Computer Methods in Applied Mechanics and Engineering 444 (2025) 118111

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Stabilizing and solving unique continuation problems by parameterizing data and learning finite element solution operators

Erik Burman [a], Mats G. Larson [b], Karl Larsson [b,*], Carl Lundholm [b]

[a] Mathematics, University College London, UK
[b] Mathematics and Mathematical Statistics, Umeå University, Sweden

ARTICLE INFO

Keywords:
Inverse problems
Nonlinear PDE
Machine learning
Unique continuation problem

ABSTRACT

We consider an inverse problem involving the reconstruction of the solution to a nonlinear partial differential equation (PDE) with unknown boundary conditions. Instead of direct boundary data, we are provided with a large dataset of boundary observations for typical solutions (collective data) and a bulk measurement of a specific realization. To leverage this collective data, we first compress the boundary data using proper orthogonal decomposition (POD) in a linear expansion. Next, we identify a possible nonlinear low-dimensional structure in the expansion coefficients using an autoencoder, which provides a parametrization of the dataset in a lower-dimensional latent space. We then train an operator network to map the expansion coefficients representing the boundary data to the finite element (FE) solution of the PDE. Finally, we connect the autoencoder's decoder to the operator network which enables us to solve the inverse problem by optimizing a data-fitting term over the latent space. We analyze the underlying stabilized finite element method (FEM) in the linear setting and establish an optimal error estimate in the H^1-norm. The nonlinear problem is then studied numerically, demonstrating the effectiveness of our approach.

1. Introduction

Technological advances have led to measurement resolution and precision improvements, shifting the paradigm from data scarcity to abundance. While these data can potentially improve the reliability of computational predictions, it still needs to be determined how to consistently merge the data with physical models in the form of partial differential equations (PDE). In particular, if the PDE problem is ill-posed, as is typical for data assimilation problems, a delicate balancing problem of data accuracy and regularization strength has to be solved. If the data is inaccurate, the PDE problem requires strong regularization; however, if the data is accurate, such a strong regularization will destroy the accuracy of the approximation of the PDE. Another question is how to use different types of data. Some large data sets, consisting of historical data of events similar to the one under study, can be available. In contrast, a small set of measurements characterizes the particular realization we want to model computationally. In this case, the former data set measures the "experience" of the physical phenomenon, while the latter gives information on the current event to be predicted.

* Corresponding author.
E-mail addresses: e.burman@ucl.ac.uk (E. Burman), mats.larson@umu.se (M.G. Larson), karl.larsson@umu.se (K. Larsson), carl.lundholm@umu.se (C. Lundholm).

https://doi.org/10.1016/j.cma.2025.118111
Received 14 February 2025; Received in revised form 5 May 2025; Accepted 18 May 2025
Available online 10 June 2025
0045-7825/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

This is the situation that we wish to address in the present work. The objective is to construct a computational method that combines machine learning techniques for the data handling parts and hybrid network/finite element methods (FEMs) for the approximation in physical space. First, the large data set is mapped to a lower-dimensional manifold using an autoencoder or some other technique for finding low-dimensional structures, such as singular value decomposition or manifold learning. Then, we train a network to reproduce the solution map from the lower-dimensional set to the finite element space. Finally, this reduced order model solves a nonlinear inverse problem under the a priori assumption that the solution resides in a neighborhood of the lower-dimensional manifold.

To ensure an underpinning of the developed methods, we consider the case of a unique continuation problem for a nonlinear elliptic operator. That is, given some interior measurement (or measurements on the part of the boundary), a solution is reconstructed despite lacking boundary data on the part of the boundary. Such problems are notoriously ill-posed, and using only the event data set, it is known that the accuracy of any approximation in the whole domain cannot be guaranteed due to the poor global stability [1]. Indeed, in general, stability is no better than logarithmic. This means that for perturbations of order ϵ, the error must be expected to be of order $|\log(\epsilon)|^{-\alpha}$ with $\alpha \in (0,1)$. In interior subdomains stability is of Hölder type, meaning that the same perturbation gives rise to an $O(\epsilon^{\alpha})$ error. Computational methods can have, at best, rates that reflect this stability of the continuous problem [2]. To improve on these estimates additional assumptions on the solution are needed. A convenient a priori assumption is that the missing data of the approximate solution is in a δ-neighborhood of a finite N-dimensional space, $\mathcal{G}$, where δ is the smallest distance from the solution to $\mathcal{G}$ in some suitable topology. In this case, it is known that the stability is Lipschitz; that is, the problem has similar stability properties to a well-posed problem, and finite element methods can be designed with optimal convergence up to the data approximation error δ. For linear model problems discretized using piecewise affine finite element methods with mesh parameter h, one can prove the error bound [3],

$$\|u - u_h\|_{H^1(\Omega)} \leq C_N(h + \delta)$$

Here, C_N is a constant that depends on the dimension N of the data set $\mathcal{G}$, the geometry of the available event data, and the smoothness of the exact solution. In particular, C_N typically grows exponentially in N.

Since the size of N must be kept down, there is a clear disadvantage in using the full large dataset. Indeed, for N sufficiently large, the experience data will have no effect. Instead, we wish to identify a lower-dimensional structure in the high-dimensional dataset, a lower-dimensional manifold such that the data resides in a δ-neighborhood of the manifold. For this task, one may use proper orthogonal decomposition in the linear case or neural network autoencoders in the general case.

In the linear case, the data fitting problem reduces to a linear system; however, an ill-conditioned optimization problem has to be solved in the nonlinear case, leading to repeated solutions of linearized finite element systems. To improve the efficiency of this step, we propose to train a network to encode the data to an FE map, giving fast evaluation of finite element approximations without solving the finite element system in the optimization.

The approach is analyzed in the linear case with error estimates for a stabilized FEM using the reduced order model.

Contributions.

- We prove that the inverse problem with boundary data in a finite-dimensional set $\mathcal{G}$ is stable and design a method that reconstructs the solution using the reduced order basis with the same dimension as $\mathcal{G}$. We prove optimal error bounds in the H^1-norm for this method, where the constant of the error bound grows exponentially with the dimension of $\mathcal{G}$.
- In the situation where a large set of perturbed random data, $\mathcal{G}_S$, from the set $\mathcal{G}$ is available, we develop a practical method for the solution of the severely ill-posed inverse problem of unique continuation, leveraging the large dataset to improve the stability properties. In order to handle nonlinearity in the PDE operator and data efficiently we adopt machine learning algorithms. The machine learning techniques are used for the following two subproblems:

  1. Identification of a potential latent space of $\mathcal{G}$ from $\mathcal{G}_S$ to find the smallest possible space for the inverse identification.
  2. Construction of a discrete approximation of the solution map

  $$\phi_u : \mathcal{G} \to H^1(\Omega) \tag{1}$$

  that gives an approximation of the finite element solution to

  $$\mathcal{P}(u) = 0 \quad \text{in } \Omega, \qquad u|_{\partial\Omega} \in \mathcal{G} \tag{2}$$

  where $\mathcal{P}$ is the nonlinear PDE operator in question. The construction is done in a way that is a special case of the approach presented in [4] which in turn is a special case of an even more general approach presented in [5].

- The performance of the combined finite element/machine learning approach is assessed against some academic data assimilation problems.

Previous works. The inverse problem we consider herein is of unique continuation type. There are many types of methods for this type of problem. In the framework we consider the earliest works considered quasi-reversibility [6]. The stabilized method we consider for unique continuation was first proposed in [7–9]. More recent works use residual minimization in dual norm [10–12]. The optimal error estimates for unique continuation with a trace in a finite-dimensional space was first considered for Dirichlet trace in [3] and for Neumann trace in [13]. The idea of combining unique continuation in finite-dimensional space with collective

data was first proposed in [14,15] using linear algebra methods for the compression and direct solution of the linear unique continuation problem. Low rank solvers for the solution of inverse problems have also been designed in [16] using proper orthogonal decomposition.

In recent years, significant advancements have been made in utilizing machine learning for solving PDEs [17–19]. One important aspect is how to suitably and efficiently represent the learned solution [20–23]. An application that comes very natural in the context of neural networks is the derivation of reduced order models [24,25].

These developments are very useful in the context of inverse problems, where they have been utilized in both data- and model-driven inverse problems. In [26] a combination of networks and traditional methods is considered to recover the diffusion coefficient in Poisson's and Burgers' equations. In general, the same is done in [27] with the traditional method being FEM and the equations being elliptic and parabolic. Yet more examples of applying deep learning to this type of problem are given by [28,29]. Anyone interested in the application of deep learning for PDE-solving has undoubtedly encountered Physics-Informed Neural Networks (PINNs) [30] which are also used for inverse problems. Works not involving deep learning but still relevant are [31,32] where projection-based reduced order models for inverse problems are presented. Taking the step to also include machine learning, some of the authors from the previous works give an overview of this mix in [33]. Another overview of using machine learning for inverse problems is given by [34]. In [35], an approach to reduce the error introduced by using operator learning for inverse problems is studied. As a contrast, [36] instead uses machine learning to reduce the error introduced by approximate forward models. Focusing instead on the other side of the computational spectrum, i.e., speed, [37] presents a physics-based deep learning methodology with applications to optimal control. The work [38] presents a modular machine learning framework for solving inverse problems in a latent space. Although using different techniques and approaches, this general description also holds for what we present here.

Comparison between this work and others. To the best of our knowledge, there are no previous works in the literature with a similar theoretical foundation addressing this type of data assimilation problem. Notably, the importance of the finite dimensionality of boundary data for stability, and thus the necessity to reduce the dimension of measured population data as much as possible, has only been considered in [15] using classical methods. Here, we apply autoencoders and operator learning to this problem for the first time. To provide context on how other approaches might perform compared to ours, we note that the stabilized method proposed here yields optimally converging approximations, contingent on the properties of the finite element (FE) space and the stability of the inverse problem. This is not the case for Tikhonov regularized approaches, where discretization is typically applied without further consideration of numerical stability. Bayesian inference methods usually share a similar shortcoming, depending on the choice of prior. We note that in our computational examples, stabilization was not necessary, indicating that the space discretization was sufficiently well-resolved. The use of PINNs in this context leads to a formulation where the strong form of the PDE is minimized. This presents complications, as boundary conditions are generally difficult to impose in network approximations, particularly on the finite-dimensional subspace. Additionally, there appears to be no way to eliminate spurious local minima in the PDE approximation when using PINNs. In our case, since we minimize a convex functional over the finite element space for all parameter values, the space discretization part does not suffer from this defect. Nonetheless, the optimization could converge to local minima when networks approximate the operator, a common shortcoming with network approximation methods.

Concerning the approach to learning the physical model, the method we use is presented in detail in [4], where the focus lies on the method itself as opposed to here, where the focus is on applying it to inverse problems. In [4], a comparison with other machine learning approaches is made so we refer to this work for details and only give a brief characterization here:

- The core concept is to learn a finite element solution operator. The output is thus an approximation of a finite element solution. An advantage of this is that the method can be combined with standard FEM for support and enhancement in both theory and practice.
- A multilayer perceptron (MLP) is used to approximate the solution operator.
- The finite element part enters by using a mesh and basing the loss function on an energy functional that when minimized gives the FE-solution. An alternative is to use the weak residual, which although is more general seems to be more computationally costly.
- The input to the network is a parametrization of problem data, e.g., right-hand side functions and boundary values. The network thus learns a parameterized family of PDE-problems as opposed to only a single problem.
- The method is by default data-free, meaning no input–output data pairs. Instead input is sampled from probability distributions. However, the method allows for the incorporation of data sets as demonstrated here.

None of these individual features is new in physics-based deep learning, but to the best of our knowledge, this specific combination of them has not been studied outside of this work and [4].

Looking through the literature for other works employing deep learning for unique continuation problems, we find two different types. The first type presented in [39] is a data-driven approach for parameterizing both boundary conditions and solutions for flow problems in a cylinder. The velocity distribution is observed in a downstream cross section and the objective is to find a matching inlet profile. Although numerical PDE simulations are used to generate data, this learning approach is physics-free in the contextual sense. The second more common type uses PINNs and seeks the full solution given pointwise observations in a subdomain of the solution domain. In [40] four standard linear problems (Poisson's equation, the heat and wave equations, and Stokes flow) are considered. For a nonlinear problem, see [41] where the 2D Navier–Stokes equations are studied. The work [42] again considers a linear problem, the Helmholtz equation. A drawback of these PINNs-based works is that the neural networks only learn a *single* PDE solution during training. Comparing these works with ours, we see some differences with our approach: First, it is physics-based

in contrast to [39]. Second, it allows for learning an entire class of related PDE solutions as opposed to a single one as in the PINNs-based works. With these points in mind, we think that our deep learning approach to unique continuation problems provides a novelty that can further the field.

Outline. In Section 2, we introduce the model problem and the finite element discretization; in Section 3, we present and prove stability and error estimates for a linear model problem; in Section 4, we develop a machine learning-based approach for solving the inverse problem; in Section 5, we present several numerical examples illustrating the performance of the method for various complexity of the given set of boundary data; and in Section 6 we summarize our findings and discuss future research directions.

Notation.

- We use $\lesssim$ to mean that there is a positive constant in the inequality (typically on the right-hand side).
- For a bounded domain or a set of mesh features D, we denote by $\|\cdot\|_D$ and $(\cdot,\cdot)_D$ the standard $L^2(D)$-norm and inner product, respectively. Some common instances in the text are $D = \omega, \Omega, \partial\Omega, \mathcal{F}_h$.
- We denote by $\|\cdot\|_{\mathbb{R}^N}$ the standard absolute value for vectors in $\mathbb{R}^N$. We note that it should not be confused with the L^2-notation in the previous point. The reason for using the notation $\|\cdot\|_{\mathbb{R}^N}$ is because we use it on the expansion coefficients of functions, thus making expressions involving norms on both functions and their coefficients more general and consistent.
- For a positive-definite bilinear form B we denote the corresponding norm by $\|\cdot\|_B$, i.e., $\|v\|_B^2 := B(v,v)$. An example from the text is $B = m_h$.

2. Inverse problem and finite element method

2.1. Inverse problem

Let Ω be a domain in $\mathbb{R}^d$, $\omega \subset \Omega$ a subdomain, and consider the minimization problem

$$\boxed{\inf_{v\in V} \frac{1}{2}\|u_0 - v\|^2_{L^2(\omega)} \quad \text{subject to} \quad \mathcal{P}(v) = 0 \quad \text{in } \Omega} \tag{3}$$

where $\mathcal{P}(\cdot)$ is a nonlinear second order differential operator and u_0 is an observation of the solution in the subdomain ω. Note that we do not have access to boundary conditions for the partial differential equation; we only know that $\mathcal{P}(u) = 0$ in Ω, and thus, the problem is, in general, ill-posed.

Assume that we have access to a dataset

$$\mathcal{G} \subset H^{1/2}(\partial\Omega) \tag{4}$$

of observed Dirichlet data at the boundary. The dataset $\mathcal{G}$ may have different properties, but here we will assume that it is of the form

$$\mathcal{G} = \left\{ g \in H^{1/2}(\partial\Omega) \mid g = \sum_{i=1}^{N} a_i \varphi_i,\ a_i \in I_i \right\} \tag{5}$$

where I_i are bounded intervals and $\varphi_i \in H^{1/2}(\partial\Omega)$. Below we will also consider access to a finite set $\mathcal{G}_S \subset \mathcal{G}$ of samples from $\mathcal{G}$,

$$\mathcal{G}_S = \{ g_i \mid i \in I_S \} \tag{6}$$

where I_S is some index set.

Including $v|_{\partial\Omega} \in \mathcal{G}$ as a constraint leads to

$$\boxed{\inf_{v\in V} \frac{1}{2}\|u_0 - v\|^2_{L^2(\omega)} \quad \text{subject to} \quad \mathcal{P}(v) = 0 \quad \text{in } \Omega, \quad v|_{\partial\Omega} \in \mathcal{G}} \tag{7}$$

A schematic illustration of a problem of form (7) is given in Fig. 1.

2.2. Finite element method

Let V_h be a finite element space on a quasi-uniform partition $\mathcal{T}_h$ of Ω into shape regular elements with mesh parameter $h \in (0, h_0]$ and assume that there is an interpolation operator $\pi_h : H^1(\Omega) \to V_h$ and a constant such that for all $T \in \mathcal{T}_h$,

$$\|v - \pi_h v\|_{H^m(T)} \lesssim h^{k-m} \|v\|_{H^k(N(T))} \tag{8}$$

for $0 \le m \le k \le p+1$. Here $N(T)$ is the union of all elements that share a node with T.

The finite element discretization of (7) takes the form

$$\boxed{\inf_{v\in V_h} \frac{1}{2}\|u_0 - v\|^2_{L^2(\omega)} \quad \text{subject to} \quad (\mathcal{P}(v), w)_{H^{-1}(\Omega),H^1(\Omega)} = 0 \quad \forall w \in V_{h,0}, \quad v|_{\partial\Omega} \in \pi_h \mathcal{G}} \tag{9}$$

where $V_{h,0} = V_h \cap H^1_0(\Omega)$.

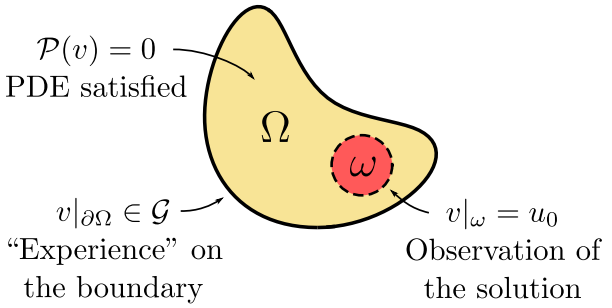


Fig. 1. Schematic view of the minimization problem setup where we seek the $v \in V$ that minimizes the error in the observation of the solution, while under a PDE constraint with boundary conditions according to experience.

3. Analysis for a linear model problem

In this section, we present theoretical results for a linear model problem. We show that the finite dimensionality leads to a well-posed continuous problem, which may, however, have insufficient stability that may cause problems in the corresponding discrete problem. We, therefore, introduce a stabilized formulation that retains the stability properties from the continuous problem, and then we prove error estimates.

3.1. The continuous problem

Consider the linear model problem

$$\mathcal{P}u = 0 \quad \text{in } \Omega, \qquad u = g \quad \text{on } \partial\Omega \tag{10}$$

where $\mathcal{P} = -\Delta$ and

$$g \in \mathcal{G} = \Big\{ \sum_{n=1}^{N} a_n g_n \mid a_n \in \mathbb{R} \Big\} \tag{11}$$

where the functions $\{g_n\}_{n=1}^N$ are linearly independent on $\partial\Omega$. Then with

$$\mathcal{P}\varphi_n = 0 \quad \text{in } \Omega, \qquad \varphi_n = g_n \quad \text{on } \partial\Omega \tag{12}$$

we may express u as the linear combination

$$u = \sum_{n=1}^{N} \widehat{u}_n \varphi_n \tag{13}$$

where $\widehat{u} \in \mathbb{R}^N$ is the coefficient vector. The inverse problem (3) is then equivalent to computing the $L^2(\omega)$-projection of u_0 on $\mathcal{U}_N = \text{span}\{\varphi_n\}_{n=1}^N$,

$$\boxed{u_N \in \mathcal{U}_N : \qquad (u_N, w)_\omega = (u_0, w)_\omega \qquad \forall w \in \mathcal{U}_N} \tag{14}$$

This is a finite-dimensional problem, and therefore, existence follows from uniqueness. To prove uniqueness, consider two solutions u_1 and u_2, we then have

$$(u_1 - u_2, w)_\omega = 0 \qquad \forall w \in \mathcal{U}_N \tag{15}$$

and taking $w = u_1 - u_2$ gives

$$\|u_1 - u_2\|_\omega = 0 \tag{16}$$

By unique continuation for harmonic functions, we conclude that $u_1 - u_2$ is zero on the boundary and therefore $u_1 = u_2$ since the set $\{g_n\}_{n=1}^N$ is linearly independent on $\partial\Omega$. It follows that $\{\varphi_n\}_{n=1}^N$ is linearly independent on ω and by finite dimensionality, there is a constant (it is $\lambda_{min}^{-1/2}$), such that

$$\boxed{\|\widehat{v}\|_{\mathbb{R}^N} \lesssim \|v\|_\omega \qquad v \in \mathcal{U}_N} \tag{17}$$

Note, however, that the constant may be huge, reflecting the often near ill-posed nature of an inverse problem.

3.2. The discrete problem

In practice, only an approximation of the basis $\{\varphi_n\}_{n=1}^N$ is available, since we observe data on the boundary and must solve for an approximate basis. Assuming that we compute an approximate basis $\{\varphi_{n,h}\}_{n=1}^N$ using Nitsche's method with continuous piecewise linears V_h, defined on a triangulation $\mathcal{T}_h$ of Ω,

$$\boxed{\varphi_{n,h} \in V_h : \qquad a_h(\varphi_{n,h}, v) = l_{h,\varphi_n}(v) \qquad \forall v \in V_h} \tag{18}$$

where the forms are defined by

$$a_h(v,w) = (\nabla v, \nabla w)_\Omega - (\nabla_n v, w)_{\partial\Omega} - (v, \nabla_n w)_{\partial\Omega} + \beta h^{-1}(v,w)_{\partial\Omega} \tag{19}$$

$$l_{h,g}(v) = -(g, \nabla_n v)_{\partial\Omega} + \beta h^{-1}(g,v)_{\partial\Omega} \tag{20}$$

with g the given Dirichlet data on $\partial\Omega$, we have the error estimates

$$\|\varphi_n - \varphi_{n,h}\|_\Omega + h\|\nabla(\varphi_n - \varphi_{n,h})\|_\Omega \lesssim h^2\|\varphi_n\|_{H^2(\Omega)} \lesssim h^2\|g_n\|_{H^{3/2}(\partial\Omega)} \tag{21}$$

provided the regularity estimate $\|\varphi_n\|_{H^2(\Omega)} \lesssim \|g_n\|_{H^{3/2}(\partial\Omega)}$ holds, which is the case for convex or smooth domains.

Next, we define the operators

$$I : \mathbb{R}^N \ni \widehat{v} \mapsto \sum_{n=1}^N \widehat{v}_n \varphi_n \in \mathcal{U}_N \tag{22}$$

$$I_h : \mathbb{R}^N \ni \widehat{v} \mapsto \sum_{n=1}^N \widehat{v}_n \varphi_{n,h} \in \mathcal{U}_{N,h} \tag{23}$$

to represent linear combinations given coefficient vectors, where $\mathcal{U}_{N,h} = \operatorname{span}\{\varphi_{n,h}\}_{n=1}^N$. By composing I and I_h with the coefficient extraction operator $\widehat{\cdot}$, we note that $v = I\widehat{v}$ for $v \in \mathcal{U}_N$ and $v = I_h\widehat{v}$ for $\mathcal{U}_{N,h}$. We also note that $I_h\widehat{v}$ is the Galerkin approximation defined by (18) of $v = I\widehat{v}$, since $\varphi_{n,h}$ is the Galerkin approximation of φ_n for $n = 1, \dots, N$, and we have the error estimate

$$\|(I - I_h)\widehat{v}\|_\Omega + h\|\nabla(I - I_h)\widehat{v}\|_\Omega + h^{1/2}\|(I - I_h)\widehat{v}\|_{\partial\Omega} \lesssim h^2\|\widehat{v}\|_{\mathbb{R}^N} \tag{24}$$

The estimate (24) follows directly using the Cauchy–Schwarz inequality and the error estimates (21) for the approximate basis

$$\|\nabla^m(v - I_h\widehat{v})\|_\Omega^2 = \Big(\sum_{n=1}^N v_n^2\Big)\Big(\sum_{n=1}^N \|\nabla^m(\varphi_n - \varphi_{n,h})\|_\Omega^2\Big) \tag{25}$$

$$\lesssim h^{2(2-m)}\Big(\sum_{n=1}^N v_n^2\Big)\Big(\sum_{n=1}^N \|g_n\|_{H^{3/2}(\partial\Omega)}^2\Big) \tag{26}$$

with $m = 0, 1$.

Now if we proceed as in (14) with the modes φ_n replaced by the approximate modes $\varphi_{n,h}$, we cannot directly use the same argument as in the continuous case to show that there is a unique solution since the discrete method does not possess the unique continuation property, and it does not appear easy to quantify how small the mesh size must be to guarantee that the bound (17) holds on $\mathcal{U}_{N,h}$.

To quantify the discrete stability, note that the constant in (17) is characterized by the Rayleigh quotient

$$\lambda_{\min} = \min_{\widehat{v}\in\mathbb{R}^N} \frac{\|I\widehat{v}\|_\omega^2}{\|\widehat{v}\|_{\mathbb{R}^N}^2} \tag{27}$$

and for the corresponding discrete estimate

$$\|\widehat{v}\|_{\mathbb{R}^N}^2 \lesssim \|v\|_\omega^2 \tag{28}$$

we instead have the constant

$$\lambda_{h,\min} = \min_{\widehat{v}\in\mathbb{R}^N} \frac{\|I_h\widehat{v}\|_\omega^2}{\|\widehat{v}\|_{\mathbb{R}^N}^2} \tag{29}$$

Using the triangle inequality and the error estimate (21) we have

$$\|I_h\widehat{v}\|_\omega \geq \|I\widehat{v}\|_\omega - \|(I_h - I)\widehat{v}\|_\omega \geq \|I\widehat{v}\|_\omega - ch^2\|\widehat{v}\|_{\mathbb{R}^N} \tag{30}$$

and thus we may conclude that

$$\lambda_{\min,h} \geq (\lambda_{\min}^{1/2} - ch^2)^2 \geq c\lambda_{\min} \tag{31}$$

for $h < h_0$ with h_0 small enough. Thus for h small enough the discrete bound (28) holds but we note that the precise characterization of how small h has to be appears difficult.

To handle this difficulty, let us instead consider the stabilized form

$$\boxed{m_h(v,w) = (v,w)_\omega + s_{h,\partial\Omega}(v,w) + s_h(v,w)} \tag{32}$$

Here

$$s_{h,\partial\Omega}(v,w) = h^{-1}(v - I\widehat{v}, w - I\widehat{w})_{\partial\Omega} + h(\nabla_T(v - I_h\widehat{v}), \nabla_T(w - I\widehat{w}))_{\partial\Omega} \tag{33}$$

where $\nabla_T = (I_{\mathrm{id}} - n \otimes n)\nabla$ is the tangential derivative on $\partial\Omega$ with n denoting the unit normal to $\partial\Omega$ and I_{id} is the d-dimensional identity matrix. The form s_h is the standard normal gradient jump penalty term

$$s_h(v,w) = \sum_{F\in\mathcal{F}_h} h([\nabla v],[\nabla w])_F \tag{34}$$

where $\mathcal{F}_h$ is the interior faces in the mesh $\mathcal{T}_h$. The role of the form $s_{h,\partial\Omega}$ is to give control of the distance of the approximation to the finite dimensional set $\mathcal{G}$ in the $H^{1/2}(\partial\Omega)$-norm. In principle the form

$$s_{\partial\Omega}(v,w) = (v - I\widehat{v}, w - I\widehat{w})_{H^{1/2}(\partial\Omega)} \tag{35}$$

could be used directly, but to obtain a stabilization term that is easier to handle in practice we note that by the Galigliardo–Nirenberg inequality, $\|v\|_{H^{1/2}(\partial\Omega)} \lesssim \|v\|^{1/2}_{L^2(\partial\Omega)}\|v\|^{1/2}_{H^1(\partial\Omega)}$, there holds $s_{\partial\Omega}(v,v) \lesssim s_{h,\partial\Omega}(v,v)$, which is sufficient for stability.

3.3. Error estimates

Our first result is that the additional stabilization terms in m_h ensure that we have stability for the discrete problem similar to (17) that holds for the exact problem.

Lemma 3.1. *Let m_h be defined by (32). Then, there is a constant, depending on N but not h, such that*

$$\boxed{\|\widehat{v}\|_{\mathbb{R}^N} \lesssim \|v\|_{m_h} \qquad v \in \mathcal{U}_{N,h}} \tag{36}$$

Proof. For $v \in \mathcal{U}_{N,h}$ we get by using the stability (17) on $\mathcal{U}_N$, adding and subtracting $I_h\widehat{v}$, and employing the triangle inequality,

$$\|\widehat{v}\|_{\mathbb{R}^N} \lesssim \|I\widehat{v}\|^2_\omega \lesssim \|I_h\widehat{v}\|^2_\omega + \|(I - I_h)\widehat{v}\|^2_\omega = \|v\|^2_\omega + \|(I - I_h)\widehat{v}\|^2_\omega \tag{37}$$

where we finally used the identity $I_h\widehat{v} = v$, which holds since $v \in \mathcal{U}_{N,h}$. Next, we bound the second term using the stabilizing terms in m_h. To that end, we observe that we have the orthogonality

$$a_h((I - I_h)\widehat{v}, w) = 0 \qquad \forall w \in V_h \tag{38}$$

since the discrete basis is, a Galerkin projection (18) of the exact basis with respect to the Nitsche form a_h. Using the dual problem

$$\mathcal{P}\phi = \psi \quad \text{in } \Omega, \qquad \phi = 0 \quad \text{on } \partial\Omega \tag{39}$$

we obtain by partial integration followed by Galerkin orthogonality

$$((I - I_h)\widehat{v}, \psi)_\Omega = ((I - I_h)\widehat{v}, \mathcal{P}\phi)_\Omega = a_h((I - I_h)\widehat{v}, \phi) = a_h((I - I_h)\widehat{v}, \phi - \pi_h\phi) \tag{40}$$

where $\pi_h : H^1(\Omega) \to V_h$ is the interpolation operator. Performing another partial integration, we get

$$a_h((I - I_h)\widehat{v}, \phi - \pi_h\phi) \tag{41}$$

$$= ([\nabla_n(I - I_h)\widehat{v}], \phi - \pi_h\phi)_{\mathcal{F}_h} - ((I - I_h)\widehat{v}, \nabla_n(\phi - \pi_h\phi))_{\partial\Omega} \tag{42}$$

$$\lesssim (h^{3/2}\|[\nabla_n(I - I_h)\widehat{v}]\|_{\mathcal{F}_h} + h^{1/2}\|(I - I_h)\widehat{v}\|_{\partial\Omega})\|\phi\|_{H^2(\Omega)} \tag{43}$$

where we used the standard trace inequality $\|w\|^2_{\partial T} \lesssim h^{-1}\|w\|^2_T + h\|\nabla w\|^2_T$ for $w \in H^1(T)$ on an element $T \in \mathcal{T}_h$. Finally, using the elliptic regularity $\|\phi\|_{H^2(\Omega)} \lesssim \|\psi\|_\Omega$, combining the results, and taking $\psi = (I - I_h)\widehat{v}$, we get

$$\|(I - I_h)\widehat{v}\|_\Omega \lesssim h^{3/2}\|[\nabla_n(I - I_h)\widehat{v}]\|_{\mathcal{F}_h} + h^{1/2}\|(I - I_h)\widehat{v}\|_{\partial\Omega} \tag{44}$$

$$\lesssim h(\|v\|_{s_h} + \|v\|_{s_{h,\partial\Omega}}) \tag{45}$$

which combined with (37) directly gives the desired estimate. □

We define the stabilized projection,

$$\boxed{u_{N,h} \in \mathcal{U}_{N,h} : \qquad m_h(u_{N,h}, v) = (u_0, v)_\omega \qquad \forall v \in \mathcal{U}_{N,h}} \tag{46}$$

We then have the following error estimate for the stabilized projection with approximate basis functions.

Proposition 3.1. *Let $u_N \in \mathcal{U}_N$ be defined by (14) and $u_{N,h} \in \mathcal{U}_{N,h}$ be defined by (46). Then, there is a constant such that,*

$$\boxed{\|u_N - u_{N,h}\|_{m_h} \lesssim h\|u_0\|_\omega} \tag{47}$$

Proof of Proposition 3.1. Using the triangle inequality

$$\|u_N - u_{N,h}\|_{m_h} \leq \|u_N - I_h\widehat{u}_N\|_{m_h} + \|I_h\widehat{u}_N - u_{N,h}\|_{m_h} \tag{48}$$

Here the first term can be directly estimated using (24),

$$\|u_N - I_h\widehat{u}_N\|_{m_h} = \|(I - I_h)\widehat{u}_N\|_{m_h} \lesssim h\|\widehat{u}_N\|_{\mathbb{R}^N} \lesssim h\|u_0\|_\omega \tag{49}$$

since for $v \in \mathcal{U}_N$ we have $v = I\widehat{v}$ and using the stability estimate (17) followed by (14) we get

$$\|\widehat{u}_N\|_{\mathbb{R}^N} \lesssim \|u_N\|_\omega \lesssim \|u_0\|_\omega \tag{50}$$

For the second term, we first note that the stabilization terms s_h and $s_{h,\partial\Omega}$ vanish on $\mathcal{U}_N$ so that

$$m_h(u_N, v) = (u_N, v)_\omega \qquad \forall v \in \mathcal{U}_{N,h} \tag{51}$$

Then by subtracting and adding u_N in the first argument to m_h, we have for any $v \in \mathcal{U}_{N,h}$,

$$m_h(I_h\widehat{u}_N - u_{N,h}, v) \tag{52}$$
$$= m_h(I_h\widehat{u}_N - u_N, v) + m_h(u_N, v) - m_h(u_{N,h}, v) \tag{53}$$
$$= m_h(I_h\widehat{u}_N - u_N, v) + (u_N, v)_\omega - (u_0, v)_\omega \tag{54}$$
$$= m_h(I_h\widehat{u}_N - u_N, v) + (u_N, v - I\widehat{v})_\omega - (u_0, v - I\widehat{v})_\omega \tag{55}$$

where we used (46) and (51) on the second and third terms in (53), respectively, and the definition (14) of u_N to subtract $I\widehat{v} \in \mathcal{U}_N$ in (54). Employing continuity of the involved forms, we get

$$m_h(I_h\widehat{u}_N - u_{N,h}, v) \tag{56}$$
$$\lesssim \|I_h\widehat{u}_N - u_N\|_{m_h}\|v\|_{m_h} + \|u_N\|_\omega\|v - I\widehat{v}\|_\omega + \|u_0\|_\omega\|v - I\widehat{v}\|_\omega \tag{57}$$
$$\lesssim h\|\widehat{u}_N\|_{\mathbb{R}^N}\|v\|_{m_h} + \|u_N\|_\omega h^2\|\widehat{v}\|_{\mathbb{R}^N} + \|u_0\|_\omega h^2\|\widehat{v}\|_{\mathbb{R}^N} \tag{58}$$
$$\lesssim \underbrace{h\,(\|\widehat{u}_N\|_{\mathbb{R}^N} + h\|u_N\|_\omega + h\|u_0\|_\omega)}_{\lesssim\|u_0\|_\omega}\|v\|_{m_h} \tag{59}$$

where we used the stability (36) and the bounds $\|\widehat{u}_N\|_{\mathbb{R}^N} \lesssim \|u_N\|_\omega$ and $\|u_N\|_\omega \lesssim \|u_0\|_\omega$. Thus by taking $v = I_h u_N - u_{N,h}$, we conclude that

$$\|I_h u_N - u_{N,h}\|_{m_h} \lesssim h\|u_0\|_\omega \tag{60}$$

which combined with (48) and (49) concludes the proof. □

We finally prove the following global result,

Proposition 3.2. *Let $u_N \in \mathcal{U}_N$ be defined by (14) and $u_{N,h} \in \mathcal{U}_{N,h}$ be defined by (46). Then, there is a constant depending on higher order Sobolev spaces of g such that,*

$$\boxed{\|u_N - u_{N,h}\|_{H^1(\Omega)} \lesssim h\|u_0\|_\omega} \tag{61}$$

Proof. With $e = u_N - u_{N,h}$, we have

$$\|e\|_{H^1(\Omega)} \lesssim \|e - I\widehat{e}\|_{H^1(\Omega)} + \|I\widehat{e}\|_{H^1(\Omega)} \tag{62}$$

By norm equivalence on discrete spaces we have

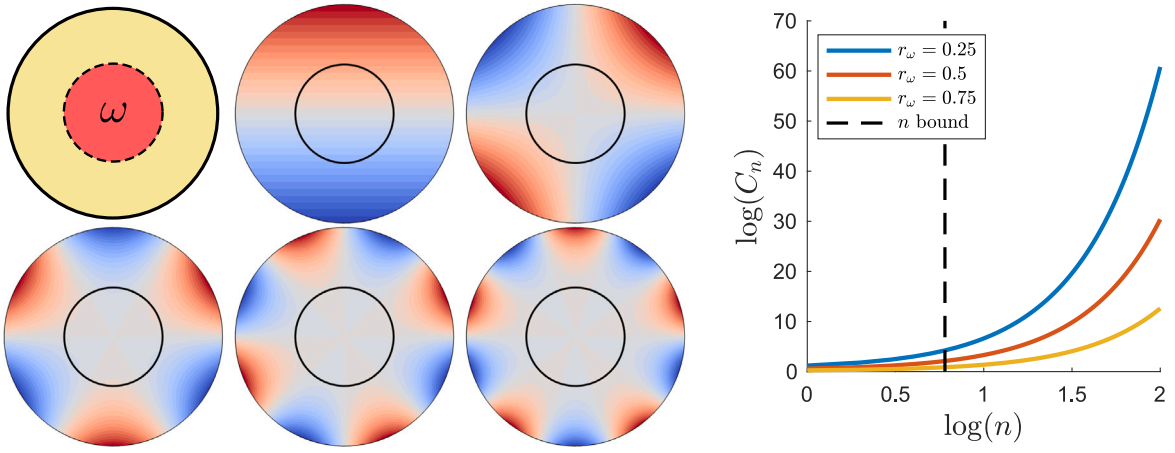
$$\|I\widehat{e}\|_{H^1(\Omega)} \lesssim \|\widehat{e}\|_{\mathbb{R}^N} \tag{63}$$

Since $I\widehat{e} \in \mathcal{U}_N$ there holds using (17),

$$\|I\widehat{e}\|_{H^1(\Omega)} \lesssim \|I\widehat{e}\|_\omega \leq \|e\|_\omega + \|u_{N,h} - I\widehat{u}_{N,h}\|_\omega \tag{64}$$

By Proposition 3.1 there holds

$$\|e\|_\omega \lesssim h\|u_0\|_\omega \tag{65}$$



(a) Domain and first five modes of $\varphi_{2n+1} = r^n \sin(n\theta)$ (b) Scaling of stability constants

Fig. 2. Illustrations for the analytical example with exponential growth of the stability constant. In (a), we show the unit disc domain containing a subdomain ω, in the form of a centered disc of radius r_ω. Looking at the first five non-zero modes $\varphi_{2n+1} = r^n \sin(n\theta)$ from the expansion Eq. (69) we see that these modes rapidly becomes very small within ω, making the problem of retrieving the coefficient values in the expansion based on the solution within ω increasingly ill-posed for larger n. In (b), we illustrate how the constants in the stability estimate Eq. (70) scales exponentially with n for different values of the radius r_ω. By utilizing observations, we may conclude an upper bound on n, which in turn puts an upper bound on the size of the stability constant.

For the second term we have using (24),

$$\|u_{N,h} - I\widehat{u}_{N,h}\|_\omega \leq \|(I_h - I)\widehat{u}_{N,h}\|_\Omega \lesssim h^2 \|\widehat{u}_{N,h}\|_{\mathbb{R}^N} \lesssim h^2 \|u_{N,h}\|_{m_h} \tag{66}$$

Similarly we have

$$\|e - I\widehat{e}\|_{H^1(\Omega)} = \|u_{N,h} - I\widehat{u}_{N,h}\|_{H^1(\Omega)} \lesssim h \|\widehat{u}_{N,h}\|_{\mathbb{R}^N} \lesssim h \|u_{N,h}\|_{m_h} \tag{67}$$

We conclude the proof by using the bound

$$\|u_{N,h}\|_{m_h} \lesssim \|u_0\|_\omega \quad \square \tag{68}$$

Remark 3.1. Observe that the stabilization is never explicitly used in order to obtain error estimates. Indeed its only role is to ensure the bound $\|\widehat{u}_{N,h}\|_{\mathbb{R}^N} \lesssim \|u_{N,h}\|_{m_h}$ without condition on the mesh.

Example (Exponential growth of the stability constant). Let Ω be the unit disc. Then the solutions to $-\Delta u = 0$ are of the form

$$u(r,\theta) = \sum_{n=0}^{\infty} a_{2n} \underbrace{r^n \cos(n\theta)}_{=\varphi_{2n}} + a_{2n+1} \underbrace{r^n \sin(n\theta)}_{=\varphi_{2n+1}} \tag{69}$$

where (r,θ) are the standard polar coordinates. Let ω be the disc centered at the origin with radius r_ω. We note that when n becomes large, the modes become small in the disc ω, and therefore, the inverse problem becomes increasingly ill-posed, see Fig. 2(a). For instance, the constant in an estimate of the type

$$\|\varphi_{2n+m}\|_\Omega \leq C_n \|\varphi_{2n+m}\|_\omega, \qquad m = 0, 1 \tag{70}$$

scales like

$$C_n = r_\omega^{-(n+1)} \tag{71}$$

and thus becomes arbitrarily large when n becomes large. But, if we, from observations, can conclude that only modes with $n < n_g$ for some n_g are present, then the stability is controlled, see Fig. 2(b). Note also that the stability is directly related to where the disc ω is placed. If it is located close to the boundary, the stability improves.

4. Methods based on machine learning

Overview. We develop a method for efficiently solving the inverse problem (7) with access to sampled data $\mathcal{G}_S$ using machine learning techniques. The main approach is:

- Construct a parametrization of the data set by first approximately expanding the samples in a finite series of functions, for instance, using Proper Orthogonal Decomposition, and secondly using an autoencoder to find a possible nonlinear low-dimensional structure in the expansion coefficients.
- Use operator learning to construct an approximation of the finite element solution operator that maps the expansion coefficients to the finite element solution.
- Composing the decoder, which maps the latent space to expansion coefficients, with the solution network, we obtain a differentiable mapping that can be used to solve the inverse problem in a lower-dimensional space.

4.1. Processing the boundary data

We combine linear and nonlinear dimensionality reduction techniques by first using PCA on the data to get a POD-basis and then using autoencoders on the POD-coefficients for further reduction. Such combinations are not uncommon, see, e.g., [43], and there are several reasons why we do this: In general, an initial linear reduction may function as a relatively cheap preprocessing step to aid a subsequent nonlinear reduction that typically is more expensive. More specifically, we consider methodology for progressing from a fully linear problem (linear PDE, linear data) to a fully nonlinear one (nonlinear PDE, nonlinear data), hence both linear and nonlinear techniques that can be combined are required. The reason for using POD is that it very cheaply and naturally gives a basis that can be used for both the linear and nonlinear PDE-solving techniques considered here. The reason for using autoencoders is simply because they have the same general network architecture already used for the operator networks which also makes combinations with them seem natural.

Proper orthogonal decomposition. To assimilate the data set $\mathcal{G}$ in a method for solving the extension problem, we seek to construct a differentiable parametrization of $\mathcal{G}$. To that end, we first use Proper Orthogonal Decomposition (POD) to represent the data in a POD basis $\{\varphi_n\}_{n=1}^N$,

$$g = \sum_{n=1}^{N} \widehat{g}_n \varphi_n \tag{72}$$

where $\widehat{g}_n = (g, \varphi_n)_{\mathbb{R}^N}$. We introduce the mapping

$$\phi_{\mathrm{POD},N} : \mathcal{G} \ni g \mapsto \widehat{g} \in G_N \subset \mathbb{R}^N \tag{73}$$

where $G_N = \phi_{\mathrm{POD},N}(\mathcal{G})$. We also need the reconstruction operator

$$\phi^{\dagger}_{\mathrm{POD},N} : \mathbb{R}^N \ni a \mapsto \sum_{n=1}^{N} a_n \varphi_n \in \mathcal{G} \tag{74}$$

Letting I_N denote the identity operator on $\mathbb{R}^N$, we have

$$\phi_{\mathrm{POD},N} \circ \phi^{\dagger}_{\mathrm{POD},N} = I_N \tag{75}$$

and we note that the operator $\phi_{\mathrm{POD},N}$ is invertible and differentiable.

Autoencoder. Next, we seek to find a possible nonlinear low-dimensional structure in the POD coefficients using an autoencoder $\phi_{\mathrm{de}} \circ \phi_{\mathrm{en}}$

$$\boxed{G_N \xrightarrow{\phi_{\mathrm{en}}} Z \xrightarrow{\phi_{\mathrm{de}}} G_N} \tag{76}$$

where ϕ_{en} denotes the encoder map and ϕ_{de} the decoder map. Letting $\mathbb{E}$ denote the expectation operator and P an arbitrary probability distribution, the autoencoder is trained to minimize the loss

$$\mathbb{E}_{\widehat{g}\sim P}\left[\| \widehat{g} - (\phi_{\mathrm{de}} \circ \phi_{\mathrm{en}})(\widehat{g}) \|^2_{\mathbb{R}^N} \right] \tag{77}$$

See Fig. 3(a) for a schematic illustration. Here $Z \sim \mathbb{R}^{n_Z}$ is the latent space with dimension $n_Z < N$. If there is a low-dimensional structure, we may often take n_Z significantly lower than N.

4.2. Operator learning

The operator learning approach taken here is the same as in [4] which is a special case of a more general method presented in [5]. We discretize the PDE problem using finite elements and train a network

$$\phi_{u,N,h} : G_N \to V_h \subset H^1(\Omega) \tag{78}$$

which approximates the finite element solution to

$$\mathcal{P}(u) = 0 \quad \text{in } \Omega, \qquad u = \phi^{\dagger}_{\mathrm{POD},N}(\widehat{g}) \quad \text{on } \partial\Omega \tag{79}$$

see Fig. 3(b). The output of the network is the finite element degrees of freedom (DoFs). For the training of the network we use the energy functional E corresponding to the differential operator $\mathcal{P}$ as the foundation for the loss function. Again, letting $\mathbb{E}$ denote the expectation operator and P an arbitrary probability distribution, the loss function that we minimize during training is

$$\mathbb{E}_{\hat{g}\sim P}\left[E(\phi_{u,N,h}(\hat{g}))\right] \tag{80}$$

If there is no corresponding energy functional, one can instead minimize the residual of the finite element problem. It should be noted though, that assembling the residual instead of the energy has a greater computational cost and that the residual is not as easily and naturally decomposed into its local contributions as the energy. For technical details about network architecture and training used in this work, we refer to Section 5.2.

4.3. Inverse problem

Finally, composing the maps, we get a solution operator

$$\boxed{Z \xrightarrow{\phi_{\text{de}}} G_N \xrightarrow{\phi_{u,N,h}} V_h} \tag{81}$$

that maps the latent space into approximate finite element solutions to the partial differential equation

$$\mathcal{P}((\phi_{u,N,h}\circ\phi_g)(z)) = 0 \tag{82}$$

see Fig. 3(c).

This mapping is differentiable and can be directly used to rewrite the optimization problem as an unconstrained problem in the form

$$\boxed{\inf_{z\in Z}\frac{1}{2}\|u_0 - (\phi_{u,N,h}\circ\phi_{\text{de}})(z)\|^2_{L^2(\omega)}} \tag{83}$$

where we note that the constraint is fulfilled by construction.

5. Examples

We consider three examples of the inverse minimization problem ordered in increased nonlinearity. The first is a fully linear case with a linear differential operator and linear boundary data. In the second example, we consider a nonlinear operator with linear data. The final example is a fully nonlinear case with both operator and data being nonlinear. The examples demonstrate how each introduced nonlinearity may be treated with machine learning methods.

The geometry is the same in all the examples. We take the solution domain $\Omega := (-0.5, 0.5)^2 \subset \mathbb{R}^2$ and the reference domain $\omega \subset \Omega$ to be the u-shaped domain defined by

$$\omega := \{(x, y) \in \mathbb{R}^2 \mid x < 0.25 \wedge (x < -0.25 \vee y < -0.25 \vee y > 0.25)\} \tag{84}$$

see Fig. 4.

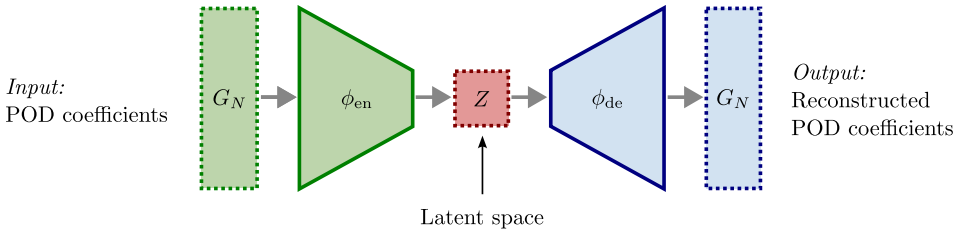
When solving the inverse problems in practice, we use data $u_0 \in V_h$. We also minimize the mean squared error (MSE) over the DoFs belonging to ω instead of the squared $L^2(\omega)$ norm of the error, which is valid since they are equivalent on V_h from the Rayleigh quotient. The only "stabilization" we use for the inverse problem is that the boundary data is finite-dimensional and that this dimension together with the mesh size h both are small enough. We point out that no additional stabilization, such as including penalty terms, is used. The criterion for when a minimization process is considered to have converged is based on the change of significant digits of the MSE. For the fully linear problem we consider it converged when at least three significant digits remain constant, and for all the nonlinear problems when at least two significant digits remain constant. This is in turn based on when both the optimization variables and the visual representation of the output do not seem to change anymore and has been obtained by testing.

The implementation used for the examples is based on the code presented in [5] which is publicly available at https://github.com/nmwsharp/neural-physics-subspaces. All inverse problem minimizations have been performed with the Adam optimizer with a step size = 0.1 on an Apple M1 CPU. The GPU computations were performed on the Alvis cluster provided by NAISS (See Acknowledgments).

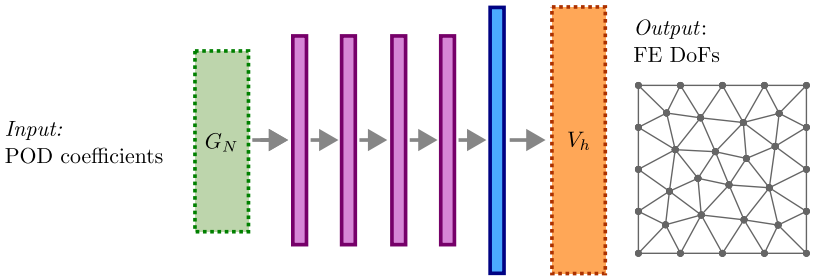
5.1. Linear operator with linear data

We start with the fully linear case which we will build upon in the later examples. To construct a linear synthetic data set $\mathcal{G}$, we may pick a set of functions $\{\varphi_j\}_{j\in J} \subset H^{1/2}(\partial\Omega)$ where J is some index set, and consider

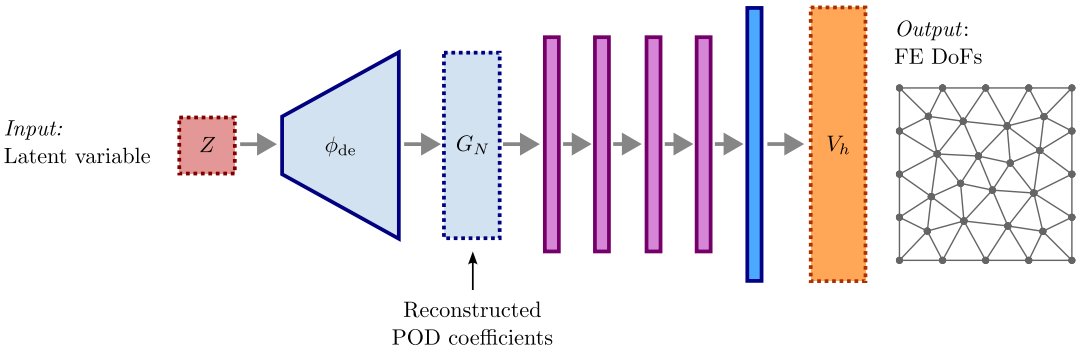
$$\mathcal{G} = \left\{ g_i = \sum_{j\in J} \xi_j \varphi_j \right\} \tag{85}$$



(a) Autoencoder network $\phi_{\text{de}} \circ \phi_{\text{en}}$



(b) Operator network $\phi_{u,N,h}$ with four nonlinear layers (purple) and one linear output layer (blue)



(c) Composition of decoder and operator network $\phi_{u,N,h} \circ \phi_{\text{de}}$

Fig. 3. Overview of networks utilized in methods based on machine learning. The autoencoder network in (a) is used for identifying a low-dimensional structure in the dataset $\mathcal{G}$. The operator network in (b) is trained to approximate the solution to the PDE, given input boundary data. The composition of the decoder part of the autoencoder and the operator network in (c) is used for solving the inverse problem in the low-dimensional latent space.

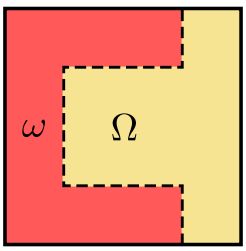


Fig. 4. Domain used in all numerical examples with the subdomain ω indicated.

where $\xi \in [s_i, t_i] \subset \mathbb{R}$. Note that we require the boundary data to be bounded. Alternatively, we can also consider taking the convex hull of the basis functions $\{\varphi_j\}_{j\in J}$, which corresponds to requiring that

$$\sum_j \xi_j = 1, \quad \xi_j \geq 0 \tag{86}$$

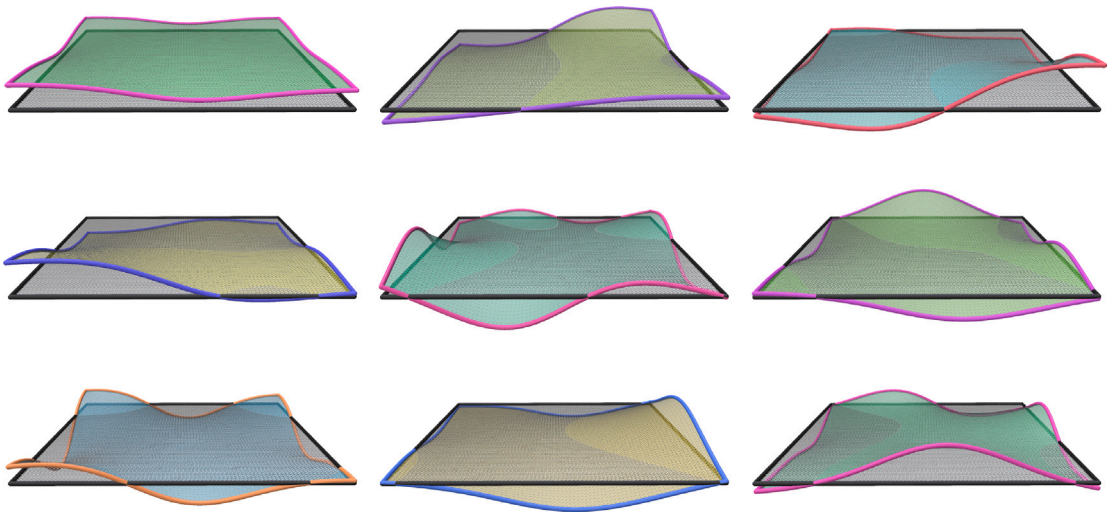


Fig. 5. FEM interior basis functions with their corresponding POD boundary basis functions on a structured uniform 82×82 triangular mesh of the unit square.

Given nodal samples of such functions, we may apply principal component analysis (PCA) to estimate a set of basis functions and use them to parametrize the data set. More precisely, assume we observe the boundary data in the nodal points at the boundary. Let X be the matrix where each observation forms a row. Then, computing the eigenvectors to the symmetric matrix $X^T X$ provides estimates of the basis.

Here, we consider two-dimensional examples. We let Ω be the unit square centered at the origin and generate four structured uniform triangular meshes of varying sizes: 10×10, 28×28, 82×82, and 244×244. The synthetic data set of boundary nodal values is in turn generated from the perturbed truncated Fourier series

$$g(x) = (\hat{g}_0 + \delta_0) + \sum_{n=1}^{(N-1)/2} (\hat{g}_{2n-1} + \delta_{2n-1}) \sin(2n\pi x/l) + (\hat{g}_{2n} + \delta_{2n}) \cos(2n\pi x/l) \tag{87}$$

where l is the circumference of Ω and x is the counter-clockwise distance along the boundary starting from the point where the boundary crosses the first coordinate axis. We sample unperturbed coefficients $\hat{g}_j \sim \mathcal{U}(-1, 1)$ and perturbations $\delta_j \sim \mathcal{N}(0, 0.0225)$. For each of the four meshes, we consider two values of the number of coefficients used to describe the boundary conditions; $N = 9$ and $N = 21$. We generate 1000 functions of the type (87) for each of the eight cases. Then, for every case, we compute a POD basis $\{\varphi_{\mathrm{POD},j}\}_{j\in J}$ for the boundary using PCA on the data set. Unsurprisingly, the number of significant singular values turns out to be the number N used in each case.

We use the truncated POD boundary basis corresponding to significant singular values to compute and interior basis. We do this by solving Laplace's equation with FEM. We take the discrete space V_h to simply be the space of piecewise linear finite elements on the triangle mesh considered. The FEM interior basis $\{\varphi_{\mathrm{FEM},n}\}_{n=1}^N$ is computed by: For each $n = 1, \dots, N$, find $\varphi_{\mathrm{FEM},n} \in V_h$ such that $\varphi_{\mathrm{FEM},n}|_{\partial\Omega} = \varphi_{\mathrm{POD},n}$ and

$$(\nabla\varphi_{\mathrm{FEM},n}, \nabla v)_\Omega = 0 \quad \forall v \in V_h \tag{88}$$

In Fig. 5, the significant POD boundary basis functions, together with their corresponding FEM interior basis functions, are presented for the case with $N = 9$ and the 82×82 mesh.

We may now use the fact that Laplace's equation is linear to superpose the FEM interior basis functions in a linear combination.

$$u_{h,\mathrm{lin},N} = \sum_{n=1}^{N} c_n \varphi_{\mathrm{FEM},n} \tag{89}$$

This enables us to solve a linear inverse minimization problem over the coefficients $(c_1, \dots, c_N) \in \mathbb{R}^N$ in the linear combination. We present a demonstration of this process for the case with $N = 9$ and the 82×82 mesh in Fig. 6. There it can clearly be observed that the finite element solution given by the linear combination approaches the noisy data and the reference solution as the optimization progresses.

5.2. Nonlinear operator with linear data

We again consider the linear data sets from the previous section, but here together with a *nonlinear* differential operator. Because of the nonlinearity, we cannot use the FEM interior basis and the superposition principle as in the fully linear case. Instead, we use a neural network to approximate the solution operator, i.e., the inverse of the nonlinear differential operator. The solution is still

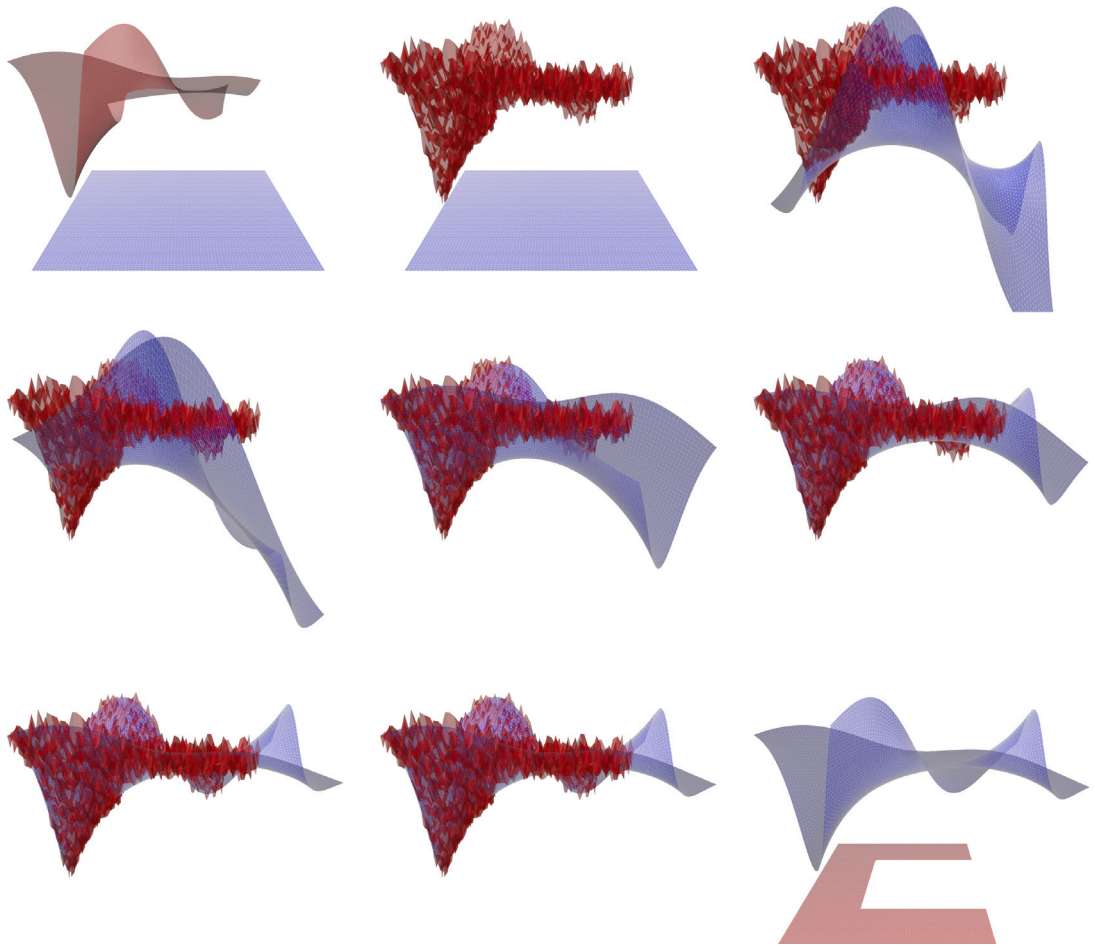


Fig. 6. Optimization process over a 9-dimensional *coefficient space* for a *linear* inverse problem with noisy data. Here, the FEM interior basis functions in Fig. 5 are used. The unperturbed data is shown in the first frame. The second frame is the same as the first but with added noise sampled from $\mathcal{U}(-0.05, 0.05)$. The last frame shows ω and the reference solution used for the data which was obtained by taking $c_1 = 10, c_9 = 3.023$, and all other c_n's $= 0$ in (89). The penultimate frame shows the optimization's MSE-converged reconstruction of the reference solution. The MSE converged after 861 iterations with the Adam optimizer with a step size $= 0.1$. This took 10.3 s on an Apple M1 CPU. The MSE's between the reference solution and the converged reconstruction are: on ω (used in optimization), $\text{MSE}_{\omega} = 8.28\text{e}{-4}$; on the convex hull of ω, $\text{MSE}_{\text{co}(\omega)} = 9.12\text{e}{-4}$; and on its complement, $\text{MSE}_{\text{co}(\omega)^c} = 2.66\text{e}{-3}$.

in the form of a finite element function, so the output of the network gives an approximation of the finite element solution. The input to the network is the POD coefficients $(p_1, \dots, p_N)$ corresponding to the same significant POD boundary basis functions as in the fully linear case. We use the following nonlinear energy functional as the foundation for the loss function during training of the network.

$$E(v) = \int_{\Omega} \frac{1}{2}(1 + v^2)|\nabla v|^2 \mathrm{d}x \tag{90}$$

This functional corresponds to the nonlinear differential operator whose inverse (the solution operator) we want to approximate with the neural network. We use a simple multilayer perceptron network architecture with 4 hidden layers of the same width X and an output layer of width O representing the finite element DoFs. For standard P1 elements considered here it is simply the finite element function's nodal values. We use the exponential linear unit (ELU) as the activation function in the 4 hidden layers and no activation function in the last layer. A schematic illustration of this network is provided in Fig. 3(b).

In each iteration during the training, we pick a fixed number (referred to as the batch size) of randomly selected coefficient vectors and use them to compute an average loss. The coefficient values are picked from $\mathcal{N}(0, 0.09)$. The optimization is performed with the Adam optimizer where we perform 10^6 iterations with a decreasing learning rate. The learning rate starts at 1e-4, and after every 250k iterations, it is decreased by a factor of 0.5.

To measure the well-trainedness of the network, we, as an initial guiding measure, use the zero energy $E(\phi_{u,N,h}(0))$, i.e., the value of the computed energy using the output from the network when an all zero vector is given as input. This, of course, corresponds to homogeneous Dirichlet boundary conditions and gives that the solution $u = 0$ and thus that $E(0) = 0$. We also perform more

Table 1
Network architectures and batch sizes used for the various mesh sizes. DoFs refers to the number of DoFs in the finite element space V_h, which is the dimension of the MLP's output vector. Width refers to the width of the four hidden layers in the MLP.

Mesh	DoFs (O)	Width (X)	Batch size
10×10	81	64	32
28×28	729	256	64
82×82	6561	512	64
244×244	59 049	1024	96

Table 2
Training info from using an A100 GPU.

(a) Input data size $N = 9$

Mesh	Els	Training time	GPU Util	Inference time	$E(\phi_{u,N,h}(0))$	H_0^1-error 1k-avg (rel)	L^2-error 1k-avg (rel)
10×10	All	358 s	45%	0.8 ms	3.9e−5	9.4e−3 (1.87%)	4.3e−4 (0.47%)
28×28	All	337 s	73%	0.8 ms	1.2e−6	2.1e−3 (0.7%)	9.7e−5 (0.18%)
82×82	All	615 s	100%	0.8 ms	3.2e−7	1.1e−3 (0.6%)	3.0e−5 (0.1%)
244×244	3k	2673 s	100%	0.7 ms	5.2e−5	1.4e−2 (14.1%)	9.6e−5 (0.5%)

(b) Input data size $N = 21$

Mesh	Els	Training time	GPU Util	Inference time	$E(\phi_{u,N,h}(0))$	H_0^1-error 1k-avg (rel)	L^2-error 1k-avg (rel)
10×10	All	339 s	47%	0.8 ms	8.9e−5	1.6e−2 (1.23%)	1.2e−3 (1.07%)
28×28	All	354 s	69%	0.8 ms	5.1e−6	5.5e−3 (0.73%)	2.9e−4 (0.44%)
82×82	All	617 s	100%	0.8 ms	3.7e−6	3.6e−3 (0.82%)	8.4e−5 (0.22%)
244×244	4k	2733 s	100%	0.8 ms	1.0e−4	2.5e−2 (9.8%)	1.6e−4 (0.72%)

rigorous studies of well-trainedness by computing the actual finite element solution with FEniCS [44] and comparing it to the network approximation. This is done by computing their average norm difference over 1000 problems, where for each problem we randomly select a coefficient vector with values from $\mathcal{N}(0, 0.09)$. The difference is computed in both the H_0^1-norm (H^1-seminorm) and the L^2-norm. We also compute both the absolute and the relative norm differences, where the relative norm difference is the absolute difference divided by the norm of the finite element solution.

For the numerical examples we have again considered the two different coefficient vector lengths (9 and 21) and the four meshes from the linear case in the previous section. The network architectures and batch sizes used during training are given in Table 1.

The hyperparameters of the network (number of layers, width, activation function) and training settings (optimizer, number of iterations, decreasing learning rate, batch size, etc.) have been obtained by trial and error, where we have looked at the zero energy. An intuition for the size of the network (number of layers and widths) is that it needs to be large enough to provide a good approximation of the solution operator, but sufficiently small so that training is feasible and economical. An intuition for using ELU is that it is smoother than many other activations, e.g., the commonly used ReLU, which is to some degree in accordance with PDE-theory, where the solution is expected to depend smoothly on the problem data, e.g., boundary values.

In Table 2, we present training info for the four mesh sizes for coefficient vector length $N = 9$ and $N = 21$. The training has been performed on a single A100 GPU. For the largest mesh case (244×244), we have not been able to train with all elements present in the energy functional loss function (It has resulted in a NaN loss function value). To make it work, we have employed the trick of randomly selecting a fixed number of elements for every input vector during training, and only considering the energy functional contribution from those elements. The number of elements used is denoted "Els" in Table 2. It can be observed from the zero energies and norm errors in Table 2 that the operator networks generally become more accurate with finer meshes *if* all elements are used in the energy computation. In both cases with the 244×244 mesh, the trend in higher accuracy is broken. This is reasonable since only a few elements, instead of all, are used in the energy computation for the loss function.

With these neural networks we may solve the inverse minimization problem over the coefficient space. In Fig. 7, a demonstration of this process is presented for the case of 21 input coefficients and the 244×244 mesh, i.e., the neural network whose training info is presented in the last row of Table 2. In Fig. 7 it can clearly be observed that the approximate finite element solution given by the operator network approaches the noisy data and the reference solution as the optimization progresses.

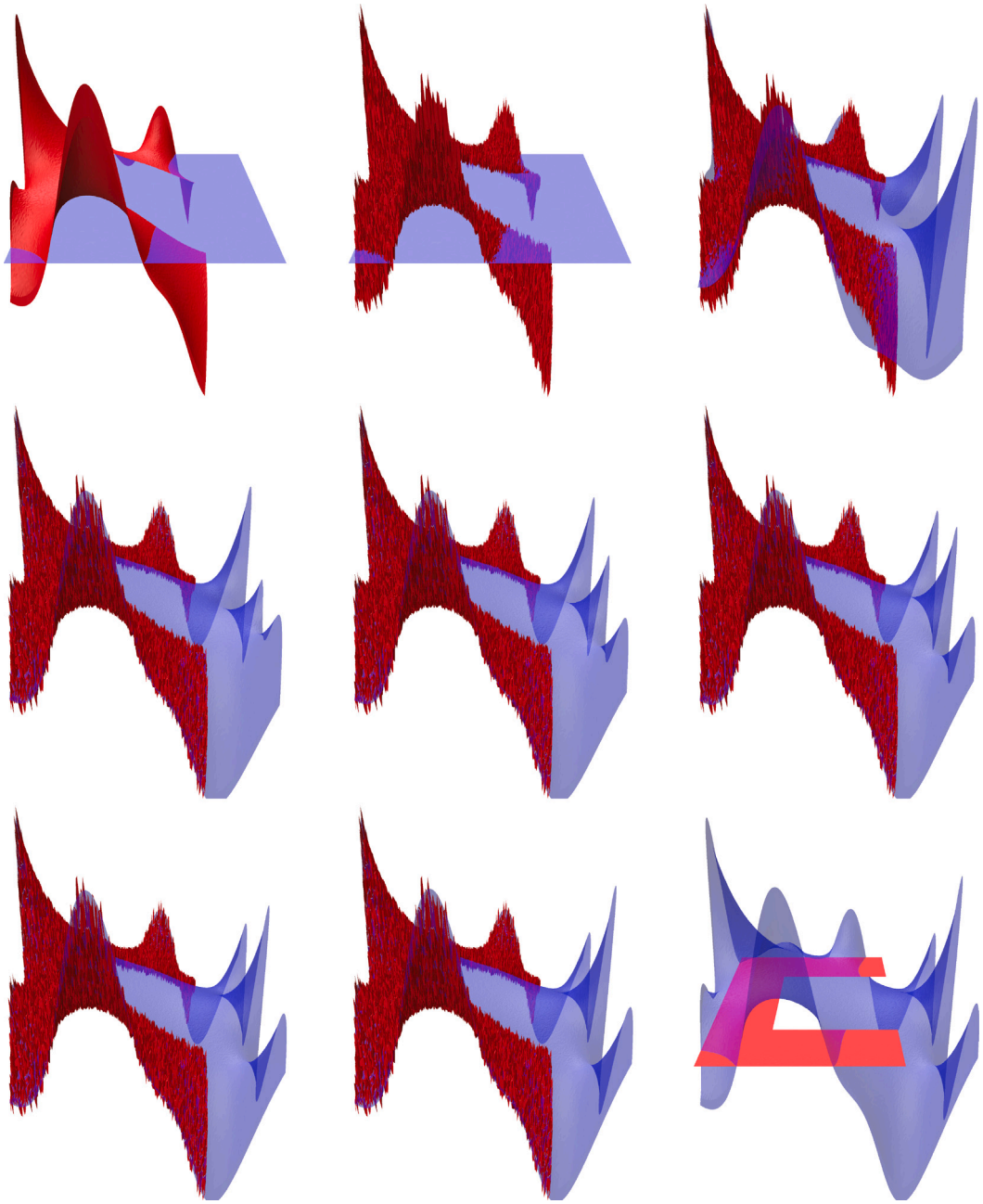


Fig. 7. Optimization process over a 21-dimensional *coefficient space* for a *nonlinear* inverse problem with noisy data. Here, the operator network in the last row of Table 2 (21 input coefficients, 59049 output DoFs) is used. The unperturbed data is shown in the first frame. The second frame is the same as the first but with added noise sampled from $\mathcal{U}(-0.05, 0.05)$. The last frame shows ω and the reference solution used for the data which was obtained by taking $p_{14} = 10$ and all other p_n's $= 0$. The penultimate frame shows the optimization's MSE-converged reconstruction of the reference solution. The MSE converged after 2843 iterations with the Adam optimizer with a step size $= 0.1$. This took 140.2 s on an Apple M1 CPU. The MSE's between the reference solution and the converged reconstruction are: on ω (used in optimization), $\text{MSE}_{\omega} = 8.36\text{e}{-4}$; on the convex hull of ω, $\text{MSE}_{\text{co}(\omega)} = 8.38\text{e}{-4}$; and on its complement, $\text{MSE}_{\text{co}(\omega)^c} = 1.91\text{e}{-3}$.

5.3. Nonlinear operator with nonlinear data

We consider the same nonlinear differential operator with the same neural networks as in the previous section but here we add complexity by introducing an underlying nonlinear dependence on the input coefficients to the network. To construct such a nonlinear dependence we may pick a smooth function $a : X \to \mathbb{R}^{|J|}$, where X is a parameter domain in $\mathbb{R}^{n_X}$, and for some index

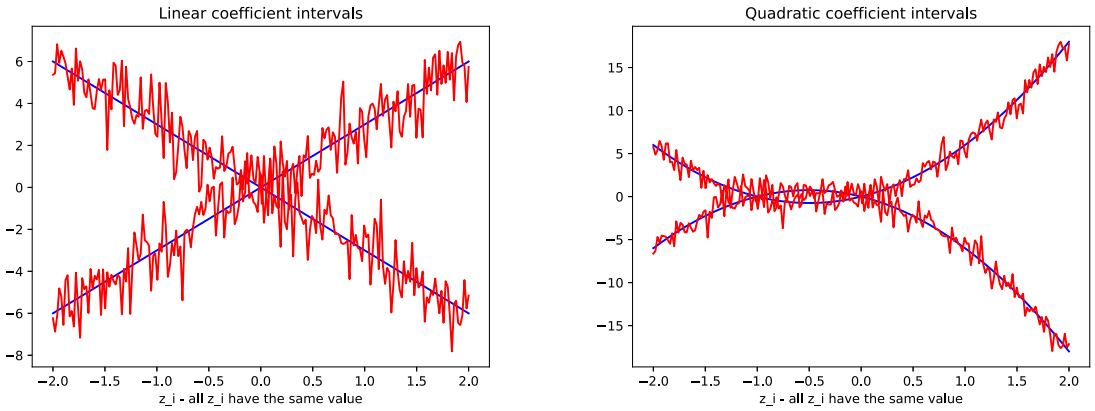


Fig. 8. Coefficients versus parameters for the linear case (left) and the quadratic case (right). All parameters have the same value which varies between −2 and 2. For each case, there is an upper and lower bound for the coefficients obtained by taking all matrix entries to either have the minimum value −1 or the maximum value 1. Both bounds are also shown as unperturbed (blue) and perturbed (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

set I, consider boundary data of the form

$$\mathcal{G}_a = \left\{ g_i = \sum_{j \in J} (a_j(x_i) + \delta_j)\varphi_{\mathrm{POD},j} \mid i \in I \right\} \tag{91}$$

where δ_j is some small probabilistic noise and $\{x_i \in X \mid i \in I\}$ is a set of samples from the parameter space X equipped with a probability measure. In this case, we expect an autoencoder with a latent space Z of at least the same dimension as X to perform well.

Polynomial data. We consider a simple polynomial example where the coefficients $a \in \mathbb{R}^{|J|}$ depend on the parameter variables $x \in \mathbb{R}^{n_X}$ as

$$a = a(x) = Ax + Bx^2 + \delta \tag{92}$$

Here the matrices $A, B \in \mathbb{R}^{|J| \times n_X}$ and their entries are randomly sampled from a uniform distribution. The perturbations $\delta \in \mathbb{R}^{|J|}$ are sampled from a normal distribution.

For the numerical results we take $|J| = 9$, $n_X = 3$ and sample matrix entries from $\mathcal{U}(-1, 1)$ which are then held fixed. To generate coefficient vectors, we sample parameter variables $x_k \sim \mathcal{U}(-2, 2)$ and perturbations $\delta_j \sim \mathcal{N}(0, 1)$. We consider two cases: the linear case with $B = 0$ and the quadratic case with $B \neq 0$. To get a sense of what the data look like, we plot the coefficients as functions of the parameters for both cases in Fig. 8. We analyze data generated for the linear case with PCA and data generated for the quadratic case with both PCA and autoencoders. For the autoencoders we have used MLPs with 6 layers (5 hidden, 1 output) with the third layer being the latent layer. The latent layer width has been varied and the remaining hidden layer widths have all been fixed at 64. The activation function ELU has been applied to all layers except the last. The training has been performed with the Adam optimizer exactly as for the operator networks, i.e., 10^6 iterations with a decreasing learning rate. The batch size has been 64. The hyperparameters of the autoencoders and training settings have, just as in the case of the operator networks, been obtained by trial and error. To measure well-trainedness, we have looked at the average mean squared reconstruction error over 1000 unperturbed samples generated in the same way as during training. The training time for a single autoencoder (fixed latent layer width) on an Apple M1 CPU has typically been in the range 240–270 s.

The results from both the PCA and autoencoder analysis are presented in Fig. 9. The PCA results give a 3-dimensional latent space in the linear case and a 6-dimensional in the quadratic. This is evident from the number of significant singular values for the different cases. The autoencoder results suggest the existence of both a 3- and a 6-dimensional latent space in the quadratic case. This can be deduced from the two plateaus for the two perturbed cases: one at latent layer widths 3–5 and one at 6–8. The autoencoders thus manage to find the underlying 3-dimensional structure in the quadratic case whereas PCA does not.

Gaussian data. We consider a more advanced nonlinear example where the coefficients $a \in \mathbb{R}^{|J|}$ depend on the parameter variables $x \in \mathbb{R}^{n_X}$ as

$$a_j = a_j(x) = \exp(-\gamma(x_k - x_{0,l})^2) + \delta_j \tag{93}$$

Here we have L number of equidistant Gaussian bell curves indexed by l where each coefficient is assigned exactly one bell curve with midpoint $x_{0,l}$ and exactly one parameter x_k according to $l = j \mod L$ and $k = j \mod n_X$, respectively. The perturbations $\delta \in \mathbb{R}^{|J|}$ are sampled from a normal distribution.

For the numerical results we take $\gamma = 2$ and sample perturbations $\delta_j \sim \mathcal{N}(0, 0.0225)$ (standard deviation = 0.15). We consider four cases:

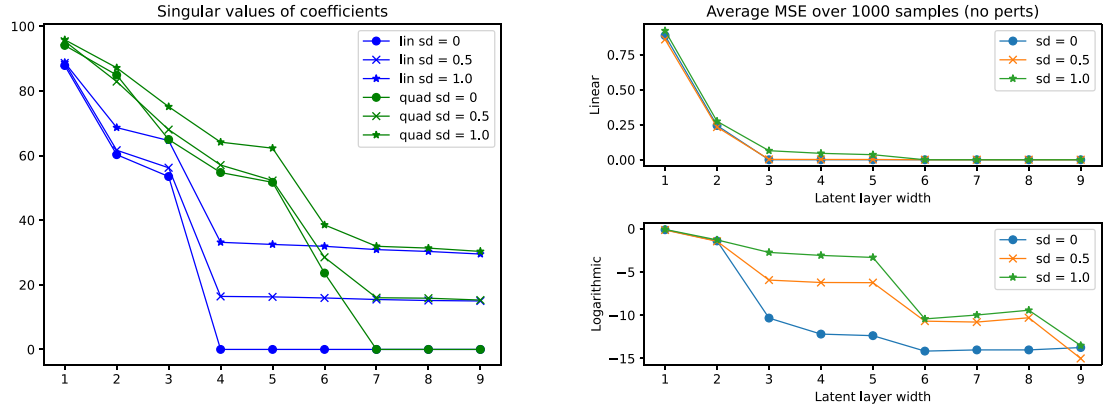


Fig. 9. Left: PCA results for both the linear and quadratic case. The plots show the singular values of the coefficients in decreasing order for three different perturbations: unperturbed and two perturbed (standard deviation = 0.5 and 1). **Right:** Autoencoder results for the quadratic case for the same three perturbations as PCA but used during training. The plots show the average mean squared reconstruction error over unperturbed test data for different latent layer widths both on a linear and logarithmic scale.

Table 3
Summary of optimization results for all *nonlinear* inverse problems using an operator network. All problems have the same reference solution and use the operator network in the last row of Table 2 (21 input coefficients, 59049 output DoFs). The optimization processes for the problems are presented in Figs. 7, 11 – 13, respectively. In the table, "Op" means the operator network, "dec" means decoder, "sd = x" means what perturbation was added to the training data for the decoder, and "δ_ω" means that noisy data was used for the inverse problem.

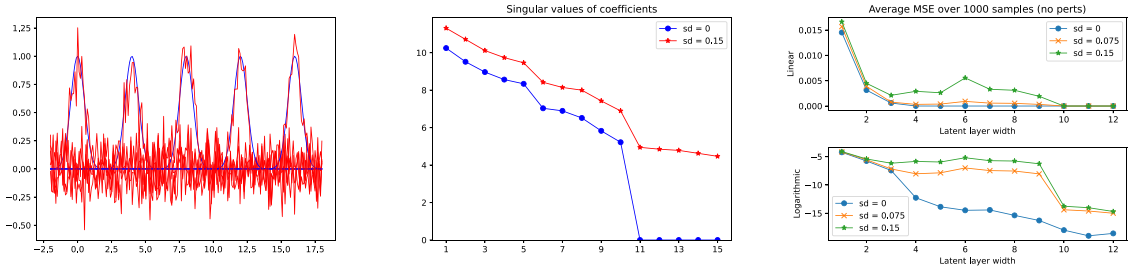
Configuration	Iterations	Avg iter time	MSE_ω	$\text{MSE}_{\text{co}(\omega)}$	$\text{MSE}_{\text{co}(\omega)^c}$
Op, δ_ω	2843	4.93e−2 s	8.36e−4	8.38e−4	1.91e−3
Op + dec "sd = 0"	1481	4.65e−2 s	2.22e−3	1.52e−3	4.83e−3
Op + dec "sd = 0", δ_ω	1018	4.67e−2 s	3.07e−3	2.37e−3	5.25e−3
Op + dec "sd = 0.15", δ_ω	212	5.19e−2 s	1.99e−2	1.44e−2	5.51e−2

- $(n_X, L) = (2, 5)$ with $x_0 = (0, 4, 8, 12, 16)$ and $x_k \sim \mathcal{U}(-2, 18)$
- $(n_X, L) = (3, 6)$ with $x_0 = (0, 2, 4, 6, 8, 10)$ and $x_k \sim \mathcal{U}(-2, 12)$
- $(n_X, L) = (3, 7)$ with $x_0 = (0, 2, 4, 6, 8, 10, 12)$ and $x_k \sim \mathcal{U}(-2, 14)$
- $(n_X, L) = (4, 8)$ with $x_0 = (0, 2, 4, 6, 8, 10, 12, 14)$ and $x_k \sim \mathcal{U}(-2, 16)$

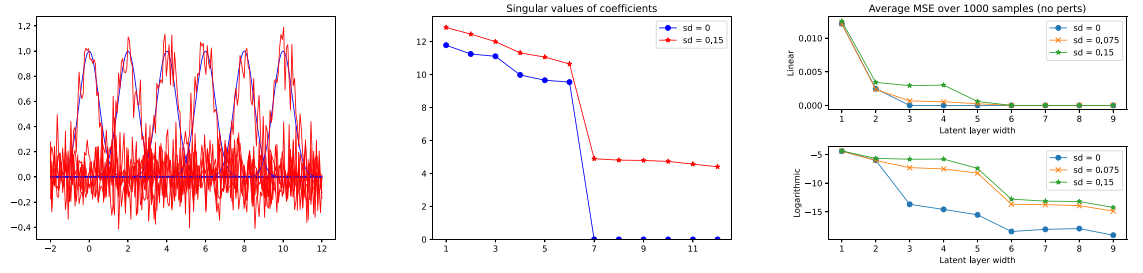
We analyze data generated for these cases with both PCA and autoencoders. For the autoencoders we have used MLPs with 5 layers (4 hidden, 1 output) with the middle layer being the latent layer. The latent layer width has been varied and the remaining hidden layer widths have all been fixed at 64. The activation function ELU has been applied to all layers except the last. The training has been performed with the Adam optimizer exactly as for the operator networks, i.e., 10^6 iterations with a decreasing learning rate. The batch size has been 64. Again, the hyperparameters of the autoencoders and training settings have been obtained by trial and error, where we have looked at the average mean squared reconstruction error over 1000 unperturbed samples generated in the same way as during training. The training time for a single autoencoder (fixed latent layer width) on an Apple M1 CPU has typically been in the range 210–250 s.

The bell curves for the coefficients, PCA results and autoencoder results are presented in Fig. 10. The PCA results show something interesting. If the number of bell curves L is divisible by the latent dimension n_X, PCA gives that the underlying structure has dimension L. If L is *not* divisible by n_X, PCA instead gives that this dimension is $n_X L$. For example, for $(n_X, L) = (2, 5)$ in Fig. 10(a), PCA gives latent dimension = 10, and for $(n_X, L) = (3, 6)$ in Fig. 10(b), PCA gives latent dimension = 6. This phenomenon is easily understood by the number of unique combinations of latent parameters x_k and bell curves, characterized by $x_{0,l}$, in the construction of the coefficients given by (93). The autoencoder results all suggest the existence of latent spaces of a lower dimension than given by PCA. This is most clearly seen from the existence of plateaus for the two perturbed cases (standard deviation = 0.075 and 0.15) on the logarithmic scale in all four cases. However, the suggested latent dimension does match the actual one as well as in the previous example with polynomial data, hinting at the higher complexity of the Gaussian data. This is especially true in the cases where n_X does not divide L.

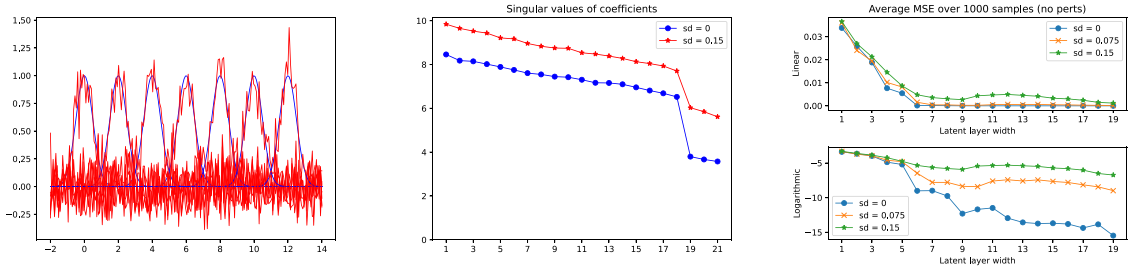
Combining operator network with decoder. In the third Gaussian data example with results presented in Fig. 10(c), we have $(n_X, L) = (3, 7)$. Here the PCA suggests that the underlying dimension is 21 (number of significant singular values), whereas the corresponding autoencoder study suggests that a reduction down to 9 dimensions could provide the same improvement as a reduction down to 17 in the case of the autoencoders trained on perturbed data (9 and 17 give roughly the same error). In light of the above, we



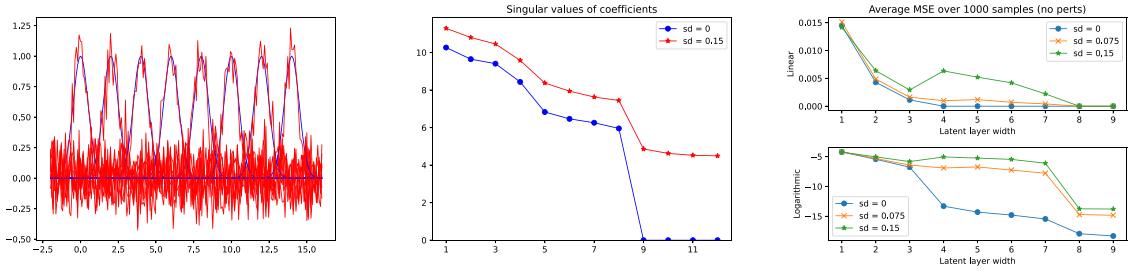
(a) $n_X = 2$, $L = 5$



(b) $n_X = 3$, $L = 6$



(c) $n_X = 3$, $L = 7$



(d) $n_X = 4$, $L = 8$

Fig. 10. Gaussian data examples. **Left:** Bell curves used for the coefficients, unperturbed (blue) and perturbed (red). **Middle:** PCA results for unperturbed data (blue) and perturbed (red, standard deviation = 0.15). The plots show the singular values of the coefficients in decreasing order. **Right:** Autoencoder results for three different perturbations used during training: unperturbed and two perturbed (standard deviation = 0.075 and 0.15). The plots show the average mean squared reconstruction error over unperturbed test data for different latent layer widths both on a linear and logarithmic scale. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

may take an autoencoder with latent layer width = 9 from this case and connect its decoder to the input of the operator network for the 244×244 mesh with 21 input coefficients. We may thus solve an inverse minimization problem over a 9-dimensional latent space instead of a 21-dimensional coefficient space. We present demonstrations of this process in Figs. 11–13. A summary of the optimization results for these three demonstrations and also the one in Fig. 7 is given in Table 3.

The main difference between the three demonstrations is the decoder used. First in Figs. 11–12, we use the decoder from the "sd = 0" autoencoder, meaning it was trained on unperturbed data. The first of these two demonstrations is for clean data, u_0 in

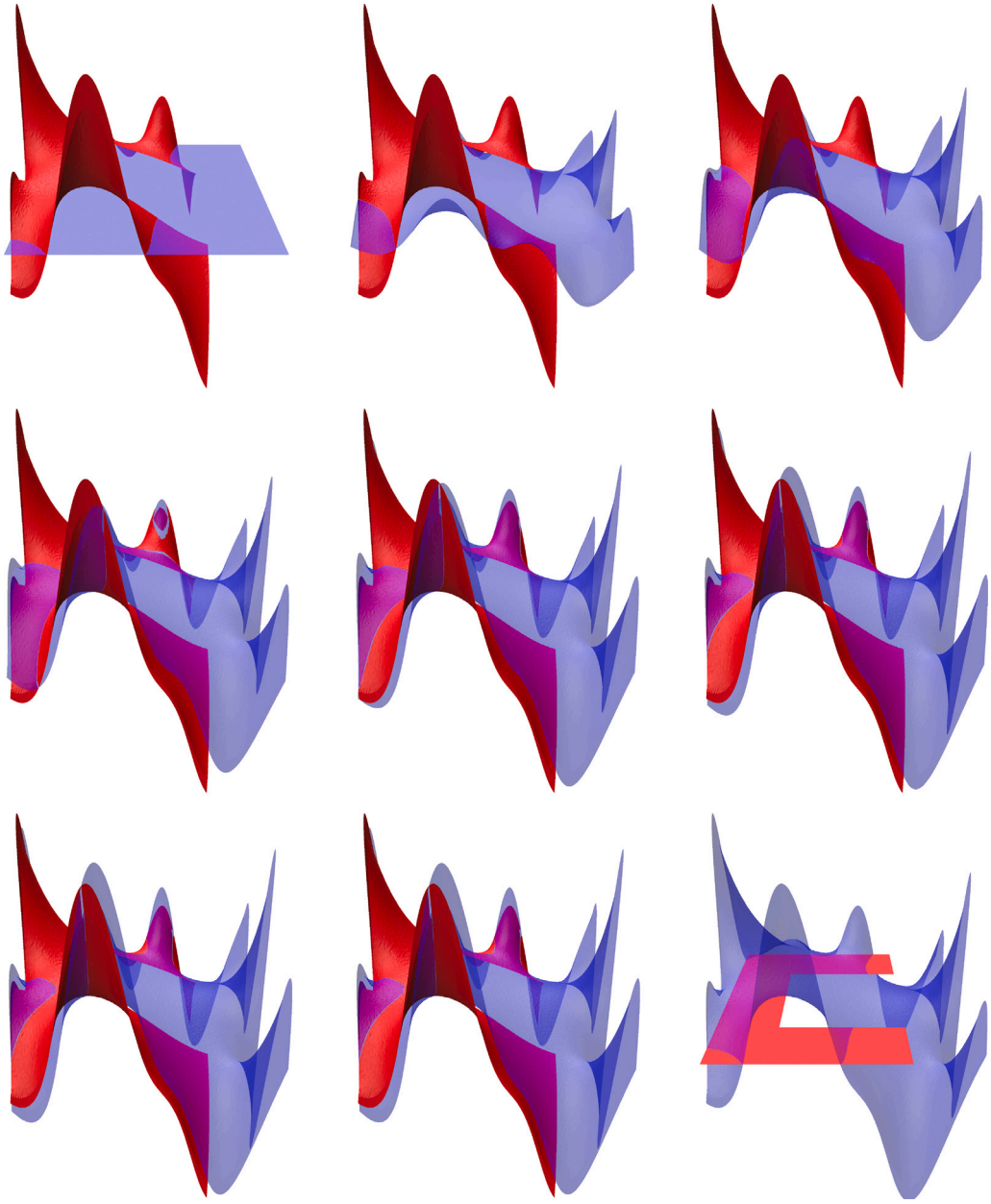


Fig. 11. Optimization process over a 9-dimensional *latent space* for a *nonlinear* inverse problem with *clean* data. Again, the operator network in the last row of Table 2 (21 input coefficients, 59049 output DoFs) is used, but here together with the "sd = 0" decoder from the right frame in Fig. 10(c). The decoder maps from a 9-dimensional latent space to a 21-dimensional coefficient space. The last frame shows ω and the reference solution used for the data which was obtained by taking $p_{14} = 10$ and all other p_n's = 0. The penultimate frame shows the optimization's MSE-converged reconstruction of the reference solution. The MSE converged after 1481 iterations with the Adam optimizer with a step size = 0.1. This took 68.9 s on an Apple M1 CPU. The MSE's between the reference solution and the converged reconstruction are: on ω (used in optimization), MSE_{ω} = 2.22e−3; on the convex hull of ω, $\text{MSE}_{\text{co}(\omega)}$ = 1.52e−3; and on its complement, $\text{MSE}_{\text{co}(\omega)^c}$ = 4.83e−3.

ω, and the second for noisy. We see that the two optimization processes are essentially the same but find it instructive to present both as the clean data case functions as a reference. Second, in Fig. 13, we use the decoder from the "sd = 0.15" autoencoder, meaning it was trained on perturbed data with perturbations from $\mathcal{N}(0, 0.0225)$. From the logarithmic scale in the right frame in Fig. 10(c) we see that the reconstruction errors of the two autoencoders differ substantially, by several orders of magnitude. Comparing the corresponding optimization processes, we also see that using the "sd = 0" decoder (Fig. 12) produces a much more accurate reconstruction compared to the "sd = 0.15" decoder (Fig. 13) that fails to do so.

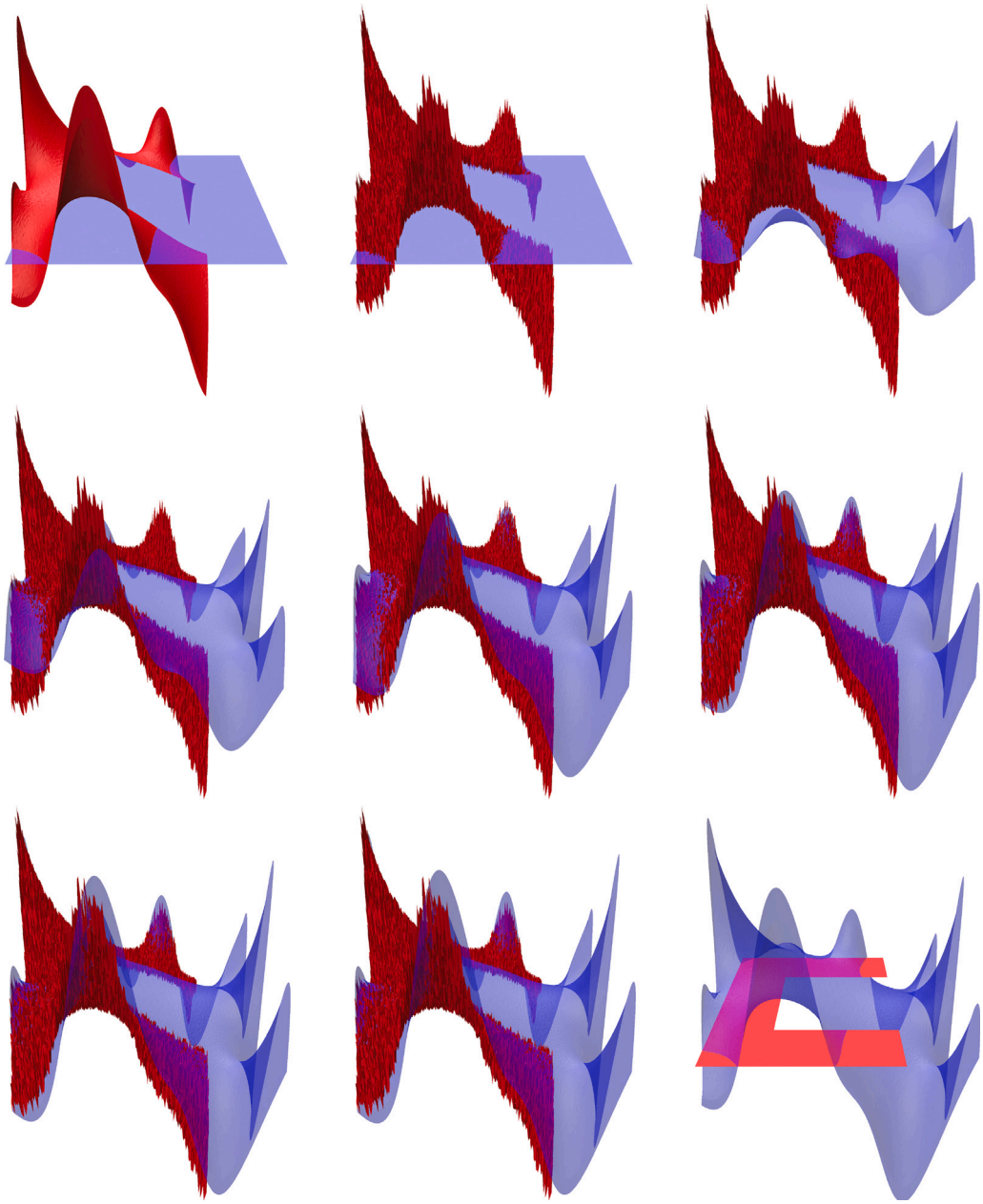


Fig. 12. Optimization process over a 9-dimensional *latent space* for a *nonlinear* inverse problem with noisy data. Again, the operator network in the last row of Table 2 (21 input coefficients, 59049 output DoFs) is used together with the "sd = 0" decoder from the right frame in Fig. 10(c). The decoder maps from a 9-dimensional latent space to a 21-dimensional coefficient space. The unperturbed data is shown in the first frame. The second frame is the same as the first but with added noise sampled from $\mathcal{U}(-0.05, 0.05)$. The last frame shows ω and the reference solution used for the data which was obtained by taking $p_{14} = 10$ and all other p_n's = 0. The penultimate frame shows the optimization's MSE-converged reconstruction of the reference solution. The MSE converged after 1018 iterations with the Adam optimizer with a step size = 0.1. This took 47.5 s on an Apple M1 CPU. The MSE's between the reference solution and the converged reconstruction are: on ω (used in optimization), MSE_{ω} = 3.07e−3; on the convex hull of ω, $\text{MSE}_{\text{co}(\omega)}$ = 2.37e−3; and on its complement, $\text{MSE}_{\text{co}(\omega)^c}$ = 5.25e−3.

The reconstructions in all three decoder cases, and especially the last, are less accurate compared to the case with only the operator network presented in Fig. 7, as can be seen from both the figures and the MSE's in Table 3. This is reasonable since the reference solution in all four cases is the same network output corresponding to a specific *coefficient* input and in the case with only the operator network we optimize in this coefficient space whereas in the decoder cases in some latent space. It is simply not guaranteed that the decoders may attain this specific coefficient input when mapping from the latent space. One reason being that a single change in any of the 9 latent variables can affect all the 21 coefficients. Comparing the MSE's on the different subdomains

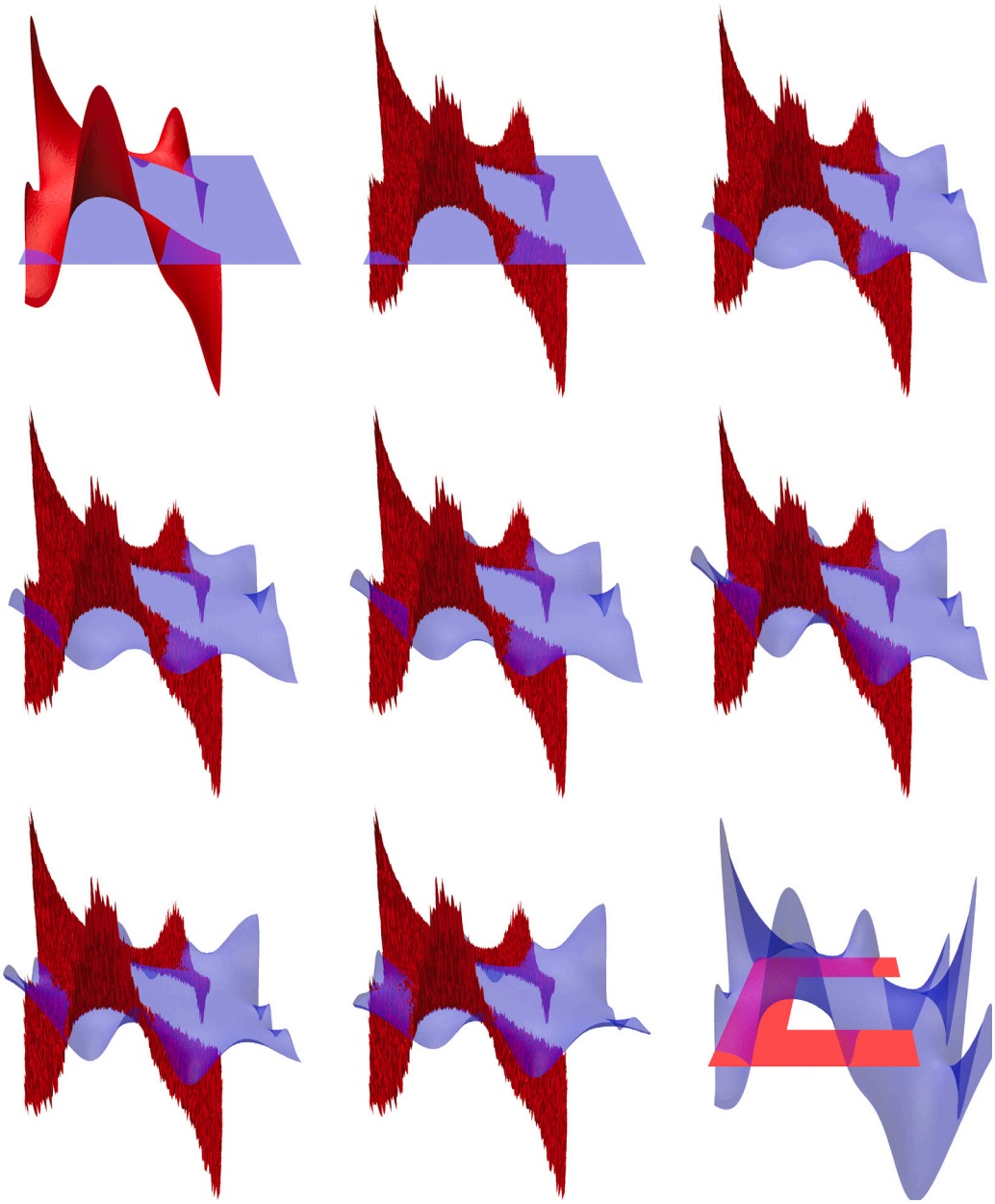


Fig. 13. Optimization process over a 9-dimensional *latent space* for a *nonlinear* inverse problem with noisy data. Again, the operator network in the last row of Table 2 (21 input coefficients, 59049 output DoFs) is used, but here together with the "sd = 0.15" decoder from the right frame in Fig. 10(c). The decoder maps from a 9-dimensional latent space to a 21-dimensional coefficient space. The unperturbed data is shown in the first frame. The second frame is the same as the first but with added noise sampled from $\mathcal{U}(-0.05, 0.05)$. The last frame shows ω and the reference solution used for the data which was obtained by taking $p_{14} = 10$ and all other p_n's = 0. The penultimate frame shows the optimization's MSE-converged reconstruction of the reference solution. The MSE converged after 212 iterations with the Adam optimizer with a step size = 0.1. This took 11.0 s on an Apple M1 CPU. The MSE's between the reference solution and the converged reconstruction are: on ω (used in optimization), $\text{MSE}_{\omega} = 1.99\text{e}{-2}$; on the convex hull of ω, $\text{MSE}_{\text{co}(\omega)} = 1.44\text{e}{-2}$; and on its complement, $\text{MSE}_{\text{co}(\omega)^c} = 5.51\text{e}{-2}$.

in Table 3, we see that in all four cases it is smaller on the convex hull of ω than on the complement as expected. This is also true for the fully linear case (corresponding results are presented in the caption of Fig. 6). The average iteration times presented in Table 3 are essentially the same for the four cases. Something that is positive for using decoders, but maybe not so surprising considering how much smaller the decoder MLP's are in comparison to the operator MLP. In summary, autoencoders may be used to reduce the dimension of the optimization space (latent instead of coefficient space), but to really gain from such a reduction and to maintain

accuracy, care needs to be taken in how the reduction mapping is constructed. We point out that the MLP approach considered here is rather simple and that we believe there is substantial room for improvement by considering more sophisticated methods.

As final remarks we point out that taking some output of the method under consideration as the reference solution, as is done here, is typically not a proper choice since it is too idealized. However, here we make this choice to put more focus on the effects of latent space optimization. We also point out that all the optimization processes involving neural networks presented here have been for the rougher networks: the operator network in the last row of Table 2 and the autoencoders in Fig. 10(c) have alternatives with better measures of well-trainedness. The idea behind this being that if the concept works to some degree in the harder cases, it should work even better in the easier ones.

6. Conclusions

The regularization of severely ill-posed inverse problems using large data sets and stabilized finite element methods was considered and shown to be feasible both for linear and nonlinear problems. In the linear case, a fairly complete theory for the approach exists, and herein, we complemented previous work with the design and analysis of a reduced-order model. In the linear case, a combination of POD for the data reduction and reduced model method for the PDE-solution was shown to be a rigorous and robust approach that effectively can improve stability from logarithmic to linear in the case where the data is drawn from some finite-dimensional space of moderate dimension. To extend the ideas to nonlinear problems we introduced a machine learning framework, both for the data compression and the reduced model. After successful training, this resulted in a very efficient method for the solution of the nonlinear inverse problem. The main observations were the following:

1. The combination of analysis of the inverse problem, numerical analysis of finite element reconstruction methods, and data compression techniques allows for the design of robust and accurate methods in the linear case.
2. Measured data can be used to improve stability, provided a latent data set of moderate size can be extracted from the data cloud.
3. Machine learning can be used to leverage the observations in the linear case to nonlinear inverse problems and data assimilation and results in fast and stable reconstruction methods.

The main open questions are related to how the accuracy of the machine learning approach can be assessed and controlled through network design and training. For recent work in this direction, we refer to [45].

CRediT authorship contribution statement

Erik Burman: Writing – review & editing, Writing – original draft, Funding acquisition. **Mats G. Larson:** Writing – review & editing, Writing – original draft, Supervision, Funding acquisition. **Karl Larsson:** Writing – review & editing, Visualization. **Carl Lundholm:** Writing – review & editing, Writing – original draft, Visualization, Software, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported in part by the Swedish Research Council Grant, No. 2021-04925, and the Swedish Research Programme Essence. EB acknowledges funding from EPSRC, United Kingdom grants EP/T033126/1 and EP/V050400/1.

The GPU computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement No. 2022-06725.

Data availability

No data was used for the research described in the article.

References

[1] G. Alessandrini, L. Rondi, E. Rosset, S. Vessella, The stability for the Cauchy problem for elliptic equations, Inverse Probl. 25 (12) (2009) 123004, 47, http://dx.doi.org/10.1088/0266-5611/25/12/123004.

[2] E. Burman, M. Nechita, L. Oksanen, Optimal approximation of unique continuation, Found. Comput. Math. (2024) http://dx.doi.org/10.1007/s10208-024-09655-w.

[3] E. Burman, L. Oksanen, Finite element approximation of unique continuation of functions with finite dimensional trace, Math. Models Methods Appl. Sci. 34 (10) (2024) 1809–1824, http://dx.doi.org/10.1142/S0218202524500362.

[4] M.G. Larson, C. Lundholm, A. Persson, Nonlinear operator learning using energy minimization and MLPs, 2024, http://dx.doi.org/10.48550/arXiv.2412.04596, arXiv Preprint.

[5] N. Sharp, et al., Data-free learning of reduced-order kinematics, in: Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings, ACM, Los Angeles CA USA, 2023, pp. 1–9, http://dx.doi.org/10.1145/3588432.3591521.

[6] L. Bourgeois, A mixed formulation of quasi-reversibility to solve the Cauchy problem for Laplace's equation, Inverse Probl. 21 (3) (2005) 1087–1104, http://dx.doi.org/10.1088/0266-5611/21/3/018.

[7] E. Burman, Stabilized finite element methods for nonsymmetric, noncoercive, and ill-posed problems. Part I: Elliptic equations, SIAM J. Sci. Comput. 35 (6) (2013) A2752–A2780, http://dx.doi.org/10.1137/130916862.

[8] E. Burman, Error estimates for stabilized finite element methods applied to ill-posed problems, C. R. Math. Acad. Sci. Paris 352 (7–8) (2014) 655–659, http://dx.doi.org/10.1016/j.crma.2014.06.008.

[9] E. Burman, P. Hansbo, M.G. Larson, Solving ill-posed control problems by stabilized finite element methods: an alternative to Tikhonov regularization, Inverse Probl. 34 (3) (2018) 035004, 36, http://dx.doi.org/10.1088/1361-6420/aaa32b.

[10] E. Chung, K. Ito, M. Yamamoto, Least squares formulation for ill-posed inverse problems and applications, Appl. Anal. 101 (15) (2022) 5247–5261, http://dx.doi.org/10.1080/00036811.2021.1884228.

[11] W. Dahmen, H. Monsuur, R. Stevenson, Least squares solvers for ill-posed PDEs that are conditionally stable, ESAIM Math. Model. Numer. Anal. 57 (4) (2023) 2227–2255, http://dx.doi.org/10.1051/m2an/2023050.

[12] E. Burman, P. Hansbo, M.G. Larson, K. Larsson, Isogeometric analysis and augmented Lagrangian Galerkin least squares methods for residual minimization in dual norm, Comput. Methods Appl. Mech. Engrg. 417 (part B) (2023) http://dx.doi.org/10.1016/j.cma.2023.116302, Paper No. 116302, 17.

[13] E. Burman, L. Oksanen, Z. Zhao, Computational unique continuation with finite dimensional Neumann trace, 2024, http://dx.doi.org/10.48550/arXiv.2402.13695, arXiv Preprint.

[14] C. James, L'interaction entre données et modélisation pour le problème d'assimilation de données (Master's thesis), Sorbonne Université, 2023, Rapport de stage de Master 2.

[15] M. Boulakia, C. James, D. Lombardi, Numerical approximation of the unique continuation problem enriched by a database for the Stokes equations, 2024, Preprint, https://inria.hal.science/hal-04721560.

[16] S. Riffaud, M.A. Fernández, D. Lombardi, A low-rank solver for parameter estimation and uncertainty quantification in time-dependent systems of partial differential equations, J. Sci. Comput. 99 (2) (2024) http://dx.doi.org/10.1007/s10915-024-02488-3, Paper No. 34, 31.

[17] N. Kovachki, et al., Neural operator: learning maps between function spaces with applications to PDEs, J. Mach. Learn. Res. 24 (2023) http://dx.doi.org/10.48550/arXiv.2108.08481, Paper No. [89], 97.

[18] L. Zhang, T. Luo, Y. Zhang, W. E., Z.-Q.J. Xu, Z. Ma, MOD-Net: a machine learning approach via model-operator-data network for solving PDEs, Commun. Comput. Phys. 32 (2) (2022) 299–335, http://dx.doi.org/10.4208/cicp.oa-2021-0257.

[19] D. Ray, O. Pinti, A.A. Oberai, Deep Learning and Computational Physics, Springer Cham, Switzerland, 2024, http://dx.doi.org/10.1007/978-3-031-59345-1.

[20] N.R. Franco, A. Manzoni, P. Zunino, Mesh-informed neural networks for operator learning in finite element spaces, J. Sci. Comput. 97 (2) (2023) http://dx.doi.org/10.1007/s10915-023-02331-1, Paper No. 35, 41.

[21] T. Xu, D. Liu, P. Hao, B. Wang, Variational operator learning: a unified paradigm marrying training neural operators and solving partial differential equations, J. Mech. Phys. Solids 190 (2024) http://dx.doi.org/10.1016/j.jmps.2024.105714, Paper No. 105714, 29.

[22] D. Patel, D. Ray, M.R. Abdelmalik, T.J. Hughes, A.A. Oberai, Variationally mimetic operator networks, Comput. Methods Appl. Mech. Engrg. 419 (2024) 116536, http://dx.doi.org/10.1016/j.cma.2023.116536.

[23] M. Bachmayr, W. Dahmen, M. Oster, Variationally correct neural residual regression for parametric PDEs: On the viability of controlled accuracy, 2024, http://dx.doi.org/10.48550/arXiv.2405.20065, arXiv Preprint.

[24] S.A. McQuarrie, P. Khodabakhshi, K.E. Willcox, Nonintrusive reduced-order models for parametric partial differential equations via data-driven operator inference, SIAM J. Sci. Comput. 45 (4) (2023) A1917–A1946, http://dx.doi.org/10.1137/21M1452810.

[25] F. Romor, G. Stabile, G. Rozza, Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method, J. Sci. Comput. 94 (3) (2023) http://dx.doi.org/10.1007/s10915-023-02128-2, Paper No. 74, 39.

[26] S. Pakravan, P.A. Mistani, M.A. Aragon-Calvo, F. Gibou, Solving inverse-PDE problems with physics-aware neural networks, J. Comput. Phys. 440 (2021) http://dx.doi.org/10.1016/j.jcp.2021.110414, Paper No. 110414, 31.

[27] S. Cen, B. Jin, Q. Quan, Z. Zhou, Hybrid neural-network FEM approximation of diffusion coefficient in elliptic and parabolic problems, IMA J. Numer. Anal. 44 (5) (2023) 3059–3093, http://dx.doi.org/10.1093/imanum/drad073.

[28] S. Cen, B. Jin, K. Shin, Z. Zhou, Electrical impedance tomography with deep Calderón method, J. Comput. Phys. 493 (2023) 112427, http://dx.doi.org/10.1016/j.jcp.2023.112427.

[29] J. Berg, K. Nyström, Neural networks as smooth priors for inverse problems for PDEs, J. Comput. Appl. Math. Data Sci. 1 (2021) 100008, http://dx.doi.org/10.1016/j.jcmds.2021.100008.

[30] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[31] C. Lieberman, K. Willcox, O. Ghattas, Parameter and state model reduction for large-scale statistical inverse problems, SIAM J. Sci. Comput. 32 (5) (2010) 2523–2542, http://dx.doi.org/10.1137/090775622.

[32] T. Cui, Y.M. Marzouk, K.E. Willcox, Data-driven model reduction for the Bayesian solution of inverse problems, Internat. J. Numer. Methods Engrg. 102 (5) (2015) 966–990, http://dx.doi.org/10.1002/nme.4748.

[33] O. Ghattas, K. Willcox, Learning physics-based models from data: perspectives from inverse problems and model reduction, Acta Numer. 30 (2021) 445–554, http://dx.doi.org/10.1017/S0962492921000064.

[34] S. Arridge, P. Maass, O. Öktem, C.-B. Schönlieb, Solving inverse problems using data-driven models, Acta Numer. 28 (2019) 1–174, http://dx.doi.org/10.1017/S0962492919000059.

[35] Z. Gao, L. Yan, T. Zhou, Adaptive operator learning for infinite-dimensional Bayesian inverse problems, SIAM/ASA J. Uncertain. Quantif. 12 (4) (2024) 1389–1423, http://dx.doi.org/10.1137/24M1643815.

[36] S. Lunz, A. Hauptmann, T. Tarvainen, C.-B. Schönlieb, S. Arridge, On learned operator correction in inverse problems, SIAM J. Imaging Sci. 14 (1) (2021) 92–127, http://dx.doi.org/10.1137/20M1338460.

[37] N. Demo, M. Strazzullo, G. Rozza, An extended physics informed neural network for preliminary analysis of parametric optimal control problems, Comput. Math. Appl. 143 (2023) 383–396, http://dx.doi.org/10.1016/j.camwa.2023.05.004.

[38] A. Dasgupta, D.V. Patel, D. Ray, E.A. Johnson, A.A. Oberai, A dimension-reduced variational approach for solving physics-based inverse problems using generative adversarial network priors and normalizing flows, Comput. Methods Appl. Mech. Engrg. 420 (2024) 116682, http://dx.doi.org/10.1016/j.cma.2023.116682.

[39] A. Ivagnes, N. Demo, G. Rozza, Towards a machine learning pipeline in reduced order modelling for inverse problems: neural networks for boundary parametrization, dimensionality reduction and solution manifold approximation, J. Sci. Comput. 95 (1) (2023) http://dx.doi.org/10.1007/s10915-023-02142-4, Paper No. 23, 24.

[40] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, IMA J. Numer. Anal. 42 (2) (2022) 981–1022, http://dx.doi.org/10.1093/imanum/drab032.

[41] P. Escapil-Inchauspé, G.A. Ruz, h-analysis and data-parallel physics-informed neural networks, Sci. Rep. 13 (17562) (2023) http://dx.doi.org/10.1038/s41598-023-44541-5.

[42] M. Nechita, Solving ill-posed Helmholtz problems with physics-informed neural networks, J. Numer. Anal. Approx. Theory 52 (1) (2023) 90–101, http://dx.doi.org/10.33993/jnaat521-1305.

[43] L. Fulton, V. Modi, D. Duvenaud, D.I.W. Levin, A. Jacobson, Latent-space dynamics for reduced deformable simulation, Comput. Graph. Forum 38 (2) (2019) 379–391, http://dx.doi.org/10.1111/cgf.13645.

[44] A. Logg, K.-A. Mardal, G.N. Wells, et al., Automated Solution of Differential Equations by the Finite Element Method, Springer, 2012, http://dx.doi.org/10.1007/978-3-642-23099-8.

[45] M.V. de Hoop, D.Z. Huang, E. Qian, A.M. Stuart, The cost-accuracy trade-off in operator learning with neural networks, J. Mach. Learn. 1 (3) (2022) 299–341, http://dx.doi.org/10.48550/arXiv.2203.13181.