IGA-ODIL: FAST PHYSICS-INFORMED LEARNING WITH B-SPLINE REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-informed neural networks (PINNs) formulate the solution of partial differential equations as residual minimization problems over neural network parameterizations. Although highly flexible, optimization of PINNs using modern variants of Stochastic Gradient Descent algorithms is expensive. On the other hand, iterative computation of PINN parameterization using the Gauss–Newton method suffers from convergence difficulties, dense Jacobian structures, and poor conditioning that limit the effectiveness of second-order optimization methods. In this work, we introduce IGA-ODIL (Optimization of DIscrete Loss with data representation following from IsoGeometric Analysis). Instead of neural-network parameterizations of PINNs, the unknown solution is represented by smooth B-spline basis functions, leading to structured, sparse Jacobians and efficient second-order Gauss–Newton optimization. For the nonlinear inverse problem, we use a damped Gauss–Newton (Levenberg–Marquardt) variant to improve robustness. In the considered implementations, the resulting sparse second-order optimization achieves two to three orders of magnitude lower wall-clock time than the corresponding first-order PINN and CRVPINN implementations. The resulting systems inherit locality, sparsity, and approximation-theoretic properties of classical finite element and isogeometric methods while preserving the residual-learning philosophy of scientific machine learning. The proposed methodology is evaluated on several benchmark problems, including Poisson equations, convection-dominated advection–diffusion equations, Helmholtz problems with highly oscillatory solutions, nonlinear Allen–Cahn equations, and inverse Helmholtz parameter identification. Numerical experiments demonstrate orders-of-magnitude speedups compared with PINNs and CRVPINNs while maintaining high accuracy and robustness.

1 INTRODUCTION

Scientific machine learning has recently emerged as a rapidly developing research direction at the intersection of numerical analysis, partial differential equations (PDEs), optimization, and machine learning. Among the most influential approaches are Physics-Informed Neural Networks (PINNs), introduced by Raissi et al. (2019). PINNs formulate the numerical solution of PDEs as residual minimization problems in which the unknown solution is represented by a neural network and the governing equations are enforced through collocation-based residual losses. Numerous extensions and variants have appeared, including Variational PINNs (VPINNs) Kharazmi et al. (2021), Deep Ritz methods E & Yu (2018), adaptive and domain-decomposition formulations, as well as operator-learning approaches such as Fourier Neural Operators Li et al. (2020). More general overviews of scientific machine learning and physics-informed learning frameworks can be found in Karniadakis et al. (2021). Recent applications show that PINNs are used particularly often in fluid mechanics, where they support flow reconstruction, inverse parameter identification, and data assimilation from sparse or noisy measurements Cai et al. (2021). They have been applied to wake flows, biomedical flows, supersonic flows, and experimental stratified flows, demonstrating their potential for combining measurement data with Navier–Stokes-type models Cai et al. (2021); Zhu et al. (2024). PINNs have been used for heat transfer in porous media, where they can estimate temperature fields, heat fluxes, and effective thermal conductivity without requiring dense labeled data Xu et al. (2023). They have also been applied to two-phase flow and heat transfer problems, where the physics-informed

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

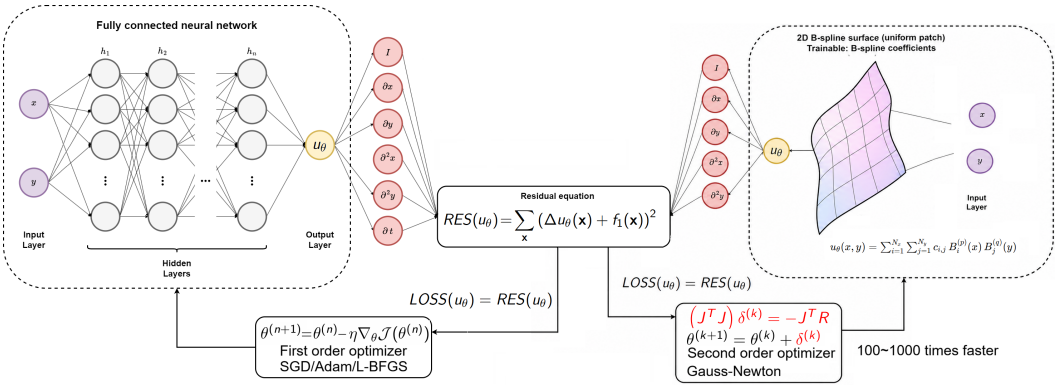

Figure 1: There are two methods described in this paper. The first reference method is the PINN method and it is illustrated as the workflow on the left-hand side of the diagram. The second method is the proposed IGA-ODIL method and it is illustrated on the right-hand side of the diagram.

loss helps enforce the governing equations while learning from simulation data Jalili et al. (2024). More recently, PINNs have been considered as components of digital twins, where the goal is to combine sensor data, reduced-order modeling, and physical constraints for real-time prediction and uncertainty-aware monitoring Yang et al. (2024).

Despite their flexibility, PINNs suffer from several important numerical and theoretical difficulties. Optimization landscapes associated with deep neural-network parameterizations are highly non-convex and frequently exhibit poor conditioning, flat directions, vanishing gradients, and parameter symmetries Wang et al. (2021). As a consequence, first-order optimization methods such as stochastic gradient descent (SGD) and ADAM (Kingma & Ba (2015)) often converge slowly and unreliably. Furthermore, second-order methods such as Newton or Gauss–Newton optimization are difficult to apply effectively because the resulting Jacobians are typically large, dense, and ill-conditioned Wang et al. (2021). There are some recent works developing second order optimizers for PINNs, Jnini & Vella (2025); Webb et al. (2026) that delivers several order-of-magnitude lower final error and faster training than ADAM/SGD optimizers. Our research goes into another direction, replacing NN representation by splines. Although spline-based KANs Patra et al. (2024); Wang et al. (2026; 2025) introduce local support at the level of individual edge functions, their multilayer fully connected architecture induces global parameter dependencies through composition. Consequently, the Jacobian with respect to network parameters generally lacks the element-local sparsity of a spline trial-space discretization such as IGA-ODIL.

An additional difficulty is that the classical PINN residual loss generally lacks robustness with respect to physically meaningful norms. In many cases, the residual loss does not accurately represent the discretization error measured in Sobolev norms, leading to poor correlation between optimization convergence and physical solution quality Rojas et al. (2024). This observation motivated the development of Robust Variational PINNs Rojas et al. (2024), based on the weak residuals of Variational PINNs (VPINNs) Kharazmi et al. (2021). Additionally, to speed up the loss estimation, Collocation-based Robust Variational Physics-Informed Neural Networks (CRVPINNs) Łoś et al. (2025) have been recently proposed. In particular, the robust loss takes the form

$$\Phi(u) = R(u)^T G^{-1} R(u), \tag{1}$$

where G is a Gram matrix associated with a suitable variational inner product Rojas et al. (2024), and $R(u)$ is a weak residual in the case of VPINNs or a discrete weak residual (computed with Kronecker product delta test functions) in the case of CRVPINNs. Although robust residual formulations improve the stability of PINN losses, the optimization challenges associated with deep nonlinear parameterizations remain. Neural-network Jacobians remain globally coupled and dense, limiting the scalability and effectiveness of second-order optimization methods.

A complementary research direction was recently proposed by Karnakov, Litvinov, and Koumoutsakos through the Optimizing DIscrete Loss (ODIL) framework Karnakov et al. (2024). The central idea of ODIL is that residual minimization does not fundamentally require neural-network parameterizations. Instead, the unknown solution is represented directly through discrete degrees of free-

dom optimized using PDE residual losses. ODIL demonstrated that many scientific machine learning ideas can be interpreted as optimization-based PDE discretizations rather than purely neural-network methods. The resulting formulation preserves many advantages of scientific machine learning, including natural extensions to inverse problems and PDE-constrained optimization.

However, point-based ODIL discretizations are not globally continuous like neural network representations. They also lack several important properties associated with classical spline and finite element approximation spaces. In particular, point-value parameterizations do not naturally provide high-order smoothness, compactly supported basis functions, and approximation-theoretic properties associated with spline spaces and classical Galerkin discretizations Quarteroni & Valli (1994).

Isogeometric Analysis, introduced by Hughes, Cottrell, and Bazilevs Hughes et al. (2005), establishes spline-based discretizations as powerful alternatives to classical finite element methods. B-splines and NURBS provide smooth, high-order approximation spaces with excellent approximation properties, sparse local support, and direct compatibility with CAD geometries Hughes et al. (2005); Cottrell et al. (2009); Bazilevs et al. (2006); Hughes et al. (2021); Buffa et al. (2014). Recent applications of isogeometric analysis (IGA) show that it is widely used in problems where smoothness, high-order accuracy, and exact geometry representation are important. In fracture mechanics, IGA has been applied to phase-field models of brittle, anisotropic, and polycrystalline fracture, where the higher continuity of spline spaces is advantageous for fourth-order and gradient-enhanced formulations Chen et al. (2020); Goswami et al. (2020); Nguyen-Thanh et al. (2020); Greco et al. (2024). Another active direction is design and topology optimization, where the common representation of geometry and analysis fields supports design-through-analysis workflows Gao et al. (2020). Recent developments also extend IGA to complex CAD geometries using analysis-suitable T-splines and G-spline surfaces Wei et al. (2022); Wen et al. (2023). In fluid mechanics and aerodynamics, IGA has been used for turbulent flows, moving domains, fluid–structure interaction, and space–time simulations of complex engineering configurations Bazilevs et al. (2023); Kuraishi et al. (2025). Applications in biomechanics include cardiac electrophysiology and ventricular mechanics, where smooth patient-specific geometries and coupled multiphysics models are essential Torre et al. (2023); Willems et al. (2024).

In this paper, we investigate whether first-order SGD/ADAM optimization can be replaced by an iterative second-order Gauss–Newton method. We observe that the Jacobian associated with neural-network parameterizations is generally dense and ill-conditioned, which can make second-order methods such as Gauss–Newton difficult to apply effectively. Thus, we propose a unified framework IGA-ODIL. The proposed methodology replaces neural-network parameterizations with smooth B-spline approximation spaces while preserving the residual-minimization philosophy of scientific machine learning. For linear PDEs, the B-spline residual formulation reduces Gauss–Newton optimization to a sparse least-squares solve Lin et al. (2020); Aurichio et al. (2010), while for nonlinear and inverse problems it enables efficient Gauss–Newton iterations through the locality of the spline Jacobian. In our experimental setting, this results in wall-clock speedups of up to two–three orders of magnitude relative to the considered first-order PINN and CRVPINN implementations. IGA-ODIL is not proposed as a new least-squares spline collocation discretization for linear PDEs. Rather, it provides a residual-learning optimization framework in which classical spline collocation emerges as the linear special case, while the same formulation extends naturally to nonlinear PDEs and joint state–parameter inverse problems. The methodology is evaluated on several benchmark problems, including Poisson equations, the Eriksson-Johnson model problem, high-frequency Helmholtz problems, nonlinear Allen–Cahn equations, and PDE-constrained inverse problems. The numerical experiments demonstrate substantial computational advantages compared with PINNs and CRVPINNs, including orders-of-magnitude speedups while maintaining high approximation quality and robustness.

2 PINNs and Robust Residual Minimization

Consider a PDE operator $\mathcal{L}u = f$ in a domain Ω. PINNs approximate the solution using a neural network, $u_\theta(x, y) = \mathrm{NN}(x, y) = \sigma(A_n\sigma(\cdots\sigma(A_1\begin{bmatrix}x\\y\end{bmatrix} + \beta_1)\cdots) + \beta_n)$ where A_i are layer weights, σ are nonlinear activation functions, and β_i are biases. In general, $\theta \in \mathbb{R}^P$ denotes the trainable parameters Raissi et al. (2019). The PINN method is summarized in Figure 1. The residual

evaluated at collocation points $\{\mathbf{x}_k\}_{k=1}^M$ is $R_k(\theta) = \mathcal{L}u_\theta(\mathbf{x}_k) - f(\mathbf{x}_k)$. The classical PINN loss $\mathcal{J}(\theta) = \frac{1}{2}\sum_{k=1}^M |R_k(\theta)|^2$, can be interpreted as a nonlinear least-squares optimization problem over neural-network parameterizations E & Yu (2018); Nocedal & Wright (2006). The classical approach to solving PINNs is to use stochastic gradient descent or ADAM (Kingma & Ba (2015)), which replaces the local gradient with the weighted average gradient from several previous iterations, is one of the most popular choices. The method is summarized in Algorithm 1.

Algorithm 1 Physics-Informed Neural Network (PINN)

1: Construct neural network approximation $u_\theta(\mathbf{x})$
2: Select collocation points $\{\mathbf{x}_k\}_{k=1}^M$
3: Initialize network parameters $\theta^{(0)}$
4: **while** not converged **do**
5: Evaluate residuals at collocation points $R_k(\theta^{(n)}) = \mathcal{L}u_{\theta^{(n)}}(\mathbf{x}_k) - f(\mathbf{x}_k), k = 1, \ldots, M$
6: Form residual vector $R(\theta^{(n)}) = \left[R_1(\theta^{(n)}), \ldots, R_M(\theta^{(n)})\right]^T$
7: Compute PINN loss $\mathcal{J}(\theta^{(n)}) = \frac{1}{2}\sum_{k=1}^M |R_k(\theta^{(n)})|^2 + \mathcal{L}_{BC}$
8: Compute gradients $\nabla_\theta \mathcal{J}(\theta^{(n)})$ using backpropagation and automatic differentiation
9: Update neural-network parameters $\theta^{(n+1)} = \theta^{(n)} - \eta\nabla_\theta \mathcal{J}(\theta^{(n)})$ using ADAM, L-BFGS, or related optimizers
10: **end while**

PINNs solved with SGD/ADAM methods are generally computationally much more demanding than Finite Element Method/Isogeometric Analysis solvers. Neural-network Jacobians are typically dense, highly nonlinear, and poorly conditioned Wang et al. (2021). In standard parameter-space formulations, replacing SGD/ADAM with Gauss–Newton can be challenging because the resulting systems may be dense, ill-conditioned, or rank-deficient. Classical residual losses employed by PINNs are generally not equivalent to physically meaningful energy norms Rojas et al. (2024). Robust variational residual minimization introduces weighted losses of the form $\Phi(u) = \frac{1}{2}R(u)^T G^{-1} R(u)$, where G is a symmetric positive definite Gram matrix associated with a variational inner product Rojas et al. (2024); Łoś et al. (2025). Following the discrete weak formulation described in Łoś et al. (2025) we have

$$\frac{1}{\mu}\sqrt{\Phi(u)} \leq \|u - u_{exact}\|_{\nabla,h} \leq \frac{1}{\alpha}\sqrt{\Phi(u)}. \tag{2}$$

Thus, the robust loss becomes equivalent to the discretization of the true error measured in suitable discrete Sobolev norms Łoś et al. (2025). The classical approach to solving CRVPINNs is to use stochastic gradient descent or ADAM. The training of the CRVPINN method is similar to the training of PINN. It additionally requires an LU factorization of the Gram matrix $G = LU$ before the while loop (before line 4) and performing forward and backward substitutions with the PINN loss. As it has been shown in Łoś et al. (2025), the training times for PINNs and CRVPINNs are similar. The CRVPINNs method gives the robust control of the loss function value and its relation with the true error. In practice, replacing SGD/ADAM with an iterative Gauss–Newton method is challenging for both PINNs and CRVPINNs because the resulting systems can be singular or numerically unstable.

3 IGA-ODIL AND FAST RESIDUAL MINIMIZATION

In the IGA-ODIL method, the unknown solution is represented using tensor-product B-splines. For example, for the Poisson equation,

$$u_c(x,y) = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} c_{i,j}\, B_i^{(p)}(x)\, B_j^{(q)}(y), \quad R(u_c) = \sum_{\hat{x},\hat{y}} \left(\frac{\partial^2 u_c(\hat{x},\hat{y})}{\partial x^2} + \frac{\partial^2 u_c(\hat{x},\hat{y})}{\partial y^2} + f(\hat{x},\hat{y}) \right)^2 \tag{3}$$

$$= \sum_{\hat{x},\hat{y}} \left(\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} c_{i,j} \left[\frac{\partial^2 B_i^{(p)}(\hat{x})}{\partial x^2} B_j^{(q)}(\hat{y}) + B_i^{(p)}(\hat{x}) \frac{\partial^2 B_j^{(q)}(\hat{y})}{\partial y^2} \right] + f(\hat{x},\hat{y}) \right)^2 \tag{4}$$

The trainable variables are the spline coefficients c. In general, the residual evaluated at collocation points $(\hat{x}_k, \hat{y}_k)$ is $R_k(c) = \mathcal{L}u_c(\hat{x}_k, \hat{y}_k) - f(\hat{x}_k, \hat{y}_k)$. The residual functional is given by $\Phi(u) = \frac{1}{2}R(u)^T R(u)$. For the IGA-ODIL method it is now possible to derive the Gauss–Newton algorithm, since the Jacobian matrix associated with IGA-ODIL is sparse and well-conditioned, contrary to PINN or CRVPINN Jacobians. Using the linearization $R(u+\delta) \approx R + J\delta$, we have $\Phi(u+\delta) = \frac{1}{2}\left(R^T R + 2\delta^T J^T R + \delta^T J^T J\delta\right)$. Differentiating with respect to δ gives $\nabla_\delta \Phi = J^T R + J^T J\delta$. Setting the gradient to zero yields the weighted normal equations: $(J^T J)\delta = -J^T R$. To solve equation 3, we require the Jacobian of the residual with respect to the spline coefficients, $J(c) = \frac{\partial R(c)}{\partial c}$. For linear PDEs the residual has the affine form $R(c) = Ac - b$, which implies $J = A$. Consequently, Gauss–Newton becomes equivalent to solving a least-squares PDE discretization. In other words, the Gauss–Newton method for linear problems is equivalent with solving a single system of linear equations. No robust loss modification is required, no robust loss value checking is needed. The overall method for solving IGA-ODIL is summarized in Algorithm 2. The Dirichlet boundary conditions are enforced strongly by eliminating the boundary B-spline coefficients. We select the collocation point uniformly over the computational domain leaving the adaptive algorithms for the future work.

Algorithm 2 Gauss–Newton Optimization for IGA–ODIL

1: Construct tensor-product B-spline basis functions $\{\phi_{i,j}(x,y)\} = \{B_i^{(p)}(x)B_j^{(q)}(y)\}$
2: Select collocation points $\{\mathbf{x}_k\}$
3: Initialize spline coefficients $c^{(0)}$
4: **while** not converged **do**
5: Evaluate spline solution $u_c^{(k)}(x,y) = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} c_{i,j}^{(k)} B_i^{(p)}(x)B_j^{(q)}(y)$
6: Evaluate residual vector $R(c^{(k)})$
7: Assemble Jacobian matrix $J(c^{(k)}) = \frac{\partial R(c^{(k)})}{\partial c}$
8: Solve the linearized system $J^T J\delta^{(k)} = -J^T R$
9: Update spline coefficients $c^{(k+1)} = c^{(k)} + \delta^{(k)}$
10: **end while**

4 Numerical Experiments

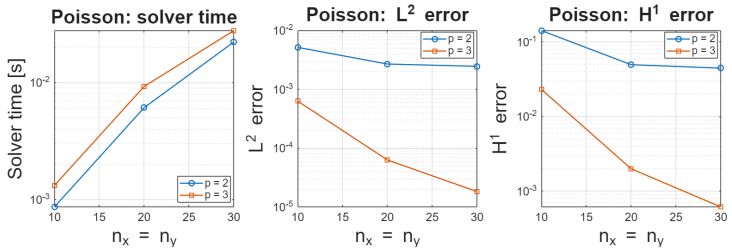

Figure 2: Accuracy and timing of IGA-ODIL solution for the Poisson problem for B-splines of order $p = 2, 3$.

4.1 Poisson Problem

We first consider the Poisson equation $-\Delta u = f$ with manufactured solution $u(x,y) = \sin(2\pi x)\sin(2\pi y)$. $f(x,y) = 8\pi^2 \sin(2\pi x)\sin(2\pi y)$. We solve the problem by CRVPINN Łoś et al. (2025) (PINN extension introducing robust loss). The training time using the ADAM algorithm with 100×100 collocation points and 2 fully connected neural network layers with 100 neurons is 6-7 minutes on Google Compute Engine GPU for both CRVPINN and PINN. The obtained MSE error is 0.0004. Replacing the ADAM method with the Gauss–Newton formulation with Neural Network parameterization results in lack of convergence. The IGA-ODIL method replaces neural-network parameterizations with spline coefficients. The IGA-ODIL system is solved

Table 1: Results for the 2D Poisson problem solved with sparse IGA–ODIL.

p	$n_x = n_y$	N_c	Solver time [s]	L^2 error	H^1 error
2	10	30	4.231×10^{-4}	5.187×10^{-3}	1.420×10^{-1}
2	20	30	2.733×10^{-3}	2.700×10^{-3}	4.960×10^{-2}
2	30	60	1.013×10^{-2}	2.466×10^{-3}	4.465×10^{-2}
3	10	30	5.787×10^{-4}	6.324×10^{-4}	2.311×10^{-2}
3	20	40	6.862×10^{-3}	6.303×10^{-5}	2.000×10^{-3}
3	30	60	1.680×10^{-2}	1.824×10^{-5}	6.171×10^{-4}

Table 2: Results for the 2D Poisson problem solved with ODIL.

N_c	Solver time [s]	L^2 error
30	2.926×10^{-4}	1.961×10^{-3}
60	1.479×10^{-3}	4.728×10^{-4}

using the Gauss–Newton method in MATLAB on CPU. Detailed evaluation of IGA-ODIL for different numbers of elements $n_x = n_y$, collocation points N_c and order p of B-splines which are C^{p-1}-continous is presented in Table 1 and Figure 2. The accuracy increases with the order of the B-splines, provided there are enough collocation points. This is consistent with Theorem 1 presented in Appendix A. We switch now to the ODIL method, where the neural network is replaced by a vector of coefficients $u_\theta = \{u_{i,j}\}$ and $R(u_\theta) = \sum_{x_{i,j}} \left(\frac{\partial u_{i,j}}{\partial x} + \frac{\partial u_{i,j}}{\partial y} + f_1(x_{i,j}) \right)^2$. The execution times and accuracy for ODIL method is summarized in Table 2. The ODIL solution is available at the collocation points, while IGA-ODIL provides continuous solution.

4.2 Eriksson–Johnson Problem

To assess the performance in convection-dominated regimes, we consider the Eriksson–Johnson advection–diffusion problem: $-\varepsilon \Delta u + \boldsymbol{\beta} \cdot \nabla u = 0, \quad (x, y) \in \Omega$, where $\varepsilon > 0$ is the diffusion coefficient and $\boldsymbol{\beta} = (1, 0)^T$ is the advection field. Boundary conditions are prescribed as: $u(0, y) = \sin(\pi y)$, on the inflow, and $u(x, y) = 0$, on remaining boundaries. For small values of ε, the solution exhibits boundary layers near the outflow boundary $x = 1$, making this a challenging benchmark for numerical methods. The problem exhibits the exact solution $u_{exact}(x, y) = \frac{(e^{(r_1(x-1))} - e^{(r_2(x-1))})}{(e^{(-r_1)} - e^{(-r_2)})} \sin(\pi y)$, $r_1 = \frac{(1+\sqrt{(1+4\epsilon^2\pi^2)})}{(2\epsilon)}$, $r_2 = \frac{(1-\sqrt{(1+4\epsilon^2\pi^2)})}{(2\epsilon)}$. A detailed evaluation of IGA-ODIL for $\epsilon = 0.1$, for different number of elements $n_x = n_y$, collocation points N_c and order p of B-splines which are C^{p-1}-continous is presented in Table 3 and Figure 3. The accuracy increases with order of B-splines, provided there are enough collocation points. This is consistent with theoretical estimate of Theorem 1 from Appendix A. The CRVPINN method solves this problem for $\epsilon = 0.1$ on Google Colab using 100×100 collocation points. The training time is around 6-7 minutes.

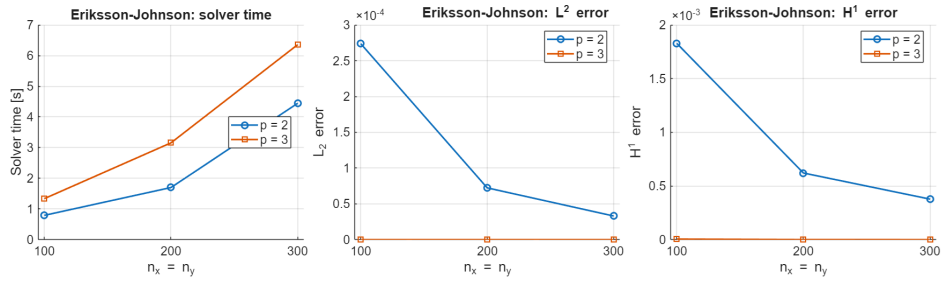

Figure 3: Accuracy and timing of IGA-ODIL solution for the Eriksson-Johnson problem for $\epsilon = 0.1$ for B-splines of order $p = 2, 3$.

Table 3: Solver time and L^2 and H^1 error of the IGA–ODIL solution for the Eriksson–Johnson problem with $\varepsilon = 0.1$ using B-splines of order $p = 2$ and $p = 3$.

p	$n_x = n_y$	N_c	Solver time [s]	L^2 error	H^1 error
2	100	500	3.440×10^{-1}	2.738×10^{-4}	1.808×10^{-3}
2	200	500	7.284×10^{-1}	7.241×10^{-5}	6.168×10^{-4}
2	300	500	1.564×10^{0}	3.310×10^{-5}	3.763×10^{-4}
3	100	500	5.366×10^{-1}	6.824×10^{-8}	5.617×10^{-6}
3	200	500	1.431×10^{0}	5.112×10^{-8}	1.146×10^{-6}
3	300	500	2.730×10^{0}	4.268×10^{-8}	7.309×10^{-7}

4.3 Three-Dimensional Helmholtz Problem

We consider the Helmholtz problem: $-\Delta u + \alpha u = f, \quad (x, y, z) \in \Omega$, with homogeneous Dirichlet boundary conditions $u(x, y, z) = 0, \quad (x, y, z) \in \partial\Omega$. A manufactured solution is chosen as $u(x, y, z) = \sin(\kappa\pi x)\sin(\kappa\pi y)\sin(\kappa\pi z)$, where $\alpha = \kappa^2 \in \mathbb{N}$ controls the frequency. The corresponding forcing term is derived by substituting the manufactured solution into the PDE. We solve the problem for $\kappa = 10$ using $20\times20\times20$ elements, cubic B-splines, $40\times40\times40$ collocation points, and the Gauss–Newton method, see Figure 4. The resulting accuracy in L^2 norm is 1.902649e-02. The problem takes 7.82 sec on a MATLAB single core CPU.

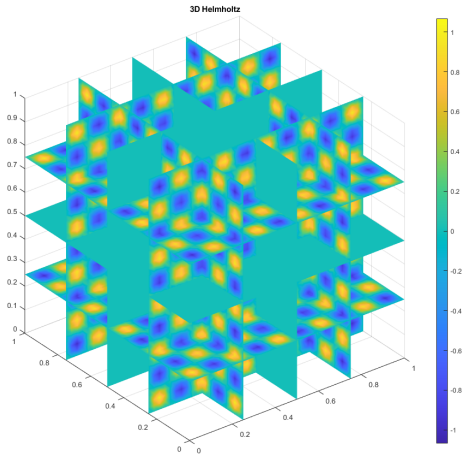

Figure 4: Multi-slice visualization of the 3D Helmholtz solution using cubic B-splines and residual minimization. $20 \times 20 \times 20$ elements and cubic B-splines with $40 \times 40 \times 40$ collocation points for $\kappa = 10$.

4.4 Allen–Cahn Equation

We consider the steady two-dimensional Allen–Cahn equation $-\varepsilon^2\Delta u+(u^3-u) = f, \quad (x, y) \in \Omega$,on $\Omega = (0, 1)^2$, subject to Dirichlet boundary conditions $u = g \quad$ on $\partial\Omega$. To construct a manufactured solution, we choose $u(x, y) = \tanh\left(\frac{x+y-1}{\sqrt{2}\,\varepsilon}\right)$. The corresponding Dirichlet data are therefore given by $g(x, y) = \tanh\left(\frac{x+y-1}{\sqrt{2}\,\varepsilon}\right), (x, y) \in \partial\Omega$. The corresponding forcing term is $f(x, y) = \tanh\left(\frac{x+y-1}{\sqrt{2}\varepsilon}\right) - \tanh^3\left(\frac{x+y-1}{\sqrt{2}\varepsilon}\right)$. The correctness of this manufactured forcing term is verified by substituting the manufactured solution into the PDE, as detailed in Appendix B. We evaluate IGA–ODIL for $\varepsilon = 0.01$ using different numbers of elements $n_x = n_y$, numbers of collocation points N_c, and B-spline orders p. The B-spline basis functions have C^{p-1} continuity within the parameterization. The Dirichlet boundary conditions are imposed strongly by including boundary collocation equations $u_h(x_b, y_b) - g(x_b, y_b) = 0, (x_b, y_b) \in \partial\Omega$, in addition to the Allen–Cahn residual equations at the interior collocation points. Detailed results are reported in Table 4 and

Table 4: Results for the Allen–Cahn problem solved with sparse IGA–ODIL/Newton iterations.

p	$n_x = n_y$	N_c	Time [s]	L^2 error	H^1 error
2	10	60	8.055×10^{-2}	2.128×10^{-1}	1.218×10^{1}
2	20	60	9.780×10^{-2}	1.113×10^{-1}	1.008×10^{1}
2	30	60	1.968×10^{-1}	6.489×10^{-2}	8.054×10^{0}
2	40	80	3.612×10^{-1}	4.028×10^{-2}	5.652×10^{0}
2	80	160	3.612×10^{-1}	3.519×10^{-2}	2.818×10^{0}
3	10	60	6.224×10^{-2}	2.131×10^{-1}	1.213×10^{1}
3	20	60	1.307×10^{-1}	1.085×10^{-1}	9.973×10^{0}
3	30	60	2.615×10^{-1}	6.070×10^{-2}	7.760×10^{0}
3	40	80	5.326×10^{-1}	3.596×10^{-2}	5.176×10^{0}
3	80	160	5.326×10^{-1}	3.602×10^{-2}	9.054×10^{-1}

Figure 6. In general, the accuracy improves as the number of elements is increased, provided that the collocation set is sufficiently rich to resolve the sharp transition layer.

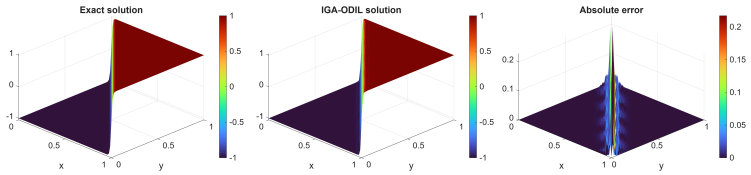

Figure 5: IGA-ODIL solution for the Allen–Cahn equation.

5 Inverse Problems

IGA-ODIL naturally extends to PDE-constrained inverse problems due to its residual-based optimization formulation. In contrast to classical finite element approaches, where inverse problems often require repeated nested forward PDE solves, the proposed framework simultaneously optimizes both the state variables and unknown physical parameters within a unified optimization procedure. We consider inverse identification of parameters in the Helmholtz equation: $-\Delta u + \alpha u = f, \quad (x, y) \in \Omega$, subject to zero homogeneous Dirichlet boundary conditions. A manufactured solution is prescribed as $u(x, y) = \sin(\kappa_{\text{true}} \pi x) \sin(\kappa_{\text{true}} \pi y)$, where κ_{true} denotes the unknown frequency parameter to be recovered. The corresponding forcing term becomes $f(x, y) = \left(2(\kappa_{\text{true}} \pi)^2 + \alpha\right) \sin(\kappa_{\text{true}} \pi x) \sin(\kappa_{\text{true}} \pi y)$. The objective of the inverse problem is to recover the unknown parameter κ from observations of the solution field. The spline approximation

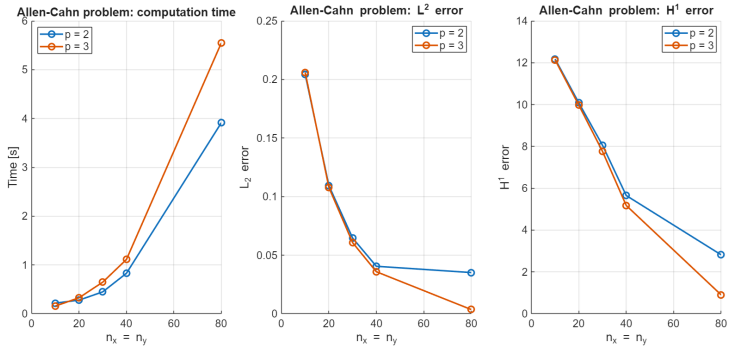

Figure 6: Accuracy and timing of IGA-ODIL solution for the Allen-Cahn problem for $\epsilon = 0.01$ for B-splines of order $p = 2, 3$.

is $u_c(x,y) = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} c_{ij} B_i^{(p)}(x) B_j^{(q)}(y)$. The inverse optimization problem is formulated as

$$\min_{c,\kappa} \left(\|R(c,\kappa)\|^2 + \lambda \|u_c - u_{\text{obs}}\|^2 \right), \tag{5}$$

where $\lambda > 0$ is a regularization parameter and u_{obs} denotes observed solution data. The spline coefficients c_{ij} and parameter κ are optimized simultaneously. The residual is given by $R_k(c,\kappa) = -\Delta u_c(x_k,y_k) + \kappa^2 u_c(x_k,y_k) - f(x_k,y_k)$. Expanding the spline representation $R_k(c,\kappa) = \sum_{i,j} c_{ij} \left[-\frac{\partial^2 B_i^{(p)}(x_k)}{\partial x^2} B_j^{(q)}(y_k) - B_i^{(p)}(x_k) \frac{\partial^2 B_j^{(q)}(y_k)}{\partial y^2} + \kappa^2 B_i^{(p)}(x_k) B_j^{(q)}(y_k) \right] - f(x_k,y_k)$. The data misfit term $\|u_c - u_{\text{obs}}\|^2 = \sum_k \left(u_c(x_k,y_k) - u_{\text{obs}}(x_k,y_k)\right)^2$. The optimization functional

$$\Phi(c,\kappa) = \frac{1}{2}\sum_k R_k(c,\kappa)^2 + \frac{\lambda}{2}\sum_k \left(u_c(x_k,y_k) - u_{\text{obs}}(x_k,y_k)\right)^2. \tag{6}$$

Let $\theta = [\mathbf{c}^T, \kappa^T]$ denotes the vector of optimization variables. The residual vector is linearized as $R(\theta + \delta\theta) \approx R(\theta) + J(\theta)\delta\theta$, where $J(\theta) = \frac{\partial R}{\partial \theta}$is the Jacobian matrix. The Gauss–Newton system becomes $J^T J\, \delta\theta = -J^T R$. Due to the local support of B-spline basis functions, the Jacobian remains sparse with respect to spline coefficients. The additional parameter κ introduces only one dense column corresponding to $\frac{\partial R_k}{\partial \kappa} = 2\kappa u_c(x_k,y_k)$. Consequently, the resulting system preserves favorable sparsity properties and remains computationally tractable. We have observed that better convergence is obtained if we replace the Gauss–Newton method with the Levenberg-Marquardt method with $\mu = 10^{-6}$ parameter. Thus, in our computations we take $\lambda = 10^7$ (similar order to μ).

We first consider the recovery of the parameter $\kappa_{\text{true}} = 2.0$ starting from the initial guess $\kappa_{\text{initial}} = 1.0$. The computation employs 30×30 elements, cubic B-splines, 40×40 collocation points, and 200 iterations. Figure 7 illustrates the convergence of the recovered parameter. We next consider

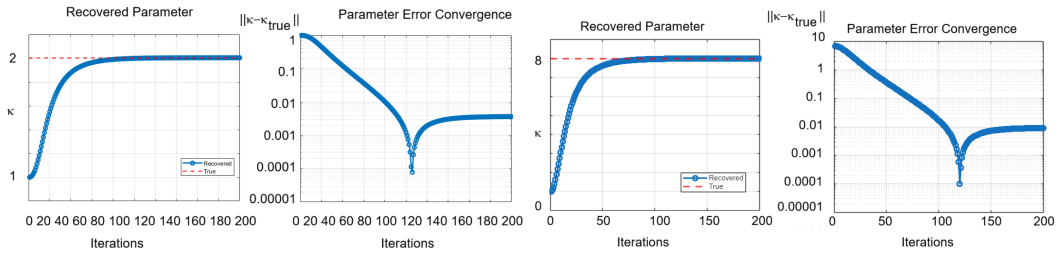

Figure 7: First and second panel: Convergence of inverse Helmholtz parameter recovery for $\kappa_{\text{true}} = 2$. Third and fourth panel: Convergence of inverse Helmholtz parameter recovery for $\kappa_{\text{true}} = 8$.

a substantially more oscillatory regime with $\kappa_{\text{true}} = 8$. The computational setup uses 40×40 spline elements, cubic B-splines, 80×80 collocation points, and 200 iterations. Figure 7 presents the convergence histories for the recovered parameter. The picks denote the moments where the recovered parameter passed κ_{true}.

6 Conclusions

The proposed method builds on the residual-minimization framework of Physics-Informed Neural Networks, and replaces the nonlinear neural-network parameterization with a B-spline representation. This replacement yields a sparse Jacobian matrix, in contrast to the dense and potentially ill-conditioned Jacobian associated with neural-network parameterizations, enabling efficient second-order Gauss–Newton optimization. In the considered implementations, IGA–ODIL achieves wall-clock speedups of up to two–three orders of magnitude over the corresponding first-order PINN and CRVPINN implementations. We call the method IGA-ODIL, optimizing discrete loss with isogeometric analysis representation. The method is verified on a sequence of computational problems, including Poisson problem, Eriksson-Johnson problem, Helmholtz problem in 2D and 3D, non-linear Allen-Cahn, and the inverse Helmholtz problem. Standard tensor-product IGA-ODIL is primarily targeted at low- to moderate-dimensional PDEs. High-dimensional extensions require sparse-grid, low-rank, hierarchical, or other structured spline representations.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the generation of MATLAB code used in this paper. In particular, we employed ChatGPT 5.2 Go, and we used the following prompts

- *Can you generate MATLAB code for ODIL for Poisson equation with 0 Dirichlet boundary conditions over* $[0,1]^2$ *using manufactured solution with exact solution* $u(x,y) = \sin(2\pi x)\sin(2\pi y)$. This prompt was supported by uploading the ODIL paper arxiv [1]. The goal of this prompt was to obtain the reference code for ODIL method. The obtained code was tested and debugged by authors without AI.
- *Can you replace optimization with iterative Gauss–Newton solver using linearization of the residual and iterative updates?* Since the generated ODIL code contained the SGD first-order optimization method, this prompt was employed to switch the code into Gauss–Newton method. The resulting code was tested and debugged by the authors without AI assistance.
- *Can you modify the MATLAB code so instead of using finite difference values it will use coefficients of B-spline function combinations?* The goal of this prompt was to replace the collocation points of ODIL method by the B-splines combinations of IGA-ODIL method. The resulting code was tested and debugged by the authors without AI assistance.
- *Can you make Erikkson-Johnson on a square domain, with* $\sin(\pi y)$ *on the boundary* $x = 0$*, with* $\beta = (1,0), \epsilon = 0.1$ *and 0 Dirichlet b.c on three other boundaries?* The goal of this prompt was to move from Poisson problem to Eriksson-Johnson problem. The resulting code was tested and debugged by the authors without AI assistance.
- *Can you update this code to non-linear Allen-Cahn equations?* The goal of this prompt was to move from Eriksson-Johnson linear problem to non-linear model problem. The obtained code was tested and debugged by authors without AI.
- *Can you make now Helmholtz problem with manufactured solution* $\sin(\kappa\pi x)sin(\kappa\pi y)$ *and put* κ *as a parameter?* The goal of this prompt was to move into Helmholtz problem. The resulting code was tested and debugged by the authors without AI assistance.
- *Can you generalize Helmholtz problem into 3D?* The goal of this prompt was to move into 3D Helmholtz problem. The resulting code was tested and debugged by the authors without AI assistance.
- *Can you generate MATLAB code for the inverse Helmholtz problem. Given* $-\Delta u(x,y) + \alpha u(x,y) = f(x,y)$ *with* $f(x,y)$ *constructed using the manufactured solution* $u(x,y) = \sin(\kappa\pi x)\sin(\kappa\pi y)$ *find* κ *using IGA-ODIL solution of the Helmholtz problem and MSE with measurements computed from* $u_{true}(x,y) = \sin(\kappa_{true}\pi x)\sin(\kappa_{true}\pi y)$*. Thus, the loss has the IGA-ODIL residuum of Helmholtz and MSE of the computed* u *with respect to* u_{true}*. Use the Gauss–Newton method for the solution.* The goal of this prompt was to use the IGA-ODIL method to solve inverse problem of identification of κ parameter for the 2D Helmholtz problem. The resulting code was tested and debugged by the authors without AI assistance.

We have not used generative AI tools to develop the PINN or CRVPINN codes. We have not used generative AI tools for developing the IGA-ODIL formulations, writing the paper and setting up the model problems. We have used ChatGPT to consult on the correctness of the convergence proof from Appendix A. We have used ChatGPT to check the correctness of the forcing for Allen-Cahn formulation in Appendix B. We have used ChatGPT to check for recent references, and verified its finding using our personal knowledge. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The Poisson problem in Section 4.1 and the Eriksson-Johnson problem in Section 4.2 use the PINN and CRVPINN python *code1.ipynb* attached to this paper. Running the code requires selection of

[1] https://arxiv.org/pdf/2205.04611

the problem by setting `EXAMPLE=1` for the Poisson problem and `EXAMPLE=3` for the Eriksson-Johnson problem. Selection between PINN and CRVPINN requires setting `RPINN=0` for PINN experiments and `RPINN=1` for CRVPINN experiments. The code implements also the Poisson problem with sin-exp solution `EXAMPLE=2` and the Helmholtz problem `EXAMPLE=4`. Running the Eriksson-Johnson problem requires setting of the `EPSILON` model parameter. Running the Helmholtz problem requires setting of `KAPPA` model parameter. The IGA-ODIL MATLAB Poisson code from Section 4.1 is attached in file *Poisson.m*. The ODIL MATLAB Poisson code from Section 4.1 is attached in file *Poisson_ODIL.m*. The IGA-ODIL MATLAB Eriksson-Johnson code from Section 4.2 is attached in file *ErikssonJohnson.m*. The IGA-ODIL MATLAB Helmholtz 3D code from Section 4.3 is attached in file *Helmholtz3D.m*. The IGA-ODIL MATLAB Allan-Cahn code from Section 4.4 is attached in file *AllanCahn.m*. The IGA-ODIL MATLAB inverse Helmholtz 2D code from Section 4.4 is attached in file *Helmholtz_inverse.m*. Notice that for linear problems, the Gauss–Newton method requires only one iteration and it is actually equivalent with solving a system of linear equations.

REFERENCES

F. Aurichio, L. Beirão 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. URL `https://doi.org/10.1142/S0218202510004878`.

Y. Bazilevs, L. Beirão 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*, 16:1031–1090, 2006.

Y. Bazilevs, K. Takizawa, T. E. Tezduyar, A. Korobenko, T. Kuraishi, and Y. Otoguro. Computational aerodynamics with isogeometric analysis. *Journal of Mechanics*, 39:24–39, 2023.

A. Buffa, G. Sangalli, and R. Vázquez. Isogeometric methods for computational electromagnetics: B-spline and t-spline discretizations. *Journal of Computational Physics*, 257:1291–1320, 2014.

S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37:1727–1738, 2021.

L. Chen, B. Li, and R. de Borst. Adaptive isogeometric analysis for phase-field modeling of anisotropic brittle fracture. *International Journal for Numerical Methods in Engineering*, 121: 4630–4648, 2020.

J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009.

Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018.

J. Gao, M. Xiao, Y. Zhang, and L. Gao. A comprehensive review of isogeometric topology optimization: Methods, applications and prospects. *Chinese Journal of Mechanical Engineering*, 33: 87, 2020.

S. Goswami, C. Anitescu, and T. Rabczuk. Adaptive fourth-order phase field analysis for brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 361:112808, 2020.

L. Greco, A. Patton, M. Negri, A. Marengo, U. Perego, and A. Reali. Higher order phase-field modeling of brittle fracture via isogeometric analysis. *Engineering with Computers*, 40:3541–3560, 2024.

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*, 194:4135–4195, 2005.

T. J. R. Hughes, G. Sangalli, T. Takacs, and D. Toshniwal. Smooth multi-patch discretizations in isogeometric analysis. In *Handbook of Numerical Analysis*, volume 22, pp. 467–543. 2021.

D. Jalili, S. Jang, M. Jadidi, G. Giustini, A. Keshmiri, and Y. Mahmoudi. Physics-informed neural networks for heat transfer prediction in two-phase flows. *International Journal of Heat and Mass Transfer*, 221:125089, 2024.

Anas Jnini and Flavio Vella. Dual natural gradient descent for scalable training of physics-informed neural networks, 2025. URL https://arxiv.org/abs/2505.21404.

P. Karnakov, S. Litvinov, and P. Koumoutsakos. Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks. *PNAS Nexus*, 3:pgae005, 2024.

G. E. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440, 2021.

E. Kharazmi, Z. Zhang, and G. E. Karniadakis. hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015. URL https://mlanthology.org/iclr/2015/kingma2015iclr-adam/.

T. Kuraishi, Z. Xu, K. Takizawa, T. E. Tezduyar, and T. Kakegami. Space–time isogeometric analysis of tire aerodynamics with complex tread pattern, road contact, and tire deformation. *Computational Mechanics*, 75:575–591, 2025.

Z. Li, N. Kovachki, B. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint*, 2020.

Hongwei Lin, Yunyang Xiong, Xiao Wang, Qianqian Hu, and Jingwen Ren. Isogeometric least-squares collocation method with consistency and convergence analysis. *Journal of Systems Science and Complexity*, 33(5):1656–1693, 2020. ISSN 1559-7067. doi: 10.1007/s11424-020-9052-9.

M. Łoś, T. Służalec, P. Maczuga, A. Vilkha, C. Uriarte, and M. Paszyński. Collocation-based robust variational physics-informed neural networks (crvpinn). *Computers and Structures*, 316:107904, 2025.

N. Nguyen-Thanh, W. Li, J. Huang, and K. Zhou. Adaptive higher-order phase-field modeling of anisotropic brittle fracture in 3d polycrystalline materials. *Computer Methods in Applied Mechanics and Engineering*, 372:113434, 2020.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

Subhajit Patra, Sonali Panda, Bikram Keshari Parida, Mahima Arya, Kurt Jacobs, Denys I. Bondar, and Abhijit Sen. Physics informed kolmogorov-arnold neural networks for dynamical analysis via efficent-kan and wav-kan, 2024. URL https://arxiv.org/abs/2407.18373.

A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer, 1994.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

S. Rojas, P. Maczuga, J. Muñoz-Matute, D. Pardo, and M. Paszyński. Robust variational physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 425:116902, 2024.

M. Torre, S. Morganti, F. S. Pasqualini, and A. Reali. Current progress toward isogeometric modeling of the heart biophysics. *Biophysics Reviews*, 4:041301, 2023.

S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43:A3055–A3081, 2021.

Yizheng Wang, Jia Sun, Jinshuai Bai, Cosmin Anitescu, Mohammad Sadegh Eshaghi, Xiaoying Zhuang, Timon Rabczuk, and Yinghua Liu. Kolmogorov–arnold-informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov–arnold networks. *Computer Methods in Applied Mechanics and Engineering*, 433: 117518, 2025. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2024.117518. URL https://www.sciencedirect.com/science/article/pii/S0045782524007722.

Zhuoyuan Wang, Raffaele Romagnoli, Saviz Mowlavi, and Yorie Nakahira. Physics-informed deep b-spline networks, 2026. URL https://arxiv.org/abs/2503.16777.

Joseph Webb, Sadok Jerad, and Coralia Cartis. An optimisation framework for the well-conditioned training of physics-informed neural networks, 2026. URL https://arxiv.org/abs/2607.02194.

X. Wei, X. Li, K. Qian, T. J. R. Hughes, Y. J. Zhang, and H. Casquero. Analysis-suitable unstructured t-splines: Multiple extraordinary points per face. *Computer Methods in Applied Mechanics and Engineering*, 391:114494, 2022.

Z. Wen, M. S. Faruque, X. Li, X. Wei, and H. Casquero. Isogeometric analysis using g-spline surfaces with arbitrary unstructured quadrilateral layout. *Computer Methods in Applied Mechanics and Engineering*, 408:115965, 2023.

R. Willems, K. L. P. M. Janssens, P. H. M. Bovendeerd, C. V. Verhoosel, and O. van der Sluis. An isogeometric analysis framework for ventricular cardiac mechanics. *Computational Mechanics*, 73:465–506, 2024.

J. Xu, H. Wei, and H. Bao. Physics-informed neural networks for studying heat transfer in porous media. *International Journal of Heat and Mass Transfer*, 217:124671, 2023.

S. Yang, H. Kim, Y. Hong, K. Yee, R. Maulik, and N. Kang. Data-driven physics-informed neural networks: A digital twin perspective. *Computer Methods in Applied Mechanics and Engineering*, 428:117075, 2024.

L. Zhu, X. Jiang, A. Lefauve, R. R. Kerswell, and P. F. Linden. New insights into experimental stratified flows obtained through physics-informed neural networks. *Journal of Fluid Mechanics*, 981:R1, 2024.

A CONVERGENCE OF IGA–ODIL

Theorem 1 (Convergence of spline-based residual minimization)**.** *Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain, and consider the boundary value problem*

$$\mathcal{L}u = f \quad \text{in } \Omega \tag{7}$$

supplemented with homogeneous essential boundary conditions such that the problem is well posed.

Let $X \subset H^s(\Omega)$ denote the space of functions satisfying the homogeneous essential boundary conditions, and let $V_h^0 \coloneqq V_h \cap X$, where $V_h \subset H^s(\Omega)$ is a tensor-product B-spline space of degree p constructed on a quasi-uniform mesh with characteristic size h, and let $W_h \coloneqq \mathcal{V}_\langle'$. Let $\mathcal{X}_M = \{x_k\}_{k=1}^M \subset \Omega$ be a set of collocation points, and let us define a seminorm

$$|w|_{Q_h}^2 = \frac{|\Omega|}{M} \sum_{i=1}^{M} |w(x_i)|^2. \tag{8}$$

Assume:

1. *$\mathcal{L}\colon H^s(\Omega) \to L^2(\Omega)$ is a linear differential operator of order s.*
2. *The exact solution satisfies*
$$u \in H^{p+1}(\Omega) \cap X. \tag{9}$$

3. *There exists a quasi-interpolant $\Pi_h u \in V_h^0$ satisfying*

$$\|u - \Pi_h u\|_{H^s(\Omega)} \leq C_\Pi h^{p+1-s} |u|_{H^{p+1}(\Omega)} \tag{10}$$

and

$$\|\mathcal{L}(u - \Pi_h u)\|_{L^2(\Omega)} \leq C_\mathcal{L} h^{p+1-s} |u|_{H^{p+1}(\Omega)}. \tag{11}$$

4. *The collocation points satisfy the stable sampling condition for all elements of W_h:*

$$\|w_h\|_{Q_h} \leq K \|w_h\|_{L^2(\Omega)} \quad \forall w_h \in W_h, \tag{12}$$

and for the interpolation residual

$$\|\mathcal{L}(u - \Pi_h u)\|_{Q_h} \leq C_{app} \|\mathcal{L}(u - \Pi_h u)\|_{L^2(\Omega)}. \tag{13}$$

5. *The pointwise residual*

$$\mathcal{L} v_h(x_k) - f(x_k) \tag{14}$$

is well defined for every $v_h \in V_h^0$ and every collocation point x_k, and

$$\mathcal{L} u(x_k) = f(x_k) \tag{15}$$

holds at all collocation points.

6. *The operator satisfies the stability estimate*

$$\|w\|_{H^s(\Omega)} \leq C_{stab} \|\mathcal{L} w\|_{L^2(\Omega)} \quad \forall w \in X. \tag{16}$$

Let

$$u_h \coloneqq \operatorname*{arg\,min}_{v_h \in V_h^0} \|\mathcal{L} v_h - f\|_{Q_h} \tag{17}$$

Then there exists a constant $C > 0$ independent of h, such that

$$\|u - u_h\|_{H^s(\Omega)} \leq C h^{p+1-s} |u|_{H^{p+1}(\Omega)}. \tag{18}$$

Proof. By the triangle inequality,

$$\|u - u_h\|_{H^s(\Omega)} \leq \|u - \Pi_h u\|_{H^s(\Omega)} + \|\Pi_h u - u_h\|_{H^s(\Omega)} \tag{19}$$

The first term can be bounded by $C_\Pi h^{p+1-s} |u|_{H^{p+1}(\Omega)}$ using equation 10. For the second term, note that

$$\begin{aligned} \|\Pi_h u - u_h\|_{H^s(\Omega)} &\leq C_{\text{stab}} \|\mathcal{L}(\Pi_h u - u_h)\|_{L^2(\Omega)} \\ &\leq C_{\text{stab}} K \|\mathcal{L}(\Pi_h u - u_h)\|_{Q_h}, \end{aligned} \tag{20}$$

by equation 16 and equation 12, since $\Pi_h u - u_h \in V_h^0$. We have

$$\begin{aligned} \|\mathcal{L}(\Pi_h u - u_h)\|_{Q_h} &\leq \|\mathcal{L}(\Pi_h u - u)\|_{Q_h} + \|\mathcal{L}(u - u_h)\|_{Q_h} \\ &\leq 2 \|\mathcal{L}(\Pi_h u - u)\|_{Q_h} \end{aligned} \tag{21}$$

since

$$\|\mathcal{L}(u_h - u)\|_{Q_h} = \|\mathcal{L}(u_h) - f\|_{Q_h} \leq \|\mathcal{L}(\Pi_h u) - f)\|_{Q_h} = \|\mathcal{L}(\Pi_h u - u)\|_{Q_h} \tag{22}$$

by definition of u_h. Using equation 13 and equation 11, we get

$$\|\mathcal{L}(\Pi_h u - u_h)\|_{Q_h} \leq C h^{p+1-s} |u|_{H^{p+1}(\Omega)}. \tag{23}$$

Combining it with equation 20 and the bound for $\|u - \Pi_h u\|_{H^s(\Omega)}$ concludes the proof. □

B MANUFACTURED SOLUTION FOR ALLEN-CAHN EQAUTIONS

We consider the steady Allen–Cahn equation

$$-\varepsilon^2 \Delta u + (u^3 - u) = f \qquad \text{in } \Omega = (0,1)^2, \tag{24}$$

subject to Dirichlet boundary conditions. As a manufactured solution, we choose

$$u(x,y) = \tanh\left(\frac{x+y-1}{\sqrt{2}\varepsilon}\right). \tag{25}$$

Introducing

$$z = \frac{x+y-1}{\sqrt{2}\varepsilon}, \tag{26}$$

we have

$$u(x,y) = \tanh(z). \tag{27}$$

Since

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} = \frac{1}{\sqrt{2}\varepsilon}, \tag{28}$$

the second derivatives are given by

$$u_{xx} = -\frac{1}{\varepsilon^2}\tanh(z)\operatorname{sech}^2(z), \qquad u_{yy} = -\frac{1}{\varepsilon^2}\tanh(z)\operatorname{sech}^2(z). \tag{29}$$

Consequently,

$$\Delta u = -\frac{2}{\varepsilon^2}\tanh(z)\operatorname{sech}^2(z). \tag{30}$$

Using the identity

$$\operatorname{sech}^2(z) = 1 - \tanh^2(z), \tag{31}$$

we obtain

$$-\varepsilon^2 \Delta u = 2\tanh(z)\left(1 - \tanh^2(z)\right) = 2(u - u^3). \tag{32}$$

Substitution into equation 24 therefore gives

$$f = 2(u - u^3) + (u^3 - u) = u - u^3. \tag{33}$$

Hence, the corresponding manufactured forcing term is

$$f(x,y) = \tanh\left(\frac{x+y-1}{\sqrt{2}\varepsilon}\right) - \tanh^3\left(\frac{x+y-1}{\sqrt{2}\varepsilon}\right). \tag{34}$$

The Dirichlet boundary conditions are prescribed directly from the manufactured solution:

$$u|_{\partial\Omega} = \tanh\left(\frac{x+y-1}{\sqrt{2}\varepsilon}\right). \tag{35}$$