Institute of Lightweight Design and Structural Biomechanics

Diplomarbeit

Deep Operator Learning in Computational Solid Mechanics

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs (Dipl.-Ing. oder DI), eingereicht an der TU Wien, Fakultät für Maschinenwesen und Betriebswissenschaften

Günther OBERMAIR

Mat.Nr.: 11913083

unter der Leitung von
Univ.Prof.in Dr.-Ing.in Stefanie Elgeti

Institut für Leichtbau und Struktur-Biomechanik

begutachtet von

Univ.Prof.in Dr.-Ing.in Stefanie Elgeti	Erstprüferin
Associate Prof. Dr.rer.nat. Matthias Möller	Zweitprüfer
ILSB, E317	Institut, Institutsnummer

Eidesstattliche Erklärung
Ich erkläre an Eides statt, dass die vorliegende Arbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen von mir selbstständig erstellt wurde.
Alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, sind in dieser Arbeit genannt und aufgelistet. Die aus den Quellen wörtlich entnommenen Stellen, sind als solche kenntlich gemacht.
Das Thema dieser Arbeit wurde von mir bisher weder im In- noch Ausland einer Beurteilerin/einem Beurteiler zur Begutachtung in irgendeiner Form als Prüfungsarbeit vorgelegt. Diese Arbeit stimmt mit der von den Begutachterinnen/Begutachtern beurteilten Arbeit überein.

Wien, Mai, 2025

[illegible]

Unterschrift

Abstract

In the design of engineering components, numerical methods such as the Finite Element Method (FEM) or Isogeometric Analysis (IGA) have become essential tools for evaluating structural performance, significantly reducing the need for physical prototyping. However, the design process frequently involves so-called multi-query scenarios – whether due to manual iterations by the designer seeking the optimal configuration, or as part of automated numerical optimization or uncertainty quantification workflows. In such cases, repeatedly running high-fidelity simulations becomes computationally prohibitive. A promising solution to this challenge is the use of Reduced Order Models (ROMs).

In this context of ROMs, the thesis investigates the application of the Isogeometric Analysis Network (IgANet), a method introduced by Matthias Möller, to structural mechanics problems. IgANet combines the smooth basis functions and exact geometry representation of IGA with neural networks trained in a physics-informed manner. Rather than relying solely on data, the network is trained by minimizing the residuals of the governing equations and boundary conditions. The underlying numerical approach is the isogeometric collocation method, which enforces the partial differential equations (PDEs) in their strong form at predefined collocation points.

The implemented IgANet framework is applied to simulate two-dimensional linear elasticity problems. Its modular code architecture allows for flexible configuration of geometries, boundary conditions, as well as supervised training using reference solutions from both Galerkin and collocation-based solvers. Two numerical test cases, defined on a unit square domain, are evaluated with respect to accuracy, convergence behavior, and training time. The results demonstrate that the trained network can achieve solution quality comparable to that of conventional numerical methods.

Particular emphasis is placed on evaluating the method's generalization capability. When trained on a single geometry, the framework is already able to predict solutions for modified domains without the need for retraining, simply by adjusting the input control points. However, training on multiple geometries can further improve the network's predictive power. This highlights the potential of IgANet as a Deep Operator Learning tool, enabling exploration of a vast design space in real time and supporting multi-query scenarios.

Kurzfassung

Numerische Methoden wie die Finite Elemente Methode (FEM) oder die Isogeometrische Analyse (IGA) sind wichtige Werkzeuge zur Bewertung mechanischer Eigenschaften und verringern den Bedarf an physischen Prototypen. In der Praxis kommt es jedoch häufig vor, dass eine Simulation mehrfach durchlaufen werden muss, etwa bei der Untersuchung verschiedener Entwurfsvarianten oder im Rahmen automatisierter Optimierungsprozesse. Solche sogenannten Multi-Query-Szenarien erfordern eine Vielzahl an Simulationen und machen den Einsatz von klassischen, rechenaufwändigen Methoden oft unpraktisch. Abhilfe schaffen hier Reduced-Order-Modelle (ROMs), die darauf abzielen, die Rechenzeit drastisch zu verringern, etwa für Anwendungen in der schnellen Entwurfsbewertung (Design Real-Time).

Diese Arbeit untersucht in diesem Zusammenhang das Isogeometric Analysis Network (IgANet), eine von Matthias Möller entwickelte Methode, für den Einsatz in der Strukturmechanik. IgANet kombiniert die glatten Basisfunktionen und die exakte Geometriebeschreibung der Isogeometrischen Analyse mit einem neuronalen Netz. Jenes Netz arbeitet nicht rein datenbasiert, sondern wird durch Minimierung der physikalischen Gleichungs- und Randbedingungsresiduen trainiert. Zum Einsatz kommt dabei die isogeometrische Kollokationsmethode, bei der die zugrunde liegenden Differentialgleichungen punktweise in starker Form erfüllt werden.

Das implementierte IgANet-Framework wird auf linear-elastische Probleme in zwei Dimensionen angewendet. Die modulare Architektur erlaubt eine flexible Vorgabe von Geometrien, Randbedingungen und gegebenenfalls auch referenzbasierten Trainingsdaten. Zwei Testszenarien auf einer Einheitsquadrat-Geometrie wurden hinsichtlich Genauigkeit, Konvergenzverhalten und Trainingsdauer analysiert. Die Ergebnisse zeigen, dass das Netzwerk in der Lage ist, Lösungen mit einer vergleichbaren Genauigkeit zu liefern wie klassische numerische Verfahren.

Ein besonderer Fokus liegt auf der Generalisierungsfähigkeit des Ansatzes. Bereits nach dem Training auf einer einzelnen Geometrie kann das Netzwerk auch für abgeänderte geometrische Domänen physikalisch valide Lösungen vorhersagen, indem lediglich die Eingabekontrollpunkte angepasst werden. Ein Training auf mehreren Geometrien kann die Vorhersagequalität dabei weiter verbessern. Dies unterstreicht das Potenzial von IgANet als Deep Operator Learning Methode, die schnelle Analysen über einen großen Parameterraum hinweg ermöglicht und sich somit ideal für den Einsatz in Multi-Query-Umgebungen eignet.

Contents

Glossary

Symbols	**Description**
x	Physical coordinate in the first dimension
y	Physical coordinate in the second dimension
ξ	Parametric coordinate in the first dimension
η	Parametric coordinate in the second dimension
E	Young's modulus
ν	Poisson's ratio
λ	First Lamé parameter
μ	Second Lamé parameter
$\mathbf{u}$	Displacement vector
$\hat{\mathbf{u}}$	Coefficients of the displacement spline
$\mathbf{P}$	Control points of the geometry spline
$\boldsymbol{\sigma}$	Cauchy stress tensor
$\boldsymbol{\varepsilon}$	Strain tensor

Acronyms

Abbreviation	**Description**
B-Spline	Basis Spline
BC	Boundary Condition
CAD	Computer-Aided Design
CP	Control Point
FEM	Finite Element Method
G+Smo	Geometry plus Simulation Modules [22]
IGA	Isogeometric Analysis
IgANet	Physics-informed Isogeometric Analysis Network [24]
Matlab	Matrix Laboratory [34]
MSE	Mean Squared Error
NURBS	Non-Uniform Rational B-Spline
PDE	Partial Differential Equation
PINN	Physics Informed Neural Network

List of Figures

1 Introduction

Since the Finite Element Method (FEM) has been established in the field of computational mechanics, it has become a standard approach for solving complex engineering problems [19]. One essential requirement towards FEM simulations is the creation of a computational mesh, usually based on already existing Computer-Aided Design (CAD) models. This process comes with a certain redundancy, since it requires the recreation of an already existing CAD model with a mesh-based representation.

Recognizing this inefficiency, Thomas J.R. Hughes and his team at the University of Texas at Austin introduced a groundbreaking alternative in 2005: Isogeometric Analysis (IGA) [15]. The fundamental idea of IGA is the direct use of the spline-based CAD basis functions for both the geometric representation of the domain and the numerical solution field. This approach has the potential to eliminate the need for typical mesh generation and allows computational analysts to perform simulations quasi directly on the CAD models. It is assumed that mesh generation is responsible for up to 80 % of time in the total FEM workflow. Due to the bypassing of this step, IGA offers a significant reduction in computational costs and enhances the seamless integration between design and numerical analysis.

Moreover, in recent years, the integration of machine learning techniques into computational mechanics has gained increasing attention [36]. With IgANet, Matthias Möller et al. have introduced a framework that unites the geometric precision of IGA with neural network-based solution strategies [23]. While conceptually related to Physics Informed Neural Networks (PINNs) [9], which solve differential equations by minimizing residuals of governing equations and boundary conditions, IgANet broadens this concept by learning mappings from input parameters to solutions. This makes it more appropriately understood as an instance of *Deep Operator Learning* [14], with particular strength in generalizing to new input configurations.

One might ask why neural networks should be used to solve these equations when efficient numerical solvers are already available. The strength of IgANet lies not in replacing classical solvers, but in its ability to generalize across varying input conditions. Once trained, the network can predict solutions to modified configurations, such as changes in geometry, without the need to retrain or reassemble a numerical system. This capability opens up new possibilities for efficient simulation in the context of parametrized Partial Differential Equations (PDEs).

Applying this method to mechanical engineering tasks could significantly accelerate design iterations. In conventional workflows, a CAD model is created, analyzed, and iteratively modified based on performance criteria, such as stress distribution. Each design change, no matter how minor, requires a new simulation, making the process time-consuming and computationally expensive. In contrast, once trained, the IgANet can instantly generate solutions for modified geometry parameters. This set-up provides feedback in design real-time and facilitates rapid prototyping not only in mechanical engineering design tasks.

While IgANet has already proven effective in a variety of scenarios, it has not yet been applied to problems in computational solid mechanics. The applicability, accuracy, and efficiency of this approach in linear elastic problems remain open research questions. Therefore, the objective of this thesis is to extend the existing IgANet framework to vector-valued elasticity equations and to explore its potential in computational solid mechanics.

The thesis is divided into the following four sections. First, the governing equations and constitutive laws are introduced in Chapter 2, then the IgANet framework and its training set-up are described in Chapter 3. The results are presented in Chapter 4, before the thesis is concluded in Chapter 5.

2 Governing and Constitutive Equations

The equations mentioned in this chapter follow the source *"Elasticity – Theory, Applications and Numerics"* by Martin Sadd [30] and describe the material behavior under the following assumptions:

- linear elasticity,
- small strains,
- isotropic material,
- homogeneous material,
- isothermal conditions,
- quasi-static consideration.

These simplifications lead to the well-known linear elastic material behavior and serve as a convenient starting point for first implementations of computational solid mechanics within the IgANet framework. Similar to many other FEM or IGA solvers, a displacement-based formulation is chosen. Thus, the displacement field $\mathbf{u}$ serves as the primary unknown, and the corresponding stress distribution is computed during postprocessing via *Hooke's law*. In contrast, also a stress-based formulation could have been considered, where the stress tensor $\boldsymbol{\sigma}$ is treated as the primary unknown, and the displacements are derived from the stress distribution. This, however, is typically only used in simulations where the accurate resolution of stresses is more important than the accuracy of displacements and where primarily traction boundary conditions are applied.

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.

2.1 Equilibrium Equations

The equilibrium equations for solid mechanics balance inner and outer forces and are given by

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{f} = \mathbf{0}, \tag{1}$$

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

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}, \tag{2}$$

the equilibrium equations can be written in matrix form as

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} + \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \mathbf{0}. \tag{3}$$

2.2 Kinematic Relations

In linear elasticity with small strains, the strain tensor is related to the displacement field $\mathbf{u}$ by

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right), \tag{4}$$

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

$$\begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_{yy} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \\ \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) & \frac{\partial u_y}{\partial y} \end{bmatrix}. \tag{5}$$

2.3 Constitutive Law (Hooke's Law)

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

$$\boldsymbol{\sigma} = \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon}, \tag{6}$$

where λ and μ are the Lamé parameters, and $\mathbf{I}$ 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

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)}. \tag{7}$$

Since the Lamé parameters are well established in literature, they are used in the remaining equations in place of Young's modulus and Poisson's ratio.

Switching to matrix notation and inserting the values for the strain tensor $\boldsymbol{\varepsilon}$ brings Hooke's law of Equation (6) into the following form

$$\begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \lambda \begin{bmatrix} \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} & 0 \\ 0 & \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} \end{bmatrix} + 2\mu \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \\ \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) & \frac{\partial u_y}{\partial y} \end{bmatrix}. \tag{8}$$

Merging the two matrices on the right-hand side, the Cauchy stress tensor $\boldsymbol{\sigma}$ results in the expression

$$\begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \begin{bmatrix} (\lambda+2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y} & \mu(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \\ \mu(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) & (\lambda+2\mu)\frac{\partial u_y}{\partial y} + \lambda\frac{\partial u_x}{\partial x} \end{bmatrix}. \tag{9}$$

2.4 Navier-Cauchy-Equations

The previously obtained expression for the stress tensor $\boldsymbol{\sigma}$ in Equation (9) can now be inserted into the equilibrium condition of Equation (1), resulting in

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} (\lambda+2\mu)\frac{\partial u_x}{\partial x} + \lambda\frac{\partial u_y}{\partial y} & \mu(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) \\ \mu(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}) & (\lambda+2\mu)\frac{\partial u_y}{\partial y} + \lambda\frac{\partial u_x}{\partial x} \end{bmatrix} + \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \mathbf{0}. \tag{10}$$

Assuming an isotropic material implies that the Lamé parameters λ and μ remain constant throughout the entire domain. This assumption simplifies the application of the divergence operator $\nabla\cdot$ to the Cauchy stress tensor $\boldsymbol{\sigma}$ and leads to the final matrix representation of the equilibrium equation

$$\begin{bmatrix} (\lambda+2\mu)\frac{\partial^2 u_x}{\partial x^2} + \mu\frac{\partial^2 u_x}{\partial y^2} + (\lambda+\mu)\frac{\partial^2 u_y}{\partial x \partial y} \\ (\lambda+2\mu)\frac{\partial^2 u_y}{\partial y^2} + \mu\frac{\partial^2 u_y}{\partial x^2} + (\lambda+\mu)\frac{\partial^2 u_x}{\partial x \partial y} \end{bmatrix} + \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \mathbf{0}. \tag{11}$$

Now finally, Equation (11) describes the static form of the *Navier-Cauchy-Equations*, which can be, once reorganized, written in compact form as

$$\mu\nabla^2\mathbf{u} + (\lambda+\mu)\nabla(\nabla\cdot\mathbf{u}) + \mathbf{f} = \mathbf{0}. \tag{12}$$

2.5 Boundary Conditions

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

$$\mathbf{u} = \mathbf{u}_D \quad \text{on } \Gamma_D \quad \text{(Dirichlet BC)}, \tag{13}$$

$$\boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{t}_N \quad \text{on } \Gamma_N \quad \text{(Neumann BC)}, \tag{14}$$

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

The condition $\mathbf{u} = \mathbf{u}_D$ prescribes the displacement field strongly on the boundary in the classical Dirichlet sense. In contrast, the Neumann condition $\boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{t}_N$ enables the application of traction forces to specific regions of the boundary.

On all remaining free boundaries Γ_{free}, a traction-free boundary condition is imposed, ensuring that no external forces act on these edges [6]. Mathematically, this follows the expression of the traction-force BCs, however, with zero as right-hand side

$$\boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{0} \quad \text{on } \Gamma_{\text{free}} \quad \text{(Traction-free BC)}. \tag{15}$$

This condition guarantees that the normal and shear stress components vanish at the unconstrained parts of the domain boundary.

3 Solution Methods for Parametrized PDEs

Solving parametrized PDEs efficiently and accurately is essential in many engineering applications. This chapter presents the numerical approach adopted in this work, which combines isogeometric collocation methods with neural networks. First, the core principles of the collocation method are introduced, followed by a detailed explanation of the architecture and training process of the proposed IgANet. The chapter concludes with the application of IgANets to parametrized simulation tasks.

3.1 Isogeometric Collocation Method

The *Collocation Method* is a numerical approach for solving differential equations by enforcing the governing equations at certain discrete points, known as *collocation points* [7]. In contrast to Galerkin-based methods, which minimize the residual over the entire domain in a weak sense, the collocation approach satisfies the differential equations at the predefined locations in a strong sense.

However, the method also comes with certain requirements for the regularity of the basis functions. Since collocation methods do not offer the possibility to balance the order of derivatives between trial and test function, the basis functions for the solutions must be sufficiently smooth to apply all derivatives of the problem's differential operator. Without adequate differentiability, the strong form of the governing equations would not even be properly defined, making a numerical solution impossible [2]. This is where IGA becomes particularly advantageous: Spline-based basis functions, such as Basis Splines (B-Splines) or Non-Uniform Rational B-Spline (NURBS), offer the required continuity across elements by design, making them an ideal fit for collocation.

The resulting *Isogeometric Collocation Method* combines the exact geometric representation from CAD with the efficiency of collocation [2]. However, since the governing equations are only enforced at specific points, the placement of these points plays a crucial role. Depending on the chosen strategy the numerical behavior can vary significantly [28]. These options and their implications are addressed in more detail in Chapter 3.1.3.

3.1.1 Mathematical Formulation

Given that the examples later in this thesis are based on two-dimensional domains, the following considers a general two-dimensional PDE in abstract form

$$\mathcal{L}[u] = f \qquad \text{on } \Omega, \tag{16}$$

where $\mathcal{L}$ is a differential operator, $u(x, y)$ is the unknown solution field, and $f(x, y)$ is a given source term.

In the Isogeometric Collocation Method, the solution $u(x, y)$ is approximated by a spline-based function $u_h(\xi, \eta)$ defined in the parametric domain (ξ, η). Specifically, u_h is expressed as a tensor product of univariate B-Spline basis functions

$$u_h(\xi, \eta) = \sum_{i=1}^{n_\xi} \sum_{j=1}^{n_\eta} N_i^p(\xi) \cdot M_j^q(\eta) \cdot \hat{u}_{i,j}, \tag{17}$$

where $N_i^p(\xi)$ and $M_j^q(\eta)$ are B-Spline basis functions of degrees p and q in the respective parametric directions, and $\hat{u}_{i,j}$ are the unknown spline coefficients associated with the solution. Additionally, n_ξ and n_η denote the number of basis functions in the ξ- and η-direction, respectively.

To determine the coefficients $\hat{u}_{i,j}$, the governing PDE is enforced at a discrete set of collocation points (ξ_α, η_β) in the parametric domain, which are mapped via the geometric mapping $\mathbf{F}$ to the physical space as

$$(x_\alpha, y_\beta) = \mathbf{F}(\xi_\alpha, \eta_\beta). \tag{18}$$

Using this mapping, the collocation system in the physical domain follows as

$$\mathcal{L}\left[u_h\right]\left(\mathbf{F}^{-1}(x_\alpha, y_\beta)\right) = f(x_\alpha, y_\beta), \qquad \alpha = 1, \ldots, n_\xi, \quad \beta = 1, \ldots, n_\eta, \tag{19}$$

where $u_h\left(\mathbf{F}^{-1}(x, y)\right)$ denotes the physical representation of the parametric spline field.

Once this system is assembled and solved, the unknown coefficients $\hat{u}_{i,j}$ are obtained. Together with the geometric mapping, they define the final approximate solution $u_h(x, y)$ over the physical domain.

3.1.2 Least-Squares Collocation

In the context of machine learning and physics-informed modeling, the concept of *Least-Squares Collocation* is also worth highlighting. Rather than enforcing the governing equations exactly at discrete collocation points, this approach minimizes the squared residual over a broader set of points [25]. This can enhance numerical stability and improve the model's ability handle inconsistencies or noise in the data [28]. Although least-squares collocation aligns well with the loss-based optimization paradigm used in neural networks, it is not further explored in this thesis.

3.1.3 Collocation Point Placement

The accuracy and efficiency of the collocation method strongly depend on the placement of the collocation points [17]. In isogeometric collocation, a common and well-established choice are the so-called *Greville abscissae*. These points are computed as the averages of p sequential knots in a p-degree B-Spline curve [11, 17]. For a univariate B-Spline of degree p, the Greville abscissa corresponding to the i-th basis function is given by

$$\xi_i^{\mathrm{G}} = \frac{1}{p} \sum_{j=1}^{p} \xi_{i+j}, \tag{20}$$

where ξ_j denotes the entries of the knot vector. In the multivariate case, collocation points are constructed as tensor products of the univariate Greville points in each parametric direction [2].

Greville points are not only easy to compute but have also proven to deliver good approximation properties in many isogeometric applications [28]. Their consistent positioning relative to the support of the basis functions ensures stable numerical convergence behavior, which is why they are widely used as default collocation points in isogeometric collocation formulations [37].

Whereas other collocation strategies, such as Demko points [2], superconvergent points [1], or Cauchy-Galerkin points [13], have been the subject of extensive research and may offer advantages in specific settings, the present work employs Greville abscissae for the IgANet application. Their ease of computation and well-documented performance in isogeometric collocation make them a reliable choice for the current implementation [28].

3.2 Isogeometric Analysis Network (IgANet)

The *Isogeometric Analysis Network* (IgANet) is a physics-informed machine learning approach that combines the strengths of classical IGA with the flexibility of artificial neural networks [23]. Instead of solving the system of equations arising from a collocation or Galerkin scheme directly, the IgANet learns to predict the unknown B-Spline coefficients that define the solution field. These coefficients are then used in a fixed B-Spline basis to evaluate the final solution $u_h(x, y)$.

The network takes geometric information like control points of the geometry and data about loads, such as body forces or boundary conditions, as input. The output is the set of B-Spline coefficients $\hat{u}_{i,j}$, which define the approximate solution. The residual of the governing PDE – for instance, $\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}$ – is evaluated at collocation points and used to compute the loss function. This loss is then minimized using gradient-based optimization, following standard neural network training procedures [3, 9].

From a broader perspective, the IgANet framework can be interpreted as a form of *Deep Operator Learning* [14]. Rather than learning a single solution, the network learns an operator that maps input data such as geometry, boundary conditions, or loads to the corresponding solution representation. Once trained, the network can instantly predict new solutions for modified configurations without the need to solve the underlying PDE again. This enables fast parameter studies and offers great potential for accelerating design and simulation workflows.

3.2.1 The IgANet Architecture

Figure 1 illustrates the structure of the IgANet approach for its application in computational solid mechanics. The central idea is to input both the spline-based geometry and the relevant boundary condition information directly into a neural network, which then predicts the corresponding displacement field $\mathbf{u}$ in vector notation. On the left side of the diagram, the inputs to the network are shown. The geometry is defined by the control points $\mathbf{P}_1, \ldots, \mathbf{P}_n$, and the boundary conditions include prescribed displacements $\mathbf{u}_{\mathbf{D}1}, \ldots, \mathbf{u}_{\mathbf{D}n_D}$ for Dirichlet boundaries and tractions $\mathbf{t}_1, \ldots, \mathbf{t}_{n_N}$ for Neumann boundaries.

These inputs are processed by a feedforward neural network, which is represented by two hidden layers with neurons denoted by $\mathbf{Z}$. The output of the network is a set of predicted spline coefficients $\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_n$. These coefficients define the displacement field $\mathbf{u}_h$, which is mapped to physical space through the geometric mapping $\mathbf{F}(\xi, \eta)$, as indicated in the center of the figure by $\mathbf{u}_h(\mathbf{F}(\xi, \eta))$.

The resulting displacement field is then evaluated by a loss function, shown on the right, which combines the residuals of the governing linear elasticity PDE and the boundary conditions. Additionally, bypasses indicate that both the geometry parameters and the boundary conditions are directly integrated into the loss function. With this network configuration, IgANet embodies the core idea of physics-informed training, where physical laws can be directly enforced in the learning process.

All in all, this architecture allows the IgANet to incorporate domain geometry, boundary conditions, and physical models into one coherent learning pipeline.

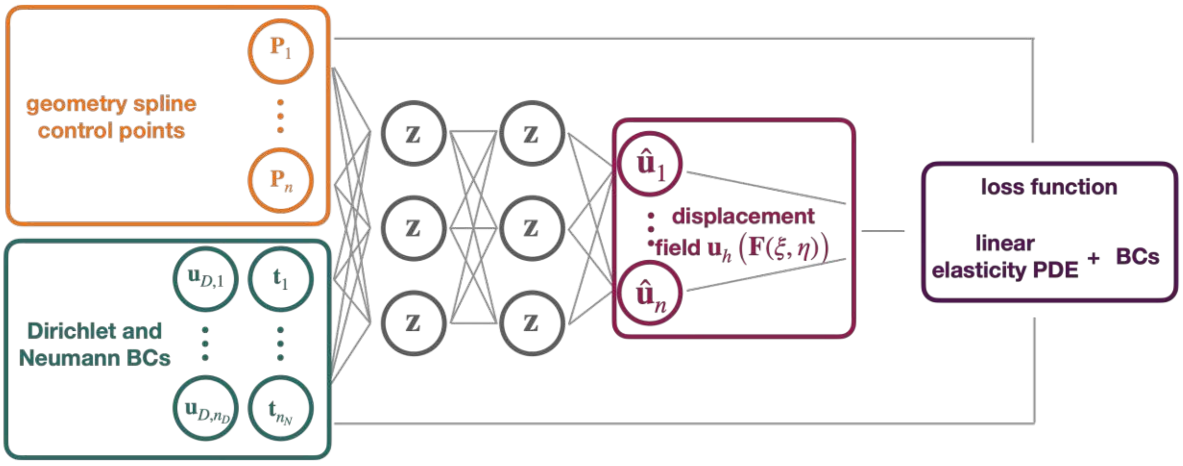


Figure 1: Schematic representation of the IgANet framework, showing how spline geometry and boundary conditions are embedded into a neural network to predict the displacement field.

3.2.2 Training Process

The training process is at the heart of the IgANet framework. It determines how the neural network learns to approximate the solution to the underlying physical problem. In contrast to traditional numerical solvers that assemble and solve a system of equations, IgANet formulates the problem as an optimization task. Here, the unknowns are not solution values at specific points, but rather the coefficients of a B-Spline basis that describes the solution globally.

This optimization is driven by a loss function that measures how well the network's prediction satisfies the governing physical laws and prescribed boundary conditions. Depending on the set-up, additional supervised terms can be included to guide the training towards a known reference solution. The training stops either when the loss reaches a specified minimum or when a maximum number of epochs is reached, preventing unnecessary computation.

Network Hyperparameters

In neural networks, hyperparameters are configuration settings that define the network architecture and govern the training process – but they are not adjusted through the learning itself [12]. In the current IgANet set-up, they include the number of layers and neurons, the choice of activation functions, as well as stopping criteria such as the maximum number of training epochs or a minimum loss threshold. These stopping criteria ensure that training halts either when sufficient accuracy is achieved or when further optimization becomes inefficient and is adjusted in every simulation setting individually.

To identify a suitable network architecture, several configurations were tested with varying numbers of neurons per layer. Networks with fewer neurons often led to faster convergence but insufficient accuracy. In contrast, at some point, increasing the number of neurons improved precision only marginally while significantly prolonging training. Based on these experiments, a basic feedforward architecture with two hidden layers of 25 neurons each was selected as a balanced compromise between computational efficiency and predictive performance.

Activation functions are a crucial component in any neural network, enabling the model to capture nonlinear relationships within the data [31]. In this work, the commonly used *sigmoid* function is applied in both hidden layers. The sigmoid compresses inputs into the range $[0, 1]$, which improves numerical stability and contributes to smoother training behavior. A linear activation is used in the output layer, meaning that the identity function $f(x) = x$ is applied and the output remains unchanged. This suits the regression character of the problem – predicting continuous B-Spline coefficients.

Loss Function

In any machine learning set-up, the loss function plays a central role [35]. It measures how well the current network prediction matches the expected output and guides the optimization process by indicating the direction in which the model parameters should be updated. A common loss function is the MSE, which quantifies the deviation between a predicted value and the desired target [27]. In physics-informed models like IgANet, the loss function typically combines several components – for example, the residuals of the underlying PDE, the violation of boundary conditions, or additional supervised terms. Each of these components contributes to the total loss, often with different weights. Choosing appropriate weights for each component is essential, as an imbalance may lead the network to prioritize one aspect of the problem while neglecting others. Once proper weights are found, IgANet minimizes the total loss during training and thus ensures that the solution not only fits available data, but also adheres to the physical laws encoded in the model.

Supervised Learning

In supervised learning, the model is provided with examples of how a system should behave, and it learns to reproduce this behavior by minimizing the error between its prediction and the known target [8]. This approach is particularly useful when a high-quality reference solution is available, as it enables the network to converge faster and more reliably.

In the context of IgANet, supervised learning can be used to complement the physics-based loss terms. Instead of relying solely on the residuals of the governing equations and boundary conditions, the network is additionally trained to match a known solution from a classical numerical solver. This allows the network to benefit from existing knowledge while still adhering to the underlying physical laws. Especially in cases where the purely physics-based training struggles to converge or leads to long training durations, supervised learning proves to be a valuable tool for improving both accuracy and efficiency.

3.2.3 Implementation Aspects

This chapter describes the code implementation of the methods and concepts introduced earlier in the context of the IgANet framework. The primary objective was to extend the existing IgANet code to enable simulations of linear elastic problems in two dimensions using isogeometric collocation. In addition to incorporating the governing equations and various boundary conditions, further focus was placed on integrating supervised learning as well as the parametrization of the PDE by changing the domain's geometry after training and predicting a new solution.

The IgANet implementation has been carried out in C++ using object-oriented programming principles [33]. The architecture is designed to be modular, allowing for easy definition of different simulation scenarios and boundary conditions. The training process relies on *libtorch*, the C++ frontend of *PyTorch* [26]. Additionally, a precomputed standard collocation solution was generated, using a Matlab script provided by Alessandro Reali. Postprocessing of all simulation results was performed with the *splinepy* library, enabled by data exchange via *JSON*-based input and output [20, 10].

The core of the implementation is realized within a newly created example file named *iganet_lin_elasticity_2D.cxx*, which contains the main logic for solving linear elastic problems in two dimensions. Apart from this, only minor adjustments of the existing code framework were necessary: a small modification was made in the calculation of the solution's Hessian matrix, and an evaluation routine was added to the backend to enable parametrization tasks. Since these changes are limited in scope, the majority of the new functionality can be found directly in the mentioned example file.

Integration of Reference Solutions

To evaluate the performance and accuracy of the IgANet predictions, two types of reference solutions are used. A Galerkin-based solution is computed with the open-source library G+Smo and embedded into the IgANet workflow [18]. This solution serves as a high-fidelity reference for convergence and validation studies.

In addition, a collocation-based solution computed with the Matlab code of Alessandro Reali is used, especially in the context of supervised learning. Due to the shared numerical foundation (collocation), this solution represents an ideal training target for the IgANet and allows perfect guidance toward the best possible approximation. Additionally, this solution has also been considered for convergence studies.

User Interface and Modularity

A strong emphasis was placed on a modular design that allows the user to fully control all relevant aspects of the IgANet simulation set-up. A dedicated input section (see Listing 1) allows for specification of material parameters, spline resolution, training configuration, boundary conditions, and reference solutions. This modular approach simplifies testing, debugging, and extending the simulation framework. Nearly all functionalities supported by the linear elasticity IgANet implementation can be adjusted within this user-input code-segment.

```
 ====== USER INPUTS ======

""" Material Parameters """
  // Young's modulus — 210 GPa for steel properties
  double YOUNG_MODULUS = 210.0;
  // Poisson's ratio — 0.25 for steel properties
  double POISSON_RATIO = 0.25;

""" Simulation Parameters """
  // max. number of epochs until training stops
  int MAX_EPOCH = 100;
  // min. loss value at which training stops
  double MIN_LOSS = 1e−8;
  // switch supervision by Matlab solution on/off
  bool SUPERVISED_LEARNING = false;
  // define path to the solution values
  std::string JSON_PATH = "/home/path/to/results_2D.json";

""" Spline Parameters """
  // nr. of control points per direction
  int NR_CTRL_PTS = 8;
  // both geometry and solution spline degree
  int DEGREE = 3;

""" Reference Simulation Parameters """
  // compute highly resolved G+Smo reference solution on/off
  bool RUN_REF_SIM = false;
  // nr. of control points per direction of ref.solution
  int NR_CTRL_PTS_REF = 100;
  // both geometry and solution spline degree of ref. solution
  int DEGREE_REF = 3;

""" Boundary Conditions """
  // Dirichlet boundary conditions
  std::vector<std::tuple<int, double, double>>
        DIRI_SIDES = {
          {1, 0.0,  0.0},      // {side, x−displ, y−displ}
        };
  // traction force boundary conditions
  std::vector<std::tuple<int, double, double>>
        FORCE_SIDES = {
          {2, 50.0,  0.0},     // {side, x−traction, y−traction}
        };
  // traction−free boundary conditions
  std::vector<int> TFBC_SIDES = {3,4};    // {sides}
  // body force (constant over the whole domain)
  std::pair<double, double> BODY_FORCE = {0.0, 0.0}; // {fx, fy}
```

Listing 1: User inputs of the IgANet simulation.

Loss Function Composition

The key function to the training process is the loss function, which is dynamically assembled based on the active simulation settings. It always begins with the so-called *Elasticity Loss*, which corresponds to the residual of the governing equation ($\nabla \cdot \boldsymbol{\sigma} = \mathbf{f}$), evaluated at all interior collocation points.

Next, the framework checks, which boundary conditions (BCs) are active. For each type – Dirichlet, applied traction, or traction-free – a separate loss term is computed based on the MSE between the predicted and target values at the respective boundary points. Since the IgANet tends to struggle particularly with enforcing Dirichlet BCs, a weight of higher than one is applied to this loss component to increase its influence within the total loss. Details on this weight are provided in Chapter 4.1.3.

If supervised learning is enabled, an additional loss term is added. This supervised loss measures the MSE between the predicted control point positions and those of the reference Matlab solution. It is scaled with a supervision weight that allows adjusting its impact to the training behavior. Among all loss components, only the Dirichlet and supervised losses are explicitly weighted. The remaining contributions – the elasticity loss, the force loss, and the traction-free loss – are included in the total loss without scaling, i.e., with a default weight of 1. Figure 2 illustrates the full structure and weighting of the loss function components.

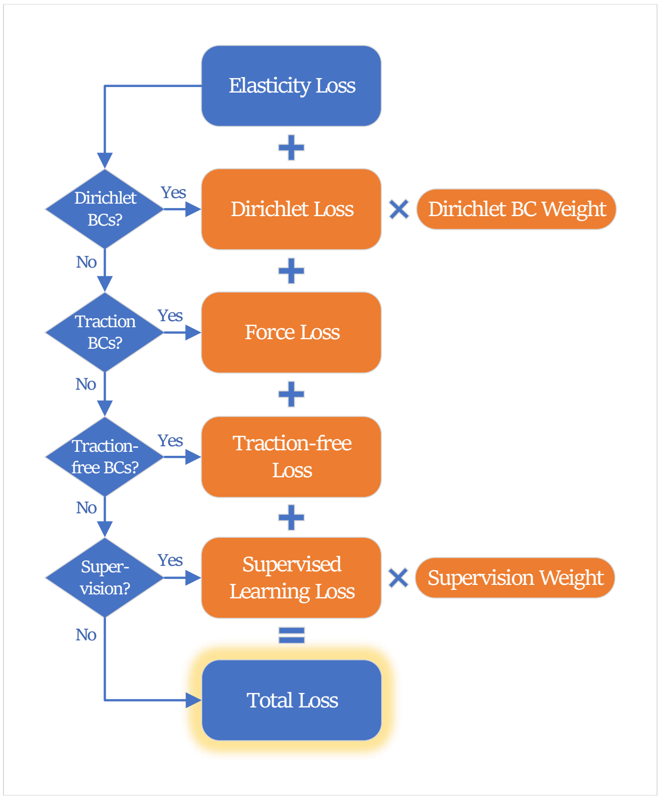


Figure 2: Composition of the total loss function in the IgANet framework, including the elasticity, boundary, and optional supervision terms.

3.3 Solving Parametrized PDEs with IgANet

One of the key motivations behind combining isogeometric analysis and neural networks lies in the ability to efficiently solve parametrized PDEs without the need to re-solve the problem from scratch for every parameter change. This concept is particularly relevant in the context of design space exploration, where engineers often evaluate the structural response of a component under several slight geometric or material modifications.

3.3.1 Motivation and Background

Traditional numerical methods such as Finite Elements or Galerkin-based IGA require a full recomputation for each new configuration, making the process time-consuming and computationally expensive [7]. The IgANet approach addresses this limitation by learning an approximation of the solution operator during training. While the network can already generalize reasonably well when trained on a single configuration, training on a variety of representative configurations further improves its ability to predict the solution coefficients for unseen inputs. This enhances the robustness of the method and allows it to capture variations in the input space more reliably. This concept is commonly referred to as *Deep Operator Learning* [14].

This capability enables rapid evaluation of new configurations and opens the door to so-called *real-time design* applications. Instead of running a complete simulation for each variant, a trained IgANet can provide fast predictions for modified domains without retraining. This significantly accelerates parameter studies and supports early-stage design processes, where many variations need to be assessed quickly.

3.3.2 Parametrized Geometry Evaluation

In this work, this idea is demonstrated by varying the geometry of the simulation domain after training. Specifically, a displacement field is learned on a unit square with predefined material parameters and boundary conditions. After training, the position of selected control points of the spline geometry are altered, while the number of control points remains unchanged. The already trained network is then used to predict the new solution without any retraining.

To enable this functionality, an evaluation function has been implemented to accept externally modified input geometries. Internally, the function simply feeds the new geometry into the already trained network and returns the corresponding displacement field. This allows the solution to be updated instantly without any retraining of the neural network.

4 Results

The following presents the results of the IgANet application in computational solid mechanics. To provide a comprehensive understanding of IgANet's behavior, two numerical experiments were conducted that use IgANet as a simple forward solver. These experiments explore different simulation set-ups, varying in boundary conditions, material parameters, and applied loads. Furthermore, in Chapter 4.3 the network's ability to generalize to modified input configurations and accurately predict the resulting material behavior is examined.

In the course of method development, it is crucial to validate the obtained results against already established methods or experimental data. This principle also applies to the implementation of the IgANet model, where the simulation outcomes are benchmarked against calculations performed by available software. Therefore, the open-source IGA solver G+Smo has been utilized in combination with a collocation code for linear elasticity provided by Alessandro Reali from the University of Pavia. The use of these software tools has facilitated a comparison of the IgANet's performance across a range of different simulation scenarios.

In the following chapters, a series of simulation figures is presented, created using the geometry tool *splinepy* [21]. *splinepy* is a powerful library for prototyping spline geometries of arbitrary dimension and degree, suited for its utilization in IGA [20]. In this particular case, *splinepy* is used as a visualization tool for the data generated by IgANet, G+Smo or the Matlab collocation code of Alessandro Reali.

Both mentioned experiments are based on a single-patch B-Spline geometry. More precisely, each set-up uses a unit patch – a unit square in 2D – with an equal number of control points in both parametric dimensions. As described in Chapter 3.1.3, Greville points were selected for collocation point placement. These points have shown good performance in previous studies and are therefore adopted for the simulations in this work [28].

The material parameters are defined through a Young's modulus of 210 and a Poisson's ratio of 0.25, both without physical units. As the geometry is modeled as a unit square, the resulting displacements and stresses are dimensionless and represent relative quantities. However, if a physical interpretation is desired, the material behavior can be associated with that of steel by assuming a Poisson's ratio of 0.25 and a Young's modulus of 210 gigapascals (GPa) [30]. In this case, the resulting stresses would be expressed in GPa accordingly.

It is noted that, in the investigated scenarios, the resulting deformations probably exceed the typical range of applicability for linear elasticity. However, as the focus of this study lies on a methodological comparison between IGA-based collocation and IgANets, the physical validity of the model equations is not further examined.

4.1 Dirichlet Experiment

The initial simulation set-up to introduce IgANet to computational solid mechanics will be called the "Dirichlet Experiment" in the following. Its name originates from the exclusive use of Dirichlet boundary conditions (BCs), without any body force or additional Neumann BC – apart from the mandatory traction-free Neumann BCs on the remaining sides.

4.1.1 Test Set-up

Dirichlet boundary conditions are applied on both the left and right sides of the reference square. On the left edge, the geometry is fixed in place with zero displacement in both x- and y-directions. On the right edge, a displacement of one unit is applied in x-direction, while zero displacement is allowed in y-direction. This set-up is illustrated in Figure 3. After running the simulation, the expected outcome is a uniformly deformed geometry, as shown in Figure 4.

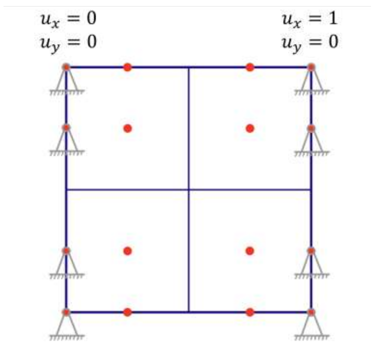


Figure 3: Reference geometry with Dirichlet BCs.

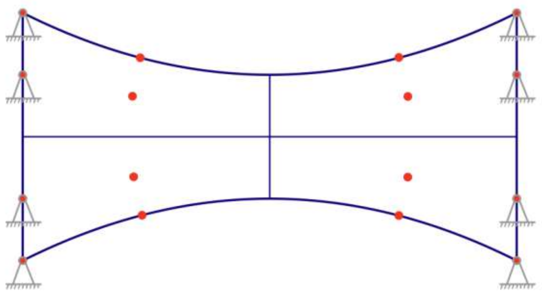


Figure 4: Expected deformation behavior of the geometry.

The body force $\boldsymbol{f}$ is assumed to be zero in this experiment. Thus, the equilibrium condition of Equation (1) simplifies to

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}. \tag{21}$$

Equation (21) is now enforced at all interior collocation points within the domain. While the collocation points on the left and right edges have mandatory displacement values due to the Dirichlet BCs, the collocation points on the top and bottom of the geometry must satisfy the traction-free BCs according to Equation (15) ($\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0}$ on Γ_{free}).

4.1.2 Traction-Free Boundary Conditions

Traction-free BCs ensure that no surface forces act on the unconstrained edges of the domain. While in a FEM context, this condition is naturally included in the weak form of the problem, it has to be imposed explicitly in collocation-based formulations.

Recalling the traction-free BC ($\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0}$), using the expression for the stress tensor $\boldsymbol{\sigma}$ of Equation (9) yields

$$\begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} \underbrace{\begin{bmatrix} 0 \\ \pm 1 \end{bmatrix}}_{\mathbf{n}} = \begin{bmatrix} \sigma_{xy} \\ \sigma_{yy} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Rightarrow \quad \begin{cases} \mu \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) = 0, \\ (\lambda + 2\mu) \frac{\partial u_y}{\partial y} + \lambda \frac{\partial u_x}{\partial x} = 0. \end{cases} \tag{22}$$

According to the set-up of the test case, Equation (22) must be enforced at all collocation points located at the top and bottom boundaries. In this case, the outward normal vector $\mathbf{n}$ aligns with the y-axis – thus, the second component is ± 1, for top and bottom side, respectively.

Figure 5 visualizes the non-physical behavior of the IgANet simulation when these traction-free BCs are omitted. Although the Dirichlet BCs on the left and right edges are well fulfilled due to a sufficiently high Dirichlet BC weight (see Chapter 4.1.3), the deformation of the remaining domain is clearly unrealistic. Varying the number of control points or spline degree may change the result, but in all cases, the simulation fails to represent the expected physical behavior – simply due to the mathematical incompleteness caused by the missing BCs. Once the traction-free BCs are correctly implemented and an appropriate Dirichlet BC weight is chosen, the simulation yields the first physically meaningful result with IgANet, as shown in Figure 6.

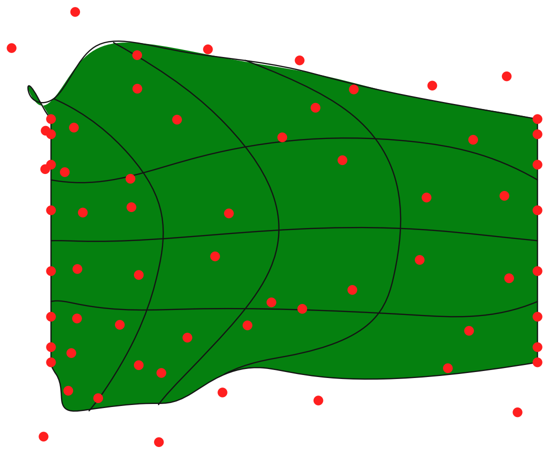


Figure 5: IgANet result **without** traction-free BCs.

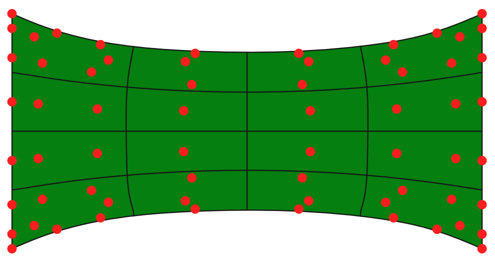
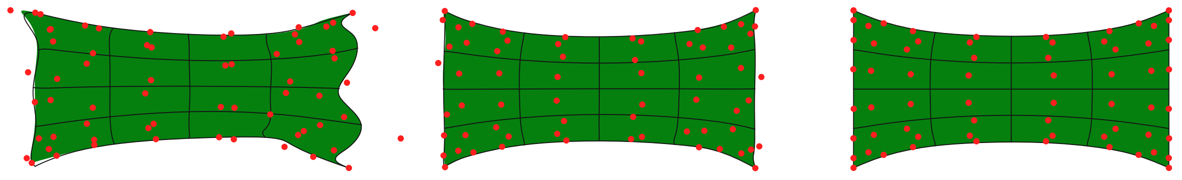


Figure 6: IgANet result **with** traction-free BCs.

4.1.3 Dirichlet Boundary Condition Weight

The initial simulations incorporating traction-free BCs yielded rather unsatisfactory results for the Dirichlet experiment, as the deformed geometry did not adhere to the prescribed Dirichlet BCs sufficiently well. This behavior, illustrated in Figure 7 (a) can be attributed to a general challenge of physics-enhanced neural networks to accurately enforce Dirichlet BCs, as for example seen for PINNs in [5]. As described in Chapter 3.2.2, the total loss is composed by several components, including the loss associated with the Dirichlet BCs. To enhance the enforcement of these constraints, the Dirichlet loss can be penalized by applying an additional weighting factor. This so-called *Dirichlet BC weight* multiplies the boundary loss term and thereby increases its relative influence during training. The impact of varying this weighting factor is shown in Figure 7, where three IgANet solutions are compared using Dirichlet BC weights of 1, 10, and 100. The results indicate that a BC weight of at least 100 is required to achieve an adequate fulfillment of the Dirichlet BCs.



(a) Dirichlet BC Weight 1 **(b)** Dirichlet BC Weight 10 **(c)** Dirichlet BC Weight 100

Figure 7: Comparison of three Dirichlet BC weights: (a) 1, (b) 10, (c) 100.

Nevertheless, the question remains whether a Dirichlet BC weight of 100 is actually the best case in terms of training time and result accuracy. To investigate this, simulations were carried out using a range of weighting factors from 10^0 to 10^{10}, analyzing their influence on the training performance and the final results. The corresponding outcomes for spline degrees 2 and 4 are shown in Figure 8. Here, the BC weight is applied on a logarithmic scale along the x-axis. The left y-axis represents both the number of epochs and the corresponding training time required by the IgANet to reach convergence. The right y-axis displays the value of the loss function at the stationary solution, i.e., the remaining loss after training has completed. A lower residual value indicates a better fulfillment of the governing equations and boundary conditions. These final loss values are also shown on a logarithmic scale to highlight their differences across the tested Dirichlet BC weights.

The trainings were performed on a standard student laptop, meaning that the reported training times can be significantly reduced by using more powerful hardware. Further, the code implementation used for training the IgANet was not optimized for computational performance, leaving additional potential for reducing runtime. Consequently, the data provided in Figure 7 should not be considered absolute, but rather serve to illustrate the influence of the Dirichlet BC weight on the network's training behavior.

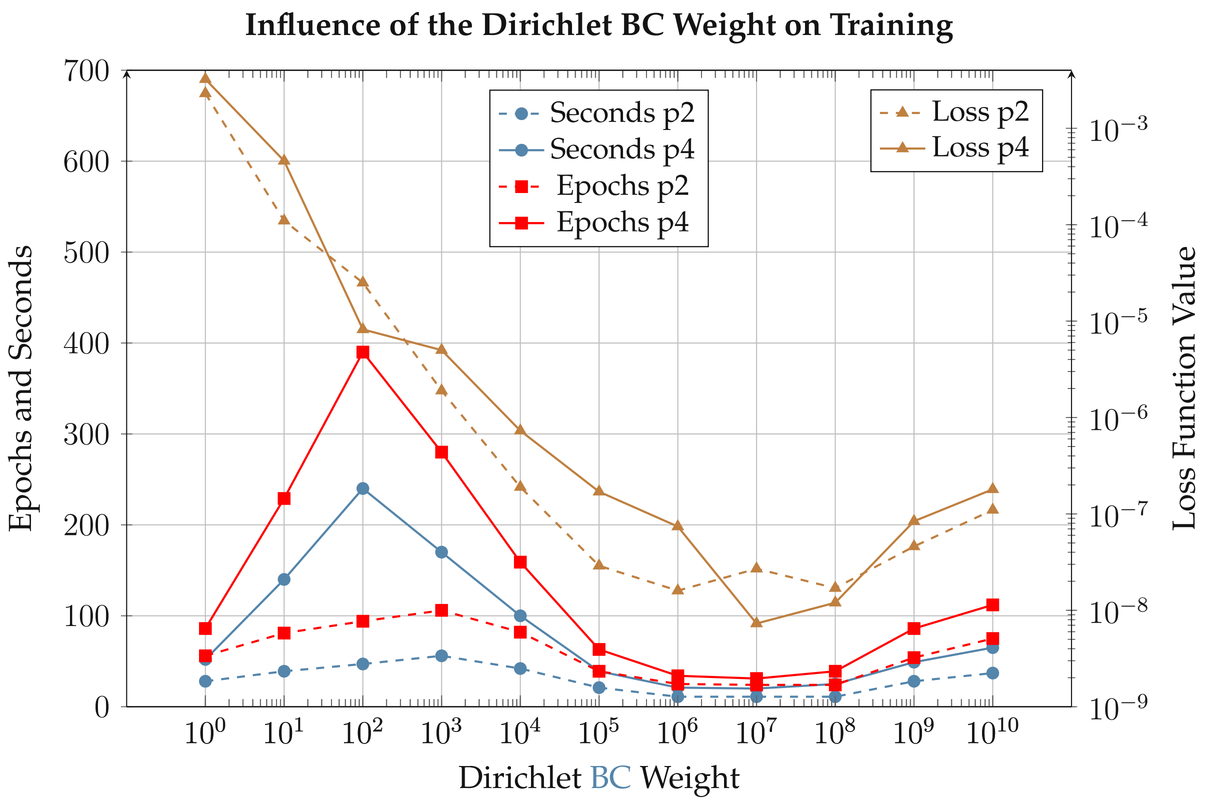


Figure 8: Comparison of the Dirichlet BC weight's influence on the training behavior of IgANet for spline degrees 2 and 4. The figure illustrates the required training time (in seconds) and the number of epochs needed to reach a stationary solution. Additionally, the values of the loss function at these stationary solutions are compared on a logarithmic scale.

For both spline degrees, the training time and the number of epochs are significantly higher for small BC 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, 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 numerical accuracy limit around 10^{-16}.

When comparing the two spline degrees, degree-4 consistently requires more training time and epochs than degree-2, which is expected due to its increased computational complexity. 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. Taking this into account, it is even more remarkable that the degree-4 IgANet solution at a Dirichlet BC weight of 10^7 yields the lowest loss value among all conducted simulations.

Overall, the results indicate that the Dirichlet BC weight has a substantial influence on both training efficiency and solution accuracy. 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.

4.1.4 The Reference Solution

The governing and constitutive equations of Chapter 2, together with the test set-up of the Dirichlet experiment, do not allow for an analytical solution of the underlying mathematical formulation. Therefore, a sufficiently accurate numerical reference solution is required, which is obtained using the previously introduced IGA framework G+Smo. Among many other features, G+Smo provides a well-established IGA solver based on the standard Galerkin formulation. As pointed out by Alessandro Reali et. al, collocation can, at best, match the accuracy of Galerkin-based IGA in the weak sense [2]. As a consequence, a Galerkin solution – such as the one computed with G+Smo – can be expected to yield better or at least equally accurate results compared to a collocation approach. For this reason, the reference solution is computed on a sufficiently fine mesh in G+Smo. Originally, a reference solution with 64×64 CPs yielded a sufficiently accurate solution, although it will be further refined later in the course of this thesis. The Galerkin-based solution is visualized in Figure 9, where 64 control points are used in each of the two parametric directions, ξ and η.

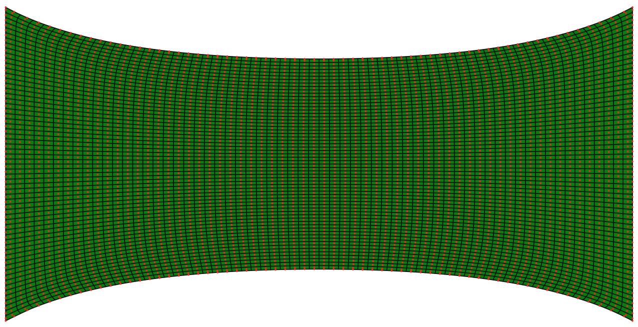


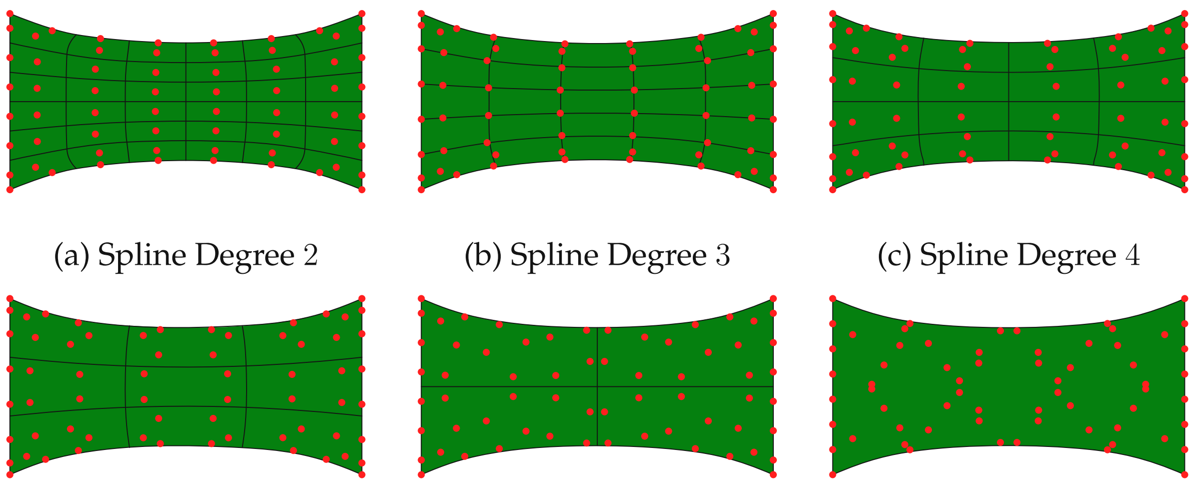
Figure 9: G+Smo reference solution on a 64×64 control point grid.

4.1.5 The Matlab Solution

Since the reference solution of Chapter 4.1.4 is computed using the Galerkin formulation, comparing it to the IgANet solution poses two potential error sources. On the one hand, discrepancies between the solutions may arise from the neural network-based calculation approach of the IgANet. On the other hand, deviations can also be caused by the different numerical formulations – namely, the Galerkin versus the collocation approach.

To distinguish between these error sources and assess the performance of the IgANet, a classical collocation-based framework is required. For this purpose, a linear elasticity collocation code, kindly provided by Alessandro Reali, was used. The original code solves a collocation problem on a geometry representing a quarter of an annulus, formulated in cylindrical coordinates. This set-up offers the advantage of providing an analytical solution, which makes it ideal for evaluating the accuracy of the collocation method.

The geometry in Alessandro Reali's Matlab code was modified to represent a square domain, and the boundary conditions were adapted to match the Dirichlet experiment. Although this set-up no longer provides an analytical solution, it still offers a great opportunity to validate the performance of the IgANet. In the following, the solution of this Matlab simulation will be referred to as the *Matlab Solution*. An interesting observation can be made when analyzing the control point distribution in these solutions. Despite producing a correct physical representation of the solution, vertical oscillations appear in the control point distribution. These artifacts increase with the degree of the spline and are shown in Figure 10.



(a) Spline Degree 2 (b) Spline Degree 3 (c) Spline Degree 4

(d) Spline Degree 5 (e) Spline Degree 6 (f) Spline Degree 7

Figure 10: Comparison of different spline degrees in the Matlab solution with $8{\times}8$ CPs.

When a sufficient number of control points is used in relation to the degree of the spline, the control point distribution appears more regular and as expected, as illustrated in Figure 11 and Figure 12. The underlying cause of these oscillations remains unclear and appears to be an open question. However, it can be stated that the effects stem from the collocation method itself rather than from any implementation error. It is also possible that the use of linear elasticity under comparatively large deformations may cause such artifacts, although this cannot be confirmed within the scope of this work. Despite these irregularities in the control point distribution, the overall accuracy of the Matlab solution is very high in comparison to the Galerkin reference solution. Therefore, the observed artifacts can be considered negligible in the context of evaluating the solution quality.

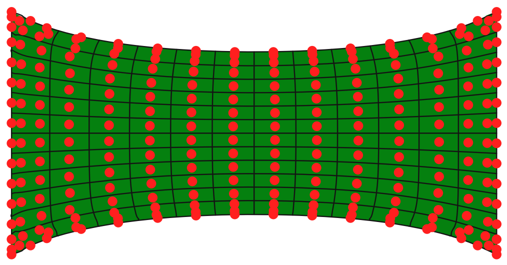


Figure 11: Matlab solution with 16×16 CPs and degree-4.

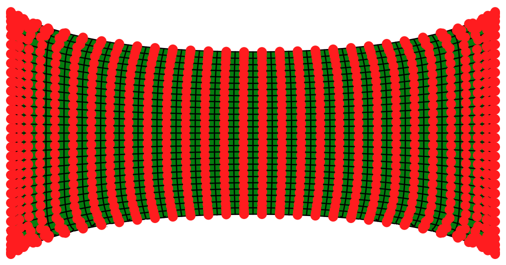


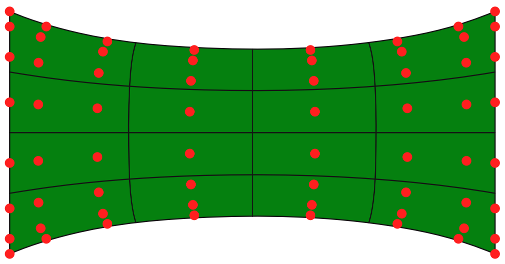
Figure 12: Matlab solution with 32×32 CPs and degree-6.

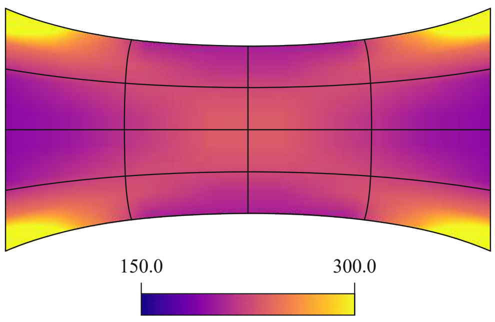
4.1.6 The G+Smo Solution

The reference solution (Chapter 4.1.4) and the Matlab solution (Chapter 4.1.5) have already been introduced and serve as two points of comparison for the IgANet results. In addition to these, a third simulation, referred to as the *G+Smo Solution*, was conducted. This simulation is based on the same number of control points and spline degree as used in the IgANet set-up, providing a direct comparison under identical numerical settings. In this way, it is possible to evaluate the performance of IgANet-collocation in comparison to G+Smo Galerkin-IGA at an equal simulation setting.

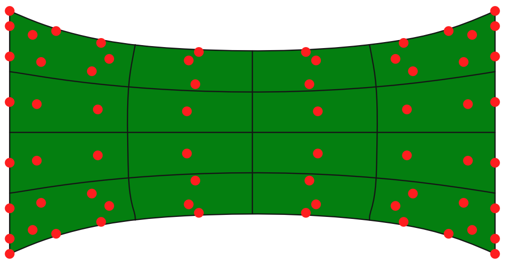
4.1.7 Simulation Results

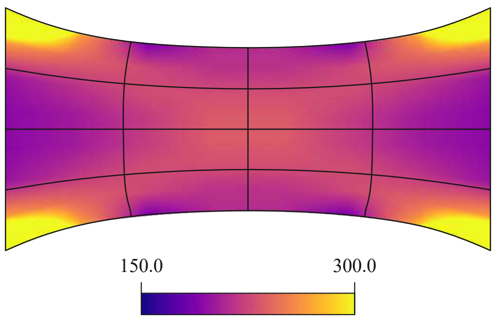
With the Dirichlet BCs, the traction-free Neumann BCs, and a sufficiently large Dirichlet BC weight in place, the first meaningful comparisons between the IgANet simulations and the previously introduced established methods can be performed. In addition to the deformation of the geometry, the internal stress fields are also computed and visualized for all methods. To enable this comparison, the G+Smo implementations for both the *Reference* and the *G+Smo solution* were extended to compute the corresponding stresses within the domain. Similarly, the *Matlab solution* was adapted to evaluate the stresses at the collocation points. Finally, the IgANet framework was also equipped with routines to calculate the von Mises stress at the collocation points. In Figure 13, a first visual comparison of the resulting displacement and stress fields are provided, based on a test set-up of eight control points per direction and a spline degree of 4.


(a) G+Smo Solution

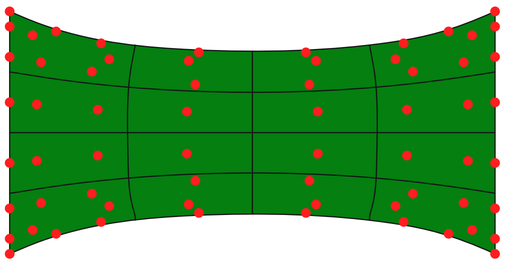


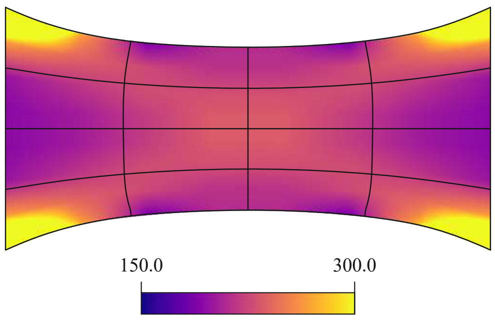
(b) G+Smo Stress Distribution


(c) Matlab Solution



(d) Matlab Stress Distribution


(e) IgANet Solution



(f) IgANet Stress Distribution

Figure 13: Comparison of different simulation methods, supplemented by their respective stress distribution. Set-up: 8×8 CPs and degree-4.

The stresses, visualized in Figure 13, are limited to a maximum value of 300 in order to account for mathematical 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 the IgANet produces a sufficiently accurate result. However, a more detailed analysis is required to precisely assess the accuracy of the IgANet solution.

Mean Squared Error

As mentioned in Chapter 4.1.4, the reference solution is considered the best achievable result within this framework. It is computed using 64 control points (CPs) in both ξ- and η-direction, resulting in a total of 4096 CPs. This configuration is simulated via G+Smo and yields a highly resolved Galerkin-IGA solution.

This solution now serves as the reference for evaluating the quality of all other simulation results. To this end, a uniform grid of 100$\times$100 evaluation points is used for comparing the solutions. The metric chosen for this comparison is the MSE over all 10.000 evaluation points, measuring the pointwise deviation between each candidate solution and the reference. This procedure evaluates the spline at the mentioned points and is thus independent of the number and placement of control points in the respective methods. Therefore, it enables a direct comparison of, for example, an IgANet solution using only 8$\times$8 CPs with the 64$\times$64 CPs degree-4 reference solution.

Whereas the G+Smo simulations are always based on the Galerkin formulation, both the Matlab and IgANet solutions rely on the collocation method. Accordingly, it is to be expected that the IgANet result resemble those of the Matlab implementation more closely than those from G+Smo. Interestingly, the comparison reveals that the IgANet solution matches the Matlab results almost perfectly. Across all tested configurations, the mean squared errors between the two were identical up to differences in the order of 10^{-11}. Based on this observation, the IgANet and Matlab solutions can be treated equivalently and referred to jointly as the *Collocation Solution*.

This leaves the coarse G+Smo simulation to be evaluated and compared against the collocation results. The corresponding MSE values are shown in Figure 14, where the collocation solution is plotted in blue and the G+Smo solution in red for various control point configurations. For each case, simulations were carried out using spline degrees 2 and 4, represented by dashed and solid lines, respectively.

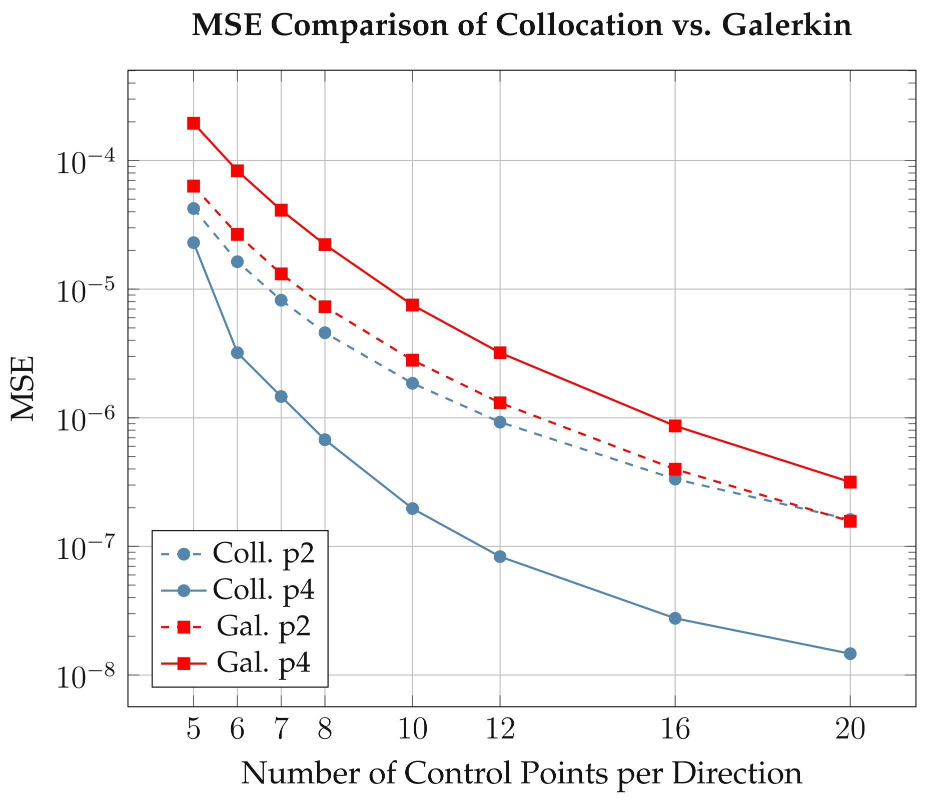


Figure 14: Comparison of Galerkin vs. Collocation at different control point grids in terms of the MSE to the highly refined G+Smo reference solution.

The graphs in Figure 14 offer valuable insights into the behavior of the different simulation approaches. First, they clearly show that the MSE decreases with an increasing number of control points across all configurations. Second, rather unexpectedly, the Galerkin solution from G+Smo using spline degree-2 appears to be more accurate than the one using degree-4. Third, the collocation method significantly outperforms the Galerkin approach in terms of accuracy within the tested range.

Moreover, the results indicate that the collocation solution with spline degree-4 is more accurate than its degree-2 counterpart, which aligns well with expectations based on the method's characteristics. In contrast, the Galerkin approach seems to require a larger number of control points to reach a level of accuracy similar to the one of collocation in this particular set-up. This observation is further investigated in Figure 15, where the simulation is extended to 64 CPs per parametric direction.

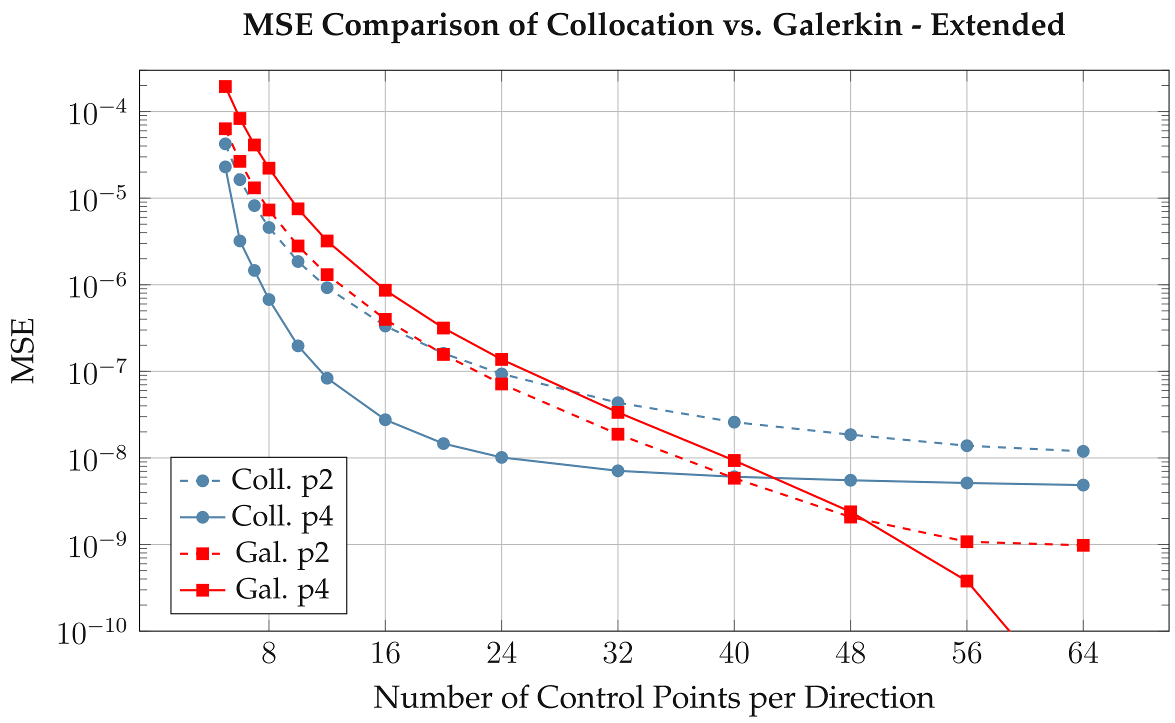


Figure 15: Comparison of Galerkin vs. Collocation at different control point grids in terms of the MSE to the highly refined G+Smo reference solution. Extension of Figure 14 to 64×64 CPs.

The extension to 64 CPs per direction offers further insight into the accuracy behavior of the two different numerical approaches. Whereas the collocation method achieves high accuracy with relatively few control points, the Galerkin method surpasses it in accuracy from around 40 CPs onward. The results indicate that while the collocation solution with degree-4 significantly outperforms the G+Smo solution in terms of accuracy for small numbers of control points, its error levels off beyond 32 CPs per direction. In contrast, the Galerkin solution computed with G+Smo continues to improve its fidelity steadily across the entire range of control point refinements. Since all MSE values are calculated with respect to the reference solution (Galerkin, degree-4, 64×64 CPs), the final Galerkin simulation with degree-4 effectively reproduces the reference itself – leading the MSE to approach zero in the last configuration.

Convergence Study

To verify and better understand the previously observed results – particularly the surprisingly strong performance of the collocation method in certain configurations – a systematic convergence study was carried out. The aim was to assess the accuracy of both collocation- and Galerkin-based solutions for increasing spline resolutions and to compare their convergence behavior. Accordingly, the accuracy of the Galerkin and collocation solutions was assessed relative to their respective high-resolution reference solutions.

For this purpose, the reference solutions were refined further. The Galerkin reference solution, computed with G+Smo, was increased from 64×64 to 100×100 CPs. In addition, a high-resolution collocation reference was introduced, obtained via the existing Matlab implementation. For both cases, the accuracy of coarser configurations was evaluated against these 100×100 CPs references using a dense evaluation grid of 1000×1000 points, ensuring that local deviations, especially near the boundaries or corners, were fully captured. The MSE was then calculated between the predicted displacement fields of various test configurations and the respective reference solutions. This was performed for spline degrees $p = 2$, $p = 3$, and $p = 4$ to obtain a more complete picture of the convergence behavior of each method.

From theoretical considerations, Galerkin-based isogeometric methods are expected to show an error of order $\mathcal{O}(h^{p+1})$, where h denotes the characteristic mesh size [2, 28]. Since the mesh size is inversely proportional to the number of control points per direction n, this corresponds to $\mathcal{O}(n^{-(p+1)})$ and thus a convergence rate of $p + 1$.

For collocation methods, the expected convergence behavior differs. For even spline degrees p, the error is typically $\mathcal{O}(h^p)$, while for odd degrees it reduces to $\mathcal{O}(h^{p-1})$. In terms of the number of control points, this translates to $\mathcal{O}(n^{-p})$ for even p, and $\mathcal{O}(n^{-(p-1)})$ for odd p.

The observed convergence behavior is visualized in Figure 16, which shows the MSE plotted on a double-logarithmic (log-log) scale with respect to the number of control points per direction n.

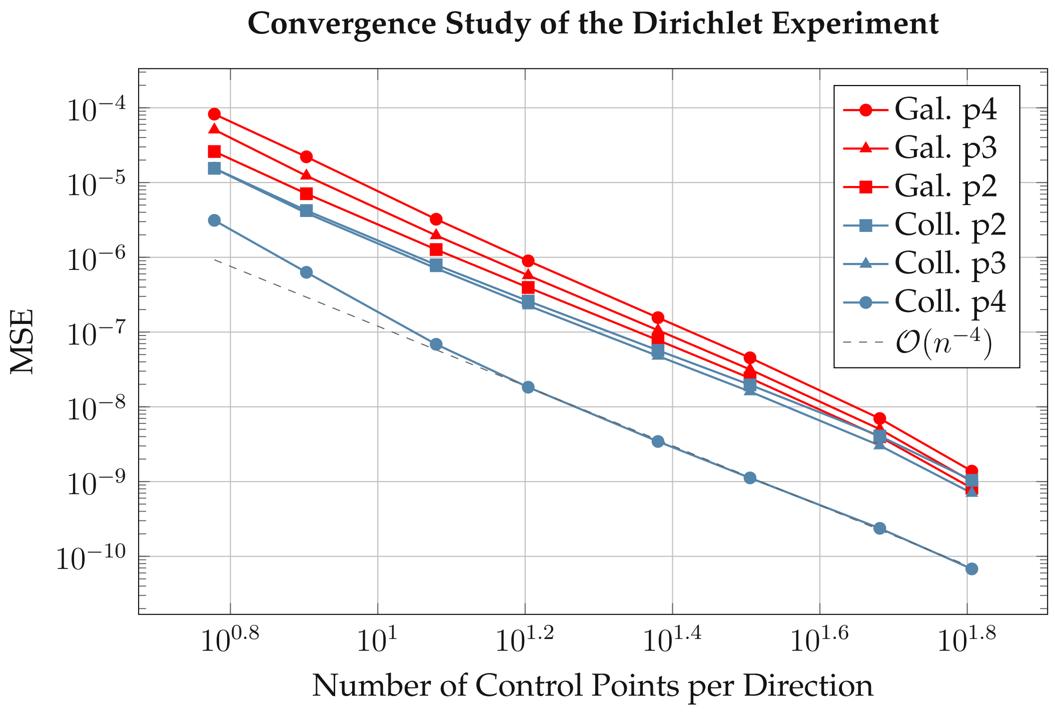


Figure 16: Convergence study for different spline degrees using collocation and Galerkin-based solutions against their respective reference solution.

The convergence plot shows that the theoretical rates are only partially observed. Among all test cases, the collocation method with degree $p = 4$ exhibits the most consistent and expected behavior. With increasing numbers of control points, the convergence rate stabilizes at approximately 4, in line with theoretical predictions. For this case, a dashed reference line with a rate of $\mathcal{O}(n^{-4})$ is added in the plot.

For other spline degrees, however, the convergence behavior appears less uniform. Some of the curves converge faster or slower than theoretically expected, and in some regions the decay rate is not linear on the log-log scale. These observations suggest that the convergence behavior is not only determined by the method and degree, but also strongly influenced by the problem set-up.

To gain further insight into these trends, the convergence rates are quantified explicitly through linear regression analysis of the data in Figure 16. The results of this comparison are presented in Figure 17, taking into account the absolute values of the convergence rates.

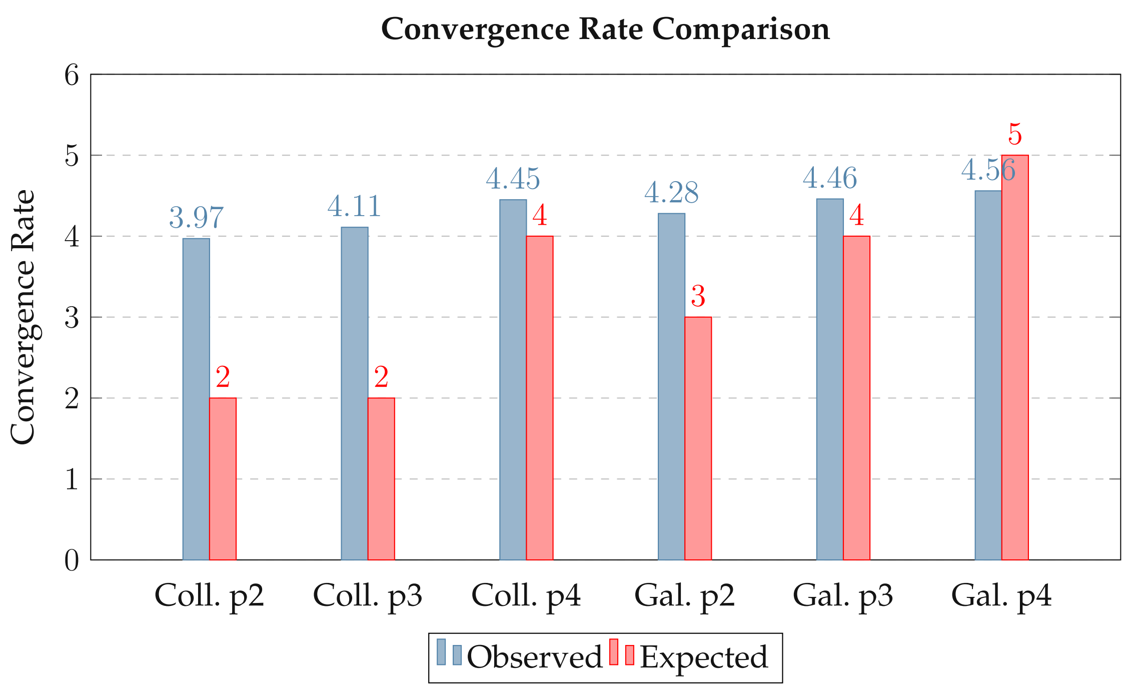


Figure 17: Comparison of observed and expected convergence rates for different methods and spline degrees.

The comparison of observed and expected convergence rates in Figure 17 reveals some interesting findings. Most notably, nearly all methods exhibit faster convergence than theoretically predicted. This applies to both Galerkin and collocation methods across all investigated spline degrees, except for Galerkin with degree $p = 4$, which shows a slightly lower rate than expected.

This overall behavior is somewhat surprising but can likely be attributed to the exceedance of the linear elastic regime or to the simplicity of the test set-up. The Dirichlet experiment uses a constant displacement on a square domain with homogeneous material properties, resulting in a rather uniform and smooth solution field. In such cases, higher convergence rates could be a familiar phenomenon.

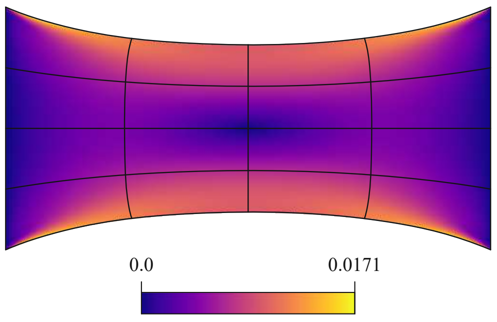
It should also be noted that, in the current set-up, the applied displacement already exceeds the limits of the linear elasticity regime. This could introduce inconsistencies in the underlying physical model and might slightly distort the observed convergence behavior. To rule out the possibility of errors originating from the Galerkin solver in G+Smo, further studies with reduced displacements were conducted. These confirmed that, under more conservative loading conditions, the expected convergence rates were indeed recovered.

Nonetheless, the results presented here remain insightful. They highlight that convergence behavior is not determined by theory alone but can be influenced significantly by the simulation context. Moreover, the expected trend of improved convergence with increasing spline degree is clearly visible in both Galerkin and collocation methods – confirming a key principle from approximation theory [4].

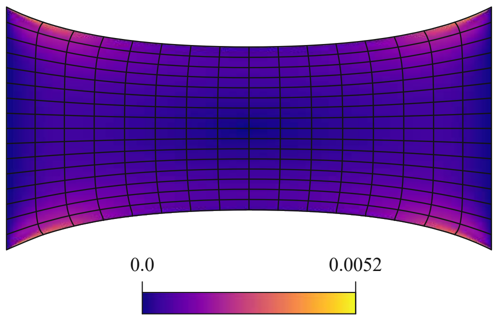
Absolute Error Distribution

While the MSE offers meaningful insight into the overall accuracy of a solution, it does not provide information about the spatial distribution of the error. To address this limitation, an additional set-up was established that enables visualization of the error distribution. For this purpose, the previously introduced 100×100 evaluation point grid is reused. At each point, the absolute error between a given solution and the Galerkin reference solution of Chapter 4.1.4 is computed and plotted onto the deformed geometry. This allows for spatial localization of the error and supports a deeper understanding of the behavior of the different simulation approaches. Some examples of such plots are presented in Figure 18 for configurations of 8 and 20 CPs per direction and spline degree-4. Each visualization includes a color bar indicating the absolute error, scaled individually from zero to the respective maximum error in each case.

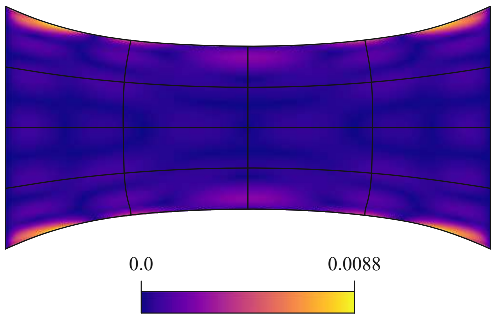
The plots in Figure 18 highlight the remarkable agreement between the Matlab and IgANet solutions. Both approaches yield exactly the same maximum error, and the overall error distribution is nearly identical. This outcome is a direct consequence of the shared collocation method, but also underlines the high potential of the IgANet. In contrast, the G+Smo solution exhibits higher error values for both control point configurations – although the reference is also a Galerkin-based solution. In all conducted simulations, both collocation and Galerkin, it becomes evident that increasing the number of control points leads to a reduction in the overall error. Additionally, the maximum error tends to shift towards the corners of the domain, which constitute geometrical singularities.



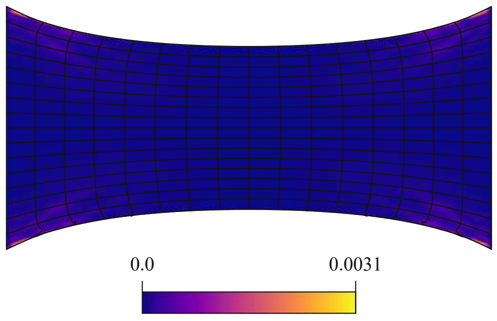
(a) Error G+Smo 8×8 CPs



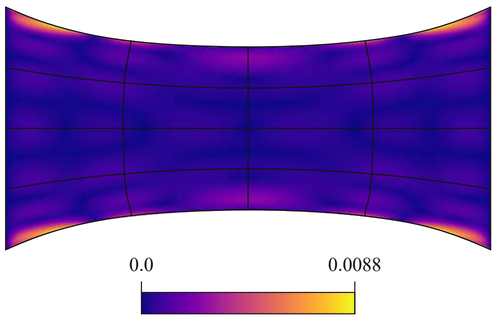
(b) Error G+Smo 20×20 CPs



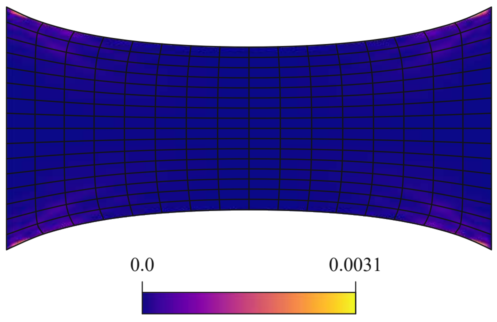
(c) Error Matlab 8×8 CPs



(d) Error Matlab 20×20 CPs



(e) Error IgANet 8×8 CPs

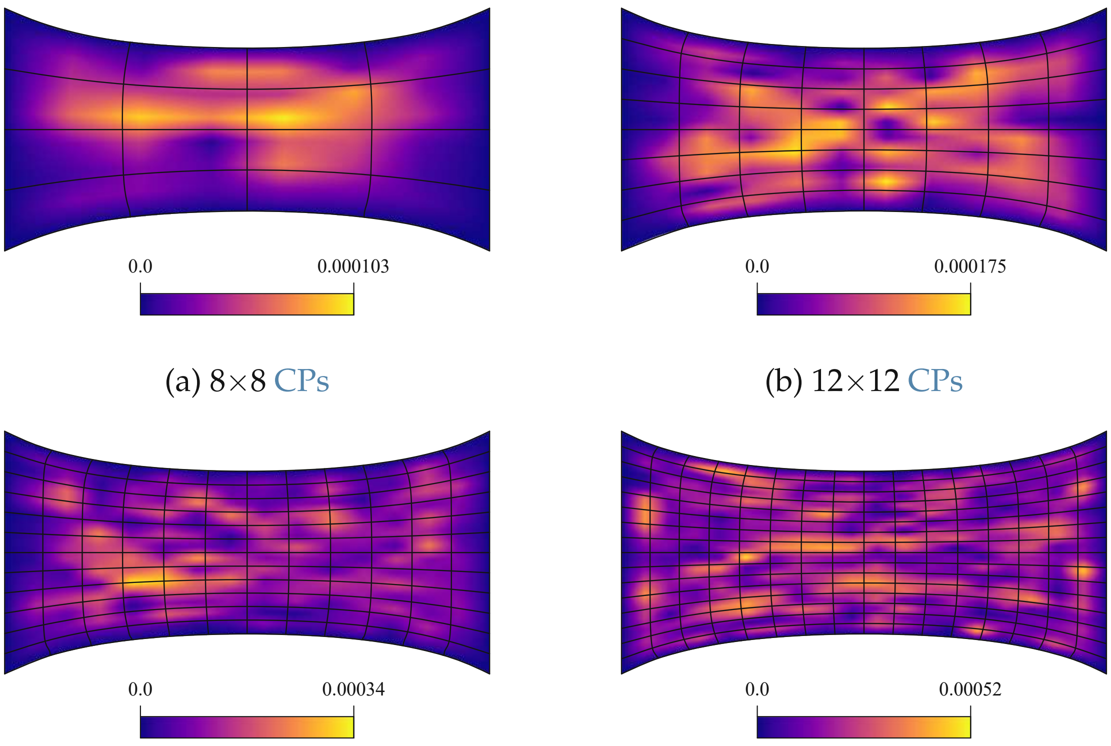


(f) Error IgANet 20×20 CPs

Figure 18: Absolute Error Distribution. Visualization of the deviation to a fine resolved reference spline. Comparison between 8×8 and 20×20 CPs at different simulation methods.

Elasticity Error Distribution

Within the IgANet framework, the different contributions to the loss function are minimized. However, unlike the Matlab solution, which enforces the equations in strong from and fulfills them numerically exact at each collocation point, the neural network-based approach of IgANet cannot guarantee such exactness. As a result, numerical errors arise due to the approximative nature of the neural network. Based on the linear elasticity's governing equilibrium equation $\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}$ (see Equation (21)), the IgANet aims to minimize the divergence of the stress tensor, ideally pushing it close to zero. After training, the residual divergence at each collocation point is evaluated, with any deviation from zero interpreted as an error, referred to here as the *Elasticity Error*. The spatial distribution of this error for different control point settings is visualized in Figure 19.



(a) 8×8 CPs (b) 12×12 CPs

(c) 16×16 CPs (d) 20×20 CPs

Figure 19: Elasticity error distributions of the IgANet for different control point numbers, using a fixed spline degree of $p = 4$.

The plots provide valuable insight into the accuracy of the IgANet. 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 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 IgANet solution remains nearly indistinguishable from that of the Matlab implementation, as discussed in the previous paragraphs.

Poisson's Ratio

The *Poisson's ratio* ν describes the ratio of transverse contraction to axial extension in a material subjected to uniaxial loading [6, 16]. When a material is stretched in one direction, it contracts in the perpendicular direction, and vice versa. This effect is characterized by

$$\nu = -\frac{\varepsilon_{\perp}}{\varepsilon_{\parallel}}, \tag{23}$$

where $\varepsilon_{\perp}$ is the transverse strain, and $\varepsilon_{\parallel}$ is the axial strain. The Poisson's ratio is a dimensionless material property that depends on the internal structure and composition of the material [32].

For most isotropic elastic materials, ν typically ranges between 0.2 and 0.5, with metals commonly exhibiting values around 0.3, and rubber-like materials approaching 0.5 [32]. A Poisson's ratio of 0.5 indicates an incompressible material, meaning that under deformation, the volume remains constant [16]. On the other hand, materials with negative Poisson's ratios, called *auxetic materials*, expand laterally when stretched, exhibiting unusual mechanical behavior [29].

Since the Poisson's ratio is a material parameter, it should ideally remain constant throughout the entire domain. However, especially in the presence of BCs, the local ratio between axial and transverse strain may not exactly reproduce the prescribed Poisson's ratio at every material point. In order to investigate this behavior, Poisson's ratio was recomputed at every collocation point using *Hooke's Law*, based on the stress results obtained from the simulations [32]. First, the strains are computed as

$$\varepsilon_{xx} = \frac{1}{E}\left(\sigma_{xx} - \nu\sigma_{yy}\right), \tag{24}$$

$$\varepsilon_{yy} = \frac{1}{E}\left(\sigma_{yy} - \nu\sigma_{xx}\right). \tag{25}$$

where E is Young's modulus and ν is Poisson's ratio. Inserting the expressions for E and ν in terms of the Lamé parameters λ and μ (see Equation (7)) into the above equations yields

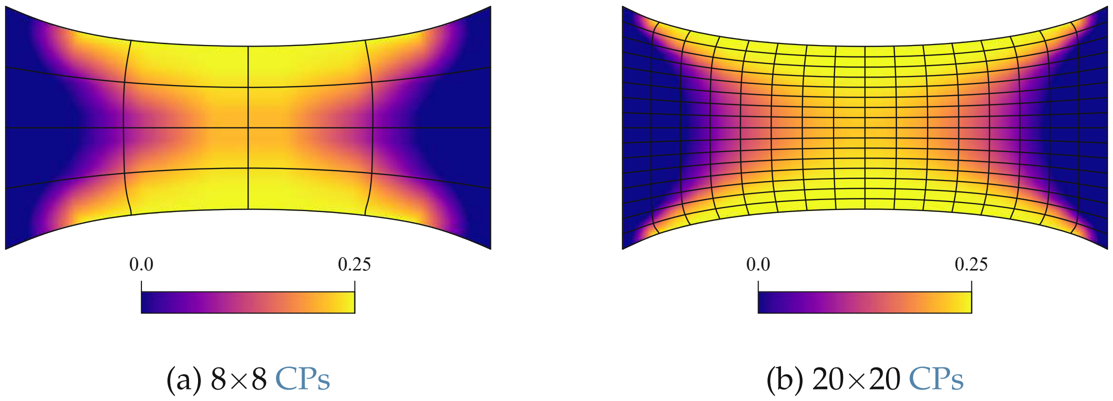
$$\varepsilon_{xx} = \frac{\lambda+\mu}{\mu(3\lambda+2\mu)}\left(\sigma_{xx} - \frac{\lambda}{\lambda+2\mu}\sigma_{yy}\right), \tag{26}$$

$$\varepsilon_{yy} = \frac{\lambda+\mu}{\mu(3\lambda+2\mu)}\left(\sigma_{yy} - \frac{\lambda}{\lambda+2\mu}\sigma_{xx}\right). \tag{27}$$

The Dirichlet test set-up is particularly well-suited for this kind of evaluation, as it induces a predominantly axial deformation. This allows the use of the definition of Poisson's ratio from Equation (23). With ε_{xx} in axial and ε_{yy} in transverse direction, the recalculation of the Poisson's ratio ν follows as

$$\nu = -\frac{\varepsilon_{yy}}{\varepsilon_{xx}}. \tag{28}$$

These calculations were performed at every collocation point to assess how well the predefined Poisson's ratio is reproduced by the solution. The results are presented in Figure 20.



(a) 8×8 CPs (b) 20×20 CPs

Figure 20: Comparison of Poisson's ratio for different numbers of control points at spline degree-4.

The plots show excellent agreement with the predefined Poisson's ratio of 0.25 at the center regions of the upper and lower edges of the domain. This behavior appears largely independent of the resolution of the collocation points. In contrast, the deviation from the target value along the left and right boundaries is expected and physically plausible, as these areas are constrained by Dirichlet BCs. There, the geometry is not free to contract laterally, preventing the typical Poisson effect from fully manifesting. Overall, the plots confirm that the IgANet solution behaves in a physically consistent manner and reproduces the material response accurately within the limitations imposed by the boundary conditions.

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 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 to 20×20. Additionally, the value of the loss function at the stationary solution was recorded in each case. The corresponding results are presented in Figure 21.

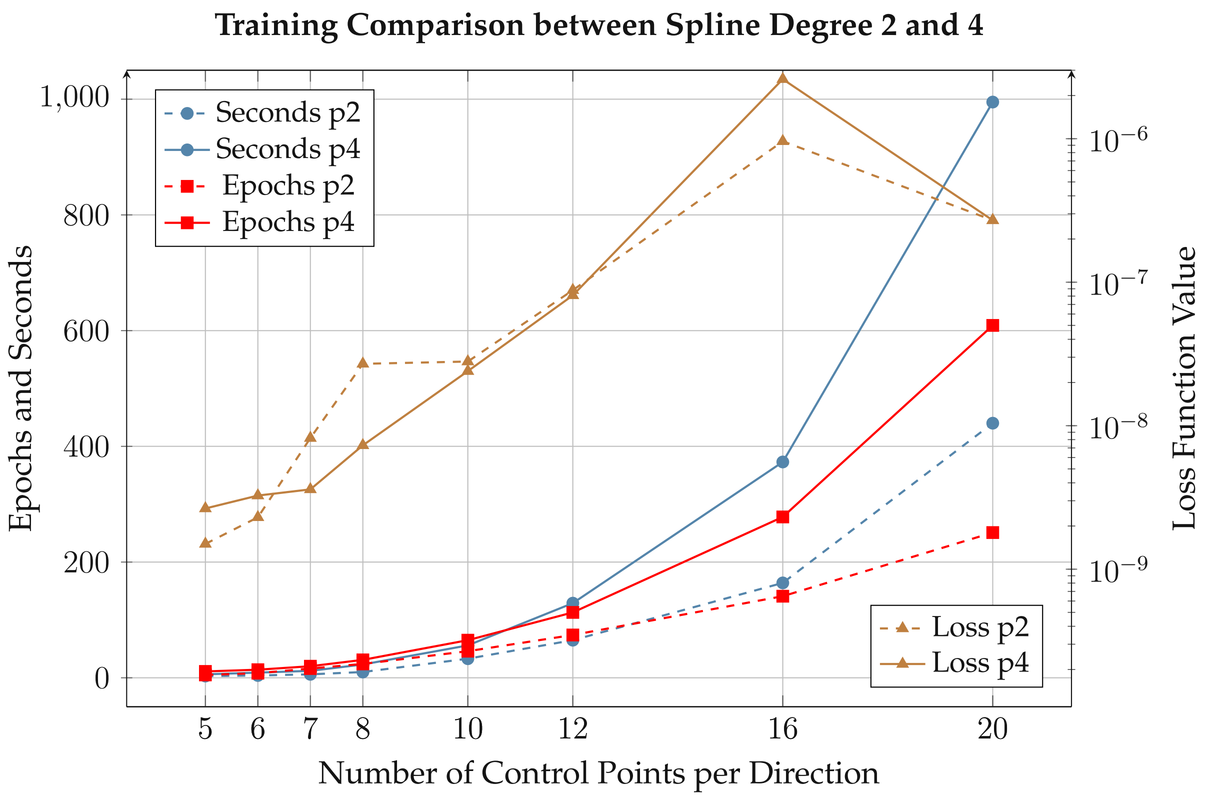


Figure 21: Comparison of the training behavior of IgANet for different control point configurations. The figure illustrates the required training time and number of epochs needed to reach a stationary solution. Additionally, the values of the loss function at these stationary solutions are compared.

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 rises 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 CPs case, the loss value decreases again, deviating from the previous trend.

4.2 Traction Force Experiment

The second experiment within the scope of applying IgANet to computational solid mechanics introduces an external traction force acting on the domain. As in the previous experiments, the simulation results obtained by IgANet are validated by comparison with the established G+Smo and Matlab implementations.

In this set-up, the spline degree is fixed to 3. This allows for analyzing the IgANet's performance at an intermediate spline degree, offering a contrast to the previously tested degrees 2 and 4.

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

4.2.1 Test 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, which leads to a clearly visible deformation while not exceeding the limits of linear elasticity too far. 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 Figure 22, and an expected deformation result is illustrated in Figure 23.

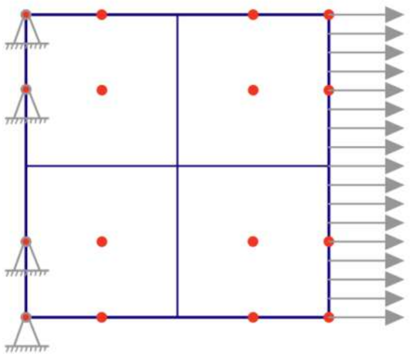


Figure 22: Reference geometry with both Dirichlet and Traction-Force BCs.

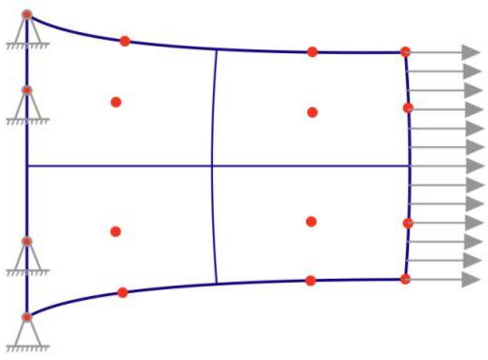


Figure 23: Expected deformation behavior of the geometry.

Analogously to the Dirichlet experiment described in Chapter 4.1, the body force **f** is set to zero throughout the domain. This again reduces the governing equilibrium condition to its simplified form in Equation (21), which is enforced at all interior collocation points.

4.2.2 Dirichlet Boundary Condition Weight

As already discussed in Chapter 3.2.3, the loss function used during IgANet training consists of several components. One of which accounts for the fulfillment of Dirichlet BCs. Since IgANet tends to struggle with enforcing these boundary conditions precisely, a Dirichlet BC weight is introduced to weight this part of the loss stronger.

Accordingly, this approach is also applied in this experiment, where the Dirichlet-related loss is multiplied by a weighting factor to improve the adherence to the prescribed displacements. Figure 24 shows three simulation results for different Dirichlet BC weights. It can be observed that a factor of at least 100 is required to ensure a sufficiently accurate representation of the Dirichlet boundaries.

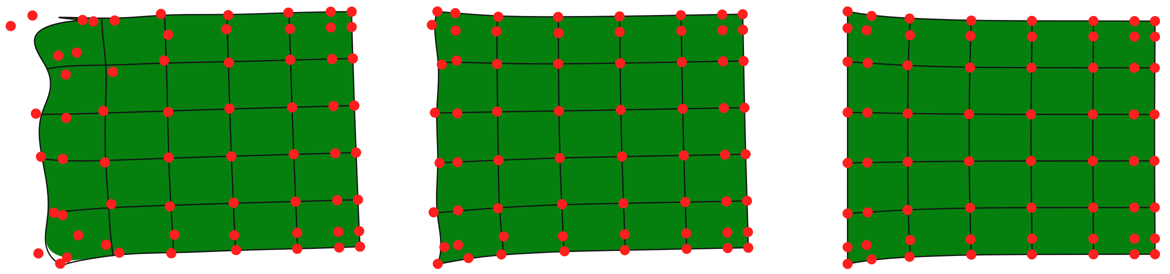

(a) Dirichlet BC Weight 1 **(b)** Dirichlet BC Weight 10 **(c)** Dirichlet BC Weight 100

Figure 24: Comparison of three Dirichlet BC weights: (a) 1, (b) 10, (c) 100.

While a BC weight of 100 seems to already enable acceptable results, its influence on the training behavior is further investigated in this experiment. Figure 25 shows the effect of varying BC weights on both training time and loss value at the end of the training. Similar to the findings from the Dirichlet experiment in Chapter 4.1.3, a weighting factor in the range of 10^5 to 10^8 proves to be optimal. In this range, the training time reaches a minimum, and especially at a factor of 10^8, the value of the loss function is minimized. Based on these observations, a BC weight of 10^8 is used for all subsequent simulations in this experiment.

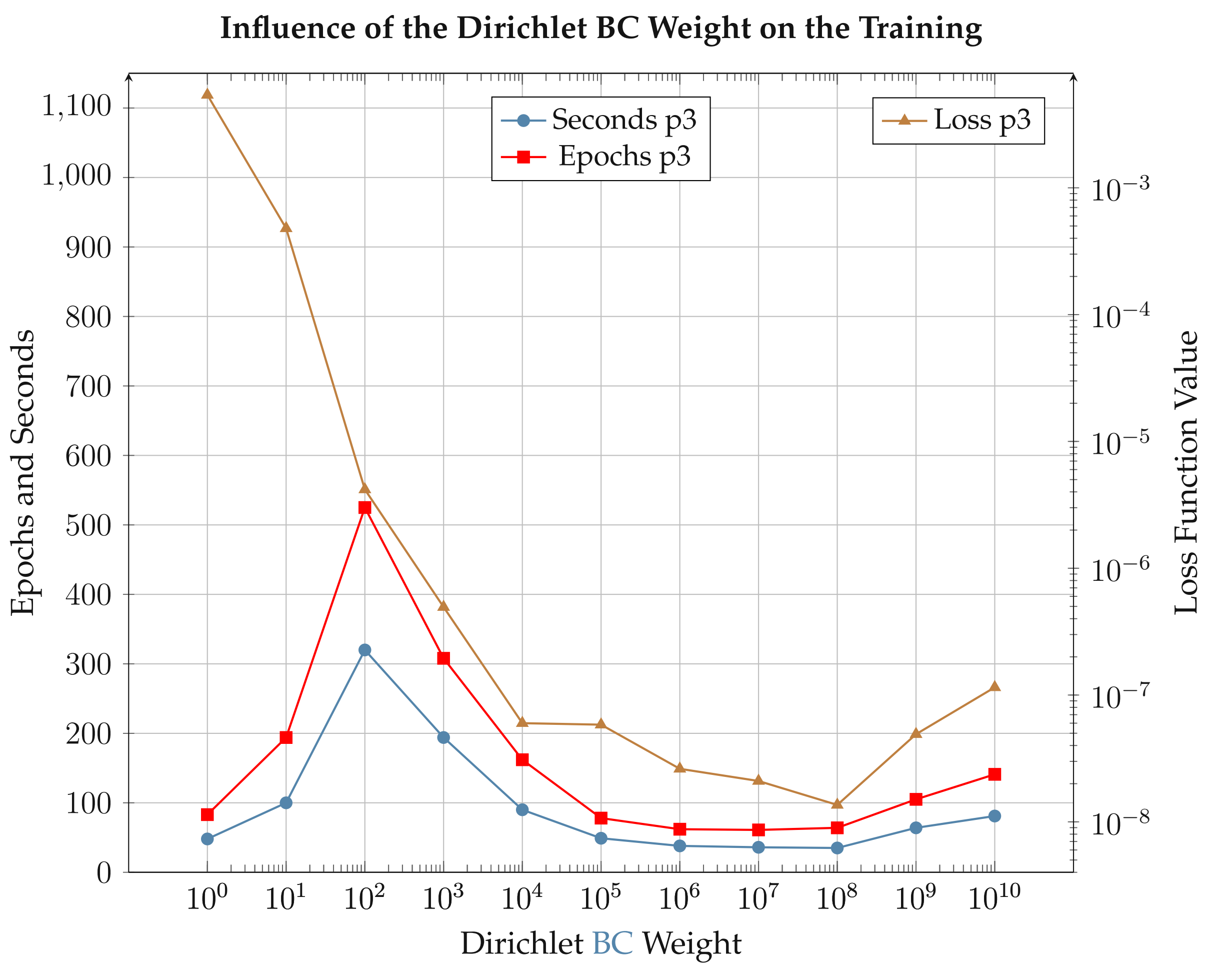
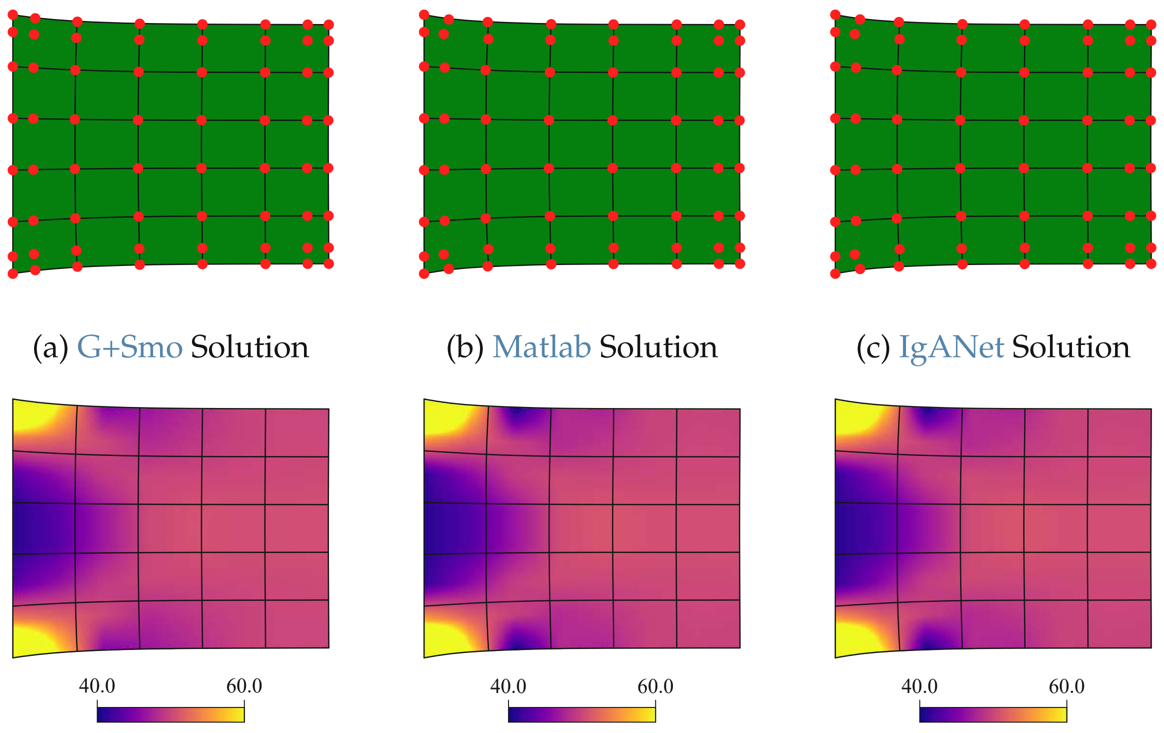


Figure 25: Investigation of the Dirichlet BC weight's influence on the training behavior of IgANet for spline degree-3. The figure illustrates the required training time and the number of epochs needed to reach a stationary solution. Additionally, the values of the loss function at these stationary solutions are compared on a logarithmic scale.

4.2.3 Simulation Results

The simulation results of G+Smo, Matlab, and IgANet for the traction force experiment are presented in the following. Based on the numerical experiments of Chapter 4.2.2, a Dirichlet BC weight of 10^8 has been chosen and it is recalled that a traction force of 50 is imposed on the right edge of the domain. Figure 26 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.



(a) G+Smo Solution (b) Matlab Solution (c) IgANet Solution

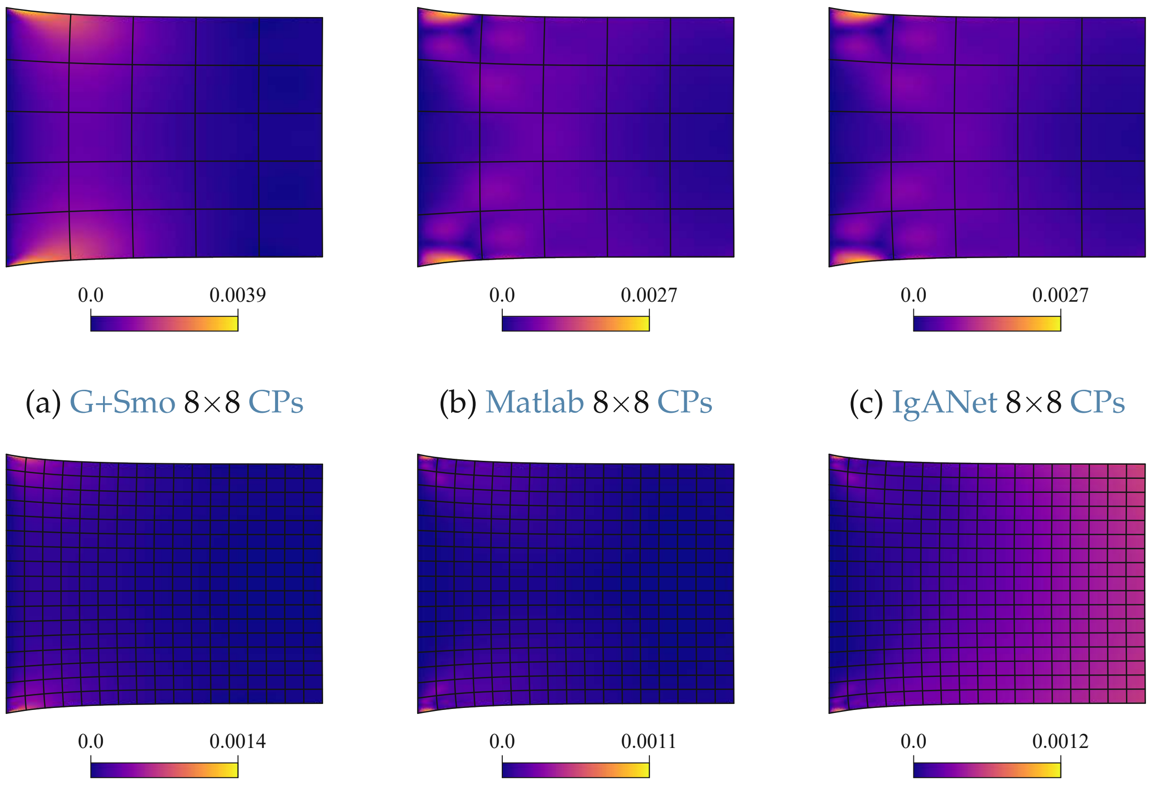
(d) G+Smo Stress (e) Matlab Stress (f) IgANet Stress

Figure 26: Comparison of the deformation results obtained by G+Smo, Matlab, and IgANet (a,b,c), supplemented by their respective stress distribution (d,e,f). Set-up: 8×8 CPs and degree-3.

The stress plots in Figure 26 (d,e,f) clearly demonstrate that visually, the traction force BC on the right edge is fulfilled. Furthermore, the comparison reveals that the results of the Matlab and IgANet simulation are identical in both displacement and stress distribution for the 8×8 CPs configuration.

Absolute Error Distribution

As in Chapter 4.1 with the Dirichlet experiment, a G+Smo-based reference solution is also used in this case. 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 G+Smo, Matlab, and IgANet 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 from 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 27. 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.



(a) G+Smo 8×8 CPs (b) Matlab 8×8 CPs (c) IgANet 8×8 CPs

(d) G+Smo 20×20 CPs (e) Matlab 20×20 CPs (f) IgANet 20×20 CPs

Figure 27: Comparison of the absolute error obtained by G+Smo, Matlab, and IgANet against the reference solution for spline degree-3 with 8×8 CPs (a,b,c) and 20×20 CPs (d,e,f).

The error plots in Figure 27 reveal that, for both 8 and 20 CPs per direction, the collocation-based solutions are more accurate than the Galerkin solution computed by G+Smo – a rather surprising result considering that the reference itself is based on a Galerkin formulation. 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.

Convergence Study

To further assess the performance comparison between the Galerkin and collocation methods, the absolute error analysis is extended by investigating their convergence behavior. For this purpose, reference solutions based on both methods are introduced. Each reference is computed on a 100×100 control point grid using spline degree-3.

Subsequently, simulations with varying control point numbers are carried out, and the MSE to the respective reference solution is computed. These error values are plotted on a log-log scale to analyze the convergence properties. As discussed in Chapter 4.1.7, for the Galerkin method, an error behavior of $\mathcal{O}(h^{p+1})$ is expected, which corresponds to a slope of 4 in this case. In contrast, the collocation method for odd spline degrees is typically expected to converge with $\mathcal{O}(h^{p-1})$, yielding a slope of 2 here. The results are shown in Figure 28, where reference lines with slope 4 are included for comparison purposes.

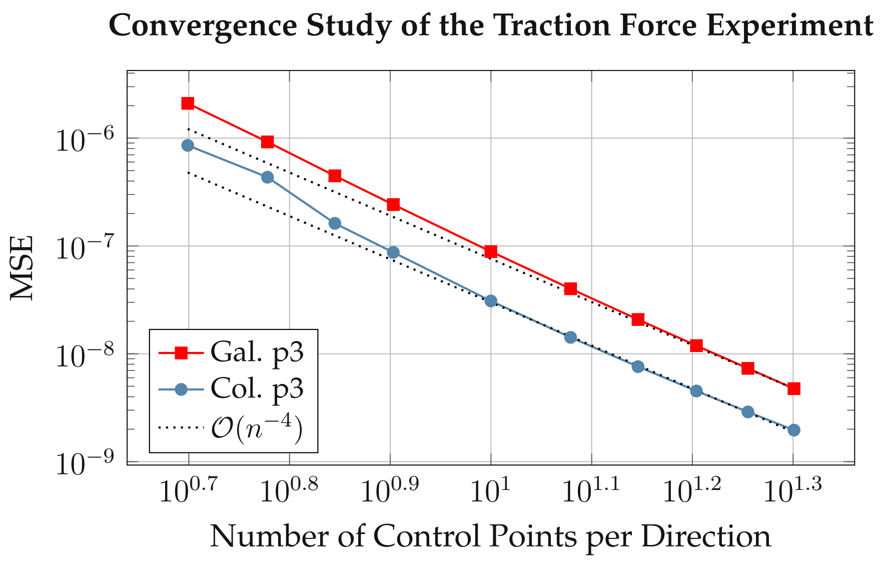


Figure 28: Comparison of the convergence rate between Galerkin vs. Collocation at 5×5 to 20×20 control point grids and spline degree-3.

The plots in Figure 28 reveal notable convergence behavior. As expected, the G+Smo-based Galerkin solution stabilizes at a convergence rate of 4, which aligns with theoretical predictions. Interestingly, the Matlab and IgANet collocation solutions also show a convergence rate close to 4, outperforming the expected rate of 2 for odd-degree splines. This better-than-expected performance of the collocation method may be attributed to the simplicity and regularity of the problem set-up, which could lead to more favorable numerical behavior than in more complex scenarios. In more complex or less symmetric set-ups, the expected convergence rate of 2 might be observed more clearly.

4.2.4 Simulation Results enhanced by Supervised Learning

In the context of the current traction force experiment, supervised learning has been introduced to support the training of the IgANet. As discussed in Chapter 3.2.2, the goal of supervised learning is to guide the network by providing a known target solution during training. In this case, a Matlab solution with similar 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 MSE between the current IgANet control points and those of the Matlab solution. This term is weighted with a factor bigger than 1 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 supervised loss is added alongside the standard components of the traction force experiment: the residuals of the elasticity equations, the traction-free BCs, the traction force BC and the Dirichlet BC.

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 29.

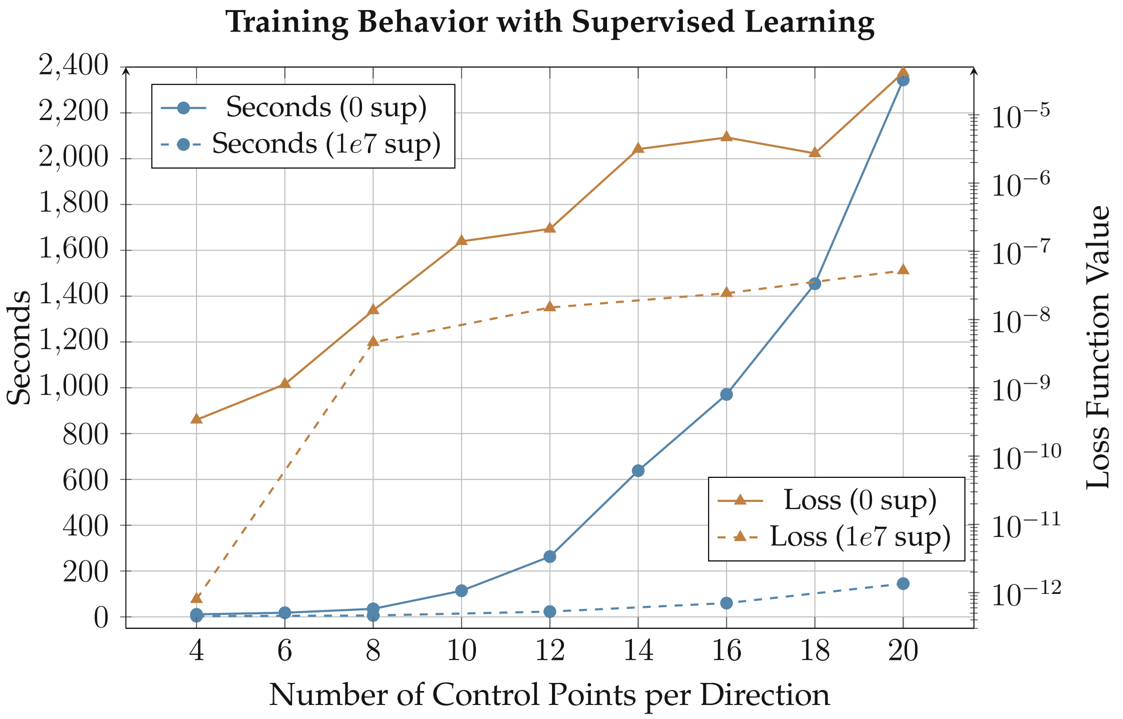


Figure 29: Comparison of the supervised and unsupervised training behavior of IgANet for different control point configurations. The figure illustrates the required training time needed to reach a stationary solution. Additionally, the values of the loss function at these stationary solutions are compared.

Figure 29 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.

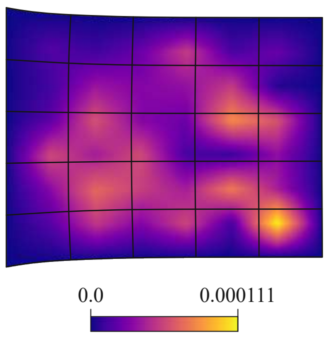
The use of a relatively high supervised learning weight of 10^7 is what enables this improvement. If the Dirichlet BC weight is simultaneously reduced (e.g., to 1) and the supervision weight increased further to 10^9, the training time drops drastically to just 24 seconds for the 20×20 CPs case. Compared 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 Matlab 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 Matlab 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 Matlab solution is currently negligible, as it can be computed in seconds, even for higher control point resolutions. The conclusion to the conducted investigations appears obvious. Offering a readily available collocation-based reference during training substantially boosts the IgANet's performance in terms of both computational effort and solution quality.

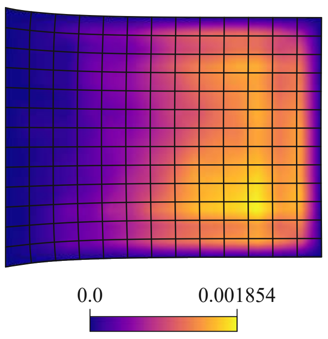
Elasticity Error Distribution

To further evaluate the quality of the IgANet solution in both supervised and unsupervised settings, its ability to satisfy the governing equations is analyzed. Recalling the equilibrium condition $\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}$, any deviation from zero in the divergence of the stress field can be interpreted as an error – previously referred to as the *Elasticity Error*.

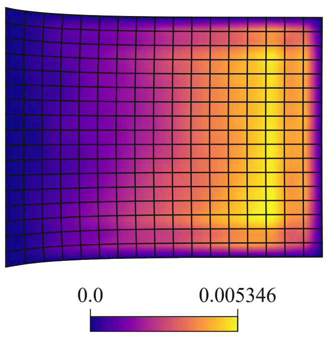
To visualize this error, the divergence of the stress tensor is computed at each collocation point. This reveals where and to what extent the IgANet struggles to fulfill the equilibrium condition. The results are gathered and compared for both the supervised and unsupervised simulations, providing insight into how supervision affects the model's ability to satisfy the underlying physics. The corresponding results are shown in Figure 30 and Figure 31.



(a) IgANet 8×8 CPs

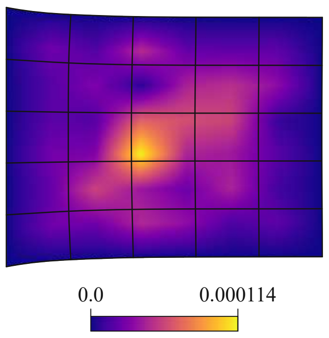


(b) IgANet 16×16 CPs

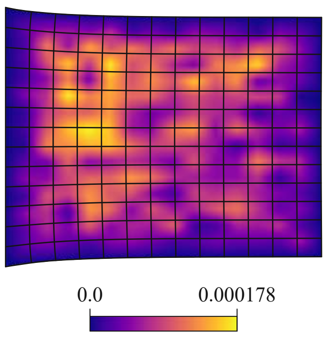


(c) IgANet 20×20 CPs

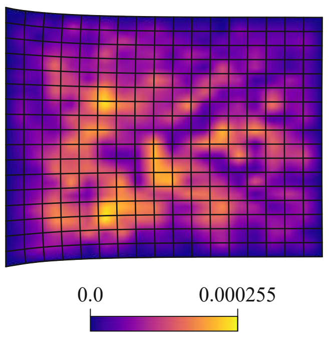
Figure 30: Elasticity error of the **unsupervised** IgANet solution for spline degree-3 at different control point configurations.



(a) IgANet 8×8 CPs



(b) IgANet 16×16 CPs



(c) IgANet 20×20 CPs

Figure 31: Elasticity error of the **supervised** IgANet solution for spline degree-3 at different control point configurations (supervised learning weight of 10^7).

The unsupervised results in Figure 30 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 Figure 31 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 toward the Matlab 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.

4.3 Parametrization Applications

This chapter presents the results of applying the IgANet framework to solve parametrized problems, where the simulation domain is modified after training. As outlined in Chapter 3.3, the goal is to evaluate whether a network trained on a specific geometry can generalize to slightly altered shapes without requiring retraining. For this purpose, the original spline geometry is changed by modifying selected control points, while the number of control points and the overall parametrization remain unchanged.

The already trained IgANet is then used to predict the new solution based on the modified geometry. This set-up serves as a first proof of concept for the idea of IgANet as a *Deep Operator Learning* tool and showcases the potential of the method for fast evaluations in design and optimization tasks. The following sections present selected examples of such predictions and analyze their performance in terms of plausibility and physical consistency.

4.3.1 Dirichlet Wave

The first experiment in the context of parametrization applications is based on the Dirichlet experiment, described in Chapter 4.1. Initially, the IgANet is trained on a unit square domain using 8×8 CPs and spline degree-4. After the training process, the geometry is altered by applying a sinusoidal perturbation in x-direction, given by

$$x_{\text{new}} = x_{\text{old}} + 0.05 \cdot \sin(0.1 \cdot CP_{row}), \tag{29}$$

where CP_{row} denotes the horizontal control point rows of the geometry, starting from zero at the bottom. The original and the parametrized geometries are illustrated in Figure 32.

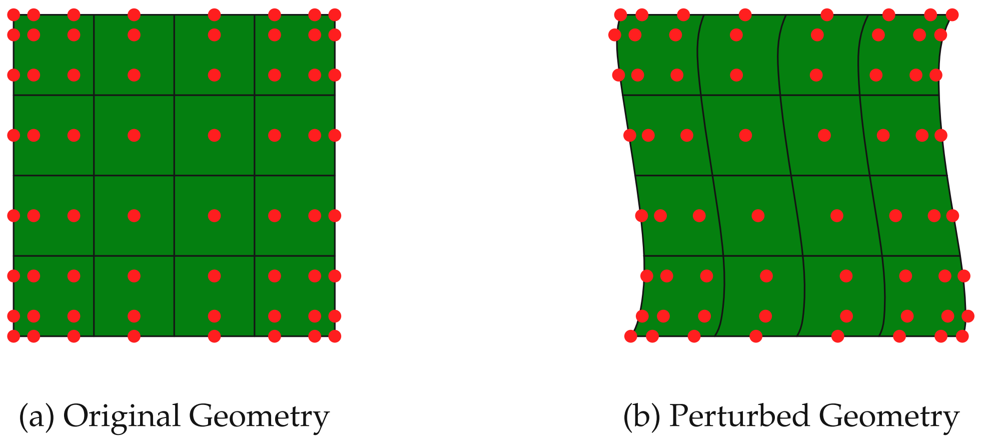
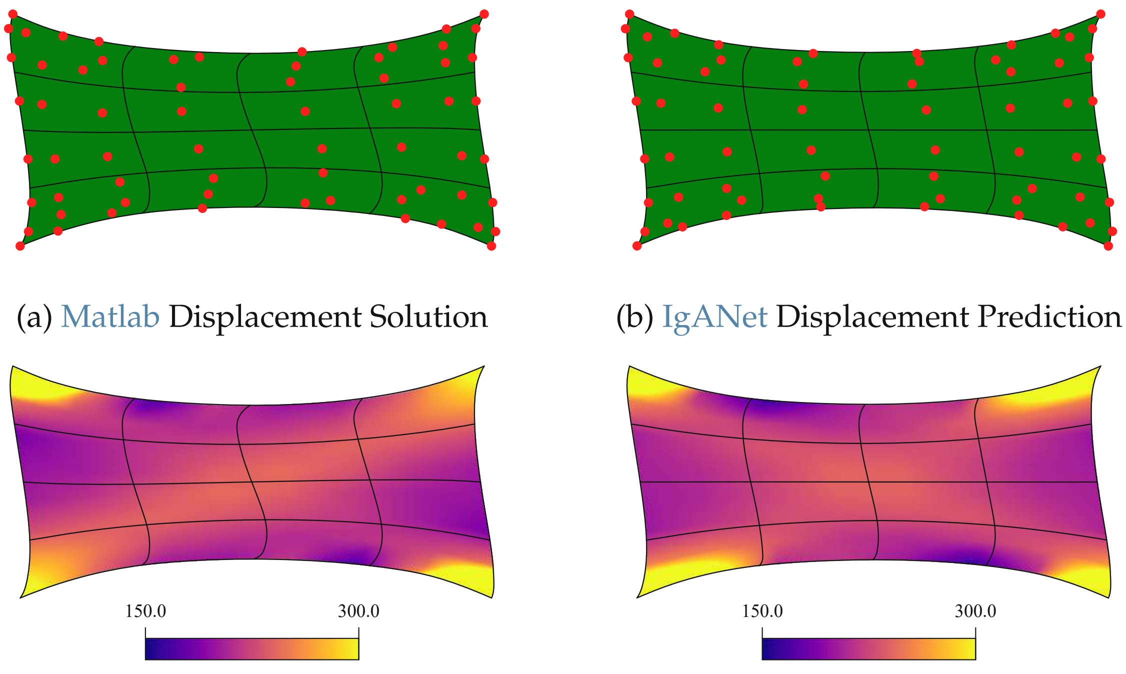


Figure 32: The original training geometry (a) and the modified geometry after applying a sinusoidal perturbation (b).

On this modified geometry, the trained IgANet is used to predict the solution without retraining. The boundary conditions are kept identical to the original Dirichlet setting: a fixed displacement on the left boundary and a prescribed unit displacement in x-direction on the right boundary. To provide a meaningful reference, a new collocation solution with equal spline configuration was computed using the Matlab code on the parametrized geometry. A comparison between the displacement and stress fields of the Matlab and IgANet solutions is presented in Figure 33.



(a) Matlab Displacement Solution (b) IgANet Displacement Prediction

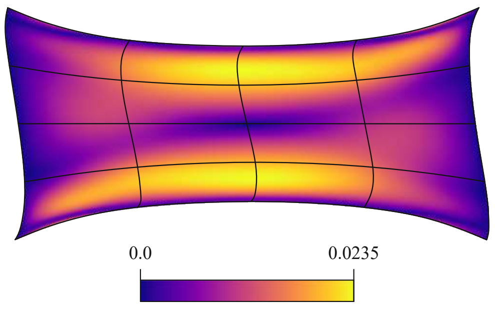
(c) Matlab Stress Solution (d) IgANet Stress Prediction

Figure 33: Comparison of IgANet prediction and Matlab collocation solution for the parametrized wavy geometry.

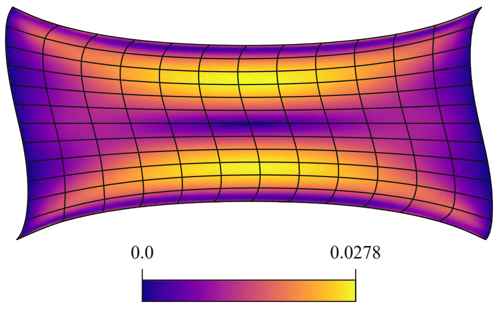
The comparison in Figure 33 shows that the predicted displacement field of the IgANet aligns well with the Matlab solution. The global deformation behavior and boundary conditions are captured correctly, indicating that the network can generalize to slightly altered geometries.

Closer inspection, however, reveals deviations in the stress distribution. Notable differences appear in the top right and bottom left corners, where the predicted stress values are overestimated compared to the Matlab reference. These local inaccuracies suggest that geometric perturbations, even if small, can significantly affect the internal stress state.

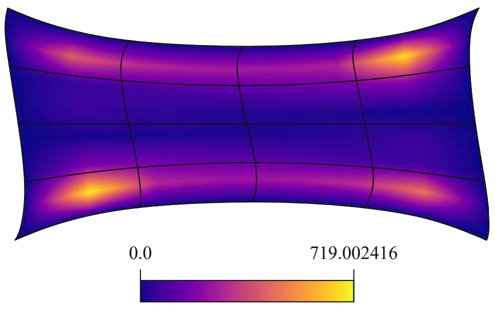
To quantify these deviations more precisely, Figure 34 presents a detailed error analysis based on absolute and elasticity error metrics.



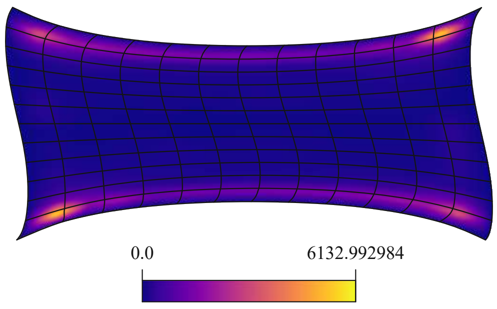
(a) Absolute Error at 8×8 CPs



(b) Absolute Error at 16×16 CPs



(c) Elasticity Error at 8×8 CPs



(d) Elasticity Error at 16×16 CPs

Figure 34: Error analysis of the IgANet prediction against the Matlab solution for the parametrized wavy geometry at 8×8 and 16×16 CPs.

The results in Figure 34 provide detailed insight into the performance of the IgANet prediction under geometric perturbation. The absolute error plots in subfigures (a) and (b) show that the deviation between the predicted and reference displacements is primarily concentrated within the interior of the domain and remains at a moderate value. In contrast, the boundary regions exhibit relatively low error levels, suggesting that the network has learned to preserve the prescribed boundary conditions even after modifying the geometry. The total absolute error does not reduce when increasing the number of control points from 8×8 to 16×16. Instead, the maximum error increases slightly. This indicates that refining the resolution does not automatically improve accuracy against the standard collocation Matlab solution.

The elasticity error, addressed in subfigures (c) and (d), reveals a different behavior. The error magnitude provides extremely high values over the whole domain – not only in peaks, as could be assumed from Figure 34 (c) and (d). The error value yet increases drastically for the 16×16 case. These high residuals – especially at the corners and along curved interior regions – outline the network's struggles to fulfill the equilibrium equations when faced with altered geometries.

Altogether, the results highlight that while the network is capable of preserving shape and boundary behavior to a certain extent, the physical validity of the predicted solutions is significantly challenged by geometric domain modifications.

4.3.2 Traction Force Rectangle

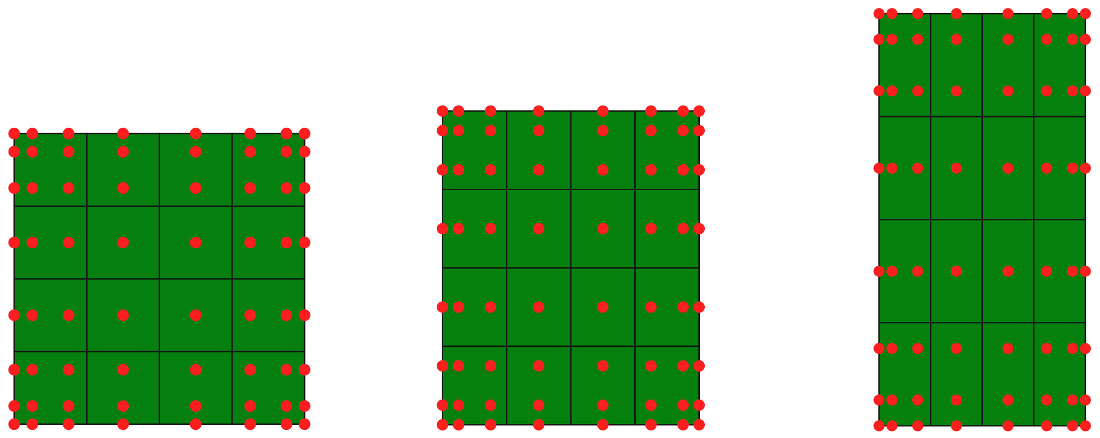
In the second parametrization study, the effect of the magnitude of geometric modifications on the prediction quality of the IgANet is examined. The goal is to determine whether larger changes in the domain geometry lead to a greater degradation in the accuracy of the predicted solution. In addition, it is demonstrated that the number of geometry modifications applied after training has no influence on the prediction result.

Therefore, two geometric modifications are applied sequentially: one small and one more substantial. After each modification, the trained network is used to predict the resulting displacement field. The test scenario is again based on the traction force experiment introduced in Chapter 4.2, where the left boundary of the domain is fixed with a zero-displacement Dirichlet condition, and a traction force of magnitude 50 is applied to the right edge.

As in the forerun Dirichlet wave parametrization, the IgANet is trained on a unit square with 8×8 CPs and spline degree-4. After training, two sequential modifications are applied to the domain:

- First parametrization: scaling the height by a factor of 1.1 and the width by a factor of 0.9.
- Second parametrization: scaling the height by a factor of 1.5 and the width by a factor of 0.75.

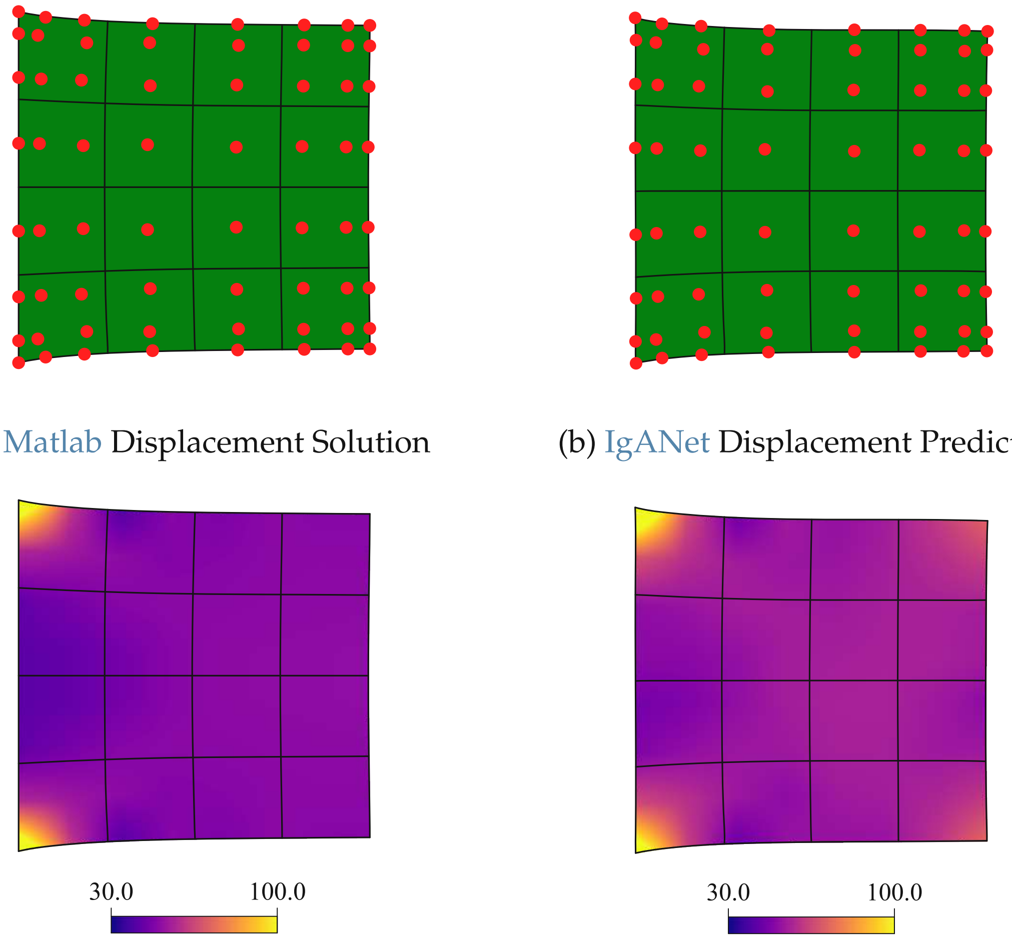
Both scaling steps are applied with respect to the original reference geometry. This means that the second parametrization does not build on the first, but is directly related to the original configuration. The original and modified geometries are shown in Figure 35.



(a) Original Geometry (b) Small Perturbation (c) Large Perturbation

Figure 35: Original geometry (a) and two subsequent geometry modifications by scaling height and width by 1.1×0.9 (b), and 1.5×0.75 (c).

The results for the first modification step are presented in Figure 36. The displacement and stress fields predicted by IgANet are compared to a reference solution computed with the Matlab collocation code at equal spline configurations.



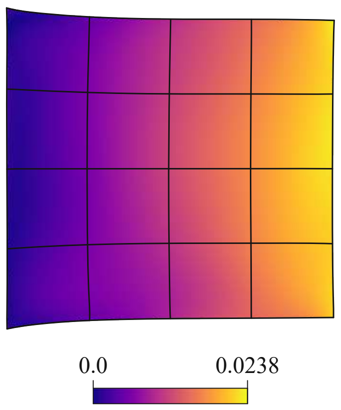
(a) Matlab Displacement Solution (b) IgANet Displacement Prediction

(c) Matlab Stress Solution (d) IgANet Stress Prediction

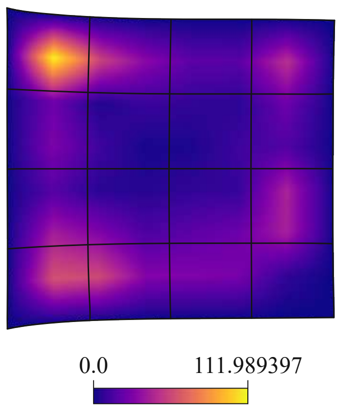
Figure 36: Parametrization results for the first rectangular modification. Comparison of the IgANet prediction against the Matlab solution.

The results in Figure 36 indicate that IgANet maintains its ability to generalize to modified geometries. The displacement field remains in good agreement with the reference solution. However, a more detailed look at the stress distribution reveals visible differences between prediction and reference. While the global stress pattern is captured reasonably well, local deviations become apparent, especially in the top right and bottom right corners of the domain. These discrepancies suggest that geometric stretching influences the internal force equilibrium in a way that challenges the network's ability to maintain physically consistent stress predictions.

A more detailed analysis of the errors is provided in Figure 37, where the absolute error and the elasticity error are plotted.



(a) Absolute Error of IgANet

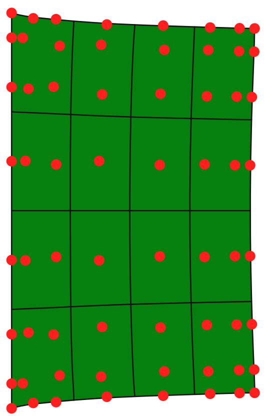


(b) Elasticity Error of IgANet

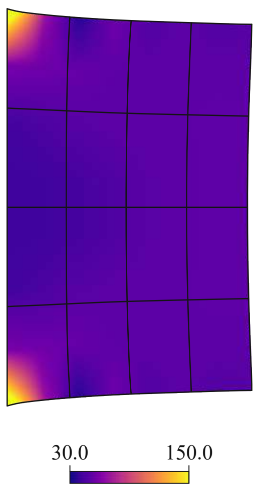
Figure 37: Absolute error and elasticity error of the IgANet solution after the **first** parametrization step.

The absolute error of Figure 37 (a) is mainly concentrated at the right boundary of the domain, where the geometric change appears to cause the largest effect. The elasticity error in subplot (b) once more reveals very high residual values, including a rectangular-shaped region with significantly increased errors in the interior of the domain.

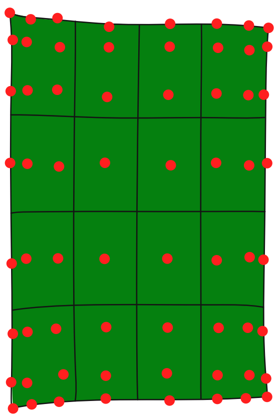
Following this first modification, the second parametrization step is performed without retraining the network. In the second parametrization, the height is scaled by a factor of 1.5 and the width by a factor of 0.75. The resulting displacement and stress fields for both the Matlab solution and IgANet prediction are shown in Figure 38.



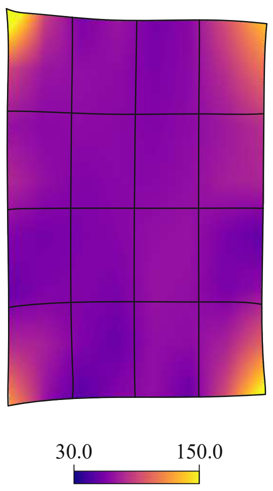
(a) Matlab Displacement Solution



(b) Matlab Stress Solution



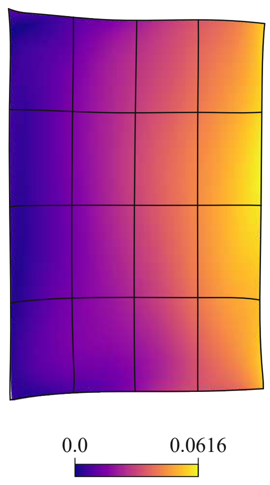
(c) IgANet Displacement Prediction



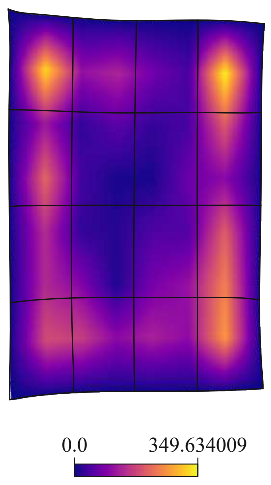
(d) IgANet Stress Prediction

Figure 38: Parametrization results for the second rectangular modification.

At this stage, the deviation between IgANet prediction and Matlab solution becomes clearly visible. In Figure 38, both the displacement and stress distributions show significant differences, with notable stress peaks occurring at the right corners of the domain in the IgANet prediction. The corresponding absolute and elasticity error plots are presented in Figure 39.



(a) Absolute Error of IgANet



(b) Elasticity Error of IgANet

Figure 39: Absolute and elasticity error of the IgANet solution after the second parametrization step.

As illustrated in Figure 39, both the absolute and elasticity errors increase significantly compared to the previous parametrization step. The absolute error field shows a substantial rise in magnitude, especially near the right-hand side of the domain, indicating that the prediction no longer aligns well with the reference solution in this region. Unlike earlier examples, even the previously well preserved boundaries now exhibit noticeable discrepancies. Similarly, the elasticity error increases drastically, indicating once more that the network struggles to maintain equilibrium of the governing PDE in a deformed geometry.

It can be concluded that the performance of the IgANet degrades with larger deviations from the original training geometry. However, the number of consecutive parametrizations itself has no impact. The second evaluation again refers to the training data gathered onto the original geometry and thus, behaves similarly whether the geometry is changed in one or multiple steps. In this context, only the overall magnitude of the geometric modification determines the prediction quality.

4.3.3 Takeaways from the Parametrization Applications

The results presented in this chapter demonstrate that the IgANet framework can already generalize reasonably well to moderately altered geometries without the need for retraining. In the current study, the network was trained on a single representative geometry and was still able to predict the displacement field for similar, slightly modified domains with acceptable accuracy, particularly near the boundaries.

However, as the geometric deviation from the training configuration increases, both the absolute and elasticity errors rise significantly, clearly indicating the limits of the current network's generalization capability. This behavior is not unexpected, as training on a single configuration is unlikely to offer sufficient information to generalize to arbitrarily altered domains.

Nevertheless, these first experiments underline the potential of IgANet for efficient, real-time design evaluations in parametrized simulation tasks. Future research should explore training on a broader set of representative geometries to improve generalization performance across a wider design space.

5 Conclusion and Outlook

This chapter wraps up the thesis by summarizing the main results, discussing current limitations of the approach, and outlining ideas for future work. The goal is to reflect on what has been achieved so far and to point out where the method could be improved or extended to cover a broader range of applications in computational solid mechanics.

5.1 Summary

This thesis has explored the application of the isogeometric collocation method in combination with neural networks for solving problems in computational solid mechanics. The work relied on the IgANet framework, which was used not only as a solver for partial differential equations, but more broadly as a tool for *Deep Operator Learning* – mapping input data, such as geometry and boundary conditions, directly to solution representations.

The theoretical foundation was built upon classical linear elasticity, formulated under simplifying assumptions such as small strains, isotropy, and isothermal conditions. The numerical formulation was based on the isogeometric collocation method, where their smoothness makes B-Splines the perfect choice for the mentioned method.

Building on this foundation, the IgANet framework was extended to simulate linear elastic problems in two dimensions. The architecture supports standard elasticity equations, Dirichlet and Neumann boundary conditions, as well as optional supervision during training. A modular input structure was developed to flexibly configure test cases and streamline usability.

To evaluate the framework's performance, two benchmark scenarios were investigated on a unit square domain – one using prescribed Dirichlet displacements and one using a Dirichlet and a traction boundary condition combined. The predicted displacement and stress fields were compared against reference solutions from classical collocation and Galerkin methods. The analysis showed that IgANet can replicate classical collocation solutions with high accuracy. The effect of supervised learning was examined in terms of training efficiency and solution quality, highlighting significant reductions in training time and improvements in solution accuracy. Convergence studies were conducted for both test cases and confirmed that increasing resolution improves the simulations' accuracy. The simplicity of the experiment set-ups even caused convergence rates better than theoretically expected.

The final and central contribution of this work was the investigation of geometry parametrization. After training on a reference configuration, the geometry was modified by adjusting control points, and the trained IgANet was used to predict new solutions without retraining. Despite being trained on a single geometry only, the framework was able to produce reasonably accurate predictions for similar configurations, highlighting its generalization potential even in this minimal training setting.

In summary, this thesis presents a holistic approach. From theory and implementation to application and evaluation, the work demonstrates how IgANet can be used as a neural operator for solving linear elasticity problems and for predicting parametrized simulations by deep operator learning.

5.2 Limitations

While the presented framework shows promising results, a number of limitations must be acknowledged that currently restrict its general applicability.

A fundamental restriction of the current implementation is the focus on simple linear elastic material behavior. More complex constitutive laws, such as plasticity or nonlinear elasticity, are not yet incorporated. The formulation is thus limited to small strain, linear regimes, which are not always sufficient for engineering applications involving large deformations or material nonlinearities.

Although the framework is technically prepared for three-dimensional problems, the current implementation is limited in flexibility. In particular, traction-free boundary conditions are not yet fully supported in 3D, and the code structure still requires significant manual adjustments. As a result, only the two-dimensional case was considered in this thesis, although an extension is foreseeable.

The simulations rely solely on standard isogeometric collocation, where the number of collocation points equals the number of control points. Although this method offers advantages in computational efficiency, it is known to be sensitive to the placement of collocation points and may struggle in terms of accuracy in certain simulation settings compared to Galerkin-based simulations. Alternative formulations such as least-squares collocation, which would conceptually align well with the optimization-based nature of neural networks, were also tested. However, initial attempts using this method led to results with limited physical validity in the present environment. This means that so far, only the standard collocation formulation yielded reliable outcomes. Additionally, apart from the (least-squares) collocation method, the underlying IgANet framework has already taken first steps towards Galerkin formulations, offering a promising direction for future work [3].

Regarding training behavior, the IgANet framework shows increasing difficulty with higher control point resolutions. While accuracy generally improves with more control points, the training time rises sharply, and the network may struggle to minimize the loss effectively. This is particularly evident in the enforcement of Dirichlet boundary conditions, which require a very high weighting factor within the loss function to be satisfied properly. This difficulty probably stems from the underlying structure of the network, which does not enforce boundary values strongly by default.

The current implementation is further limited by the use of a unit square as training domain. While sufficient for demonstration purposes, real-world geometries are typically more complex and would require either an adapted training strategy or significantly more training time to resolve all physical constraints. For such configurations, the time required to satisfy the governing equations during training could quickly become impractical.

The parametrization capability is also restricted in several ways. It is currently only possible to modify the geometry after training, while other model parameters such as boundary conditions or material properties remain fixed. Moreover, the generalization performance decreases with larger geometric changes. Small perturbations to the control point layout can be predicted well, but larger deviations lead to loss of physical accuracy. In both cases, some non-physical behavior was observed within the domain's interior, as the governing equations were not fulfilled anymore in the prediction scenarios. It is important to note that these limitations are expected, given that the network was trained on a single reference geometry. Training on a more diverse set of geometries would likely improve the generalization performance and reduce non-physical artifacts in the predictions.

Further, the supervised learning option improves both accuracy and training time, but requires access to a reliable reference solution, such as a high-quality collocation result. In practical applications, however, such reference data may not always be available, which limits the autonomy and scalability of the method.

In summary, the training process remains the primary computational bottleneck. For complex geometries or highly refined configurations, the time required to solve the physical equations through gradient-based optimization can become very large, particularly in the absence of supervision.

5.3 Outlook

The results of this work demonstrate the feasibility of using the IgANet framework to solve parametrized linear elasticity problems. However, the current implementation serves primarily as a proof of concept, and there is significant potential for extension and refinement.

An important direction for future development lies in the extension of the framework to three-dimensional applications. The underlying code already supports 3D functionality in principle, but further adjustments are required to ensure full flexibility, especially regarding boundary condition definitions and the handling of traction-free surfaces.

Beyond geometric dimensionality, the training set-up itself could be improved. In the current implementation, all simulations are performed on single-patch unit square domains with equal spline resolution in both parametric directions. To increase the robustness and generalization ability of the network, future work should focus on training across a broader variety of geometries within a single training cycle. This includes domains of arbitrary size and shape, with varying numbers of control points or spline degrees per spatial direction. Such an extension is expected to improve prediction quality, particularly for previously unseen configurations. Additionally, first steps are already being taken to adapt the IgANet framework to multi-patch geometries. While multi-patch support is already implemented at the core level, the extension to solid mechanics problems remains an ongoing task.

Another important direction is the inclusion of more advanced material models. Thus far, the implementation has been limited to small-strain linear elasticity. However, ongoing work by a follow-up thesis is addressing the incorporation of nonlinear material behavior and constitutive modeling. Such enhancements will substantially broaden the scope of problems that IgANet can handle and mark an essential step toward the application of the method to complex real-world simulations.

From a technical perspective, code performance remains an important area of improvement. While training times are acceptable for small configurations, they quickly become prohibitive for larger problems. Here, potential optimizations include more efficient loss computation, adaptive collocation point strategies, graphics processing unit (GPU) acceleration, or even alternative optimization techniques. Additionally, the current user interface is already modular and accessible, but future versions could integrate a graphical user interface (GUI) to further improve usability. This would make the framework more approachable to users without a programming background and allow easier configuration of simulation parameters.

Looking further ahead, the overarching goal of IgANet in the context of computational solid mechanics is to enable efficient simulation and design workflows. In many engineering scenarios, such as design space exploration, numerous similar simulations are required under slightly varying conditions. Instead of solving each configuration from scratch, a trained IgANet could provide instant predictions for modified geometries or parameters, dramatically reducing computation times. This would allow the method to be integrated into iterative design processes, where rapid feedback is essential.

In summary, while the current implementation still has limitations, the IgANet framework provides a solid foundation for future developments. Although the present results are based on training with a single geometry, the framework already demonstrates a notable capacity to generalize to moderately altered configurations. Expanding the training to multiple geometries is expected to further improve prediction quality. With its physics-informed nature and adaptability, IgANet shows strong potential for next-generation simulation and design workflows.

References

[1] C. Anitescu, Y. Jia, J. Y. Zhang, and T. Rabczuk. An isogeometric Collocation Method using superconvergent Points. *Computer Methods in Applied Mechanics and Engineering*, 284:1073–1097, 2015.

[2] F. Auricchio, L. Beirao da Veiga, T. J. R. Hughes, A. Reali, and G. Sangalli. Isogeometric Collocation Methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107, 2010. doi: 10.1142/S0218202510004878.

[3] M. Backmeyer, S. Kurz, M. Möller, and S. Schöps. Solving electromagnetic Scattering Problems by Isogeometric Analysis with Deep Operator Learning. In *2024 Kleinheubach Conference*, pages 1–4. IEEE, 2024.

[4] Y. Bazilevs, L. Beirao da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli. Isogeometric Analysis: Approximation, Stability and Error Estimates for h-refined Meshes. *Mathematical Models and Methods in Applied Sciences*, 16(07):1031–1090, 2006.

[5] S. Berrone, C. Canuto, M. Pintore, and N. Sukumar. Enforcing Dirichlet Boundary Conditions in Physics-informed Neural Networks and Variational Physics-informed Neural Networks. *Heliyon*, 9(8), 2023.

[6] C. D. Coman. *Continuum Mechanics and linear Elasticity*. Springer, 2020.

[7] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, 2009.

[8] P. Cunningham, M. Cord, and S. J. Delany. Supervised Learning. In *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, pages 21–49. Springer, 2008.

[9] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific Machine Learning through Physics-informed Neural Networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[10] ECMA International. JSON (Javascript Object Notation). `https://www.json.org`, 2017. Standard ECMA-404.

[11] G. Farin. *Curves and Surfaces for Computer-aided geometric Design: A practical Guide*. Elsevier, 2014.

[12] L. Franceschi, M. Donini, V. Perrone, A. Klein, C. Archambeau, M. Seeger, M. Pontil, and P. Frasconi. Hyperparameter Optimization in Machine Learning. 2024.

[13] H. Gomez and L. De Lorenzis. The Variational Collocation Method. *Computer Methods in Applied Mechanics and Engineering*, 309:152–181, 2016. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2016.06.003.

[14] L. Gonon, A. Jentzen, B. Kuckuck, S. Liang, A. Riekert, and P. von Wurstemberger. An Overview on Machine Learning Methods for Partial Differential Equations: From Physics-informed Neural Networks to Deep Operator Learning. 2024.

[15] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric Analysis: Cad, Finite Elements, nurbs, exact Geometry and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005.

[16] F. Ihlenburg. Lineare Elastizität. In *Kontinuumsmechanik fester Körper: Mit mathematischen Grundlagen und Anwendungen in der Strukturmechanik*, pages 177–211. Springer, 2024.

[17] R. W. Johnson. Higher order B-Spline Collocation at the Greville Abscissae. *Applied Numerical Mathematics*, 52(1):63–75, 2005.

[18] B. Jüttler, U. Langer, A. Mantzaflaris, S. E. Moore, and W. Zulehner. Geometry+Simulation Modules: Implementing Isogeometric Analysis. *PAMM*, 14(1): 961–962, 2014.

[19] B. Klein. *FEM*. Springer, 2010.

[20] J. Lee, J. M. Zwar, and S. Elgeti. Splinepy: A Prototyping Toolkit for Spline-based Design and Simulation. Poster presentation at the 11th International Conference on Isogeometric Analysis (IGA 2023), June 2023. Lyon, France.

[21] J. Lee et al. splinepy. `https://github.com/tataratat/splinepy/`, 2023. Accessed: 2025-04-29.

[22] A. Mantzaflaris et al. G+smo – Geometry + Simulation Modules. `https://github.com/gismo/gismo`, 2015. Accessed: 2025-04-29.

[23] M. Möller, D. Toshniwal, and F. van Ruiten. Physics-informed Machine Learning embedded into Isogeometric Analysis. *Mathematics: Key enabling Technology for scientific Machine Learning. Platform Wiskunde, Amsterdam*, pages 57–59, 2021.

[24] M. Möller et al. Iganet – Physics-informed Neural Networks for Isogeometric Analysis. `https://github.com/IgANets/iganet`, 2022. Accessed: 2025-04-29.

[25] H. Moritz. Least-Squares Collocation. *Reviews of Geophysics*, 16(3):421–430, 1978.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[27] A. F. Psaros, K. Kawaguchi, and G. E. Karniadakis. Meta-Learning pinn Loss Functions. *Journal of Computational Physics*, 458:111121, 2022.

[28] J. Ren and H. Lin. A Survey on Isogeometric Collocation Methods with Applications. *Mathematics*, 11(2):469, 2023.

[29] X. Ren, R. Das, P. Tran, T. D. Ngo, and Y. M. Xie. Auxetic Metamaterials and Structures: A Review. *Smart Materials and Structures*, 27(2):023001, 2018.

[30] M. H. Sadd. *Elasticity: Theory, Applications, and Numerics*. Academic Press, 2009.

[31] S. Sharma, S. Sharma, and A. Athaiya. Activation Functions in Neural Networks. *Towards Data Science*, 6(12):310–316, 2017.

[32] C. Spura. *Technische Mechanik 2. Elastostatik*. Springer, 2019.

[33] B. Stroustrup. An Overview of the C++ Programming Language. *Handbook of Object Technology*, 72, 1999.

[34] The MathWorks, Inc. *MATLAB Version R2023b*. The MathWorks, Inc., Natick, Massachusetts, 2023. URL https://www.mathworks.com/products/matlab.html.

[35] Q. Wang, Y. Ma, K. Zhao, and Y. Tian. A comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science*, 9(2):187–212, 2022.

[36] G. Yagawa and A. Oishi. *Computational Mechanics with Neural Networks*. Springer, 2021.

[37] E. Zampieri and L. F. Pavarino. A numerical Comparison of Galerkin and Collocation isogeometric Approximations of acoustic Wave Problems. *Applied numerical Mathematics*, 200:453–465, 2024.

Note: Spelling and language editing were supported using ChatGPT by OpenAI.