-1}$  and  $h_\xi = \frac{1}{np-1}$ , respectively, where  $np-1$  is the number of elements. The accuracy of numerical solution  $u^h$  is measured by the relative  $L^2$ -norm error defined by

$$error = \frac{\|u^h - u^{Ext}\|_{L^2}}{\|u^{Ext}\|_{L^2}}. \quad (44)$$

Before the convergence study, we investigate the accuracy of C-IGA results with varying dilation parameter  $a$  and the patch size  $s = 2, 3, 4, 5$  for cubic spline and Gaussian radial functions as shown in Fig. 10. For the cubic spline radial function in Fig. 10(a), the error oscillates for low  $a$ , but converges to a specific value when  $a$  is large enough. The oscillation observed in small  $a$  is attributed to the piecewise form of the cubic spline function with a compact support size 2. The lowest error occurs at some specific values of  $a$ , but it requires optimization of  $a$  which we call  $a$ -adaptivity. On the other hand, the error converges for large  $a$  with moderate accuracy. Note that the value  $a$  at which the error converges increases as the patch size  $s$  enlarges. When  $s = 2$ , the error converges for  $a/h_\xi > 7$  while for  $s = 5$ , the error converges for  $a/h_\xi > 19$ . Therefore, when we use the cubic spline radial function, it is recommended to use a large  $a$  if we want to skip the  $a$ -adaptivity.



**Fig. 10.** The  $L^2$ -norm error comparison for C-IGA with varying  $s$  and fixed  $p = 2$ . (a) Cubic spline radial function and (b) Gaussian radial function in Eq. (6). 640 elements are used.  $h_\xi$  is the element size in the parametric domain.

Fig. 10(b) shows the case when the Gaussian radial basis is used. Unlike the cubic spline, the error curves diverge for large  $a$  and there is a global optimum for  $a$  that yields considerably accurate results. In general, for those optimal  $a$ , the Gaussian function is more accurate than the cubic spline. Therefore, when the Gaussian radial function is used, we need to optimize  $a$ .

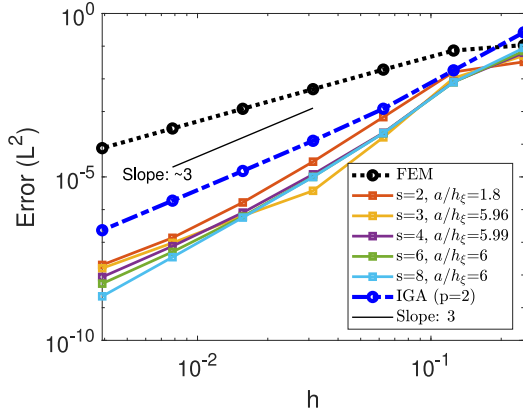
Based on the above discussion, we choose the optimal dilation parameter  $a$  for different patch sizes to study their  $L^2$ -norm error convergence rates and compare the C-IGA solutions with linear FEM and IGA results. Here, the cubic spline radial function is used. The results are shown in Fig. 11. As expected, the  $L^2$ -norm error of linear FEM solution has a theoretical convergence rate of 2, and the  $p$ th order IGA solution reaches the  $(p + 1)$ th order of convergence. For  $p = 2, 3$ , the convergence rates of C-IGA are asymptotically  $p + 1$  but the solution accuracy is better than linear FEM and IGA. For  $p = 4$ , C-IGA ( $s = 4, a/h_\xi = 9.9$ ) almost coincides with IGA results, and the other solutions also reach the fifth-order convergence rate. We have similar conclusions on  $H^1$ -norm error convergence rates given in Appendix F. These observations verify our theoretical analysis. When optimal dilation parameters are used, the patch size  $s$  has less influence on the convergence and accuracy. However, large  $s$  broadens the bandwidth of the stiffness matrix, which leads to more computational costs both in time and memory. Hence, small  $s$ , e.g.,  $s = p$ , might be a more effective option. Furthermore, we would like to point out that both FEM and IGA need additional DoFs when the order  $p$  is elevated, although such DoFs increases for IGA are relatively small. In contrast, the C-IGA can achieve high order convergence rates by simply adjusting the polynomial order in the convolution patch function without increasing the number of nodes and DoFs.

We further perform a numerical study on the condition number of the stiffness matrix (see Appendix F for details). The numerical results demonstrate that the condition number of C-IGA for different solution orders ( $p = 2, 3, 4$ ) is similar to that of linear FEM, slightly larger than IGA for orders ( $p = 2, 3, 4$ ), and significantly lower than the quadratic FEM. For fine mesh, C-IGA gives the best accuracy at a given condition number compared to IGA and FEM.

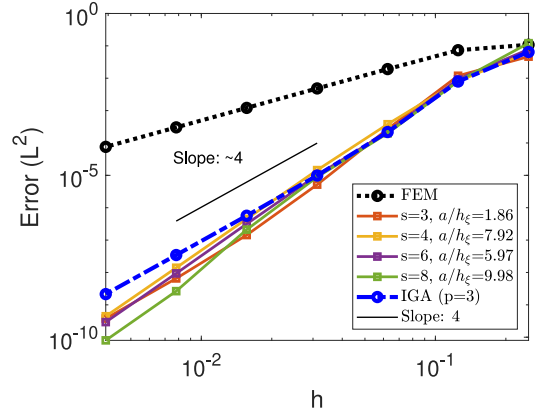
### 5.1.2. $r$ -adaptivity

For the  $r$ -adaptivity, C-IGA can adjust nodal coordinates in the parametric domain then the physical mesh is obtained by the geometric mapping  $x^{Geo}(\xi)$ . To avoid the singularity due to extremely small element size, we limit the element size to be bigger than  $h_\xi/5$  where  $h_\xi$  is the original element size in the parametric domain. There are two kinds of optimization algorithms: gradient-based methods and global optimization methods. Here, since the problem size is moderate and we aim for the global optimum, we adopt one of the global optimization methods, particle swarm optimization (PSO) [48], to optimize nodal positions except for the boundary nodes. Our experience showed that PSO achieves more accurate results than the Adam optimizer [49] since PSO is designed to search for the global minima from many different random starting points.

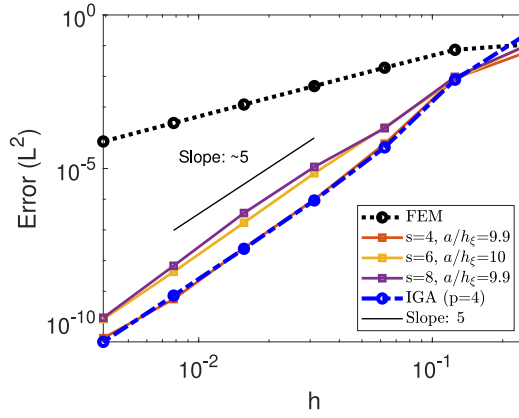
Fig. 12 illustrates the final optimized coarse mesh of C-IGA ( $s = 2, p = 2, a/h_\xi = 1.8$ ) for  $h = L/40$  and  $h = L/80$ . As seen in the figure, the nodes tend to gather at peaks and feet of two Gaussian humps, since the solution



(a)  $p = 2$  ( $L^2$  error)

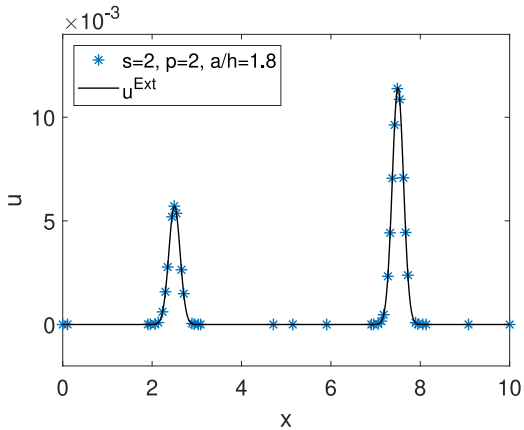


(b)  $p = 3$  ( $L^2$  error)

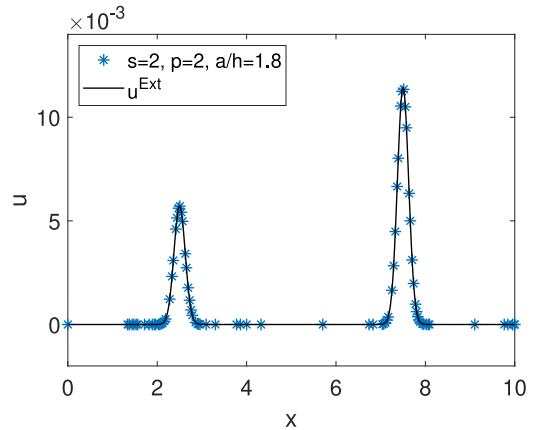


(c)  $p = 4$  ( $L^2$  error)

**Fig. 11.** The  $L^2$ -norm error convergence rates against the mesh refinement for different methods and different sets of parameters for C-IGA with cubic spline radial function: (a)  $p = 2$  ( $L^2$ -norm error); (b)  $p = 3$  ( $L^2$ -norm error); (c)  $p = 4$  ( $L^2$ -norm error).

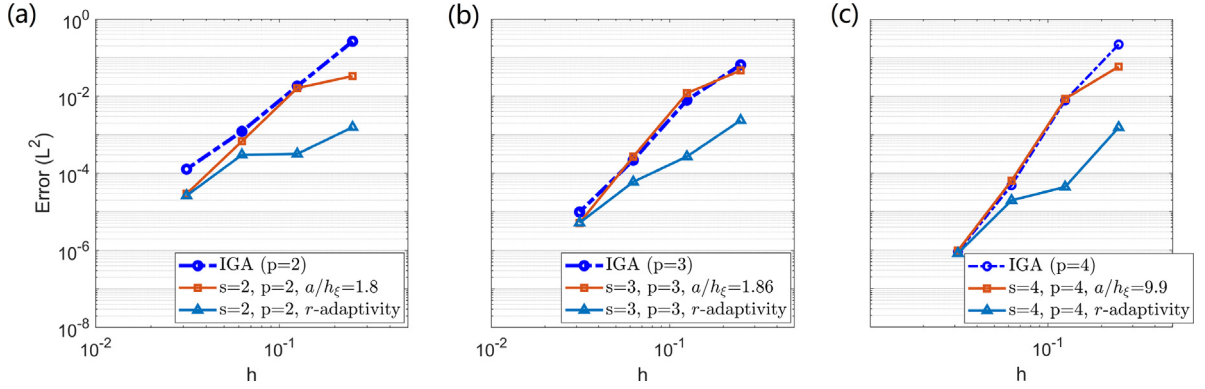


(a)  $h = L/40$

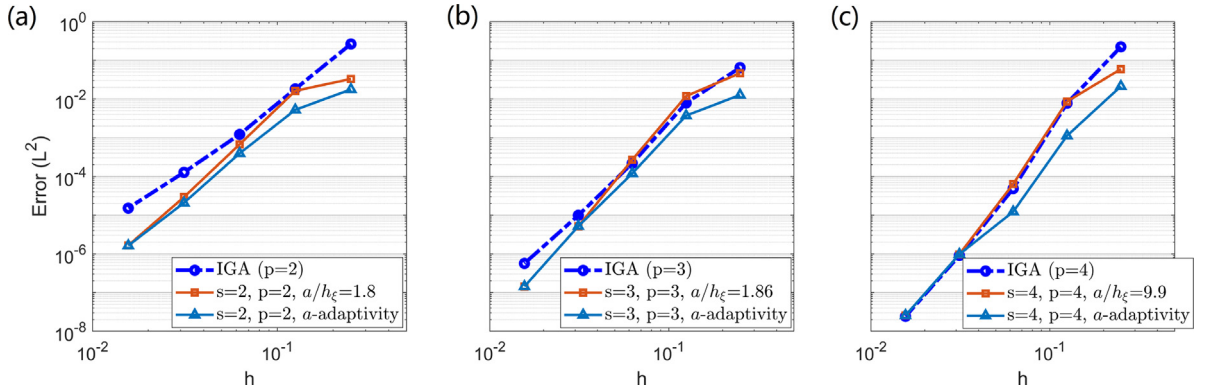


(b)  $h = L/80$

**Fig. 12.** Optimized mesh for C-IGA with  $r$ -adaptivity: initial mesh size (a)  $h = L/40$ ; (b)  $h = L/80$ . The parameters are set to be  $s = 2$ ,  $p = 2$ ,  $a/h_\xi = 1.8$ .



**Fig. 13.** Numerical results for C-IGA with  $r$ -adaptivity: (a)  $p = 2$ ; (b)  $p = 3$ ; (c)  $p = 4$ . The parameters are set to be  $s = 2, a/h_\xi = 1.8$  for  $p = 2$ ,  $s = 3, a/h_\xi = 1.86$  for  $p = 3$ , and  $s = 4, a/h_\xi = 9.9$  for  $p = 4$ .



**Fig. 14.** Numerical results for C-IGA with  $a$ -adaptivity: (a)  $p = 2$ ; (b)  $p = 3$ ; (c)  $p = 4$ .

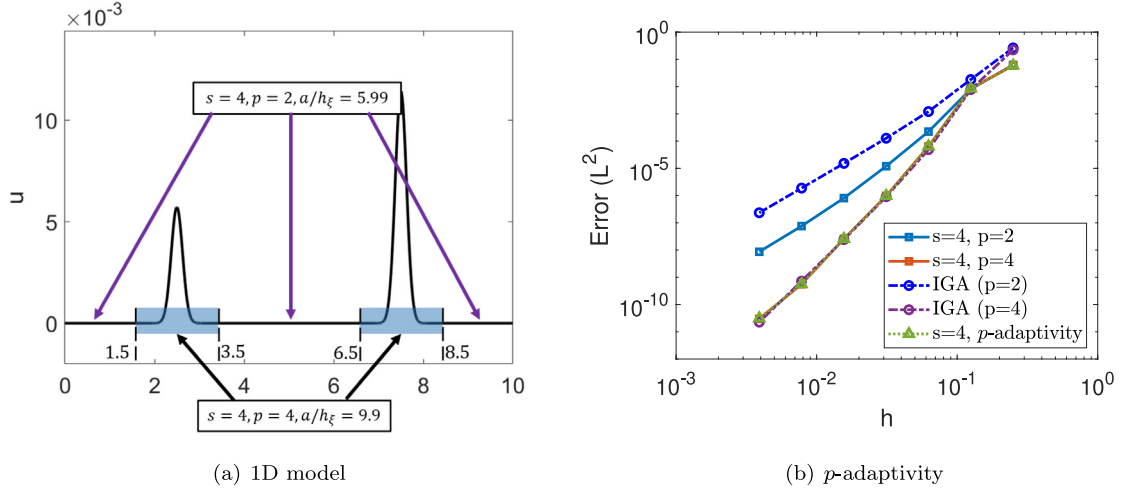
curve at these places needs more nodes compared to other flat parts. The  $L^2$ -norm error is plotted in Fig. 13. The  $r$ -adaptivity improves the accuracy considerably for coarse mesh. C-IGA errors with  $r$ -adaptivity are merely 0.5  $\sim$  5% of the original C-IGA errors for meshes  $h = L/40$  and  $L/80$ . While for fine mesh, the improvement from  $r$ -adaptivity becomes small because there are already enough nodes to capture the solution exactly.

### 5.1.3. $a$ -adaptivity

The C-IGA has an additional parameter  $a$ , called the dilation parameter, compared to FEM and IGA. We have observed a significant effect of  $a$  on the accuracy in previous discussions. However, the previous analyses are based on a constant  $a$ , i.e., all nodal patches share the same  $a$ . Here,  $a$ -adaptivity allows different  $a^{\xi_i}$  for each nodal patch  $A_s^{\xi_i}$ , and aims to seek for the optimal distribution of  $a^{\xi_i}$  in the global domain. Again, we adopt the PSO algorithm because of the oscillatory nature of the cubic spline radial function with small  $a$  as shown in Fig. 10(a), which brings a big challenge for gradient-based optimizers; Gradient-based methods are more suitable for Gaussian radial functions. One of the advantages of  $a$ -adaptivity over the  $r$ -adaptivity is that we do not change the mesh during the optimization, and avoid the mesh distortion issue of the  $r$ -adaptivity. Here, for numerical convenience, we limit  $a^{\xi_i}/h_\xi \in (1, 20]$ . The numerical results from  $a$ -adaptivity are shown in Fig. 14. The  $a$ -adaptivity helps our method achieve better results than those for the best constant  $a$ , and also better than IGA results.

### 5.1.4. $p$ -adaptivity

Larger  $p$  results in a faster convergence rate and better accuracy, but this also leads to additional computational costs. To balance accuracy and efficiency, we expect to use high-order interpolants in high-gradient regions such as the two Gaussian humps in Fig. 8, and low-order interpolants in the remaining regions. This is the spirit of



**Fig. 15.** Illustrations numerical results for  $p$ -adaptivity: (a) illustrations of 1D model for  $p$ -adaptivity; (b) convergence study for  $p$ -adaptivity. We use the fourth order C-IGA interpolants in the blue region ( $x \in [1.5, 3.5] \cup [6.5, 8.5]$ ), and the second order interpolants for other regions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$p$ -adaptivity. The flexibility of the C-HiDeNN framework allows such operation by simply elevating the order  $p$  of the convolution patch functions corresponding to the nodes in the high-gradient regions without adding nodes or changing mesh. Note that the patch size  $s$  should be adjusted correspondingly to guarantee  $p$ th order reproducing property according to the requirements in Table 3. For illustrative purposes, the computational domain illustrated in Fig. 15(a) is divided into two parts: the high-gradient region where the two Gaussian humps are located, and the remaining regions for the flat part of the solution. The first region is covered by the blue box ( $x \in [1.5, 3.5] \cup [6.5, 8.5]$ ) in the figure. We use the fourth order C-IGA interpolants ( $s = 4, p = 4, a/h_\xi = 9.9$ ) in the first region, and the second order interpolants ( $s = 2, p = 2, a/h_\xi = 1.8$ ) for other regions. As shown in Fig. 15(b), we can observe that the numerical results with  $p$ -adaptivity reach the same order of accuracy as the complete fourth-order interpolants. This verifies the efficiency of  $p$ -adaptivity.

## 5.2. 2D Poisson's problem on a quarter-ring domain

Starting from 2D examples, we use GPU programming to speed up both the construction of convolution patch functions and the iterative solver. The authors have already demonstrated orders of magnitude speed up with GPU programming in the JAX-FEM paper [50] and the C-HiDeNN paper [12]. Note that JAX is the high-level Python library for GPU programming developed by Google [47]. As C-IGA formulation is similar to C-HiDeNN, the parallel program of C-IGA is developed according to that of C-HiDeNN (readers may refer to Algorithm 1 of the Ref. [12]). We use an NVIDIA A6000 GPU with 48 GB of GPU memory.

### 5.2.1. Convergence study

Let us consider a quarter ring as illustrated in Fig. 16. The ring is the region between two concentric circles whose inner and outer radii are 10 and 20 (unitless), respectively. We solve a Poisson's problem given by:

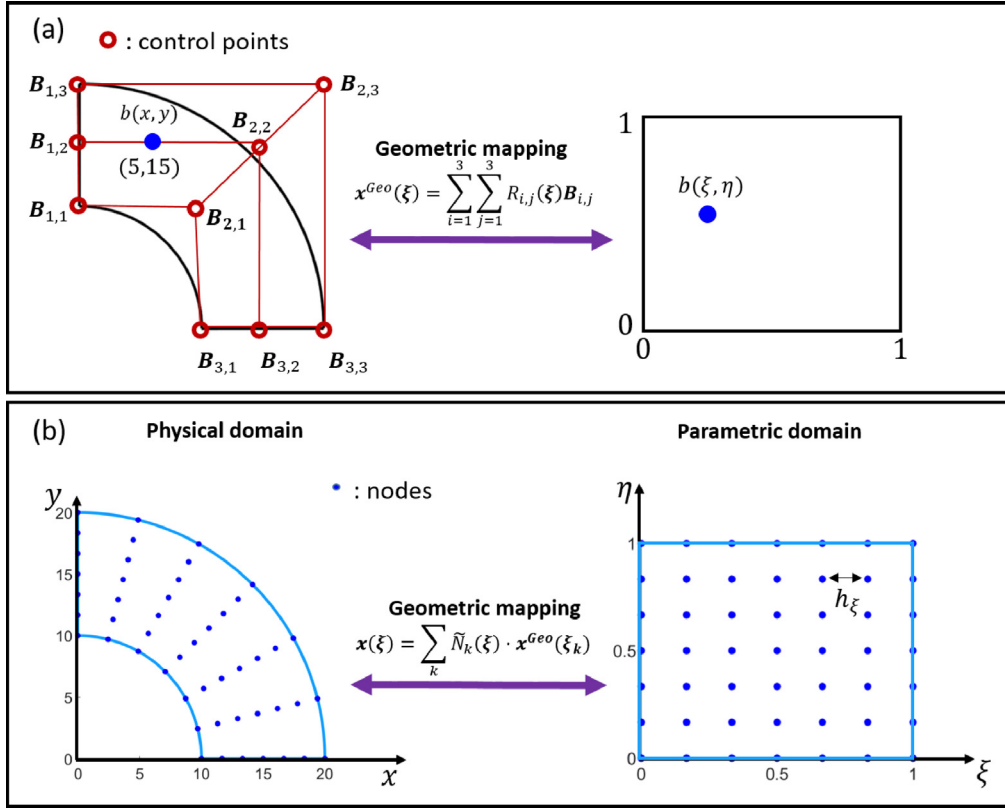
$$\Delta u(x, y) + b(x, y) = 0, (x, y) \in \Omega. \quad (45)$$

Let the body force  $b(x, y)$  be centered at (5, 15):

$$\begin{aligned} b(x, y) &= -\nabla u^{Ext}(x, y) \\ &= -(4\pi^2(x-5)^2 + 4\pi^2(y-15)^2 - 4\pi)e^{-\pi(x-5)^2}e^{-\pi(y-15)^2}, \end{aligned} \quad (46)$$

leading to an analytical solution (a Gaussian hump at (5, 15)):

$$u^{Ext}(x, y) = e^{-\pi(x-5)^2}e^{-\pi(y-15)^2}. \quad (47)$$



**Fig. 16.** 2D model with a body force centered at (5, 15) and geometric mapping. (a) Let IGA mapping with NURBS basis functions be the geometric mapping  $\mathbf{x}^{Geo}(\xi)$ . (b) Isoparametric mapping of C-IGA transforms the physical mesh to the parametric mesh.

The boundary condition is

$$u|_{\partial\Omega} = u^{Ext}|_{\partial\Omega}, \quad (48)$$

which is almost zero.

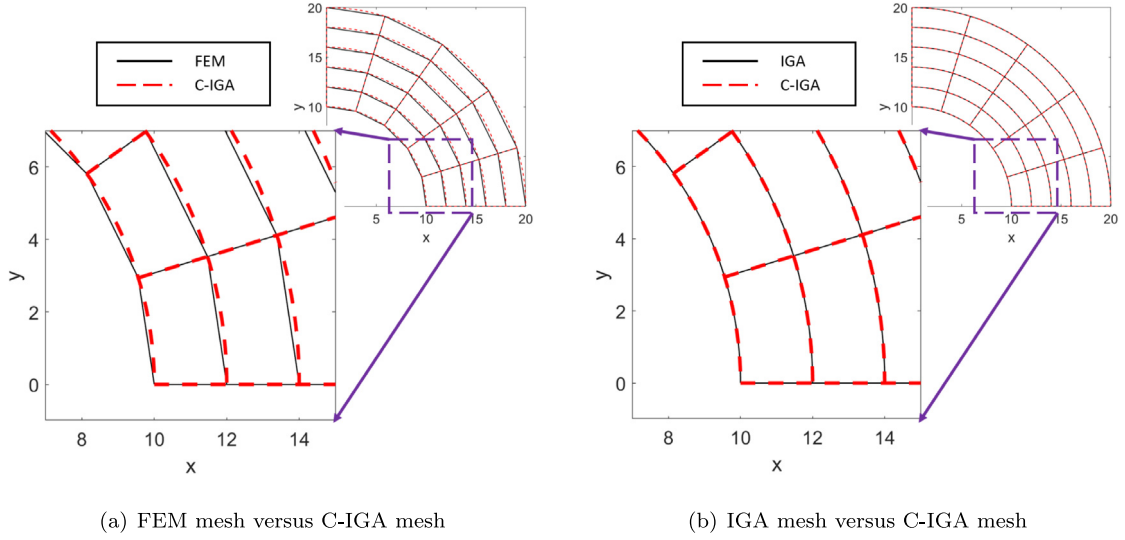
We use an IGA mapping as the geometric mapping  $\mathbf{x}^{Geo}(\xi)$ , given by

$$\mathbf{x}^{Geo}(\xi) = \sum_{i=1}^3 \sum_{j=1}^3 R_{i,j}(\xi, \eta) B_{i,j}, \quad (49)$$

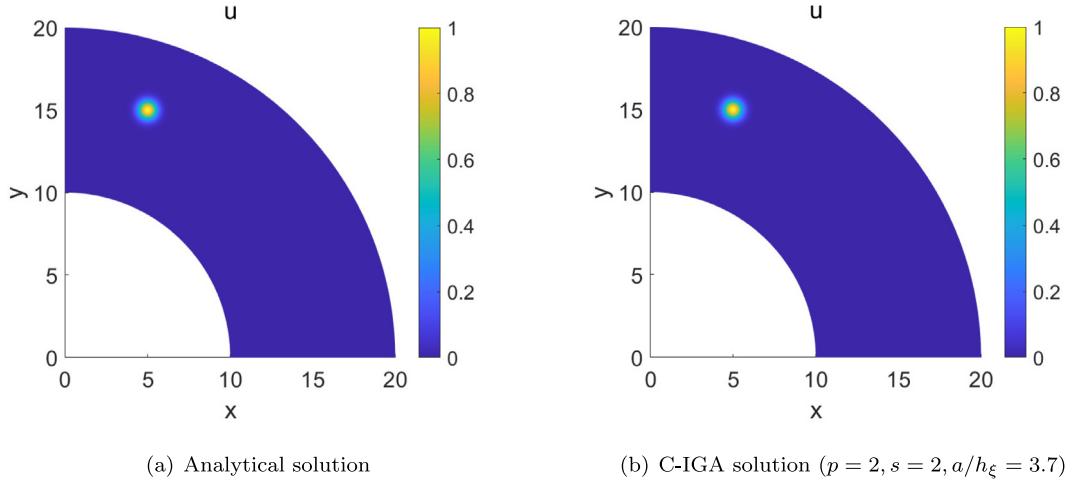
where  $R_{i,j}(\xi, \eta)$  are NURBS basis functions and  $B_{i,j}$  are control points. The detailed geometric data is given in [Appendix H](#). The IGA geometric mapping from the parametric domain to the physical domain is shown in [Fig. 16\(a\)](#) with control points. [Fig. 16\(b\)](#) illustrates how the nodes and elements defined in the parametric domain are mapped into the physical domain using the IGA mapping. It is important to note that once the nodes in the parametric domain are mapped into the physical domain with IGA mapping ( $\mathbf{x}_k = \mathbf{x}^{Geo}(\xi_k)$ ), the field mapping constructed by C-IGA ( $\mathbf{x}(\xi) = \sum_k \tilde{N}_k(\xi) \xi_k$ ) becomes exactly the same as the IGA mapping according to the proof in [Section 3.2](#). This renders the isoparametric feature of C-IGA ( $\mathbf{u}(\xi) = \sum_k \tilde{N}_k(\xi) \mathbf{u}_k$ ). We define the relative energy-norm error to measure the accuracy:

$$error = \frac{\|u^h - u^{Ext}\|_E}{\|u^{Ext}\|_E}, \quad \|u\|_E = \int_{\Omega} |\nabla u|^2 dx dy. \quad (50)$$

For comparison, we also created a FEM mesh based on the nodes and elements in the physical domain. Simply put, the nodes in the physical domain are linearly connected like standard FEM meshes as illustrated in [Fig. 17\(a\)](#).



**Fig. 17.** Mesh comparisons among linear FEM, IGA and C-IGA ( $p = 2$ ). IGA and C-IGA meshes can reproduce the geometry exactly, whereas linear FEM mesh approximates the geometry with straight lines.



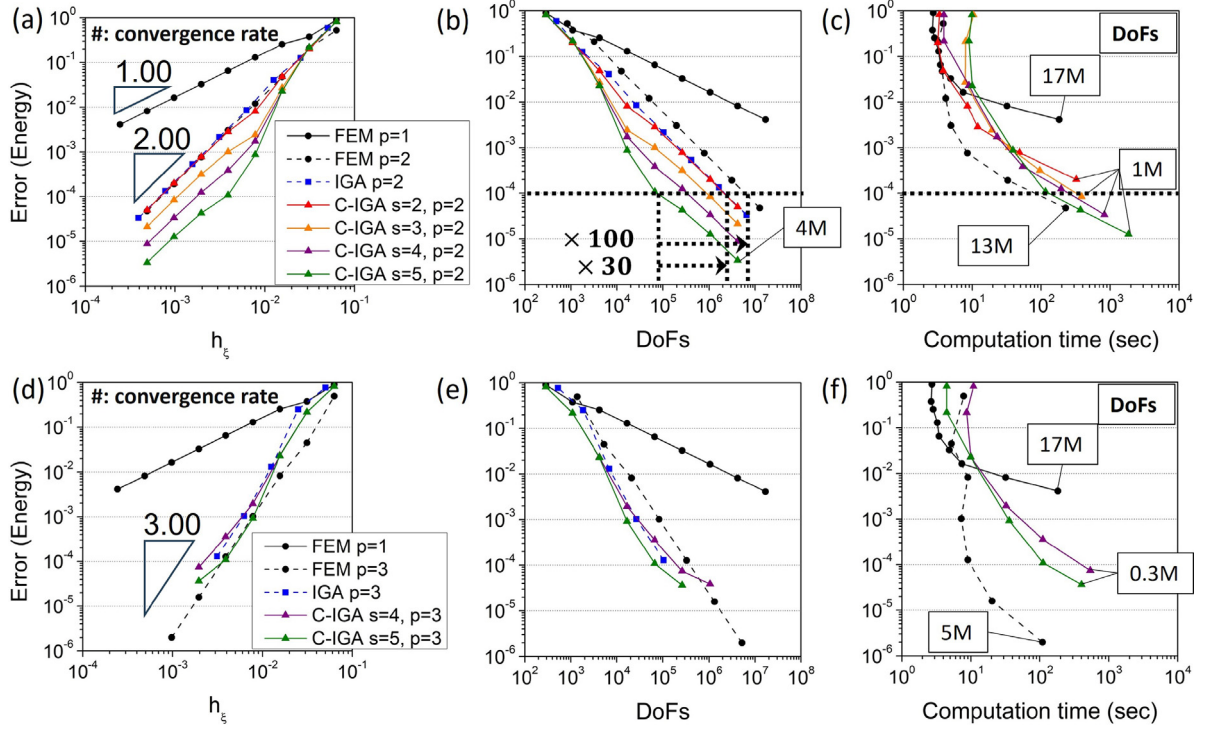
**Fig. 18.** Exact solution (a) versus C-IGA solution with 160 by 160 elements (b).

On the other hand, in Fig. 17(b), we can see the equivalence between IGA and C-IGA mappings. *The difference takes place when the order  $p$  is increased: IGA has additional repeated knots at the boundary with more basis functions and DoFs while C-IGA has no repeated nodes and keeps DoFs unchanged.*

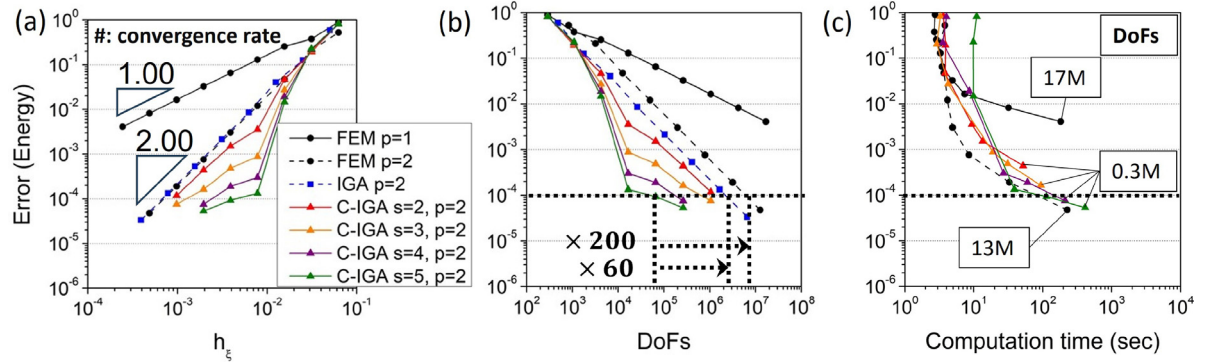
Fig. 18 illustrates the C-IGA solution ( $p = 2$ ) compared with the analytical one  $u^{Ext}$ . The results of convergence studies are plotted in Fig. 19 for cubic spline radial functions and Fig. 20 for Gaussian radial functions. In those figures, we compare the energy-norm error of FEM, IGA, and C-IGA with different  $s$  and fixed  $p$ . Fig. 19(a, b, c) and Fig. 20(a, b, c) are for  $p = 2$  and Fig. 19(d, e, f) are for  $p = 3$ . Note that the largest C-IGA problem has around 4 million DoFs.

In Fig. 19(a), IGA ( $p = 2$ ), FEM ( $p = 2$ ), and C-IGA ( $s = 2, p = 2$ ) have almost the same error for a given mesh size. However, as  $s$  increases, C-IGA shows faster convergence around  $h_\xi = 10^{-2}$ , thus resulting in better accuracy for finer meshes. Note that the convergence rates of all cases are the same as 2, except for the linear FEM ( $p = 1$ ). Although the FEM ( $p = 2$ ) has a similar error compared to IGA ( $p = 2$ ) and C-IGA ( $s = 2, p = 2$ ) at the given mesh size, it requires more DoFs than both IGA and C-IGA because of the higher order element. As



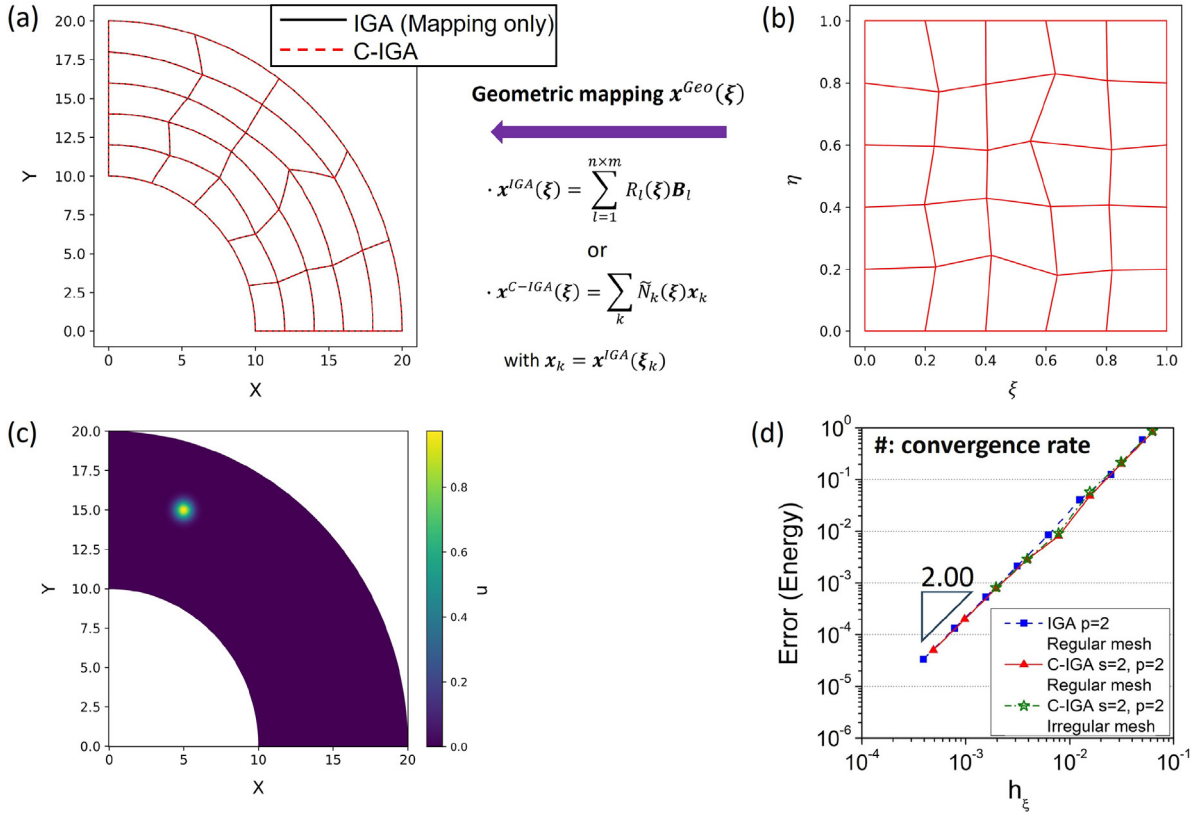


**Fig. 19.** Convergence study results for 2D Poisson's problem with cubic spline radial function and  $al/h_\xi = 50$ . (a–c) Energy norm error for  $p = 2$  vs. (a) mesh size  $h_\xi$ , (b) DoFs, and (c) computation time. (d–f) Energy norm error for  $p = 3$  vs. (d) mesh size  $h_\xi$ , (e) DoFs, and (f) computation time. Convergence rates are denoted in (a, d).  $h_\xi$  is the element size in the parametric domain. In (b, c), horizontal lines are drawn at  $10^{-4}$  error to compare the efficiency of C-IGA ( $s = 5, p = 2$ ) with IGA and FEM ( $p = 2$ ). FEM ( $p = 2$ ) and FEM ( $p = 3$ ) use serendipity elements: 8-node and 12-node elements, respectively. All computations are conducted using single GPU: NVIDIA A6000.



**Fig. 20.** Convergence study results for 2D Poisson's problem with Gaussian radial function and  $al/h_\xi = 3.2$ . (a–c) Energy norm error for  $p = 2$  vs. (a) mesh size  $h_\xi$ , (b) DoFs, and (c) computation time. Degrees of freedom (DoFs) are marked. In (b, c), horizontal lines are drawn at  $10^{-4}$  error to compare the efficiency of C-IGA ( $s = 5, p = 2$ ) with IGA and FEM ( $p = 2$ ).

shown in Fig. 19(b), to achieve  $10^{-4}$  error, C-IGA ( $s = 5, p = 2$ ) only requires around  $6 \times 10^4$  DoFs while IGA ( $p = 2$ ) and FEM ( $p = 2$ ) needs around  $2 \times 10^6$  and  $6 \times 10^6$  DoFs, respectively, that is, around 30 times and 100 times more DoFs than C-IGA. The accuracy of C-IGA improves when we use the Gaussian radial function with a properly chosen dilation parameter. As shown in Fig. 20(b), IGA ( $p = 2$ ) and FEM ( $p = 2$ ) require around 60 and 200 times more DoFs than C-IGA ( $s = 5, p = 2$ ).



**Fig. 21.** C-IGA mapping from (b) the irregular parametric mesh to (a) the physical mesh for 5 by 5 elements. IGA can only do the mapping; it cannot solve the problem. (c) Solution field of 128 by 128 irregular mesh in the parametric domain. (d) Convergence plot for IGA ( $p = 2$ ), C-IGA ( $s = 2, p = 2$ ) with regular and irregular mesh. The cubic spline radial function is used with  $ah_\xi = 50$ .

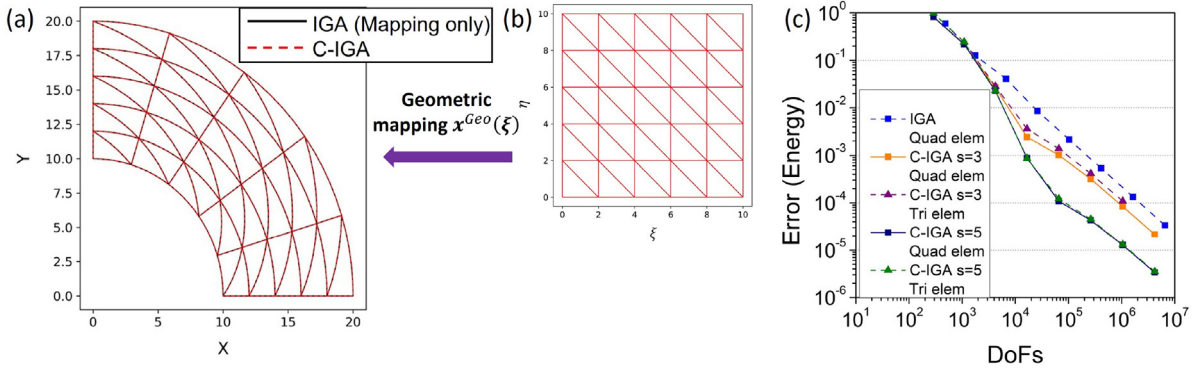
General trends of  $p = 3$  shown in Fig. 19(d, e) are similar to those of  $p = 2$ . C-IGA accuracy improves as  $s$  goes up. Although it seems FEM ( $p = 3$ ) has the lowest error among all others given the mesh size, it requires much larger DoFs (almost 5 times more than 4-node elements) because of the 12-node serendipity element. In Fig. 19(e), C-IGA with higher  $s$  still gives the lowest error given the same DoFs.

Computation time is plotted against the error in Fig. 19(c, f) for the cubic spline radial function and in Fig. 20(c) for the Gaussian radial function. When  $p = 2$ , C-IGA with a large  $s$  is faster than that with a small  $s$  when the error is  $10^{-4}$ , indicating that the increased computation cost for higher  $s$  is compensated by enhanced accuracy. A similar trend is observed in Fig. 19(f) when  $p = 3$ . In general, FEM is slightly faster than C-IGA when the error level is high (problems with small DoFs), but the gap is reduced or even reversed as the problem size grows. At the error level of  $10^{-4}$  in Fig. 19(c), C-IGA ( $s = 5, p = 2$ ) is slightly slower than FEM ( $p = 2$ ). On the other hand, in Fig. 20(c), C-IGA ( $s = 5, p = 2$ ) is slightly faster than FEM ( $p = 2$ ). We note that most computational costs of C-IGA come from the construction of convolution patch functions, which can be further parallelized with multiple GPUs. Implementation of C-IGA programming on multiple GPUs is therefore the next step we should pursue.

Although the Gaussian radial function is more accurate than the cubic spline radial function, there is an unstable moment matrix issue with the Gaussian radial function as the mesh refines and  $s$  increases. This is the reason why C-IGA plots in Fig. 20 have fewer points than those in Fig. 19. This needs to be addressed in future work.

### 5.2.2. Irregular and triangular parametric mesh

As discussed in Section 3.7 and Fig. 7, C-IGA can handle irregular meshes in the parametric domain. Fig. 21 shows that we can create an irregular mesh in the parametric domain (b) and directly map it to the physical domain (a) using both IGA and C-IGA mappings (they are equivalent). However, IGA cannot create shape functions based



**Fig. 22.** C-IGA mapping from the parametric mesh with triangular elements (b) to the physical mesh (a). IGA can only do the mapping; it cannot solve the problem. (c) Error vs. DoFs plot for IGA ( $p = 2$ ), C-IGA ( $s = 3$  and  $s = 5$ ) with quadrilateral and triangular elements. The cubic spline radial function is used with  $a/h_\xi = 50$ .

on the irregular parametric mesh shown in Fig. 21(b). Fig. 21(c) is the solution field computed from a 128 by 128 irregular mesh in the parametric domain, which still accurately captures the exact solution field displayed in Fig. 18(a). The convergence plot in Fig. 21(d) also confirms the irregular mesh can reliably capture the solution field. This naturally leads to the capability of r-adaptivity of C-IGA, similar to standard FEM.

C-IGA can also handle triangular elements in IGA framework. To the best of the author's knowledge, IGA formulated with triangular elements has never been reported in the literature. Fig. 22(a, b) illustrates how the triangular elements in the parametric domain are mapped into the physical domain. Again, the C-IGA mapping is identical to the IGA mapping but the IGA formulation cannot solve the problem with triangular elements. In Fig. 22(c), the energy norm error is plotted with DoFs for  $p = 2$  cases: IGA (Quadrilateral elements), C-IGA ( $s = 3$ , Quadrilateral and triangular elements), and C-IGA ( $s = 5$ , Quadrilateral and triangular elements). It is clearly shown that triangular elements can achieve the same level of accuracy as quadrilateral elements for the same DoFs.

### 5.3. 2D linear elasticity: Infinite plate with a circular hole

An infinite plate with a circular hole problem [34,51] is solved. The problem is defined in Fig. 23(a) and the exact solution is given below:

$$\sigma_{xx}(r, \theta) = 1 - \frac{R^2}{r^2} \left( \frac{3}{2} \cos 2\theta + \cos 4\theta \right) + \frac{3}{2} \frac{R^4}{r^4} \cos 4\theta \quad (51)$$

$$\sigma_{yy}(r, \theta) = -\frac{R^2}{r^2} \left( \frac{1}{2} \cos 2\theta - \cos 4\theta \right) - \frac{3}{2} \frac{R^4}{r^4} \cos 4\theta \quad (52)$$

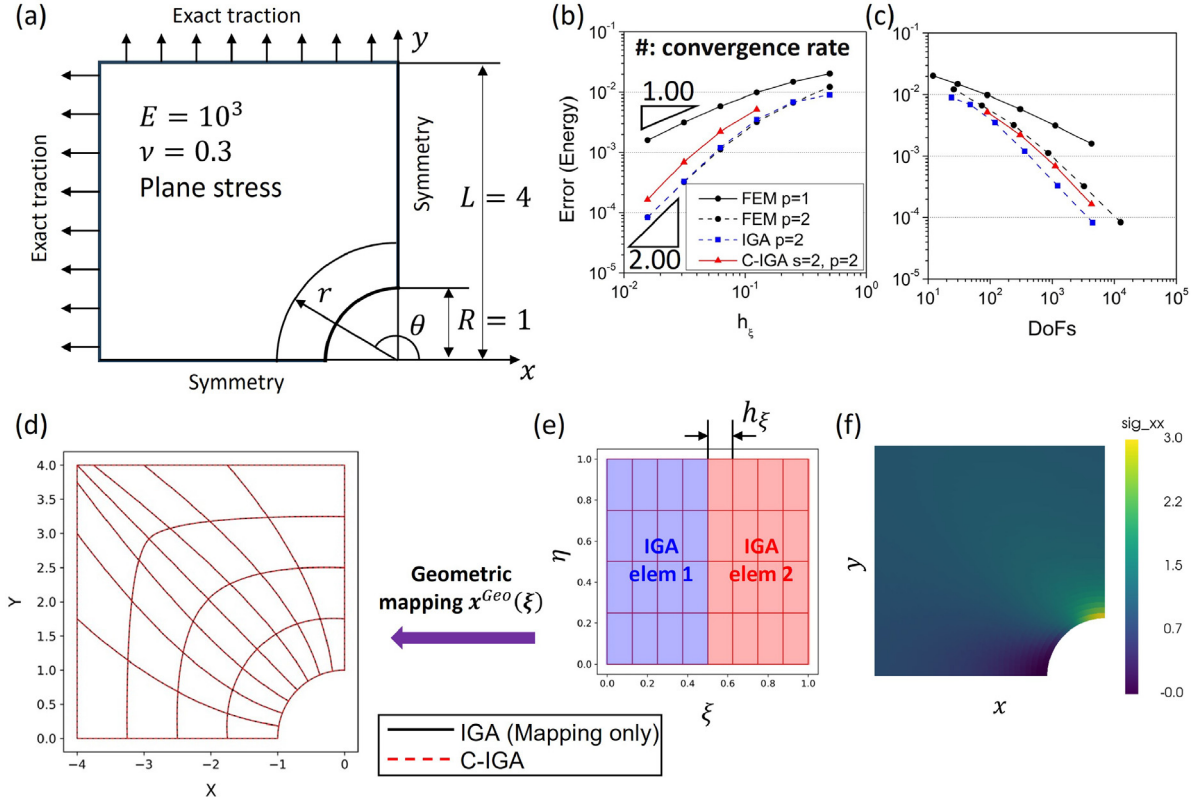
$$\sigma_{xy}(r, \theta) = -\frac{R^2}{r^2} \left( \frac{1}{2} \sin 2\theta + \sin 4\theta \right) + \frac{3}{2} \frac{R^4}{r^4} \sin 4\theta \quad (53)$$

where  $r$  and  $\theta$  are the polar coordinates centered at the center of the circle and  $R$  is the radius of the circle, which is set as 1. We assume a plane stress condition with the elastic modulus of  $10^3$  and the Poisson's ratio of 0.3. All values are unitless. One-quarter of the plate is simulated under the axis-symmetric boundary condition and the exact traction force is applied at  $x = -4$  and  $y = 4$  surface.

The minimum order IGA mapping is defined by the knot vectors:

$$\Xi = \{0, 0, 0, 0.5, 1, 1, 1\}, H = \{0, 0, 0, 1, 1, 1\} \quad (54)$$

with order  $p = 2$ . Corresponding control points and weights for the NURBS basis function can be found in Appendix A.3 of the original IGA paper [34]. Two IGA elements are defined with these knot vectors: element 1 -  $[0, 0.5] \times [0, 1]$ ; element 2 -  $[0.5, 1] \times [0, 1]$ , which are denoted in Fig. 23(e). These IGA elements are different



**Fig. 23.** 2D linear elastic example: infinite plate with a circular hole. (a) Problem definition with parameters. Convergence study results: energy norm error vs. (b)  $h_\xi$  and (c) DoFs. Gaussian radial function is used for the convergence study with  $alh_\xi = 8$ . C-IGA mapping from (e) the parametric mesh to (d) the physical mesh. The parametric mesh size  $h_\xi$  is denoted. Two IGA elements from the lowest order are denoted in (e): element 1 -  $[0, 0.5] \times [0, 1]$ ; element 2 -  $[0.5, 1] \times [0, 1]$ . (f)  $\sigma_{xx}$  plot (unitless) of 64 by 64 regular elements in the parametric domain. The Gaussian radial function is used with  $alh_\xi = 8$ ,  $s = 2$ ,  $p = 2$ .

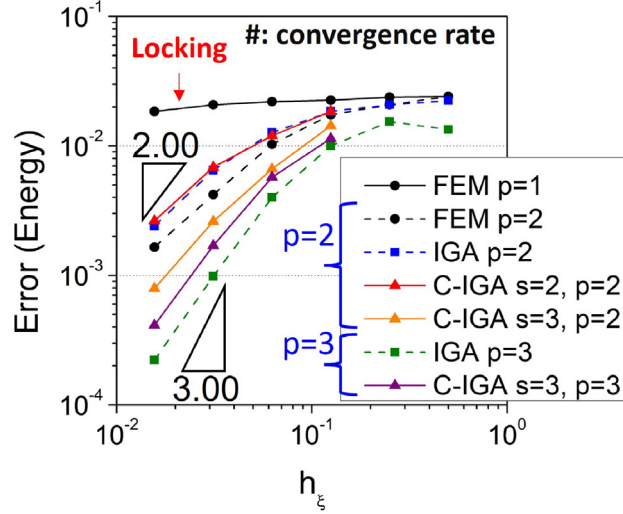
from the elements on which the convolution patch functions are built. They are fixed elements once the minimum order IGA mapping is defined. It is important to note that when we construct convolution patch functions in the parametric domain, both the nodal patch  $A_s^{\xi_i}$  and the elemental patch  $A_s^e$  must be confined to one IGA element where the element  $e$  belongs to. This is because each IGA element has different basis functions. If a patch domain invades multiple IGA elements, the equivalence between the IGA and C-IGA mappings (stated in Eq. (21)) no longer holds because the reproducing condition is violated.

Since we have the analytical solution, convergence studies are available. Fig. 23(b, c) are the results where the energy norm error is measured and the Gaussian radial function is used with  $alh_\xi = 8$ . As shown in Fig. 23(b), the C-IGA error is slightly higher than IGA or FEM ( $p = 2$ ) for a given mesh size although all of them have the same theoretical convergence rate of 2. When it comes to the DoFs in Fig. 23(c), for a given level of error, IGA needs the smallest DoFs while C-IGA and FEM ( $p = 2$ ) need more than IGA. Note that we have not conducted any adaptivity for C-IGA. A careful adaptivity of C-IGA such as  $r$ ,  $s$ ,  $a$ ,  $p$ -adaptivity might significantly reduce the error. Finally, Fig. 23(f) shows the  $\sigma_{xx}$  plot computed from a 64 by 64 regular element domain. As expected from the exact solution, there is a stress peak at  $r = 1$  and  $\theta = \frac{\pi}{2}$ , which is 3.0.

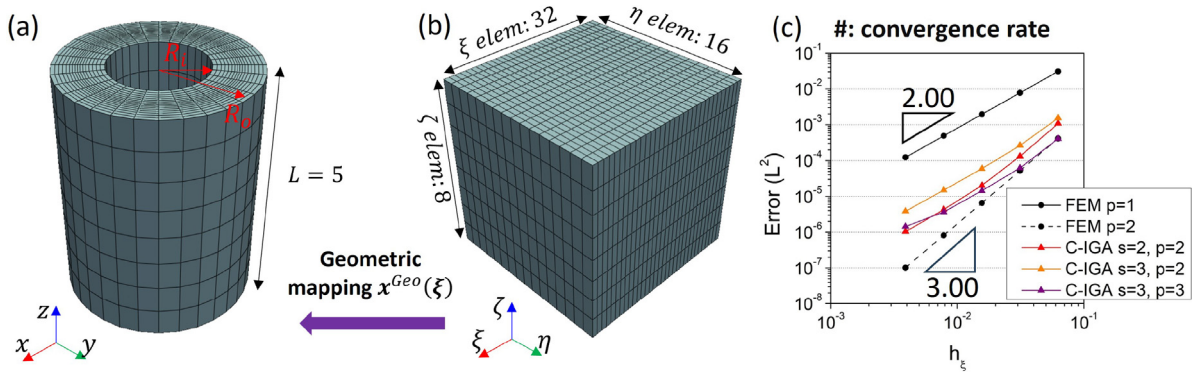
### 5.3.1. Removal of locking

We changed the problem to plane strain elasticity and Poisson's ratio to 0.4999 to study the locking behavior. As shown in Fig. 24, a severe locking is observed from linear FEM ( $p = 1$ ) while the other cases are locking-free. The IGA is expected to be locking-free because the minimum order of the NURBS basis of the current problem is 2. C-IGA is also locking-free because we can achieve an arbitrary reproducing order according to the C-HiDeNN





**Fig. 24.** Convergence study of 2D linear elasticity with plane strain and Poisson's ratio of 0.4999. Convergence rates are denoted. A severe locking is observed from linear FEM ( $p = 1$ ). The Gaussian radial function uses  $alh_\xi = 8$  with varying  $p$  and  $s$ .  $h_\xi$  is the element size in the parametric domain.



**Fig. 25.** 3D linear elastic example: solid circular cylinder subjected to internal pressure loading. Physical mesh (a) and Parametric mesh (b) of  $32 \times 16 \times 8$  elements.  $R_i = 1$ ,  $R_o = 2$ , and  $L = 5$ . The top and bottom surfaces of the cylinder are given symmetric boundary conditions. (c) Convergence study results for L2 norm error.  $h_\xi$  is the element size in the parametric domain. The cubic spline radial function is used with  $alh_\xi = 50$ .

interpolation theory. As long as  $p$  is greater than 1, C-IGA can remove locking similar to C-HiDeNN (readers may refer to [12]). Especially when  $p = 2$ , C-IGA ( $s = 3$ ) converges faster than C-IGA ( $s = 2$ ).

#### 5.4. 3D linear elasticity: Solid circular cylinder subjected to internal pressure loading

In this subsection, a 3D linear elasticity problem is solved. The circular cylinder shown in Fig. 25(a) is constructed from the control points and weights provided in [34]'s Section 4.2 and Appendix A.4. Elastic modulus and Poisson's ratio are set as  $E = 1000$  and  $\nu = 0.3$ , respectively. The internal pressure of  $P = 100$  is applied. All quantities are unitless. A  $32 \times 16 \times 8$  parametric mesh is generated in Fig. 25(b) while the corresponding physical mesh is shown in Fig. 25(a).

The exact solution to this problem under plane strain assumption is:

$$u_r(r) = \frac{PR_i^2}{R_o^2 - R_i^2} \frac{1 + \nu}{E} \left[ (1 - 2\nu)r + \frac{R_o^2}{r} \right] \quad (55)$$

where  $r$  is the radial distance from the center of the cylinder,  $R_i^2$  and  $R_o^2$  are the inner and outer radius of the cylinder, respectively. The  $L^2$  norm error convergence study is plotted in Fig. 25(c). The convergence rate of C-IGA ( $s = 2$ ,  $p = 2$ ) is close to its theoretical rate of 3.0 for coarse meshes but is reduced to 2.0 for fine meshes. Increasing  $s$  or  $p$  to 3 did not improve the accuracy and convergence rate. We attribute these suboptimal convergence rates to several reasons: (1) a suboptimal convergence of error on the boundary; (2) a relatively smooth exact solution; and (3) the rational form of the  $P(\xi)$  basis function. As discussed in Appendix D, the basis functions  $P(\xi)$  in C-IGA are set as polynomial functions divided by the NURBS weighting function to reproduce exact geometry. This might cause the suboptimal convergence for finer meshes. Further studies on these three aspects are needed.

### 5.5. Limitations of the current computational scheme

Although we have demonstrated the various capabilities and versatile adaptivity of C-IGA, there are some limitations of the current computational scheme that need further investigation.

- When we impose the Kronecker delta property to the convolution patch function, the moment matrix becomes unstable as the order  $p$  increases, especially when the Gaussian radial function is used. This issue occurs at large  $s$  or large  $a$ . Hence, it is recommended to take a smaller dilation parameter  $a$  for a larger patch size  $s$ , or a larger dilation parameter  $a$  for a smaller patch size  $s$  according to the order  $p$ . Considering that large  $s$  requires too much computation costs, the latter set might be a more efficient choice. Further mathematical analysis is needed to determine the optimal combination of  $s$  and  $a$  for a given  $p$ .
- In the case of the Gaussian radial basis, when the dilation parameter  $a$  is suboptimal, the iterative solver does not converge. This might hinder the automatic  $a$ -adaptivity.
- Most CAD models have only 3D surface information although the 3D IGA is defined on the 3D volumetric mesh. There are many approaches to resolve this issue within the IGA formulation [52–59], and some of these may be suitable with C-IGA. Further studies are required.
- As discussed in Section 5.4, the suboptimal convergence especially for large  $p$  needs to be investigated.

## 6. Outlook and conclusions

### 6.1. Multi-GPU programming

The recent surge in the development of generative AI, particularly due to the ChatGPT hype, has resulted in unprecedented attention to multi-GPU-based parallel computing (or accelerated computing) [60]. NVIDIA, which provided more than 10,000 A100 GPU computers for ChatGPT, has joined the \$1 trillion valuation club in June 2023. Many other big tech companies are also competing to develop parallel supercomputers for AI. Here are a few examples:

- NVIDIA announced that it will introduce the DGX GH200 supercomputer, capable of 1 Exaflop and 144 TB GPU memory, using NVLink interconnect technology [61].
- Tesla developed the D1 Dojo supercomputer specialized for autonomous driving and announced that it has achieved superior computing performance compared to NVIDIA [62].
- Google developed a parallel processor optimized for machine learning called TPU and announced that the supercomputer made with the latest model, TPU v4, surpassed NVIDIA's A100 supercomputer in performance and efficiency [63].

It may seem like these amazing computers are still far off for ordinary researchers, but computers evolve very quickly, and models often become outdated within a few years of development. We assert that in 5 to 10 years, the most recent NVIDIA supercomputer will be available for use in at least university research labs.

*Therefore, we need to develop mechanistic machine learning tools suitable for rapidly evolving AI supercomputers.*

According to the two papers [12,50], one of the key features of the GPU solver is that it can solve a linear system of equations considerably fast if we can store all the variables in GPU memory. For example, solving a 1 million DoFs linear FEM takes around 10 s with GPU, while it takes more than 5 min with Abaqus running on CPU [50]. However, as the size of the GPU memory is limited, we are limited by the problem size. The largest problem that

we can solve with a single A6000 GPU (48 GB memory) is around 10 million DoFs. However, NVIDIA's DGX GH200 supercomputer, scheduled for release later this year, supports 144 TB of GPU memory, which means it can solve problems with up to 30 billion DoFs using highly smooth interpolation. It is not easy for existing FEM tools to solve 1 billion DoFs with linear elements. We expect C-HiDeNN and C-IGA to become the most widely used FEM analysis methods in the parallel computing era.

## 6.2. Isogeometric C-HiDeNN-TD (C-IGA-TD)

In order to enhance the efficiency of solving PDEs, especially for extra-large scale problems with large numbers of DoFs, C-HiDeNN-TD has been developed in our previous work [11,13]. It is a reduced order model of C-HiDeNN using the so-called separation of variables [10,64,65]. The C-HiDeNN-TD approximation for a 3D problem is written as

$$u^{C-HiDeNN-TD} = \sum_{q=1}^Q \left( \sum_{i=1}^{n_\xi} \tilde{N}_{\xi,i}(\xi) d_{\xi,i}^{(q)} \right) \left( \sum_{j=1}^{n_\eta} \tilde{N}_{\eta,j}(\eta) d_{\eta,j}^{(q)} \right) \left( \sum_{k=1}^{n_\gamma} \tilde{N}_{\gamma,k}(\gamma) d_{\gamma,k}^{(q)} \right), \quad (56)$$

where  $\tilde{N}_{\xi,i}$ ,  $\tilde{N}_{\eta,j}$ ,  $\tilde{N}_{\gamma,k}$  are the 1D C-HiDeNN convolution shape functions along  $\xi$ ,  $\eta$ ,  $\gamma$ -directions respectively,  $d_{\xi,i}^{(q)}$ ,  $d_{\eta,j}^{(q)}$ ,  $d_{\gamma,k}^{(q)}$  are corresponding nodal coefficients for the  $q$ th mode,  $Q$  is the number of modes, and  $n_\xi$ ,  $n_\eta$ ,  $n_\gamma$  are the number of nodes in the corresponding directions. By tensor decomposition, we only need to solve 1D problems in each direction instead of solving the full 3D problem. Thus the number of DoFs of the final discretized system is reduced from  $n_\xi \times n_\eta \times n_\gamma$  to  $(n_\xi + n_\eta + n_\gamma) \times Q$  (i.e., *exponential growth* to *linear growth*), which greatly enhances the efficiency. In general, we can expect the number of modes  $Q$  is much fewer than the number of nodes in each direction, i.e.,  $Q \ll n_\xi, n_\eta, n_\gamma$ .

It can be seen that the above TD form has one limitation: when the computational domain is an arbitrary domain and is not intrinsically separable, the separated representation cannot be applied directly. Some techniques have been developed to resolve this issue. For example, [66] immersing the non-separable domains onto a fully separable one and [64] using geometrical mapping to deal with layered domains where interfaces are not planar. In our previous work [10], we resolved this issue by using FEM mapping and an affine to map an arbitrary domain into a regular domain as illustrated in Fig. 26(a). In this work, C-IGA provides a straightforward way to map an arbitrary physical domain to a regular parametric domain ( $\tilde{\Omega} = [0, 1]^d$  with dimension  $d$ ). The proposed method is able to adopt the IGA mapping as the geometric mapping directly instead of the 2-step mapping in [10], as shown in Fig. 26(b). Compared with the original strategy, geometric mapping is more mature and convenient. The parametric domain facilitates the application of TD to C-HiDeNN, leading to isogeometric C-HiDeNN-TD. More detailed implementations and numerical tests of isogeometric C-HiDeNN-TD will be investigated in our future work.

## 6.3. Nonlinear isogeometric C-HiDeNN

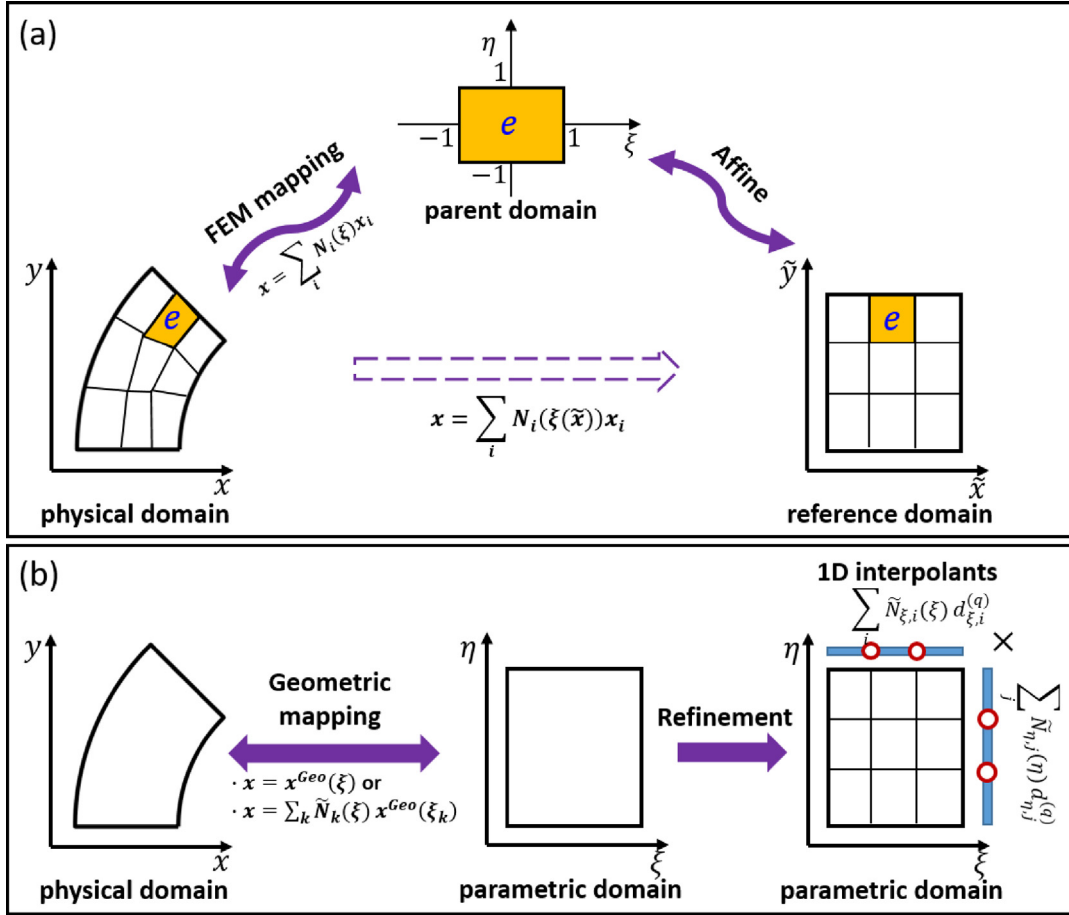
C-HiDeNN theory has been successfully applied to model nonlinear materials with large deformation using the total Lagrangian formulation. As discussed before, classical numerical solvers belong to the realm of engineering software 1.0, where the solvers have to be adapted for different material models. For example, in implicit analysis, the material tangent stiffness matrix has to be derived from different constitutive equations. Based on the idea of *Engineering Software 2.0*, the nonlinear C-HiDeNN solver is differentiable where it uses automatic differentiation to obtain the tangent stiffness matrix for a wide range of nonlinear materials, as shown in the equation below.

$$K_{iIjJ} = AD(f_{iI}, u_{jJ}) \quad (57)$$

where  $K_{iIjJ}$  is the tangent stiffness matrix,  $f_{iI}$  is the internal nodal force,  $u_{jJ}$  is the nodal displacement.  $AD$  can be efficiently implemented using JAX [47,50].  $AD$  can also make the C-HiDeNN nonlinear solver very powerful for inverse problems with multiple design variables since the gradient information can be obtained directly from the forward analysis.

To demonstrate the powerfulness of nonlinear C-HiDeNN differentiable solver, Fig. 27 shows a numerical example where a 3D hyperelastic plate with a hole is under tension in the  $x$  direction. Due to the symmetry of the problem, only a quarter plate is used for the current analysis where the left and bottom sides are fixed and





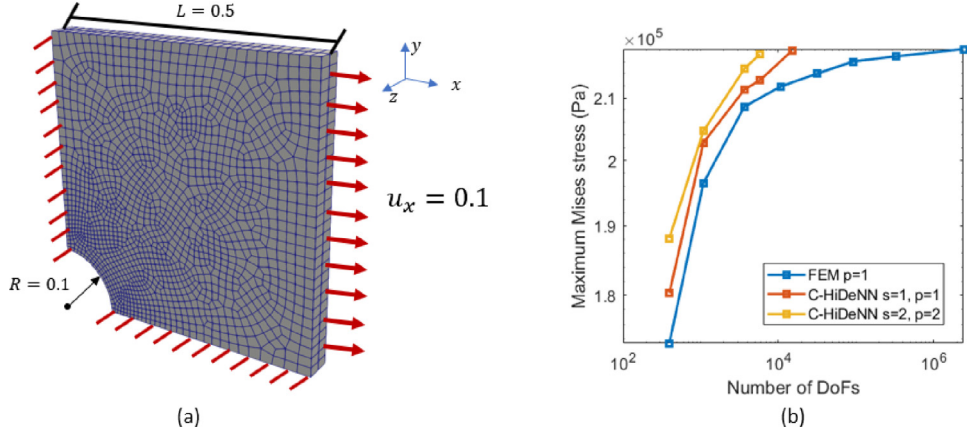
**Fig. 26.** Illustration for the geometrical mapping of isogeometric C-HiDeNN-TD. [10] proposed a 2-step mapping to transform an irregular physical domain to a regular reference domain by FEM mapping and an affine. The regular domain is intrinsically separable for TD. C-IGA uses geometric mapping instead, e.g., IGA mapping, which is mature and straightforward. (a) 2-step mapping in [10]. (b) Geometric mapping of isogeometric C-HiDeNN-TD.

a displacement boundary condition  $u_x = 0.1$  is applied on the right. The Neo-Hookean model is used as the constitutive law for the analysis where the strain energy density is shown as:

$$W = \frac{\mu}{2} (\bar{I}_1 - 3) + \frac{K}{2} (J - 1)^2 \quad (58)$$

where  $\mu$  is the initial shear modulus,  $K$  is the bulk modulus,  $\bar{I}_1 = J^{-\frac{2}{3}} I_1$ ,  $J = \det(\mathbf{F})$  and  $I_1$  is the first invariant of right Cauchy–Green tensor.  $\mu = 232$  kPa,  $K = 500$  kPa are adopted for the current analysis. The dimension of the quarter plate is given in Fig. 27(a). Fig. 27(b) shows the obtained maximum von Mises stress as a function of the numbers of DoFs. When the maximum von Mises stress is close to 218 kPa, nonlinear FEM requires 2435830 DoFs. However, C-HiDeNN  $p = 1$ ,  $s = 1$  requires 15183 DoFs (6.2% total DoFs of nonlinear FEM), C-HiDeNN  $p = 2$ ,  $s = 2$  requires 5850 DoFs (0.24% total DoFs of nonlinear FEM). As can be observed, nonlinear C-HiDeNN converges much faster than classical nonlinear finite element analysis.

Based on automatic differentiation, we will extend the current C-IGA framework to nonlinear materials with large deformation.



**Fig. 27.** (a) Tension of 3D hyperelastic plate with a hole; (b) Maximum Mises stress as a function of the number of DoFs.

#### 6.4. Isogeometric C-HiDeNN-TD (C-IGA-TD) for topology design and manufacturing

##### 6.4.1. C-HiDeNN in nested topology design

C-HiDeNN theory has been applied for nested scales topology optimization in a bamboo-like structure with tensor decomposition (TD) [10,13], to address a seven-scale concurrent design problem. Bamboo is a prevalent natural material with a nested structure, offering a lightweight design while retaining considerable strength [67]. The presented results are based on mechanistic nested topology design theory [13], implemented through the C-HiDeNN-TD approach. Interested readers may refer to [68,69] for more details of this design theory. C-HiDeNN theory is an excellent choice for topology optimization problem as it provides a built-in convolution filter with arbitrary smoothness. The design goal of this example problem is to optimize a nested bamboo-like structure under applied axial, bending, and torsional force for minimum compliance. The equality constraints are the equilibrium conditions and the volume fractions of the materials. We apply a Lagrange multiplier to enforce these constraints while minimizing the objective.

Three key parameters are considered for the nested TO design: nested scales ( $[n]$ ), design in each scale ( $[d]$ ), and material choice or design ( $m$ ). The resulting optimized structure for  $[n] = 7$  is presented in Fig. 28. With a constant volume fraction of 0.7 in each scale, a nested self-similar fractal design is obtained. The video of the 7-scale topology design can be found in the supplementary material. On the other hand, using different volume fractions in each scale yields a non-self-similar design. These results indicate that the method can generate fractal and beyond fractal type, opening new opportunities for multiscale topology optimization in product design. An animation of these structures is provided in the supplementary video. A summary of these designs is presented in Table 6 which demonstrates the capability of the C-HiDeNN-TD-TO (Topology Optimization) for an extremely large scale high-resolution ( $\sim 10^{30}$  voxels) topology design problem.

The performance of the bamboo-like structure for the seven scales is presented in Fig. 29. Introducing new lattice scales increases the stiffness for axial, bending, and torsional loading up to the fifth lattice scale. However, after that, introducing additional scales scarcely contributes to the stiffness of the structure. We printed the predicted structures using PLA filament as shown in Fig. 30. The zoomed-in version of the structure highlights the method's capability to create a smooth design.

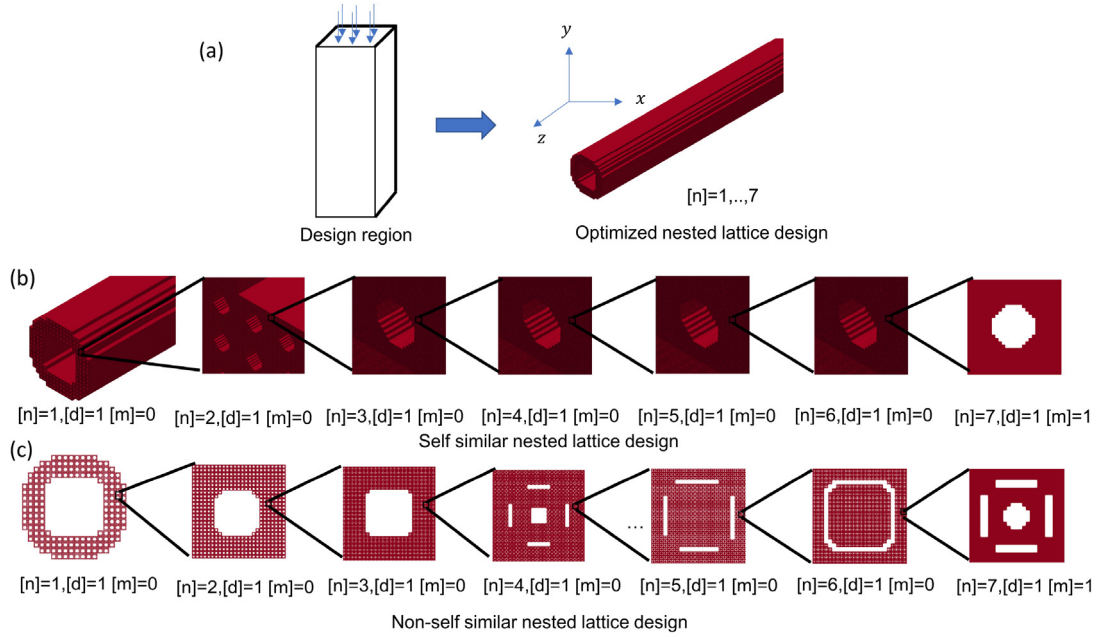
##### 6.4.2. C-IGA-TD in nested topology design of advanced materials systems

C-IGA-TD can facilitate complex geometries and boundary conditions in engineering product design. By utilizing the isogeometric technique, the C-IGA-TD topology optimization (TO) method directly incorporates CAD files into the design process, thereby eliminating the tedious post-processing for manufacturing. Moreover, employing the same shape function for both the solution and optimization ensures an accurate representation of the CAD file within the optimization scheme. C-IGA-TD holds the potential to be a powerful tool for transitioning from traditional *Engineering Software 1.0* to the more advanced *Engineering Software 2.0* across various applications, including topology design and additive manufacturing of advanced materials systems.

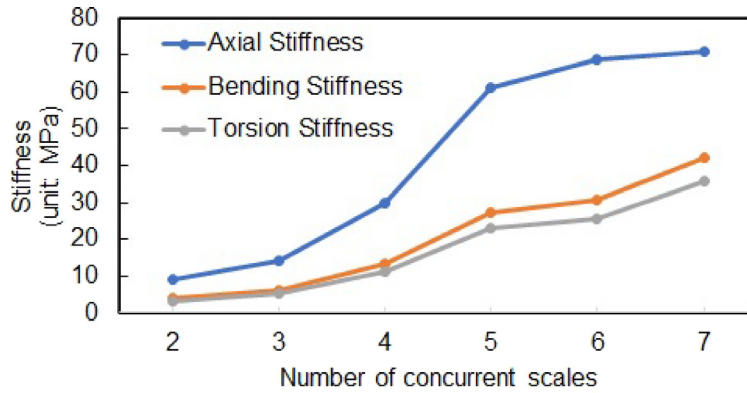
**Table 6**

Design parameters, objectives, performance for the nested scales design of bamboo-like structure.

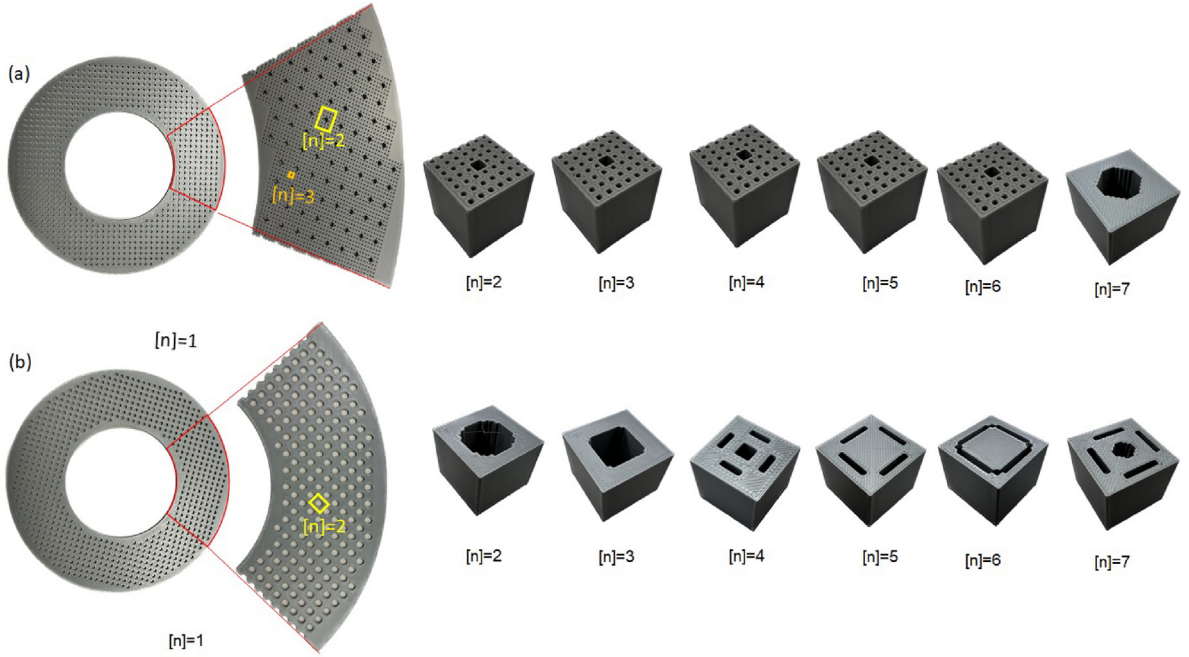
Design	Parameters	Objective	Force	Resolution (voxels)	Optimized performance (N · mm)
Self-similar design	$[n] = 1, \dots, 7$ $[d] = 1, \forall [n]$ $[m] = 1$	Compliance	Axial Bending Torsion	$2.27 \times 10^{29}$	Axial compliance: 140.9543 Bending compliance: 237.8368 Torsion compliance: 279.773
Non self-similar design	$[n] = 1, \dots, 7$ $[d] = 1, \forall [n]$ $[m] = 1$	Compliance	Axial Bending Torsion	$2.27 \times 10^{29}$	Axial compliance: 124.5211 Bending compliance: 215.2245 Torsion compliance: 286.2291



**Fig. 28.** (a) Initial design region and the optimized nested scales topology design showing a bamboo-like structure, (b) seven-scale self-similar topology design with the same volume fraction in each scale, and (c) seven-scale non self-similar design topology design with a varying volume fraction at different scales. For (b) and (c),  $[n] = 1$  has a resolution of  $25 \times 25 \times 250$ , and  $[n] = 2, \dots, 7$  have a resolution of  $25 \times 25 \times 25$ .



**Fig. 29.** Performance of the nested lattice structure design.



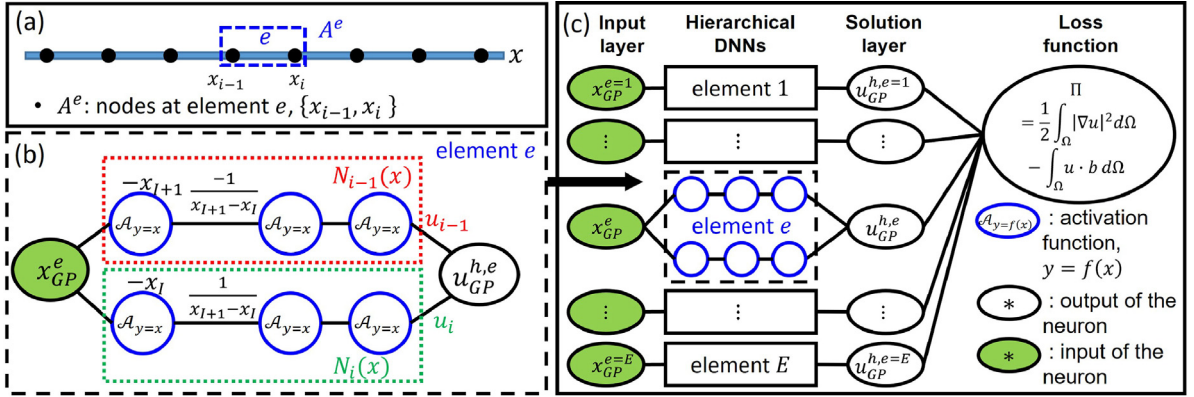
**Fig. 30.** 3D-printed bamboo-like structure for (a) self-similar nested lattice design, (b) non self-similar nested lattice design. A scaled lattice structure is printed for  $[n] = 2, \dots, 7$  to demonstrate the optimized structure. The existing manufacturing capability using fused filament fabrication additive manufacturing allows for the simultaneous observation of three scales.

### 6.5. Conclusions

An isogeometric version of the C-HiDeNN framework, called C-IGA, has been developed. C-IGA introduces an intermediate domain, parametric domain, and corresponding geometric mapping like IGA. The given geometric mapping is able to transform the parametric domain into the exact geometry in the physical domain. C-IGA reproduces such geometric mapping in the isoparametric framework by (1) assigning discrete physical nodes according to the geometric mapping and (2) reproducing the basis of the mapping function. C-IGA inherits the advantages of C-HiDeNN in the aspects of high-fidelity solutions and versatile adaptivity. Compared to FEM and IGA, C-IGA can achieve arbitrarily high-order smoothness and convergence rates without increasing DoFs. In addition to the order elevation, C-IGA allows the adjustment of other parameters to improve the performance, including (1) the patch size  $s$  that controls the number of neighboring nodes contributing to the nodal shape function and (2) the dilation parameter  $a$  that controls characteristic length scales. Another benefit of C-IGA is the Kronecker delta property, which eases the enforcement of boundary conditions and avoids the loss of optimal convergence rates. Lastly, unlike IGA, C-IGA can handle irregular meshes or low-order elements such as 3-node triangles without sacrificing accuracy.

Both theoretical analysis and numerical tests show that C-IGA can achieve  $(p + 1)$ th order of convergence rates for  $L^2$  norm error and  $p$ th order of convergence rates for  $H^1$  norm error, if the interpolants satisfy  $p$ th order reproducing conditions. Importantly, the adaptive nature of C-IGA is significantly flexible. The  $r$ ,  $a$ ,  $p$ -adaptivities are investigated in the numerical examples. The  $r$ -adaptivity optimizes nodal positions for more accurate mesh by minimizing the variational energy. The  $a$ -adaptivity searches for the optimal distribution of patch-wise dilation parameter  $a$  of convolution patch functions associated with each node. Both  $r$ -adaptivity and  $a$ -adaptivity improve the solution accuracy, especially for the coarse mesh. We explore the idea of  $p$ -adaptivity that adopts high-order interpolations in the important regions and low-order ones in the background region, resulting in a more efficient solution scheme.

The future research direction lies in developing multi-GPU programming of C-IGA, and its reduced-order extension, Isogeometric C-HiDeNN-TD, aiming to achieve significantly improved performance for extremely large-scale



**Fig. A.31.** Neural network structure of HiDeNN for a 1D scientific problem. (a) Nodal coordinates in the physical domain. Element  $e$  consists of nodes  $A^e = \{x_{i-1}, x_i\}$ . (b) Elemental neural network that mimics the linear shape function. A Gauss quadrature (or integration) point  $x_{GP}^e$  comes in, and the interpolated field variable  $u_{GP}^{h,e}$  is produced. The upper three neurons construct the linear shape function of node  $x_{i-1}$  and the lower three neurons construct the linear shape function of node  $x_i$ . Between the third hidden layer and the output layer, nodal variables  $u_{i-1}, u_i$  are set as weights. (c) The HiDeNN structure for the entire problem. The variational energy is used as a loss function of the problem. For more details, readers may refer to the original HiDeNN paper [9].

optimization problems. Isogeometric C-HiDeNN-TD will be designed to effectively handle boundary conditions and integrate CAD files seamlessly into the design process. This will significantly enhance the smoothness and accuracy of optimized designs with dramatically reduced DoFs and orders of magnitude of speed up. The ongoing development and application of C-IGA methodology pave the way for the transition from *Engineering Software 1.0* to *Engineering Software 2.0*, unlocking the immense potential for interconnectivity between simulation-based product development, industry-standard design, and real-time controlled additive manufacturing.

## 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.

## Data availability

The data that has been used is confidential.

## Acknowledgment

The authors would like to thank Dr. Mark Fleming and Dr. Dong Qian for their sincere and rigorous feedback.

## Appendix A. HiDeNN approximation

Hierarchical Deep-learning Neural Network (HiDeNN) [9,28] shown in Fig. A.31 is the foundation of the recently developed convolution HiDeNN (C-HiDeNN) theory [11–13]. The key idea of HiDeNN is that we can rewrite finite element shape functions with partially connected neural networks and simple activation functions (e.g., ReLU and identity) whose weights and biases are functions of nodal coordinates and nodal field variables. These elemental neural networks (Fig. A.31(b)) hierarchically construct “HiDeNN” in Fig. A.31(c), whose input is element-wise Gauss quadrature (or integration) points and loss function is the variational energy. If we fix the nodal coordinates and optimize the nodal field variables, the solution is mathematically equivalent to FEM. If the nodal coordinates are also optimized, we can achieve  $r$ -adaptivity, which in general improves local accuracy because the nodes tend to move toward the high gradient region [9,28].



## Appendix B. Radial basis functions

The radial basis function [41,42] is a specific option for the convolution patch function  $\mathcal{W}_{a,j}^{\xi_i}$  that simultaneously satisfies Kronecker delta and reproducing conditions, defined in Eq. (4). Denote a vector  $\mathcal{W}_{a,s}^{\xi_i} = [\mathcal{W}_{a,1}^{\xi_i}, \mathcal{W}_{a,2}^{\xi_i}, \dots, \mathcal{W}_{a,ns}^{\xi_i}]$  composed of all convolution patch functions in the nodal patch  $A_s^{\xi_i}$ . Denote nodes in  $A_s^{\xi_i}$  are  $\xi_j^{(i)}$ ,  $j = 1, 2, \dots, ns$ .  $ns$  is the number of nodes in  $A_s^{\xi_i}$ , i.e.,  $ns = n(A_s^{\xi_i})$ . Here,  $\xi$  refers to parent coordinates in parent domain for C-HiDeNN, or parametric coordinates in parametric domain for C-IGA. For a scalar patch size  $s$  in all directions,  $ns = (2s + 1)^d$  with dimension  $d$ . We write Eq. (4) in the matrix form

$$\mathcal{W}_{a,s}^{\xi_i}(\xi) = \Psi_a(\xi)A + P(\xi)K, \quad (\text{B.1})$$

where  $A = [\alpha_1, \alpha_2, \dots, \alpha_{ns}]$ ,  $K = [\kappa_1, \kappa_2, \dots, \kappa_{ns}]$  are coefficient matrices. The radial functions  $\Psi_a(\xi) = [\Psi_{a,1}(\xi), \Psi_{a,2}(\xi), \dots, \Psi_{a,ns}(\xi)]$  can be chosen as cubic spline functions Eq. (6) or Gaussian functions Eq. (7).  $P(\xi) = [P_1(\xi), P_1(\xi), \dots, P_m(\xi)]$  ( $m \leq ns$ ) are the functions to be reproduced in the reproducing conditions, e.g., polynomials ( $P(\xi) = [1, \xi, \xi^2, \dots, \xi^p]$  in 1D for example,  $m = p + 1$ ).

To satisfy Kronecker delta and reproducing conditions, we set coefficients to be

$$A = R^{-1}(I_{ns \times ns} - QK), K = (Q^T R^{-1} Q)^{-1} Q^T R^{-1} \quad (\text{B.2})$$

with  $I_{ns \times ns}$  denoting an  $ns$ -dimensional identical matrix. The moment matrices  $Q$  and  $R$  are both matrices composed of nodal values of  $\Psi_{a,j}(\xi)$  and  $P_k(\xi)$ :

$$R = \begin{bmatrix} \Psi_{a,1}(\xi_1^{(i)}) & \Psi_{a,2}(\xi_1^{(i)}) & \dots & \Psi_{a,ns}(\xi_1^{(i)}) \\ \Psi_{a,1}(\xi_2^{(i)}) & \Psi_{a,2}(\xi_2^{(i)}) & \dots & \Psi_{a,ns}(\xi_2^{(i)}) \\ \vdots & \vdots & & \vdots \\ \Psi_{a,1}(\xi_{ns}^{(i)}) & \Psi_{a,2}(\xi_{ns}^{(i)}) & \dots & \Psi_{a,ns}(\xi_{ns}^{(i)}) \end{bmatrix} \quad (\text{B.3})$$

and

$$Q = \begin{bmatrix} P_1(\xi_1^{(i)}) & P_2(\xi_1^{(i)}) & \dots & P_m(\xi_1^{(i)}) \\ P_1(\xi_2^{(i)}) & P_2(\xi_2^{(i)}) & \dots & P_m(\xi_2^{(i)}) \\ \vdots & \vdots & & \vdots \\ P_1(\xi_{ns}^{(i)}) & P_2(\xi_{ns}^{(i)}) & \dots & P_m(\xi_{ns}^{(i)}) \end{bmatrix} \quad (\text{B.4})$$

Detailed derivation of coefficients  $A$  and  $K$  is given as below.

The Kronecker delta and reproducing conditions are

$$\mathcal{W}_{a,j}^{\xi_i}(\xi_k^{(i)}) = \delta_{jk}, \quad j, k = 1, 2, \dots, ns \quad (\text{Kronecker delta}), \quad (\text{B.5})$$

$$\sum_{j=1}^{ns} \mathcal{W}_{a,j}^{\xi_i}(\xi) P_k(\xi_j^{(i)}) = P_k(\xi), \quad k = 1, 2, \dots, m \quad (\text{Reproducing conditions}). \quad (\text{B.6})$$

They are written in the matrix form as

$$RA + QK = I_{ns \times ns} \quad (\text{Kronecker delta}), \quad (\text{B.7})$$

$$(\Psi_a(\xi)A + P(\xi)K)Q = P(\xi), \quad (\text{Reproducing conditions}). \quad (\text{B.8})$$

Solving Eq. (B.7), we obtain

$$A = R^{-1}(I_{ns \times ns} - QK). \quad (\text{B.9})$$

Substituting it into the second equation Eq. (B.8), we have

$$\Psi_a(\xi)R^{-1}(I_{ns \times ns} - QK)Q + P(\xi)KQ = P(\xi), \quad (\text{B.10})$$

and then

$$\Psi_a(\xi)R^{-1}Q(I_{m \times m} - KQ) = P(\xi)(I_{m \times m} - KQ). \quad (\text{B.11})$$



Finally, we have

$$(\Psi_a(\xi)R^{-1}Q - P(\xi))(I_{m \times m} - KQ) = \mathbf{0}. \quad (\text{B.12})$$

For arbitrary  $\Psi_a(\xi)$  and  $P(\xi)$ , a general solution is

$$KQ = I_{m \times m}. \quad (\text{B.13})$$

Since  $Q$  is an  $ns \times m$  matrix, and might not be a square matrix, the coefficient matrix  $K$  is the pseudo-inverse of  $Q$ , which is not unique for  $m < ns$ . The radial basis function [41,42] adopts one specific solution

$$K = (Q^T R^{-1} Q)^{-1} Q^T R^{-1}. \quad (\text{B.14})$$

Remark that the least-square form

$$K = (Q^T Q)^{-1} Q^T \quad (\text{B.15})$$

is also the solution to Eq. (B.13). In fact,  $K$  can be

$$K = (Q^T M Q)^{-1} Q^T M \quad (\text{B.16})$$

with arbitrary  $ns$ -dimensional invertible matrix  $M$ . Remark that we can choose different  $M$  to adjust the condition number of  $Q^T M Q$ .

### Appendix C. Elemental interpolation of C-HiDeNN in matrix form

The C-HiDeNN formulation becomes similar to FEM if we re-write the interpolation in the matrix form:

$$\mathbf{u}^{h,e}(\tilde{\xi}) = \sum_{i \in A^e} (\tilde{N}_s^i(\tilde{\xi}))^T \mathbf{U}_s^i = (\tilde{N}_s^e)^T \mathbf{U}_s^e \quad (\text{C.1})$$

with  $\tilde{N}_s^i(\tilde{\xi}) = \mathcal{N}_i(\tilde{\xi}) \mathcal{W}_{a,s}^{\tilde{\xi}_i^e}(\tilde{\xi})$ . It is worth mentioning that the summation in Eq. (C.1) is only over the element nodes,  $A^e$ , like the first summation in Eq. (3). The  $i$ th node in the element  $e$  is denoted by  $\tilde{\xi}_i^e$ .  $\mathbf{U}_s^i$  is a matrix of nodal displacements in the nodal patch  $A_s^{\tilde{\xi}_i^e}$  as illustrated in Fig. 3(b). It has a dimension of  $[n(A_s^{\tilde{\xi}_i^e}), d]$  if the dimension of solution is  $d$ .  $\mathcal{W}_{a,s}^{\tilde{\xi}_i^e}(\tilde{\xi})$  is a vector of convolution patch functions  $\mathcal{W}_{a,j}^{\tilde{\xi}_i^e}(\tilde{\xi})$  associated with nodes  $\tilde{\xi}_j^e$  in the nodal patch  $A_s^{\tilde{\xi}_i^e}$ . Likewise, the length of the column vectors  $\mathcal{W}_{a,s}^{\tilde{\xi}_i^e}$  and  $\tilde{N}_s^i$  equals  $n(A_s^{\tilde{\xi}_i^e})$ , that is, it depends on the patch size  $s$ .

In FEM, nodal displacements are dealt as a column vector under Voigt notation. C-HiDeNN can do the same with the two column vectors, shape function vector  $\tilde{N}_s^e$  and nodal displacement vector  $\mathbf{U}_s^e$ , with a length of  $n(A_s^e) \times d$ . Note that the components of  $\tilde{N}_s^e$  are from  $\tilde{N}_k(\tilde{\xi})$  in Eq. (3).

The variational energy (for the elastic system) in one element  $e$  is

$$\begin{aligned} \Pi^e[\mathbf{u}^h(\mathbf{x})] &= \frac{1}{2} \int_{\Omega^e} \boldsymbol{\sigma} : \boldsymbol{\varepsilon} dV(\mathbf{x}) - \int_{\Omega^e} b(\mathbf{x}) \mathbf{u}^h dV(\mathbf{x}) \\ &= \frac{1}{2} \int_{\tilde{\Omega}^e} (\mathbf{U}_s^e)^T (\tilde{\mathbf{B}}_s^e)^T \mathbf{D} \tilde{\mathbf{B}}_s^e \mathbf{U}_s^e \det(\mathbf{J}) dV(\tilde{\xi}) \\ &\quad - \int_{\tilde{\Omega}^e} \mathbf{b}(\tilde{\xi}) (\tilde{N}_s^e)^T \mathbf{U}_s^e \det(\mathbf{J}) dV(\tilde{\xi}), \end{aligned} \quad (\text{C.2})$$

where  $\boldsymbol{\sigma}$  and  $\boldsymbol{\varepsilon}$  are the stress and strain respectively, and  $\mathbf{b}(\tilde{\xi})$  is the body force.  $\tilde{\mathbf{B}}_s^e$  is the strain-displacement matrix whose size is  $[3, n(A_s^e) \times d]$  for a 2D element.  $\mathbf{D}$  is the modulus matrix while  $\mathbf{J}$  is the Jacobian matrix defined by

$$\mathbf{J}(\tilde{\xi}) = \frac{\partial \mathbf{x}}{\partial \tilde{\xi}} \quad (\text{C.3})$$

with the isoparametric mapping  $\mathbf{x}(\tilde{\xi})$ . The integral in the parent domain  $\tilde{\Omega}^e = [-1, 1]^d$  can be realized by Gauss quadrature in the same way as FEM:

$$\begin{aligned} \Pi^e[\mathbf{u}^h(\mathbf{x})] &= \frac{1}{2} \sum_{GP} w_{GP} (\mathbf{U}_s^e)^T (\tilde{\mathbf{B}}_s^e(\tilde{\xi}_{GP}))^T \mathbf{D} \tilde{\mathbf{B}}_s^e(\tilde{\xi}_{GP}) \mathbf{U}_s^e \det(\mathbf{J}(\tilde{\xi}_{GP})) \\ &\quad - \sum_{GP} w_{GP} b(\tilde{\xi}_{GP}) (\tilde{\mathbf{N}}_s^e(\tilde{\xi}_{GP}))^T \mathbf{U}_s^e \det(\mathbf{J}(\tilde{\xi}_{GP})), \end{aligned} \quad (\text{C.4})$$

where  $w_{GP}$  and  $\tilde{\xi}_{GP}$  are the Gauss weights and points, respectively.

#### Appendix D. Design of $P(\xi)$ to reproduce B-spline and NURBS basis functions

In C-IGA, we need to design  $P(\xi)$  in the convolution patch functions Eq. (4) in order to reproduce the basis functions of geometric mapping. Especially for IGA mapping or CAD model as geometric mapping, B-spline and NURBS basis functions need to be reproduced accordingly. A straightforward way is to choose B-spline or NURBS basis functions  $R_l(\xi)$  as  $P(\xi)$  directly. An alternative way is to use polynomials or polynomials over a weighting function (denominator of NURBS basis functions) instead, which is convenient to be rescaled in the parent domain.

- The first way is to use B-spline or NURBS basis functions  $R_l(\xi)$  as  $P(\xi)$  directly:

$$\mathbf{P}(\xi) = [R_1(\xi), R_2(\xi), \dots]. \quad (\text{D.1})$$

- The second way is to use polynomials or polynomials over a weighting function instead. We notice that B-spline basis functions are polynomials in their elements. Thus in one element, we can use polynomials instead, e.g., 1D  $p$ th order polynomials for  $p$ th order B-spline basis functions,

$$\mathbf{P}(\xi) = [1, \xi, \xi^2, \dots, \xi^p], \quad (\text{D.2})$$

or in 2D,

$$\mathbf{P}(\xi) = [1, \xi, \eta, \xi^2, \eta^2, \dots, \xi^p \eta^p]. \quad (\text{D.3})$$

It is important to note that the number of basis functions for C-IGA is determined by the tensor product while that of C-HiDeNN is from the complete polynomials of Pascal's triangle. For example, when  $p = 2$  and the problem dimension is 2D, the basis functions for C-HiDeNN are  $\mathbf{P}(\xi) = [1, \xi, \eta, \xi^2, \xi\eta, \eta^2]$  while those of C-IGA are  $\mathbf{P}(\xi) = [1, \xi, \eta, \xi^2, \eta^2, \xi^2\eta, \xi\eta^2, \xi^2\eta^2]$ . Therefore, C-IGA requires more polynomials in general.

As for NURBS basis functions, they are represented by B-spline functions over a weighting function  $W(\xi)$ . Thus we can use polynomials over the same weighting function  $W(\xi)$  instead, e.g., 2D rational form for  $p$ th order NURBS basis functions is

$$\mathbf{P}(\xi, \eta) = \frac{[1, \xi, \eta, \xi^2, \eta^2, \dots, \xi^p \eta^p]}{W(\xi, \eta)}. \quad (\text{D.4})$$

#### Appendix E. Derivatives with respect to dilation parameter $a$

In this Subsection, we study the derivatives of variational energy with respect to dilation parameter  $a$  for  $a$ -adaptivity or seeking for optimal global  $a$  by optimization. Although in an isoparametric framework, the first mapping is precisely given geometric mapping, which is independent of  $a$ . The second mapping Eq. (15) is also independent of  $a$ . Thus we only need to consider the derivatives of interpolants with respect to  $a$  in the variational energy Eq. (16), i.e.,

$$\frac{\partial \mathbf{u}^h(\xi; a)}{\partial a} = \sum_{i \in A^e} \mathcal{N}_i(\xi) \sum_{j \in A_{a,i}^{\xi_i}} \frac{\partial \mathcal{W}_{a,j}^{\xi_i}(\xi; a)}{\partial a} \mathbf{u}_j. \quad (\text{E.1})$$

Note that the coefficient matrices  $\mathbf{A}$  and  $\mathbf{K}$  in the convolution patch functions  $\mathcal{W}_{a,i}^{\xi_i}(\xi; a)$  (see Eqs. (B.1) and (B.2)) are also function of  $a$ :

$$\mathbf{A}(a) = \mathbf{R}^{-1}(a)(\mathbf{I}_{ns \times ns} - \mathbf{Q}\mathbf{K}(a)), \mathbf{K} = (\mathbf{Q}^T \mathbf{R}^{-1}(a) \mathbf{Q})^{-1} \mathbf{Q}^T \mathbf{R}^{-1}(a). \quad (\text{E.2})$$

Thus the derivatives of convolution patch functions  $\frac{\partial \mathcal{W}_{a,j}^{\xi_i}(\xi; a)}{\partial a}$  written in the matrix form is

$$\begin{aligned} \frac{\partial \mathcal{W}_{a,s}^{\xi_i}(\xi; a)}{\partial a} &= \frac{\partial \Psi_a(\xi)}{\partial a} \mathbf{R}^{-1}(a)(\mathbf{I}_{ns \times ns} - \mathbf{Q}\mathbf{K}(a)) + \Psi_a(\xi) \frac{\partial \mathbf{R}^{-1}(a)}{\partial a}(\mathbf{I}_{ns \times ns} - \mathbf{Q}\mathbf{K}(a)) \\ &\quad - \Psi_a(\xi) \mathbf{R}^{-1}(a) \mathbf{Q} \frac{\partial \mathbf{K}(a)}{\partial a} + \mathbf{P}(\xi) \frac{\partial \mathbf{K}(a)}{\partial a} \\ &= \frac{\partial \Psi_a(\xi)}{\partial a} \mathbf{A} - \Psi_a(\xi) \mathbf{R}^{-1}(a) \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{A} - \Psi_a(\xi) \mathbf{R}^{-1}(a) \mathbf{Q} \frac{\partial \mathbf{K}(a)}{\partial a} + \mathbf{P}(\xi) \frac{\partial \mathbf{K}(a)}{\partial a}. \end{aligned} \quad (\text{E.3})$$

Here, the derivatives of coefficient matrix  $\mathbf{K}$  with respect to  $a$  is

$$\begin{aligned} \frac{\partial \mathbf{K}(a)}{\partial a} &= \frac{\partial (\mathbf{Q}^T \mathbf{R}^{-1}(a) \mathbf{Q})^{-1}}{\partial a} \mathbf{Q}^T \mathbf{R}^{-1}(a) + (\mathbf{Q}^T \mathbf{R}^{-1}(a) \mathbf{Q})^{-1} \mathbf{Q}^T \frac{\partial \mathbf{R}^{-1}(a)}{\partial a} \\ &= \mathbf{K} \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{R}^{-1}(a) \mathbf{Q} \mathbf{K} - \mathbf{K} \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{R}^{-1}(a) \\ &= -\mathbf{K} \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{A}. \end{aligned} \quad (\text{E.4})$$

Substituting Eq. (E.4) into Eq. (E.3), we obtain a simple form

$$\begin{aligned} \frac{\partial \mathcal{W}_{a,s}^{\xi_i}(\xi; a)}{\partial a} &= \frac{\partial \Psi_a(\xi)}{\partial a} \mathbf{A} - \Psi_a(\xi) \mathbf{R}^{-1}(a) \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{A} \\ &\quad + \Psi_a(\xi) \mathbf{R}^{-1}(a) \mathbf{Q} \mathbf{K} \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{A} - \mathbf{P}(\xi) \mathbf{K} \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{A} \\ &= \frac{\partial \Psi_a(\xi)}{\partial a} \mathbf{A} - \mathcal{W}_{a,s}^{\xi_i}(\xi; a) \frac{\partial \mathbf{R}(a)}{\partial a} \mathbf{A}. \end{aligned} \quad (\text{E.5})$$

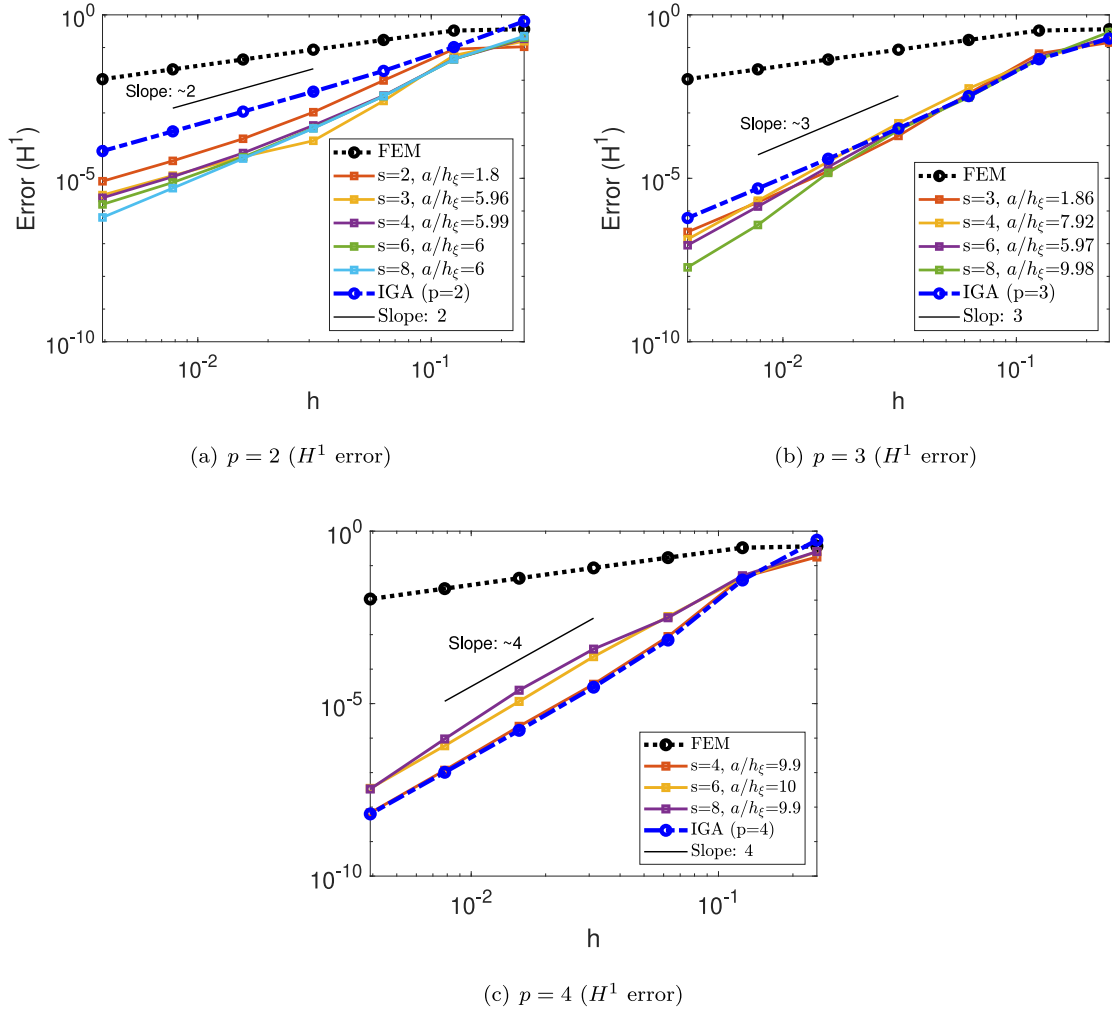
## Appendix F. Supplementary results for 1D model: $H^1$ -norm error convergence rates and condition number of stiffness matrix

The  $H^1$ -norm error convergence rates of C-IGA for 1D model, and comparisons with linear FEM, IGA are plotted in Fig. F.32. As expected, the convergence rates of C-IGA are  $p$  or even larger than  $p$  if satisfying  $p$ th order reproducing conditions. This is consistent with our theoretical analysis.

The condition number of stiffness matrix for 1D model are plotted in Fig. F.33. Among IGA ( $p = 2, 3, 4$ ), FEM ( $p = 1, 2$ ) and C-IGA ( $p = 2, 3, 4, s = p$ ), IGA gives the smallest condition number, and quadratic FEM gives the largest one at the same mesh, as DoFs of quadratic FEM are almost twice those of other methods. The condition numbers of IGA, linear FEM and C-IGA have the same order of magnitude. The lines of different orders of C-IGA almost coincide, so we only plot one line to represent them. We do the same thing for IGA. The condition number of C-IGA is close to that of linear FEM for the same mesh. In addition, it is noticed that all condition numbers are proportional to the square of element size no matter IGA ( $p = 2, 3, 4$ ), FEM ( $p = 1, 2$ ) and C-IGA ( $p = 2, 3, 4$ ). The accuracy vs. condition number plot is shown in Fig. F.33(b). When considering accuracy, C-IGA ( $p = 2$ ) gives the smallest condition number at the same level of error (e.g.,  $10^{-4}$  energy-norm error) for fine mesh compared to FEM ( $p = 1, 2$ ) and IGA ( $p = 2$ ). C-IGA with larger patch size  $s$  has a similar condition number, so we only present results for  $s = 2$ . Remark that the condition number of C-IGA also varies with  $s$  and  $a$  altered like accuracy. The condition number shown in the figure might not be the smallest. The effect of  $s$  and  $a$  on condition number would be studied in the future work.

## Appendix G. Degeneration of C-IGA to IGA

C-IGA can be degenerated to IGA by a special choice of nodes and convolution patch function. Note that knots  $\{\xi_i\}$  and basis functions  $\{N_{j,p}\}$  are not in a one-to-one correspondence in IGA, and the DoFs increase as the order elevates. Therefore, anchors  $\{t_j\}$  are used to locate basis functions in the parametric domain [35]. For the  $j$ th basis



**Fig. F.32.** The  $H^1$ -norm error convergence rates against the mesh refinement for different methods and different sets of parameters for C-IGA with cubic spline radial function: (a)  $p = 2$  ( $H^1$ -norm error); (b)  $p = 3$  ( $H^1$ -norm error); (c)  $p = 4$  ( $H^1$ -norm error).

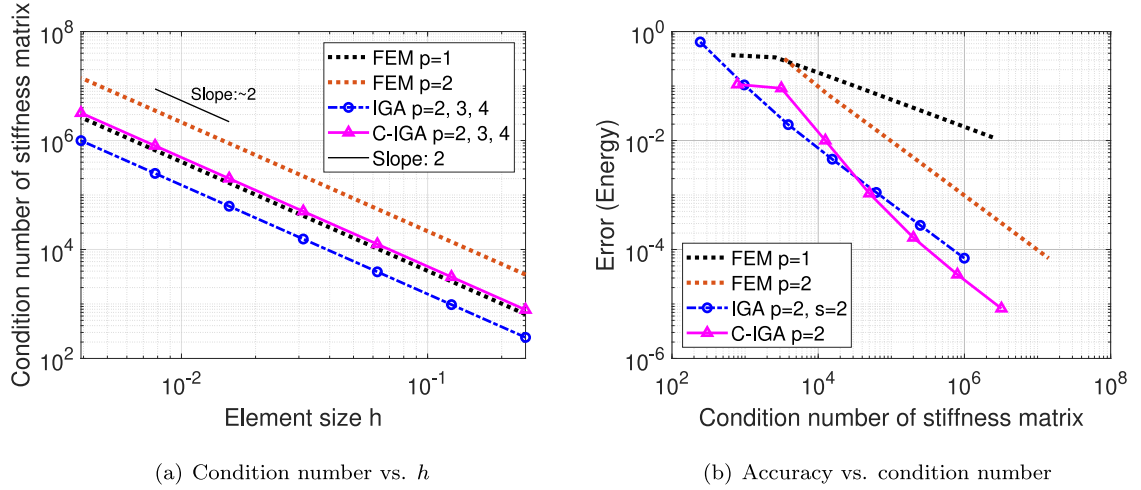
function  $N_{j,p}$ , the corresponding anchor, denoted  $t_j$ , is given by

$$t_j = \begin{cases} \xi_{j+(p+1)/2}, & \text{if } p \text{ is odd,} \\ (\xi_{j+p/2} + \xi_{j+p/2+1})/2, & \text{if } p \text{ is even.} \end{cases} \quad (\text{G.1})$$

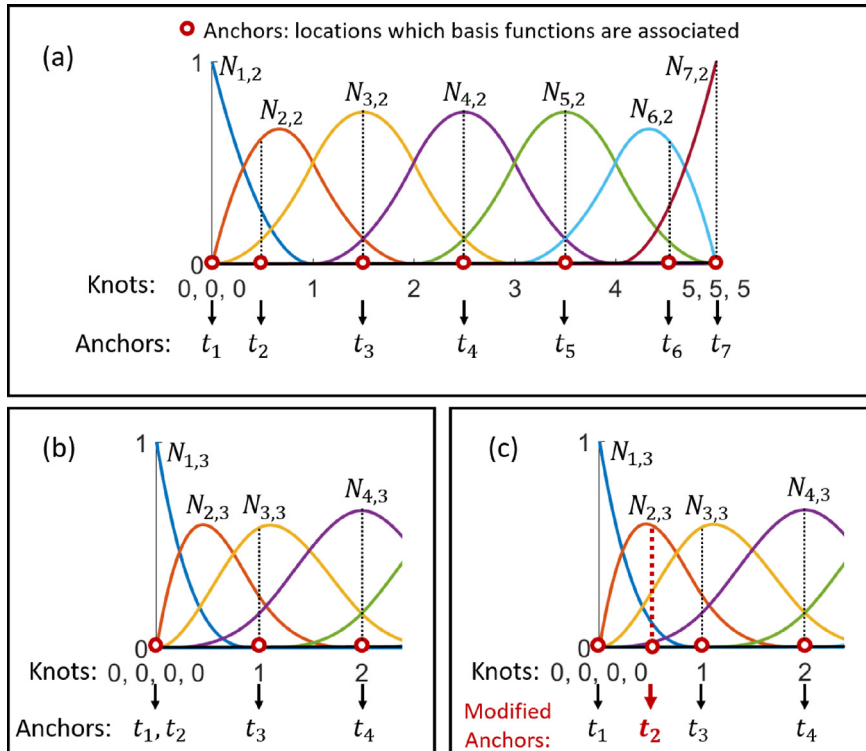
The anchors are illustrated in Fig. G.34.

If we use these anchors as nodes to construct the C-IGA mesh in the parametric domain, C-IGA will have the same DoFs as IGA. We note that repeating anchors appear when knots are repeated and  $p > 2$ , especially at the boundary of the computational domain, as illustrated in Fig. G.34(b, c). We modify anchors by redistributing their positions locally in the region between the two adjacent non-repeating anchors. As shown in Fig. G.34(c), the repeating anchors  $t_1$  and  $t_2$  are redistributed uniformly between  $t_1$  and  $t_3$ , i.e.,  $t_2 = (t_1 + t_3)/2$ . After this modification, we adopt anchors as nodes and construct C-IGA interpolation while using IGA basis functions in the convolution patch function, given by

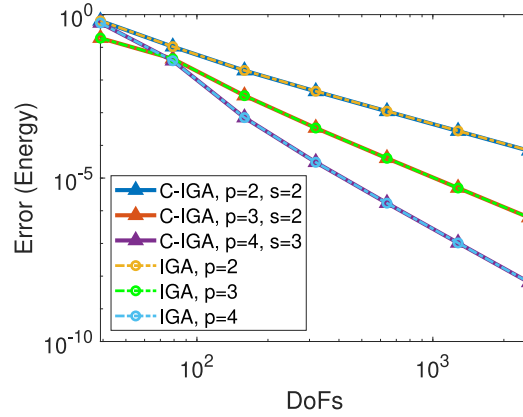
$$u^{C-IGA}(\xi) = \sum_i N_i(\xi; t_i^*) \sum_{j \in A_s^{t_i}} N_{j,p}(\xi) d_j. \quad (\text{G.2})$$



**Fig. F.33.** Condition number of stiffness matrix for different meshes. (a) Condition number vs.  $h$ ; (b) accuracy vs. condition number. The lines of different orders of C-IGA (here,  $s = p$  is taken) almost coincide, so we only plot one line to represent them. We do the same thing for IGA. Among C-IGA results, the condition number of C-IGA ( $p = 4$ ) is the largest, and that of C-IGA ( $p = 2$ ) is the smallest. Remark that the condition number of C-IGA also varies with  $s$  and  $a$  altered. The condition number shown in the figure might not be the smallest. The effect of  $s$  and  $a$  on condition number would be studied in the future work.



**Fig. G.34.** Illustration for anchors in IGA. (a) Quadratic basis functions ( $p = 2$ ) and anchors for open knot vector  $\Xi = \{0, 0, 0, 1, 2, 3, 4, 5, 5, 5\}$ . (b) Cubic basis functions ( $p = 3$ ) and anchors for open knot vector  $\Xi = \{0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 5, 5\}$ . Anchors  $t_1, t_2$  locates at the same parametric position. (c) Modified anchors by redistributing repeating anchors.  $t_2$  is moved to the middle point of  $t_1$  and  $t_3$ .



**Fig. G.35.** Numerical results of C-IGA (Eq. (G.2)) and IGA. C-IGA results completely coincide with IGA results.

**Table H.7**

1D B-spline basis functions in two directions.

$i$	$i = 1$	$i = 2$	$i = 3$
$M_{j,p=2}(\xi)$	$(1 - \xi)^2$	$2\xi(1 - \xi)$	$\xi^2$
$N_{i,p=2}(\eta)$	$(1 - \eta)^2$	$2\eta(1 - \eta)$	$\eta^2$

$\mathbf{t}_i^*$  represents the set of anchors associated with FEM shape function  $N_i$ , such as  $\mathbf{t}_i^* = [t_{i-1}, t_i, t_{i+1}]$  in 1D linear case.  $A_s^{t_i}$  called nodal patch is the set of nodes (anchors) in a patch domain of node (anchor)  $t_i$ .  $\mathbf{d}_j$  is the nodal coefficient for  $t_j$ . The new interpolation does not satisfy the Kronecker delta property. In fact, it is precisely IGA interpolations, while written in the C-HiDeNN form, so the increase of patch size  $s$  does not influence the performance of interpolations. The only requirement is that patch size  $s$  should be larger than the support size of IGA basis functions.

Then we use the new C-IGA Eq. (G.2) and IGA to solve 1D problems in Section 5.1. The results plotted in Fig. G.35 clearly show that C-IGA results completely coincide with IGA results, which verifies the equivalence of C-IGA (Eq. (G.2)) and IGA. Thus we can conclude that C-IGA can degenerate to IGA by using modified anchors as nodes and IGA basis functions as convolution patch functions.

## Appendix H. Geometric data of 2D model: A quarter-ring

The ring illustrated in Fig. 16 has an inner radius 10, and an outer radius 20. The coarse mesh uses rational quadratic basis functions, i.e., NURBS, in two directions. The NURBS basis functions  $R_{i,j}(\xi, \eta)$  is defined by

$$R_{i,j}(\xi, \eta) = \frac{w_{ij} M_{i,p=2}(\xi) N_{j,p=2}(\eta)}{W(\xi, \eta)}, \quad i, j = 1, 2, 3, \quad (\text{H.1})$$

with

$$W(\xi, \eta) = \sum_{k=1}^3 \sum_{l=1}^3 w_{kl} M_{k,p=2}(\xi) N_{l,p=2}(\eta), \quad (\text{H.2})$$

where  $M_{i,p=2}(\xi)$ ,  $N_{j,p=2}(\eta)$  are 1D quadratic B-spline basis functions given in Table H.7, and  $w_{ij}$  are weights listed in Table H.8.  $W(\xi, \eta)$  is the so-called weighting function. In this model, with above weights and B-spline basis, it is

$$W(\xi, \eta) = 1 + (\sqrt{2} - 2)\xi(1 - \xi). \quad (\text{H.3})$$

We also use the weighting function to construct  $\mathbf{P}(\xi)$  in C-IGA. Details refer to Appendix D. In addition to NURBS basis functions, control points  $\mathbf{B}_{i,j}$  in the geometric mapping Eq. (49) are listed in Table H.9. Remark that the basis in  $\eta$ -direction, the radial direction, can be linear. Here, we use quadratic basis in both two directions for consistency.



**Table H.8**

Weights.

$w_{ij}$	$i = 1$	$i = 2$	$i = 3$
$j = 1$	1	$\sqrt{2}/2$	1
$j = 2$	1	$\sqrt{2}/2$	1
$j = 3$	1	$\sqrt{2}/2$	1

**Table H.9**

Control points.

$B_{i,j}$	$i = 1$	$i = 2$	$i = 3$
$j = 1$	(0, 10)	(10, 10)	(10, 0)
$j = 2$	(0, 15)	(15, 15)	(15, 0)
$j = 3$	(0, 20)	(20, 20)	(20, 0)

## Appendix I. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.cma.2023.116356>.

## References

- [1] A. Karpathy, Software 2.0, 2017, URL <https://karpathy.medium.com/software-2-0-a64152b37c35>.
- [2] E. Meijer, Behind every great deep learning framework is an even greater programming languages concept (keynote), in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, p. 1.
- [3] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.
- [4] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, J.Z. Kolter, End-to-end differentiable physics for learning and control, Adv. Neural Inf. Process. Syst. 31 (2018).
- [5] S. Engell, Feedback control for optimal process operation, J. Process Control 17 (3) (2007) 203–219.
- [6] F. Chinesta, A. Leygue, F. Bordeu, J.V. Aguado, E. Cueto, D. González, I. Alfaro, A. Ammar, A. Huerta, PGD-based computational vademecum for efficient design, optimization and control, Arch. Comput. Methods Eng. 20 (1) (2013) 31–59.
- [7] Y. Lu, K.K. Jones, Z. Gan, W.K. Liu, Adaptive hyper reduction for additive manufacturing thermal fluid analysis, Comput. Methods Appl. Mech. Engrg. 372 (2020) 113312.
- [8] S. Saha, O.L. Kafka, Y. Lu, C. Yu, W.K. Liu, Microscale structure to property prediction for additively manufactured IN625 through advanced material model parameter identification, Integr. Mater. Manuf. Innov. 10 (2) (2021) 142–156.
- [9] 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.
- [10] L. Zhang, Y. Lu, S. Tang, W.K. Liu, HiDeNN-TD: Reduced-order hierarchical deep learning neural networks, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114414.
- [11] Y. Lu, H. Li, L. Zhang, C. Park, S. Mojmader, S. Knapik, Z. Sang, S. Tang, D.W. Apley, G.J. Wagner, et al., Convolution Hierarchical Deep-learning Neural Networks (C-HiDeNN): Finite elements, isogeometric analysis, tensor decomposition, and beyond, Comput. Mech. (2023) 1–30.
- [12] C. Park, Y. Lu, S. Saha, T. Xue, J. Guo, S. Mojmader, D.W. Apley, G.J. Wagner, W.K. Liu, Convolution hierarchical deep-learning neural network (C-HiDeNN) with graphics processing unit (GPU) acceleration, Comput. Mech. (2023) 1–27.
- [13] H. Li, S. Knapik, Y. Li, C. Park, J. Guo, S. Mojmader, Y. Lu, W. Chen, D.W. Apley, W.K. Liu, Convolution hierarchical deep-learning neural network tensor decomposition (C-HiDeNN-TD) for high-resolution topology optimization, Comput. Mech. (2023) 1–20.
- [14] O. Huang, S. Saha, J. Guo, W.K. Liu, An introduction to kernel and operator learning methods for homogenization by self-consistent clustering analysis, Comput. Mech. (2023) 1–25.
- [15] S. Saha, C. Park, S. Knapik, J. Guo, O. Huang, W.K. Liu, Deep learning discrete calculus (DLDC): A family of discrete numerical methods by universal approximation for STEM education to frontier research, Comput. Mech. (2023) 1–21.
- [16] W.K. Liu, S. Jun, Y.F. Zhang, Reproducing kernel particle methods, Internat. J. Numer. Methods Fluids 20 (8–9) (1995) 1081–1106.
- [17] W.K. Liu, S. Jun, S. Li, J. Adee, T. Belytschko, Reproducing kernel particle methods for structural dynamics, Internat. J. Numer. Methods Engrg. 38 (10) (1995) 1655–1679.
- [18] J.-S. Chen, M. Hillman, S.-W. Chi, Meshfree methods: Progress made after 20 years, J. Eng. Mech. 143 (4) (2017) 04017001.
- [19] T. Belytschko, Y.Y. Lu, L. Gu, Element-free Galerkin methods, Internat. J. Numer. Methods Engrg. 37 (2) (1994) 229–256.
- [20] B. Nayroles, G. Touzot, P. Villon, Generalizing the finite element method: Diffuse approximation and diffuse elements, Comput. Mech. 10 (5) (1992) 307–318.
- [21] M.B. Liu, G. Liu, Smoothed particle hydrodynamics (SPH): An overview and recent developments, Arch. Comput. Methods Eng. 17 (2010) 25–76.

- [22] W.-K. Liu, S. Li, T. Belytschko, Moving least-square reproducing kernel methods (I) methodology and convergence, *Comput. Methods Appl. Mech. Engrg.* 143 (1–2) (1997) 113–154.
- [23] W.K. Liu, Y. Chen, R.A. Uras, C.T. Chang, Generalized multiple scale reproducing kernel particle methods, *Comput. Methods Appl. Mech. Engrg.* 139 (1–4) (1996) 91–157.
- [24] S. Li, W.K. Liu, Moving least-square reproducing kernel method Part II: Fourier analysis, *Comput. Methods Appl. Mech. Engrg.* 139 (1–4) (1996) 159–193.
- [25] D. Wang, Y. Sun, An efficient Galerkin meshfree formulation for shear deformable beam under finite deformation, *Theor. Appl. Mech. Lett.* 1 (5) (2011) 051010.
- [26] Y.E. Wang, G.-Y. Wei, D. Brooks, Benchmarking TPU, GPU, and CPU platforms for deep learning, 2019, arXiv preprint [arXiv:1907.10701](https://arxiv.org/abs/1907.10701).
- [27] J. Nguyen, J. Lee, I. Hulseman, New video: What runs ChatGPT? 2023, URL <https://developer.nvidia.com/blog/new-video-what-runs-chatgpt/>.
- [28] Y. Liu, C. Park, Y. Lu, S. Mojmader, W.K. Liu, D. Qian, HiDeNN-FEM: A seamless machine learning approach to nonlinear finite element analysis, *Comput. Mech.* (2023) 1–22.
- [29] 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.
- [30] Y. Lu, H. Li, S. Saha, S. Mojmader, A. Al Amin, D. Suarez, Y. Liu, D. Qian, W. Kam Liu, Reduced order machine learning finite element methods: Concept, implementation, and future applications, *CMES Comput. Model. Eng. Sci.* 129 (1) (2021).
- [31] J. Plocher, A. Panesar, Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures, *Mater. Des.* 183 (2019) 108164, <http://dx.doi.org/10.1016/j.matdes.2019.108164>, URL <https://www.sciencedirect.com/science/article/pii/S0264127519306021>.
- [32] J.-E. Kim, K. Park, Multiscale topology optimization combining density-based optimization and lattice enhancement for additive manufacturing, *Int. J. Precis. Eng. Manuf. Green Technol.* 8 (4) (2021) 1197–1208, <http://dx.doi.org/10.1007/s40684-020-00289-1>, URL <https://link.springer.com/10.1007/s40684-020-00289-1>.
- [33] Y. Lu, L. Tong, Concurrent multiscale topology optimization of metamaterials for mechanical cloak, *Comput. Methods Appl. Mech. Engrg.* 409 (2023) 115966, <http://dx.doi.org/10.1016/j.cma.2023.115966>, URL <https://linkinghub.elsevier.com/retrieve/pii/S0045782523000890>.
- [34] T.J.R. 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.
- [35] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, T.W. Sederberg, Isogeometric analysis using T-splines, *Comput. Methods Appl. Mech. Engrg.* 199 (5–8) (2010) 229–263.
- [36] L. De Lorenzis, P. Wriggers, T.J.R. Hughes, Isogeometric contact: A review, *GAMM-Mitt.* 37 (1) (2014) 85–123.
- [37] T. Belytschko, W.K. Liu, B. Moran, K. Elkhodary, *Nonlinear Finite Elements for Continua and Structures*, John Wiley & Sons, 2014.
- [38] G.J. Wagner, W.K. Liu, Hierarchical enrichment for bridging scales and mesh-free boundary conditions, *Internat. J. Numer. Methods Engrg.* 50 (3) (2001) 507–524.
- [39] M.A. Bessa, J.T. Foster, T. Belytschko, W.K. Liu, A meshfree unification: Reproducing kernel peridynamics, *Comput. Mech.* 53 (6) (2014) 1251–1264.
- [40] S. Li, W.K. Liu, *Meshfree Particle Methods*, Springer Science & Business Media, 2007.
- [41] H. Wendland, Meshless Galerkin methods using radial basis functions, *Math. Comp.* 68 (228) (1999) 1521–1531.
- [42] J.G. Wang, G. Liu, A point interpolation meshless method based on radial basis functions, *Internat. J. Numer. Methods Engrg.* 54 (11) (2002) 1623–1648.
- [43] W.K. Liu, W. Han, H. Lu, S. Li, J. Cao, Reproducing kernel element method. Part I: Theoretical formulation, *Comput. Methods Appl. Mech. Engrg.* 193 (12–14) (2004) 933–951.
- [44] W. Wang, Y. Zhang, L. Liu, T.J.R. Hughes, Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology, *Comput. Aided Des.* 45 (2) (2013) 351–360.
- [45] X. Wei, X. Li, K. Qian, T.J. Hughes, Y.J. Zhang, H. Casquero, Analysis-suitable unstructured T-splines: Multiple extraordinary points per face, *Comput. Methods Appl. Mech. Engrg.* 391 (2022) 114494.
- [46] M. AI, PyTorch, URL <https://pytorch.org/>.
- [47] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: Composable transformations of Python+NumPy programs, 2018, URL <http://github.com/google/jax>.
- [48] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, IEEE, 1995, pp. 1942–1948.
- [49] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [50] T. Xue, S. Liao, Z. Gan, C. Park, X. Xie, W.K. Liu, J. Cao, JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science, *Comput. Phys. Comm.* (2023) 108802.
- [51] V.P. Nguyen, C. Anitescu, S.P. Bordas, T. Rabczuk, Isogeometric analysis: An overview and computer implementation aspects, *Math. Comput. Simulation* 117 (2015) 89–116.
- [52] M. Chasapi, L. Mester, B. Simeon, S. Klinkel, Isogeometric analysis of 3D solids in boundary representation for problems in nonlinear solid mechanics and structural dynamics, *Internat. J. Numer. Methods Engrg.* 123 (5) (2022) 1228–1252.
- [53] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, A.-V. Vuong, Swept volume parameterization for isogeometric analysis, in: *Mathematics of Surfaces XIII: 13th IMA International Conference York, UK, September 7-9, 2009 Proceedings* 13, Springer, 2009, pp. 19–44.

- [54] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications, *Comput. Aided Des.* 45 (2) (2013) 395–404.
- [55] R. Sevilla, S. Fernández-Méndez, A. Huerta, NURBS-enhanced finite element method (NEFEM), *Internat. J. Numer. Methods Engrg.* 76 (1) (2008) 56–83.
- [56] R. Sevilla, S. Fernández-Méndez, A. Huerta, 3D NURBS-enhanced finite element method (NEFEM), *Internat. J. Numer. Methods Engrg.* 88 (2) (2011) 103–125.
- [57] L. Liu, Y. Zhang, T.J. Hughes, M.A. Scott, T.W. Sederberg, Volumetric T-spline construction using Boolean operations, *Eng. Comput.* 30 (2014) 425–439.
- [58] H. Al Akhras, T. Elguedj, A. Gravouil, M. Rochette, Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models, *Comput. Methods Appl. Mech. Engrg.* 307 (2016) 256–274.
- [59] H. Al Akhras, T. Elguedj, A. Gravouil, M. Rochette, Towards an automatic isogeometric analysis suitable trivariate models generation—Application to geometric parametric analysis, *Comput. Methods Appl. Mech. Engrg.* 316 (2017) 623–645.
- [60] NVIDIA, What is accelerated computing? URL <https://blogs.nvidia.com/blog/2021/09/01/what-is-accelerated-computing/>.
- [61] NVIDIA, NVIDIA announces DGX GH200 AI supercomputer, URL <https://nvidianews.nvidia.com/news/nvidia-announces-dgx-gh200-ai-supercomputer>.
- [62] Forbes, Tesla's biggest news at AI day was the dojo supercomputer, not the optimus robot, URL <https://www.forbes.com/sites/jamesmorris/2022/10/06/teslas-biggest-news-at-ai-day-was-the-dojo-supercomputer-not-the-optimus-robot/?sh=273d22fb80bd>.
- [63] J. Hampton, Google claims Its TPU v4 outperforms Nvidia A100, URL <https://www.datanami.com/2023/04/05/google-claims-its-tpu-v4-outperforms-nvidia-a100/>.
- [64] 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.
- [65] M.J. Kazemzadeh-Parsi, A. Ammar, J.L. Duval, F. Chinesta, Enhanced parametric shape descriptions in PGD-based space separated representations, *Adv. Model. Simul. Eng. Sci.* 8 (2021) 1–28.
- [66] 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.
- [67] X. Zhang, F. Wang, L.M. Keer, Influence of surface modification on the microstructure and thermo-mechanical properties of bamboo fibers, *Materials* 8 (10) (2015) 6597–6608.
- [68] S. Mojumder, H. Li, Y. Lu, Y. Li, T. Abbott, S. Knapik, W. Chen, G. Wagner, W.K. Liu, Mechanistic nested topology design theory for multifunctional materials systems, 2023, in preparation.
- [69] H. Li, Mechanistic Concurrent Nested Topology Design Theory for Advanced Materials Systems (Ph.D. thesis), Northwestern University, 2023.