Hybridized Data Driven Flux-Conservative Solvers: Towards Foundation Models for PDE-Solving

Dirk Hartmann
Siemens Industry Software GmbH
81739 München, Germany
hartmann.dirk@siemens.com

Abstract

Numerical solution of Partial Differential Equations (PDE) is an indispensable tool in science and engineering. While Scientific Machine Learning offers potentially unique opportunities, current methods face challenges in scaling to real world applications. This work in progress report introduces Hybridized Data-Driven Flux-Conservative Solvers (H-DD-FCS). These combine three core principles: scalable domain decomposition, explicit flux conservation enforcement, and Newton-like iterative solvers leveraging modern differentiable ML frameworks. We demonstrate the feasibility of this approach along the 2D Poisson problem. Compared to state-of-the-art approaches, H-DD-FCS explicitly considers flux conservation and thus allows for better robustness, scalability, and accessibility to mathematical analysis. It offers a promising direction towards Foundation Models for PDE-solving.

1 Introduction

Computer simulations of Partial Differential Equations (PDEs) have become an indispensable tool in scientific practice [34]. Simulations allow to gain cost-effective insights and to virtually test hypotheses across a wide range of science and engineering disciplines. Thus, users continuously require faster and more accurate simulation technology to address more complex problems in shorter times. Scientific Machine Learning (SciML) [13, 29] offers a unique opportunity to address these needs. In this work in progress contribution, we introduce a novel concept which we coin Hybridized Data-Driven Flux-Conservative Solvers (H-DD-FCS). The concept is build on three core principles: scalable domain composition, explicit flux conservation enforcement, and Newton-like iterative solvers enabled by modern differentiable Machine Learning (ML) frameworks. These three principles provide a promising direction to overcome challenges of State-of-The-Art (SoTA) ML approaches with respect to scalable adoption to real world problems.

Background: SoTA SciML approaches for PDE-solving cover surrogate modeling and hybrid approaches. *Surrogate Modeling* has a long history in Computational Science and Engineering (CSE). Next to *Deep Learning* (addressing finite dimensional learning), *Operator Learning* addresses maps between infinite dimensional spaces as considered in PDEs [5, 9, 19, 20, 23]. Neural operators take parameter fields as input and predict the solution fields, mostly relying on transformer architectures [1, 2, 24, 37]. *Hybrid Approaches* exploit physics knowledge and offer a promising alternative. Among these *ML-based Modeling* has a long history [7, 15, 21, 22, 32, 33] and is typically used for deriving effective (sub-) models based on (synthetic) data. Solved with classical algorithms underlying conservation laws can be guaranteed. *Solver-inspired ML architectures* aim for better scalability by taking inspirations from classical solver architectures, e.g., finite elements [18], finite differences [28, 30], collocation methods [31], or spectral methods [36]. Given the robustness, data requirement, and better scalability of hybrid approaches, these are promising candidates for PDE Foundation Models (FMs) [10].

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

Challenges: Most SciML methods do not yet scale to real world cases. For example, many applications require the strict adherence to natural laws, specifically conservation laws. In SoTA approaches, these are usually only ensured via learning biases, e.g., [8], and rarely via inductive biases, e.g., [16, 25]. Furthermore, ML for 3D PDE-solving requires enormous dataset sizes. Industrial computational fluid dynamics simulations often have hundred millions of discretization cells [2]. However, most methods are only demonstrated along simplistic cases [6]. Scalable ML-methods are lacking. This includes also training strategies, specifically for stationary cases[1] as adopted in most industrial simulations. Most hybrid approaches rely on simple fix point schemes [31, 38] which are easy to implement but hard to scale.

2 Hybridized Data Driven Discontinuous Galerkin Solvers

Reference Case: This work in progress report focuses on the most simplistic problem of PDE-solving, the 2D Poisson problem on a rectangular domain $\Omega \subset \mathbb{R}^2$, i.e.,

$$-\Delta u = f \quad \text{in } \Omega \tag{1}$$

with $f \in L_2(\Omega)$ and zero Dirichlet boundary conditions, i.e., $u = 0$ on $\partial\Omega$.

Three Core Principles: The proposed hybrid ML architecture is inspired by SoTA solver paradigms in the field of CSE, so-called hybridized methods [11]. The proposed concept is based on three core principles: adopting a *divide and conquer* approach via domain decomposition following [14, 38, 10]; explicitly reinforcing *flux conservation* (difference to SoTA approaches) - a highly desirable property at the heart of many CSE algorithms [27, 17, 3]; leveraging *gradient-based*[2] Newton-like solvers, exploiting the differentiability of the ML frameworks and thus allowing more effective inference.

ML-based Static Condensation: Many modern domain decomposition schemes are based on static condensation [35]. With domain coupling depending only on variables on domain boundaries, individual sub-problems can be solved stand-alone first (static condensation) simplifying the solution procedure. We follow this concept by decomposing the domain into sub-elements $e \in \mathbb{E}$ and learning sub-element *solution operators* $\mathcal{S}$ and *flux operators* $\mathcal{F}$ using Neural Networks (NNs). The corresponding operators map for each sub-element e, the boundary variables[3] $u_{|V_e}$ and $u^e_{|F_e}$ (defined on vertices V_e and faces F_e) and the source $f_{|E_e}$ (defined on full elements E_e) to the solution field $u_{|E_e}$ (defined on full elements E_e) as well as the corresponding fluxes $J_{|F_e} = \vec{n} \cdot \nabla u_{|F_e}$ (defined on faces F_e with interface normal $\vec{n}$). Instead of working with the full set of physical variables directly, we adopt a (nonlinear) dimension identification, e.g., Principal Component Analysis (PCA) and Autoencoders (AEs), with encoders $\mathcal{E}$ and decoders $\mathcal{D}$. That is, we work with corresponding latent variables, indicated by a tilde.

Training Process: The training process is shown in Figure 1. First, the entire domain is discretized into sub-elements $e \in \mathbb{E}$ forming the dataset, which are effectively stand-alone problems with corresponding boundary conditions $u_{|V_e}$, $u_{|F_e}$ and forcing / source $f_{|E_e}$. In the next step, the relevant variables $u_{|E_e}$, $f_{|E_e}$, $u_{|F_e}$, $u_{|V_e}$, and $J_{|F_e}$ are encoded in a latent space. In the last step, the actual latent solution $\mathcal{S}$ and flux $\mathcal{F}$ operators are learned.

Inference Process: The inference process is shown in Figure 2. It starts by encoding boundary conditions (of the full domain) on $\partial\Omega$ to corresponding latent variables $\tilde{u}_{|V_e}$ and $\tilde{u}_{|F_e}$ of elements located at the boundary. All other vertex $\tilde{u}_{|V_e}$ and face variables $\tilde{u}_{|F_e}$ can be initialized arbitrarily. In the next step, the system to be solved is assembled. That is, for each inner face the balance of fluxes $\Delta J_{|F_i} = J_{|F_i,\text{left}} - J_{|F_i,\text{right}}$ is modeled using $\mathcal{F}$. Since each face only involves the two neighboring cells, effectively a sparse model is realized. This eases the solution process if appropriate solvers are used. The corresponding flux mismatch is then minimized, i.e., $\min_{\tilde{u}_{|V_e}, \tilde{u}_{|F_e}} \sum_i \|\Delta J_{|F_i}\|^2$. Leveraging a modern differentiable ML framework, we use gradient-based optimization[4]. In addition to sparsity patterns this allows for a highly effective solution process.

[1] In dynamic problems trajectory-based training has been identified as a very successful strategy [26].

[2] This refers to the inference and not to the training, which is always gradient-based.

[3] We distinguish between vertex and face variables to facilitate convergence of the hybridized approach.

[4] During the solution we use derivatives with respect to variables $\tilde{u}_{|V_e}$ and $\tilde{u}_{|F_e}$, not NN parameters θ.

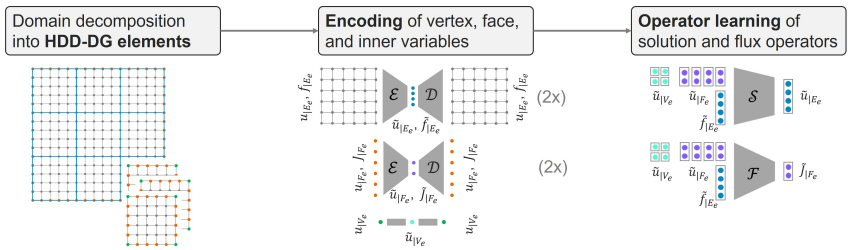

Figure 1: Schematic visualization of the H-DD-FCS training workflow. (We have two face and two element encoders $\mathcal{E}$: for fields and sources, respectively fields and fluxes.)

The full solution of the problem is now given in terms of latent vertex $\tilde{u}_{|V_e}$ and face $\tilde{u}_{|F_e}$ variables. Together with the original latent source information $\tilde{f}_{|E_e}$, the latent field solution $\tilde{u}_{|E_e}$ in each element can be recovered by $\mathcal{S}$. The full field is then retrieved via $\mathcal{D}_{u_{|E_e}}$.

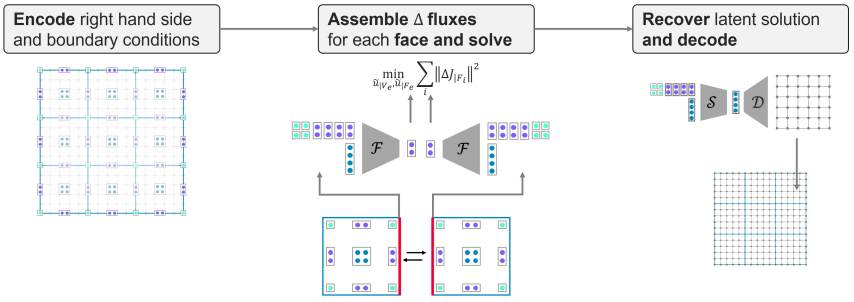

Figure 2: Schematic visualization of the H-DD-FCS inference workflow.

3 Experiments

The method is demonstrated along the Poisson problem (1). Using the finite element method with 70×70 elements and $\delta x = 0.1$, a dataset composed of 100 random source functions $f = a \exp(-(\vec{x} - \vec{\mu})^2/2\sigma^2)$ is generated ($a \in [0, 1]$, $\vec{\mu} \in [0, 7]^2$, and $\sigma \in [0.1, 0.5]$ equally distributed). The 100 data points are enriched to 700 data points[5] by rotations of 90°, 180°, and 270° as well as reflections in $x-$ and $y-$direction enforcing an invariance data-bias. The chosen sub-elements are of a size 8×8. Experiments are carried out on a standard Laptop with an Intel $i7 - 1185G7$ processor and 32GB RAM using the Julia language [4].

Physics Encoding: To demonstrate the effectiveness of the approach, we use a PCA-based linear dimension identification. We use four latent variables for encoding of sub-element volume fields and two latent variables for encoding of sub-element face fields. The results are shown in Figure 3. In addition we have explored an AE approach with similar results (see Appendix A).

Operator Learning: All operators are based on a dense NN with $32 \times 32 \times 16 \times 8$ neurons and ReLU activation function. The NN is trained with an Adams optimizer for 400 epochs (batch size 128), learning rate 10^{-3}, and regularization 10^{-5} without performing any hyperparameter studies. The results of the operator learning for the PCA encoding are shown in Figure 4. The corresponding training of AE-based operators shows a similar efficiency (see Appendix A).

Solving and Field Reconstruction: The approach is validated along two random source functions f not present in the training set. The obtained non-linear system is solved by means of a BFGS optimizer [12] (with 400 fixed steps) exploiting the differentiability of the ML-framework. The results are shown in Figure 5 for the PCA encoding and for AE encoding we refer to Appendix A. Using a standard BFGS optimizer is a main bottleneck, since it does not exploit the underlying

[5]The data will be made available upon request.

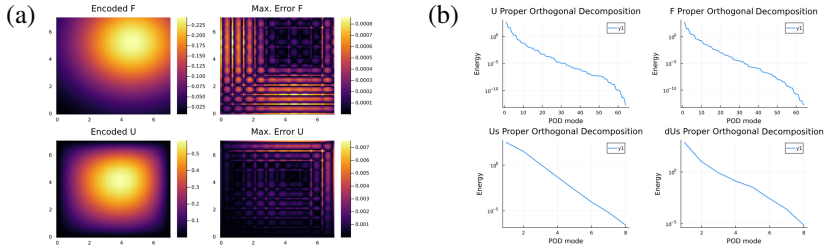

Figure 3: PCA-based element encoding: (a) comparison with original field; (b) Eigen-decomposition of field $u_{|E_e}$, source $f_{|E_e}$, face $u_{|F_e}$, and flux $J_{|F_e}$ values.

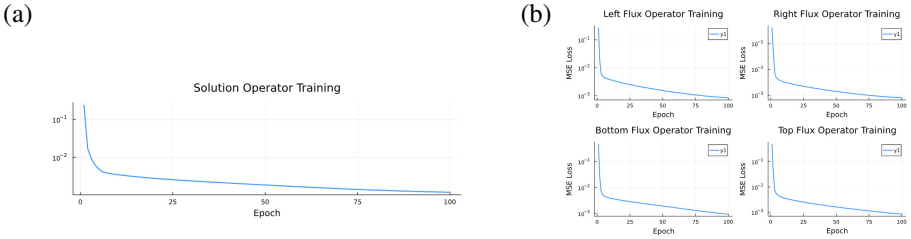

Figure 4: Loss evolution of the operator training (PCA encoding)): (a) solution operator $\mathcal{S}$, (b) flux operators $\mathcal{F}$.

structure of the system. This makes the inference process rather slow. We are currently investigating more effective non-linear inference methods based on SoTA CSE algorithms.

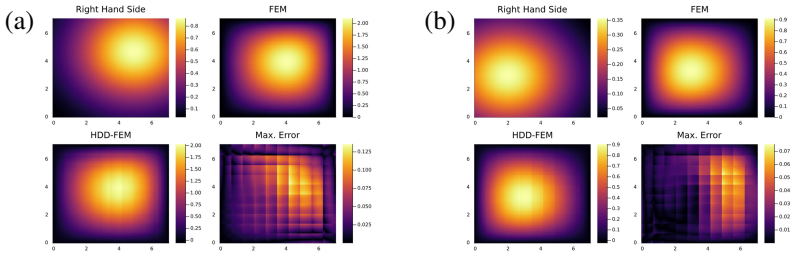

Figure 5: Validation of the method using PCA-encoding for two randomly chosen source functions f which have not been seen during the training.

4 Discussion and Ongoing work

In this contribution, we report on our work in progress on a new ML-based PDE-solver coined H-DD-FCS. Compared to SoTA approaches the solver explicitly enforces flux-conservation and leverages non-linear solving techniques with better convergence properties than SoTA iterative solvers increasing accuracy and robustness. The feasibility has been demonstrated along the 2D Poisson problem. In the next steps we will extend the approach to more complicated applications, specifically addressing non-linear and 3D problems. We will further benchmark it to ML SoTA approaches[6]. To do so, we plan to enhance the solving routine (so far a simple BFGS) exploiting better the structure of the underlying system. We believe that his will significantly increase the accuracy and capability of the approach. Based on a domain composition approach, H-DD-FCS is a candidate towards realizing FM [10] for industrial PDE-solvers.

[6]Preliminary comparisons with U-net architectures indicate a significantly superior performance.

References

[1] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

[2] Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPT: Scaling neural CFD surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. *arXiv preprint arXiv:2502.09692*, 2025.

[3] L Beirão da Veiga, Franco Brezzi, Luisa Donatella Marini, and Alessandro Russo. The hitchhiker's guide to the virtual element method. *Mathematical models and methods in applied sciences*, 24(08):1541–1573, 2014.

[4] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[5] Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. In *Handbook of Numerical Analysis*, volume 25, pages 83–125. Elsevier, 2024.

[6] Johannes Brandstetter. Envisioning better benchmarks for machine learning pde solvers. *Nature Machine Intelligence*, 7(1):2–3, 2025.

[7] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020.

[8] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[9] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[10] Youngsoo Choi, Siu Wun Cheung, Youngkyu Kim, Ping-Hsuan Tsai, Alejandro N Diaz, Ivan Zanardi, Seung Whan Chung, Dylan Matthew Copeland, Coleman Kendrick, William Anderson, et al. Defining foundation models for computational science: A call for clarity and rigor. *arXiv preprint arXiv:2505.22904*, 2025.

[11] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[12] Yu-Hong Dai. Convergence properties of the bfgs algoritm. *SIAM Journal on Optimization*, 13 (3):693–701, 2002.

[13] Felix Dietrich and Wil Schilders. Scientific machine learning. *Mathematische Semesterberichte*, 72(2):89–115, 2025.

[14] Victorita Dolean, Alexander Heinlein, Siddhartha Mishra, and Ben Moseley. Finite basis physics-informed neural networks as a schwarz domain decomposition method. In *International Conference on Domain Decomposition Methods*, pages 165–172. Springer, 2022.

[15] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51(1):357–377, 2019.

[16] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[17] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer, 2008.

[18] Dinh Bao Phuong Huynh, David J Knezevic, and Anthony T Patera. A static condensation reduced basis element method: approximation and a posteriori error estimation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(1):213–251, 2013.

[19] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[20] Nikola B Kovachki, Samuel Lanthaler, and Andrew M Stuart. Operator learning: Algorithms and analysis. *Handbook of Numerical Analysis*, 25:419–467, 2024.

[21] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[22] Katharina Krischer, Ramiro Rico-Martinez, Ioannis G Kevrekidis, Harm H Rotermund, Gerhard Ertl, and John L Hudson. Model identification of a spatiotemporally varying catalytic reaction. *AIChE Journal*, 39(1):89–98, 1993.

[23] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[24] Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

[25] Stefano Massaroli, Michael Poli, Federico Califano, Angela Faragasso, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Port–hamiltonian approach to neural network training. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6799–6806. IEEE, 2019.

[26] Hugo Melchers, Daan Crommelin, Barry Koren, Vlado Menkovski, and Benjamin Sanderse. Comparison of neural closure models for discretised PDEs. *Computers & Mathematics with Applications*, 143:94–107, 2023.

[27] Fadl Moukalled, Luca Mangani, and Marwan Darwish. The finite volume method. In *The finite volume method in computational fluid dynamics: An advanced introduction with OpenFOAM® and Matlab*, pages 103–135. Springer, 2015.

[28] Sheel Nidhan, Haoliang Jiang, Lalit Ghule, Clancy Umphrey, Rishikesh Ranade, and Jay Pathak. A domain decomposition-based autoregressive deep learning model for unsteady and nonlinear partial differential equations. *arXiv e-prints*, pages arXiv–2408, 2024.

[29] Alfio Quarteroni, Paola Gervasio, and Francesco Regazzoni. Combining physics-based and data-driven models: advancing the frontiers of research with scientific machine learning. *arXiv preprint arXiv:2501.18708*, 2025.

[30] Rishikesh Ranade, Chris Hill, Lalit Ghule, and Jay Pathak. A composable machine-learning approach for steady-state simulations on high-resolution grids. *Advances in Neural Information Processing Systems*, 35:17386–17401, 2022.

[31] Rishikesh Ranade, Mohammad Amin Nabian, Kaustubh Tangsali, Alexey Kamenev, Oliver Hennigh, Ram Cherukuri, and Sanjay Choudhry. Domino: A decomposable multi-scale iterative neural operator for modeling large scale engineering simulations. *arXiv preprint arXiv:2501.13350*, 2025.

[32] Ramiro Rico-Martinez, Katharina Krischer, Ioannis G Kevrekidis, MC Kube, and John L Hudson. Discrete-vs. continuous-time nonlinear signal processing of cu electrodissolution data. *Chemical Engineering Communications*, 118(1):25–48, 1992.

[33] Ramiro Rico-Martinez, JS Anderson, and Ioannis G Kevrekidis. Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 596–605. IEEE, 1994.

[34] Ulrich Rüde, Karen Willcox, Lois Curfman McInnes, and Hans De Sterck. Research and education in computational science and engineering. *Siam Review*, 60(3):707–754, 2018.

[35] Edward L Wilson. The static condensation algorithm. *International Journal for Numerical Methods in Engineering*, 8(1):198–203, 1974.

[36] Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023.

[37] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

[38] Rui Wu, Nikola Kovachki, and Burigede Liu. A learning-based domain decomposition method. *arXiv preprint arXiv:2507.17328*, 2025.

A Additional Experiments

Next to a PCA-based encoding we have also investigated an encoding by an AE. We use a dense NN with 128×64 neurons for field encoding and 16×8 neurons for face encoding with ReLU activation functions. The AE is trained with an Adams optimizer for 200 epochs (batch size 128), learning rate 10^{-3}, and regularization 10^{-6} without performing any hyperparameter studies.

The encoding experiments are shown in Figure 6. The convergence plots of the corresponding operator learning are shown in Figure 7, and the validation with two unseen fields f is provided in Figure 8. The results are comparable to the results obtained with a PCA encoding as shown in Section 3.

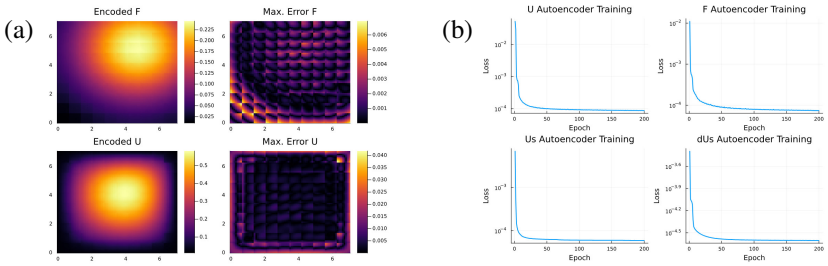

Figure 6: AE-based element encoding: (a) comparison with original field; (b) loss evolution for training the field $u_{|E_e}$, source $f_{|E_e}$, face u_{F_e}, and flux J_{F_e} encoding.

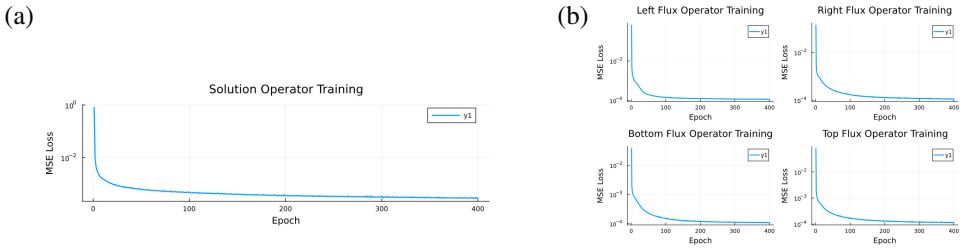

Figure 7: Loss evolution of the operator training (AE-encoding): (a) solution operator $\mathcal{S}$; (b) flux operators $\mathcal{F}$.

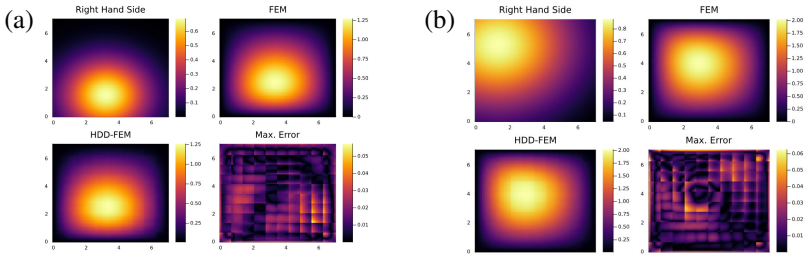

Figure 8: Validation of the method using AE-encoding for two randomly chosen source functions f which have not been seen during the training.