

Data-Driven Model Order Reduction for Problems with Parameter-Dependent Jump-Discontinuities

Neeraj Sarna^{*} Peter Benner[†]

^{*}*Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany.*

Email: sarna@mpi-magdeburg.mpg.de, ORCID: 0000-0003-0607-2067

[†]*Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany, and Faculty for Mathematics, Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany.*

Email: benner@mpi-magdeburg.mpg.de, ORCID: 0000-0003-3362-4103

Abstract: We propose a data-driven model order reduction (MOR) technique for parametrized partial differential equations that exhibit parameter-dependent jump-discontinuities. Such problems have poor-approximability in a linear space and therefore, are challenging for standard MOR techniques. We build upon the methodology of approximating the map between the parameter domain and the expansion coefficients of the reduced basis via regression. The online stage queries the regression model for the expansion coefficients and recovers a reduced approximation for the solution. We propose to apply this technique to a transformed solution that results from composing the solution with a spatial transform. Unlike the (untransformed) solution, it is sufficiently regular along the parameter domain and thus, is well-approximable in a low-dimensional linear space. To recover an approximation for the (untransformed) solution, we propose an online efficient regression-based technique that approximates the inverse of the spatial transform. Our method features a decoupled online and offline stage, and benchmark problems involving hyperbolic and parabolic equations demonstrate its effectiveness.

Keywords: Data-driven methods, Model order reduction, Parametrized PDEs, Image registration, Gaussian process regression

Novelty statement:

1. Combination of image registration and regression to approximate problems with parameter-dependent jump-discontinuities.
2. Image registration provides a transformed solution, which is well approximable in a linear reduced space.
3. Regression approximates the mapping from the parameter domain to the reduced basis coefficients of the transformation solution.
4. A de-transformation step recovers an approximation for the (untransformed) solution.
5. Numerical experiments report significant improvements over a standard linear approximation.

1 Introduction

We consider parametrized partial differential equations (pPDEs) in a multi-query scenario where we seek a solution at multitudes of different parameter instances. Such scenarios arise in applications related to active control [9], design optimization [3, 20, 48], uncertainty quantification [10, 14], etc. Although a high-fidelity finite-difference/element/volume type solver can approximate the solution at any given parameter, it is prohibitively expensive for multi-query scenarios. We, therefore, resort to a MOR-based surrogate.

The MOR technique splits the solution procedure into an offline and an online stage. The offline stage bears the cost of the most expensive computations and is decoupled from the online stage, allowing for online efficiency. Using the high-fidelity solver, it collects solution snapshots and computes a set of reduced basis vectors $\mathcal{X}_n := \{v_i\}_{i=1,\dots,n}$. The online stage computes an approximation in the span of this basis. Precisely, let $u(\cdot, z)$, defined over a spatial domain Ω , represent a solution (or its high-fidelity approximation) to our pPDE. Furthermore, let $z \in \mathcal{Z}$ be some parameter-of-interest, and let $u_n(\cdot, z) \in \text{span}(\mathcal{X}_n)$ be an approximation to $u(\cdot, z)$ such that

$$u_n(\cdot, z) := \sum_{i=1}^n \alpha_{u,i}(z) v_i. \quad (1)$$

Then, the online stage computes the coefficients $\alpha_{u,i}(z)$. We collect all these coefficients in a vector $\alpha_u(z) \in \mathbb{R}^n$.

We consider the proper-orthogonal-decomposition (POD) approach to construct \mathcal{X}_n . Since the POD is data-driven, the intrusive or non-intrusive nature of our MOR technique hinges on the methodology used to compute the vector $\alpha_u(z)$. An intrusive technique computes $\alpha_u(z)$ by projecting the pPDE onto the approximation space $\text{span}(\mathcal{X}_n)$. In doing so, it accesses the discrete (high-fidelity) evolution operators—the books [8, 21] and the review paper [7] discuss this approach at length. In contrast to the intrusive approach, a non-intrusive approach treats the high-fidelity solver as a black-box. It computes $\alpha_u(z)$ either via regression or via a data-driven approximation of its evolution operator.

Broadly speaking, non-intrusive techniques are either physics or non-physics informed. The former requires structural information of the non-linearities in the underlying pPDE. This information is then used to infer the evolution operator of $\alpha_u(z)$ [1, 6, 31, 32]. In contrast, the latter “ignores” the underlying pPDE altogether. It treats the solution $u(\cdot, z)$ like a generic parametrized function that might as well not even correspond to the solution of a pPDE.

The non-physics informed methods either (i) directly approximate the vector $\alpha_u(z)$ using regression; or (ii) first approximate $\alpha_u(z)$ ’s evolution operator—without using the structural properties of the non-linearities—followed by time-stepping. Authors in [4, 13], [19] and [22] undertake the first approach and use radial basis functions (RBFs), Gaussian processes (GP) and neural networks, respectively, for regression. Authors in [45, 46] opt for the second approach and reduce the Navier–Stokes equations by approximating the evolution operator using RBFs—[47] and [26] provide extensions to problems involving fluid-structure interaction and moving boundaries, respectively.

1.1 Regression based non-intrusive MOR

We consider a non-physics informed technique that approximates the map between the parameter and the POD coefficients using regression. In particular, consider a set of parametrized functions given as $\{g(\cdot, z) : z \in \mathcal{Z}\}$. The function $g(\cdot, z)$ can correspond to a solution of a pPDE, a transformed solution of a pPDE (discussed below), or any other quantity-of-interest. [Algorithm 1](#) and [Algorithm 2](#) outline the building blocks of an algorithm that can approximate any function of the above set in a data-driven fashion. The main ingredient of the algorithm is a regression model for the mapping $z \mapsto \alpha_g(z)$, which the online phase queries to recover an approximation for $g(\cdot, z)$.

We use GPR to perform regression. The main reason being that we also develop a regression-based surrogate for the error introduced by our MOR technique. Since a GPR is probabilistic, its confidence region is helpful in devising an accurate error surrogate. However, if one is not interested in error modelling then, any other regression technique would also suffice—RBFs being a noteworthy example.

Algorithm 1 Regression based MOR: Offline stage

-
- 1: Collect the training samples $\{z^{(i)}\}_i$ and the snapshots $\{g(\cdot, z^{(i)})\}_i$.
 - 2: Compute the reduced basis $\mathcal{X}_n := \{v_i\}_{i=1,\dots,n}$.
 - 3: Orthogonally project $\{g(\cdot, z^{(i)})\}_i$ onto $\{v_i\}_{i=1,\dots,n}$ to compute the training data $\{(z^{(i)}, \alpha_g(z^{(i)}))_i\}$.
 - 4: Using regression, approximate $z \mapsto \alpha_g(z)$ by $z \mapsto \alpha_g^R(z)$.
-

Algorithm 2 Regression based MOR: Online stage

-
- 1: For any $z \in \mathcal{Z}$, query the regressed mapping $z \mapsto \alpha_g^R(z)$.
 - 2: Approximate $g(\cdot, z)$ by $g_n(\cdot, z) := \sum_i \alpha_{g,i}^R(z) v_i$.
-

1.1.1 Shortcomings of the standard approach

We consider pPDEs that exhibit parameter-dependent jump-discontinuities or steep-gradients. Several problems of practical interest exhibit such a behaviour. A standard example being that of non-linear hyperbolic equations, which mostly appear in applications involving fluid flows. Even for a smooth initial and boundary data, such problems can develop spatial discontinuities that move in the parameter domain [11, 44]. A diffusion equation whose diffusion coefficient has a moving discontinuity, also belongs to a similar category [39].

In the above algorithms, one can choose

$$g(\cdot, z) = u(\cdot, z) \quad (2)$$

and recover a non-intrusive MOR technique [4, 19, 22]. However, such an approach is inefficient for the aforementioned problems. Since a solution with parameter-dependent jump-discontinuities (or steep-gradients) has poor approximability in a linear space, only a large set of POD modes can provide a reasonable accuracy. This makes the MOR technique inefficient—see [8, 18, 44] for related proofs. Note that in the context of POD, a slow singular value decay is indicative of the poor approximability in POD basis.

It is noteworthy that even a sufficiently large set of POD modes does not guarantee a physically accurate and stable solution. Discontinuities in the parameter domain can trigger oscillations in the POD modes. Analogous to the Gibbs phenomenon for a Fourier series expansion, the oscillation frequency increases with the order of the POD modes. Therefore, although increasing the number of POD modes can provide a better approximation in the L^2 (or L^1)-sense, it results in a solution with un-physical high frequency oscillations—results in [12, 25, 47] showcase these oscillations.

1.1.2 A transformation and de-transformation approach

For the above reasons, we refrain from directly approximating the solution. Rather, we undertake a two step procedure comprising of a solution transformation followed by de-transformation. In the transformation step, we introduce a spatial transform

$$\varphi(\cdot, z_{\text{ref}}, z) : \Omega \rightarrow \Omega, \quad (3)$$

with $z_{\text{ref}} \in \mathcal{Z}$ being a reference parameter. This spatial transform is such that the transformed solution given by

$$g(\cdot, z) = u(\varphi(\cdot, z_{\text{ref}}, z), z), \quad (4)$$

at least ideally, has no discontinuities along \mathcal{Z} . This ensures that the transformed solution (with some additional assumptions) is well approximable in a sufficiently low-dimensional linear space—we refer to [36, 43, 44] for further details. One may associate a physical relevance to φ by interpreting it as a transformation to a Lagrangian coordinate system where the discontinuities do not move in \mathcal{Z} [28, 39]. This robs $g(\cdot, z)$ of its *transport-dominant* nature and makes it well-approximable in a low-dimensional linear space.

The algorithms discussed earlier provide the data-driven approximation $g(\cdot, z) \approx g_n(\cdot, z)$, where $g_n(\cdot, z)$ belongs to the POD space constructed using the snapshots of $g(\cdot, z)$. Let us recall that our goal was to approximate the solution $u(\cdot, z)$. Therefore, we need to circle back and recover an approximation for $u(\cdot, z)$ from $g_n(\cdot, z)$. This is what we refer to as de-transformation. De-transformation requires an inversion of $\varphi(\cdot, z_{\text{ref}}, z)$. Since an exact inversion is prohibitively expensive, we propose an efficient GPR-based technique to approximate $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$. Analogues to the approximation for $g(\cdot, z)$, our technique first constructs POD basis for $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ and then uses GPR to approximate the POD coefficients.

1.2 Intrusive vs. non-intrusive approach

Comments that motivate a non-intrusive approach are in order. We particularly emphasize on the first two points, which, we believe, are exclusive to problems that exhibit parameter-dependent jump-discontinuities.

1. Firstly, an intrusive approach derives a Lagrangian pPDE for $u(\varphi(\cdot, z_{\text{ref}}, z), z)$ and reduces it via a Galerkin projection over the POD modes [28, 39]. Even for an affine-in-parameter pPDE, the Lagrange equations can be non-affine. This makes the reduced-order model as expensive as the high-fidelity solver. To gain efficiency, one resorts to hyper-reduction, which adds a layer of approximation and complexity to the MOR technique. In contrast, decoupled from the underlying pPDE, the non-intrusive technique treats the affine and non-affine parameter dependence alike—similar comments holds for non-linear pPDEs. Note that rather than transforming to Lagrangian coordinates, one can directly reduce the pPDE in Eulerian coordinates [29, 35]. In Eulerian coordinates, the reduced approximation space is non-linear, and gaining efficiency requires sophisticated non-standard hyper-reduction techniques.
2. Secondly, the Lagrangian pPDE contains the inverse of the derivatives of $\varphi(\cdot, z_{\text{ref}}, z)$. If ill-conditioned, we speculate, these terms can result in stability issues with the Galerkin projection.
3. Lastly, the underlying pPDE might be unavailable or the discrete finite-difference/element operators might be inaccessible. The first scenario, for instance, corresponds to solution snapshots collected from experiments, and the latter corresponds to legacy codes or commercial solvers that provide only the solution and not the discrete evolution operators.

1.3 Relation to previous works

Particularly in the context of hyperbolic pPDEs with moving discontinuities, non-intrusive MOR techniques are not new in the literature. Authors in [17] embed the solution manifold in the Wasserstein space and approximate the solution using Wasserstein barycentres. The scheme works well for conservative hyperbolic pPDE. However, extensions to multi-dimensional spatial domains and non-conservative pPDEs with boundary conditions are unavailable, as of yet. Closer to our approach is the transformed-snapshot-interpolation (TSI) proposed in [44]. TSI is an Eulerian method that approximates $u(\cdot, z)$ using Lagrange polynomial interpolation over the transformed snapshots. The sample parameters must lie on a tensorized grid over \mathcal{Z} . In contrast, our method is Lagrangian and allows for a POD based approximation. The sample parameters need not be tensorized and can result from any of the sampling techniques summarized in [33]. This is particularly appealing for high-dimensional parameter domains where tensorized grids result in a large number of parameter samples, making the offline step unaffordable. Furthermore, a GPR is probabilistic and provides a confidence region that quantifies the quality of regression. Such a quantification can either be used to increase the size of the training set [34], to quantify the extrapolation capabilities [26], or to develop an error model [19]. A Lagrange polynomial interpolation is deterministic and does not offer such flexibility.

The novelty of our work is in combining the regression based MOR techniques developed earlier in [19, 22] with two additional steps: transformation and de-transformation. Thereby, we extend the validity of these techniques to pPDEs that exhibit parameter-dependent jump-discontinuities. We emphasize that these additional steps do not interfere with a pre-existing numerical implementation of the earlier mentioned two algorithms—recall that these algorithms were the building

blocks of the MOR techniques developed in [19, 22]. Therefore, having implemented the (de-)transformation step, a pre-existing numerical implementation can be smoothly extended to accommodate parameter-dependent jumps. Furthermore, as we clarify later, both of these steps can be easily deactivated for problems that do not exhibit parameter-dependent jumps or steep gradients and thus, the extra cost associated with (de-)transformation can be avoided for such problems. These problems can be identified by first passing the solution snapshots through a shock detector [38]. Our recommendation is to deactivate the above two steps in case the shock detector returns an empty set.

Our transformation step is inspired by the registration-based MOR technique developed in [39, 40]. Indeed, to compute the transform φ , we use the same image registration technique as that developed in [39]. Nonetheless, there are some key differences that we outline as follows. Firstly, our technique is non-intrusive as opposed to the intrusive technique developed in [39, 40]. Secondly, and most importantly, we also perform the de-transformation step to recover an approximation for the (untransformed) solution. To the best of our knowledge, authors in [39, 40], cater to developing an accurate approximation for the transformed solution and do not perform any de-transformation. The L^2 -error in approximating the (untransformed) solution is computed using the Jacobians of φ , which does not require an explicit computation of φ^{-1} . In our framework, we explicitly approximate φ^{-1} and recover an (explicit) approximation for the solution. We acknowledge that the possibility of de-transformation, and the associated difficulties, were discussed in Remark-3.3 of [39]. Our de-transformation step is a possible solution to the problem posed in this remark.

1.4 Organization

Rest of the article is organized as follows. [Section 2](#) provides a brief summary of the GPR employed. [Section 3](#) outlines a data-driven technique to approximate the transformed solution described above. [Section 4](#) outlines the de-transformation step and presents an efficient computation of $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$. [Section 5](#) presents a summary of our technique. [Section 6](#) presents numerical results, and [Section 7](#) closes the article with a conclusion.

2 Gaussian process regression (GPR)

The forthcoming sections will extensively use GPR and for completeness, we summarize it here. We refer the reader to the book [34] for an exhaustive discussion. GPR solves the following regression problem: Given the training data $\{(x_i, h(x_i))\}_{i=1,\dots,m_{\text{tr}}}$, where $x_i \in \mathbb{R}$ and $h : \mathbb{R} \rightarrow \mathbb{R}$, approximate $h(x^*)$, where $x^* \notin \{x_i\}_i$. For convenience, we collect the training points in the set

$$\mathcal{D}_x := \{x_i\}_i, \quad (5)$$

and denote the approximation via

$$h(x^*) \approx \mathcal{R}_\lambda(\theta, \mathcal{D}_x, x^*), \quad (6)$$

where $\mathcal{R}_\lambda(\theta, \mathcal{D}_x, x^*)$ is the GPR model, θ is a hyper-parameter, and λ is a user-defined parameter—below, we clarify the definition of θ and λ .

The GPR model follows from a two-step online-offline decomposition based procedure:

1. Offline, the training phase computes θ using the training data;
2. Online, the prediction phase assigns a value to the GPR model at a given $x^* \in \mathbb{R}$.

We start with the details of the training phase.

2.1 Training a GPR

Training a regression model corresponds to computing its hyper-parameters that we collect in the vector $\theta \in \mathbb{R}^r$. The value of r and the type of θ is regression technique-dependent. To present a GPR's hyper-parameters, we first define a few objects. A GPR models $h(x)$ as a Gaussian process (GP) meaning that for any $m \in \mathbb{N}$, the vector $(h(x_1), \dots, h(x_m))^T$ is normally distributed. A precise definition of a GP is as follows.

Definition 1 (Gaussian process (GP)) For some $m \in \mathbb{N}$, let $\mathcal{D} = \{x_i\}_i$ be a set of m points in \mathbb{R} . Let $\vartheta : \mathbb{R} \rightarrow \mathbb{R}$ be a mean function, and let $\zeta : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ be a positive definite kernel function. Let $h : \mathbb{R} \rightarrow \mathbb{R}$ be a random function, and let $h_{\mathcal{D}} \in \mathbb{R}^m$ be a random vector such that

$$(h_{\mathcal{D}})_i := h(x_i), \quad \forall i \in \{1, \dots, m\}. \quad (7)$$

Then, $h(x)$ is a Gaussian process if $h_{\mathcal{D}}$ follows a multivariate normal distribution with the mean vector $\vartheta_{\mathcal{D}} \in \mathbb{R}^m$ and the covariance matrix $\zeta_{\mathcal{D}\mathcal{D}} \in \mathbb{R}^{m \times m}$ given as

$$\vartheta_{\mathcal{D}} := (\vartheta(x_1), \dots, \vartheta(x_m))^T, \quad (\zeta_{\mathcal{D}\mathcal{D}})_{ij} := \zeta(x_i, x_j), \quad \forall i, j \in \{1, \dots, m\}. \quad (8)$$

In practise, to well-define GPR, we restrict to a parametrized sub-class of mean and kernel functions. For simplicity, we do not introduce any new notations and denote a mean function and a kernel function of this sub-class via $\vartheta(\cdot, \beta)$ and $\zeta(\cdot, \cdot, \kappa)$, respectively, where β and κ are the parameters. A first-order polynomial in \mathbb{R} is our mean function $\vartheta(\cdot, \beta)$, and the kernel function $\zeta(\cdot, \cdot, \kappa)$ is the automatic relevance determination (ARD) squared exponential (SE) kernel. Explicit forms read

$$\vartheta(x, \beta) = \beta^T \Phi(x), \quad \zeta(x, y, \kappa) = \kappa_1^2 \exp\left(-\frac{1}{2} \frac{(x-y)^2}{\kappa_2^2}\right). \quad (9)$$

The above choice of the kernel function works well for a function $h(x)$ that is smooth [23]—for our applications, the smoothness property indeed holds true. The function $\Phi(x) := (1, x)^T$ maps the sample space to the feature space. The vectors $\beta \in \mathbb{R}^2$ and $\kappa := (\kappa_1, \kappa_2)^T$ together form the hyper-parameter given as

$$\theta := (\beta^T, \kappa^T)^T. \quad (10)$$

To estimate θ , we consider the Bayesian approach of maximum likelihood estimation (MLE). We compute θ such that the joint normal distribution corresponding to the random vector $h_{\mathcal{D}_x}$ has the maximum possible log-likelihood. This approach, along with the fact that $h_{\mathcal{D}_x} \sim \mathcal{N}(\vartheta_{\mathcal{D}_x}, \zeta_{\mathcal{D}_x \mathcal{D}_x})$, leads to the maximization problem

$$\begin{aligned} \theta = \arg \max_{\theta^* \in \mathbb{R}^4} & \left(-\frac{1}{2} (h_{\mathcal{D}_x} - (\beta^*)^T \Phi_{\mathcal{D}_x})^T \zeta_{\mathcal{D}_x \mathcal{D}_x} (\kappa^*)^{-1} (h_{\mathcal{D}_x} - (\beta^*)^T \Phi_{\mathcal{D}_x}) \right. \\ & \left. - \frac{1}{2} \log |\zeta_{\mathcal{D}_x \mathcal{D}_x} (\kappa^*)| \right). \end{aligned} \quad (11)$$

We train the GPR using the `fitgrp` routine in MATLAB.

2.2 Prediction with GPR

The GPR model $\mathcal{R}_{\lambda}(\theta, \mathcal{D}_x, x^*)$ follows from the conditional probability $h(x^*)|h_{\mathcal{D}_x}$ i.e., the probability of $h(x^*)$ given the observed data $h_{\mathcal{D}_x}$. Since $h(x)$ is a GP, the conditional probability $h(x^*)|h_{\mathcal{D}_x}$ is normally distributed with the mean-value $\vartheta^*(\theta, \mathcal{D}_x, x^*) \in \mathbb{R}$ and the covariance $\zeta^*(\theta, \mathcal{D}_x, x^*) \in \mathbb{R}$ given as

$$\begin{aligned} \vartheta^*(\theta, \mathcal{D}_x, x^*) &:= \vartheta(x^*, \beta) + \zeta_{x^* \mathcal{D}_x} (\kappa) \zeta_{\mathcal{D}_x \mathcal{D}_x} (\kappa)^{-1} (h_{\mathcal{D}_x} - \vartheta_{\mathcal{D}_x} (\beta)), \\ \zeta^*(\theta, \mathcal{D}_x, x^*) &:= \zeta_{\mathcal{D}_x x^*} (\kappa) - \zeta_{x^* \mathcal{D}_x} (\kappa) \zeta_{\mathcal{D}_x \mathcal{D}_x} (\kappa)^{-1} \zeta_{\mathcal{D}_x x^*} (\kappa). \end{aligned} \quad (12)$$

Recall that ϑ is the mean function given in (9), and $\zeta_{\mathcal{D}_x \mathcal{D}_x}$ is the covariance matrix corresponding to the kernel function. We set

$$\mathcal{R}_{\lambda}(\theta, \mathcal{D}_x, x^*) = \vartheta^*(\theta, \mathcal{D}_x, x^*) + \lambda \sqrt{\zeta^*(\theta, \mathcal{D}_x, x^*)}. \quad (13)$$

Thus, $\mathcal{R}_{\lambda}(\theta, \mathcal{D}_x, x^*)$ is λ standard-deviations (given by $\sqrt{\zeta^*(x^*, \mathcal{D}_x)}$) away from the mean. For $\lambda = 0$, we recover the so-called mean-value prediction. We use the `predict` function from MATLAB to compute $\mathcal{R}_{\lambda}(\theta, \mathcal{D}_x, x^*)$.

3 Approximation of the transformed solution

3.1 Problem description

Consider a general pPDE of the form

$$\mathcal{L}(u(x, z), z) = 0 \quad \forall (x, z) \in \Omega \times \mathcal{Z}, \quad (14)$$

where $\mathcal{L}(\cdot, z)$ is some spatio-temporal differential operator, and $(x, z) \mapsto u(x, z) \in \mathbb{R}$ is the solution. For simplicity of notation, we consider a scalar-valued solution. For a vector-valued $u(\cdot, z)$, the same technique applies to each of its components. We assume that for all $z \in \mathcal{Z}$, $u(\cdot, z) \in L^2(\Omega)$. The set $\mathcal{Z} \subset \mathbb{R}^p$ is the parameter-domain, which can (and for the later test cases will) include the temporal domain. We assume that \mathcal{Z} is (or could be mapped via a bijection to) a hyper-cube. It is noteworthy that $\mathcal{L}(\cdot, z)$ can be non-linear or $\mathcal{L}(u(x, z), \cdot)$ could be non-affine. Our technique treats linear, non-linear, affine and non-affine problems alike.

We restrict to a square spatial domain i.e., $\Omega = (0, 1)^2$. The only (major) complexity introduced by a general curved domain is in the computation of the spatial transform φ given in (4). Other than that, the entire technique remains the same—we refer to [41] for the computation of φ on curved domains. As for now, we refrain from introducing this additional complexity and study the performance of our method on a unit square.

We denote a high-fidelity approximation of the above pPDE via

$$u(\cdot, z) \approx u_N(\cdot, z) \in \mathcal{X}_N \subset L^2(\Omega), \quad (15)$$

where \mathcal{X}_N is the high-fidelity approximation space with $\dim(\mathcal{X}_N) = N$. Usually, \mathcal{X}_N is a finite-element/volume type space. We define \mathcal{X}_N over a shape-regular discretization of Ω defined as

$$\Omega = \bigcup_{i=1}^N \mathcal{I}_i, \quad (16)$$

where \mathcal{I}_i represents the i -th spatial cell. Note that for the sake of notational simplicity and consistency with our numerical experiments, the number of grid cells equals the dimensionality of \mathcal{X}_N ; in general, these two numbers can also be different. With $\Pi : L^2(\Omega) \rightarrow \mathcal{X}_N$ we denote the orthogonal projection operator.

We consider problems where, at least for some $x \in \Omega$, the function $u(x, \cdot)$ has jump-discontinuities or steep-gradients. As explained earlier, for such problems, we first apply the MOR technique (outlined in Algorithm 1 and Algorithm 2) to a transformed solution given as

$$g(\cdot, z) = \Pi u_N(\varphi(\cdot, z_{\text{ref}}, z), z), \quad (17)$$

where $\varphi(\cdot, z_{\text{ref}}, z) : \Omega \rightarrow \Omega$ is the spatial transform, and $z_{\text{ref}} \in \mathcal{Z}$ is the reference parameter, the choice of which will be discussed later. We recall that the spatial transform is such that the transformed solution (at least ideally) is not discontinuous along the parameter domain. This allows (along with some additional regularity assumption) for an accurate approximation in a sufficiently low-dimensional linear reduced space—further details can be found in [36, 44]. The following discussion elaborates on the different steps involved in a data-driven approximation of the transformed solution.

3.2 Snapshots of the transformed solution

We collect $m_{\text{tr}} \in \mathbb{N}$ different parameter samples in a set denoted by

$$\mathcal{D}_z := \{z^{(i)}\}_{i=1, \dots, m_{\text{tr}}}. \quad (18)$$

These samples can either be chosen uniformly, randomly, or with the Lattice hyper-cube sampling technique given in [27]. At all of these parameter samples, we need snapshots of $g(\cdot, z)$. This entails computing snapshots of the high-fidelity solution $\{u_N(\cdot, z^{(i)})\}_i$ and the spatial transform $\{\varphi(\cdot, z_{\text{ref}}, z^{(i)})\}_i$. As stated earlier, the former we compute in \mathcal{X}_N . For the latter, we use an optimization-based image registration technique summarized below. We refer to the review paper [37] and the article [39] for an image analysis and a MOR perspective, respectively, on the registration technique.

3.2.1 Snapshots of φ

We express $\varphi(\cdot, z_{\text{ref}}, z)$ as

$$\varphi(\cdot, z_{\text{ref}}, z) = \text{Id} + \Psi(\cdot, z_{\text{ref}}, z). \quad (19)$$

The function $\Psi(\cdot, z_{\text{ref}}, z)$ is referred to as the displacement field—it displaces a point x by a distance of $\Psi(x, z_{\text{ref}}, z)$. Furthermore, the identity mapping is denoted by Id . [Remark 1](#) below motivates the above splitting.

We seek a $\Psi(\cdot, z_{\text{ref}}, z)$ that lies in a span of polynomials $\mathcal{P}_M := \mathbb{P}_M \times \mathbb{P}_M$ that reads

$$\mathbb{P}_M := \text{span}\{L_{ij}\Upsilon\}_{i,j=1,\dots,M}. \quad (20)$$

The function $L_{ij}(x)$ is a product of the Legendre polynomials $l_i(x_1)$ and $l_j(x_2)$, where i and j denote the degrees of the respective Legendre polynomials. Thus, \mathcal{P}_M a $2M^2$ -dimensional space. Furthermore, the function $\Upsilon(x) := \prod_{k=1}^2 x_k(1 - x_k)$ ensures that for all $x \in \partial\Omega$, we have the boundary conditions $\Psi(\partial\Omega, z_{\text{ref}}, z) = 0$. Note that following [29, 44], we have imposed a stricter set of boundary conditions than in [39]. At least for the test cases we considered, these boundary conditions provide reasonable results without any additional constraints on the Jacobian of φ . We will later study these Jacobians empirically.

The expansion coefficients for $\Psi(\cdot, z_{\text{ref}}, z)$ result from the minimization problem

$$\Psi(\cdot, z_{\text{ref}}, z) = \arg \min_{\Psi^* \in \text{span}(\mathcal{P}_M)} \mathcal{F}(\Psi^*, z_{\text{ref}}, z). \quad (21)$$

where

$$\mathcal{F}(\Psi^*, z_{\text{ref}}, z) := \mathcal{M}(\Psi^*, z, z_{\text{ref}})^2 + \epsilon \times \mathcal{R}(\Psi^*)^2, \quad (22)$$

with $\epsilon \geq 0$ being a penalty parameter. We minimize a summation of two objects, the so-called matching criterion \mathcal{M} and the regularizer \mathcal{R} . The matching criterion (a term we borrow from image analysis [37]) should be chosen such that the transformed solution $g(\cdot, z)$ is sufficiently regular along the parameter domain, making it well approximable in a linear reduced space. An appropriate choice of the matching criterion requires some information of the underlying physical process. For instance, in case of hyperbolic equations, \mathcal{M} can be chosen as the $L^2(\Omega)$ distance between the transformed and the reference snapshots [29, 39, 40, 44]. [Section 6](#) further elaborates on the choice of \mathcal{M} . Note that the term matching criterion is justified for \mathcal{M} in the sense that minimizing $\mathcal{M}(\Psi^*, z_{\text{ref}}, z)$ matches the discontinuities between snapshots, which induces regularity in the transformed solution $g(\cdot, z)$.

The regularizer $\mathcal{R}(\Psi^*)$ penalizes the spatial regularity of Ψ^* . Thereby, preventing spurious oscillations and promoting a diffeomorphic $\varphi(\cdot, z_{\text{ref}}, z)$. Several previous works deem the diffeomorphism property desirable [35, 39, 44]—[Remark 1](#) below provides further elaboration. Furthermore, empirically, one observes that the optimization routine used to compute the above problem provides a more stable and better solution with spatial regularization [24]. Here, we make the standard choice [37]

$$\mathcal{R}(\Psi^*) := \|\Delta\Psi^*\|_{L^2(\Omega; \mathbb{R}^2)}, \quad (23)$$

where Δ represents the Laplace operator.

Despite the regularization term, computing the above minimization problem is a challenging task, with the (probable) non-convexity of the objective functional being a major concern. Owing to the limited scope of this article, we refrain from devising a specialized optimization toolbox for the above problem. Rather, following the works in [29, 39], we resort to the standard interior point algorithm implemented in the `fmincon` routine of MATLAB. At least for the test cases presented later, this routine provides reasonable results with all the parameter values set to their default.

Following comments cater to the several practical considerations one encounters while solving the above problem.

- 1. Initial guess:** An initial guess of $\Psi^* = 0$ can be far-off the desired solution, resulting in inaccuracies—due to the possible non-convexity of the objective functional, the optimization

algorithm can get stuck in an undesirable local-minimum. Therefore, we first solve a few sub (optimization) problems and use their solution as an initial guess. Section-3.1.2 of [39] presents the details, which are not repeated here for brevity.

2. **Choice of z_{ref} :** The reference parameter z_{ref} is largely determined by the *discontinuity-topology* i.e., the number and the relative orientation of the discontinuities [43, 44]. With a parameter invariant discontinuity-topology, in theory, any $z_{\text{ref}} \in \mathcal{Z}$ suffices. In case the topology changes along \mathcal{Z} , one can either: (i) partition \mathcal{Z} such that the topology is preserved on each of the subsets [36]; or (ii) choose multiple reference parameters and optimize over both $\Psi(\cdot, z_{\text{ref}}, z)$ and the span of reference snapshots [40]. Following [29, 44], for now, we restrict to the examples where the topology is parameter invariant, and choose z_{ref} as the center of \mathcal{Z} . Studies in [39] indicate that the result of the registration technique might change with z_{ref} . However, trying to optimize over z_{ref} introduces additional complexity to the numerical scheme; therefore, for now, we fix a value for z_{ref} and accept the results our choice provides.
3. **Choice of ϵ :** Intuitively, it seems reasonable to choose a non-zero ϵ smaller than one. Otherwise, we will heavily penalize the regularity of $\Psi(\cdot, z_{\text{ref}}, z)$ at the expense of an accurate solution transformation. Empirically, we observe that with minor differences, all the different ϵ in the set $\{10^{-4}, 10^{-3}, 10^{-2}\}$ provide reasonable results. In our numerical experiments we set $\epsilon = 10^{-2}$.
4. **Choice of M :** We choose M iteratively. Corresponding to $\Psi(\cdot, z_{\text{ref}}, z) \in \text{span}(\mathcal{P}_M)$, consider the parameter-average of the \mathcal{M} -mismatch between the transformed and the reference solution defined as

$$\Xi_M := \frac{1}{m_{\text{tr}}} \sum_{z \in \mathcal{D}_z} \mathcal{M}(\Psi(\cdot, z_{\text{ref}}, z), z, z_{\text{ref}}). \quad (24)$$

Starting with an initial guess of $M = 1$, we continue to increase M till, for some user-defined TOL_M , we satisfy

$$\frac{|\Xi_M - \Xi_{M-1}|}{\Xi_{M-1}} \leq \text{TOL}_M. \quad (25)$$

We set $\text{TOL}_M = 10^{-3}$. In our experience, decreasing TOL_M further offered minuscule improvements at an additional computational cost.

Remark 1 (Diffeomorphic φ) *Instead of directly approximating $\varphi(\cdot, z_{\text{ref}}, z)$, we approximate the displacement field $\Psi(\cdot, z_{\text{ref}}, z)$ in the polynomial space \mathcal{P}_M . The reason being that in case $\Psi(\cdot, z_{\text{ref}}, z)$ is small as compared to Id , we can expect a diffeomorphic $\varphi(\cdot, z_{\text{ref}}, z)$, which, as detailed in [36, 39, 44], is a desirable property. In the image registration literature, such a displacement is referred to as a small-displacement.*

At least the test cases considered later exhibit small-displacements. Obviously, in general, a small-displacement is not guaranteed and therefore, the above technique does not guarantee a diffeomorphic $\varphi(\cdot, z_{\text{ref}}, z)$. This is a limitation of the technique that might be resolved by instead approximating the velocity field induced by φ —see [5].

Remark 2 (Recovery of the standard approach) *Note that by choosing*

$$\mathcal{M}(\Psi^*, z, z_{\text{ref}}), \mathcal{R}(\Psi^*) = 0, \quad (26)$$

we find that $\varphi(\cdot, z_{\text{ref}}, z) = \text{Id}$, for all $z \in \mathcal{Z}$. Consequently, we recover the regression based MOR technique developed in [19]. This justifies our earlier claim that the transformation and the de-transformation step can be easily deactivated for problems that do not exhibit parametric jump-discontinuities. Such problems can be identified by first passing the snapshots through a shock (or discontinuity or steep gradient) detector—the work in [42] presents one of the many discontinuity detectors. Our recommendation is to make the above choice for \mathcal{M} and \mathcal{R} in case the shock detector returns an empty set.

3.3 Computing the POD basis

Using the snapshots $\{g(\cdot, z^{(i)})\}_i$, we compute the POD basis

$$\mathcal{X}_N = \{v_i\}_{i=1,\dots,n}. \quad (27)$$

The computation relies on the singular value decomposition (SVD) of a snapshot matrix $\mathcal{S}_G \in \mathbb{R}^{N \times m_{\text{tr}}}$ given as

$$\mathcal{S}_G := \left(G(z^{(1)}), \dots, G(z^{(m_{\text{tr}})}) \right), \quad (28)$$

where the vector $G(z) \in \mathbb{R}^N$ contains the degrees-of-freedom (Dofs) of the transformed solution $g(\cdot, z)$. SVD of the snapshot matrix provides

$$\mathcal{S}_G = \mathcal{U}\Sigma\mathcal{V}^T,$$

where $\mathcal{U} \in \mathbb{R}^{N \times N}$ and $\mathcal{V} \in \mathbb{R}^{m_{\text{tr}} \times m_{\text{tr}}}$ are orthogonal matrices containing the left and the right singular vectors of \mathcal{S}_G , respectively. Furthermore, $\Sigma \in \mathbb{R}^{N \times m_{\text{tr}}}$ contains, at its diagonals, the singular values of \mathcal{S}_G that we assume are arranged in a descending order.

Let the vector $V_i \in \mathbb{R}^N$ contain the Dofs of $v_i \in \mathcal{X}_N$. We set V_i to be the i -th column of the matrix \mathcal{U} . Equivalently,

$$(V_i)_j = \mathcal{U}_{ji}, \quad \forall i \in \{1, \dots, n\}, j \in \{1, \dots, N\}.$$

Using the Schmidt-Eckart-Young theorem (see [16]), we can quantify the best-approximation error of approximating the snapshots in the POD basis. Collecting all the POD modes in a matrix

$$X_n := (V_1, \dots, V_n), \quad (29)$$

a bound for the projection error reads

$$E_n^{\text{proj}}(\mathcal{S}_G) := \frac{1}{\|\mathcal{S}_G\|_F} \|\mathcal{S}_G - X_n X_n^T \mathcal{S}_G\|_F = \frac{1}{\|\mathcal{S}_G\|_F} \sqrt{\sum_{i=n+1}^{\min(N, m_{\text{tr}})} \sigma_i^2}, \quad (30)$$

where $\|\cdot\|_F$ represents the Frobenius norm of a matrix, and σ_i is the i -th singular value of \mathcal{S}_G . Later (in Section 6), using numerical experiments, we will study how the POD projection error $E_n^{\text{proj}}(\mathcal{S}_G)$ decays with n .

3.4 GPR for the POD coefficients

Let $\alpha_g(z) \in \mathbb{R}^n$ denote a vector containing the POD coefficients of $g(\cdot, z)$. The crux of our technique is that for any given $z \in \mathcal{Z}$, we approximate the vector $\alpha_g(z)$ using GPR via $\alpha(z) \approx \alpha^{\mathcal{R}}(z)$ and recover the approximation

$$g(\cdot, z) \approx g_n(\cdot, z) := \sum_{i=1}^n \alpha_{g,i}^{\mathcal{R}}(z) v_i \in \text{span}(\mathcal{X}_n). \quad (31)$$

The discussion below outlines a technique to compute $\alpha_g^{\mathcal{R}}(z)$.

As explained in Section 2, GPR relies on an offline training and an online prediction step. The former computes the hyper-parameters of a GPR, whereas the latter, for any $z \in \mathcal{Z}$, predicts a value for $\alpha_g(z)$. Following are the details of these two steps.

1. **Training step:** We orthogonally project each of the snapshots in $\{g(\cdot, z^{(i)})\}_i$ onto the corresponding POD basis vectors $\{v_i\}_i$ and compute the POD coefficients given as

$$\alpha_{g,i}(z^{(j)}) = \frac{\langle v_i, g(\cdot, z^{(j)}) \rangle_{L^2(\Omega)}}{\|v_i\|_{L^2(\Omega)}^2}, \quad \forall i \in \{1, \dots, n\}, j \in \{1, \dots, m_{\text{tr}}\}. \quad (32)$$

Note that, by definition, the basis vector $\{v_i\}_i$ are L^2 -orthogonal. The above computation provides the training data $\{(z^{(j)}, \alpha_g(z^{(j)}))\}_j$, which we use to train a GPR. For each component of $\alpha_g(z)$, we train a separate uncorrelated GPR—Remark 3 below elaborates on this further. As Section 2.1 explains, training a GPR for the i -th component of $\alpha_g(z)$ entails computing the hyper-parameter $\theta_{\alpha,i} \in \mathbb{R}^{2 \times (p+1)}$ with p being the dimension of \mathcal{Z} .

2. Prediction step: With the hyper-parameters $\{\theta_{\alpha,i}\}_i$ at hand, we consider the mean-value prediction for $\alpha_g(z)$ that reads

$$\alpha_{g,i}(z) \approx \alpha_{g,i}^{\mathcal{R}}(z) := \mathcal{R}_0(\theta_{\alpha,i}, \mathcal{D}_z, z), \quad \forall i \in \{1, \dots, n\}. \quad (33)$$

The above approximation finally provides the POD approximation $g_n(\cdot, z)$ given in (31). Recall that \mathcal{D}_z is a set containing the parameter samples and is defined in (18). Furthermore, \mathcal{R} denotes the GPR model given in (13).

Remark 3 (GPR for vector-valued functions) While training the GPR, we assume that $\alpha_g(z)$ has uncorrelated components [19, 26]. This certainly introduces some inaccuracies because for most pPDEs, $\alpha_g(z)$ has coupled correlated components. As of yet, particularly for non-linear pPDEs, it is unclear how one can account for this coupling using the cross-correlation technique proposed in [2].

Remark 4 (Extrapolation capabilities) Regression is known to be inaccurate with extrapolation [19, 22, 34]. Our MOR technique is no exception to this limitation. Heuristics suggest (see [26]) that the variance of the GP can be used to quantify the extrapolation capabilities, but the success of such a technique—particularly for hyperbolic pPDEs considered in our numerical experiments—is unclear. We include the corners of the parameter domain \mathcal{Z} in the parameter samples and avoid extrapolation altogether.

We emphasize that our non-intrusive technique shares this limitation with the intrusive techniques proposed in [29, 39]. In entirety, these intrusive techniques are only partially intrusive—regression is used to compute the spatial transform φ , which introduces inaccuracies during extrapolation.

4 Approximation of the untransformed solution

We want to recover an approximation for $u_N(\cdot, z)$ from the POD approximation of the transformed snapshot $g_n(\cdot, z)$ given in (31). We emphasize that this recovery needs to be performed online. At first glance, the approximation $u_N(\cdot, z) \approx \Pi g_n(\varphi(\cdot, z_{\text{ref}}, z)^{-1}, z)$ seems reasonable, with $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ computed using the non-linear least-squares problem

$$\varphi(x, z_{\text{ref}}, z)^{-1} := \arg \min_{y \in \Omega} \|\varphi(y, z_{\text{ref}}, z) - x\|_{l^2}^2. \quad (34)$$

However, a solution to the above problem comes at a high cost [30]. We need $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ inside all the N spatial grid cells i.e., we need to solve the above problem at least $\mathcal{O}(N)$ times, with each solution requiring a few iterations. This procedure can easily dominate the cost of our MOR technique and can make it more expensive than a high-fidelity solver.

For the above reason, we refrain from solving the least-squares problem online and instead approximate its solution in the POD basis. We follow the same line of procedures as that used to approximate the transformed snapshot $g(\cdot, z)$ in Section 3. In the offline phase, we solve the above problem and collect snapshots of φ^{-1} , compute the POD basis, collect training data for the POD coefficients and train a GPR. In the online phase, we query the GPR for the POD coefficients and recover an approximation for φ^{-1} . The details are as follows:

4.1 Snapshots of φ^{-1}

We solve the above least-squares problem and collect the snapshots $\{\varphi(\cdot, z_{\text{ref}}, z^{(i)})^{-1}\}_{i=1, \dots, m_{\text{tr}}}$; recall that $\{z^{(i)}\}_i$ are the parameter samples defined in (18). Similar to the solution $u(\cdot, z)$, we compute these snapshots in a finite-dimensional high-fidelity approximation space.

As Remark 1 states, we expect $\varphi(\cdot, z_{\text{ref}}, z)$ to be a diffeomorphism. Therefore, it is reasonable to approximate $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ in a finite-element space of continuous functions. We choose this space to be the span of continuous piecewise linear functions (or the so-called hat-functions) defined over the triangulation of Ω given in (16). We denote this finite-element space by $\tilde{\mathcal{X}}_{\tilde{N}}$, which is such that $\dim(\tilde{\mathcal{X}}_{\tilde{N}}) = \tilde{N}$. Furthermore, the value of \tilde{N} equals the number of vertices in the spatial mesh.

We label the high-fidelity approximation of $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ by

$$\varphi(\cdot, z_{\text{ref}}, z)^{-1} \approx \tilde{\varphi}_N(\cdot, z_{\text{ref}}, z) \in \tilde{\mathcal{X}}_{\tilde{N}}, \quad (35)$$

and compute it by projecting $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ onto $\tilde{\mathcal{X}}_{\tilde{N}}$ i.e., we set

$$\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z) := \tilde{\Pi}(\varphi(\cdot, z_{\text{ref}}, z)^{-1}),$$

where $\tilde{\Pi} : L^2(\Omega) \rightarrow \tilde{\mathcal{X}}_{\tilde{N}}$ is the orthogonal projection operator. We consider a quadrature routine to compute the projection. The value of $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$ at the quadrature points results from solving the least-squares problem given in (34); to this end, we use the `lsqnonlin` routine from MATLAB. For the experiments reported later, we use a tensorized set of 3×3 Gauss-Legendre quadrature points in each spatial cell.

4.2 Computing the POD basis

Using the snapshots $\{\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z^{(i)})\}_{i=1,\dots,m_{\text{tr}}}$ computed above, for any $z \in \mathcal{Z}$, we seek an efficient approximation of $\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z)$. As discussed in Remark 1, we prefer to split $\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z)$ as

$$\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z) = \text{Id} + \tilde{\Psi}_N(\cdot, z_{\text{ref}}, z), \quad (36)$$

and approximate the displacement field $\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z)$. In contrast to the solution $u(\cdot, z)$, which has a jump-discontinuity along \mathcal{Z} , the displacement field is sufficiently regular along \mathcal{Z} —Remark 6 below provides further elaboration. Therefore, it is reasonable to expect that it is well-approximable in a sufficiently low-dimensional linear reduced space; numerical experiments will further justify our expectations. We construct this reduced space using POD.

Before discussing further details, let us recall that $\tilde{\Psi}_N(x, z_{\text{ref}}, z)$ is a two-dimensional vector i.e.,

$$\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z) = \left(\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z), \tilde{\Psi}_N^{(2)}(\cdot, z_{\text{ref}}, z) \right)^T.$$

In the following, for brevity, we present a technique to approximate $\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z)$. The same procedure applies to $\tilde{\Psi}_N^{(2)}(\cdot, z_{\text{ref}}, z)$. Similar to the snapshot matrix \mathcal{S}_G defined earlier in (28), using snapshots of $\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z)$, we define a snapshot matrix $\mathcal{S}_{\tilde{\Psi}^{(1)}}$. Then, applying the SVD-based technique outlined earlier in Section 3.3, we compute the POD basis for the displacement field given as

$$\mathcal{X}_{n_{\tilde{\Psi}}} := \{v_i^{\tilde{\Psi}^{(1)}}\}_{i=1,\dots,n_{\tilde{\Psi}}},$$

where the parameter $n_{\tilde{\Psi}} \in \mathbb{N}$ denotes the dimensionality of $\mathcal{X}_{n_{\tilde{\Psi}}}$. Note that we use the same $n_{\tilde{\Psi}}$ to approximate both the components of $\tilde{\Psi}_N$.

4.3 GPR for the POD coefficients

Let $\alpha_{\tilde{\Psi}^{(1)}}(z) \in \mathbb{R}^{n_{\tilde{\Psi}}}$ denote a vector containing the POD coefficients of the displacement field $\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z)$. For any given $z \in \mathcal{Z}$, we approximate $\alpha_{\tilde{\Psi}^{(1)}}(z)$ using GPR. The procedure remains exactly the same as that used to approximate the POD coefficients of $g(\cdot, z)$ in Section 3.4. Following is a brief recall.

In the offline phase, by projecting each of the snapshots in $\{\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z^{(i)})\}_i$ onto the corresponding POD basis vector $\{v_i^{\tilde{\Psi}^{(1)}}\}_i$, we collect training data from the mapping $z \mapsto \alpha_{\tilde{\Psi}^{(1)}}(z)$. With this training data, for the i -th component of $\alpha_{\tilde{\Psi}^{(1)}}(z)$, we compute the hyper-parameter $\theta_{\tilde{\Psi}^{(1)},i}$ of the GPR. In the online phase, for any $z \in \mathcal{Z}$, we approximate the POD coefficient $\alpha_{\tilde{\Psi}^{(1)}}(z)$ using the mean-value estimate given as

$$\alpha_{\tilde{\Psi}^{(1)},i}(z) \approx \alpha_{\tilde{\Psi}^{(1)},i}^{\mathcal{R}}(z) := \mathcal{R}_0(\theta_{\tilde{\Psi}^{(1)},i}, \mathcal{D}_z, z), \quad \forall i \in \{1, \dots, n_{\tilde{\Psi}}\}, \quad (37)$$

and recover an approximation for the first-component of the displacement field

$$\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z) \approx \tilde{\Psi}_{n_{\tilde{\Psi}}}^{(1)}(\cdot, z_{\text{ref}}, z) := \sum_{i=1}^{n_{\tilde{\Psi}}} \alpha_{\tilde{\Psi}^{(1)},i}^{\mathcal{R}} v_i^{\tilde{\Psi}^{(1)}}.$$

Similarly, we can approximate the second-component $\tilde{\Psi}_N^{(2)}(\cdot, z_{\text{ref}}, z)$ and recover

$$\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z) \approx \tilde{\Psi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z) := \left(\tilde{\Psi}_{n_{\tilde{\Psi}}}^{(1)}(\cdot, z_{\text{ref}}, z), \tilde{\Psi}_{n_{\tilde{\Psi}}}^{(2)}(\cdot, z_{\text{ref}}, z) \right)^T. \quad (38)$$

This finally provides the following approximation for the spatial transform

$$\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z) \approx \tilde{\varphi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z) := \text{Id} + \tilde{\Psi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z), \quad (39)$$

that we use to approximate $u_N(\cdot, z)$ via

$$u_N(\cdot, z) \approx u_{n, n_{\tilde{\Psi}}}(\cdot, z) := \Pi g_n(\tilde{\varphi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z), z). \quad (40)$$

Recall that Π is an orthogonal projection operator from $L^2(\Omega)$ onto the high-fidelity space \mathcal{X}_N . Furthermore, $g_n(\cdot, z)$ is the reduced approximation for the transformed solution $g(\cdot, z)$ and is given in (17).

Remark 5 (Both n and $n_{\tilde{\Psi}}$ determine the approximation quality) Note that two parameters— n and $n_{\tilde{\Psi}}$ —control the accuracy of our approximation. Increasing both n and $n_{\tilde{\Psi}}$ increases the number of POD modes used to approximate the transformed solution $g(\cdot, z)$ and the displacement field $\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z)$, respectively. As one might expect, keeping one parameter fixed and increasing the other offers diminishing returns in terms of the approximation accuracy. Numerical experiments will further study this behaviour.

Remark 6 (Approximability of $\tilde{\varphi}_N$ in the POD basis) We assume that $\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z)$ is well-approximable in a POD space, which is linear by construction. The observation that for all $x \in \Omega$, $\tilde{\varphi}_N(x, z_{\text{ref}}, \cdot)$ is sufficiently regular along \mathcal{Z} motivates this assumption. As stated earlier—and further explained in [36, 44]— $\tilde{\varphi}_N(\cdot, z_{\text{ref}}, z)$ matches discontinuities between the reference solution $u_N(\cdot, z_{\text{ref}})$ and some other solution $u_N(\cdot, z)$. For several problems involving parameter-invariant discontinuity-topology (see Section 3), the locations of these spatial discontinuities vary smoothly in \mathcal{Z} , resulting in a $\tilde{\varphi}_N$ that is smooth along \mathcal{Z} . Our numerical experiments will provide further elaboration.

4.4 Error surrogate

To certify the quality of our approximation, following the works in [15, 19], we develop a surrogate for the relative L^1 -error $E(z, n, n_{\tilde{\Psi}})$ defined as

$$E(z, n, n_{\tilde{\Psi}}) := \frac{1}{\|u_N(\cdot, z)\|_{L^1(\Omega)}} \|u_N(\cdot, z) - u_{n, n_{\tilde{\Psi}}}(\cdot, z)\|_{L^1(\Omega)}, \quad (41)$$

where $u_N(\cdot, z)$ and $u_{n, n_{\tilde{\Psi}}}(\cdot, z)$ is the high-fidelity and the reduced approximation, respectively. We represent the surrogate via

$$E(z, n, n_{\tilde{\Psi}}) \approx E^{\mathcal{R}}(z, n, n_{\tilde{\Psi}}),$$

and compute it using a standard GPR-based technique. Offline, at the training points $\{z^{(i)}\}_{i=1, \dots, m_{\text{tr}}}$, we collect the training data $\{(z^{(i)}, E(z^{(i)}, n, n_{\tilde{\Psi}}))\}_i$. Using this training data, we train a GPR i.e., we compute the hyper-parameters $\theta_E \in \mathbb{R}^{2 \times (p+1)}$. Online, for any $z \in \mathcal{Z}$, we compute the surrogate $E^{\mathcal{R}}(z, n, n_{\tilde{\Psi}})$ via

$$E^{\mathcal{R}}(z, n, n_{\tilde{\Psi}}) := \mathcal{R}_2(\theta_E, \mathcal{D}_z, z), \quad (42)$$

where \mathcal{R}_2 is the GPR model given in (13). We quantify the accuracy of our error surrogate by an efficiency index $\eta(z, n, n_{\tilde{\Psi}})$ defined as

$$\eta(z, n, n_{\tilde{\Psi}}) := \frac{E^{\mathcal{R}}(z, n, n_{\tilde{\Psi}})}{E(z, n, n_{\tilde{\Psi}})}. \quad (43)$$

Later, we study $\eta(z, n, n_{\tilde{\Psi}})$ using numerical experiments.

Note that $\mathcal{R}_2(\theta_E, \mathcal{D}_z, z)$ is two standard deviations away from the mean-value and therefore, is pessimistic as compared to a mean-value estimate. Since we use the same sample parameters to compute the POD basis and then train the GPR, we cannot expect much accuracy from a mean-value prediction of $E(z, n, n_{\tilde{\Psi}})$. Therefore, we instead consider a pessimistic estimate.

5 Summary

[Algorithm 3](#) and [Algorithm 4](#) summarize the offline and the online stages of the algorithm, respectively.

Algorithm 3 Summary: Offline phase

- 1: **Input:** $n, n_{\tilde{\Psi}}, m_{\text{tr}}$
 - 2: **Output:** $\theta_\alpha, \theta_{\tilde{\Psi}}, \theta_E, \mathcal{D}_z, \{v_i\}_{i=1,\dots,n}, \{v_i^{\tilde{\Psi}}\}_{i=1,\dots,n_{\tilde{\Psi}}}$
 - 3: Collect the parameter samples $\mathcal{D}_z := \{z^{(i)}\}_{i=1,\dots,m_{\text{tr}}}$ from \mathcal{Z} .
 - 4: Compute the transformed snapshots $\{g(\cdot, z^{(i)})\}_{i=1,\dots,m_{\text{tr}}}$.
 - 5: Compute the POD basis $\{v_i\}_{i=1,\dots,n}$.
 - 6: Compute the snapshots $\{\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z^{(i)})\}_{i=1,\dots,m_{\text{tr}}}$.
 - 7: Compute the POD basis $\{v_i^{\tilde{\Psi}}\}_{i=1,\dots,n_{\tilde{\Psi}}}$.
 - 8: At the parameter samples in \mathcal{D}_z , collect the training data from the mappings $z \mapsto \alpha_g(z)$, $z \mapsto \alpha_{\tilde{\Psi}}(z)$ and $z \mapsto E(z, n, n_{\tilde{\Psi}})$ and use it to compute the hyper-parameters $\theta_\alpha, \theta_{\tilde{\Psi}}$ and θ_E , respectively, for the GPR.
-

Algorithm 4 Summary: Online phase

- 1: **Input:** $\theta_\alpha, \theta_{\tilde{\Psi}}, \theta_E, \mathcal{D}_z, \{v_i\}_{i=1,\dots,n}, \{v_i^{\tilde{\Psi}}\}_{i=1,\dots,n_{\tilde{\Psi}}}, z \in \mathcal{Z}$
 - 2: **Output:** $u_{n,n_{\tilde{\Psi}}}(\cdot, z), E^R(z, n, n_{\tilde{\Psi}})$
 - 3: Compute $\alpha_g^R(z)$ using (33) and recover $g_n = \sum_j \alpha_{g,j}^R(z) v_j$.
 - 4: Compute $\alpha_{\tilde{\Psi}}^R(z)$ using (37) and recover $\tilde{\Psi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z) := \sum_{i=1}^{n_{\tilde{\Psi}}} \alpha_{\tilde{\Psi},i}^R v_i^{\tilde{\Psi}}$.
 - 5: Compute $u_{n,n_{\tilde{\Psi}}} = \Pi g_n (\text{Id} + \tilde{\Psi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z), z)$.
 - 6: Compute $E^R(z, n, n_{\tilde{\Psi}})$ using (42).
-

5.1 Computational costs: online phase

We compute the costs of the different steps outlined in [Algorithm 4](#).

1. **line-3:** Querying the GPR model for the n components of $\alpha_g^R(z)$ requires $\mathcal{O}(m_{\text{tr}}n)$ operations [34]. Computing $g_n(\cdot, z)$ requires $\mathcal{O}(nN)$ operations.
2. **line-4:** Querying the GPR model for the $n_{\tilde{\Psi}}$ components of $\alpha_{\tilde{\Psi}}^R(z)$ requires $\mathcal{O}(m_{\text{tr}}n_{\tilde{\Psi}})$ operations. Computing $\tilde{\Psi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z)$ requires $\mathcal{O}(n_{\tilde{\Psi}}N)$ operations.
3. **line-5:** Computing $u_{n,n_{\tilde{\Psi}}}(\cdot, z)$ on a structured grid requires $\mathcal{O}(N)$ operations.
4. **line-6:** Computing the scalar $E^R(z, n, n_{\tilde{\Psi}})$ requires $\mathcal{O}(m_{\text{tr}})$ operations.

The total cost sums up to

$$\begin{aligned} \mathcal{C}_{\text{tot}}^{\text{MOR}} = & \underbrace{\mathcal{O}(m_{\text{tr}}n) + \mathcal{O}(nN)}_{\text{POD approximation of } g(\cdot, z)} + \underbrace{\mathcal{O}(m_{\text{tr}}n_{\tilde{\Psi}}) + \mathcal{O}(n_{\tilde{\Psi}}N)}_{\text{POD approximation for } \tilde{\Psi}_N(\cdot, z_{\text{ref}}, z)} \\ & + \underbrace{\mathcal{O}(N)}_{\text{Computation of } u_{n,n_{\tilde{\Psi}}}(\cdot, z)} + \underbrace{\mathcal{O}(m_{\text{tr}})}_{\text{Computation of } E^R(z, n, n_{\tilde{\Psi}})}. \end{aligned} \quad (44)$$

We study how $\mathcal{C}_{\text{tot}}^{\text{MOR}}$ scales with the parameter-domain dimension p . Assume that to collect the parameter samples from \mathcal{Z} , we take a tensor-product of $m \in \mathbb{N}$ uniformly placed parameter samples along each parameter dimension. Then, the number of parameter samples scales as $m_{\text{tr}} = \mathcal{O}(m^p)$, which provides

$$\begin{aligned} \mathcal{C}_{\text{tot}}^{\text{MOR}} = & \mathcal{O}(m^p n) + \mathcal{O}(m^p n_{\tilde{\Psi}}) + \mathcal{O}(m^p) \\ & + \mathcal{O}(nN) + \mathcal{O}(n_{\tilde{\Psi}}N) + \mathcal{O}(N). \end{aligned} \quad (45)$$

Observe that for extremely large values of p , the cost of querying a GPR—which introduces the m^p dependence above—might outweigh all the other costs. This might result in a MOR technique that is more expensive than a high-fidelity solver. Thus the practical applicability of our method is limited to moderate parameter domain dimensions. We share this limitation with the intrusive technique developed in [39]. The reason being that this technique approximates φ using RBFs (see Remark 4) which, similar to a GPR, are expensive to query for extremely large values of p .

6 Numerical Results

We abbreviate our MOR technique as GPR-TS-MOR. The abbreviation TS stands for transformed snapshots. The numerical experiments compare it to the S-PROJ technique. In S-PROJ, we collect solution snapshots (S)—do not perform any snapshot transformation—compute the POD basis and approximate the solution in the POD basis. We orthogonally project the solution onto the POD basis. The high-fidelity solver is abbreviated as HF.

6.1 Description of the test cases

We consider the following test cases involving hyperbolic (test case-1 and 2) and parabolic (test case-3) equations.

1. **Test-1 (1D wave equation):** We consider the 1D (in space) wave equation (rewritten as a first order system)

$$\partial_t u(x, t, \mu) + A \partial_x u(x, t, \mu) = 0, \quad \forall (x, t, \mu) \in \Omega \times [0, T] \times \mathcal{P}, \quad (46)$$

where $u = (u_1, u_2)^T$ is the vector-valued solution, and the matrix A reads

$$A := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (47)$$

We choose $\Omega = (-0.3, 3)$, and $T = 0.8$. As the initial data, for all $x \in \Omega$, we consider the linear superposition

$$\begin{aligned} u_1(x, t = 0, \mu) &= \frac{1}{\sqrt{2}} (w_1(x, \mu) + w_2(x, \mu)), \\ u_2(x, t = 0, \mu) &= \frac{1}{\sqrt{2}} (-w_1(x, \mu) + w_2(x, \mu)), \end{aligned} \quad (48)$$

where $w_1(\cdot, \mu)$ and $w_2(\cdot, \mu)$ are two sin-function bumps given as

$$\begin{aligned} w_1(x, \mu) &:= \mu \times (\sin(2\pi(x + 0.2)) + 1) \mathbb{1}_{[\delta_1 - 0.5, \delta_1]}(x), \\ w_2(x, \mu) &:= \mu \times (\sin(2\pi(x - 2.3)) + 1) \mathbb{1}_{[\delta_2 - 0.5, \delta_2]}(x). \end{aligned} \quad (49)$$

We set $\delta_1 := 0.3$, and $\delta_2 := 2.8$. Furthermore, we set $\mathcal{P} := [0.5, 2]$. Thus, the parameter domain is two-dimensional and reads $\mathcal{Z} = [0, T] \times \mathcal{P}$. Along the boundary $\partial\Omega \times \mathcal{Z}$, we prescribe $u = 0$.

2. **Test-2 (2D Burgers' equation):** We consider the 2D (in space) Burgers' equation given as

$$\partial_t u(x, t) + \left(\frac{1}{2}, \frac{1}{2} \right)^T \cdot \nabla u(x, t)^2 = 0, \quad \forall (x, t) \in \Omega \times [0, T]. \quad (50)$$

The initial data is a characteristic function over a square and reads

$$u(x, t = 0) = \begin{cases} 1, & x \in [0, 0.5]^2 \\ 0, & \text{else} \end{cases}. \quad (51)$$

We set $\Omega = (-0.1, 1.5)^2$ and $T = 2$. Along the boundary $\partial\Omega \times \mathcal{Z}$, we prescribe $u = 0$. Time is the sole parameter for this problem, i.e., $\mathcal{Z} = [0, T]$.

3. Test-3 (2D Heat conduction): We consider the 2D (in space) heat conduction problem from [39] given as

$$-\nabla \cdot (\beta(x, z) \nabla u(x, z)) = 1, \quad \forall (x, z) \in \Omega \times \mathcal{Z}, \quad (52)$$

where $\Omega = [0, 1]^2$, and $\mathcal{Z} = [-0.05, 0.05]$. The heat conductivity coefficient $\beta(x, z) \in \mathbb{R}$ is discontinuous in $\Omega \times \mathcal{Z}$ and reads

$$\beta(x, z) := 0.1 + 0.9 \times \mathbb{1}_{\Omega_z}, \quad \Omega_z := \left\{ x \in \Omega : \|x - \bar{x}_z\|_\infty \leq \frac{1}{4} \right\}. \quad (53)$$

The center \bar{x}_z equals $[1/2 + z, 1/2 + z]$. Along the boundary $\partial\Omega \times \mathcal{Z}$, we prescribe $u = 0$.

Remark 7 (Software and hardware details) All the simulations are run using MATLAB, in serial, and on a computer with two Intel Xeon Silver 4110 processors, 16 cores each and 92GB of RAM.

Remark 8 (HF solver) For test case-1 and 2, our HF solver is a second-order finite-volume scheme with a Van-Leer flux-limiter combined with a second-order explicit Runge-Kutta time-stepping scheme. We use the local-Lax-Friedrich numerical flux, and set the CFL number to 0.5. For test case-3, our HF solver is a continuous Galerkin \mathbb{P}^1 finite-element solver.

6.2 Choice of the matching criterion \mathcal{M}

Following the empirical success reported in [29, 39, 40, 44], for test cases 1 and 2, which involve hyperbolic equations, we choose \mathcal{M} (appearing in (21)) as the L^2 -distance between the transformed and the reference snapshot i.e.,

$$\mathcal{M}(\Psi^*, z, z_{\text{ref}}) = \|u_N(\text{Id} + \Psi^*, z) - u_N(\cdot, z_{\text{ref}})\|_{L^2(\Omega)}^2. \quad (54)$$

For test case-3, we assume that one has access to both the solution snapshots and a parametrized description of the boundary $\partial\Omega_z$. Usually this description is already available during the spatial mesh generation process. The jump in the heat conductivity coefficient along $\partial\Omega_z$ can be viewed as a change in the material properties, which is known a-priori, and might be used, for instance, to refine the mesh along the boundary $\partial\Omega_z$. With an access to the boundary $\partial\Omega_z$, we set the matching criterion to [39]

$$\mathcal{M}(\Psi^*, z, z_{\text{ref}}) = \sum_{i=1}^{N_p} \|\hat{x}_z^{(i)} - (\hat{x}_{z_{\text{ref}}}^{(i)} + \Psi^*(\hat{x}_{z_{\text{ref}}}^{(i)}))\|_2^2, \quad (55)$$

where $\{\hat{x}_z^{(i)}\}_{i=1, \dots, N_p}$ represent a set of uniformly placed points placed along $\partial\Omega_z$. We set N_p to 400. Observe that the minimization of the above matching criterion pushes the boundary $\varphi(\partial\Omega_{z_{\text{ref}}}, z_{\text{ref}}, z)$ to be aligned with that of $\partial\Omega_z$. Consequently, the transformed solution $g_N(\cdot, z)$ does not exhibit steep gradients along \mathcal{Z} .

Since the choice of \mathcal{M} is application dependent, our technique does not run in an entire black-box fashion and requires at least some information about the underlying physical process. Nevertheless, this information is often available because a user is usually aware of the physical process being simulated—even though one might not have access to the underlying discrete evolution operators.

6.3 Average error

We quantify the error over the entire parameter domain via the average relative L^1 -error $E_a(n, n_{\tilde{\Psi}})$ defined as

$$E_a(n, n_{\tilde{\Psi}}) := \frac{1}{\#\mathcal{D}_z^{tst}} \sum_{z \in \mathcal{D}_z^{tst}} E(z, n, n_{\tilde{\Psi}}). \quad (56)$$

The error $E(z, n, n_{\tilde{\Psi}})$ is as defined in (41). The test samples \mathcal{D}_z^{tst} consist of 200 uniformly and independently sampled parameters from \mathcal{Z} . Furthermore, $\#\mathcal{D}_z^{tst}$ represents the size of \mathcal{D}_z^{tst} . Replacing $E(z, n, n_{\tilde{\Psi}})$ by the surrogate $E^{\mathcal{R}}(z, n, n_{\tilde{\Psi}})$ defined in (42), we recover the approximation

$$E_a(n, n_{\tilde{\Psi}}) \approx E_a^{\mathcal{R}}(n, n_{\tilde{\Psi}}).$$

6.4 Test-1

We discretize Ω with $N = 10^3$ grid-cells, resulting in a grid size of $\Delta x = 3.3 \times 10^{-3}$. The training data \mathcal{D}_z is a set of uniformly placed 40×20 points inside \mathcal{Z} . We approximate the displacement field $\Psi(\cdot, z_{\text{ref}}, z)$ in the polynomial space $\mathcal{P}_{M=4}$, where the value of M results from the procedure outlined in [Section 3.2.1](#). Recall that the solution here is vector-valued with $u(\cdot, z) = (u_1(\cdot, z), u_2(\cdot, z))^T$. As stated earlier, we apply the technique developed in the previous sections to each of the components of $u(\cdot, z)$. For brevity, in the following, we present the results for u_1 . For u_2 , the results are similar.

6.4.1 Study of the POD projection error

Consider the POD projection error $E_n^{\text{proj}}(\mathcal{S})$ defined in [\(30\)](#). We compare this error for two different snapshot matrices, $\mathcal{S} = \mathcal{S}_G$ and $\mathcal{S} = \mathcal{S}_U$. The former is defined in [\(28\)](#) and contains transformed snapshots. The latter contains the snapshots of the (untransformed) solution and reads

$$\mathcal{S}_U := \left(U(z^{(1)}), \dots, U(z^{(m_{\text{tr}})}) \right), \quad (57)$$

where $U(z) \in \mathbb{R}^N$ contains the Dofs of $u_N^{(1)}(\cdot, z)$ with $u_N^{(1)}(\cdot, z)$ being a HF approximation to $u_1(\cdot, z)$. Note that GPR-TS-MOR and S-PROJ use the POD modes of \mathcal{S}_G and \mathcal{S}_U , respectively, to approximate the transformed and the untransformed solution, respectively.

[Figure 1a](#) presents the results. A few observations are in order. Firstly, for all $n \leq 23$, the relative error is smaller for \mathcal{S}_G . Thus, at least for these smaller values of n , snapshot transformation improves the approximability of a snapshot matrix in its POD modes. Secondly, the difference between the two relative errors is dramatic for $n \leq 10$, with $E_n^{\text{proj}}(\mathcal{S}_G)$ being at least four time smaller than $E_n^{\text{proj}}(\mathcal{S}_U)$. The difference is the most pronounced for $n = 1$, for which we find

$$E_{n=1}^{\text{proj}}(\mathcal{S}_G) \approx 0.11, \quad E_{n=1}^{\text{proj}}(\mathcal{S}_U) \approx 0.70. \quad (58)$$

We emphasize that $n = 1$ is just 0.1% of the high-fidelity space dimension N . Remarkably, for such a small value of n , snapshot transformation facilitates a relative error of just 11%. Lastly, around $n > 10$, the decay in the relative error $E_n^{\text{proj}}(\mathcal{S}_G)$ slows down. The error (almost) starts to stagnate and eventually, at $n \approx 23$, it overshoots $E_n^{\text{proj}}(\mathcal{S}_U)$ —[\[39\]](#) reports a similar behaviour.

A plausible reason for this stagnation (as discussed in [\[36\]](#)) is that the transformed snapshots have slightly misaligned discontinuities—we further showcase the misalignment below. Beyond a certain n , the error from this misalignment dominates the total error, causing (almost) an error stagnation. Usually, the misalignment is $\mathcal{O}(1/N)$ —therefore, the point of stagnation decreases upon increasing N . [Figure 1b](#) depicts this behaviour. Furthermore, the error at the stagnation point is $\mathcal{O}(1/\sqrt{N})$, which is the same order of accuracy as the HF approximation [\[36\]](#). The implication being that the error introduced by the misalignment is of little practical interest.

6.4.2 Study of the transformed solution

Let us compare the transformed solution to the untransformed one. We choose $\mu = 1$ and study the time-evolution of the transformed $g^{(1)}(\cdot, t, \mu)$ and the untransformed $u_N^{(1)}(\cdot, t, \mu)$ solution. [Figure 2](#) presents the results. The spatial discontinuities in $u_N^{(1)}(\cdot, t, \mu)$ (shown in red) move along the time-domain, resulting in poor-approximability in a linear reduced space. However, a composition with a spatial transform almost halts the temporal movement of the discontinuities. This results in a transformed solution that is well-approximable in a linear reduced space.

It is noteworthy that although the movement of discontinuities in the transformed solution is small, it is not exactly zero i.e., there is some misalignment between the discontinuities. One reason being the L^2 objective functional in [\(22\)](#), which, due to its non-convexity, does not guarantee a perfect alignment of spatial discontinuities. To further improve the alignment, one may consider the so-called geometric registration techniques discussed in [\[37\]](#). We plan to pursue such an approach in the future.

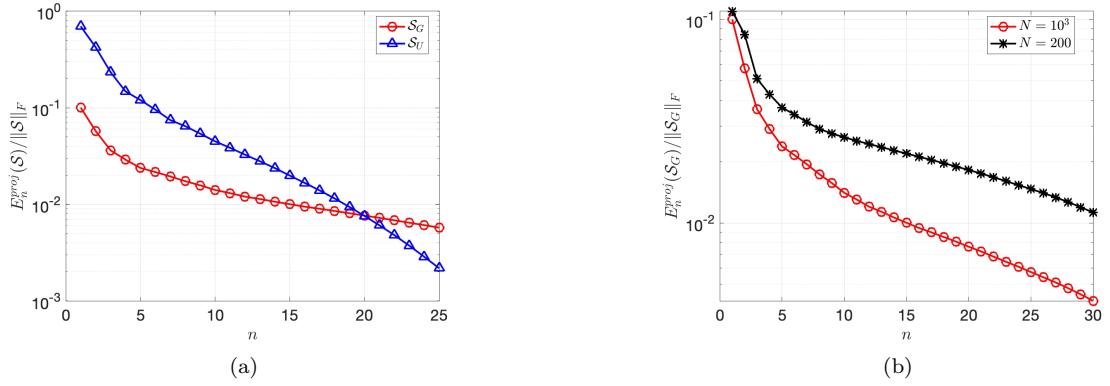


Figure 1: Results for test-1. (a) Compares the POD projection error for the transformed and the untransformed snapshot matrices. (b) Compares the POD projection error for the transformed snapshots for two different grid sizes.

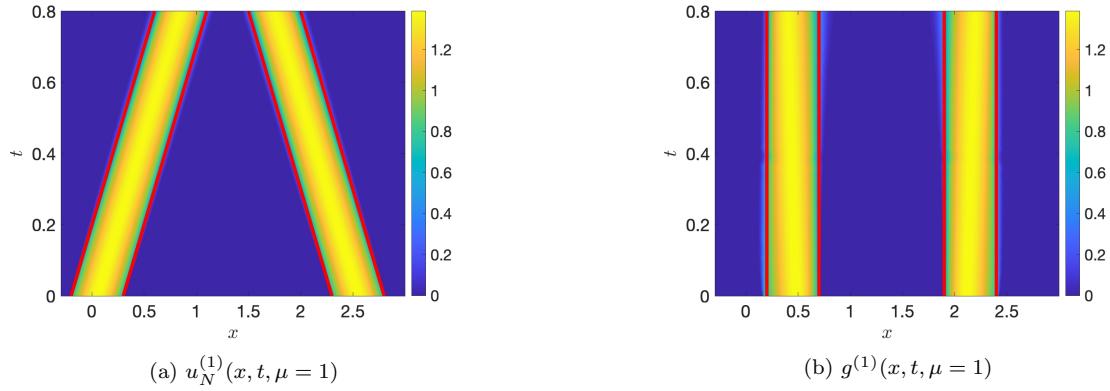


Figure 2: Results for test-1. Time-trajectory of the (a) untransformed and (b) transformed solution. Location of spatial discontinuities shown in red.

6.4.3 Study of the average error

For GPR-TS-MOR, Figure 3a presents the average error $E_a(n, n_{\tilde{\Psi}})$. We vary both n and $n_{\tilde{\Psi}}$ in the set $\{1, 2, \dots, 20\}$. For smaller values of n , the error of approximating the transformed solution outweighs the error of approximating the inverse of the spatial transform and irrespective of how large we make $n_{\tilde{\Psi}}$, it dominates the total error. Consider $n = 1$, for instance, where increasing $n_{\tilde{\Psi}}$ beyond one offers no error decrement—the error stagnates at $\approx 11\%$. Similar observation holds for a fixed $n_{\tilde{\Psi}}$ and a variable n .

As anticipated, increasing both n and $n_{\tilde{\Psi}}$ simultaneously, decreases the error monotonically. However, for larger values of n and $n_{\tilde{\Psi}}$, the error almost stagnates at 1.3%—Table 1 further highlights this stagnation. Following are two plausible reasons. First, the misalignment of spatial discontinuities referred to earlier, which stagnates the singular value decay. Second, due to a limited size of the training set \mathcal{D}_z , the GPR can only provide as much accuracy. The observation that increasing the size of the training data lowers the error at the stagnation point—see Table 1—corroborates our explanation. We emphasize that for practical purposes, an error of 1.3% is reasonable, especially because, compared to the true solution, the HF solver also results in an error of $\approx 2\%$. Note that other MOR techniques that also rely on regression (driven by neural networks, for instance) report a similar error stagnation [22, 25].

For $n_{\tilde{\Psi}} = n$, Figure 3b compares the average error between GPR-TS-MOR and S-PROJ. Recall that for S-PROJ, the value of $n_{\tilde{\Psi}}$ is irrelevant. For $n \lesssim 17$, GPR-TS-MOR outperforms S-PROJ.

# \mathcal{D}_z	$E_a(n, n_{\tilde{\Psi}})$				
	$n_{\tilde{\Psi}}, n = 1$	$n_{\tilde{\Psi}}, n = 3$	$n_{\tilde{\Psi}}, n = 5$	$n_{\tilde{\Psi}}, n = 7$	$n_{\tilde{\Psi}}, n = 9$
40×20	2.2×10^{-1}	4.3×10^{-2}	2.0×10^{-2}	1.4×10^{-2}	1.3×10^{-2}
80×20	2.2×10^{-1}	4.3×10^{-2}	2.0×10^{-2}	1.3×10^{-2}	1.1×10^{-2}

Table 1: Results for test case-1. Average-error for different values of n and $n_{\tilde{\Psi}}$. The set \mathcal{D}_z contains all the training parameters.

Already for $n = 1$, GPR-TS-MOR results in a relative error of $\approx 22\%$. In comparison, S-PROJ results in an error of $\approx 70\%$. The difference is the most pronounced for $n = 5$, for which GPR-TS-MOR results in an error of 2% , which is ≈ 5.6 times smaller than the error resulting from S-PROJ. Due to the stagnation in the POD projection error reported earlier, S-PROJ outperforms GPR-TS-MOR for $n \gtrsim 17$.

Note that as n is increased, although the solution from S-PROJ appears to converge in the L^1 -sense, it results in a highly oscillatory approximation—Figure 3c compares the different solutions for $z = (0.8, 1)$ and $n, n_{\tilde{\Psi}} = 10$. The discontinuities in the (untransformed) solution trigger oscillations in the POD basis, which results in an oscillatory S-PROJ solution. In contrast, owing to the solution transformation, GPR-TS-MOR exhibits no such oscillations.

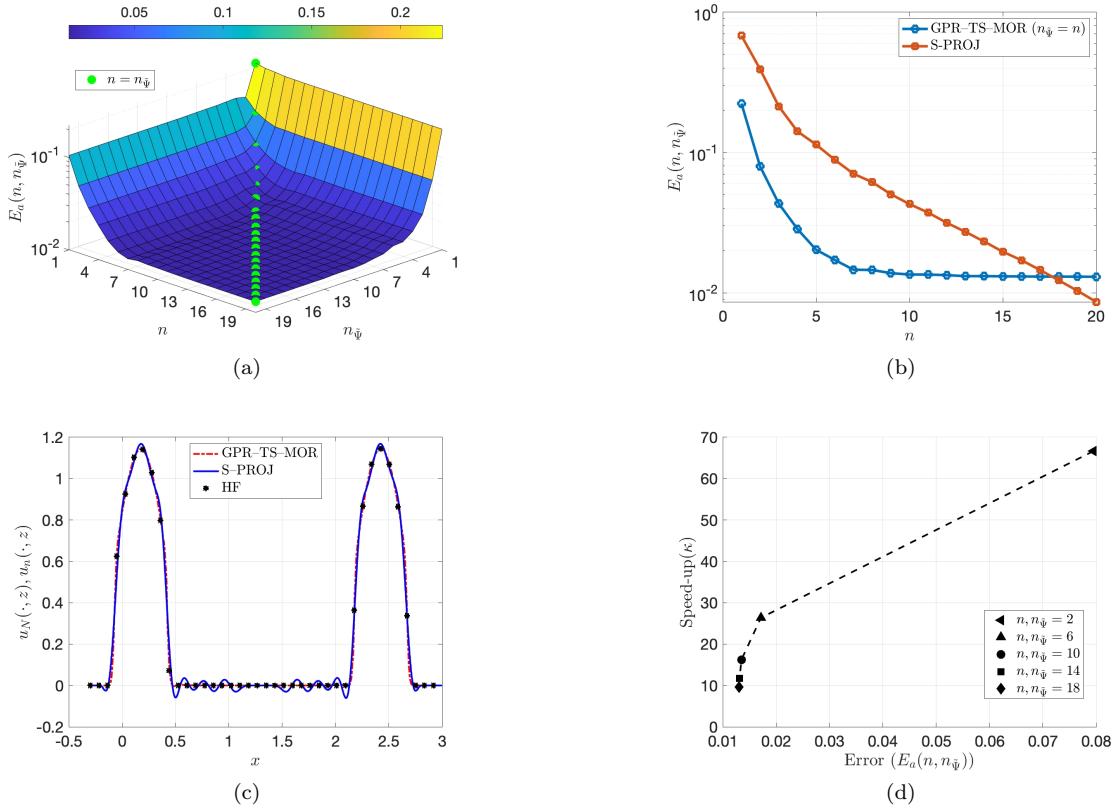


Figure 3: Results for test-1. (a) Convergence of the average-error with n and $n_{\tilde{\Psi}}$. (b) Error comparison between S-PROJ and GPR-TS-MOR. (c) Solution comparison for $z = (0.8, 1)$. (d) Error vs. speed-up for different n and $n_{\tilde{\Psi}}$.

6.4.4 Speed-up vs. accuracy

We denote the speed-up by κ and define it as

$$\kappa := \frac{\sum_{z \in \mathcal{D}_z^{tst}} \tau_{HF}(z)}{\sum_{z \in \mathcal{D}_z^{tst}} \tau_{MOR}(z)}, \quad (59)$$

where $\tau_{MOR}(z)$ and $\tau_{HF}(z)$ denote the CPU-time required by the GPR-TS-MOR and the high-fidelity solver, respectively, to compute a solution at $z \in \mathcal{Z}$. We measure the CPU-time with the MATLAB's built-in function `tic-toc`.

Figure 3d plots the speed-up against the average error $E_a(n, n_{\tilde{\Psi}})$. As expected, increasing both n and $n_{\tilde{\Psi}}$ reduces both the error and the speed-up. The minimum speed-up of ≈ 10 corresponds to $n, n_{\tilde{\Psi}} = 20$ and is associated with an error of $\approx 1.3\%$. The maximum speed-up of ≈ 70 corresponds to $n, n_{\tilde{\Psi}} = 1$ and is associated with an error of $\approx 22\%$. Note that for time-dependent problems, our reduced approximation does not require any further iterations or time-stepping. To recover an approximation for any $z \in \mathcal{Z}$, we directly compute the approximation in (40). This explains why, on average, we observe a significant speed-up compared to a time-stepping based finite-volume solver.

6.5 Test-2

We discretize Ω with $N = 300 \times 300$ grid cells, which results in a grid size of $\Delta x = 5.3 \times 10^{-3}$. The training data \mathcal{D}_z is a set of 100 uniformly placed points inside $[0, T]$. We approximate the displacement field $\Psi(\cdot, z_{\text{ref}}, z)$ in the polynomial space $\mathcal{P}_{M=6}$, where the value of M results from the procedure outlined in Section 3.2.1.

6.5.1 Study of the POD projection error

For the two snapshot matrices \mathcal{S}_G and \mathcal{S}_U defined in (28) and (57), respectively, Figure 4 compares the relative POD projection error defined in (30). Similar to the last test case, $E_n^{\text{proj}}(\mathcal{S}_G)$ decays drastically for $n \lesssim 10$, followed by a steady decay for $n \gtrsim 10$. For all $n \leq 30$, it remains at least 2.2 times smaller than $E_n^{\text{proj}}(\mathcal{S}_U)$. The difference is the most pronounced for $n = 13$, for which we find

$$E_{n=13}^{\text{proj}}(\mathcal{S}_G) \approx 2 \times 10^{-2}, \quad E_{n=13}^{\text{proj}}(\mathcal{S}_U) \approx 9 \times 10^{-2}. \quad (60)$$

Let us recall that $n = 13$ is a tiny fraction of N -precisely, $1.5 \times 10^{-2}\%$ of N . For such a small fraction of N , a relative error of 2% seems reasonable.

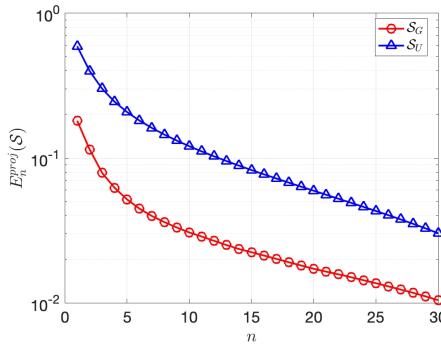


Figure 4: Results for test-2. Comparison of the POD projection error $E_n^{\text{proj}}(\mathcal{S})$ for the transformed \mathcal{S}_G and the untransformed \mathcal{S}_U snapshot matrices.

6.5.2 Study of the transformed solution

Let us further elaborate on the reason why the singular values of \mathcal{S}_G decay faster than those of \mathcal{S}_U . Consider three different parameter samples: $z_1 = 0.2$, $z_2 = 1$, and $z_3 = 1.8$. For these parameters, Figure 5 compares the solution $u_N(\cdot, z)$ to the transformed solution $g(\cdot, z)$. We observe that $u_N(\cdot, z_2) = g(\cdot, z_2)$. This is because z_2 is our reference parameter for which $\varphi(\cdot, z_{\text{ref}}, z_2) = \text{Id}$. For the untransformed solution $u_N(\cdot, z)$, the surface of discontinuity in the spatial domain moves (almost along the diagonal of the spatial domain) as z changes. This movement is the reason why the singular values of \mathcal{S}_U (reported in Figure 4) decay slowly. In comparison, the surface of discontinuity in the transformed solution shows very little movement with z , which then induces a fast singular value decay in the snapshot matrix \mathcal{S}_G .

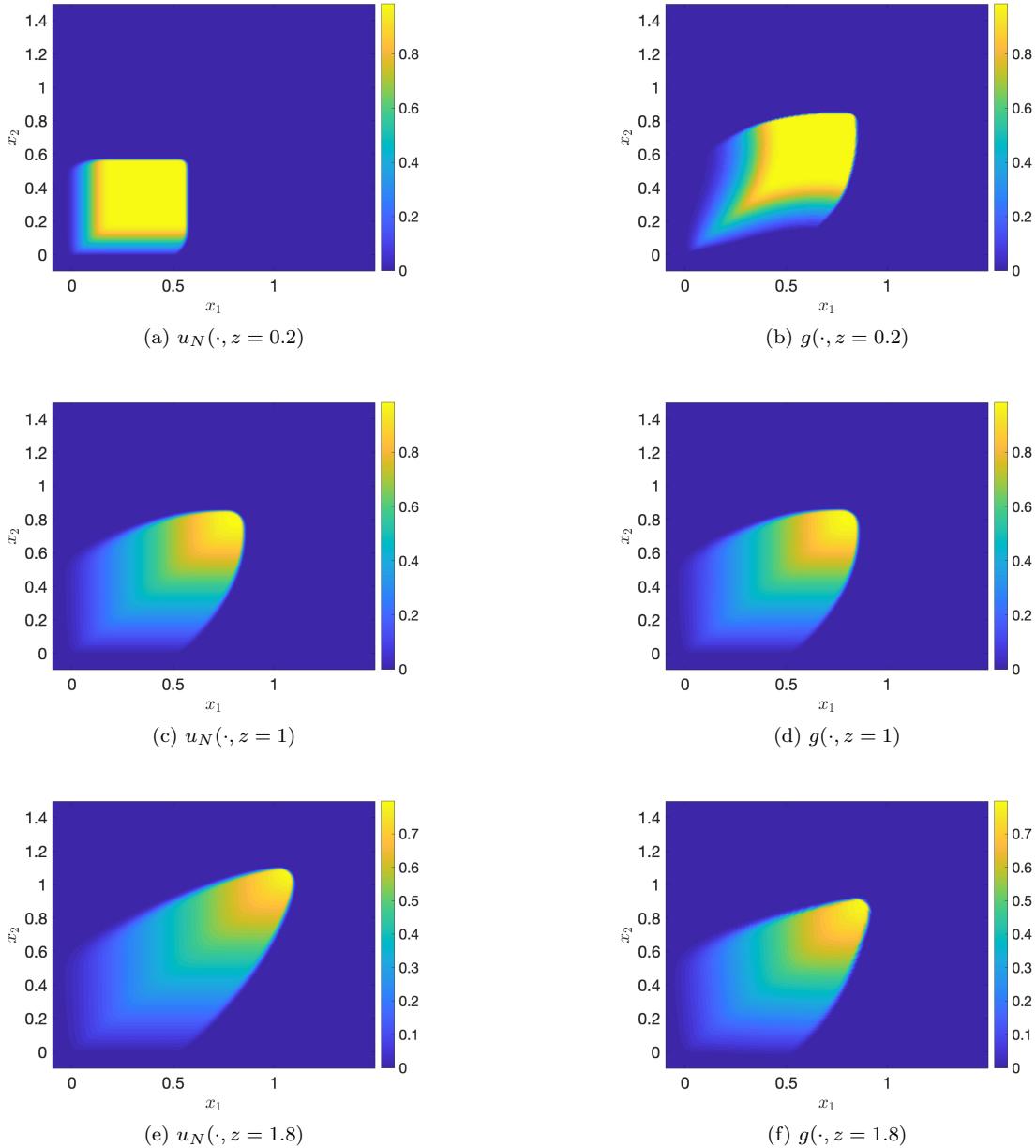


Figure 5: Results for test-2. Comparison between the transformed $u_N(\cdot, z)$ and the untransformed solution $g(\cdot, z)$. Left panel: untransformed solution. Right panel: transformed solution.

6.5.3 Study of φ and φ^{-1}

Consider the Jacobians

$$\mathcal{J}(x, z) := \det(\nabla\varphi(x, z_{\text{ref}}, z)), \quad \tilde{\mathcal{J}}(x, z) := \det(\nabla\tilde{\varphi}_{n_{\tilde{\Psi}}}(x, z_{\text{ref}}, z)). \quad (61)$$

Recall that φ results from the optimization problem in (21), and $\tilde{\varphi}_{n_{\tilde{\Psi}}}$ is a POD based approximation to φ^{-1} and is given in (39). For the present study, we fix $n_{\tilde{\Psi}} = 20$. Qualitatively, results remain the same for other values of $n_{\tilde{\Psi}}$.

For different parameter instances, Figure 6a presents the minimum values of the above two Jacobians. Both the Jacobians stay well above zero, which is desirable. Since $\varphi(\cdot, z_{\text{ref}}, z) \in C^1(\Omega)$, a positive $\mathcal{J}(x, z)$ together with Theorem-2.1 of [39] implies that $\varphi(\cdot, z_{\text{ref}}, z)$ is a diffeomorphism from Ω to Ω . Equivalently, in the sense of Remark 1, the present example exhibits a small displacement. Similarly, a positive $\tilde{\mathcal{J}}(x, z)$ implies that $\tilde{\varphi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z)$ is a homeomorphism from Ω to Ω .

Note that $z_{\text{ref}} = 1$. Therefore, as $z \rightarrow 1$, $\varphi(\cdot, z_{\text{ref}}, z) \rightarrow \text{Id}$, which, for all $x \in \Omega$, results in $\mathcal{J}(x, z) \rightarrow 1$. However, as $z \rightarrow 1$, $\tilde{\mathcal{J}}(x, z) \not\rightarrow 1$. This is because $\tilde{\varphi}_{n_{\tilde{\Psi}}}(\cdot, z_{\text{ref}}, z)$ is only an approximation to $\varphi(\cdot, z_{\text{ref}}, z)^{-1}$. Also note that $\inf_x \mathcal{J}(x, z)$ is better behaved than $\inf_x \tilde{\mathcal{J}}(x, z)$ because the former results from directly solving the optimization problem in (21), whereas the latter results from a POD and GPR-based approximation.

Observe that as $|z - z_{\text{ref}}|$ increases, the minimum value $\inf_x \mathcal{J}(x, z)$ (and also, in general, the value $\inf_x \tilde{\mathcal{J}}(x, z)$) decreases monotonically. This is expected because increasing $|z - z_{\text{ref}}|$ increases the distance between the surface of spatial discontinuities in the solution $u(\cdot, z)$ and the reference $u(\cdot, z_{\text{ref}})$. Therefore, to move these surfaces closer and align them, the spatial transform needs to displace points by larger distances. This significantly *compresses* some regions of the spatial domain, resulting in smaller Jacobians. We speculate that by increasing $\sup_{z \in \mathcal{Z}} |z - z_{\text{ref}}|$ —i.e., by increasing the size of the parameter domain—one can make $\inf_x \mathcal{J}(x, z)$ negative thus, violating the small displacement assumption referred to in Remark 1. For such problems, one might have to resort to the large deformation registration considered in [5].

In Section 4, we assumed that the displacement $\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z)$ is well approximable in a sufficiently low-dimensional POD space. Here, we justify this assumption empirically. Consider the first component of $\tilde{\Psi}_N(\cdot, z_{\text{ref}}, z)$ given by $\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z)$. Let $\mathcal{S}_{\tilde{\Psi}_N^{(1)}}$ be a snapshot matrix for $\tilde{\Psi}_N^{(1)}(\cdot, z_{\text{ref}}, z)$. For this snapshot matrix, Figure 6b presents the POD projection error defined in (30). The projection error decays fast. Already for $n_{\tilde{\Psi}} = 5$, we get a relative error of less than 1%. The fast decay becomes obvious when we compare with the projection error of the snapshot matrix \mathcal{S}_U defined in (57). This matrix contains untransformed snapshots and thus, has a slow decay in the projection error.

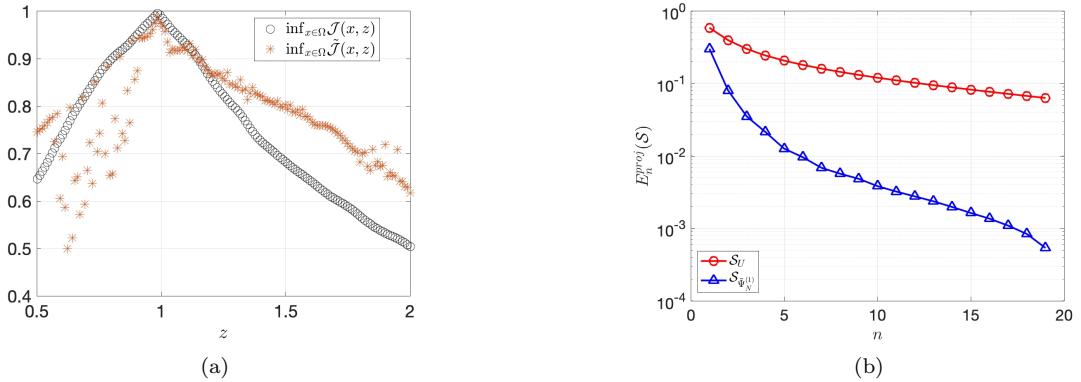


Figure 6: Results for test-2. (a) Minimum value of the Jacobian defined in (61). (b) POD projection error for the snapshot matrix $\mathcal{S}_{\tilde{\Psi}_N^{(1)}}$ and \mathcal{S}_U .

6.5.4 Study of the average error

For GPR-TS-MOR, Figure 7a presents the average error $E_a(n, n_{\tilde{\Psi}})$. We vary both n and $n_{\tilde{\Psi}}$ in the set $\{1, 2, \dots, 20\}$. Results are similar to that of the previous test case. Keeping n fixed and increasing $n_{\tilde{\Psi}}$ beyond a certain point (or vice-versa) offers no benefit. Nevertheless, increasing both n and $n_{\tilde{\Psi}}$ simultaneously decreases the error monotonically. However, due to the reasons explained earlier, eventually, the error starts to stagnates. The lowest relative error we attain is of 1.6%.

Figure 7b compares the average error between GPR-TS-MOR and S-PROJ. For all values of $n \leq 20$, GPR-TS-MOR outperforms S-PROJ. It results in an error that is at least two times smaller than the error resulting from S-PROJ. The difference is the most pronounced for $n = 6$; GPR-TS-MOR results in an error of $\approx 2.3\%$, which is ≈ 6 times smaller than the error resulting from S-PROJ. Let us recall that $n = 6$ is just $5.6 \times 10^{-3}\%$ of N . For such small value of n , an error of 2.3% can be considered reasonable.

The error from GPR-TS-MOR stagnates at a value of 1.6%. The minimum value of n that provides this error is $n = 11$. Although not shown in the plot, to achieve the same error, S-PROJ requires 50 POD modes. This is four times the number of modes required by GPR-TS-MOR. The takeaway being that despite the stagnation, GPR-TS-MOR significantly outperforms S-PROJ.

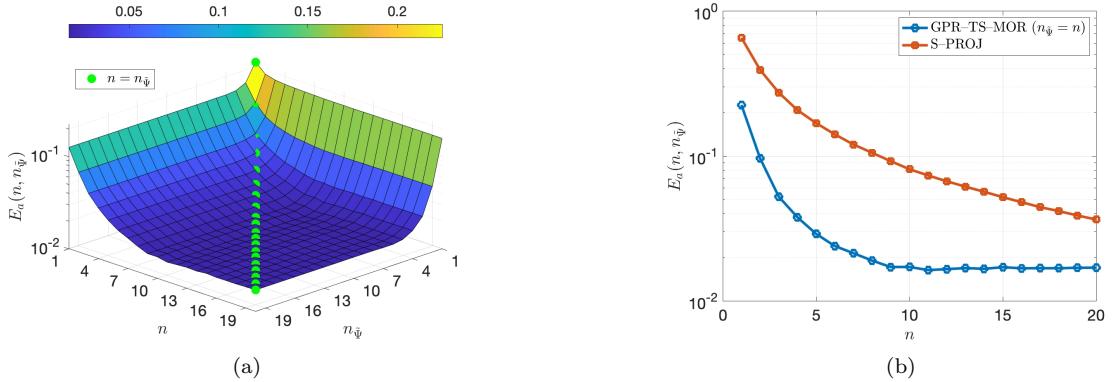


Figure 7: Results for test-2. (a) Convergence of the average error for GPR-TS-MOR with n and $n_{\tilde{\Psi}}$. (b) Error comparison between S-PROJ and GPR-TS-MOR.

6.5.5 Solution comparison

In Figure 8, we visually compare the solution resulting from GPR-TS-MOR and S-PROJ. We set $z = 1.2$, and $n, n_{\tilde{\Psi}} = 10$. As anticipated, due to the moving discontinuities in the solution, S-PROJ results in an oscillatory solution. These oscillations are spread-out over the entire spatial domain and appear to originate close to the discontinuity. In contrast, GPR-TS-MOR exhibits no such oscillations. The reason being that it approximates the transformed solution in the POD basis. As studied earlier, discontinuities in the transformed solution do not move much, resulting in non-oscillatory POD modes. Observe that GPR-TS-MOR exhibits some minor over and under shoots near the front-end of the surface of discontinuity. The reason being the minor misalignment of discontinuities reported earlier.

6.5.6 Study of the error surrogate

For $n, n_{\tilde{\Psi}} = 20$, Figure 9a compares the true error $E(z, n, n_{\tilde{\Psi}})$ to its surrogate $E^R(z, n, n_{\tilde{\Psi}})$ defined in (42). For most parts of the parameter domain, the surrogate well-approximates the error. This results in an efficiency index (see Figure 9b) that stays close to one. The efficiency index fluctuates between 0.8 and 1.7. The average value of the efficiency index is 1.2 i.e., on average, we overestimate the error by 20%.

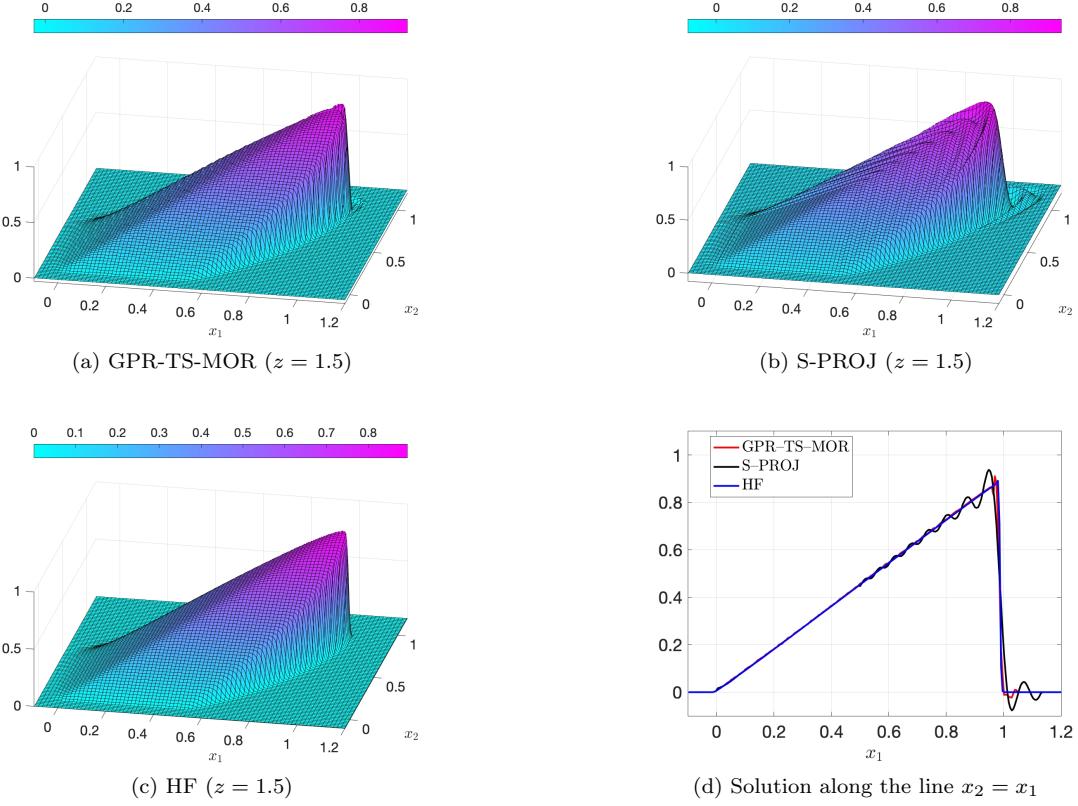


Figure 8: Results for test-2. Comparison between the different solutions at $z = 1.5$.

Observe that the error is particularly large close to $z = 0$. This is because the solution at $z = 0$ has a different discontinuity-topology than all the other solution instances—see Section 3.2 for the relevance of discontinuity-topology. The initial data has two surfaces of discontinuities: (i) the lower and the left edge of the square over which the characteristic function in (51) is defined, and (ii) the top and the right edge of the same square. For $z \neq 0$, the first surface manifests into a rarefaction fan, which is continuous. The second surface, however, results in a moving shock. This sudden breakdown of the discontinuity-topology at $z = 0$ results in a slightly inaccurate solution transformation close to $z = 0$, which, then, results in comparatively larger error values.

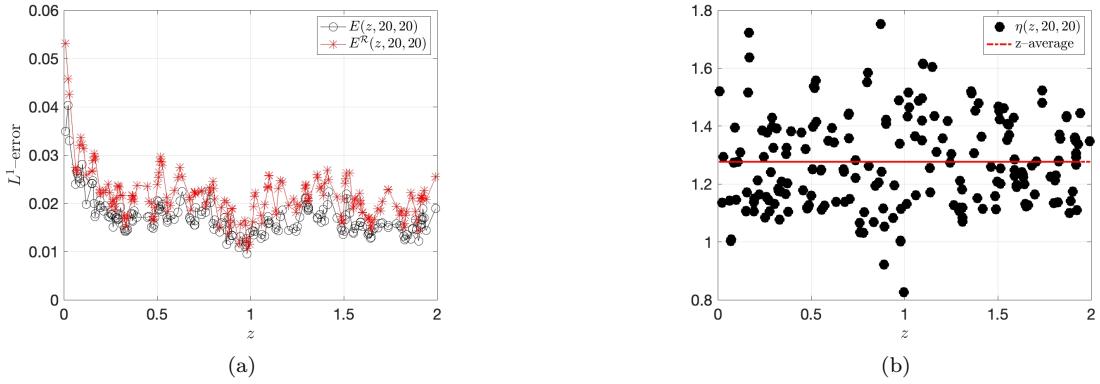


Figure 9: Results for test-2. (a) Compares the true average-error to its surrogate. (b) Presents the efficiency index defined in (43).

6.5.7 Speed-up vs. accuracy

For the GPR-TS-MOR, Figure 10 plots the speed-up defined in (59) against the average error. As anticipated, both the speed-up and the error decrease upon simultaneously increasing n and n_{Ψ} . The minimum speed-up of 800 and an error of 1.6% corresponds to $n, n_{\Psi} = 20$. The maximum speed-up of 2000 and an error of 12% corresponds to $n, n_{\Psi} = 1$. Note that the speed-up is at least two orders-of-magnitude larger than in the previous test case. Since the current problem is two-dimensional, the HF space has a large dimension of 300^2 , which makes a HF solver much more expensive than in the previous test case.

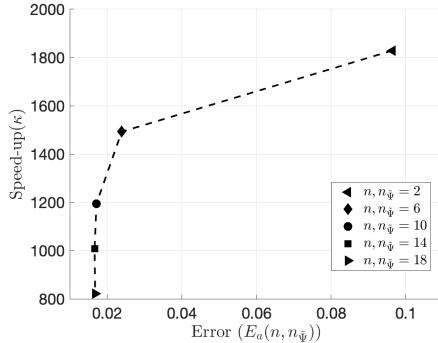


Figure 10: Results for test-2. Speed-up vs. average error plot.

6.6 Test-3

We discretize Ω with 300×300 grid cells. The training data \mathcal{D}_z is a set of uniformly placed 40 points inside \mathcal{Z} . We approximate the displacement field $\Psi(\cdot, z_{\text{ref}}, z)$ in the polynomial space $\mathcal{P}_{M=6}$, where the value of M results from the procedure outlined in Section 3.2.1.

6.6.1 Solution comparison, average error and speed-up

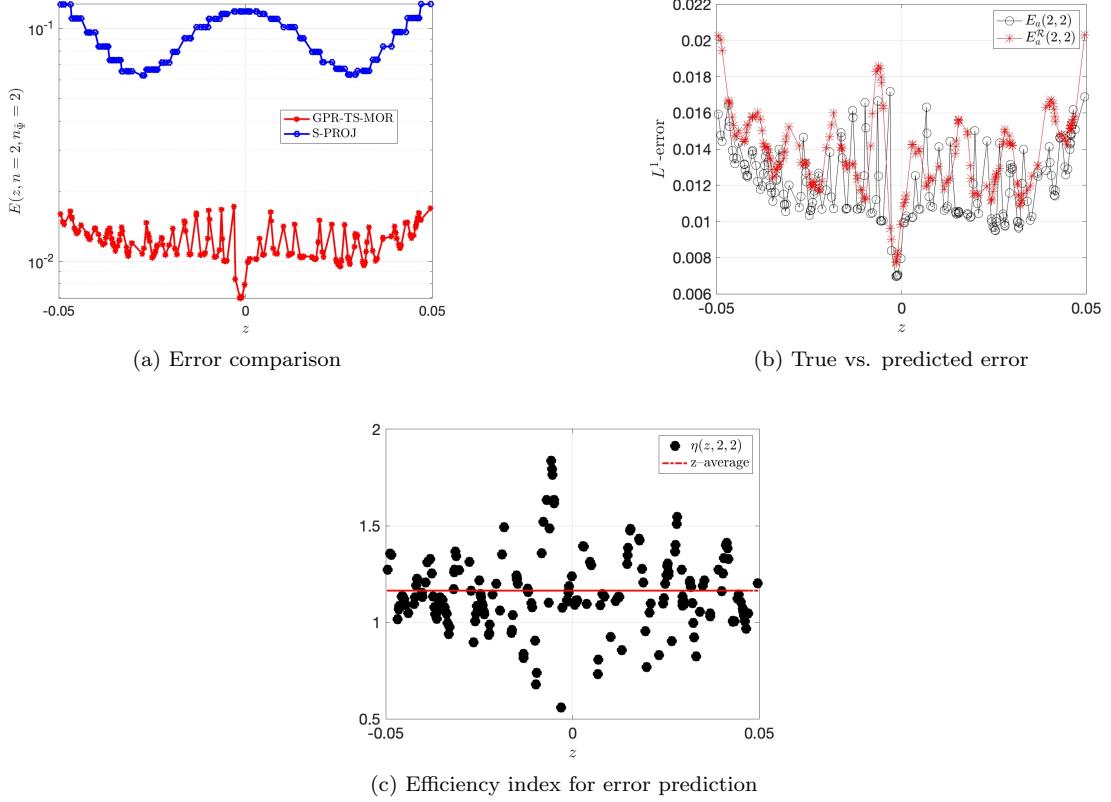
Results for the convergence study remain similar to the previous test case and we do not repeat them here for brevity. We set $n, n_{\Psi} = 2$, study the resulting average error and perform solution comparison. As for the average error, we find

$$\begin{aligned} \text{GPR-TS-MOR: } E_a(n = 2, n_{\Psi} = 2) &= 1.2 \times 10^{-2}, \\ \text{S-PROJ: } E_a(n = 2, n_{\Psi} = 2) &= 9 \times 10^{-2}. \end{aligned} \quad (62)$$

Clearly, GPR-TS-MOR outperforms S-PROJ. It results in an average error that is almost 5.5 times smaller than that resulting from S-PROJ. Figure 11a depicts the L^1 -error for several different parameter instances. For each of the tested parameter instances, the error from GPR-TS-MOR is five to ten times smaller than that from S-PROJ. Figure 11b compares the error surrogate to the true error and Figure 11c presents the corresponding efficiency index. As before, on average, our surrogate accurately approximates the true error, with an efficiency index that oscillates between 0.6 and 1.8.

For $z = 2.5 \times 10^{-2}$, Figure 12 compares the different solutions. With just two modes, GPR-TS-MOR provides an accurate approximation of the solution. Note that the solution from S-PROJ does not exhibit spurious oscillations reported in the previous test case because the value of n is smaller—oscillations are only present in the higher order POD modes. However, close to the boundaries of the inner-box Ω_z , it does exhibit a staircase type effect. This staircase effect is typical for linear reduced approximations of such problems—see [29, 44], for further examples.

Consider the speed-up κ defined in (59). For the current test case, we observed a speed of 51.5 with an average error of 1.2%. Note that the speed-up is lower than in the previous test case because the HF solver of the current problem requires no time iterations.

Figure 11: *Results for test-3.*

7 Conclusions and discussion

We have proposed a data-driven MOR technique to approximate parameterized partial differential equations that exhibit parameter-dependent jump-discontinuities. Our technique hinges on a two step procedure: transformation followed by de-transformation. The transformation step (ideally) removes the parametric discontinuities by composing the solution with a spatial transform. This results in a transformed solution that is well-approximable in a low-dimensional reduced space. After we approximate this transformed solution, we de-transform the approximation by composing it with an inverse of the spatial transform and recover an approximation to the solution of the differential equation. An offline-online paradigm based procedure guarantees an efficient transformation and de-transformation step.

Two data-driven methodologies are the building-blocks of our MOR technique: (i) Gaussian process regression, and (ii) optimization-based image registration. With GPR, we approximate the map between the parameter domain and the expansion coefficients of the reduced basis. With image registration on the other hand, we compute the spatial transform that allows for the solution transformation. Owing to the GPR, our MOR technique is purely data-driven i.e., it does not even require the knowledge of the structural non-linearities in the differential equation. This way it treats linear, non-linear, affine-in-parameter and non-affine-in-parameter problems alike and doesn't rely on any hyper-reduction technique.

We performed numerical experiments involving hyperbolic and parabolic differential equations. We compared our technique to a standard MOR technique that does not perform any solution transformation. Main takeaways from our experiments are as follows. Firstly, our technique results in an almost oscillation free solution. We attribute this to the solution transformation that halts the movement of the discontinuities in the parameter domain, resulting in a well-behaved set of POD modes. In contrast, due to moving discontinuities, the standard technique results in a highly oscillatory solution. Secondly, for a given number of POD modes, we outperform the

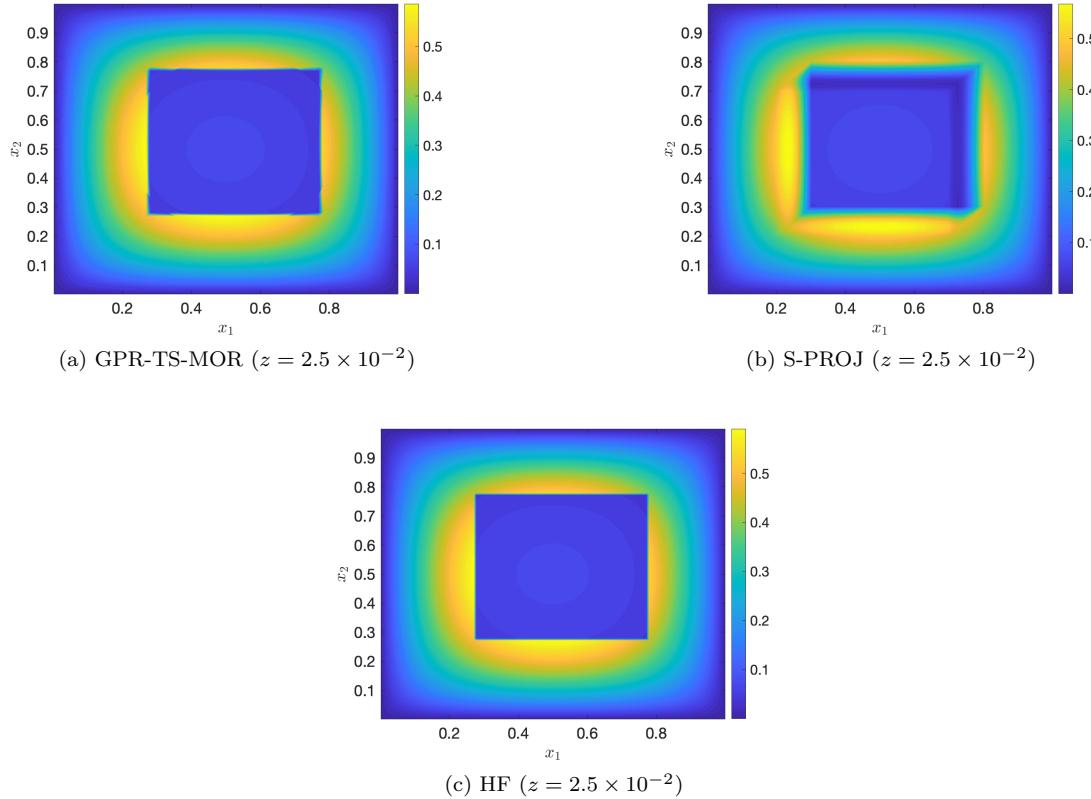


Figure 12: Results for test-3. Comparison between the different solutions at $z = 2.5 \times 10^{-2}$.

standard technique in terms of accuracy, with the error being two to ten times smaller than that resulting from the standard technique. Lastly, for moderate parameter dimensions, at least for the test cases we considered, we observed speed-ups of one to upto three orders-of-magnitude.

Acknowledgements

N.S and P.B are supported by the German Federal Ministry for Economic Affairs and Energy (BMWi) in the joint project "MathEnergy - Mathematical Key Technologies for Evolving Energy Grids", sub-project: Model Order Reduction (Grant number: 0324019B).

References

- [1] A. Alla and J. N. Kutz. Nonlinear model order reduction via dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 39(5):B778–B796, 2017.
- [2] M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.
- [3] D. Amsallem, M. Zahr, Y. Choi, and C. Farhat. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4):919–940, 2015.
- [4] C. Audouze, F. De Vuyst, and P. B. Nair. Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. *Numerical Methods for Partial Differential Equations*, 29(5):1587–1628, 2013.

- [5] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61(2):139–157, 2005.
- [6] P. Benner, P. Goyal, B. Kramer, B. Peherstorfer, and K. Willcox. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Computer Methods in Applied Mechanics and Engineering*, 372:113433, 2020.
- [7] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.
- [8] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox. *Model Reduction and Approximation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.
- [9] P. Benner, E. Sachs, and S. Volkwein. Model order reduction for PDE constrained optimization. In G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, and S. Ulbrich, editors, *Trends in PDE Constrained Optimization*, pages 303–326. Springer, 2014.
- [10] P. Benner and J. Schneider. Uncertainty quantification for Maxwell’s equations using stochastic collocation and model order reduction. *International Journal for Uncertainty Quantification*, 5(3):195–208, 2015.
- [11] N. Cagniart, Y. Maday, and B. Stamm. Model order reduction for problems with large convection effects. In B. N. Chetverushkin, W. Fitzgibbon, Y. Kuznetsov, P. Neittaanmäki, J. Periaux, and O. Pironneau, editors, *Contributions to Partial Differential Equations and Applications*, pages 131–150. Springer International Publishing, Cham, 2019.
- [12] K. Carlberg. Adaptive h-refinement for reduced-order models. *International Journal for Numerical Methods in Engineering*, 102(5):1192–1210, 2015.
- [13] W. Chen, J. S. Hesthaven, B. Junqiang, Y. Qiu, Z. Yang, and Y. Tihamo. Greedy nonintrusive reduced order model for fluid dynamics. *AIAA Journal*, 56(12):4927–4943, 2018.
- [14] R. Crisovan, D. Torlo, R. Abgrall, and S. Tokareva. Model order reduction for parametrized nonlinear hyperbolic problems as an application to uncertainty quantification. *Journal of Computational and Applied Mathematics*, 348:466–489, 2019.
- [15] M. Drohmann and K. Carlberg. The ROMES method for statistical modeling of reduced-order-model error. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):116–145, 2015.
- [16] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.
- [17] V. Ehrlacher, D. Lombardi, O. Mula, and F.-X. Vialard. Nonlinear model reduction on metric spaces. Application to one-dimensional conservative PDEs in Wasserstein spaces. *ESAIM: Mathematical Modelling and Numerical Analysis*, 2019.
- [18] C. Greif and K. Urban. Decay of the Kolmogorov N-width for wave problems. *Applied Mathematics Letters*, 96:216 – 222, 2019.
- [19] M. Guo and J. S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99, 2019.
- [20] J. S. Han, E. B. Rudnyi, and J. G. Korvink. Efficient optimization of transient dynamic problems in MEMS devices using model order reduction. *Journal of Micromechanics and Microengineering*, 15(4):822–832, 2005.
- [21] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, Cham, 2016.

- [22] J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.
- [23] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv:1807.02582*, 2018.
- [24] S. Klein, M. Staring, and J. P. W. Pluim. Evaluation of optimization methods for nonrigid medical image registration using mutual information and B-Splines. *IEEE Transactions on Image Processing*, 16(12):2879–2890, 2007.
- [25] K. Lee and K. T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.
- [26] Z. Ma and W. Pan. Data-driven nonintrusive reduced order modeling for dynamical systems with moving boundaries using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 373:113495, 2021.
- [27] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.
- [28] R. Mojgani and M. Balajewicz. Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows. *arXiv:1701.04343*, 2017.
- [29] N. J. Nair and M. Balajewicz. Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks. *International Journal for Numerical Methods in Engineering*, 117(12):1234–1262, 2019.
- [30] V. Noblet, C. Heinrich, F. Heitz, and J.-P. Armsbach. Accurate inversion of 3-D transformation fields. *IEEE transactions on image processing*, 17(10):1963–1968, 2008.
- [31] B. Peherstorfer and K. Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.
- [32] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.
- [33] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2016.
- [34] C. E. Rasmussen and C. K. Williams. *Gaussian Processes in Machine Learning*. MIT press Cambridge, 2006.
- [35] D. Rim, B. Peherstorfer, and K. T. Mandli. Manifold approximations via transported subspaces: Model reduction for transport-dominated problems. *arXiv:1912.13024*, 2019.
- [36] N. Sarna, J. Giesselmann, and P. Benner. Data-driven snapshot calibration via monotonic feature matching. *arXiv:2009.08414*, 2020.
- [37] A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *IEEE Transactions on Medical Imaging*, 32(7):1153–1190, 2013.
- [38] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011, 1984.
- [39] T. Taddei. A registration method for model order reduction: Data compression and geometry reduction. *SIAM Journal on Scientific Computing*, 42(2):A997–A1027, 2020.
- [40] T. Taddei and L. Zhang. Space-time registration-based model reduction of parameterized one-dimensional hyperbolic PDEs. *arXiv:2004.06693*, 2020.

- [41] T. Taddei and L. Zhang. Registration-based model reduction in complex two-dimensional geometries. *arXiv:2101.10259*, 2021.
- [42] M. J. Vuik and J. K. Ryan. Multiwavelet troubled-cell indicator for discontinuity detection of discontinuous Galerkin schemes. *Journal of Computational Physics*, 270:138–160, 2014.
- [43] G. Welper. h and hp -adaptive interpolation by transformed snapshots for parametric and stochastic hyperbolic PDEs. *arXiv:1710.11481*, 2017.
- [44] G. Welper. Interpolation of functions with parameter dependent jumps by transformed snapshots. *SIAM Journal on Scientific Computing*, 39(4):A1225–A1250, 2017.
- [45] D. Xiao, F. Fang, A. Buchan, C. Pain, I. Navon, and A. Muggeridge. Non-intrusive reduced order modelling of the Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293:522–541, 2015.
- [46] D. Xiao, F. Fang, C. Pain, and G. Hu. Non-intrusive reduced-order modelling of the Navier–Stokes equations based on RBF interpolation. *International Journal for Numerical Methods in Fluids*, 79(11):580–595, 2015.
- [47] D. Xiao, P. Yang, F. Fang, J. Xiang, C. Pain, and I. Navon. Non-intrusive reduced order modelling of fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 303:35–54, 2016.
- [48] Y. Yue and K. Meerbergen. Accelerating optimization of parametric linear systems by model order reduction. *SIAM Journal on Optimization*, 23(2):1344–1370, 2013.