Engineering with Computers (2026) 42:102
https://doi.org/10.1007/s00366-026-02312-6

ORIGINAL ARTICLE

IGANets: Isogeometric analysis networks and their applications to linear structural analysis problems

Matthias Möller[1] · Günther Obermair[2] · Isabella Singer[2] · Christian Gollmann[2] · Alessandro Reali[3] · Stefanie Elgeti[2]

Received: 30 December 2025 / Accepted: 17 March 2026

Abstract

Fast numerical predictions have become an indispensable component of modern engineering design workflows, whether in interactive design within computer-aided design (CAD) environments or in multi-query numerical tasks such as design optimization and uncertainty quantification. Depending on the context, "fast" may refer to near real-time predictions within a few seconds, or simply to methods that are significantly faster than high-fidelity simulations, for example those based on the finite element method (FEM). With the aim of providing a tool that not only enables such accelerated predictions but also integrates seamlessly into established workflows, we introduce the concept of IGANets. IGANets are spline-based, physics-informed machine learning models that can be integrated naturally between CAD representations and numerical analysis tools, particularly those based on isogeometric analysis (IGA). Unlike purely data-driven approaches, IGANets do not inherently rely on precomputed training data; instead, they are formulated in a collocation setting directly from physical models. In this paper, we present the IGANets concept and demonstrate its feasibility through numerical experiments for the Poisson equation and linear elasticity. In addition, we investigate a multi-instance linear-elasticity setting with varying I-beam-like geometries and boundary conditions in order to assess the generalization capability of the framework. The results show that IGANets can predict solutions for previously unseen problem instances within the training range with improved accuracy as the number of training samples increases.

Keywords Physics-informed machine learning · Surrogate models · Real-time design · Isogeometric collocation

1 Introduction

In modern engineering design workflows, numerical analysis plays a crucial role in providing insight into the performance of engineering components even before fabrication. Conceptually, this can be described as the creation of a digital twin prototype – a virtual representation of the physical component that allows for comprehensive investigation, evaluation, and optimization prior to manufacturing. Conventional numerical methods, such as finite element analysis, are often computationally intensive and time-consuming, which prevents their integration into real-time, interactive design processes. By "real-time," we refer to timescales compatible with the designer's workflow, typically in the order of seconds or less per interaction.

Within this context, this paper introduces IGANet, a novel methodology aimed at enabling real-time digital twin prototyping with particular emphasis on geometry-driven design and varying loading or usage scenarios. The proposed approach empowers designers to obtain immediate

✉ Matthias Möller
m.moller@tudelft.nl

Günther Obermair
guenther.obermair@tuwien.ac.at

Isabella Singer
isabella.singer@tuwien.ac.at

Christian Gollmann
christian.gollmann@tuwien.ac.at

Alessandro Reali
alessandro.reali@unipv.it

Stefanie Elgeti
stefanie.elgeti@tuwien.ac.at

[1] Department of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

[2] Institute of Lightweight Design and Structural Biomechanics, TU Wien, Gumpendorferstr. 7, 1070 Vienna, Austria

[3] Department of Civil Engineering and Architecture, University of Pavia, via A.Ferrata 3, 27100 Pavia, Italy

Published online: 11 May 2026

feedback on the physical behavior of design artifacts as they modify geometry or boundary conditions, thereby fostering an interactive and informed design process. IGANets combine the mathematical framework of isogeometric analysis (IGA) with the concept of physics-informed neural operators. Parametric objects (e.g., geometries, boundary conditions, materials) are encoded through their coefficients relative to fixed spline bases and provided as inputs to a neural network. The network outputs the solution to parametric partial differential equations (PPDE) in the form of spline coefficients. Building on the theory of isogeometric collocation methods, IGANets yield converged solutions upon increasing the number of unknowns alongside the sampling points. The physics model is implemented in the loss function in a least-squares sense, thereby leveraging the higher continuity of splines to represent higher derivatives without the need to introduce auxiliary variables. IGANets are trained over a family of inputs that represent relevant instances of the expected design space. The training can be performed in full self-supervised manner or assisted by available simulation/experimental data, thereby overcoming the major roadblocks of existing methods. The ultimate goal of this work is to bridge the gap between high-fidelity physical simulation and real-time interactivity, thereby facilitating both manual and automated optimization processes within digital design environments. The comparison of IGANets with the state of the art is discussed in the following.

A digital twin prototype (DTP) is a virtual representation or simulation of a physical object, system, or process enabled through digital technologies, i.e., computer-aided design (CAD) and analysis (CAA). One way in which a DTP can be obtained are surrogate modelling techniques. The generation of such surrogate models is based on the fast generation of accurate and reliable solutions of parametric partial differential equations (PPDE), where the considered parameters can be geometric shapes, material parameters, etc. Surrogate modelling approaches suitable for design processes can be grouped as follows ([1]): (1) dimensional/analytical model reduction (e.g., 3D-to-2D or 2.5D reduction by exploiting symmetries, replacement of nonlinear by linear model equations); (2) computational/model-order reduction (e.g., reduced basis (RB) method or proper orthogonal decomposition (POD)); (3) physics-informed machine learning (e.g., physics-informed neural networks (PINN), deep operator networks (DeepONets)).

IGANets fall into the last category, so in the following, we will briefly describe seminal technologies in the field of physics-informed machine learning and then give an overview over contributions within the frame of linear elasticity. Within this domain, one method stands out as a particularly natural fusion of machine learning and scientific computing – physics-informed neural networks (PINNs). Although the foundational ideas can be traced back to early work in the 1990s [2–4], PINNs only gained widespread attention and rapid development following their reinvention by Raissi, Perdikaris, and Karniadakis in 2017 [5]. The core concept of PINNs is to approximate the solution of physical problems governed by (partial) differential equations. In contrast to conventional supervised learning approaches that rely solely on input-output data pairs, PINNs incorporate the governing physical laws directly into the learning process by embedding the differential equations into the loss function of the neural network. This integration enables the model to learn physically consistent solutions, even in the absence of extensive labeled data. In the context of linear elasticity, PINNs have for example been employed in the following works [6–12].

Instead, deep operator networks (DeepONets) are a class of neural network architectures designed to learn nonlinear operators – that is, mappings between infinite-dimensional function spaces. Unlike PINNs that approximate functions, DeepONets aim to approximate operators, such as the solution map of a differential equation. Introduced by Lu et al. [13, 14], a DeepONet consists of two subnetworks: a branch network, which processes input functions (typically represented by pointwise evaluations), and a trunk network, which handles the evaluation locations. Given their superior generalization capabilities as compared to PINNs, DeepONets are particularly well-suited for solving families of parametric PDEs. In the field of linear elasticity, DeepONets have for example been applied in the following works [15, 16].

In addition to these two main variants of physics-informed machine learning, a wide range of alternative approaches has emerged, offering diverse strategies for integrating physical knowledge into learning frameworks. A comprehensive review of these methods can be found in [17].

Another central concept in IGANets are collocation approaches. Collocation methods solve PDEs by enforcing the governing equations at a discrete set of points, known as collocation points [18]. These methods originated in the 1960s and 70s as a way to solve boundary value problems for ordinary and partial differential equations [19–21], offering a simpler alternative to weighted residual or Galerkin methods by avoiding domain integration. A modern and efficient extension is spline-based collocation [22–26], which integrates collocation with the spline-based geometry representation used in CAD; thereby exploiting the smoothness and geometric exactness of spline basis functions. A

further development is least-squares collocation [27–29], where the differential equation is not enforced exactly at each collocation point but rather in a least-squares sense by minimizing the squared residuals over all points. This relaxation improves robustness, particularly when dealing with noisy data, over-constrained systems, or ill-posed problems.

In the remainder of the paper, the IGANets approach is introduced in Section 2, followed by a concept validation based on the Poisson equation in Section 3. The application of IGANets to linear elasticity is demonstrated in Section 4.

2 The IGANets approach

To introduce the IGANets approach, we consider the abstract boundary-value problem (BVP)

$$\mathrm{x} \in \Omega : \quad \mathcal{R}_\Omega[u; f, \boldsymbol{\alpha}](\mathrm{x}) = 0, \tag{1}$$

$$\mathrm{x} \in \Gamma : \quad \mathcal{R}_\Gamma[u; g, \boldsymbol{\alpha}](\mathrm{x}) = 0, \tag{2}$$

where $\mathcal{R}_\Omega$ and $\mathcal{R}_\Gamma$ are the differential operators in residual form acting on the solution u in the domain Ω and its boundary $\Gamma = \partial\Omega$, respectively. Here, f and g denote possible right-hand sides and boundary values, and $\boldsymbol{\alpha}$ represents problem-specific parameters such as the Lamé parameters utilized in Section 4.1.3.

Simply put, the task of any numerical method for solving BVPs is to find a mapping

$$\mathcal{M} : (\Omega, \Gamma, f, g, \boldsymbol{\alpha}) \mapsto u \tag{3}$$

such that Equations (1)-(2) are satisfied for all x in the domain and its boundary.

Although traditional physics-informed machine learning approaches often hardcode the geometry, right-hand sides, and boundary values into the loss function and learn over a range of problem parameters $\boldsymbol{\alpha}$, the core idea of IGANets is to encode the entire problem set-up in terms of basis coefficients relative to a priori chosen bases. The basis coefficients are then fed into the neural network as inputs and the network outputs are reinterpreted as coefficients of the solution relative to an again a priori determined basis.

This is in stark contrast to classical PINNs, where the network input is a physical coordinate in the domain or at its boundary and the network's task is to predict the pointwise solution value at that location. Although this concept ensures a very small number of network inputs and outputs, it makes it nontrivial to provide complex problem set-ups (i.e., Ω, Γ, f, g) as network input so that even parameterized PINNs are often restricted to learn over only a few problem parameters $\boldsymbol{\alpha}$ that can be provided as scalar inputs.

The IGANets approach also differs from the concept of DeepONets, where the basis is not determined a priori but learned in a separate network (termed *branch network*). Although this concept is particularly interesting when the 'optimal' basis is not known a priori, e.g., when the solution has sharp localized features such as shock fronts, the IGANets approach is designed as a learning-based drop-in replacement for classical isogeometric analysis (IGA) and, as such, it builds on the idea of representing both the geometry and the solution in compatible spline spaces.

For simplicity, we illustrated the IGANets concept in two dimensions and for a scalar BVP but remark that it naturally generalizes to higher dimensions, vector-valued problems, and even non-tensor-product spline spaces. To begin with, let

$$\mathcal{S} = S^{p,c}(\Xi) \otimes S^{p,c}(\Xi) \tag{4}$$

denote the space of bivariate B-spline basis functions $b_i b_j \in \mathcal{S}$, each of degree p and continuity $c \leq p-1$, defined on the open knot vector

$$\Xi = [\underbrace{\xi_1 = \xi_2 = \cdots = \xi_{p+1}}_{p+1 \text{ times}}, \ldots, \xi_i, \ldots, \underbrace{\xi_n = \cdots = \xi_{n+p+1}}_{p+1 \text{ times}}] \tag{5}$$

consisting of $n+p+1$ knots – a sequence of non-decreasing real values from the interval [0, 1] – and, hence, n basis functions per dimension, respectively.

The exact geometry $\Omega \subset \mathbb{R}^2$ can then be discretized as

$$\boldsymbol{\xi} \in \hat{\Omega} : \quad \hat{\mathrm{x}}_h(\boldsymbol{\xi}) = \sum_{i=1}^{n}\sum_{j=1}^{n} b_i(\xi^{(1)}) b_j(\xi^{(2)}) \mathrm{x}_{ij}, \quad \mathrm{x}_{ij} \in \mathbb{R}^2, \tag{6}$$

which maps from the parametric domain $\hat{\Omega} = [0,1]^2$ to the physical domain Ω_h and its boundary Γ_h. If the forward mapping defined in (6) is bijective, its inverse, the so-called pull-back mapping, is well defined

$$\mathrm{x} \in \Omega_h : \quad \boldsymbol{\xi}_h(\mathrm{x}) = \hat{\mathrm{x}}_h^{-1}(\mathrm{x}). \tag{7}$$

The solution u, the right-hand side f and the boundary values g are approximated in the same spline space

$$\mathrm{x} \in \Omega_h : \quad u_h(\mathrm{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} b_i(\xi_h^{(1)}(\mathrm{x})) b_j(\xi_h^{(2)}(\mathrm{x})) u_{ij}, \qquad u_{ij} \in \mathbb{R}, \tag{8}$$

$$\mathrm{x} \in \Omega_h : \quad f_h(\mathrm{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} b_i(\xi_h^{(1)}(\mathrm{x})) b_j(\xi_h^{(2)}(\mathrm{x})) f_{ij}, \qquad f_{ij} \in \mathbb{R}, \tag{9}$$

$$\mathrm{x} \in \Gamma_h \; : \quad g_h^{\gamma}(\mathrm{x}) = \sum_{i=1}^{n} b_i(\xi_h^{d(\gamma)}(\mathrm{x}))g_i, \qquad g_i \in \mathbb{R}, \tag{10}$$

where $\xi_h^{(d)}$ denotes the dth parametric direction of the inverse map defined in (7), $\gamma \in \Gamma_h$ is a boundary segment, and $d(\gamma)$ is defined to select the spatial direction that corresponds to the boundary segment γ. As an example, a segment γ located at the upper or lower boundary of the unit square would be mapped to the first parametric direction $\xi_h^{(1)}$, whereas segments at the left and right boundary would be mapped to $\xi_h^{(2)}$. Let us remark that in principle each of the above quantities can be defined in a different, not necessarily tensor-product, spline space.

With the above definitions, the task of the IGANets (and a puristic IGA solver) is to find a mapping

$$\mathcal{M}_h \; : \; (\{\mathrm{x}_{ij}\}, \{f_{ij}\}, \{g_i\}, \boldsymbol{\alpha}) \mapsto \{u_{ij}\} \tag{11}$$

such that the discretized counterparts of Equations (1)-(2) are satisfied for a finite set of sampling points x_k in the domain and its boundary. This can be accomplished by concatenating all flattened[1] inputs to a single vector[2]

$$\text{input} := \text{concat}\left(\text{flat}\,\{\mathrm{x}_{ij}\}, \text{flat}\,\{f_{ij}\}, \text{flat}\,\{g_i\}, \text{flat}\,(\boldsymbol{\alpha})\right) \in \mathbb{R}^{3n^2+4n+m} \tag{12}$$

with m being the number of parameters $\boldsymbol{\alpha}$ and feeding it as input into a feed-forward neural network. The network's flat output is then reshaped[3] into an $n \times n$ matrix whose entries are the solution coefficients relative to the specified B-spline basis:

$$\{u_{ij}\} := \text{reshape}\,(\text{output}, n, n) \in \mathbb{R}^{n^2} \tag{13}$$

The network's loss function is defined as

$$\text{loss} = \text{loss}_{\text{pde}} + \text{loss}_{\text{bdr}}. \tag{14}$$

The previous (14) is comprised of the two components

$$\text{loss}_{\text{pde}} = \frac{\omega_\Omega}{N_\Omega} \sum_{k=1}^{N_\Omega} \left(\mathcal{R}_{\Omega_h}[u_h; f_h, \boldsymbol{\alpha}](\mathrm{x}_k)\right)^2, \tag{15}$$

$$\text{loss}_{\text{bdr}} = \frac{\omega_\Gamma}{N_\Gamma} \sum_{k=1}^{N_\Gamma} \left(\mathcal{R}_{\Gamma_h}[u_h; f_h, \boldsymbol{\alpha}](\mathrm{x}_k)\right)^2, \tag{16}$$

where N_Ω and N_Γ are the number of sampling points x_k in the domain and on the boundary, respectively, and ω_Ω and ω_Γ are weights to balance between the two components of the loss function. Inspired by the isogeometric collocation method by [23], we choose the sampling points as the images of the *Greville abscissae*

$$\bar{\xi}_k = \frac{\xi_{k+1} + \cdots + \xi_{k+p}}{p} \tag{17}$$

under the push-forward mapping, i.e., $\mathrm{x}_k = \hat{\mathrm{x}}_h(\bar{\boldsymbol{\xi}}_k)$, or that of a virtually refined spline space leading to a learning-based counterpart of the isogeometric least-squares collocation method by [29]. In both cases, some parts of the loss function simplify, e.g., the basis functions in Equations (9)-(10) can be evaluated at $\bar{\boldsymbol{\xi}}_k$ directly.

As an alternative to the weak imposition of boundary conditions through residual terms in the loss function, Dirichlet boundary conditions can be imposed strongly by either overwriting the corresponding entries of the network output by the coefficients of the Dirichlet values and excluding the corresponding terms in the loss functions or by eliminating the solution coefficients corresponding to Dirichlet values from the network output and augmenting the reduced output by the prescribed coefficients afterwards. Both approaches overcome the general challenge of physics-enhanced neural networks to enforce Dirichlet boundary conditions accurately through the careful tuning of the weighting coefficient ω_Γ, as for example seen for PINNs in [30]. The latter approach moreover leads to slightly smaller networks but requires some bookkeeping in the reinterpretation of the network output as spline function, cf. (13).

It is furthermore possible to augment this purely physics-informed loss function by a supervised data component, e.g.,

$$\text{loss}_{\text{data}} = \frac{\omega_{\text{data}}}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} \left(u_h(\mathrm{x}_k) - u_{\text{data}}(\mathrm{x}_k)\right)^2. \tag{18}$$

The overall working principle of IGANets is depicted in Fig. 1, whereby $N = n^2$.

[1] Mathematically, the operation *flat* corresponds to the standard **vec**torization operation for matrix $\{g_{ij}\}$ and its tensorial counterpart for the tensor $\{\mathrm{x}_{ij}\}$. Whether row- or column major flattening is performed is irrelevant since the main purpose of flattening is to generate a single input vector for the neural network.

[2] The term $3n^2$ comprises the n^2 vector-valued coefficients $\{\mathrm{x}_{ij}\}_{i,j=1}^n$ of the two-dimensional geometry map x_h and the scalar coefficients $\{f_{ij}\}_{i,j=1}^n$ of the right-hand side f. The term $4n$ results from the scalar coefficients $\{g_i\}_{i=1}^n$ representing the boundary values g_h at the four sides of the boundary Γ.

[3] Mathematically, the *reshape*$(\cdot, n, n)$ operation is the inverse of the *flat* operation as it reinterprets the values of the flat vector as $n \times n$ matrix.

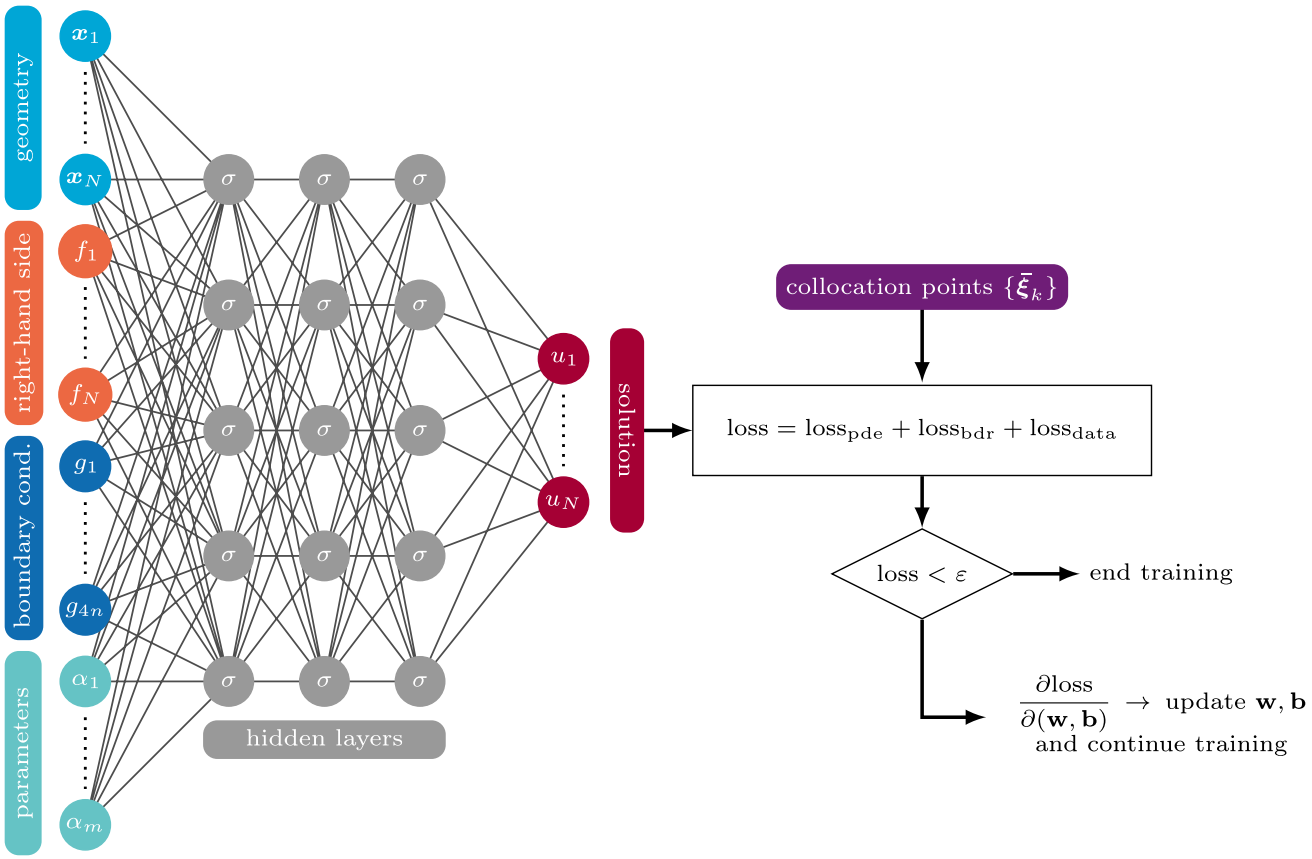


Fig. 1 General working principle of IGANets

3 Concept validation for the Poisson problem

In this section, we demonstrate the correct functioning of the proposed IGANets concept for the two-dimensional Poisson problem solved on the unit square $\Omega = [0,1]^2$ with homogeneous Dirichlet boundary conditions, i.e.,

$$\Delta u = f \quad \text{in } \Omega, \tag{19}$$

$$u = 0 \quad \text{on } \Gamma. \tag{20}$$

The right-hand side is set to

$$f(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y) \tag{21}$$

so that the exact solution equals

$$u(x,y) = \sin(\pi x)\sin(\pi y). \tag{22}$$

Let u and f be approximated in the spline space $S^{p,p-1}(\Xi) \otimes S^{p,p-1}(\Xi)$, where Ξ is an open uniform knot vector consisting of n equidistant knots. The coefficients f_{ij} in (9) are obtained via B-spline interpolation at the Greville abscissae $\{\bar{\boldsymbol{\xi}}_k\}_k$.

Putting it all together, the IGANets' loss function reads

$$\text{loss} = \frac{\omega_\Omega}{N_\Omega}\sum_{k=1}^{N_\Omega}\left(\Delta u_h(\bar{\boldsymbol{\xi}}_k) - f_h(\bar{\boldsymbol{\xi}}_k)\right)^2 + \frac{\omega_\Gamma}{N_\Gamma}\sum_{k=1}^{N_\Gamma}\left(u_h(\bar{\boldsymbol{\xi}}_k)\right)^2, \tag{23}$$

where the homogeneous Dirichlet values simplify the boundary loss term. Since IGANets adopt the classical FEM/IGA paradigm for computing derivatives of their outputs with respect to physical variables, derivative evaluation reduces to the appropriate differentiation of the basis functions, e.g.,

$$\Delta u_h(\boldsymbol{\xi}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(b_i''(\xi^{(1)})b_j(\xi^{(2)}) + b_i(\xi^{(1)})b_j''(\xi^{(2)})\right)u_{ij}. \tag{24}$$

It is worth noting that this stands in contrast to PINNs, DeepONets, and other physics-informed machine learning approaches for differential equations that compute derivatives of the network outputs with the help of algorithmic differentiation (AD). The involved computational costs and

memory consumption especially for higher-order derivatives are considered one of the major limitations of AD-based approaches, whereas the computational costs and memory consumption for computing spatial derivatives in IGANets are comparable to that of classical FEM/IGA formulations.

Consequently, computationally expensive and memory consuming algorithmic differentiation is only used for computing the derivatives of the loss function with respect to the network weights. As such, the computational costs are independent of the order of the differential equations, which is not the case in other physics-informed machine learning approaches. For an efficient algorithm to evaluate multivariate B-spline basis functions and their derivatives, the reader is referred to Appendix A.

A direct comparison between IGANets and a regular collocation IGA method is given in Tables 1 and 2 for bi-cubic and bi-quartic tensor-product B-splines, respectively. From left to right, the columns indicate the number of coefficients/basis functions (#coeffs), the number of network weights (#weights), the number of epochs until loss-value stagnation (#epochs), the mean-squared loss value ($\mathrm{MSE}_{\mathrm{loss}}$), the mean-squared error between the IGANets' solution and the exact solution interpolated in the spline space ($\mathrm{MSE}_{\mathrm{IGANets}}$), the number of BiCGStab iterations of the collocation IGA method (#iter), and the mean-squared error between the collocation IGA solution and the exact solution interpolated in the spline space ($\mathrm{MSE}_{\mathrm{C\text{-}IGA}}$). The network is initialized with Xavier-uniform (Glorot) weights and zero bias and optimized using the L-BFGS optimizer with the following configuration:

```
torch::optim::LBFGSOptions(
  lr               = 1,
  max_iter         = 20,
  max_eval         = 25,
  tolerance_grad   = 1e-09,
  tolerance_change = 1e-12,
  history_size     = 100,
  line_search_fn   = strong_wolfe
);
```

Table 1 Comparison between IGANets and the collocation IGA method for bi-cubic ($p = 3$) tensor-product B-splines

#coeffs	#weights	#epochs	$\mathrm{MSE}_{\mathrm{loss}}$	$\mathrm{MSE}_{\mathrm{IGANets}}$	#iter	$\mathrm{MSE}_{\mathrm{C\text{-}IGA}}$
1 layer, 1 neuron, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 1$						
4×4	73	1	$1.37 \cdot 10^{+1}$	$3.36 \cdot 10^{-2}$	1	$1.31 \cdot 10^{-3}$
8×8	233	6	$1.37 \cdot 10^{-3}$	$1.46 \cdot 10^{-4}$	86	$9.86 \cdot 10^{-5}$
16×16	841	65	$3.19 \cdot 10^{-6}$	$\mathbf{3.89 \cdot 10^{-6}}$	297	$\mathbf{3.88 \cdot 10^{-6}}$
32×32	3209	930	$3.85 \cdot 10^{-6}$	$\mathbf{3.59 \cdot 10^{-7}}$	1408	$\mathbf{1.96 \cdot 10^{-7}}$
64×64	12553	2340	$6.14 \cdot 10^{-5}$	$1.49 \cdot 10^{-5}$	> 10000	$1.11 \cdot 10^{-8}$
1 layer, 1 neuron, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 10^3$						
4×4	73	1	$1.87 \cdot 10^{+1}$	$1.09 \cdot 10^{-2}$	1	$1.31 \cdot 10^{-3}$
8×8	233	1	$1.44 \cdot 10^{-3}$	$\mathbf{9.79 \cdot 10^{-5}}$	86	$\mathbf{9.86 \cdot 10^{-5}}$
16×16	841	24	$3.07 \cdot 10^{-6}$	$\mathbf{3.88 \cdot 10^{-6}}$	297	$\mathbf{3.88 \cdot 10^{-6}}$
32×32	3209	65	$1.79 \cdot 10^{-8}$	$\mathbf{1.97 \cdot 10^{-7}}$	1408	$\mathbf{1.96 \cdot 10^{-7}}$
64×64	12553	315	$3.19 \cdot 10^{-8}$	$\mathbf{1.26 \cdot 10^{-8}}$	> 10000	$\mathbf{1.11 \cdot 10^{-8}}$
10 layers, 10 neurons per layer, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 1$						
4×4	1576	3	$1.37 \cdot 10^{+1}$	$3.36 \cdot 10^{-2}$	1	$1.31 \cdot 10^{-3}$
8×8	2744	16	$1.37 \cdot 10^{-3}$	$1.46 \cdot 10^{-4}$	86	$9.86 \cdot 10^{-5}$
16×16	7096	467	$3.08 \cdot 10^{-6}$	$\mathbf{3.89 \cdot 10^{-6}}$	297	$\mathbf{3.88 \cdot 10^{-6}}$
32×32	23864	1225	$1.76 \cdot 10^{-7}$	$\mathbf{1.27 \cdot 10^{-7}}$	1408	$\mathbf{1.96 \cdot 10^{-7}}$
64×64	89656	4557	$2.53 \cdot 10^{-6}$	$8.09 \cdot 10^{-7}$	> 10000	$1.11 \cdot 10^{-8}$
10 layers, 10 neurons per layer, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 10^3$						
4×4	1576	2	$1.71 \cdot 10^{+1}$	$1.09 \cdot 10^{-2}$	1	$1.31 \cdot 10^{-3}$
8×8	2744	4	$1.44 \cdot 10^{-3}$	$\mathbf{9.79 \cdot 10^{-5}}$	86	$\mathbf{9.86 \cdot 10^{-5}}$
16×16	7096	131	$3.07 \cdot 10^{-6}$	$\mathbf{3.88 \cdot 10^{-6}}$	297	$\mathbf{3.88 \cdot 10^{-6}}$
32×32	23864	285	$1.92 \cdot 10^{-8}$	$\mathbf{1.95 \cdot 10^{-7}}$	1408	$\mathbf{1.96 \cdot 10^{-7}}$
64×64	89656	575	$1.91 \cdot 10^{-8}$	$\mathbf{1.09 \cdot 10^{-8}}$	> 10000	$\mathbf{1.11 \cdot 10^{-8}}$

Values where both methods yield results with comparable accuracy are marked in bold. From left to right, the columns indicate the number of coefficients/basis functions (#coeffs), the number of network weights (#weights), the number of epochs until loss-value stagnation (#epochs), the mean-squared loss value ($\mathrm{MSE}_{\mathrm{loss}}$), the mean-squared error between the IGANets' solution and the exact solution interpolated in the spline space ($\mathrm{MSE}_{\mathrm{IGANets}}$), the number of BiCGStab iterations of the collocation IGA method (#iter), and the mean-squared error between the collocation IGA solution and the exact solution interpolated in the spline space ($\mathrm{MSE}_{\mathrm{C\text{-}IGA}}$)

Table 2 Comparison between IGANets and the collocation IGA method for bi-quartic ($p = 4$) tensor-product B-splines

#coeffs	#weights	#epochs	MSE_{loss}	$\text{MSE}_{\text{IGANets}}$	#iter	$\text{MSE}_{\text{C-IGA}}$
1 layer, 1 neuron, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 1$						
5×5	104	4	$3.29 \cdot 10^{-2}$	$1.37 \cdot 10^{-4}$	12	$4.47 \cdot 10^{-5}$
10×10	349	17	$2.78 \cdot 10^{-5}$	$8.90 \cdot 10^{-7}$	48	$2.19 \cdot 10^{-9}$
20×20	1289	99	$1.19 \cdot 10^{-7}$	$3.23 \cdot 10^{-9}$	582	$3.24 \cdot 10^{-12}$
40×40	4969	1530	$5.11 \cdot 10^{-6}$	$7.53 \cdot 10^{-7}$	1511	$7.71 \cdot 10^{-15}$
80×80	19529	> 5000	$2.75 \cdot 10^{-3}$	$2.17 \cdot 10^{-4}$	> 10000	$7.69 \cdot 10^{-11}$
1 layer, 1 neuron, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 10^3$						
5×5	104	5	$3.29 \cdot 10^{-2}$	$1.37 \cdot 10^{-4}$	10	$4.47 \cdot 10^{-5}$
10×10	349	6	$2.60 \cdot 10^{-5}$	$\mathbf{1.84 \cdot 10^{-9}}$	48	$\mathbf{2.19 \cdot 10^{-9}}$
20×20	1289	2164	$4.36 \cdot 10^{-8}$	$\mathbf{3.04 \cdot 10^{-12}}$	582	$\mathbf{3.24 \cdot 10^{-12}}$
40×40	4969	> 5000	$5.33 \cdot 10^{-9}$	$3.54 \cdot 10^{-12}$	1511	$7.71 \cdot 10^{-15}$
80×80	19529	> 5000	$7.57 \cdot 10^{-8}$	$3.14 \cdot 10^{-11}$	> 10000	$7.69 \cdot 10^{-11}$
10 layers, 10 neurons per layer, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 1$						
5×5	1805	4	$3.29 \cdot 10^{-2}$	$1.37 \cdot 10^{-4}$	12	$4.47 \cdot 10^{-5}$
10×10	3580	6	$2.58 \cdot 10^{-5}$	$5.40 \cdot 10^{-8}$	48	$2.19 \cdot 10^{-9}$
20×20	10280	> 5000	$3.11 \cdot 10^{-7}$	$4.40 \cdot 10^{-8}$	582	$3.24 \cdot 10^{-12}$
40×40	36280	> 5000	$1.09 \cdot 10^{-6}$	$2.34 \cdot 10^{-7}$	1511	$7.71 \cdot 10^{-15}$
80×80	138680	> 5000	$5.34 \cdot 10^{-4}$	$8.67 \cdot 10^{-5}$	> 10000	$7.69 \cdot 10^{-11}$
10 layers, 10 neurons per layer, Sigmoid activation function, $\omega_\Omega = 1, \omega_\Gamma = 10^3$						
5×5	1805	3	$3.29 \cdot 10^{-2}$	$1.37 \cdot 10^{-4}$	12	$4.47 \cdot 10^{-5}$
10×10	3580	17	$2.60 \cdot 10^{-5}$	$\mathbf{1.84 \cdot 10^{-9}}$	48	$\mathbf{2.19 \cdot 10^{-9}}$
20×20	10280	1206	$4.37 \cdot 10^{-8}$	$\mathbf{3.26 \cdot 10^{-12}}$	582	$\mathbf{3.24 \cdot 10^{-12}}$
40×40	36280	318	$2.92 \cdot 10^{-8}$	$5.49 \cdot 10^{-11}$	1511	$7.71 \cdot 10^{-15}$
80×80	138680	1629	$8.12 \cdot 10^{-7}$	$1.91 \cdot 10^{-9}$	> 10000	$7.69 \cdot 10^{-11}$

Values where both methods yield results with comparable accuracy are marked in bold. From left to right, the columns indicate the number of coefficients/basis functions (#coeffs), the number of network weights (#weights), the number of epochs until loss-value stagnation (#epochs), the mean-squared loss value (MSE_{loss}), the mean-squared error between the IGANets' solution and the exact solution interpolated in the spline space ($\text{MSE}_{\text{IGANets}}$), the number of BiCGStab iterations of the collocation IGA method (#iter), and the mean-squared error between the collocation IGA solution and the exact solution interpolated in the spline space ($\text{MSE}_{\text{C-IGA}}$)

Numerical experiments with alternative optimizers have been performed but demonstrated worse convergence behavior. The training has been stopped after 5000 epochs, which is marked as '> 5000' in Tables 1 and 2. Likewise, the BiCGStab solver was terminated after 10000 iterations, marked as '> 10000' in the aforementioned tables.

It is noteworthy that, in certain cases, already very small networks – one hidden layer with a single neuron – can capture the solution with the same accuracy as the collocation IGA method. In contrast to collocation IGA, IGANets naturally require the use of an optimization algorithm, which not only introduces additional hyperparameters, but also may lead to convergence issues, e.g., apparent in Table 2, where the mean-squared error does not drop below 10^{-12}. All experiments have been repeated for a larger network with 10 hidden layers and 10 neurons per layer. Next to varying the network configuration, we have performed all experiments with equal weighting of the PDE and boundary loss ($\omega_\Omega = 1,\ \omega_\Gamma = 1$) and a much stronger weighting of the boundary loss ($\omega_\Omega = 1, \omega_\Gamma = 10^3$). While the latter requires more epochs to achieve convergence, it enhances the solution accuracy, especially for the bi-quartic case, cf. Table 2.

4 A model problem towards engineering applications

In this section, we demonstrate how IGANets can be used in a typical linear structural mechanics scenario, which represents a common class of PDE problems in engineering. In the following sections, we first describe the underlying governing equations, then present two test cases and conclude with a multi-geometry example demonstrating a potential future engineering workflow.

4.1 Governing and constitutive equations

The equations in this section describe the material behavior under the following assumptions: linear elasticity, plane strain, small strains, isotropic material, homogeneous

material, isothermal conditions, no body forces and quasi-static conditions. These simplifications lead to the well-known linear elastic material behavior. In this work, a displacement-based formulation is chosen. Thus, the displacement field u serves as the primary unknown, and the corresponding stress distribution is computed during post-processing via *Hooke's law*.

The displacement-based formulation and the above-mentioned assumptions combined lead to the static form of the *Navier-Cauchy equations*, consisting of the equilibrium equations, the kinematic relations, and Hooke's law as the constitutive equation. In the following sections, these individual equations are presented in their strong form.

4.1.1 Equilibrium equations

The equilibrium equations for solid mechanics balance inner and outer forces and are (under the condition of no body forces) given by

$$\nabla \cdot \boldsymbol{\sigma} = 0, \tag{25}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor. In two dimensions, with the stress tensor $\boldsymbol{\sigma}$ defined as

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}, \tag{26}$$

the equilibrium equations can be written in matrix form as

$$\begin{bmatrix} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} \\ \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{27}$$

4.1.2 Kinematic relations

In linear elasticity under the hypothesis of small strains, the strain tensor is related to the displacement field u as

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla \mathrm{u} + (\nabla \mathrm{u})^T\right), \tag{28}$$

which results in the following matrix expression for the strain tensor $\boldsymbol{\varepsilon}$

$$\begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_{yy} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) & \frac{\partial u_y}{\partial y} \end{bmatrix}. \tag{29}$$

4.1.3 Constitutive law (Hooke's law)

For an isotropic, homogeneous, linear elastic material without thermal influence, the stress-strain relationship (Hooke's law) is

$$\boldsymbol{\sigma} = \lambda \operatorname{tr}(\boldsymbol{\varepsilon})\mathrm{I} + 2\mu\boldsymbol{\varepsilon}, \tag{30}$$

where λ and μ are the Lamé parameters, and I is the identity tensor. The Lamé parameters are material properties and can be evaluated via the material's Young's modulus E and its Poisson's ratio ν in the following relations

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)}. \tag{31}$$

Switching to matrix notation and inserting the values for the strain tensor $\boldsymbol{\varepsilon}$ brings Hooke's law of (30) into the following form

$$\begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \begin{bmatrix} (\lambda+2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y} & \mu\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \\ \mu\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) & (\lambda+2\mu)\frac{\partial u_y}{\partial y} + \lambda\frac{\partial u_x}{\partial x} \end{bmatrix}. \tag{32}$$

4.1.4 Navier-Cauchy equations

The expression previously obtained for the stress tensor $\boldsymbol{\sigma}$ in (32) can now be inserted into the equilibrium condition of (25), resulting in

$$\begin{bmatrix} (\lambda+2\mu)\frac{\partial^2 u_x}{\partial x^2} + \mu\frac{\partial^2 u_x}{\partial y^2} + (\lambda+\mu)\frac{\partial^2 u_y}{\partial x \partial y} \\ (\lambda+2\mu)\frac{\partial^2 u_y}{\partial y^2} + \mu\frac{\partial^2 u_y}{\partial x^2} + (\lambda+\mu)\frac{\partial^2 u_x}{\partial x \partial y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{33}$$

Now finally, (33) describes the static form of the *Navier-Cauchy Equations*, which can be, once reorganized, written in compact form as

$$\mu\nabla^2 \mathrm{u} + (\lambda+\mu)\nabla(\nabla \cdot \mathrm{u}) = 0. \tag{34}$$

4.2 Boundary conditions

The problem is fully defined by specifying boundary conditions (BCs)

$$\mathrm{u} = \mathrm{u}_D \quad \text{on } \Gamma_D \quad \text{(Dirichlet BC)}, \tag{35}$$

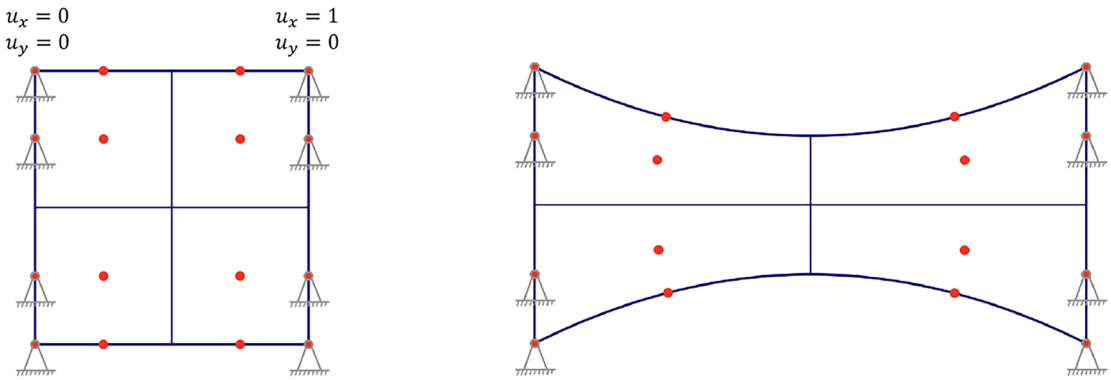


Fig. 2 Set-up for numerical example 1 including the reference geometry with Dirichlet BCs in (**a**) and the expected deformation in (**b**)

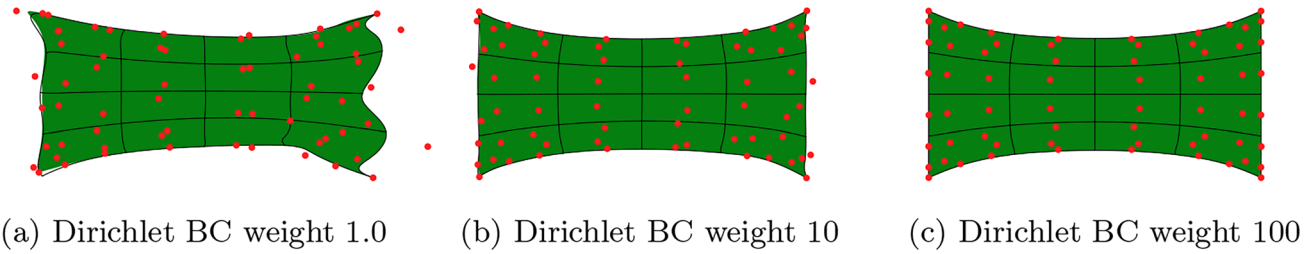


Fig. 3 Comparison of different values for the weight of the Dirichlet BC term in the loss function

$$\boldsymbol{\sigma} \cdot \mathrm{n} = \mathrm{t}_N \quad \text{on } \Gamma_N \quad \text{(Neumann BC)}, \tag{36}$$

where Γ_D and Γ_N are the Dirichlet and Neumann boundaries, respectively.

Notice that in collocation methods – in contrast to the finite element method – Neumann boundary conditions are not naturally satisfied and need to be explicitly imposed even in the case of free boundaries [31].

4.3 Single instance example 1: Dirichlet boundary conditions

In order to ensure proper training of IGANets, it is important that the individual loss terms have comparable magnitudes, an issue IGANets share with PINNs. To this end, the loss terms must be appropriately weighted. The purpose of this section is to investigate such weighting strategies.

4.3.1 Set-up

The computational domain is a unit square with Dirichlet boundary conditions applied on both the left and right edges. On the left edge, the geometry is fixed with zero displacement in both x- and y-directions. On the right edge, a displacement of 1.0 units is applied in the x-direction, while zero displacement is enforced in the y-direction. This set-up is illustrated in Fig. 2a. The expected deformation is illustrated in Fig. 2b. The magnitude of the deformation was chosen in such a way that clear displacements can be observed and the approximation errors when comparing different methods become apparent.

4.3.2 Determining loss function weights: Dirichlet boundary condition weight as example

As discussed in Sect. 2, the weighting of individual contributions to the loss function plays a crucial role in the training process. To illustrate this, we focus on the example of the Dirichlet boundary condition term. The influence of varying this weighting factor is demonstrated in Fig. 3, where three IGANets solutions corresponding to weights of 1, 10, and 100 are compared. The results indicate that a weighting factor of at least 100 is required to achieve a visually satisfactory enforcement of the Dirichlet boundary condition.

To further explore the influence of the weight, a series of simulations was conducted using weighting factors ranging from 10^0 to 10^{10}, analyzing their effect on both training

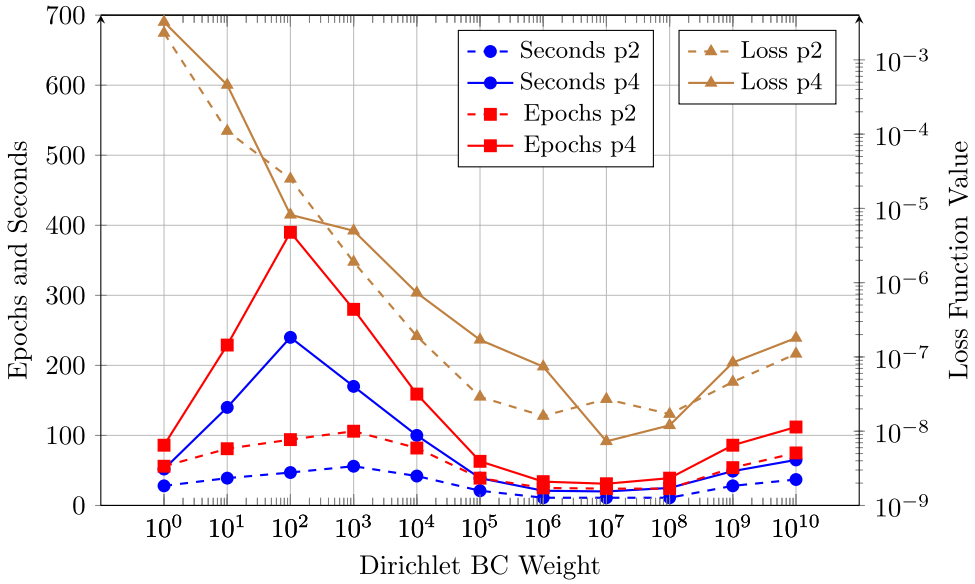


Fig. 4 Comparison of the Dirichlet BC weight's influence on the training behavior of the IGANet in terms of required training time (in seconds) and the number of epochs needed to reach a stationary solution. The comparison was carried out for a spline of degree 2 (p2) and a spline of degree 4 (p4)

performance and solution accuracy (Fig. 4). Note that the IGANet was trained on a standard laptop and the implementation was not optimized. Consequently, the data in Fig. 4 should only be interpreted as a general indication on how the weight influences the training dynamics and final accuracy. The comparison was carried out for two spline types: a degree-2 spline and a degree-4 spline[4].

For both spline degrees, the training time and the number of epochs are significantly higher for small weights, particularly in the range from 10^1 to 10^4. A distinct peak appears at 10^2 for both training time and epochs, especially for degree 4, indicating a potential instability in the convergence process. However, as the weight increases beyond 10^4, both the training time and the number of epochs stabilize and reach a minimum around a Dirichlet BC weight of 10^7.

The loss function values decrease sharply with increasing Dirichlet BC weights and also reach their lowest values around 10^6 and 10^7, for degree 2 and 4, respectively. Beyond these BC weights, the residual loss values slightly increase again, which may be attributed to the limitations of double-precision floating-point arithmetic (FP64). Very large BC weights require extremely small loss values to balance the equation, potentially approaching the machine accuracy limit around 10^{-16}.

When comparing the two spline degrees, degree 4 consistently requires more training time and epochs than degree 2. This is expected, since higher-degree splines have a larger support and therefore more basis functions are nonzero at each collocation point, which increases the cost per training step. Consequently, degree 2 tends to produce lower loss function values in most cases. However, this does not necessarily imply greater accuracy of the resulting solution. For example, reducing the number of control points to just four per direction would probably decrease the residual loss value even further, as a smaller number of collocation points makes it easier for the network to satisfy the governing equations and boundary conditions. Nevertheless, this does not imply that the solution obtained with a 4×4 control point grid would also be more accurate in a physical or an engineering sense.

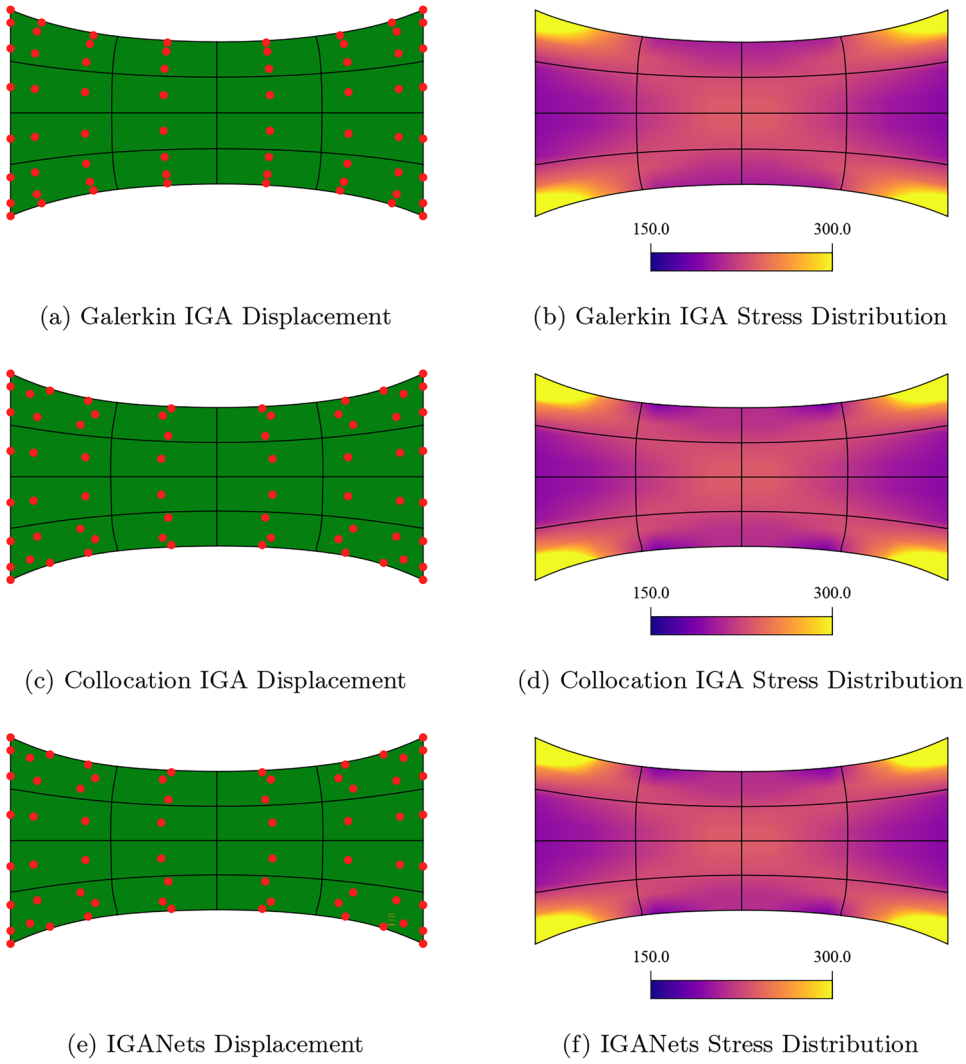
Overall, the results indicate that the Dirichlet BC weight has a substantial influence on both training efficiency and solution accuracy. In this case, weighting factors below 10^4 tend to result in unstable training behavior, whereas values in the range of 10^6 to 10^8 appear to be optimal. Increasing the weight beyond 10^8 does not provide further improvements – in fact, it even degrades the solution due to the numerical issues mentioned above. For this reason, a Dirichlet BC weight of 10^7 is used for the following experiment.

4.3.3 Simulation results

In this section, we compare the IGANets results for the test case with a Galerkin-based IGA solution computed in *G+Smo* [32] and a standard collocation IGA solution computed with an in-house *MATLAB* code. In Fig. 5, a first visual comparison of the resulting displacement and stress fields is provided, based on a test set-up of eight control points per direction and a spline degree of 4.

The stresses, visualized in Fig. 5, are limited to a maximum value of 300 in order to account for mathematical

[4] Notice that even degree splines are more efficient in collocation than odd degrees.



(a) Galerkin IGA Displacement (b) Galerkin IGA Stress Distribution

(c) Collocation IGA Displacement (d) Collocation IGA Stress Distribution

(e) IGANets Displacement (f) IGANets Stress Distribution

Fig. 5 Comparison of different simulation methods in terms of displacement and von Mises stress. Set-up: 8×8 control points and degree 4

singularities occurring at the corners of the domain. These stress peaks grow with increasing mesh resolution or collocation point density and would otherwise dominate the visualization, rendering most of the plot nearly monochrome. At first glance, all three solutions appear visually similar, suggesting that IGANets produce a sufficiently accurate result.

4.3.4 Error analysis

In order to quantitatively assess the solution quality of IGANets, in Fig. 6 we plot the absolute error for configurations of 8 and 20 control points per direction and spline degree 4 with respect to a very fine Galerkin solution with 64×64 control points. The absolute error is evaluated on a grid of 100×100 evaluation points.

Within the IGANets framework, the various contributions to the loss function are minimized during training. Unlike the collocation-based reference solution, which enforces the governing equations in strong form and satisfies them exactly at each collocation point, the neural network-based approach of IGANets offers only approximate fulfillment of the physical laws. Consequently, numerical errors can arise due to the inherent approximative nature of the network. Specifically, IGANets minimize the divergence of the stress tensor based on the governing equilibrium equation of linear elasticity, (25), with the aim of reducing this term as close to zero as possible. After training, the residual divergence is evaluated at each collocation point; any nonzero value is interpreted as an error, referred to here as the *Elasticity Error*. The spatial distribution of this error, for various control point configurations, is illustrated in Figure 7. The results indicate that the maximum elasticity error increases with a higher number of control points. This observation appears plausible, as it is easier for the network to satisfy

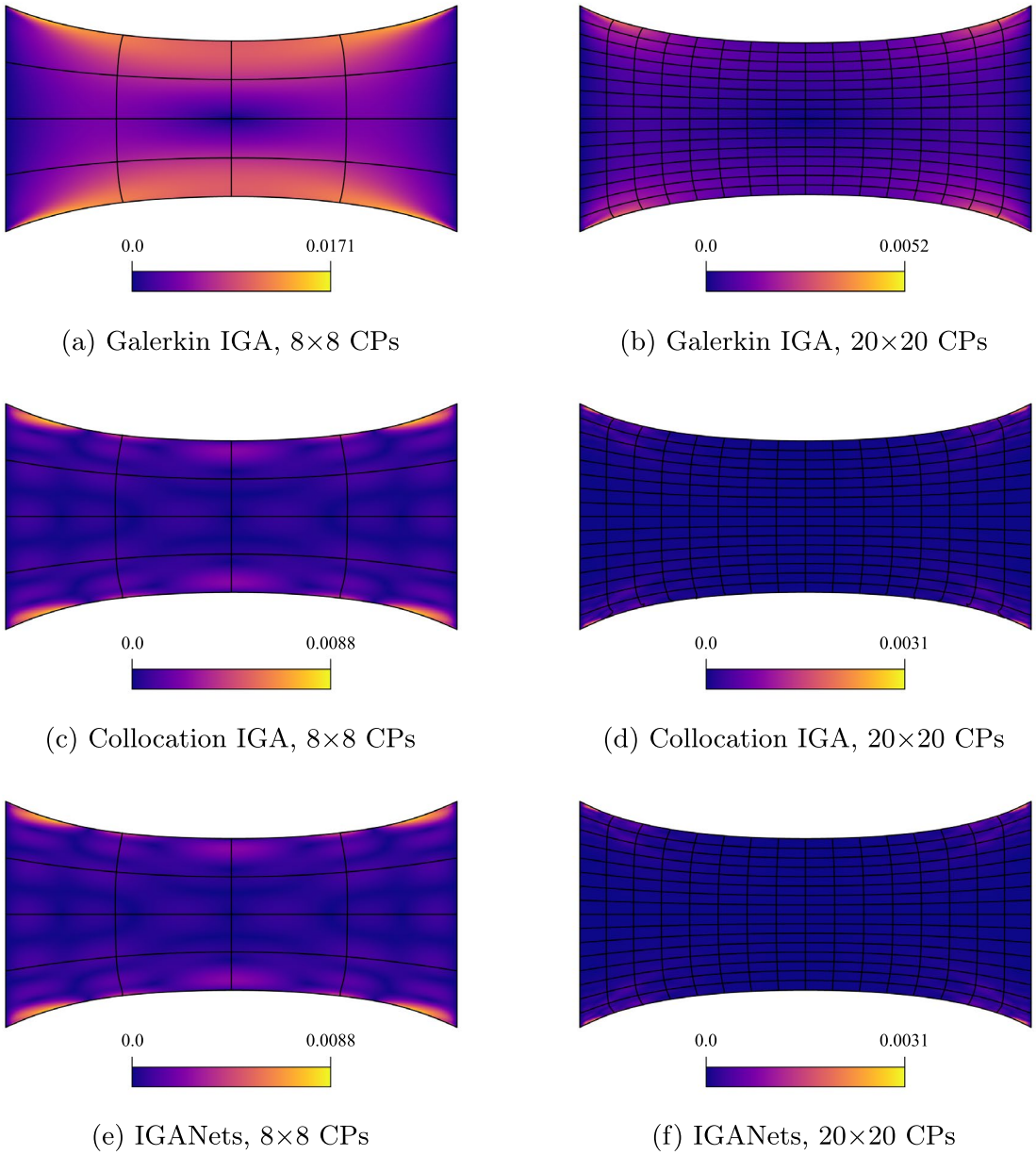


Fig. 6 Absolute error distribution with respect to a fine Galerkin IGA simulation for two refinement levels, namely 8×8 and 20×20 control points (CPs)

the governing equations at fewer collocation points than across a denser distribution. However, what is particularly noteworthy is the nature of the error distributions. While the solution fields in previous analyses exhibited symmetry in both horizontal and vertical directions, the elasticity error appears non-symmetric across the domain. Despite this, the magnitude of the error remains relatively small, ranging from $1 \cdot 10^{-4}$ to $5 \cdot 10^{-4}$ and does not significantly affect the overall solution quality. Even in the presence of these residual errors, the geometric behavior of the IGANets solution remains nearly indistinguishable from that of the standard collocation solution.

4.3.5 Training behavior

The IGANet was set up in the following way: The neural network architecture consists of two hidden layers with 25 neurons each, using *Sigmoid* activation functions. A minimum loss threshold of $1 \cdot 10^{-8}$ is defined, while the maximum number of epochs is adjusted individually for each

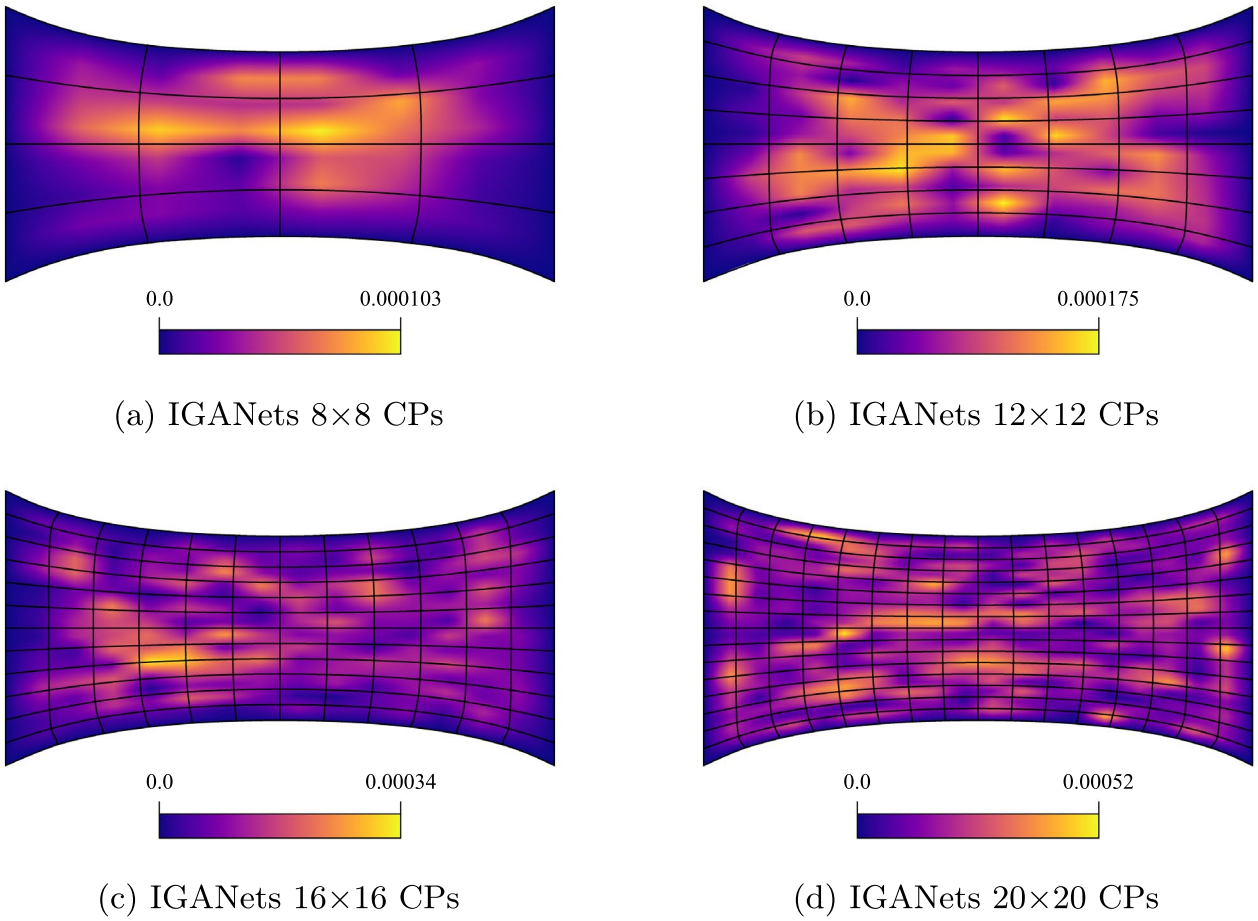


Fig. 7 Elasticity error distributions of IGANets for different control point numbers, using a fixed spline degree of $p = 4$. Notice that the maximum elasticity error increases with a higher number of control points. While seemingly counter-intuitive, it is in fact plausible, as the network architecture does not change during the experiment and it is easier for the network to satisfy the governing equations at fewer collocation points than across a denser distribution

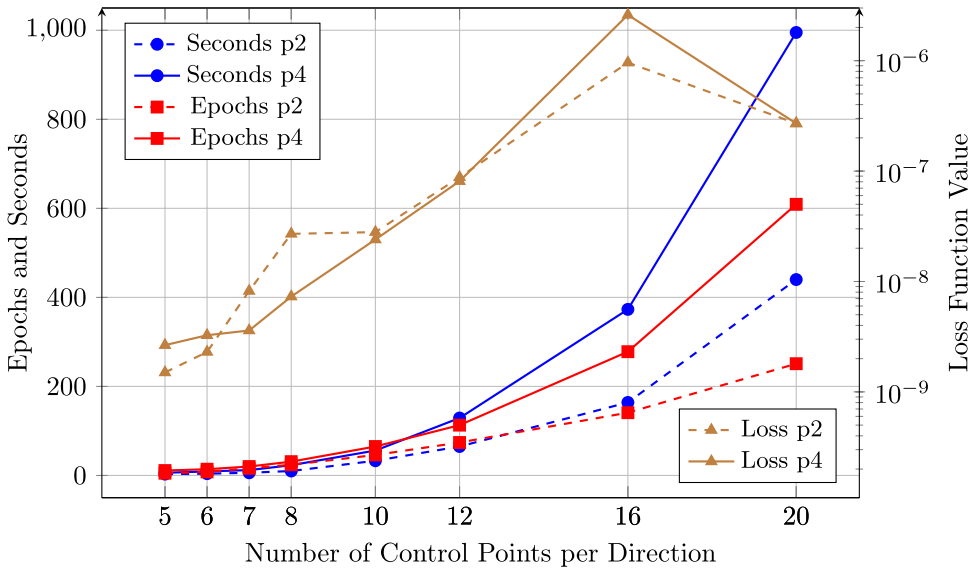


Fig. 8 Comparison of the training behavior of IGANets for different control point configurations. The figure illustrates the required training time and number of epochs needed to reach a stationary solution. Additionally, the values of the loss function at these stationary solutions are compared

simulation run. If the IGANet converges to a stationary solution before reaching the minimum loss, training continues until the specified epoch limit is met. The duration and convergence behavior of the training process depend significantly on the spline degree and the number of control points used. This dependency has been systematically investigated for spline degree 2 and 4 at control point grids ranging from 5×5 CPs to 20×20 CPs. Additionally, the value of the loss function at the stationary solution was recorded in each case. The corresponding results are presented in Figure 8. The figure illustrates that the training time increases exponentially with the number of control points. The same trend is observed for the number of epochs required to reach a stationary solution. For example, considering spline degree 4, the training time increases from 23 seconds for an 8×8 control point grid to 373 seconds for a 16×16 configuration – representing a sixteen-fold increase in time for a four-fold increase in control points. This suggests that the training time scales approximately quadratically with the number of control points in the case of spline degree 4. In addition to training time and epoch count, the values of the loss function at the stationary solution are also shown. For both spline degrees 2 and 4, the plots reveal a general increase in residual loss values with a growing number of control points – up to 16 per direction. Interestingly, for the 20×20 CP case, the loss value decreases again, deviating from the previous trend.

4.4 Single instance example 2: Traction boundary condition

The second experiment within the scope of applying IGANets to computational solid mechanics introduces an external force acting on one of the domain boundaries. As in the previous experiments, the simulation results obtained by IGANets are validated by comparison with the established Galerkin and collocation implementations. In this set-up, the spline degree is fixed to 3 in order to show applicability also to odd degree splines.

A particular focus of this experiment lies on the effect of supervised learning within the IGANets framework. By enabling or disabling the supervised component during training, its impact on solution accuracy and training time can be assessed.

4.4.1 Set-up

The reference geometry in this experiment is subject to a zero-displacement Dirichlet BC on the left edge and a traction Neumann BC on the right edge. The traction magnitude is set to 50. As in the previous case, traction-free BCs are applied to the remaining top and bottom edges. The set-up of the experiment is shown in Fig. 9a, and an expected deformation result is illustrated in Fig. 9b. Similarly to the previous test case, the body force f is set to zero throughout the domain.

4.4.2 Simulation results

The simulation results of Galerkin IGA, collocation IGA, and IGANets for the test case are presented below. Based on numerical experiments similar to Sect. 4.3.2, a Dirichlet BC weight of 10^8 has been chosen for the loss function. Figure 10 compares the results obtained by the three simulation methods, including the corresponding stress distributions. Since the traction acts on the right boundary, it is particularly relevant to observe the resulting stress values in this region. Ideally, the stress distribution should reflect this boundary condition and reach values close to 50 near the affected edge.

The stress plots in Fig. 10 (d,e,f) clearly demonstrate that visually, the traction BC on the right edge is fulfilled. Furthermore, the comparison reveals that the results of the standard collocation and IGANets simulation are identical

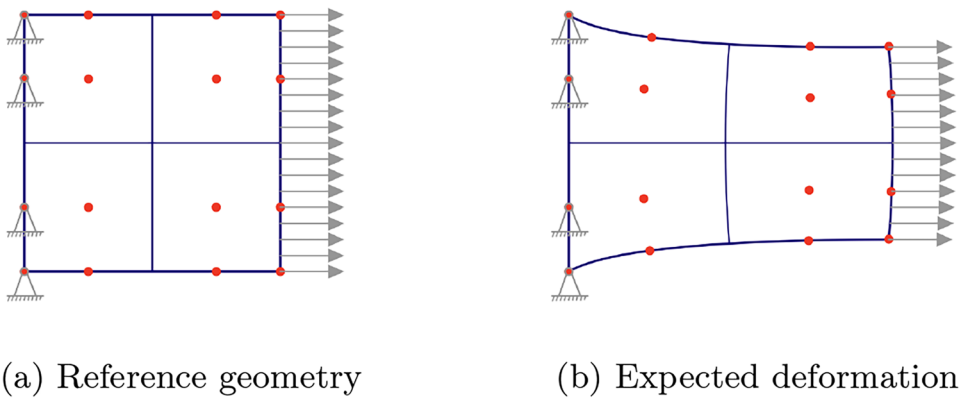


Fig. 9 Set-up for numerical example 2 including the reference geometry with Neumann BC in (**a**) and the expected deformation in (**b**)

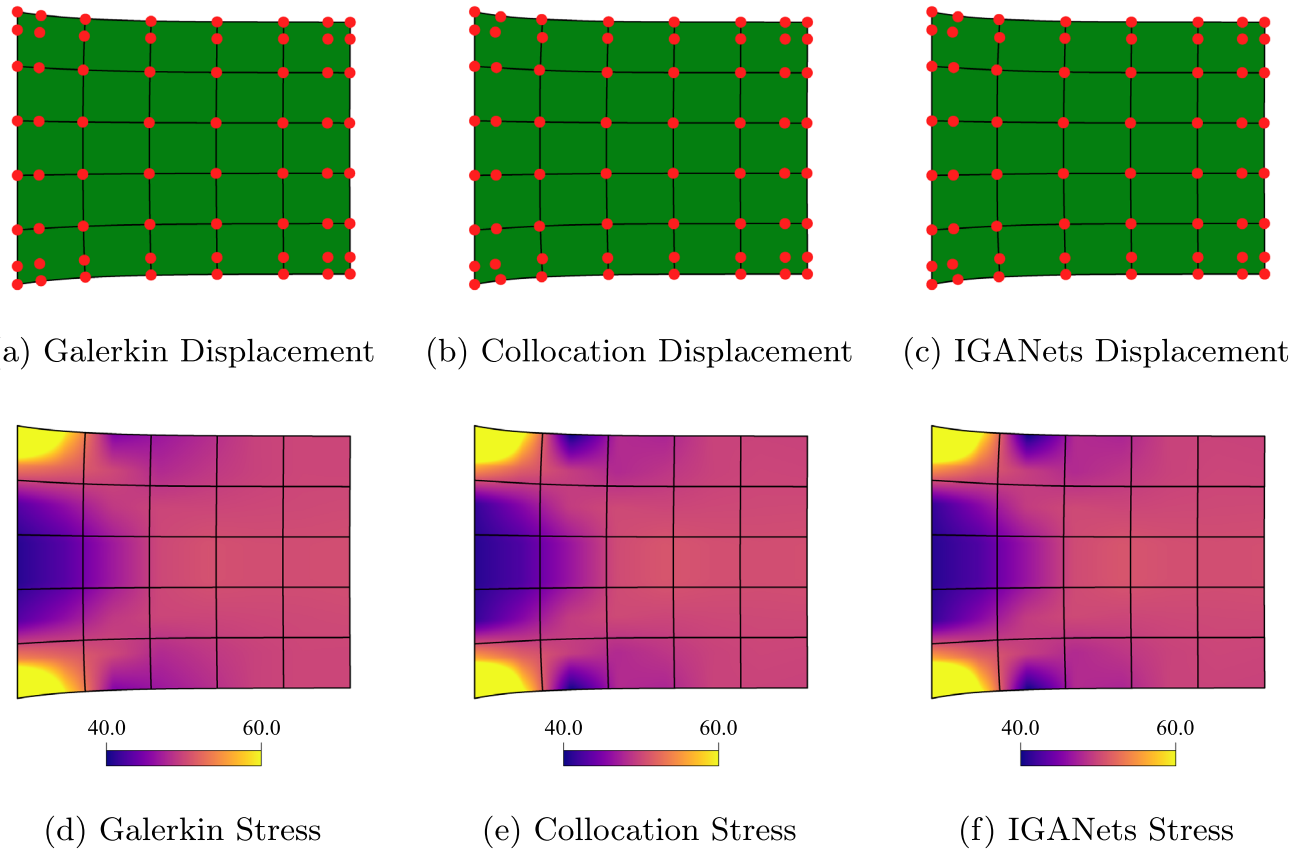


Fig. 10 Comparison of the deformation results obtained by a Galerkin-based, a standard collocation, and IGANets simulation (a,b,c), supplemented by their respective stress distribution (d,e,f). Set-up: 8×8 CPs and degree 3

in both displacement and stress distribution for the 8×8 CP configuration.

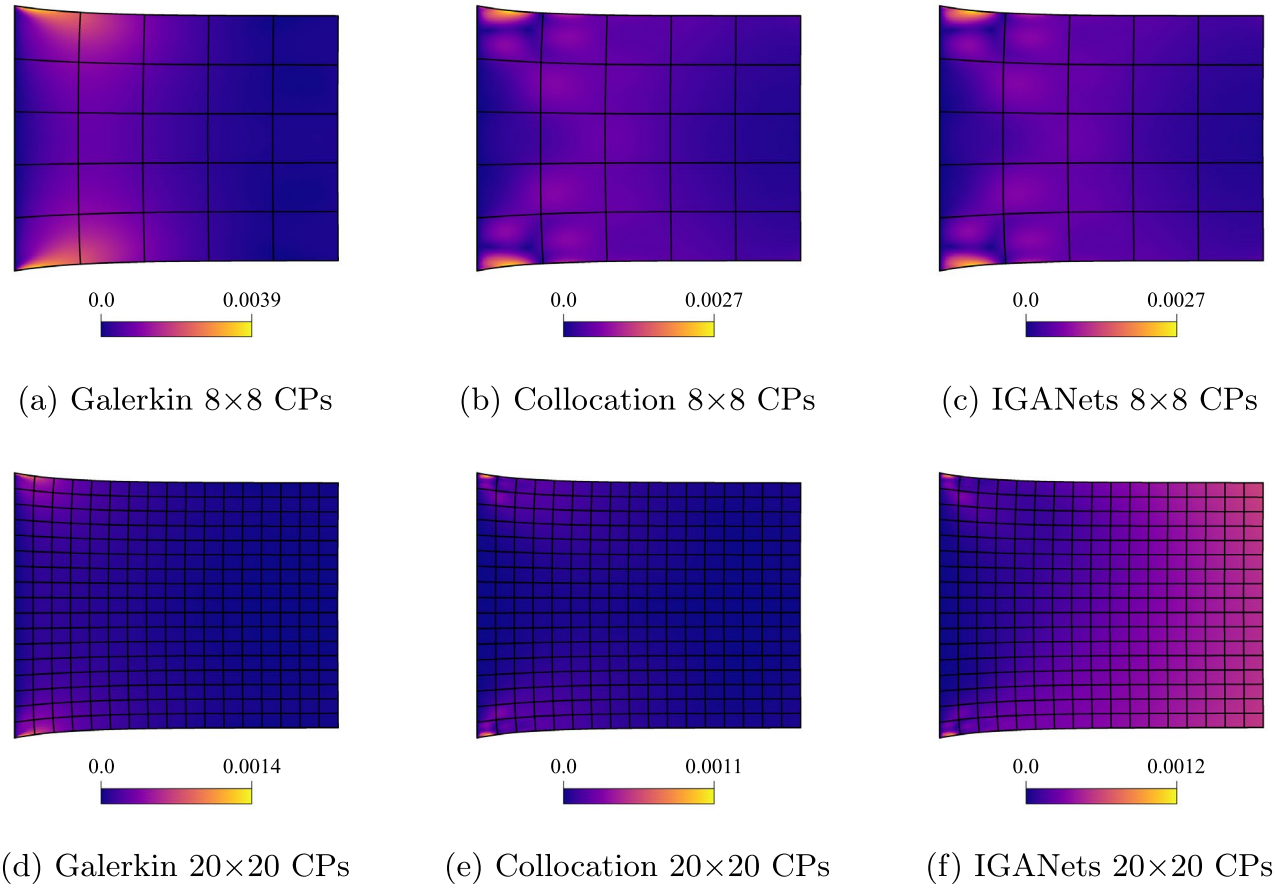
4.4.3 Absolute error distribution

As in the previous test case, a Galerkin-based reference solution has been computed. For that purpose, a domain with now 100×100 CPs is defined, and the corresponding solution is computed. This high-resolution solution serves as a reference for evaluating the accuracy of the (coarse) Galerkin, collocation, and IGANets simulations. To determine the error, a very dense evaluation grid with 1000×1000 points is created. At each of these points, the absolute difference between the Galerkin-based reference solution and the corresponding solution of the three methods is calculated. The resulting error distributions are then mapped onto the deformed geometries to visualize where and how the deviations occur. These results are illustrated in Figure 11. The values represent units relative to the domain size. For instance, if the domain is assumed to be 1×1 mm, an absolute error of 0.001 corresponds to a deviation of 1 μm from the reference solution.

The error plots in Fig. 11 reveal that, for both 8 and 20 CPs per direction, the collocation-based solutions are more accurate than the Galerkin solution computed with G+Smo. Additionally, it becomes evident that the IGANet tends to lose accuracy as the number of control points increases. In the right region of subplot (f), a visible deviation from the other two error distributions with 20 CPs per direction (subplots d and e) can be observed. This trend continues in the following sections, where the IGANet consistently shows slight difficulties in maintaining accuracy for higher control point counts combined with the traction force boundary condition, which might be improved through a more suitable choice of hyperparameters.

4.4.4 Simulation results enhanced via supervised learning

In the context of the current traction force experiment, supervised learning has been introduced to support the

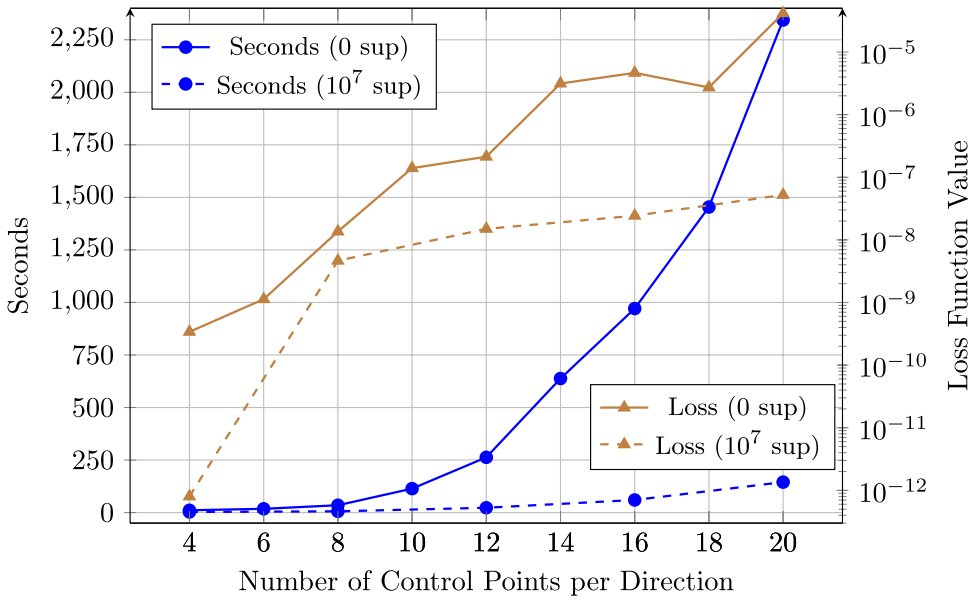


(a) Galerkin 8×8 CPs (b) Collocation 8×8 CPs (c) IGANets 8×8 CPs

(d) Galerkin 20×20 CPs (e) Collocation 20×20 CPs (f) IGANets 20×20 CPs

Fig. 11 Comparison of the absolute error obtained by Galerkin IGA, collocation IGA, and IGANets against the reference solution for spline degree 3 with 8×8 CPs (**a**, **b**, **c**) and 20×20 CPs (**d**, **e**, **f**)

Fig. 12 Comparison of the supervised and unsupervised training behavior of IGANets for different control point configurations. The figure illustrates the required training time needed to reach a stationary solution. Additionally, the values of the loss function at these stationary solutions are compared



IGANets training. The goal of supervised learning is to guide the network by providing a known target solution during training. In this case, a standard collocation solution with the same control point number as the current IGANet geometry serves as supervision data. To incorporate this additional information, a supervised loss term is added to the overall loss function. It is computed as the mean squared error (MSE) between the current IGANet control points and those of the collocation-based IGA solution. This term is weighted with a factor larger than 1.0 that controls the influence of the supervised component. A higher weight places greater emphasis on matching the known solution, guiding the network more strongly toward it. In this experiment, the data loss is added alongside the standard components of the traction force experiment: the residuals of the elasticity equations, the traction-free BC, the traction BC and the Dirichlet BC.

4.4.5 Training behavior

In this particular experiment, supervised learning proves especially beneficial, as the unsupervised training process is relatively slow. This can be attributed to the additional complexity introduced by the traction force loss, which requires the network to match the applied traction at each collocation point along the right boundary of the domain. To accelerate convergence, an arbitrarily chosen supervised learning weight of 10^7 is used, in combination with a Dirichlet BC weight of 10^8. This configuration helps the network prioritize the fulfillment of the Dirichlet and supervised components of the loss. A comparison between the training behavior of supervised and unsupervised runs is shown in Figure 12.

Figure 12 illustrates two key aspects – the exponential increase in training time and the moderate increase of the loss value with growing control point numbers, and the strong effect of supervised learning on reducing both. For instance, while training the IGANet with 20×20 CPs in the unsupervised case takes around 2300 seconds, the supervised version with a weight of 10^7 reduces this time to only 145 seconds. Despite the typical increase of the residual loss value with more collocation points – caused by the rising complexity of satisfying all conditions – the overall physical solution accuracy still improves with finer discretizations. Remarkably, the supervised version consistently shows lower loss values in all cases, meaning that the governing equations and boundary conditions are better fulfilled compared to the unsupervised set-up.

The use of a relatively high supervised learning weight of 10^7 is what enables this improvement. If the Dirichlet BC weight is simultaneously reduced (e.g., to 1) and the supervision weight increased further to 10^9, the training time drops drastically to just 24 seconds for the 20×20 CPs case. Compared to the simulation with a supervision weight of 10^7, a weight of 10^9 allows for a further sixfold reduction in training time (24 vs. 145 seconds). One might expect this gain to come at the cost of accuracy. However, the results suggest otherwise as the loss function value also decreases. This is not surprising, as the supervised target – the collocation IGA solution – is already the mathematically best solution the IGANet can approximate. Since both methods are based on the same numerical formulation (collocation), the ideal outcome for the IGANet is to replicate the standard collocation IGA result.

Providing this target has multiple advantages: First, the training time is reduced significantly. Second, the accuracy improves toward the best achievable solution, and third, the cost of obtaining the standard collocation solution is currently negligible, as it can be computed in seconds, even for higher control point resolutions. We conclude that offering

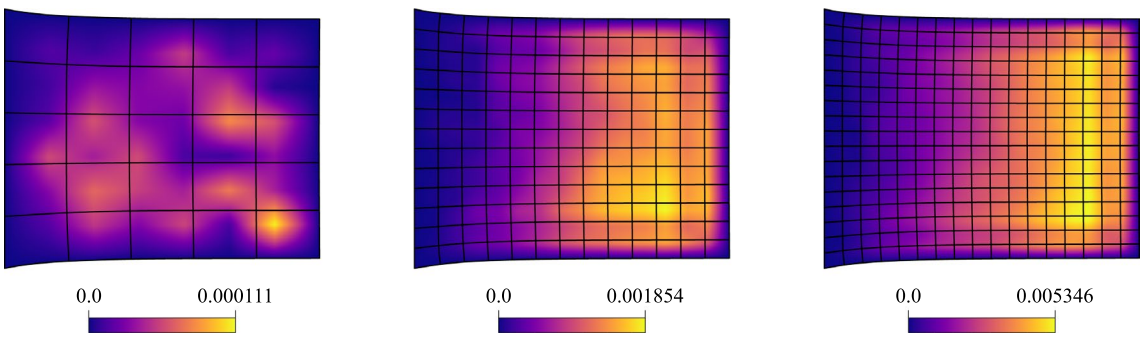
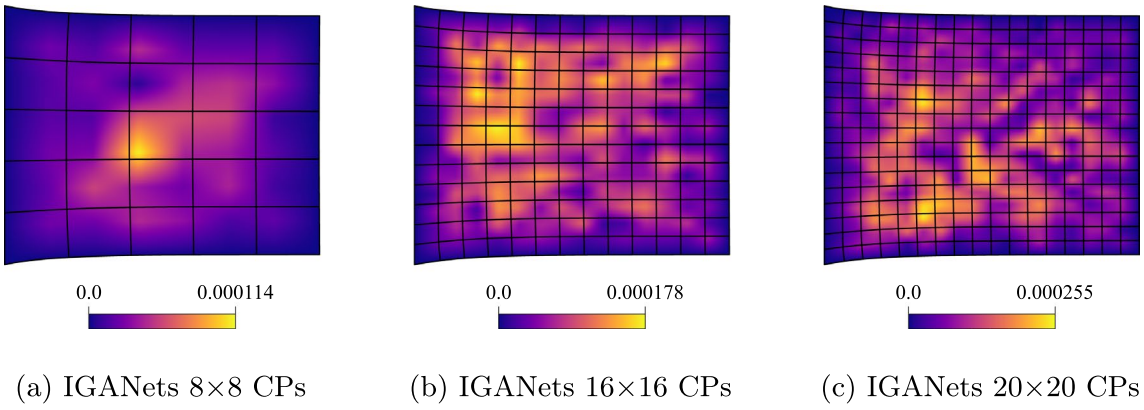


Fig. 13 Elasticity error of the **unsupervised** IGANets solution for spline degree 3 at different control point configurations



(a) IGANets 8×8 CPs (b) IGANets 16×16 CPs (c) IGANets 20×20 CPs

Fig. 14 Elasticity error of the **supervised** IGANets solution for spline degree 3 at different control point configurations (supervised learning weight of 10^7)

a readily available collocation-based reference during training substantially boosts the IGANet's performance in terms of both computational effort and solution quality.

To further evaluate the quality of the IGANets solution in both supervised and unsupervised settings, its ability to satisfy the governing equations is analyzed. Recalling the equilibrium condition $\nabla \cdot \boldsymbol{\sigma} = 0$, any deviation from zero in the divergence of the stress field can be interpreted as an error – previously referred to as the *Elasticity Error*. To visualize this error, the divergence of the stress tensor is computed at each collocation point. This reveals where and to what extent the IGANet struggles to fulfill the equilibrium condition. The results are gathered and compared for both the supervised and unsupervised simulations, providing insight into how supervision affects the model's ability to satisfy the underlying physics. The corresponding results are shown in Figure 13 and Figure 14.

The unsupervised results in Fig. 13 reveal accuracy issues in the right half of the domain, similar to the behavior observed in the absolute error plots. As the number of control points increases, the elasticity error also grows, since the IGANet struggles to satisfy the governing equations at a higher number of collocation points.

In contrast, the supervised results in Fig. 14 show a similar trend of increasing error with more control points, but at a significantly lower scale. While the unsupervised solution reaches a maximum elasticity error of approximately $5 \cdot 10^{-3}$ for 20×20 CPs, the supervised version shows a much smaller error peak of about $2.5 \cdot 10^{-4}$, making it roughly 20 times more accurate in this case.

These results highlight the strength of supervised learning. Guiding the training towards the standard collocation solution – which represents the best possible outcome for the IGANet – not only reduces training time considerably, but also increases accuracy. Taking these aspects into consideration, the utilization of supervised learning provides significant advantages against the unsupervised IGANet training.

4.5 Multi-instance example: Complex multi-geometry query

To further assess the potential of IGANets for integration into design-oriented workflows, the framework must be evaluated not only on simple benchmark problems but also on more challenging geometries and training scenarios. So far, the numerical examples have been restricted to square domains and single-instance learning, providing a convenient setting to validate the basic methodology. In the following, we therefore consider a more advanced two-dimensional geometry resembling an I-beam. In this setting, particular attention must be paid to the computation of normal vectors required for the traction-free boundary conditions, since the boundary segments are no longer exclusively aligned with straight Cartesian directions. Moreover, while the previous examples kept all network inputs fixed, such that the IGANet effectively acted as solver for a single problem instance, the present study extends this setting to training across multiple geometries and load cases. This allows us to investigate both the approximation accuracy for unseen geometries with respect to an IGA collocation reference solution and the resulting training behavior in a genuine multi-instance setting.

4.5.1 Set-up

In this example, we consider a varying geometry, while all instances remain similar to a two-dimensional I-beam. The

considered I-beam geometries vary randomly in height and width from 1×1 to 3×3 length units. In this way, a more advanced geometric setting is introduced compared to the previously considered square domains.

The boundary conditions are chosen consistently for all instances. A homogeneous Dirichlet boundary condition is imposed on the bottom edge, traction-free boundary conditions are prescribed on the left and right edges, and a fixed displacement is applied to the top edge. The prescribed displacement is chosen to be comparatively small in order to ensure realistic results within the elastic small-strain regime. More specifically, the displacement of the top edge is varied in both x- and y-direction within the interval $[-0.1, 0.1]$.

As in the previous examples, the Dirichlet boundary conditions are imposed weakly during training, using a Dirichlet weight of 10^6. The collocation points are again chosen as Greville abscissae. However, in contrast to the previous examples, both the geometry and the solution variable are represented using spline spaces of degree 3 with 7×7 control points.

4.5.2 Training method

In this example, the generalization capability of the trained IGANet is investigated. While the previous examples demonstrated that the framework performs very well as an equation solver for a single fixed problem instance, the present setting is designed to evaluate its predictive performance on unseen data.

To this end, the network is trained on datasets consisting of 1, 10, 100, and 1000 training geometries, respectively. These geometries are generated randomly within a prescribed range of geometric variation, yet still living in the same spline space. In addition to the geometry, the boundary conditions are also varied. More specifically, the displacement-based Dirichlet condition on the upper boundary is changed such that the prescribed displacement ranges within $\Delta x, \Delta y \in [-0.1, 0.1]$.

During training, a set of 100 validation samples is considered. At selected stages of the training process, these validation samples are passed individually through the current IGANet, and the predicted solution coefficients are compared to the corresponding reference solutions. Based on this validation step, training is continued only as long as the network improves its performance on the validation set – in this way, overfitting is avoided.

After training, the final model is evaluated on a set of 25 unseen test samples. The predicted solutions are then compared to the corresponding reference solutions obtained by standard IGA collocation.

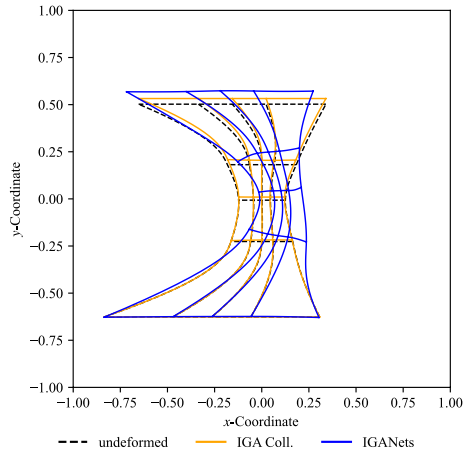
4.5.3 Simulation results

The considered setting is intended to mimic a simplified engineering workflow. A network is first trained on a set of geometries and boundary conditions within a prescribed range. Afterwards, a new geometry together with new boundary conditions, both lying within this range but not contained in the training or validation data, is provided as input to the trained IGANet. The goal is then to obtain the corresponding deformation result directly from the network without solving a new collocation problem. To assess the quality of this prediction, the resulting deformation is compared to the corresponding standard IGA collocation solution by evaluating the Cartesian error.

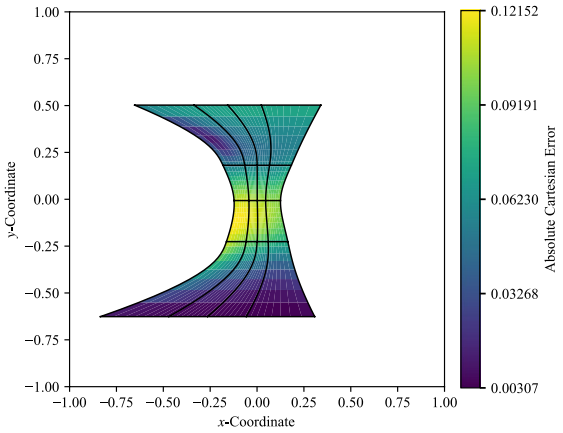
More specifically, the following procedure is carried out. First, the network is trained on varying numbers of training samples, including the validation process described above. Next, previously unseen evaluation geometries are generated and assigned boundary conditions within the training range. These geometry and boundary condition coefficients are then provided as inputs to the trained network, yielding a predicted deformation field. Finally, the prediction is compared to the corresponding reference solution obtained by standard IGA collocation.

Figure 15 shows the results for three different evaluation geometries, denoted by A, B, and C, when the network has been trained on only a single training sample. It can be seen that the predicted deformations and the collocation reference solutions do not yet coincide perfectly and the network struggles to capture the overall deformation behavior appropriately for the one-training sample case. However, the accuracy improves significantly once the number of training samples is increased.

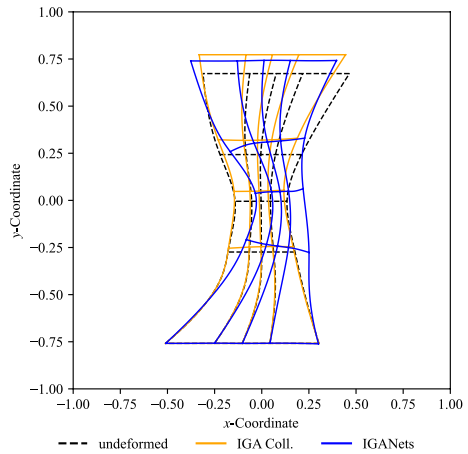
This trend is illustrated further in Fig. 16, now for increasing numbers of training samples. For clarity, not all training set sizes are displayed for every geometry. Instead, the detailed comparison for 10, 100, and 1000 training samples is shown for geometry A only. Overall, the prediction quality remains similar across the considered geometries, indicating that the network performance is not restricted to a specific individual shape. At the same time, a clear improvement in accuracy can be observed with increasing number of training samples. For geometry A, the maximum error is of the order of 10^{-1} when training is performed with only one sample, decreases to the order of 10^{-2} for 10 training



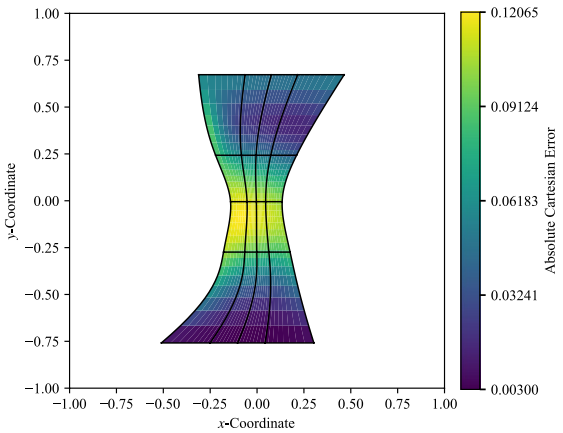
(a) Deformation | Geometry: A | Samples: 1



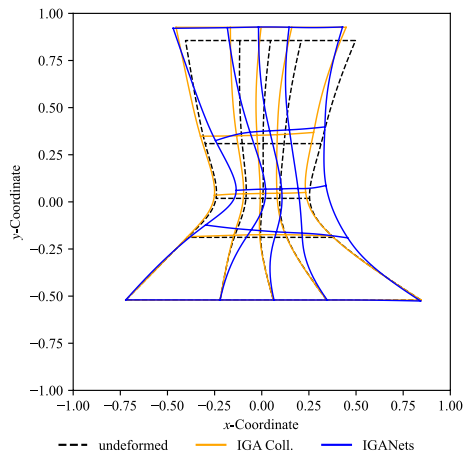
(b) Error | Geometry: A | Samples: 1



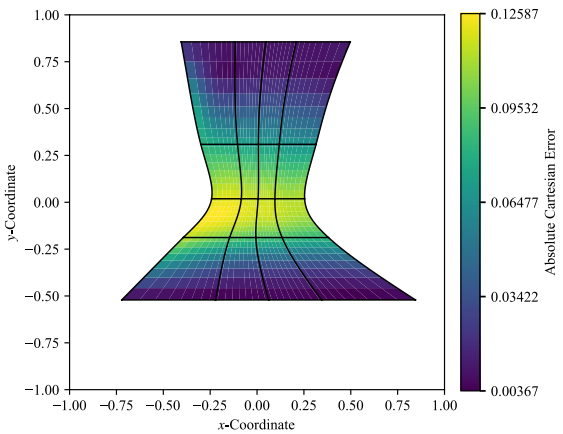
(c) Deformation | Geometry: B | Samples: 1



(d) Error | Geometry: B | Samples: 1

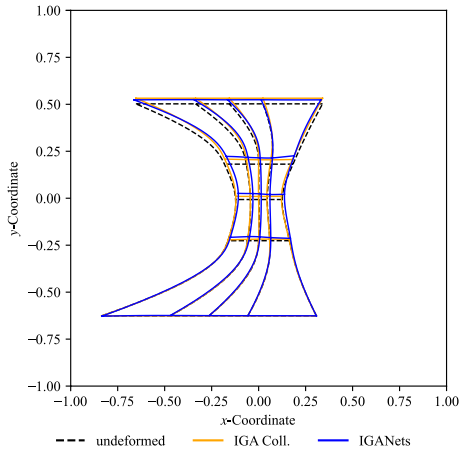


(e) Deformation | Geometry: C | Samples: 1

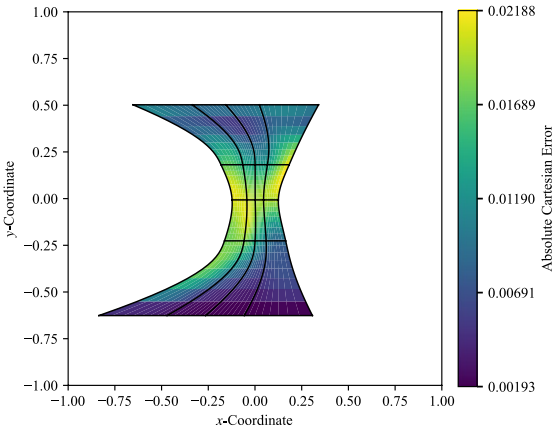


(f) Error | Geometry: C | Samples: 1

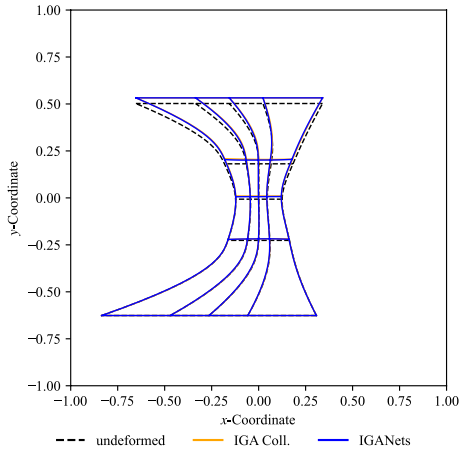
Fig. 15 Results of the multi-geometry and multi-boundary condition training for three representative geometries A, B, and C. The IGANets predictions are compared to the corresponding standard IGA collocation solutions. The left-hand side shows the deformed configurations for visual comparison, while the right-hand side presents the Cartesian error. In all cases, the network was trained using only a single training sample



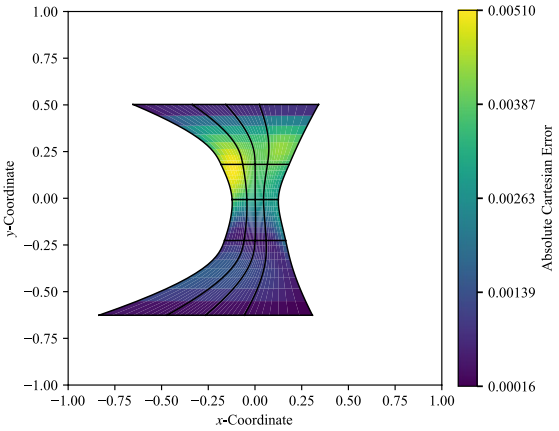
(a) Deformation | Geom.: A | Samples: 10



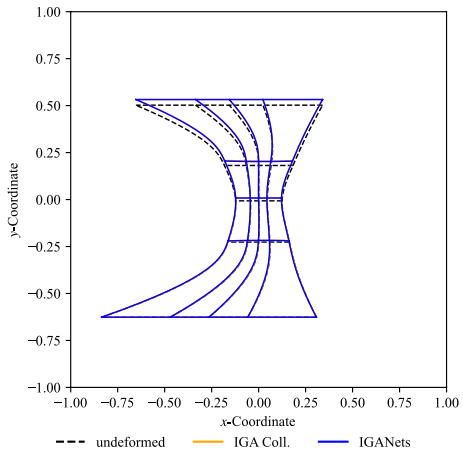
(b) Error | Geom.: A | Samples: 10



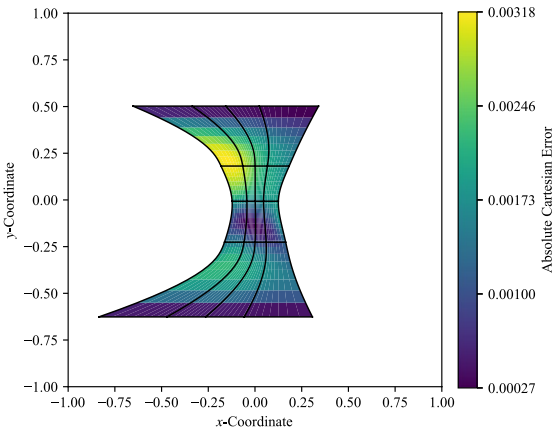
(c) Deformation | Geom.: A | Samples: 100



(d) Error | Geom.: A | Samples: 100



(e) Deformation | Geom.: A | Samples: 1000



(f) Error | Geom.: A | Samples: 1000

Fig. 16 Results of the multi-geometry and multi-boundary condition training for evaluation geometry A. The IGANets predictions are compared to the corresponding standard IGA collocation solutions. The left-hand side shows the deformed configurations for visual comparison, while the right-hand side presents the corresponding Cartesian error. The number of training samples is varied between 10 and 1000

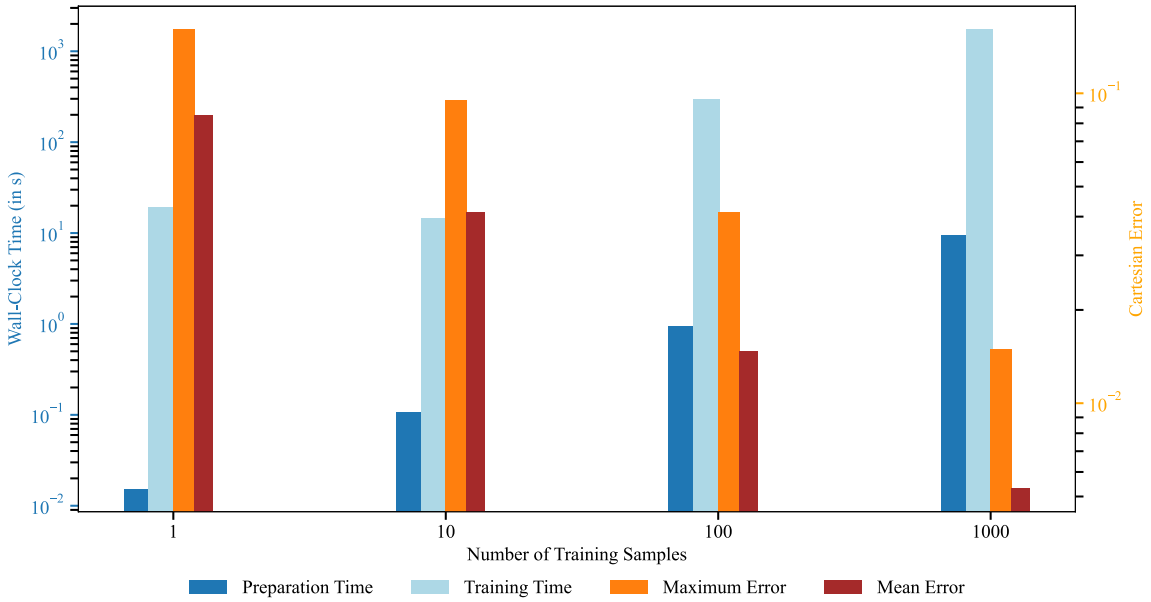


Fig. 17 Influence of the number of training samples on computational cost and prediction accuracy in the multi-geometry and multi-boundary condition setting. The left axis shows the wall-clock time for preparation and training, while the right axis reports the maximum and mean Cartesian error over all 25 evaluation samples. With more training samples, the computational cost increases and the prediction error decreases.

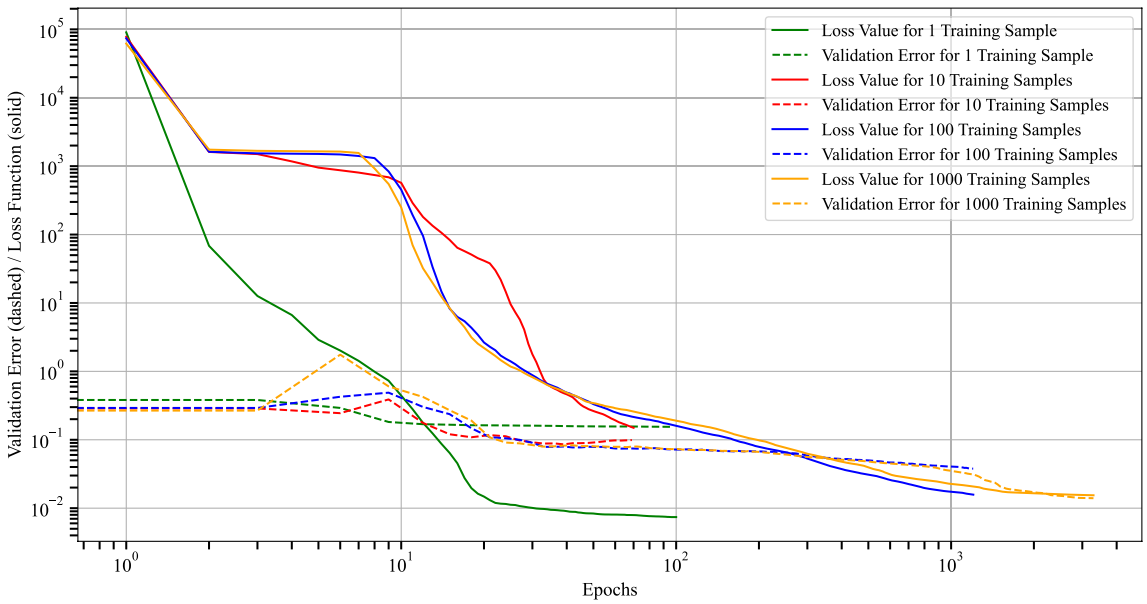


Fig. 18 Training loss (solid lines) and validation error (dashed lines) over the epochs for different numbers of training samples in the multi-geometry and multi-boundary condition setting. Smaller training sets lead to faster convergence, while all cases show a substantial loss reduction and a later stabilization of the validation error

samples, and reaches the order of 10^{-3} for both 100 and 1000 samples.

To further quantify the effect of the training set size, Fig. 17 summarizes both the computational cost and the prediction accuracy for increasing numbers of training samples. The left axis reports the wall-clock time in seconds, distinguishing between preparation time and training time, while the right axis shows the corresponding Cartesian errors in terms of the maximum and mean error over all 25 evaluation samples. Here, the preparation time denotes the cost of setting up the collocation points and evaluating the required derivatives for each geometry before training. The training time, in contrast, corresponds to the time required to optimize the IGANet on the respective dataset. As expected,

both preparation time and training time increase with the number of training samples. At the same time, both error measures decrease consistently, showing that broader coverage of the geometry and boundary-condition space leads to improved predictive accuracy on unseen instances.

A more detailed look at the error measures reveals a clear improvement in generalization performance. The maximum Cartesian error over all evaluation samples decreases with the number of training samples, along with the mean error, following the same trend with lower magnitudes. This indicates that increasing the number of seen geometries and loading conditions substantially improves the robustness and accuracy of the network predictions, although this comes at the expense of higher offline computational cost.

The training histories shown in Fig. 18 provide additional insight into this behavior. The solid lines represent the training loss, while the dashed lines denote the validation error, each plotted over the training epochs for training sets ranging from 1 to 1000 samples. The validation error is computed by averaging over all 100 validation samples, where for each sample the corresponding maximum validation error is taken into account. For all cases, the loss decreases significantly over the course of training, indicating that the network successfully fits the underlying problem family. The validation error shows a similar overall trend, although small fluctuations can be observed, particularly during the early stage of training. These variations should be interpreted with care, since the validation error is evaluated only every three epochs, whereas the training loss is recorded at every epoch. Overall, both quantities stabilize at later stages of training, indicating convergence of the optimization process.

A further observation is that smaller training sets lead to substantially shorter training runs in terms of epochs, whereas larger datasets require more epochs before convergence is reached. Together with the results of Fig. 17, this indicates that enlarging the training set primarily improves the predictive quality on unseen geometries while increasing the offline cost of preparing and training the model.

5 Conclusions

In this work, we presented IGANets as a prototype digital twin technology that enables rapid evaluation of the physical behavior of new engineering designs. Its ability to approximate PDE solutions with high accuracy makes it a promising tool for both manual use by designers and integration into automated optimization workflows.

IGANets share conceptual ground with PINNs and DeepONets, yet the method distinguishes itself through the explicit use of a spline basis. This choice maintains a direct connection to CAD models, facilitates seamless integration into the design process, and ensures the smoothness required to employ the strong form of the governing PDEs in a collocation setting. Moreover, IGANets build on the theory of IGA collocation, leveraging established results such as the optimal placement of collocation points.

We demonstrated the capabilities of IGANets on two representative PDE problems – the Poisson equation and linear elasticity. Through these examples, we assessed the overall performance of the framework, highlighted the importance of appropriately weighting the individual loss components associated with the PDE and boundary conditions, and showed that the inclusion of even small amounts of labeled data can significantly reduce training time.

Beyond its use as an equation solver for a single fixed problem instance, we further investigated the ability of IGANets to generalize across varying geometries and boundary conditions. For this purpose, a multi-instance linear-elasticity example with I-beam-like geometries was considered. The results showed that IGANets can successfully predict solutions for previously unseen geometries and boundary conditions within the range covered during training. In particular, increasing the number of training samples led to a clear reduction in the prediction error, while naturally increasing the offline effort required for data preparation and training. These findings support the intended use of IGANets in design scenarios in which an offline training phase is followed by rapid online evaluation of new design instances.

Overall, IGANets provide a promising direction for integrating machine learning into engineering analysis and design, combining geometric flexibility with physically informed learning. At the same time, the presented results indicate that its effectiveness in such applications depends strongly on the coverage of the training space in terms of geometry and boundary condition variations. This makes the framework particularly attractive for parametric design studies and digital twin applications, where repeated evaluations within a prescribed design family are required.

Appendix

A Efficient evaluation of B-spline basis functions

For the efficient evaluation of *multi-variate* B-spline basis functions (or their derivatives), Algorithm 1 can be applied to each univariate B-spline basis independently from which the final function value can be computed as follows, e.g.,

$$f(\bar{\boldsymbol{\xi}}) = \left(\mathrm{b}^{(1)}_{[i_1-p_1:i_1]}(\bar{\xi}^{(1)}) \otimes \cdots \otimes \mathrm{b}^{(3)}_{[i_3-p_3:i_3]}(\bar{\xi}^{(3)})\right) \cdot \mathrm{c}_{[\mathrm{i-p:i}]}. \tag{37}$$

Here, $[i. - p. : i.]$ denotes the selection operator that selects the subset of univariate B-spline basis functions ranging from $i - p$ to i and $[\mathrm{i} - \mathrm{p} : \mathrm{i}]$ is its multi-dimensional generalization that extracts the corresponding subset of basis coefficients c.

products with matrices composed of Kronecker products. Dropping the selection operator for ease of readability, (37) can be cast into

$$\left(\mathrm{b}^{(1)} \otimes \mathrm{b}^{(2)} \otimes \mathrm{b}^{(3)}\right) \cdot \mathrm{c} = \left(\mathrm{I} \otimes \mathrm{I} \otimes \mathrm{b}^{(3)}\right) \cdot \left(\mathrm{I} \otimes \mathrm{b}^{(2)} \otimes \mathrm{I}\right) \cdot \left(\mathrm{b}^{(1)} \otimes \mathrm{I} \otimes \mathrm{I}\right) \cdot \mathrm{c}. \tag{38}$$

Inputs: Parameter value $\bar{\xi} \in [\xi_1, \xi_{n+p+1})$ and derivative order r

Output: r-th derivative of B-spline basis functions

$$\mathbf{b} = \left[\mathrm{d}^r b_{i-p}(\bar{\xi}), \ldots, \mathrm{d}^r b_i(\bar{\xi})\right]$$

1: Find positive integer $i \leq n + p + 1$ such that $\bar{\xi} \in [\xi_i, \xi_{i+1})$
2: $\mathbf{b} = 1$
3: **for** $k = 1, \ldots, p - r$ **do**
4: $\quad \mathbf{t}_1 = \left[\xi_{i-k+1}, \ \ldots, \ \xi_i\right]$
5: $\quad \mathbf{t}_{21} = \left[\xi_{i+1}, \ \ldots, \ \xi_{i+k}\right] - \mathbf{t}_1$
6: $\quad \mathbf{mask} = (\mathbf{t}_{21} < \mathrm{tol})$ ▷ element-wise comparison '$<$'
7: $\quad \mathbf{w} = \left(\bar{\xi} - \mathbf{t}_1 - \mathbf{mask}\right) \div (\mathbf{t}_{21} - \mathbf{mask})$ ▷ element-wise division '$\div$'
8: $\quad \mathbf{b} = \left[(1 - \mathbf{w}) \odot \mathbf{b} \ 0\right] + \left[0 \ \mathbf{w} \odot \mathbf{b}\right]$ ▷ element-wise multiplication '$\odot$'
9: **end for**
10: **for** $k = p - r + 1, \ldots, p$ **do**
11: $\quad \mathbf{t}_1 = \left[\xi_{i-k+1}, \ \ldots, \ \xi_j\right]$
12: $\quad \mathbf{t}_{21} = \left[\xi_{i+1}, \ \ldots, \ \xi_{i+k}\right] - \mathbf{t}_1$
13: $\quad \mathbf{mask} = (\mathbf{t}_{21} < \mathrm{tol})$
14: $\quad \mathbf{w} = (1 - \mathbf{mask}) \div (\mathbf{t}_{21} - \mathbf{mask})$
15: $\quad \mathbf{b} = \left[-\mathbf{w} \odot \mathbf{b} \ 0\right] + \left[0 \ \mathbf{w} \odot \mathbf{b}\right]$
16: **end for**

Algorithm 1 Efficient evaluation of univariate B-spline functions

In a practical implementation, the values of $\mathrm{c}_{[\mathrm{i-p:i}]}$ are typically not stored at contiguous memory positions unless the selection operator makes a deep copy. Often, the data is contiguous in the leading dimension, say, $\xi^{(1)}$ with offsets of n along the second direction, offsets of n^2 along the third direction, etc. We therefore suggest using Algorithm 993 [33] for the memory efficient computation of matrix-matrix

This expression can be evaluated in three steps over the number of univariate directions as given in Algorithm 2. The reshape operation in line 2 assumes that matrices are stored in column-major format, that is, after the transposition in line 3, c is a matrix with $p + 1$ rows. After the successful execution of the algorithm, c contains the scalar value of f or its derivative in the point $\bar{\boldsymbol{\xi}}$.

Inputs: Univariate B-spline basis functions

$$\mathbf{b}^{(1)} = \mathbf{b}^{(1)}_{[i_1-p_1:i_1]}, \ldots, \mathbf{b}^{(3)} = \mathbf{b}^{(3)}_{[i_3-p_3:i_3]}$$

evaluated in $\bar{\boldsymbol{\xi}} = \left(\bar{\xi}^{(1)}, \ldots, \bar{\xi}^{(3)}\right)$ and selection of B-spline coefficients $\mathbf{c} = \mathbf{c}_{[\mathbf{i}-\mathbf{p}:\mathbf{i}]}$

Output: B-spline function evaluated in $\bar{\boldsymbol{\xi}}$

$$f(\bar{\boldsymbol{\xi}}) = \left(\mathbf{b}^{(1)} \otimes \mathbf{b}^{(2)} \otimes \mathbf{b}^{(3)}\right) \cdot \mathbf{c}$$

1: **for** $k = 1, \ldots, p$ **do**
2: $\quad \mathbf{c} = \text{reshape}(\mathbf{c}, [\cdot], p+1)$
3: $\quad \mathbf{c} = \mathbf{b}^{(k)} \cdot \mathbf{c}^T$
4: **end for**

Algorithm 2 Efficient evaluation of B-spline function (Algorithm 993 [33])

Acknowledgements This research was funded in part by the Austrian Science Fund (FWF) 10.55776/I6957, a project that is embedded into the TRR 402/1-525069572 funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation). Matthias Möller has received support from SURF, the Dutch supercomputing center through grants ETP0001 and EINF-7907. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission.

Author contributions All authors contributed to the conception and design of the work. In particular, M.M. developed the IGANets concept, implemented the IGANets core library (https://github.com/iganets/iganet), performed the validation experiments in Section 3, and created Figure 1. G.O. developed the linear elasticity application (https://github.com/iganets/iganet-elasticity), performed all numerical experiments, and created Figures 2-14 in Sections 4.1-4.4. C.G. and G.O. developed the multi-instance examples in Section 4.5. A.R. provided the MATLAB code for creating the collocation reference solutions. I.S. and C.G. contributed to the general development of the IGANets library. The first draft of the manuscript was written by M.M., G.O., and S.E. and all authors reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Code availability The IGANets source code is available online at https://github.com/iganets.

Declarations

Conflicts of Interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest in the subject matter or materials discussed in this manuscript. Authors Stefanie Elgeti and Alessandro Reali declare that they serve on the editorial board of Engineering with Computers for which they receive no compensation.

References

1. Manzoni A, Quarteroni A, Rozza G (2012) Computational reduction for parametrized PDEs: strategies and applications. Milan J Math 80:283–309. https://doi.org/10.1007/s00032-012-0182-y
2. Psichogios DC, Ungar LH (1992) A hybrid neural network-first principles approach to process modeling. AIChE J 38(10):1499–1511. https://doi.org/10.1002/aic.690381003
3. Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Networks 9(5):987–1000. https://doi.org/10.1109/72.712178
4. Dissanayake MG, Phan-Thien N (1994) Neural-network-based approximations for solving partial differential equations. Commun Numer Methods Eng 10(3):195–201. https://doi.org/10.1002/cnm.1640100303
5. 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. https://doi.org/10.1016/j.jcp.2018.10.045
6. Roy AM, Bose R, Sundararaghavan V, Arróyave R (2023) Deep learning-accelerated computational framework based on Physics Informed Neural Network for the solution of linear elasticity. Neural Netw 162:472–489. https://doi.org/10.1016/j.neunet.2023.03.014

7. Rao C, Sun H, Liu Y (2021) Physics-Informed Deep Learning for Computational Elastodynamics without Labeled Data. J Eng Mech 147(8). https://doi.org/10.1061/(asce)em.1943-7889.0001947
8. Kafkas P, Giannakopoulos G, Rekatsinas C (2024) Solving Linear Elasticity Problems using Physics-Informed Neural Networks. In: Proceedings of the 13th Hellenic Conference on Artificial Intelligence, pp. 1–8 . https://doi.org/10.1145/3688671.3688746
9. Haghighat E, Raissi M, Moure A, Gomez H, Juanes R (2020) A deep learning framework for solution and discovery in solid mechanics arXiv:2003.02751
10. Kag V, Gopinath V (2023) Physics-informed neural network for modeling dynamic linear elasticity arXiv:2312.15175
11. Bharadwaja BVSS, Nabian MA, Sharma B, Choudhry S, Alankar A (2022) Physics-Informed Machine Learning and Uncertainty Quantification for Mechanics of Heterogeneous Materials. Integrating Materials and Manufacturing Innovation 11(4):607–627. https://doi.org/10.1007/s40192-022-00283-2
12. Kamali A, Sarabian M, Laksari K (2022) Elasticity Imaging Using Physics-Informed Neural Networks: Spatial Discovery of Elastic Modulus and Poisson's Ratio. SSRN Electron J. https://doi.org/10.2139/ssrn.4203110
13. Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE (2021) Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence 3(3):218–229. https://doi.org/10.1038/s42256-021-00302-5
14. Wang S, Wang H, Perdikaris P (2021) Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Sci Adv 7(40):8605. https://doi.org/10.1126/sciadv.abi8605
15. Ahmed B, Qiu Y, Abueidda DW, El-Sekelly W, Soto BG, Abdoun T, Mobasher ME (2024) Physics-informed DeepONet with stiffness-based loss functions for structural response prediction arXiv:2409.00994
16. Marcati C, Schwab C (2023) Exponential Convergence of Deep Operator Networks for Elliptic Partial Differential Equations. SIAM J Numer Anal 61(3):1513–1545. https://doi.org/10.1137/21m1465718
17. Toscano JD, Oommen V, Varghese AJ, Zou Z, Ahmadi Daryakenari N, Wu C, Karniadakis GE (2025) From PINNs to PIKANs: Recent advances in physics-informed machine learning. Machine Learning for Computational Science and Engineering 1(1):1–43. https://doi.org/10.1007/s44379-025-00015-1
18. Jiang W, Gao X (2024) Review of collocation methods and applications in solving science and engineering problems. CMES-Computer Modeling in Engineering & Sciences 140(1):96. https://doi.org/10.32604/cmes.2024.048313
19. Kaspshitskaya MF, Luchka AY (1968) The collocation method. USSR Comput Math Math Phys 8(5):19–39
20. Douglas J, Dupont T (1973) A finite element collocation method for quasilinear parabolic equations. Math Comput 27(121):17–28. https://doi.org/10.2307/2005243
21. Wright K (1964) Chebyshev collocation methods for ordinary differential equations. Comput J 6(4):358–365. https://doi.org/10.1093/comjnl/6.4.358
22. Fairweather G, Meade D (2020) A survey of spline collocation methods for the numerical solution of differential equations. In: Diaz J (ed) Mathematics for Large Scale Computing. CRC Press, Boca Raton, pp 297–341
23. Auricchio F, Da Veiga LB, Hughes TJ, Reali A, Sangalli G (2010) Isogeometric collocation methods. Math Models Methods Appl Sci 20(11):2075–2107. https://doi.org/10.1142/S0218202510004878
24. Anitescu C, Jia Y, Zhang YJ, Rabczuk T (2015) An isogeometric collocation method using superconvergent points. Comput Methods Appl Mech Eng 284:1073–1097. https://doi.org/10.1016/j.cma.2014.11.038
25. Evans JA, Hiemstra RR, Hughes TJ, Reali A (2018) Explicit higher-order accurate isogeometric collocation methods for structural dynamics. Comput Methods Appl Mech Eng 338:208–240. https://doi.org/10.1016/j.cma.2018.04.008
26. Reali A, Hughes TJ (2015) An introduction to isogeometric collocation methods. Isogeometric Methods for Numerical Simulation 173–204. https://doi.org/10.1007/978-3-7091-1843-6_4
27. Eason ED (1976) A review of least-squares methods for solving partial differential equations. Int J Numer Meth Eng 10(5):1021–1046. https://doi.org/10.1002/nme.1620100505
28. Shapeev V, Belyaev V, Golushko S, Idimeshev S (2018) New possibilities and applications of the least squares collocation method. In: EPJ Web of Conferences, vol. 173, p. 01012. https://doi.org/10.1051/epjconf/201817301012 . EDP Sciences
29. Lin H, Xiong Y, Wang X, Hu Q, Ren J (2020) Isogeometric least-squares collocation method with consistency and convergence analysis. J Syst Sci Complexity 33(5):1656–1693. https://doi.org/10.1007/s11424-020-9052-9
30. Berrone S, Canuto C, Pintore M, Sukumar N (2023) Enforcing Dirichlet Boundary Conditions in Physics-informed Neural Networks and Variational Physics-informed Neural Networks. Heliyon 9(8). https://doi.org/10.1016/j.heliyon
31. Coman CD (2020) Continuum Mechanics and Linear Elasticity. Springer, Dordrecht . https://doi.org/10.1007/978-94-024-1771-5
32. Mantzaflaris A, et al (2015) G+Smo – Geometry + Simulation Modules. https://github.com/gismo/gismo. Accessed: 2025-04-29
33. Fackler PL (2019) Algorithm 993: Efficient computation with Kronecker products. ACM Transactions on Mathematical Software 45(2). https://doi.org/10.1145/3291041