Engineering with Computers (2026) 42:116
https://doi.org/10.1007/s00366-026-02351-z

ORIGINAL ARTICLE

Multi-patch neural solver with isogeometric mappings for partial differential equations on computer-aided design domains

Moritz von Tresckow[1] · Ion Gabriel Ion[2] · Dimitrios Loukrezis[3]

Received: 29 September 2025 / Accepted: 9 May 2026

Abstract
This work develops a computational framework that combines physics-informed neural networks with multi-patch isogeometric mappings to solve partial differential equations on complex computer-aided design geometries. The method utilizes patch-local neural networks that operate on the parametric space of each patch, i.e. the reference domain of the local isogeometric map. A custom output layer enables the strong imposition of Dirichlet boundary conditions. Solution conformity across interfaces between non-uniform rational B-spline patches is enforced using dedicated interface neural networks. Training is performed using the variational framework by minimizing the energy functional derived after the weak form of the partial differential equation. The effectiveness of the suggested method is demonstrated on a benchmark problem and on two highly non-trivial engineering applications. The results show excellent agreement to reference solutions obtained with high-fidelity finite element solvers, thus highlighting the potential of the suggested neural solver to tackle complex engineering problems given the corresponding computer-aided design models.

Keywords Computer-aided design · Isogeometric analysis · Neural solver · Partial differential equations · Physics-informed neural networks

1 Introduction

1.1 Motivation

In recent years, physics-informed neural networks (PINNs) [1] have emerged as a possible mesh-free alternative to traditional, mesh-based numerical methods for the solution of partial differential equations (PDEs). Within the PINN framework, neural networks (NNs) are trained to learn a PDE solution by minimizing a loss function which incorporates the boundary value problem (BVP) that describes the underlying physics of the system under investigation [2, 3]. It must be noted that methods based on very similar ideas were developed prior to the seminal PINN paper, such as the deep Ritz and deep Galerkin methods [4, 5]. However, PINN has been established as a blanket term for most related approaches and is used as such within this paper as well.

The PINN framework and its numerous variants [6–10] have already found applications in several fields of computational science and engineering, as in computational electromagnetics [11–14], fluid dynamics [15–18], and mechanics [19–22], to name only a few indicative application areas. Nonetheless, due to a number of weaknesses, PINNs cannot yet be regarded as competitive to traditional numerical solvers [23, 24]. This work focuses on the severe difficulties that PINNs face when applied to complex domains, in particular shapes and geometry parametrizations stemming from computer-aided design (CAD). In turn, these difficulties

Moritz von Tresckow and Ion Gabriel Ion contributed equally to this work.

✉ Moritz von Tresckow
moritz.von_tresckow@tu-darmstadt.de

✉ Dimitrios Loukrezis
d.loukrezis@cwi.nl

Ion Gabriel Ion
igi@terraquantum.swiss

[1] Institute for Accelerator Science and Electromagnetic Fields, Technische Universität Darmstadt, Schlossgartenstr. 8, 64289 Darmstadt, Germany

[2] Terra Quantum AG, Kornhausstr. 25, St., 9000 Gallen, Switzerland

[3] Scientific Computing, Centrum Wiskunde & Informatica, Science Park 123, 1098 XG Amsterdam, The Netherlands

Published online: 02 June 2026

hinder the application of PINNs to real-world engineering problems, where complex, multi-patch CAD geometries are ubiquitous.

In this context, three main challenges remain largely unaddressed. The first challenge concerns the enforcement of solution conformity across complex, multi-patch domains. Solution continuity in particular is crucial when dealing with different materials across domain interfaces. The second challenge concerns the imposition of Dirichlet boundary conditions (BCs). The standard PINN framework and most related approaches use a soft-constraints approach, such that the PINN learns the BCs during training. However, in many settings, the BCs must be strongly conformed to. The third challenge concerns handling geometry parts with significant variations in size and shape. Such differences make data normalization difficult, especially under physical conformity requirements.

1.2 Contribution

This work aims to enable the application of PINNs to CAD geometries by addressing the three aforementioned limitations. Of particular interest are complex, multi-patch CAD geometries, which are rarely addressed in the relevant literature. To that end, we introduce a novel neural solver which combines PINNs with tools stemming from isogeometric analysis (IGA) [25–27]. More specifically, the isogeometric-mapped neural solver developed in this work is rooted in the nite element tearing and interconnecting (FETI) method [28] and its extension to IGA [29], as well as in mortaring methods [30–32]. Therein, the PINN framework is integrated.

Our approach consists of developing NN-based ansatz functions that approximate the PDE solution by operating on the parametric space of each patch (i.e., the reference domain for the patch-local isogeometric mapping). The solution on the physical domain is obtained by using the pushforward map, i.e., the geometrical mapping from the reference to the physical domain as given by the non-uniform rational B-spline (NURBS)-based parametrization of the geometry, similar to conventional IGA. The suggested ansatz functions also satisfy Dirichlet BCs where necessary, ensure differentiability within the subdomains (patches) of the physical geometry, and ensure solution continuity across patches.

One set of ansatz functions is used to approximate the PDE solution within the reference domain of each individual patch, thereby leveraging the geometric accuracy of IGA. The output layer of these NNs is modified to strongly impose the Dirichlet BCs where necessary. Another set of ansatz functions is defined on the interfaces between neighboring patches, which appropriately connect the solutions of the individual patches, thus ensuring solution conformity across them. Training is performed by utilizing the variational PINN framework [4, 7, 33], where the loss function to be minimized is the discretized energy functional derived after the weak form of the PDE. This is a natural choice, as it aligns with the principles of IGA, which is also built upon variational formulations.

The suggested neural solver offers three distinct contributions, each connected to one of the challenges identified above. First, it ensures control over solution continuity across patch interfaces. Second, it enables the strong imposition of Dirichlet BCs. Third, by operating on the reference domain, it naturally normalizes NN inputs. The efficacy of the method is demonstrated on an analytical benchmark problem and on two real-world engineering applications. The first engineering use-case is drawn from the field of computational electromagnetics and concerns a 2D magnetostatics model of a quadrupole magnet. The second engineering use-case is drawn from the field of computational mechanics and concerns a 3D nonlinear solid mechanics model of a mechanical holder. For the latter, we employ a hyperelastic material model combined with contact BCs and introduce parametric dependencies in the solution.

1.3 Related work

Prior works have explored the combination of NNs and PINNs with IGA, e.g., for applications in solid mechanics [34–36] or electromagnetics [37, 38], including IGA variants such as isogeometric collocation methods [39] and the isogeometric boundary element method [38]. Differentiable NURBS [40] were developed as a means to integrate CAD models into deep learning frameworks. Isogeometric NNs [41] combine NURBS with NNs to predict NURBS coefficients and approximate PDE solutions for varying physical and geometric parameters. Deep NNs have been used to determine the optimum number of quadrature points for evaluating IGA stifness matrices [42]. The deep NURBS method [43] introduces an ansatz for boundary-conforming NNs. Graph NNs have been employed to learn the PDE solution at the control points of splines [44, 45], while convolutional NNs have also been used in a similar context [46–49]. However, most, if not all, of these related works concern single-patch geometries. Hence, their application for complex, multi-patch domains is not straightforward. For the same reason, solution conformity across interfaces is rarely a concern. One exception is the work of von Tresckow et al. [37], which does consider multi-patch CAD geometries and introduces a discontinuous Galerkin-based PINN formulation to weakly impose interface conditions and Dirichlet BCs. The isogeometric-mapped neural solver developed in this work is substantially different. For clarity, we note that

the term "isogeometric" in this context refers to the utilization of the exact NURBS geometry parametrization and its associated mapping operators. The solution field is approximated by neural networks rather than the same spline basis, thus distinguishing the aforementioned approaches (and ours) from classical IGA, while retaining its geometric benefits. Therefore, while departing from its traditional use in IGA literature, the term is used in alignment with the growing body of work on the development of neural solvers that utilize IGA concepts and tools [41, 44–49].

In relation to solution conformity across interfaces, prior works have combined PINNs with domain decomposition methods, using different strategies such as physics-based coupling terms [50, 51], network combinations [52], or tailored NN architectures [53]. In contrast to the present work, these methods are not applied in the context of CAD and consider relatively simple computational domains. Additionally, the present work assigns interface-specific NNs to enforce the correct solution continuity, which is a new approach, at least to the knowledge of the authors. Note that we restrict ourselves to conforming multi-patch geometries only. While this case in principle allows to merge the subdomains into a single patch representation, hence also to a single, global PINN, preserving the multi-patch topology is preferable for two main reasons. First, considering domains where patches are assigned according to material distribution (see Sect. 5.2), a single PINN would struggle to capture gradient discontinuities resulting from the different material properties [37]. Second, the spatial domain decomposition arising naturally from the multi-patch structure allows for localized PINN capacity tuning by assigning more layers/neurons to complex regions, while also helping to overcome spectral bias, i.e., the tendency of PINNs (and standard NNs) to struggle with high-frequency components or complex global domains [54, 55].

Concerning the imposition of Dirichlet BCs, most approaches based on the PINN framework enforce BCs only weakly, by incorporating them into the loss function. A few works have considered the exact imposition of Dirichlet BCs, mainly by means of dedicated distance functions [56, 57]. In contrast, our method employs a custom output layer with residual connection as part of the NN-based ansatz functions, which is responsible for the strong enforcement of Dirichlet BCs.

To overcome limitations related to large variations in the data, normalization has been attempted also in the context of PINNs and other neural solvers. Specific examples include the use of scaling factors [58], hierarchical normalization [59], dynamic normalization [60], normalization by non-dimensionalization [61], and variable linear transformation [62]. A disadvantage shared by these approaches is that the normalization procedure must be derived anew for each specific problem. Contrarily, normalization is performed naturally within our method, as it operates on the reference domain (i.e., the parametric space) of each patch.

1.4 Paper organization

The remaining of this paper is organized as follows. In Sect. 2, we recall the variational PINN framework, which concerns the use of NNs to solve energy functional minimization problems arising from variational PDE formulations. In Sect. 3, we recall some fundamental tools of CAD and IGA, to be used throughout the paper. Section 4 presents the core methodological contribution of this work, that is, the construction of NN-based ansatz functions tailored for multi-patch CAD-parametrized domains, as well as the complete neural solver. In Sect. 5, our method is first evaluated on an analytical L-shaped-domain benchmark (see Sect. 5.1.2) and then applied to two practical use-cases, namely, a 2D magnetostatics simulation of a quadrupole magnet and a 3D solid and contact mechanics simulation of a mechanical holder. For each use-case, we outline the theoretical formulation for the particular problem at hand, followed by a comprehensive presentation of the numerical results. Last, Sect. 6 summarizes the findings of this work and discusses follow-up research possibilities.

2 Variational physics-informed neural networks

We focus on elliptic BVPs, for which energy functional formulations exist. This choice is motivated by the fact that the energy formulation provides a natural optimization setting aligned with NN training. Additionally, it avoids computing higher order derivatives, which can oftentimes be problematic [5].

As solution function, we assume a scalar-field state, $u : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$, $d \in \{1, 2, 3\}$, denotes the computational domain. Typically, there is no general rule for the existence of energy functionals. An elliptic differential equation in particular admits an energy functional if and only if it is the Euler-Lagrange equation of a variational problem [63]. This means that the equation can be derived as the condition for a critical point (typically a minimum) of some functional, which usually takes the form

$$I(u) = \int_{\Omega} L(\mathrm{x}, u, \mathrm{grad}u)\ \mathrm{dx}, \tag{1}$$

where L is a function, typically referred to as the Lagrangian of the associated problem. The solution function needs to

fulfill the Euler-Lagrange equation, which for a scalar-field state u reads

$$\frac{\partial L}{\partial u} - \operatorname{div}\left(\frac{\partial L}{\partial\left(\operatorname{grad} u\right)}\right) = 0. \tag{2}$$

Solving the Euler-Lagrange equation (2) is equivalent to minimizing the energy functional (1). The system's equilibrium state, u^*, is obtained as the result of this minimization problem, such that

$$u^* = \underset{u\in\mathcal{V}}{\operatorname{argmin}}\, I(u), \tag{3}$$

where $\mathcal{V}$ is a function space containing all functions that satisfy the necessary regularity conditions and BCs of the BVP at hand. The Lagrangian exists for many elliptic PDEs, including the use-cases considered in Sect. 5. The specific energy functional minimization formulations and numerical-error definitions used in the results section are detailed in Sects. 5.1.2, 5.2, and 5.3. For the same use-cases, $\mathcal{V}$ is the Sobolev space $H^1(\Omega) = \{u \in L^2 \,:\, \operatorname{grad} u \in L^2(\Omega)\}$. Accordingly, u^* is the solution to the corresponding BVP.

Utilizing the variational PINN framework [4, 7, 33], the trial functions $u \in \mathcal{V}$ are approximated by an NN. We denote this approximation as $u_{\boldsymbol{\theta}} \approx u$, where $\boldsymbol{\theta} \in \Theta$ is a vector containing the NN's trainable parameters and Θ denotes the corresponding parameter space. Feedforward NNs are used in the following. Assuming an NN with L layers and n_l neurons in the l-th layer, $l = 1, \ldots, L$, the trainable parameter vector is given as $\boldsymbol{\theta} = \{\mathrm{W}_l, \mathrm{b}_l\}_{l=1}^{L}$, where $\mathrm{W}_l \in \mathbb{R}^{n_{l-1}\times n_l}$ and $\mathrm{b}_l \in \mathbb{R}^{n_l}$ are the l-th layer's weight matrix and bias vector, respectively. For consistency of notation, the input vector has size n_0. Substituting u with $u_{\boldsymbol{\theta}}$ in (3), the NN that best approximates the equilibrium state and, accordingly, the solution to the BVP, is the one with the parameter vector

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, I\left(u_{\boldsymbol{\theta}}\right). \tag{4}$$

Note that, in order to have a well-defined problem, the essential BCs must be embedded into the structure of the NN. That is, $u_{\boldsymbol{\theta}}$ must fulfill the essential BCs for all $\boldsymbol{\theta} \in \Theta$. This will be further discussed in Sect. 4.1.

From the above, it is obvious that the energy functional is the natural loss function to be minimized during NN training, in order to compute the optimal parameter vector $\boldsymbol{\theta}^*$. This is accomplished by first discretizing the energy functional, typically by means of numerical quadrature [4, 5, 7], and then using a gradient-based optimization method, typically a variant of stochastic gradient descent [64, 65]. The solution of the resulting nonlinear optimization problem yields the set of optimal parameters, i.e., the ones that minimize the discrete energy functional.

3 Computer-aided design and isogeometric analysis

3.1 Pushforward and pullback operators of non-uniform rational B-splines

NURBS is the central mathematical tool in CAD, which provides mathematical representations for modeling curves, surfaces, and volumes [66]. NURBS offers the flexibility and precision needed to represent both simple geometric shapes and complex freeform objects, as it can accurately model both standard analytic forms and arbitrary shapes defined by control points. The NURBS parametrization $\boldsymbol{f} : \hat{\Omega} \to \Omega$ is defined over a d-dimensional reference domain, commonly called the pre-image and here denoted as $\hat{\Omega} = [-1, 1]^d$, with its image representing the computational domain $\Omega \subset \mathbb{R}^d$ [67, 68]. The core concept of IGA is to use NURBS as basis functions for the numerical solution of PDEs, essentially integrating CAD tools into the finite element method (FEM). Utilizing the NURBS parametrization, $\boldsymbol{f}$, the solution of the PDE is first represented on the reference domain and then mapped onto the physical domain.

Similar to Sect. 2, scalar-field solutions are assumed in both the physical and the reference domain, respectively denoted as $u : \Omega \to \mathbb{R}$ and $\hat{u} : \hat{\Omega} \to \mathbb{R}$. Then, the *pushforward* operation maps the solution from the reference to the physical domain, and is defined as

$$u = \boldsymbol{f}_* \hat{u} = \hat{u} \circ \boldsymbol{f}^{-1}, \tag{5}$$

such that $\boldsymbol{f}_* : \hat{\Omega} \to \Omega$, where the symbol $\circ$ denotes function composition. The inverse *pullback* operation transforms the solution back to the reference domain, and is defined as

$$\hat{u} = \boldsymbol{f}^* u = u \circ \boldsymbol{f}, \tag{6}$$

such that $\boldsymbol{f}^* : \Omega \to \hat{\Omega}$. Essentially, the pushforward and pullback operators are induced by the forward and inverse geometric mappings between the reference and physical domains, respectively.

Using the integral transformation rule to express the energy functional as an integral over the reference domain and integrating the scalar function u in the physical domain leads to

$$\int_{\Omega} u(\mathrm{x})\mathrm{d}\mathrm{x} = \int_{\hat{\Omega}} \hat{u}(\hat{\mathrm{x}}) \left|\det\left(\partial \boldsymbol{f}\right)\right| \mathrm{d}\hat{\mathrm{x}}, \tag{7}$$

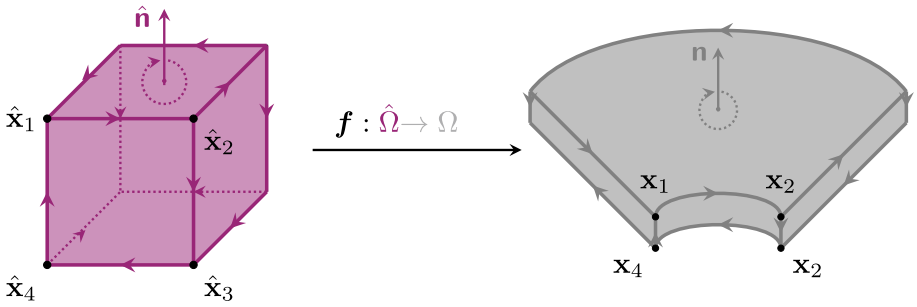


Fig. 1 Illustration of the NURBS-based geometry map from the reference to the physical domain. We have $\mathrm{x}_k = \boldsymbol{f}(\hat{\mathrm{x}}_k)$, $k = 1, \ldots, 4$. The normal $\hat{\mathrm{n}}$ is transformed to n

where $\partial \boldsymbol{f}$ denotes the Jacobian of the NURBS parametrization. The following transformations hold for the gradient and the surface normal:

$$\begin{aligned} \operatorname{grad} u &= (\partial \boldsymbol{f})^{-\top} \operatorname{grad} \hat{u}, \\ \mathrm{n} &= |\det(\partial \boldsymbol{f})| (\partial \boldsymbol{f})^{-\top} \hat{\mathrm{n}}, \end{aligned} \tag{8}$$

where $\operatorname{grad} \hat{u}$ is the gradient expressed in reference domain coordinates and $\hat{\mathrm{n}}$ is the outer normal on the boundary of the reference domain. Figure 1 presents an explanatory illustration.

Note that the pushforward and pullback operations defined in equations (5) and (6), respectively, hold for scalar fields only. However, in certain cases, they can be applied element-wise to the components of a vector field. One such example is considered in the numerical use-case presented in Sect. 5.3. Also note that the geometric mappings and the resulting transformation of the integral terms are indendent of the underlying PDE. Therefore, the framework is general enough to represent the energy functionals arising in diverse physical systems, as integration over the physical domain relies solely on the Jacobian of the mapping and the functional's dependence on the mapped field variables.

3.2 Multi-patch CAD geometries

A single NURBS has only limited ability to represent a wide variety of shapes, thus making it unsuitable for parametrizing arbitrary, non-trivial CAD-parametrized domains. Consequently, multiple NURBS parametrizations are required, each parametrizing a subdomain of the geometry. Collectively, these subdomains form the physical geometry. In the following, the term "patch" is used interchangeably for both a subdomain of the physical geometry and its corresponding NURBS parametrization.

We assume N_Ω NURBS parametrizations, denoted as $\boldsymbol{f}_i$, $i = 1, \ldots, N_\Omega$, each representing a subdomain $\Omega_i \subset \Omega$. To construct a multi-patch geometry, the individual patches are generated by mapping a reference domain $\hat{\Omega}_i = [-1, 1]^d$ onto the physical geometry, such that $\boldsymbol{f}_i\left(\hat{\Omega}_i\right) = \Omega_i$. This mapping ensures that the collection $\{\Omega_i\}_{i=1}^{N_\Omega}$ forms a domain

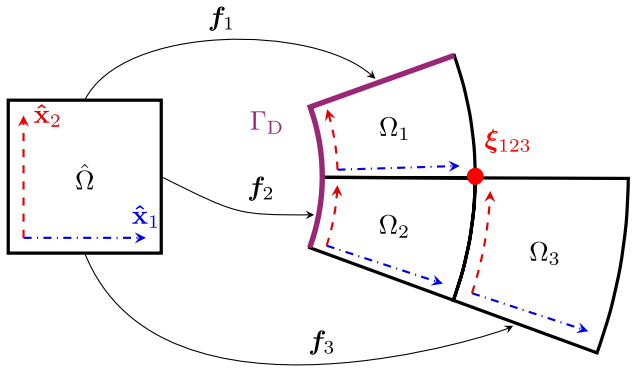


Fig. 2 Example of a two-dimensional domain represented by three patches, $\Omega_1, \Omega_2, \Omega_3$. The reference domain $\hat{\Omega}$ and the mapping of the reference coordinate system onto each patch are also shown (red and blue dotted lines). The Dirichlet boundary Γ_D is marked in purple. The point $\boldsymbol{\xi}_{123} \in \Omega$ belongs to all three patches simultaneously

decomposition, such that $\Omega = \bigcup_{i=1}^{N_\Omega} \Omega_i$. Note that, although the reference domains $\hat{\Omega} := \hat{\Omega}_1 = \cdots = \hat{\Omega}_{N_\Omega} = [-1, 1]^d$ are identical, individual indexing is necessary, because the points $\hat{\mathrm{x}} \in \hat{\Omega}_i$ are mapped differently onto Ω, depending on the patch-specific NURBS parametrization $\boldsymbol{f}_i$.

The patches are assumed to be non-overlapping and conforming at their interfaces. Non-overlapping means that the interiors of the patches are mutually disjoint. Denoting the boundary of a subdomain Ω_i as $\partial\Omega_i$, this means that $(\Omega_i \backslash \partial\Omega_i) \cap (\Omega_j \backslash \partial\Omega_j) = \emptyset$, for all $i, j = 1, \ldots, N_\Omega$. Conformity implies that, for any two adjacent patches Ω_i and Ω_j, $i \neq j$, with a nonempty $(d-1)$-dimensional interface $\Gamma_{ij} := \partial\Omega_i \cap \partial\Omega_j$, the pre-images $\hat{\Gamma}_{i\to j} = \boldsymbol{f}_i^{-1}(\Gamma_{ij})$ and $\hat{\Gamma}_{j\to i} = \boldsymbol{f}_j^{-1}(\Gamma_{ij})$ must correspond to facets of the respective reference domains $\hat{\Omega}_i$ and $\hat{\Omega}_j$. Due to the patch conformity, the tangent vectors at any point $\mathrm{x} \in \Gamma_{ij}$, computed using the two different parametrizations, also coincide [29].

Figure 2 illustrates an example of three conforming patches on a generic two-dimensional domain. The reference domain $\hat{\Omega}$ is mapped onto the three patches Ω_i, $i = 1, 2, 3$, where conformity is enforced at their interfaces and on points where multiple subdomains meet. Each interface between the individual patches corresponds to either an edge or a corner in the respective reference domain.

4 Multi-patch isogeometric-mapped neural solver

4.1 Neural network-based ansatz in the reference domain

The main building blocks of the suggested neural solver are NN-based ansatz functions capable of approximating solutions within the reference domains (i.e., the parametric space of each patch), to be then transformed into solutions on the physical geometry. In the following we assume that, for each subdomain (patch) Ω_i, the solution in the reference domain is represented by NN-based ansatz functions $\hat{u}_{\boldsymbol{\theta}_i} : \hat{\Omega}_i \to \mathbb{R}$, $i = 1, \dots, N_\Omega$. These ansatz functions consist of products and superpositions of NNs and polynomials, defined on the reference domain. To ease the notation, the vector $\boldsymbol{\theta} \in \Theta$ concatenates the trainable parameters of all NNs employed in the approximation, such that $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{N_\Omega}$. Accordingly, the patch-specific NN $\hat{u}_{\boldsymbol{\theta}_i}$ contains the trainable parameters that correspond to the *i*-th patch only.

The construction of the ansatz functions $\{\hat{u}_{\boldsymbol{\theta}_i}\}_{i=1}^{N_\Omega}$ must satisfy the Dirichlet BCs, while also ensuring differentiability within each subdomain and C^0-continuity across the interfaces between patches. These conditions are particularly advantageous when addressing discontinuous material properties between patches. Utilizing the previously defined pushforward and pullback operations, continuity requires that

$$\hat{u}_{\boldsymbol{\theta}_i}\left(\hat{\Gamma}_{i\to j}\right) = \hat{u}_{\boldsymbol{\theta}_j}\left(\hat{\Gamma}_{j\to i}\right), \tag{9}$$

for every interface Γ_{ij}. Similar to classical approaches found in IGA, condition (9) ensures a continuous transition between individual patches of the multi-patch geometry, by constraining the solution representation in the reference domain. To correctly represent intersections between parts of patch boundaries of different dimensionalities, for example boundary faces (2D), boundary lines (1D), and boundary points (0D), we introduce multi-indices. This necessity arises in particular at points where three or more patches of a multi-patch geometry coincide. To guarantee that the continuity condition (9) is satisfied in such configurations, the interface contributions must enforce continuity simultaneously across all interacting patches. To fulfill these requirements, we define our ansatz function within each patch as a superposition of an interior term, $\hat{u}^{\text{int}}_{\hat{\boldsymbol{\theta}}_i}$, representing the solution inside each patch and interface-conforming terms, $\hat{u}^{\boldsymbol{j}\to i}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}$, imposing continuity across neighboring patches. The definition reads as follows:

$$\hat{u}^{(i)}_{\boldsymbol{\theta}_i} = \hat{u}^{(i,\text{int})}_{\hat{\boldsymbol{\theta}}_i} + \sum_{\boldsymbol{j}\in\mathcal{J}(i)} \hat{u}^{\boldsymbol{j}\to i}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}, \tag{10}$$

where $\mathcal{J}(i)$ is a multi-index set associated with the *i*-th patch, which comprises the multi-indices of all intersections of boundary parts between the *i*-th patch and neighboring patches, i.e., boundary faces (2D), boundary lines (1D), and boundary points (0D), and $\boldsymbol{j}$ are the multi-indices contained therein. For example, considering the two-dimensional domain depicted in Fig. 2, $\mathcal{J}(1) = \{(1,2),(1,2,3)\}$ due to the interface Γ_{12} shared by subdomains Ω_1 and Ω_2 and the point ξ_{123} shared by all three subdomains. Accordingly, $\mathcal{J}(2) = \{(1,2),(2,3),(1,2,3)\}$ and $\mathcal{J}(3) = \{(2,3),(1,2,3)\}$. The interface terms $\hat{u}^{\boldsymbol{j}\to i}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}$, $\boldsymbol{j} \in \mathcal{J}(i)$, connect intersecting patches.

The term $\hat{u}^{(i,\text{int})}_{\hat{\boldsymbol{\theta}}_i}$ contributes solely to the interior of the subdomain Ω_i and vanishes on the interfaces defined by the multi-index set $\mathcal{J}(i)$. The trainable parameters per patch are then given as $\boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i \cup \bigcup_{\boldsymbol{j}\in\mathcal{J}(i)} \hat{\boldsymbol{\theta}}_{\boldsymbol{j}}$. The ansatz functions $\hat{u}_{\boldsymbol{\theta}_i}$ are evaluated separately for the specific patch they correspond to, and collectively yield a global solution. The corresponding ansatz space $\mathcal{V}_\Theta$ for the physical domain is then defined by the individual parameter spaces Θ of the NNs and the patch-wise inverse pullback:

$$\mathcal{V}_\Theta := \left\{\hat{u}_{\boldsymbol{\theta}_i} \circ \boldsymbol{f}_i^{-1} \,:\, i = 1, \dots, N_\Omega\right\}. \tag{11}$$

The parameter space Θ corresponds to the Cartesian product of the patch-specific parameter spaces which include the parameters of the NNs $\hat{u}^{(i)}_{\boldsymbol{\theta}_i}$, $i = 1, \dots, N_\Omega$. The global ansatz function (in the physical domain) is denoted as $u_{\boldsymbol{\theta}} \in \mathcal{V}_\Theta$. Note that, for vector-valued ansatz functions $\hat{\boldsymbol{u}} : \hat{\Omega} \to \mathbb{R}^n$, the considerations outlined above can be applied independently to each component (see Sect. 5.3).

4.1.1 Ansatz inside the domains

We first focus on constructing the ansatz within the domain. We assume that the Dirichlet boundary, Γ_{D}, is obtained as the image of specific facets of the reference domains under the pushforward map f_*. That is, for each patch, applying the pushforward map to points on a reference facet yields the corresponding physical boundary segment where Dirichlet conditions are enforced. Further, we assume that the Dirichlet data is defined by a function $g_{\text{D}} : \Gamma_{\text{D}} \to \mathbb{R}^d$ on the Dirichlet boundary. The ansatz is composed of the product of the output of the NN used to approximate the solution in the reference domain, and a polynomial that ensures the enforcement of Dirichlet BCs. The value of the ansatz on the Dirichlet boundary is imposed by adding an additional

term to the ansatz. Specifically, the ansatz inside the domain is defined as

$$\hat{u}^{(i,\mathrm{int})}_{\hat{\boldsymbol{\theta}}_i}(\hat{\mathrm{x}}) = \mathcal{N}_{\hat{\boldsymbol{\theta}}_i}(\hat{\mathrm{x}})\,\hat{p}(\hat{\mathrm{x}}) + (g_\mathrm{D} \circ \boldsymbol{f})(\hat{\mathrm{x}})\hat{p}_\mathrm{D}(\hat{\mathrm{x}}), \tag{12}$$

where $\mathcal{N}_{\hat{\boldsymbol{\theta}}_i} : [-1,1]^d \to \mathbb{R}$ is an NN depending on the trainable parameters $\hat{\boldsymbol{\theta}}_i$ and the polynomial $\hat{p}(\hat{\mathrm{x}}) = \prod_{k=1}^{d}(\hat{x}_k - 1)^{\alpha_k}(\hat{x}_k + 1)^{\beta_k}$ is chosen so that its pullback vanishes on interfaces and Dirichlet boundaries i.e.

$$\hat{p} \circ \boldsymbol{f}_i^{-1} = 0 \quad \text{for} \quad \mathrm{x} \in \partial\Omega_i \cap \left(\Gamma_\mathrm{D} \cup \bigcup_{i \neq j} \partial\Omega_j\right). \tag{13}$$

This construction ensures that the product of $\mathcal{N}_{\hat{\boldsymbol{\theta}}_i}$ and $\hat{p}$ yields a parametric function which does not influence Dirichlet boundaries and patch interfaces, however, is variable inside the patch interiors. The inclusion of interfaces in (13) is intentional, as the interface is handled separately by an additional term introduced in the next section. Using similar principles, we construct the term representing the Dirichlet BC. The choice of the polynomial $\hat{p}_\mathrm{D}$ must ensure that

$$\begin{aligned} &\hat{p}_\mathrm{D} \circ \boldsymbol{f}_i^{-1} = 1 \text{ for } \mathrm{x} \in \Gamma_\mathrm{D}, \\ &\hat{p}_\mathrm{D} \circ \boldsymbol{f}_i^{-1} = 0 \text{ for } \mathrm{x} \in \Gamma_\mathrm{D}^{\mathrm{opp}}. \end{aligned} \tag{14}$$

In (14), $\Gamma_\mathrm{D}^{\mathrm{opp}}$ denotes the physical interface obtained from the facet in the reference domain that is the spatial reflection of the Dirichlet boundary's preimage with respect to the center of the reference domain. From an implementation point of view, this approach can be interpreted as a custom output layer with a residual connection.

As an example, we revisit the 2D domain composed by 3 patches shown in Fig. 2. In the same figure, the mapping of the reference coordinate system onto the physical subdomains is also shown. For simplicity, we assume homogeneous Dirichlet BCs i.e. $g_D = 0$ on Γ_D, where Γ_D is represented by the green line. In the reference domain of Ω_2, this translates to $\hat{x}_1 = -1$, while for Ω_2 it maps to $\hat{x}_2 = 1$ and $\hat{x}_1 = -1$. There are two interfaces, Γ_{12} and Γ_{13}, and all three domains share a common point $\boldsymbol{\xi}_{123}$ for the multi-index $\boldsymbol{j} = (1,2,3)$. Then, the boundary-conforming ansatz inside the domains are:

$$\begin{aligned} \hat{u}^{(1,\mathrm{int})}_{\hat{\boldsymbol{\theta}}_1}(\hat{\mathrm{x}}) &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_1}(\hat{\mathrm{x}})(\hat{x}_1 + 1)(\hat{x}_2 + 1)(\hat{x}_2 - 1), \\ \hat{u}^{(2,\mathrm{int})}_{\hat{\boldsymbol{\theta}}_2}(\hat{\mathrm{x}}) &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_2}(\hat{\mathrm{x}})(\hat{x}_1 + 1)(\hat{x}_1 - 1)(\hat{x}_2 - 1), \\ \hat{u}^{(3,\mathrm{int})}_{\hat{\boldsymbol{\theta}}_3}(\hat{\mathrm{x}}) &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_3}(\hat{\mathrm{x}})(\hat{x}_1 + 1)(\hat{x}_2 + 1)(\hat{x}_2 - 1). \end{aligned} \tag{15}$$

4.1.2 Ansatz on domain interfaces

To connect the solutions defined on individual patches, we construct interface-conforming terms. This process is iterative and can be applied to an arbitrary number of dimensions.

We first consider the 0-dimensional intersections that do not belong to the Dirichlet boundary:

$$\Xi_0 = \left\{\partial\Omega_i \cap \partial\Omega_j \;:\; \partial\Omega_i \cap \partial\Omega_j \text{ is 0-dimensional and } \partial\Omega_i \cap \partial\Omega_j \not\subseteq \Gamma_\mathrm{D}\right\}. \tag{16}$$

Note that the indices from the set Ξ_0 may have different sizes. Since an element of Ξ_0 may be shared by multiple patches, its pre-image corresponds to different points in the respective reference domains. Given a point $\boldsymbol{\xi} \in \Xi_0$, let $\boldsymbol{j} = \mathcal{I}(\boldsymbol{\xi})$ contain the indices of all patches sharing the 0-dimensional interface $\boldsymbol{\xi}$. Then, the ansatz in all the patches $i \in \boldsymbol{j}$ is expressed as

$$\hat{u}^{\boldsymbol{j}\to i}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}(\hat{\mathrm{x}}) = \hat{\boldsymbol{\theta}}_{\boldsymbol{j}} \prod_{k=1}^{d}\left((\hat{x}_k - 1)^{\alpha_k}(\hat{x}_k + 1)^{\beta_k}\right), \tag{17}$$

where the exponents α_k and β_k are chosen so that the function vanishes at all vertices of the reference domain but $\boldsymbol{f}_i^{-1}(\boldsymbol{\xi})$. In this case, the trainable parameters $\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}$ are responsible for the value of the ansatz on $\boldsymbol{\xi}$.

Next, we construct the q-dimensional interface ansatz based on the sets $\Xi_{q-1}, \ldots, \Xi_0$. The set Ξ_q contains all the q-dimensional interfaces:

$$\Xi_q = \left\{\partial\Omega_i \cap \partial\Omega_j \;:\; \partial\Omega_i \cap \partial\Omega_j \text{ is } q\text{-dimensional and } \partial\Omega_i \cap \partial\Omega_j \not\subseteq \Gamma_\mathrm{D}\right\}. \tag{18}$$

Similarly to the 0-dimensional case, for an element $\boldsymbol{\xi} \in \Xi_q$ shared by patches indexed by $\boldsymbol{j} = \mathcal{I}(\boldsymbol{\xi})$, the corresponding ansatz in the i-th patch, $i \in \boldsymbol{j} = \mathcal{I}(\boldsymbol{\xi})$, is defined by

$$\hat{u}^{\boldsymbol{j}\to i}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}(\hat{\mathrm{x}}) = \mathcal{N}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}(\hat{x}_{l_1}, \ldots, \hat{x}_{l_q}) \prod_{k=1}^{d}\left((\hat{x}_k - 1)^{\alpha_k}(\hat{x}_k + 1)^{\beta_k}\right), \tag{19}$$

where the NN $\mathcal{N}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}$ is defined on the pre-image $\boldsymbol{f}_i^{-1}(\boldsymbol{\xi})$, and the coefficients α_k and β_k are chosen such that $\hat{u}^{\boldsymbol{j}\to i}_{\hat{\boldsymbol{\theta}}_{\boldsymbol{j}}}(\hat{\mathrm{x}}) \circ \boldsymbol{f}_i^{-1}$ vanishes on all q-dimensional faces of the reference domain that do not contain $\boldsymbol{\xi}$, as well as on all elements of $\Xi_{q-1} \cup \cdots \cup \Xi_0$. This latter condition ensures that the contribution from the q-dimensional interface does not overlap with those from lower-dimensional interfaces. The procedure is iteratively repeated until the set Ξ_{d-1} and the corresponding ansatz functions are constructed.

As an example, we revisit the domain presented in Fig. 2. In this case, the set $\Xi_0 = \{\boldsymbol{\xi}_{123}\}$ contains a single element

and its corresponding index set is $\mathcal{I}(\boldsymbol{\xi}_{123}) = (1, 2, 3)$. Then, the interface ansatz functions are

$$\begin{aligned}\hat{u}^{(1,2,3)\to 1} &= \hat{\boldsymbol{\theta}}_{(1,2,3)}(\hat{x}_1 + 1)(\hat{x}_2 - 1),\\ \hat{u}^{(1,2,3)\to 2} &= \hat{\boldsymbol{\theta}}_{(1,2,3)}(\hat{x}_1 + 1)(\hat{x}_2 + 1),\\ \hat{u}^{(1,2,3)\to 3} &= \hat{\boldsymbol{\theta}}_{(1,2,3)}(\hat{x}_1 - 1)(\hat{x}_2 + 1),\end{aligned} \tag{20}$$

where $\hat{\boldsymbol{\theta}}_{(1,2,3)}$ are the trainable parameters corresponding to the common point $\boldsymbol{\xi}_{123}$. The set Ξ_1 contains the two interfaces Γ_{12} and Γ_{23}, with $\mathcal{I}(\Gamma_{12}) = (1, 2)$, $\mathcal{I}(\Gamma_{23}) = (2, 3)$. The corresponding interface functions are

$$\begin{aligned}\hat{u}^{(1,2)\to 1}_{\hat{\boldsymbol{\theta}}_{(1,2)}} &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_{(1,2)}}(\hat{x}_1)(\hat{x}_1 + 1)(\hat{x}_1 - 1)(\hat{x}_2 - 1),\\ \hat{u}^{(1,2)\to 2}_{\hat{\boldsymbol{\theta}}_{(1,2)}} &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_{(1,2)}}(\hat{x}_1)(\hat{x}_1 + 1)(\hat{x}_1 - 1)(\hat{x}_2 + 1),\\ \hat{u}^{(2,3)\to 2}_{\hat{\boldsymbol{\theta}}_{(2,3)}} &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_{(2,3)}}(\hat{x}_2)(\hat{x}_1 + 1)(\hat{x}_2 + 1)(\hat{x}_2 - 1),\\ \hat{u}^{(2,3)\to 3}_{\hat{\boldsymbol{\theta}}_{(2,3)}} &= \mathcal{N}_{\hat{\boldsymbol{\theta}}_{(2,3)}}(\hat{x}_2)(\hat{x}_1 - 1)(\hat{x}_2 + 1)(\hat{x}_2 - 1).\end{aligned} \tag{21}$$

In this case, the NNs used are restricted to only one of the coordinates. Moreover, the Dirichlet boundary as well as the set Ξ_0 are taken into account for the interfaces corresponding to Γ_{12}.

To illustrate how the ansatz functions are constructed on domain interfaces, we refer to Fig. 3. On the left-hand side, the interface functions from equation (20) spanning three subdomains are omitted, but the planes where continuity must be enforced in both the parametric and physical domains are indicated. The red circle marks the coincidence point of all three subdomains (similar to point $\boldsymbol{\xi}_{123}$ in Fig. 1). On the right-hand side, the complete ansatz function is shown, incorporating all interface functions. As can be observed, continuity is enforced across all patches.

5 Numerical examples

In this section, the suggested neural solver is first applied to a 3D benchmark problem and then to two engineering use-cases, namely, a 2D magnetostatics simulation and a 3D solid mechanics simulation involving nonlinear material behavior and nonlinear BCs. In all cases, we begin with the formulation of the problem at hand and then proceed with implementation details and the corresponding numerical results. All numerical examples are available in a dedicated online repository[1]

5.1 Poisson equation in L-shaped domain

5.1.1 Problem formulation

To demonstrate the capabilities of the proposed multi-patch isogeometric-mapped neural solver, we first consider a 3D BVP governed by the Poisson equation. The computational domain $\Omega \subset \mathbb{R}^3$ is an L-shaped thick quarter-annulus, constructed with three conforming NURBS patches $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$. In cylindrical coordinates (r, θ, z), the subdomains are defined as $\Omega_1 = [1, 2] \times [0, \pi/4] \times [0, 1]$, $\Omega_2 = [2, 3] \times [0, \pi/4] \times [0, 1]$, and $\Omega_3 = [2, 3] \times [\pi/4, \pi/2] \times [0, 1]$. The geometric parametrization of the domain is depicted in Fig. 4.

The Poisson BVP with homogeneous Dirichlet BCs is given by $-\Delta u = f$ in Ω, with $u = 0$ on $\partial\Omega$. To rigorously evaluate the accuracy of the neural solver, we employ the method of manufactured solutions. The exact solution is prescribed as

$$\begin{aligned}u^{\text{exact}}(r, \theta, z) &= \sin\big(\pi(r - 1)\big)\sin(4\theta)\\ &\sin(\pi z) + \sin\big(2\pi(r - 1)\big)\sin(8\theta)\sin(2\pi z).\end{aligned} \tag{22}$$

By design, u^{exact} vanishes on all boundaries of the domain, thus satisfying the homogeneous Dirichlet BCs. The source term f is derived analytically by applying the negative

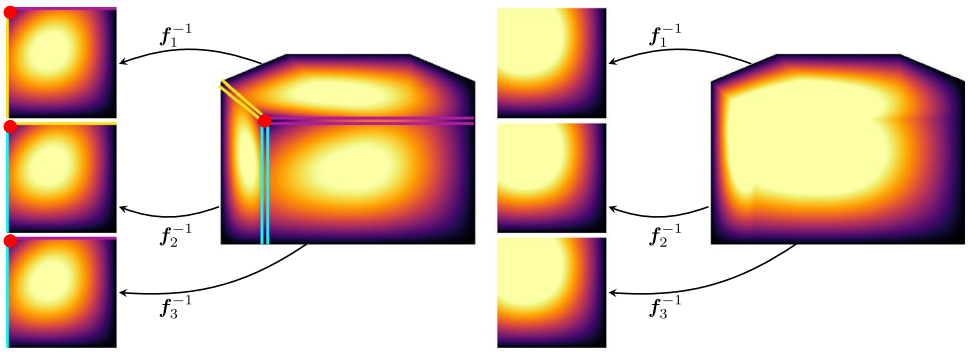


Fig. 3 Illustration of the construction of interface ansatz functions. At the point where multiple subdomains meet, indicated by the red circle, the interface functions take zero values. Continuity is enforced at the interfaces indicated by the colored lines (yellow, cyan, and purple) in the reference and physical domains

[1] https://github.com/ion-g-ion/Paper-IGA-PINNs..

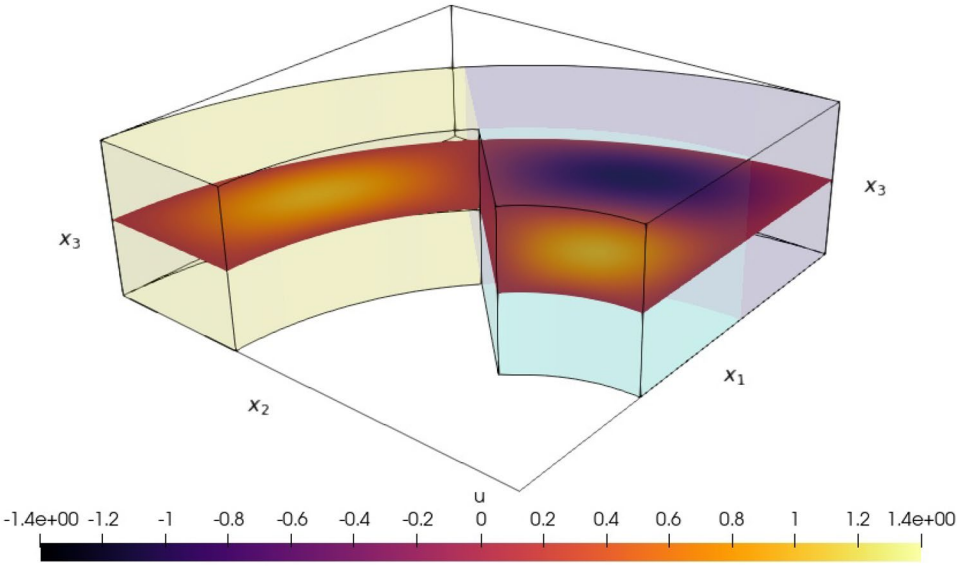


Fig. 4 The three-patch domain and a slice of the manufactured solution at $x_3 = 0.5$. The three patches are marked with different colors

Laplacian to the exact solution, i.e., $f = -\Delta u^{\text{exact}}$. A snapshots of the manufactured solution is shown in Fig. 4.

Following the variational framework introduced in Sect. 2, the corresponding energy functional to be minimized is given by

$$I(u) = \frac{1}{2}\int_\Omega \text{grad}u \cdot \text{grad}u \,\text{dx} - \int_\Omega fu \,\text{dx}. \tag{23}$$

In the multi-patch neural setting, we minimize the same functional with the conformal ansatz $u_{\boldsymbol{\theta}}$ in place of u, i.e., $I(u_{\boldsymbol{\theta}})$. For the three-patch configuration, we use

$$u_{\boldsymbol{\theta}} = u_1 + u_2 + u_3 + u_{12} + u_{23} + u_{123}, \tag{24}$$

where u_1, u_2, u_3 denote the patch-interior NN contributions, u_{12}, u_{23} denote the NN contributions defined on the two-dimensional interfaces between neighboring patches, and u_{123} denotes the contribution defined on the common line shared by the three patches. The exact expressions of these contributions are those given in Sect. 4.1, in particular equations (20) and the associated interface/domain ansatz definitions. The energy functional is evaluated by pulling back the integrals to the reference domains $\hat{\Omega}_i = [-1, 1]^3$, $i = 1, 2, 3$, of the individual patches.

5.1.2 Experimental setup and parametric study

The ansatz is constructed to strongly enforce the homogeneous Dirichlet BCs and ensure C^0-continuity across the two internal patch interfaces, i.e, Γ_{12} and Γ_{23}. This requires three domain-NNs (i.e., u_1, u_2, and u_3), two interface-NNs for the interfaces Γ_{12} and Γ_{23} (i.e., u_{12} and u_{23}), and one additional interface contribution for the common line shared by all three patches (i.e., u_{123}).

To assess the robustness and convergence of the proposed method, we perform a comprehensive parametric study over different NN architectures and training hyperparameters. For the NNs, we utilize a residual network (ResNet) architecture with a squared leaky ReLU activation function. We systematically vary NN capacity by testing the number of neurons per hidden layer, $n_l \in \{32, 64, 128\}$, and the number of residual blocks, $n_{\text{blocks}} \in \{2, 3\}$. Furthermore, we vary the integration density by testing the number of batches per epoch, $N_{\text{batch}} \in \{512, 1024, 2048\}$, and the number of Monte Carlo collocation points per batch, $N_{\text{points}} \in \{2048, 4096, 8192\}$. All configurations are trained for 100 epochs using the Adam optimizer [65].

To quantify the accuracy of the neural solver, we compute the relative L^2 and H^1 errors over the entire domain Ω , respectively defined as

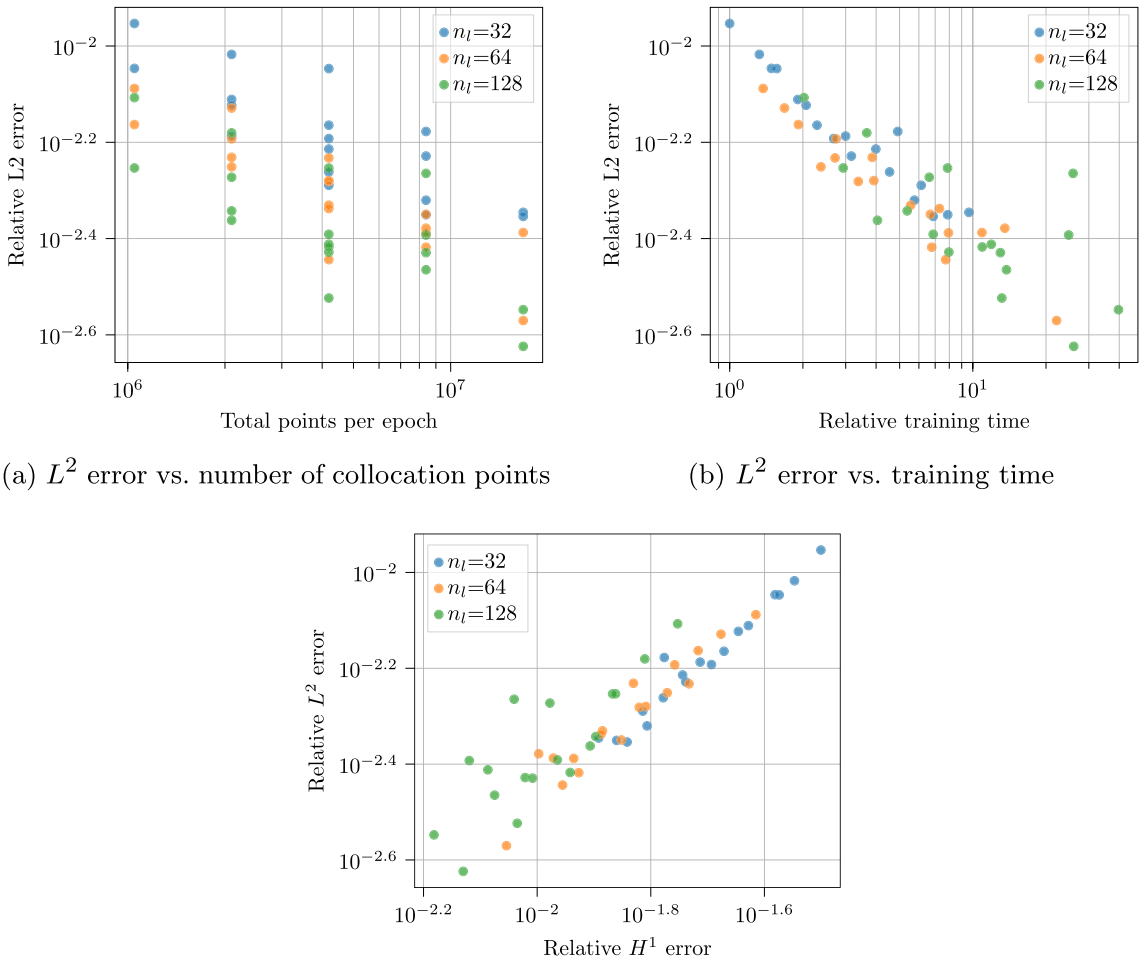
$$\epsilon_{\text{rel},L^2(\Omega)} = \frac{\|u_{\boldsymbol{\theta}} - u^{\text{exact}}\|_{L^2(\Omega)}}{\|u^{\text{exact}}\|_{L^2(\Omega)}}, \tag{25}$$

$$\epsilon_{\text{rel},H^1(\Omega)} = \frac{\sqrt{\|u_{\boldsymbol{\theta}} - u^{\text{exact}}\|^2_{L^2(\Omega)} + \|\nabla u_{\boldsymbol{\theta}} - \nabla u^{\text{exact}}\|^2_{L^2(\Omega)}}}{\sqrt{\|u^{\text{exact}}\|^2_{L^2(\Omega)} + \|\nabla u^{\text{exact}}\|^2_{L^2(\Omega)}}}, \tag{26}$$

where, $u_{\boldsymbol{\theta}}$ is the solution of the proposed multi-patch isogeometric-mapped neural solver. The results of the parametric study reveal that the neural solver consistently converges towards the exact solution. The best-performing configuration ($n_l = 128$, $n_{\text{blocks}} = 2$, $N_{\text{batch}} = 2048$, and $N_{\text{points}} = 8192$) achieves a relative L^2 error of 2.38×10^{-3} and a relative H^1 error of 7.41×10^{-3}.

Figure 5 illustrates the relationships between the relative L^2 error, the total number of collocation points per epoch, the total training time, and the relative H^1 error. As can be observed, increasing the number of collocation points per epoch consistently decreases the generalization error across all NN sizes, highlighting the data-driven nature of the variational PINN formulation and the necessity of adequate numerical integration. However, this accuracy comes at the cost of increased computational time. The plots reveal a distinct Pareto frontier: while wider NNs ($n_l = 128$) achieve the lowest asymptotic errors given sufficient integration points, narrower NNs ($n_l = 64$) offer a highly favorable trade-off between accuracy and computational cost in the pre-asymptotic regime, converging much faster for moderate error tolerances. Furthermore, a strong positive correlation between the L^2 and the H^1 errors can be observed across all NN architectures. This indicates that the variational training procedure, which minimizes the energy functional containing the gradient of the solution, effectively and simultaneously minimizes both the solution error and its derivative error without requiring explicit multi-objective weighting.

Finally, Fig. 6 presents the pointwise absolute error $|u_{\boldsymbol{\theta}} - u^{\text{exact}}|$ evaluated within the individual subdomains and across the internal interfaces. The error distribution demonstrates that approximation accuracy remains consistent across the entire geometry. Crucially, the error exhibits smooth transitions across the patch boundaries without localized spiking, thus validating the efficacy of the dedicated interface NNs in seamlessly combining the local patch solutions together while maintaining C^0-conformity.

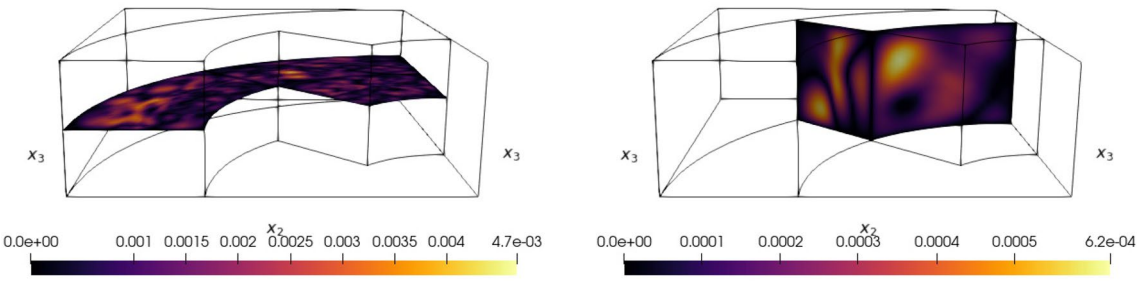


(a) L^2 error vs. number of collocation points

(b) L^2 error vs. training time

(c) L^2 error vs. H^1 error

Fig. 5 Parametric study results. Top left: Relative L^2 error as a function of the total number of Monte Carlo collocation points per epoch. Top right: Relative L^2 error as a function of the total training time. Bottom: Relative L^2 error as a function of the relative H^1 error



(a) Error at $x_3 = 0.5$. (b) Error across the interfaces Γ_{12} and Γ_{23}.

Fig. 6 Pointwise absolute error $|u_{\boldsymbol{\theta}} - u^{\text{exact}}|$ distributed across the individual subdomains

5.2 Magnetostatic simulation of quadrupole magnet

5.2.1 Magnetostatic formulation

We consider the magnetostatic problem setting, which can be retrieved by considering the eddy-current problem and dropping the time dependencies and conductive materials [69]. Under these assumptions, the problem can be formulated in terms of the magnetic vector potential. If the problem is reduced to 2D, only the longitudinal component $u : \Omega \to \mathbb{R}$ of the magnetic vector potential is required [69]. In this case, the governing equation for a linear isotropic material is

$$\begin{aligned} -\text{grad}\,(\nu\,\text{grad}u) &= j_z, \quad \text{in } \Omega, \\ u &= 0, \quad \text{on } \Gamma_{\text{D}}, \\ \nu\,\text{grad}u \cdot \boldsymbol{n} &= 0, \quad \text{on } \Gamma_N, \end{aligned} \tag{27}$$

where Ω is the computational domain, j_z is the z-component of the excitation current density, Γ_{D} the Dirichlet boundary and Γ_{N} the Neumann boundary. An energy functional minimization problem of the form (3) can be derived, where the energy functional is given as

$$I(u) = \frac{1}{2}\int_{\Omega} \nu\,\text{grad}u \cdot \text{grad}u\ \text{dx} - \int_{\Omega} u j_z\ \text{dx}. \tag{28}$$

By taking into consideration a multi-patch decomposition of the domain Ω as shown in Sect. 3.2, the evaluation of the energy functional in the reference domain yields

$$\begin{aligned} I\left(\hat{u}_{\boldsymbol{\theta}}\right) = &\frac{1}{2}\sum_{i=1}^{N_\Omega} \int_{\hat{\Omega}_i} \nu\,\text{grad}\left(\hat{u}_{\boldsymbol{\theta}}\right)^{\intercal}\,\text{K}^{(i)}(\hat{\text{x}})\,\text{grad}\left(\hat{u}_{\boldsymbol{\theta}}\right)\,\text{d}\hat{\text{x}} \\ &- \int_{\hat{\Omega}_{\text{s}}} j_z \cdot \hat{u}_{\boldsymbol{\theta}}\,|\det D\boldsymbol{f}_i|\,\text{d}\hat{\text{x}}, \end{aligned} \tag{29}$$

where $\text{K}^{(i)}(\hat{\text{x}}) = (D\boldsymbol{f}_i)^{-\intercal}(D\boldsymbol{f}_i)^{-1}\,|\det D\boldsymbol{f}_i|$ and $\hat{\Omega}_{\text{s}}$ is the reference domain corresponding to the patch that contains the source current excitation. The ansatz function $\hat{u}_{\boldsymbol{\theta}}$ is constructed to naturally enforce homogeneous Dirichlet BCs, hence, we do not require additional penalization terms.

For the material distribution, we consider linear magnetic materials, where $\mu_0 = 4\pi \cdot 10^{-7}\frac{H}{m}$ is the permeability of free space, $\mu_{\text{Fe}} = 2000\mu_0$ the permeability of iron, and $\mu_{\text{Cu}} \approx \mu_0$ the permeability of copper. For the excitation, we consider a constant current density $j_z = 7.95 \cdot 10^8 \frac{A}{m^2}$.

To quantify error, we use the relative L^2 error, given by the formula

$$\epsilon_{\text{rel},L^2(\Omega)} = \frac{\left\|u_{\boldsymbol{\theta}} - u^{\text{FEM}}\right\|_{L^2(\Omega)}}{\left\|u^{\text{FEM}}\right\|_{L^2(\Omega)}}, \tag{30}$$

where u^{FEM} is a highly resolved FEM solution. For both computational examples, we calculate the relative L^2 error, once on the complete domain Ω and for the individual subdomains $\Omega_1, \ldots, \Omega_{N_\Omega}$. This approach enables us to assess the overall error across the domain and analyze its distribution.

5.2.2 Simple quadrupole magnet model

First, we consider a relatively simple quadrupole magnet geometry, depicted in Fig. 7. On the left hand side of Fig. 7, the complete cross-section of the quadrupole magnet is depicted. On the right hand side of Fig. 7, one eighth of the magnet's geometry is shown, exploiting its rotational symmetry. The latter constitutes the domain geometry of the computational model and is partitioned into patches $\Omega_1, \ldots, \Omega_4$. The characterization "simple" is due to the fact that only four patches are required for the complete parametrization of Ω. The purple shading highlights the location of the computational model's domain on the complete cross-section domain. The iron yoke is shown in gray, the current excitations in yellow, and the air domain in white.

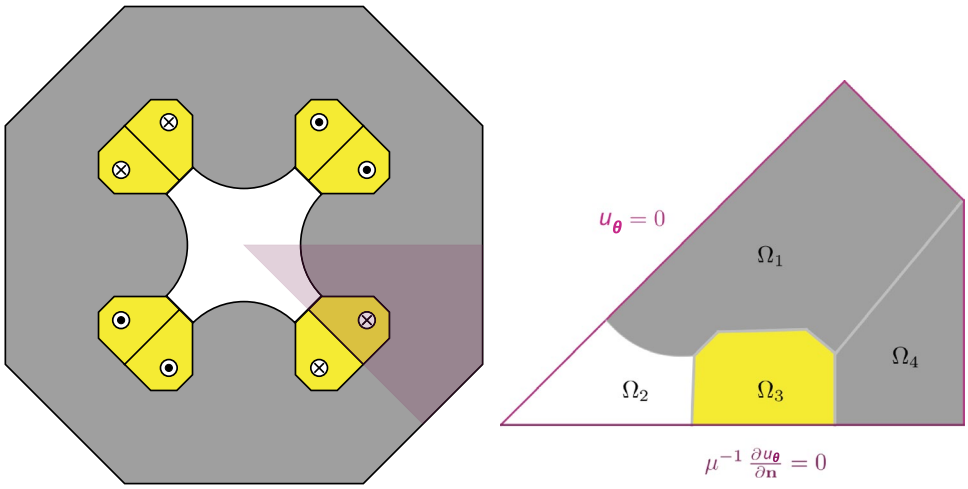


Fig. 7 Domain geometry of the simple quadrupole model. The iron yoke is shown in gray, the current excitations in yellow, and the air gap in white. The purple shading indicates the model section in the symmetry planes. Left: Full cross-section. Right: One-eighth of cross-section, exploiting rotational symmetry. The domain is partitioned into patches Ω_i, $i = 1, \ldots, 4$

Table 1 Distribution of NNs and trainable parameters for the simple quadrupole model

	Number of NNs	Number of trainable parameters
Point interfaces (0D)	–	2
Interface NNs (1D)	5	108
Domain NNs (2D)	4	360
Total	9	1982

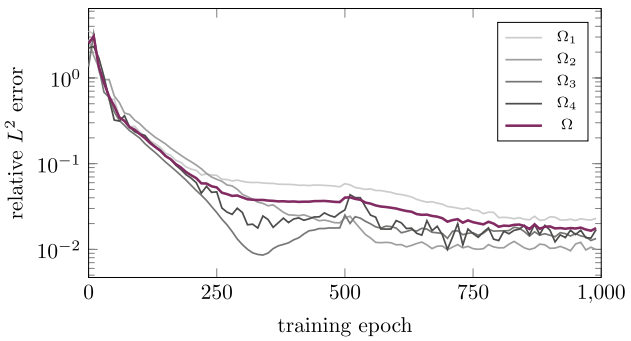


Fig. 8 Relative L^2 error during training for the entire computational domain and for each subdomain and the simple quadrupole model

Dirichlet and Neumann BCs are applied at the boundaries as indicated.

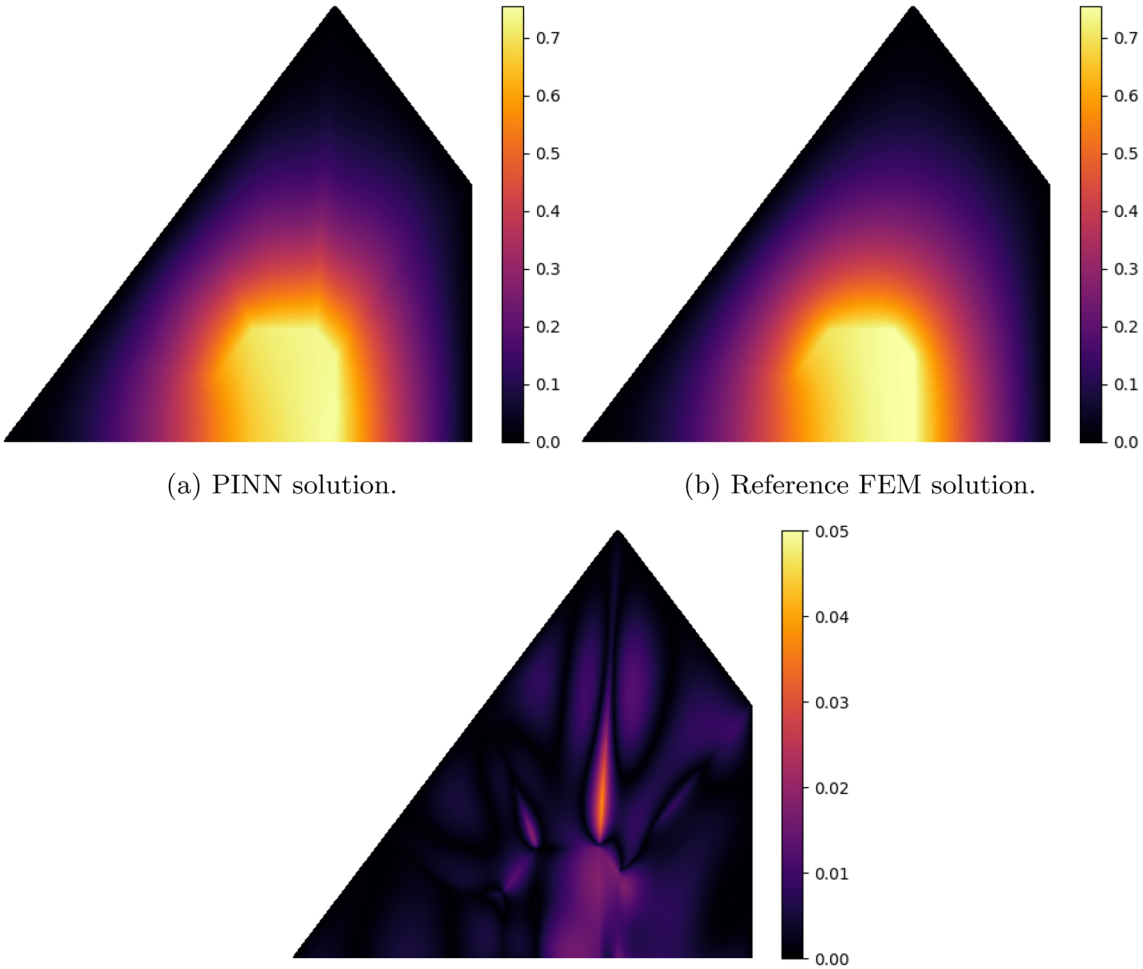
A residual neural network (ResNet) architecture [70] is chosen for the NNs on both the subdomains and on the interfaces. In particular, we employ NNs with four fully connected (FC) layers, including an input layer, one ResNet block, and an output layer. The number of neurons in each layer varies depending on whether the NN is allocated to the domain or to an interface. That is, more neurons are allocated for the domain compared to the interface. For the domain-NNs, we employ 16 neurons per layer, and the input and output dimensions for each FC layer vary throughout the NN according to $(2, 16) \to (16, 16) \to (16, 16) \to (16, 1)$. For the interface-NNs, we employ eight neurons per layer, and the input and output dimensions for each FC layer vary according to $(1, 8) \to (8, 8) \to (8, 8) \to (8, 1)$. In both cases, there exist two skip connections, before the second and fourth layer, respectively. The hyperbolic tangent activation function is used. The NNs are trained for 1000 epochs using the Adam optimizer and utilizing 4×10^4 Monte Carlo sampling points in each epoch.

For the reference solution, we employ the FEniCS FEM solver [71, 72], using a first-order tetrahedral mesh consisting of 13957 elements. Table 1 provides an overview of the NN and trainable parameters employed in this use-case.

A plot of the relative L^2 error over the entire training process is displayed in Fig. 8, concerning the full computational domain Ω and the individual patches $\Omega_1, \ldots, \Omega_4$. Even though a low absolute error is obtained throughout the domain, the maximal error occurs in the subdomain Ω_1, in particular in areas where the NURBS parametrization exhibits non-smooth behavior. This is also observed in Fig. 9, which shows the absolute error over the computational domain. The minimum relative L^2 error over Ω is $1.77 \cdot 10^{-2}$.

5.2.3 Complex quadrupole magnet model

Next, we consider a more complex quadrupole magnet geometry, depicted in Fig. 10. The left hand side of Fig. 10 shows the complete cross-section of the quadrupole magnet, while the right hand side shows one eighth of its geometry, exploiting its rotational symmetry. The latter constitutes the domain geometry of the computational model and is partitioned into patches $\Omega_1, \ldots, \Omega_9$. The material distribution and coloring scheme is identical to Fig. 7. There are 5 more NURBS parametrizations than in the previous section, hence the characterization "complex". The ansatz function for this example is more intricate, as conformity must be ensured over nine patches instead of four. Again, Dirichlet and Neumann BCs are applied at the boundaries as indicated.



(a) PINN solution. (b) Reference FEM solution.

(c) Pointwise error.

Fig. 9 Field solutions and corresponding error field for the simple quadrupole magnet model

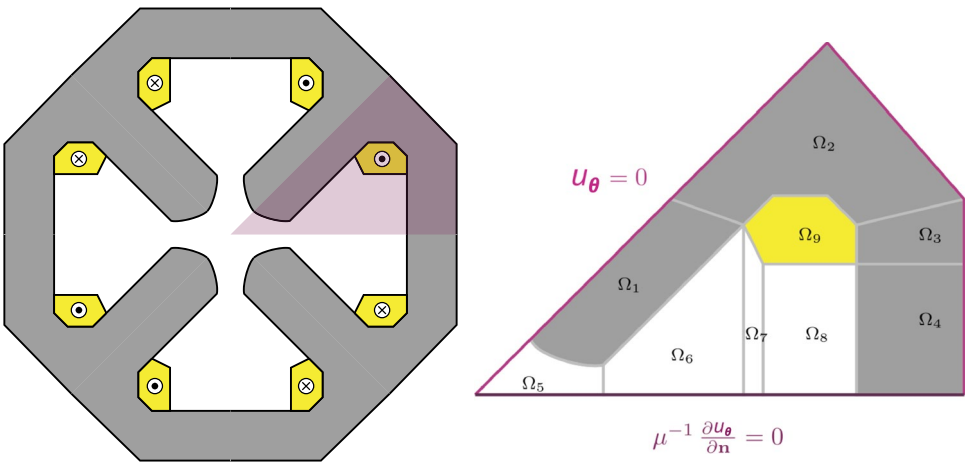


Fig. 10 Domain geometry of the complex quadrupole model. The iron yoke is shown in gray, the current excitations in yellow, and the air gap in white. The purple shading indicates the model section in the symmetry planes. Left: Full cross-section. Right: One-eighth of cross-section, exploiting rotational symmetry. The domain is partitioned into patches Ω_i, $i = 1, \ldots, 9$

Again, a ResNet architecture is employed for both the domain- and interface-NNs. In this case, we employ NNs with five FC layers, including an input layer, one ResNet block, and an output layer. For the domain-NNs, we employ two variations of the same architecture. In subdomains Ω_1 and Ω_2, we allocate 16 neurons per layer, such that the input and output dimensions for each FC layer vary throughout the NN according to $(2, 16) \to (16, 16) \to (16, 16) \to (16, 16) \to (16, 1)$. For

Table 2 Distribution of NNs and trainable parameters for the complex quadrupole model

	Number of NNs	Number of trainable parameters
Point interfaces (0D)	–	5
Interface NNs (1D)	13	74
Domain NNs for $\Omega_3 - \Omega_9$ (2D)	7	183
Domain NNs for Ω_1, Ω_2 (2D)	2	615
Total	22	3478

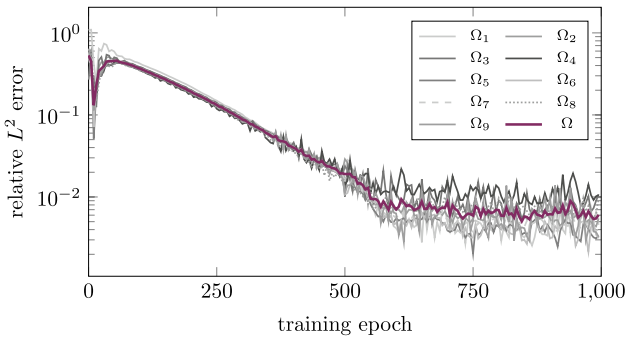


Fig. 11 Relative L^2 error during training for the entire computational domain and for each subdomain and the complex quadrupole model

the rest subdomains, we allocate eight neurons per layer, such that the input and output dimensions vary according to $(2, 8) \to (8, 8) \to (8, 8) \to (8, 8) \to (8, 1)$. For the interface-NNs, we employ five neurons per layer and the input and output dimensions for each FC layer vary according to $(1, 5) \to (5, 5) \to (5, 5) \to (5, 5) \to (5, 1)$. In both cases there are two skip connections, before the third and the final layer, respectively. The activation function used is the hyperbolic tangent function $\sigma(x) = \tanh(x)$. The NNs are trained for 1000 epochs with the Adam optimizer, utilizing 1.8×10^5 Monte Carlo sampling points per epoch. For the reference solution, we employ the FEniCS FEM solver [71, 72], using a first-order tetrahedral mesh consisting of 770,743 elements. Table 2 we provides an overview of the NN and trainable parameters employed in this use-case.

Similar to the previous section, the NNs are trained for 1000 training epochs and the relative L^2 error is calculated with respect to an FEM reference solution using $4 \cdot 10^4$ quadrature points. The relative L^2 error over the entire training process is displayed in Fig. 11 for both the full computational domain and its subdomains. Although the absolute error is consistently low across the domain, the highest error occurs within the subdomain Ω_2, particularly in areas where the NURBS parameterization displays non-smooth behavior. This is also evident in Fig. 12c, which shows of the absolute pointwise error over the quadrupole's computational domain. The minimum relative L^2 error over Ω is $6.05 \cdot 10^{-3}$.

5.3 Solid mechanics simulation of mechanical holder

5.3.1 Solid mechanics formulation

We now consider a problem in the setting of nonlinear solid mechanics. Given a domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, composed of N_Ω patches, we define the displacement field $\boldsymbol{u} : \Omega \to \mathbb{R}^d$, which describes the deformation of the domain Ω. Then, given a point $\mathrm{x} \in \Omega$, its position in the deformed configuration is given by $\boldsymbol{\varphi}(\mathrm{x}) = \mathrm{x} + \boldsymbol{u}(\mathrm{x})$. The task is to find the displacement that minimizes the total potential energy [73]:

$$\min_{\boldsymbol{u} \in \mathcal{V}} I(\boldsymbol{u}) = \min_{\boldsymbol{u}} \sum_{i=1}^{N_\Omega} \left(\int_{\Omega_i} \psi(\boldsymbol{u}) \, \mathrm{dx} - \int_{\Omega_i} \boldsymbol{b} \cdot \boldsymbol{u} \, \mathrm{dx} - \int_{\Gamma_i} \boldsymbol{T} \cdot \boldsymbol{u} \, \mathrm{d}\sigma(\mathrm{x}) \right), \tag{31}$$

where $\mathcal{V}$ is a suitable space of all possible displacements satisfying the BCs, ψ is the stored energy density, $\boldsymbol{b}$ is the body force, and $\boldsymbol{T}$ is the surface traction prescribed on some parts of the boundary $\Gamma_i \subset \partial\Omega, i = 1, ..., N_\Omega$. The boundaries are considered to be the union of images of facets in the reference domain, such that $\Gamma_i = \boldsymbol{f}_i(\hat{\Gamma}_i)$. Both body forces and surface traction are defined in the undeformed configuration. In this work we use the Saint Venant-Kirchhoff material model [73] with the stored energy density

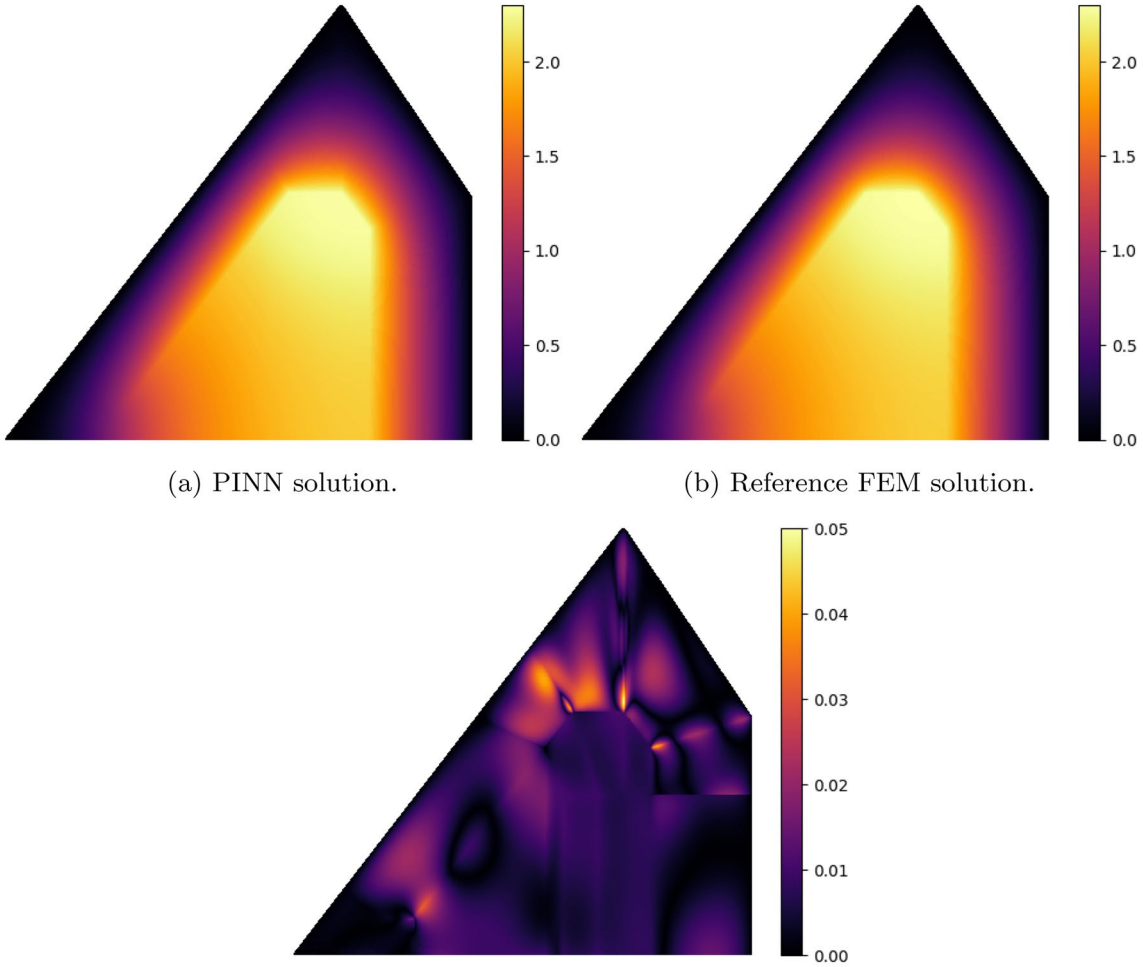
$$\psi = \frac{\lambda}{2}(\mathrm{tr}\boldsymbol{E})^2 + \mu \boldsymbol{E} \cdot \boldsymbol{E}, \tag{32}$$

where $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}\boldsymbol{F}^\top - \boldsymbol{I})$ is the Green strain tensor, $\boldsymbol{F} = \boldsymbol{I} + \mathrm{grad}\boldsymbol{u}$ is the deformation gradient and λ, μ are the Lamé coefficients. The formulation can be expressed in the reference domains $\hat{\Omega}_i$ as

$$\begin{aligned} I(\hat{\boldsymbol{u}}) = \sum_{i=1}^{N_\Omega} \Bigg(& \int_{\hat{\Omega}_i} \hat{\psi}(\hat{\boldsymbol{u}}) |\mathrm{det} D\boldsymbol{f}_i| \, \mathrm{d}\hat{\mathrm{x}} - \int_{\Omega_i} \boldsymbol{b} \cdot \hat{\boldsymbol{u}} \, |\mathrm{det} D\boldsymbol{f}_i| \, \mathrm{d}\hat{\mathrm{x}} \\ & - \int_{\Gamma_i} \boldsymbol{T} \cdot \boldsymbol{u} || D\boldsymbol{f}_i \boldsymbol{\nu} || |\mathrm{det} D\boldsymbol{f}_i| \, \mathrm{d}\sigma(\hat{\mathrm{x}}) \Bigg), \end{aligned} \tag{33}$$

where $\hat{\psi} = \frac{\lambda}{2}(\mathrm{tr}\hat{\boldsymbol{E}})^2 + \mu \hat{\boldsymbol{E}} \cdot \hat{\boldsymbol{E}}$, $\hat{\boldsymbol{E}} = \frac{1}{2}(\hat{\boldsymbol{F}}\hat{\boldsymbol{F}}^\top - \boldsymbol{I})$ and $\hat{\boldsymbol{F}} = \boldsymbol{I} + (\mathrm{grad}\ \hat{\boldsymbol{u}})(D\boldsymbol{f}_i)^{-1}$ and $\boldsymbol{\nu}_i, i = 1, ..., N_\Omega$ are the unit normals.

A further topic of interest in solid mechanics simulations are contact BCs. In this work, we consider a second body, denoted as $\Omega^{\mathrm{r}} \subset \mathbb{R}^d$, which is rigid and not part of the computational domain (see Fig. 13). To prevent the penetration of the two bodies, we use the penalty approach [74]. The idea is to apply a traction along the boundary subdomain where the penetration occurs. To that end, we define the



(a) PINN solution. (b) Reference FEM solution.

(c) Pointwise error.

Fig. 12 Field solutions and corresponding error field for the complex quadrupole magnet model

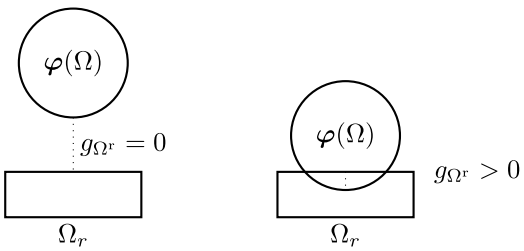


Fig. 13 Illustration of the penetration distance between a rigid body Ω^r and the deformed configuration $\boldsymbol{\varphi}(\Omega)$. On the left, the penetration distance is zero as no penetration yet exists. On the right, the penetration distance becomes strictly positive

penetration distance function from a point $\boldsymbol{\varphi}(\mathrm{x}), \mathrm{x} \in \partial\Omega$ to the body Ω^r as

$$g_{\Omega^r}(\boldsymbol{\varphi}, \mathrm{x}) = \begin{cases} \min\limits_{\boldsymbol{x}' \in \Omega^r} ||\boldsymbol{\varphi}(\mathrm{x}) - \mathrm{x}'||_2^2, & \boldsymbol{\varphi}(\mathrm{x}) \text{ penetrates } \Omega^r, \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

The following penalty is introduced:

$$I_C = \frac{1}{2} \int_{\partial\Omega} \epsilon_N g_{\Omega^r}^2(\boldsymbol{\varphi}, \mathrm{x}) \, ||\boldsymbol{F}^{-\top} \boldsymbol{N}||_2 \, \det \boldsymbol{F} \; \mathrm{d}\sigma(\mathrm{x}), \tag{35}$$

where $\boldsymbol{N}$ is the normal vector on the undeformed boundary and $\epsilon_N > 0$ is a penalty constant. The integral can be expressed in the reference configuration using (33). By adding the penalty and the potential energy and introducing the conformal ansatz functions, denoted by $\boldsymbol{u_\theta}$, the final optimization problem to be solved is:

$$\min_{\boldsymbol{u_\theta} \in \mathcal{V}} I(\boldsymbol{u_\theta}) + I_C(\boldsymbol{u_\theta}). \tag{36}$$

The penalty term expressed in (35) is also nonlinear with respect to the displacement. Conventional FEM based methods employ Newton's method to cope with the nonlinearity. In the case of nonlinear solid mechanics with contact BCs, incremental loading (displacement applied on boundary, body forces or boundary traction) is used to avoid the divergence of the Newton iterations due to badly selected

starting points, and the previous increment is used as a starting point for the next. In the framework developed in this work, we solve the stationary problem directly. The load can be added as an additional input to the NNs that approximate the solution.

5.3.2 Mechanical holder model

In this section, we examine a structural holder, the geometry of which is depicted in Fig. 14. The computational domain consists of five NURBS patches, also shown in Fig. 14. To model the material behavior, we utilize the Saint Venant-Kirchhoff constitutive law presented in (32), with the Lamé parameters $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$, where the Young's modulus is $E = 2000$ MPa and the Poisson's ratio is $\nu = 0.3$.

The mechanical setup includes two rigid half-planes, defined by the relations $x_2 = -0.75 - 0.05\phi_1$ for $x_1 \leq 1$, and $x_2 = 0.75 + 0.05\phi_2$ for $x_1 \leq 1$, which make contact with the domain. The variables $\phi_1, \phi_2 \in [-1, 1]$ serve as additional inputs to the NNs that control the BCs. The object is fixed at $x_1 = 2.75$ by imposing a zero Dirichlet BC on the displacement.

The solution is approximated using a total of twelve NNs, where five NNs are employed for the five subdomains, five NNs are defined on the surface interfaces between the domains, and two NNs are defined on the line interfaces where three domains meet. All NNs have the same architecture, consisting of a single fully connected input layer, followed by two residual blocks. Each residual block contains two hidden layers with 16 neurons each, thus resulting in a total of 14676 trainable parameters. The layer by layer structure is $(d+2, 16) \to [(16, 16) \to (16, 16)] \to [(16, 16) \to (16, 16)] \to (16, 3)$, where d is the dimensionality of the manifold upon which the NN is defined, and 2 stands for the number of parameters. A detailed overview is given in Table 3. The activation function used is the squared ReLU $\sigma(x) = \max(x^2, 0)$. The NNs are trained for 64 epochs, utilizing 8×10^6 Monte Carlo sampling points per epoch, divided into 1000 batches. The Adam optimizer is used. The training process is executed on an Nvidia RTX 4090 laptop GPU, with a total runtime of 13 min. For the reference solution, we employ the CalculiX finite element solver [75], using a first-order tetrahedral mesh consisting of 770,743 elements.

The results are presented in Figs. 16, 17, 18, where both the deformed configurations and the pointwise

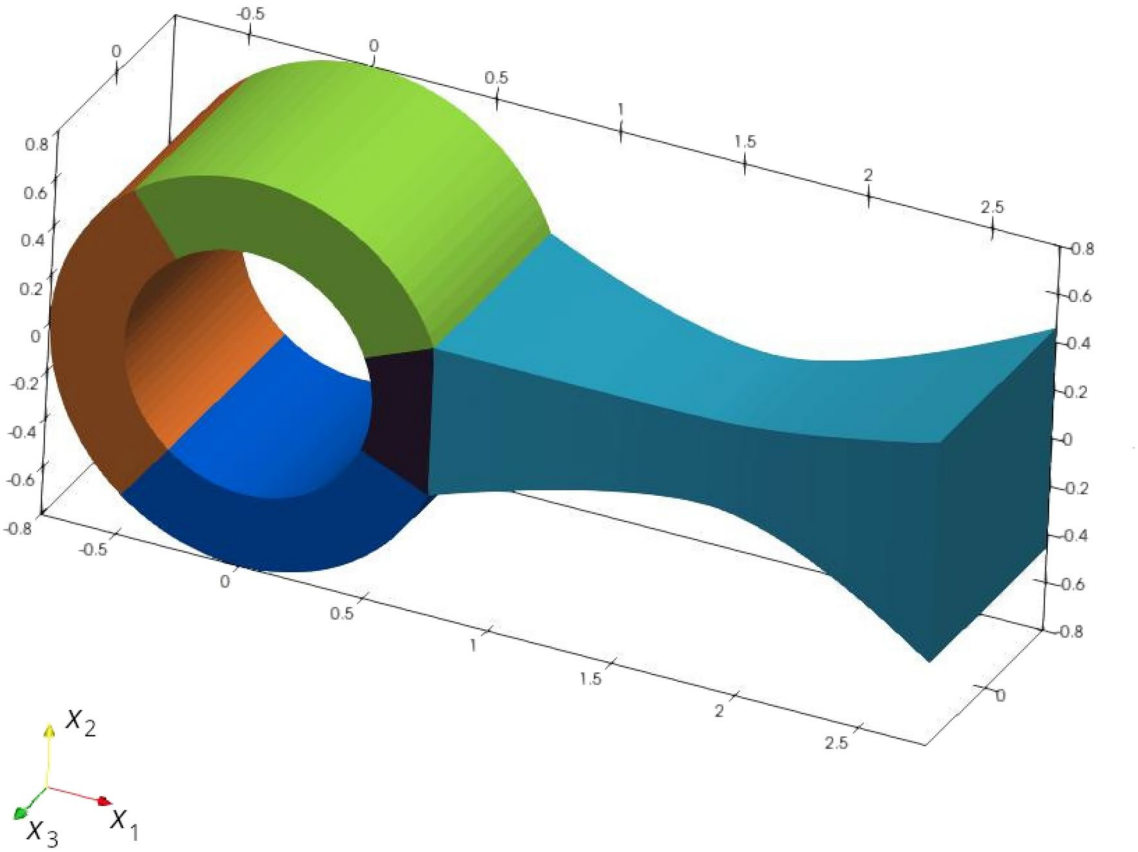


Fig. 14 Geometry of the mechanical holder model, parametrized with five NURBS patches, each shown with a different color. All dimensions are expressed in mm

Table 3 Distribution of NNs and trainable parameters for the mechanical holder model

	Number of NNs	Number of trainable parameters
Domain NNs (3D)	5	1235
Interface NNs (2D)	5	1219
Interface NNs (1D)	2	1203
Total	12	14,676

Table 4 Relative L^2 error for different combinations of ϕ_1 and ϕ_2

ϕ_1	ϕ_2	$\epsilon_{\text{rel},L^2}$
-1	-1	0.0136
-1	1	0.0264
1	-1	0.0266

error fields are shown for the parameter combinations $(\phi_1, \phi_2) \in \{(-1,-1),(-1,1),(1,-1)\}$. All shown results correspond to a 2D slice at $x_3 = 0$. The case $(\phi_1, \phi_2) = (1,1)$ represents a configuration with no contact and where no deformation exists. As a parameter dependent error metric, we use the error metric defined in (30), where $\boldsymbol{u}$ represents the predicted displacement field and $\boldsymbol{u}^{\text{FEM}}$ is the reference solution. The error is evaluated over a Cartesian grid defined on each individual NURBS patch. The number of grid points is $N = 75000$, corresponding to a uniform grid on the reference domain. The error values for the different parameter combinations of ϕ_1 and ϕ_2 are given in Table 4. The behavior of the error during training is shown in Fig. 15.

6 Conclusion

This paper presented a novel methodology combining PINNs with the IGA framework tailored for complex, multi-patch domains common in CAD. Our core contribution lies in formulating the PINN approximation within the reference domain of individual patches, ensuring geometric fidelity. Solution continuity across patch interfaces was achieved through the introduction of dedicated interface NNs, while the strong enforcement of Dirichlet BCs was realized via a specific NN output ansatz, drawing parallels to domain decomposition techniques similar to FETI and mortaring methods.

The efficacy and versatility of the developed multi-patch isogeometric-mapped neural solver was validated through complex numerical examples related to real-world applications. We successfully applied the framework to both 2D magnetostatic simulations, involving simple and complex quadrupole magnet geometries, and a 3D nonlinear solid mechanics problem featuring hyperelastic material behavior, contact BCs, and parametric dependencies. Quantitative numerical error analyses against reference solutions demonstrated the accuracy and robustness of the proposed method.

This work contributes towards bridging the gap between native CAD representations and mesh-free simulation capabilities offered by PINNs. By effectively handling complex multi-patch geometries and ensuring strong enforcement of essential boundary and interface conditions, our framework overcomes key limitations of standard PINNs and provides a powerful tool for analyzing complex physical systems directly within their design context.

Naturally, the suggested neural solver comes with a set of limitations, which we plan to address in future studies. First, suitable NN architectures are selected empirically and by trial and error. Since the focus of this work is on embedding

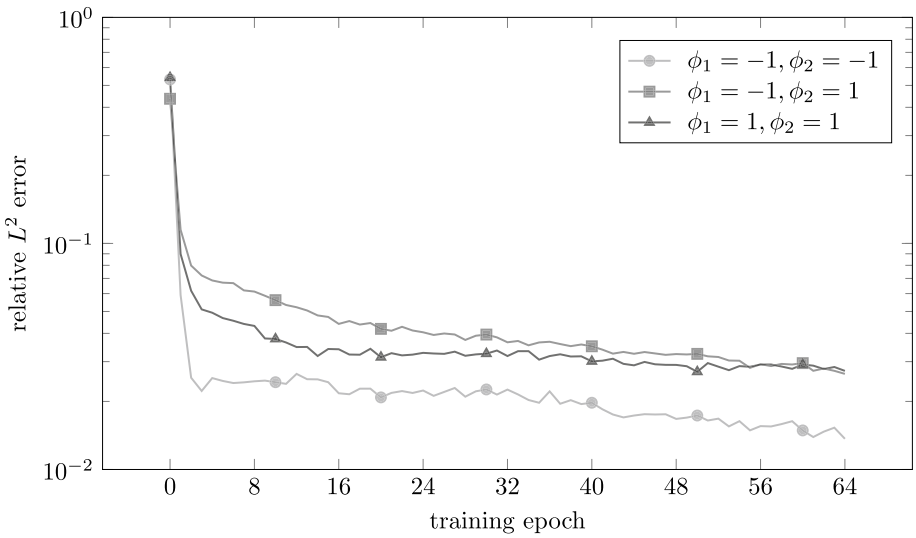
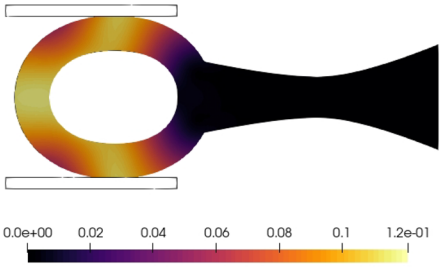
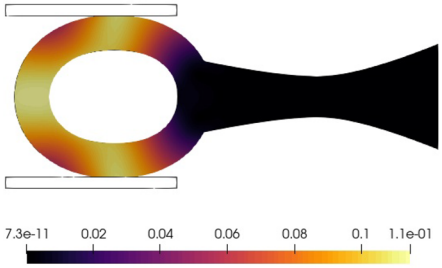


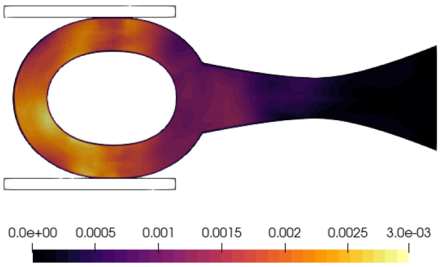
Fig. 15 Relative L^2 error during training for the entire computational domain and for different combinations of the parameters ϕ_1, ϕ_2



(a) PINN solution for $\phi_1 = -1, \phi_2 = -1$.

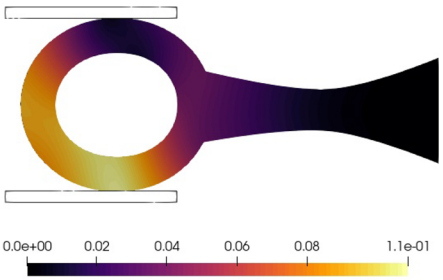


(b) Reference FEM solution for $\phi_1 = -1, \phi_2 = -1$.

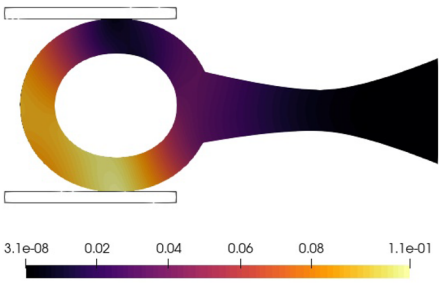


(c) Pointwise error for $\phi_1 = -1, \phi_2 = -1$.

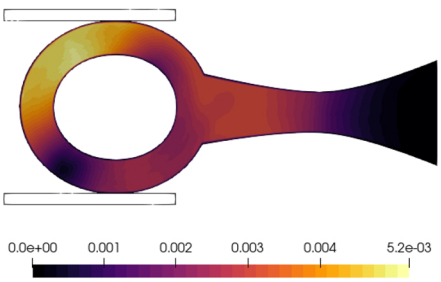
Fig. 16 Field solutions and corresponding error field for the mechanical holder model with contact parameters $\phi_1 = -1$, $\phi_2 = -1$. A 2D slice at $x_3 = 0$ is used for the visualization



(a) PINN solution for $\phi_1 = -1, \phi_2 = 1$.



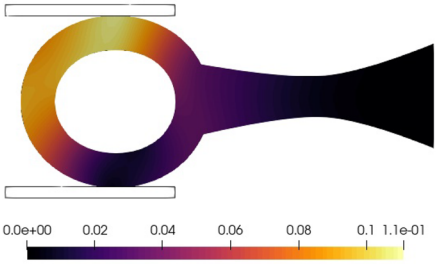
(b) Reference FEM solution for $\phi_1 = -1, \phi_2 = 1$.



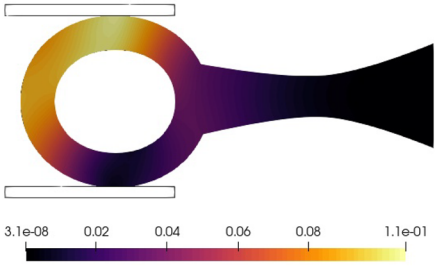
(c) Pointwise error for $\phi_1 = -1, \phi_2 = 1$.

Fig. 17 Field solutions and corresponding error field for the mechanical holder model with contact parameters $\phi_1 = -1$, $\phi_2 = 1$. A 2D slice at $x_3 = 0$ is used for the visualization

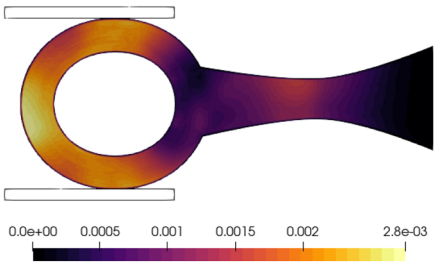
PINNs into the IGA framework, a systematic study on specific NN architectures remains out of scope, leaving many questions regarding design and optimization open for future research. Second, the neural solver is also not yet competitive to traditional numerical solution methods in terms of computational efficiency, due to the required NN training times. Additional limitations related to those encountered in conventional IGA also exist. For example, the suggested neural solver requires interface-conforming patches, for which no automatic generation procedure exists, even in



(a) PINN solution for $\phi_1 = 1, \phi_2 = -1$.



(b) Reference FEM solution for $\phi_1 = 1, \phi_2 = -1$.



(c) Pointwise error for $\phi_1 = 1, \phi_2 = -1$.

Fig. 18 Field solutions and corresponding error field for the mechanical holder model with contact parameters $\phi_1 = 1, \phi_2 = -1$. A 2D slice at $x_3 = 0$ is used for the visualization

conventional IGA. Extensions to non-conforming patches (e.g., non-matching knot vectors or hanging nodes) should be pursued. Moreover, the imposition of the Dirichlet BC still lacks the Kronecker delta property, which is a known limitation of IGA compared to classical FEM. In the same vein, the current approach assumes Dirichlet boundaries that correspond to parametric boundaries of the reference domain. As such, it cannot accommodate the case of trimmed boundaries, which is often encountered in industrial design settings [76, 77]. Integrating Nitsche's method [30] to the neural solver could provide a remedy. Last, the neural solver is currently limited to enforcing C^0-continuity only. Extensions to higher regularities remain to be addressed.

In addition to resolving the aforementioned limitations, the generalization of the suggested neural solver to parametric problem formulations, in particular high-dimensional ones, constitutes an open and highly relevant future research direction. From the perspective of the authors, research on parametric problems that leverages optimized, problem-specific NN architectures holds significant potential to surpass conventional methods, similar to the successes of NNs in fields such as image classification and natural language processing. A particularly promising direction is the integration of operator learning methods, such as deep operator networks, to learn a geometry-to-solution operator [78] that maps patch geometries and boundary information to the corresponding field solution, enabling a single model to generalize across multiple isogeometric patches and geometrical deformations.

Author Contributions MvT and IGI developed the methodology, implemented the software, and performed all simulations and numerical studies. DL acquired funding and provided supervision. All authors contributed to writing and reviewing the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability The data and software used to generate the results reported in this paper are available in a dedicated online repository, namely: https://github.com/ion-g-ion/Paper-IGA-PINNs.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward

and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707
2. Anitescu C, İsmail Ateş B, Rabczuk T (2023) Physics-informed neural networks: theory and applications. Machine learning in modeling and simulation: methods and applications. Springer, Cham, pp 179–218
3. Kim D, Lee J (2024) A review of physics informed neural networks for multiscale analysis and inverse problems. Multiscale Sci Eng 6(1):1–11
4. EW, Yu B, (2018) The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun Math Stat 6(1):1–12
5. Sirignano J, Spiliopoulos K (2018) Dgm: a deep learning algorithm for solving partial differential equations. J Comput Phys 375:1339–1364
6. Yang L, Meng X, Karniadakis GE (2021) B-PINNs: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. J Comput Phys 425:109913
7. Kharazmi E, Zhang Z, Karniadakis GE (2021) hp-VPINNs: variational physics-informed neural networks with domain decomposition. Comput Methods Appl Mech Eng 374:113547
8. Chiu P-H, Wong JC, Ooi C, Dao MH, Ong Y-S (2022) CAN-PINN: a fast physics-informed neural network based on coupled-automatic-numerical differentiation method. Comput Methods Appl Mech Eng 395:114909
9. Zhu Z, Hao J, Huang J, Huang B (2023) BC-PINN: an adaptive physics informed neural network based on biased multiobjective coevolutionary algorithm. Neural Comput Appl 35(28):21093–21113
10. Su C, Liang J, He Z (2024) E-PINN: a fast physics-informed neural network based on explicit time-domain method for dynamic response prediction of nonlinear structures. Eng Struct 321:118900
11. Khan A, Lowther DA (2022) Physics informed neural networks for electromagnetic analysis. IEEE Trans Magn 58(9):1–4
12. Beltrán-Pulido A, Bilionis I, Aliprantis D (2022) Physics-informed neural networks for solving parametric magnetostatic problems. IEEE Trans Energy Convers 37(4):2678–2689
13. Baldan M, Di Barba P, Lowther DA (2023) Physics-informed neural networks for inverse electromagnetic problems. IEEE Trans Magn 59(5):1–5
14. Lippert JR, Tresckow M, De Gersem H, Loukrezis D (2024) Transfer learning-based physics-informed neural networks for magnetostatic field simulation with domain variations. Int J Numer Model Electron Netw Dev Fields 37(4):3264
15. Cai S, Wang Z, Wang S, Perdikaris P, Karniadakis GE (2021) Physics-informed neural networks for heat transfer problems. J Heat Transf 143(6):060801
16. Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE (2021) Physics-informed neural networks (pinns) for fluid mechanics: a review. Acta Mech Sin 37(12):1727–1738
17. Eivazi H, Tahani M, Schlatter P, Vinuesa R (2022) Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. Phys Fluids 34(7)
18. Zhao C, Zhang F, Lou W, Wang X, Yang J (2024) A comprehensive review of advances in physics-informed neural networks and their applications in complex fluid dynamics. Phys Fluids 36(10)
19. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh VM, Guo H, Hamdia K, Zhuang X, Rabczuk T (2020) An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Comput Methods Appl Mech Eng 362:112790
20. Bai J, Rabczuk T, Gupta A, Alzubaidi L, Gu Y (2023) A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics. Comput Mech 71(3):543–562
21. Linden L, Klein DK, Kalina KA, Brummund J, Weeger O, Kästner M (2023) Neural networks meet hyperelasticity: a guide to enforcing physics. J Mech Phys Solids 179:105363
22. Hu H, Qi L, Chao X (2024) Physics-informed neural networks (PINNs) for computational solid mechanics: numerical frameworks and applications. Thin-Wall Struct:112495
23. Markidis S (2021) The old and the new: can physics-informed deep-learning replace traditional linear solvers? Front Big Data 4:669097
24. Grossmann TG, Komorowska UJ, Latz J, Schönlieb C-B (2024) Can physics-informed neural networks beat the finite element method? IMA J Appl Math 89(1):143–174
25. Hughes TJ, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput Methods Appl Mech Eng 194(39–41):4135–4195
26. Cottrell JA, Hughes TJ, Bazilevs Y (2009) Isogeometric analysis: toward integration of cad and fea. Wiley, Hoboken
27. Nguyen VP, Anitescu C, Bordas SP, Rabczuk T (2015) Isogeometric analysis: an overview and computer implementation aspects. Math Comput Simul 117:89–116
28. Farhat C, Roux F-X (1991) A method of finite element tearing and interconnecting and its parallel solution algorithm. Int J Numer Meth Eng 32(6):1205–1227
29. Kleiss SK, Pechstein C, Jüttler B, Tomar S (2012) IETI - Isogeometric tearing and interconnecting. Comput Methods Appl Mech Eng 247:201–215
30. Apostolatos A, Schmidt R, Wüchner R, Bletzinger K-U (2014) A Nitsche-type formulation and comparison of the most common domain decomposition methods in isogeometric analysis. Int J Numer Meth Eng 97(7):473–504
31. Brivadis E, Buffa A, Wohlmuth B, Wunderlich L (2015) Isogeometric mortar methods. Comput Methods Appl Mech Eng 284:292–319
32. Merkel M, Kapidani B, Schöps S, Vázquez R (2022) Torque computation with the isogeometric mortar method for the simulation of electric machines. IEEE Trans Magn 58(9):1–4
33. Khodayi-Mehr R, Zavlanos M (2020) VarNet: variational neural networks for the solution of partial differential equations. In: Learning for dynamics and control, pp 298–307 (PMLR)
34. Goswami S, Anitescu C, Chakraborty S, Rabczuk T (2020) Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theoret Appl Fract Mech 106:102447
35. Haghighat E, Raissi M, Moure A, Gomez H, Juanes R (2021) A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comput Methods Appl Mech Eng 379:113741
36. Cao G, Wang X (2025) Parametric extended physics-informed neural networks for solid mechanics with complex mixed boundary conditions. J Mech Phys Solids 194:105944
37. von Tresckow M, Kurz S, De Gersem H, Loukrezis D (2022) A neural solver for variational problems on CAD geometries with application to electric machine simulation. J Mach Learn Model Comput 3(1):49–75
38. Kostas KV, Politis CG, Zhanabay I, Kaklis PD (2024) A physics-informed parametrization and its impact on 2D IGABEM analysis. Eng Comput:1–20
39. Möller M, Toshniwal D, Ruiten F (2021) Physics-informed machine learning embedded into isogeometric analysis. In: Mathematics: key enabling technology for scientific machine learning. Platform Wiskunde, Amsterdam, pp 57–59
40. Prasad AD, Balu A, Shah H, Sarkar S, Hegde C, Krishnamurthy A (2022) Nurbs-diff: a differentiable programming module for nurbs. Comput Aid Des 146:103199
41. Gasick J, Qian X (2023) Isogeometric neural networks: a new deep learning approach for solving parameterized partial differential equations. Comput Methods Appl Mech Eng 405:115839

42. Nath D, Neog DR, Gautam SS (2025) Transfer learning enhanced deep neural network application in Gauss quadrature for isogeometric analysis. Eng Appl Artif Intell 145:110182
43. Saidaoui H, Espath L, Tempone R (2024) Deep NURBS-admissible physics-informed neural networks. Eng Comput:1–15
44. Li A, Zhang YJ (2023) Isogeometric analysis-based physics-informed graph neural network for studying traffic jam in neurons. Comput Methods Appl Mech Eng 403:115757
45. Xu G, Xie J, Zhong W, Toyoura M, Ling R, Xu J, Gu R, Wang CC, Rabczuk T (2025) IGA-Graph-Net: isogeometric analysis-reuse method based on graph neural networks for topology-consistent models. J Comput Phys 521:113544
46. Wang D, Xu J, Gao F, Wang CC, Gu R, Lin F, Rabczuk T, Xu G (2022) IGA-Reuse-NET: a deep-learning-based isogeometric analysis-reuse approach with topology-consistent parameterization. Comput Aid Geometr Des 95:102087
47. Lu Y, Li H, Zhang L, Park C, Mojumder S, Knapik S, Sang Z, Tang S, Apley DW, Wagner GJ et al (2023) Convolution hierarchical deep-learning neural networks (c-hidenn): finite elements, isogeometric analysis, tensor decomposition, and beyond. Comput Mech 72(2):333–362
48. Zhang L, Park C, Lu Y, Li H, Mojumder S, Saha S, Guo J, Li Y, Abbott T, Wagner GJ et al (2023) Isogeometric convolution hierarchical deep-learning neural network: isogeometric analysis with versatile adaptivity. Comput Methods Appl Mech Eng 417:116356
49. Zhang L, Park C, Hughes TJ, Liu WK (2025) Multi-patch isogeometric convolution hierarchical deep-learning neural network. Comput Methods Appl Mech Eng 434:117582
50. Jagtap AD, Karniadakis GE (2020) Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun Comput Phys 28(5)
51. Jagtap AD, Kharazmi E, Karniadakis GE (2020) Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput Methods Appl Mech Eng 365:113028
52. Zhang B, Wu G, Gu Y, Wang X, Wang F (2022) Multi-domain physics-informed neural network for solving forward and inverse problems of steady-state heat conduction in multilayer media. Phys Fluids 34(11)
53. Sarma AK, Roy S, Annavarapu C, Roy P, Jagannathan S (2024) Interface pinns (I-PINNs): a physics-informed neural networks framework for interface problems. Comput Methods Appl Mech Eng 429:117135
54. Seroussi I, Miron A, Ringel Z (2024) Spectral-bias and kernel-task alignment in physically informed neural networks. Mach Learn Sci Technol 5(3):035048
55. Chai X, Cao W, Li J, Long H, Sun X (2024) Overcoming the spectral bias problem of physics-informed neural networks in solving the frequency-domain acoustic wave equation. IEEE Trans Geosci Remote Sens 62:1–20
56. Sukumar N, Srivastava A (2022) Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. Comput Methods Appl Mech Eng 389:114333
57. Wang J, Mo Y, Izzuddin B, Kim C-W (2023) Exact Dirichlet boundary physics-informed neural network EPINN for solid mechanics. Comput Methods Appl Mech Eng 414:116184
58. Luong KA, Le-Duc T, Lee S, Lee J (2024) A novel normalized reduced-order physics-informed neural network for solving inverse problems. Eng Comput 40(5):3253–3272
59. Le-Duc T, Lee S, Nguyen-Xuan H, Lee J (2024) A hierarchically normalized physics-informed neural network for solving differential equations: application for solid mechanics problems. Eng Appl Artif Intell 133:108400
60. Deguchi S, Asai M (2023) Dynamic & norm-based weights to normalize imbalance in back-propagated gradients of physics-informed neural networks. J Phys Commun 7(7):075005
61. Tresckow M, De Gersem H, Loukrezis D (2024) Error approximation and bias correction in dynamic problems using a recurrent neural network/finite element hybrid model. Appl Math Model 129:428–447
62. Xu S, Dai Y, Yan C, Sun Z, Huang R, Guo D, Yang G (2025) On the preprocessing of physics-informed neural networks: how to better utilize data in fluid mechanics. J Comput Phys:113837
63. Evans LC (2022) Partial differential equations, vol 19. American Mathematical Society, Providence
64. Bottou L et al (1991) Stochastic gradient learning in neural networks. Proc Neuro-Nımes 91(8):12
65. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: 3rd International conference for learning representations. San Diego, pp 1–10
66. Piegl L, Tiller W (2012) The NURBS book. Springer, Berlin
67. Buffa A, Giannelli C (2016) Adaptive isogeometric methods with hierarchical splines: error estimator and convergence. Math Models Methods Appl Sci 26(01):1–25
68. Buffa A, Sangalli G, Vázquez R (2010) Isogeometric analysis in electromagnetics: B-splines approximation. Comput Methods Appl Mech Eng 199(17–20):1143–1152
69. Jackson JD (2021) Classical electrodynamics. Wiley, Hoboken
70. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
71. Alnaes MS, Logg A, Ølgaard KB, Rognes ME, Wells GN (2014) Unified form language: a domain-specific language for weak formulations of partial differential equations. ACM Trans Math Softw 40
72. Logg A, Wells GN (2010) Dolfin: automated finite element computing. ACM Trans Math Softw. https://doi.org/10.1145/1731022.1731030
73. Bonet J, Gil AJ, Wood RD (2021) Nonlinear solid mechanics for finite element analysis: dynamics. Cambridge University Press, Cambridge
74. Wriggers P, Laursen TA (2006) Computational contact mechanics, vol 2. Springer, Berlin
75. Dhondt G (2004) The finite element method for three-dimensional thermomechanical applications. Wiley, Chichester
76. Breitenberger M, Apostolatos A, Philipp B, Wüchner R, Bletzinger K-U (2015) Analysis in computer aided design: nonlinear isogeometric b-rep analysis of shell structures. Comput Methods Appl Mech Eng 284:401–457
77. Teschemacher T, Bauer AM, Oberbichler T, Breitenberger M, Rossi R, Wüchner R, Bletzinger K-U (2018) Realization of cad-integrated shell simulation based on isogeometric b-rep analysis. Adv Model Simul Eng Sci 5(1):19
78. Backmeyer M, Kurz S, Möller M, Schöps S (2024) Solving electromagnetic scattering problems by isogeometric analysis with deep operator learning. In: 2024 Kleinheubach conference, pp 1–4