

Learning Reduced Order Models from Data for Hyperbolic PDEs

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](https://orcid.org/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](https://orcid.org/0000-0003-3362-4103)

Abstract: Given a set of solution snapshots of a hyperbolic PDE, we are interested in learning a reduced order model (ROM). To this end, we propose a novel decompose then learn approach. We decompose the solution by expressing it as a composition of a transformed solution and a de-transformer. Our idea is to learn a ROM for both these objects, which, unlike the solution, are well approximable in a linear reduced space. A ROM for the (untransformed) solution is then recovered via a recombination. The transformed solution results from composing the solution with a spatial transform that aligns the spatial discontinuities. Furthermore, the de-transformer is the inverse of the spatial transform and lets us recover a ROM for the solution. We consider an image registration technique to compute the spatial transform, and to learn a ROM, we resort to the dynamic mode decomposition (DMD) methodology. Several benchmark problems demonstrate the effectiveness our method in representing the data and as a predictive tool.

Keywords: Data-driven methods, Learning reduced order models, Hyperbolic PDEs, Decompose then learn

Novelty statement:

1. A decompose then learn approach for hyperbolic PDEs.
2. The solution is decomposed into a transformed solution and a de-transformer.
3. A ROM is learned for the transformed solution and the de-transformer.
4. A re-composition recovers a ROM for the (untransformed) solution.
5. Numerical experiments showcase improvements over a standard DMD approach.

1. Introduction

Scientific computing often involves applications where a numerical solution is sought at several different time or parameter instances. For such applications, a high-fidelity (HF) finite-element (or volume or difference) type solver can be prohibitively expensive and one seeks to replace it with an accurate and efficient surrogate. The reduced order modelling paradigm offers a reliable architecture to construct such surrogates. It performs the onerous task of snapshot collection offline, thus allowing for online efficiency [44]. We are interested in developing ROMs that can well approximate solutions to time-dependent, possibly non-linear, hyperbolic PDEs. Such PDEs are ubiquitous in science and engineering applications involving wave-type phenomenon, with the Euler and the wave equation being two classic examples.

Standard reduced order modelling techniques approximate the solution in a linear reduced space, which is usually a span of reduced or POD type basis vectors [7, 25]. Although a linear reduced approximation is accurate for a broad range of applications, it is largely inadequate for hyperbolic PDEs. The primary reason being moving spatial discontinuities or steep gradients. Even for smooth initial and boundary data, solutions to such PDEs exhibit spatial discontinuities that often move in space and thus trigger temporal discontinuities. Temporal (or parametric) discontinuities (or steep gradients) have mainly two negative consequences for the approximability in a linear reduced space. Firstly, a large reduced space might be required to well approximate the solution, hindering the efficiency of a ROM [9, 22, 65]. Secondly, temporal discontinuities trigger oscillations in reduced basis vectors, resulting in an oscillatory approximation [13, 21, 39, 65]. These inadequacies have motivated several authors to seek non-linear approximations. Broadly speaking, these approximations rely either on: (i) Wasserstein spaces [18, 47]; (ii) online adaptivity of basis [3, 41]; (iii) auto-encoders [21, 29, 32]; (iv) domain decomposition [15, 58], or (v) manifold calibration/transformation [12, 36, 39, 57, 59, 65].

Running parallel to these previous works, we propose a novel ROM technique for hyperbolic PDEs. Our interest lies in a non-intrusive data-driven approach where one learns a ROM from a set of solution snapshots, without having access to the underlying discrete evolution operators—in the context of problems that do not exhibit transport dominated effects, works along these lines can be found in [4, 6, 23, 26, 43, 46]. Such a configuration is appealing for applications where one relies either on experiments, or huge legacy/commercial codes, for snapshot collection. Note that intrusive techniques compute a ROM by projecting a PDE onto the reduced space, which requires access to the underlying discrete evolution operator—related to our context, a few of the many noteworthy works involving the intrusive technique can be found in [13, 24, 49, 50, 57, 59].

1.1. Decompose then learn approach

Since solutions to hyperbolic PDEs have poor approximability in a linear reduced space, following the standard approach of learning a ROM directly for the solution can be ineffective. Therefore, we propose to express the solution $u(\cdot, t)$ as a composition of two functions, which, unlike the solution, are sufficiently regular along the temporal domain and thus (with some additional assumptions [55, 64]) exhibit good approximability in a linear reduced space. Our idea is to learn a ROM for these two functions, followed by a re-composition that approximates the solution.

We decompose the solution as

$$u(\cdot, t) = g(\tilde{\varphi}(\cdot, t), t), \quad (1)$$

where

$$g(\cdot, t) := u(\varphi(\cdot, t), t), \quad \text{and} \quad \tilde{\varphi}(\cdot, t) := \varphi(\cdot, t)^{-1}. \quad (2)$$

We refer to $\varphi(\cdot, t) : \Omega \rightarrow \Omega$ as the spatial transform. When composed with the solution $u(\cdot, t)$, it results in a transformed solution $g(\cdot, t)$, which, at least ideally, has no temporal discontinuities [39, 57, 65]. We refer to $\tilde{\varphi}$ as the *de-transformer* since it reverts solution transformation. Akin to the transformed solution, the de-transformer is also sufficiently regular in time; as noted in [55], this claim requires some assumptions that we brief later. Owing to the temporal regularity of the transformed solution and the de-transformer, we can expect them to be well approximable in a linear reduced space and thus, an accurate ROM can be learned for these two objects. A ROM for the solution is then recovered via the above relation. Figure 1 provides a high-level summary of our idea.

We find the following attributes desirable in a technique used to compute φ .

- (P1) It should not require the discrete HF evolution operators.
- (P2) It should not require an explicit form of the underlying hyperbolic PDE.
- (P3) It should be independent of the technique used to learn the ROM.

Violation of (P1) defeats the purpose to developing a non-intrusive technique. In a data-driven setting, an explicit form of the underlying PDE might be unavailable, thus (P2) is desirable. Note

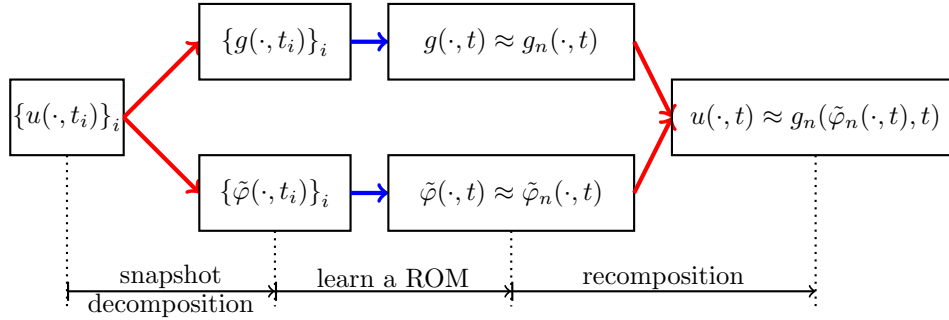


Figure 1.: A high-level depiction of our idea. Snapshots are collected at the time-instances $\{t_i\}_{i=1,\dots,K}$. The standard approach directly learns a ROM for $u(\cdot, t)$, without any prior decomposition. The functions g_n and $\tilde{\varphi}_n$ represent ROMs for g and $\tilde{\varphi}$, respectively.

that the framework in [33] does not satisfy (P2)—towards the end of this introduction, we elaborate further.

Property (P3) has two crucial consequences. Firstly, it ensures that any of the different learning techniques—for instance, the operator inference approach [42], radial basis functions approach [67], DMD approach [62]—can be used as a black-box to learn a ROM for g and $\tilde{\varphi}$. Secondly, it detaches the decomposition and the re-composition step from any pre-existing numerical implementation of a ROM learning toolbox. Consequently, one does not need to maintain an additional implementation for PDEs that are well approximable via standard techniques. By simply choosing $\varphi(\cdot, t) = \text{Id}$, where Id is the identity operator, and keeping the rest of the algorithm as such, one can recover a standard data-driven reduced order modelling technique. We resort to an optimization-based image registration technique to compute φ . Later sections make it obvious that this technique satisfies (P1)–(P3).

We undertake the (plain) DMD approach to learn a ROM [1, 52, 62]. The DMD approach approximates the snapshot trajectory via the action of a best-fit linear operator. The resulting linear evolution equation is then projected onto the POD modes. Owing to the linearity, the reduced approximation is available in an explicit form and requires no further time stepping. This results in a highly efficient ROM, which, even for non-linear problems, is often orders of magnitude faster than a HF solver [1, 45, 61]. It is noteworthy that for non-linear problems, intrusive techniques are usually as expensive as HF solvers and require an additional layer of hyper-reduction to restore efficiency [14].

1.2. Relation to previous works

Several previous works have successfully used DMD to learn fast and accurate ROMs—see [1, 8, 11, 27, 61], for instance. However, to the best of our knowledge, only the work in [33] tailors DMD for hyperbolic PDEs. Ideologically, this work is similar to ours—it also tries to learn a ROM for a transformed solution and a de-transformer. However, a major difference is that it chooses (the de-transformer) $\tilde{\varphi}$ as the mapping to the Lagrangian coordinates and computes it by solving an ODE that governs the evolution of the characteristic curves. Following are the possible shortcomings of such an approach. Firstly, the resulting ODE is well posed only when the characteristic curves do not intersect i.e., when the solution does not exhibit spatial discontinuities. This limits the applicability of the method to solutions without spatial discontinuities. In contrast, the image registration technique that we use to compute $\tilde{\varphi}$ is well-equipped to handle space-time discontinuous solutions [57, 59]. Secondly, computing Lagrangian coordinates requires the explicit form of the underlying PDE, which, in a data-driven setting, might not be available. Thirdly, for non-linear problems and multi-D spatial domains, tracing characteristics can be complicated. As a result, the examples in [33] are mostly limited to 1D spatial domains. On the other hand, the image registration technique that we use is extendible to even complex multi-D spatial domains [60].

The symmetry-reduction (SR) framework can be viewed as an inspiration behind our decompo-

sition in (1)—we refer to [10, 38, 40, 51] for details of the SR framework. Roughly speaking, this framework decomposes u as $\mathcal{H}(\hat{u})$, where \mathcal{H} is a continuous transformation that acts on the so-called template function \hat{u} . A finite-dimensional transformation \mathcal{H} results from a careful analysis of the underlying PDE, and is evolved using the so-called phase conditions. A PDE for the template function \hat{u} —derived using the equivariant property of the evolution operator stated below—is reduced using standard reduced order modelling techniques. The approximation for \mathcal{H} and the ROM for \hat{u} are then recombined to approximate the solution u .

One might be tempted to interpret the template function \hat{u} as the transformed solution g and the continuous transformation $\mathcal{H}(g)$ as $g \circ \tilde{\varphi}$, and conclude that our decomposition is the same as that of SR. However, there are some key differences that one needs to consider. Firstly, the SR technique assumes that the evolution operator is equivariant i.e., it commutes with \mathcal{H} . Only then can one derive an evolution equation for \hat{u} . In contrast, we make no such assumptions on the evolution operator. Our ROM for \hat{u} is learned directly from the data and the learning technique does not require the equivariance property, or for that matter, any knowledge of the underlying dynamical system. As a result, our technique applies to a much broader class of evolution equations. Secondly, we compute both φ and $\tilde{\varphi}$ in a purely data-driven manner without any prior explicit knowledge of the underlying hyperbolic PDE—recall that the SR framework carefully analyzes the underlying hyperbolic PDE to have a representation for \mathcal{H} . Lastly, as already pointed out in [57] and as the name suggests, SR decomposes the solution to reduce the symmetries in a dynamical system. We, on the other hand, decompose the solution solely from approximation considerations.

Recent works indicate that auto-encoders can provide accurate non-linear reduced spaces for hyperbolic PDEs [21, 29, 32]. However, as noted in [37], an auto-encoder applied directly to the solution can result in an oscillatory approximation. Furthermore, despite a large training set, even short-time predictions can be inaccurate. Authors in [37] circumvent both these problems by first transforming the snapshot matrix via a transformation of the space-time domain. An auto-encoder is then applied to the transformed snapshots and a long short-term memory (LSTM) architecture is used for time evolution. The takeaway is that even for modern artificial intelligence-based learning techniques, a decomposition of the form (1) seems like a crucial first step—particularly when one seeks accurate predictions. The bounds on the latent space of an auto-encoder developed in [20] further elucidate the importance of (at least) Lipschitz continuity while applying auto-encoders.

Note that by using POD for data compression (which is a type of an auto-encoder) and DMD for temporal evolution instead of LSTM, at an abstract level, one can relate our framework to that of [37]. However, the work in [37] is limited to 1D spatial domains. Furthermore, its efficiency, as compared to a HF solver, is unclear. Our approach on the other hand caters to multi-D spatial domains. Also, owing to the efficiency of the DMD approach, it is orders of magnitude faster than a HF solver. We will later empirically establish the efficiency of our method.

1.3. Organization

Rest of the article is organized as follows. Section 2 presents the (plain) DMD framework in a general setting. Section 3 specializes the DMD framework from Section 2 to hyperbolic PDEs. Section 4 summarizes our algorithm. Section 5 presents the numerical examples, and Section 6 closes the article with a conclusion.

2. Learning ROMs with DMD

We briefly summarize the (plain) DMD approach for learning a ROM from snapshot data. We consider a general setting where we collect snapshots of a time-dependent vector $\mathcal{F}(t)$ and learn a ROM for the same. Later in Section 3, we choose $\mathcal{F}(t)$ as the transformed solution and the de-transformer, thereby, specializing our framework to hyperbolic PDEs. We closely follow the formulation from [1, 62], and as such, introduce no new formulations here. We re-iterate that only for the sake of completeness and ease of implementation, we use the DMD approach. In practise, any other learning techniques mentioned earlier would also suffice.

We acknowledge that since its inception, DMD has underwent numerous updates like the multi-resolution version [31]; randomized version [11]; extended version [66]; sparsity promoting version

[28], etc. Whether choosing a different DMD version improves the results that we report later is an interesting question in its own right. However, given the limited scope of our article, we refrain from comparing the different DMD versions here.

2.1. Best-fit linear operator

Consider a set of discrete time instances

$$\mathcal{Z}_{train} := \{t_i\}_{i=1,\dots,K} \subset [0, T], \quad (3)$$

ordered such that $0 = t_1 < t_2 < \dots < t_K = T$. For simplicity, we consider a constant time step size i.e., $t_k = t_1 + \Delta t \times (k - 1)$, where $\Delta t > 0$ denotes the time step size—extension to non-uniform time steps can be found in [62]. We consider a time-dependent vector $\mathcal{F}(t) \in \mathbb{R}^N$ and collect its snapshots in a snapshot matrix $\mathcal{S}_{1,K}(\mathcal{F}) \in \mathbb{R}^{N \times K}$ given as

$$\mathcal{S}_{1,K}(\mathcal{F}) := (\mathcal{F}(t_1), \dots, \mathcal{F}(t_K)). \quad (4)$$

These snapshots can result either from a dynamical system of the form $\mathcal{F}(t_k) = \mathcal{L}(\mathcal{F}(t_{k-1}))$, or from experiments. Neither the explicit form of the evolution operator \mathcal{L} , nor the details of the experiments, are required by the subsequent procedure.

Consider a linear best-fit evolution operator $\tilde{\mathcal{A}} : \mathbb{R}^N \rightarrow \mathbb{R}^N$, the action of which well approximates the discrete time trajectory of our snapshots. Equivalently,

$$\mathcal{F}(t_{k+1}) \approx \tilde{\mathcal{A}}\mathcal{F}(t_k). \quad (5)$$

By representing the above expression in a matrix form, the linear operator can be computed by a least-squares best-fit given as

$$\tilde{\mathcal{A}} := \arg \min_{\mathcal{A}^* \in \mathbb{R}^{N \times N}} \|\mathcal{S}_{2,K}(\mathcal{F}) - \mathcal{A}^*\mathcal{S}_{1,K-1}(\mathcal{F})\|_F, \quad (6)$$

where $\|\cdot\|_F$ represents the Frobenius norm of a matrix. Notice that $\mathcal{S}_{2,K}(\mathcal{F})$ and $\mathcal{S}_{1,K-1}(\mathcal{F})$ contain snapshots for the time intervals $[t_2, t_K]$ and $[t_1, t_{K-1}]$, respectively. Therefore, $\mathcal{S}_{2,K}(\mathcal{F}) \approx \tilde{\mathcal{A}}\mathcal{S}_{1,K-1}(\mathcal{F})$ is a matrix representation of the relation in (5). A solution to the above problem is given in terms of the pseudo-inverse $\mathcal{S}_{1,K-1}(\mathcal{F})^\dagger$ and reads

$$\tilde{\mathcal{A}} = \mathcal{S}_{2,K}(\mathcal{F})\mathcal{S}_{1,K-1}(\mathcal{F})^\dagger. \quad (7)$$

2.2. Reduced order model (ROM)

We now develop a ROM for the vector $\mathcal{F}(t)$. We approximate this vector in a set of POD modes and derive a governing equation for the POD coefficients. Owing to the linearity of our best-fit evolution operator, eventually, we will have an explicit expression for these POD coefficients. The details are as follows.

Consider a low-rank singular value decomposition (SVD) based approximation of the snapshot matrix $\mathcal{S}_{1,K-1}(\mathcal{F})$ given as

$$\mathcal{S}_{1,K-1}(\mathcal{F}) \approx \mathcal{U}_n \Sigma_n \mathcal{V}_n^*, \quad (8)$$

where $(\cdot)^*$ represents the Hermitian transpose of a matrix. The matrix $\mathcal{U}_n \in \mathbb{R}^{N \times n}$ and $\mathcal{V}_n \in \mathbb{R}^{K \times n}$ contain the first $n \in \mathbb{N}$ left and right singular vectors, respectively, of the snapshot matrix. Furthermore, $\Sigma_n \in \mathbb{R}^{n \times n}$ is a diagonal matrix, which, at its diagonal, contains the first n singular values of the snapshot matrix. Later, we will be more interested in understanding how the reduced order modelling error decays with n , and to this end, we will vary n over a range of numbers. In practice, the value of n can be chosen such that the relative error of the above approximation—which follows from the Schmidt-Eckart-Young theorem [17]—stays below a given user-defined tolerance. The work in [53] outlines an implementation along these lines.

The columns of the matrix \mathcal{U}_n are the so-called POD modes. Approximating $\mathcal{F}(t)$ in the span of these modes provides

$$\mathcal{F}(t) \approx \mathcal{U}_n \alpha_{\mathcal{F}}(t), \quad (9)$$

where $\alpha_{\mathcal{F}}(t) \in \mathbb{R}^n$ is a vector containing all the POD coefficients. A discrete-in-time evolution equation for this vector results from replacing the above approximation into our best-fit linear evolution equation (5). This provides

$$\alpha_{\mathcal{F}}(t_{k+1}) = \tilde{\mathcal{A}}_n \alpha_{\mathcal{F}}(t_k) \quad \text{where} \quad \tilde{\mathcal{A}}_n := \mathcal{U}_n^* \tilde{\mathcal{A}} \mathcal{U}_n, \quad (10)$$

with the initial data $\alpha_{\mathcal{F}}(t_1) := \mathcal{U}_n^* \mathcal{F}(t_1)$. The matrix $\tilde{\mathcal{A}}_n$ is often referred to as the POD reduced matrix.

With the eigendecomposition of $\tilde{\mathcal{A}}_n$, we derive an explicit solution for the above equation. Let $\{(\tilde{\lambda}_i, \tilde{W}_i)\}_{i=1, \dots, n}$ represent a pair of eigenvalues and eigenvectors of $\tilde{\mathcal{A}}_n$. By repeatedly applying the operator $\tilde{\mathcal{A}}_n$ to the projected initial data $\tilde{b} := (\tilde{W})^\dagger \alpha_{\mathcal{F}}(t_1)$, we recover an explicit expression for $\alpha_{\mathcal{F}}(t)$ that reads

$$\alpha_{\mathcal{F}}(t) = \sum_{i=1}^n \tilde{W}_i \exp(\tilde{\omega}_i t) \tilde{b}_i = \tilde{W} \exp(\tilde{\omega} t) \tilde{b}, \quad (11)$$

where, for convenience, we define $\tilde{\omega}_i := \log(\tilde{\lambda}_i)/\Delta t$, and the diagonal matrix $\tilde{\omega} \in \mathbb{R}^{n \times n}$ collects the different $\tilde{\omega}_i$ at its diagonal. Furthermore, the matrix $\tilde{W} \in \mathbb{R}^{n \times n}$ contains the vectors \tilde{W}_i as its columns. With the above expression, we recover a ROM for $\mathcal{F}(t)$ that reads

$$\mathcal{F}(t) \approx \mathcal{U}_n \alpha_{\mathcal{F}}(t) = \mathcal{U}_n \tilde{W} \exp(\tilde{\omega} t) \tilde{b}. \quad (12)$$

Note that the above approximation is equation-free i.e., for any given $t \in [0, \infty)$, no further time stepping is required to approximate $\mathcal{F}(t)$.

2.3. Practical considerations: DMD algorithm

In practice, two considerations hinder the applicability of the above approach. Firstly, the matrix $\tilde{\mathcal{A}}$, computed via (5), can be severely ill-conditioned. Secondly, the value of N , which often denotes the dimensionality of a HF solver, can be extremely large. Under memory restrictions, it might be infeasible to store the $N \times N$ matrix $\tilde{\mathcal{A}}$ and perform any subsequent algebraic manipulations. Note that in the above formulation, $\tilde{\mathcal{A}}$ is required to compute the POD reduced matrix $\tilde{\mathcal{A}}_n$.

The DMD algorithm proposed in [62] circumvents both the above issues by replacing the pseudo-inverse of the snapshot matrix $\mathcal{S}_{1, K-1}(\mathcal{F})$, appearing in (5), by the pseudo-inverse of its low-rank SVD approximation given in (8). This replacement provides

$$\tilde{\mathcal{A}} \approx \mathcal{A} := \mathcal{S}_{2, K}(\mathcal{F}) \mathcal{V}_n \Sigma_n^{-1} \mathcal{U}_n^*. \quad (13)$$

Now, replacing $\tilde{\mathcal{A}}$ by \mathcal{A} in the definition for $\tilde{\mathcal{A}}_n$ given in (10), we find

$$\tilde{\mathcal{A}}_n \approx \mathcal{A}_n := \mathcal{U}_n^* \mathcal{A} \mathcal{U}_n = \mathcal{U}_n^* \mathcal{S}_{2, K}(\mathcal{F}) \mathcal{V}_n \Sigma_n^{-1}. \quad (14)$$

We emphasize that to compute \mathcal{A}_n , one does not need to explicitly compute the $N \times N$ matrix \mathcal{A} .

We approximate the eigendecomposition $(\tilde{W}, \tilde{\lambda})$ of $\tilde{\mathcal{A}}_n$ by the eigendecomposition (W, λ) of \mathcal{A}_n . Replacing this approximation in our ROM derived earlier in (12), we finally arrive at the DMD-based ROM given by

$$\mathcal{F}(t) \approx \mathcal{F}_n(t) := \mathcal{U}_n W \exp(\omega t) b, \quad (15)$$

where $b = (\mathcal{U}_n W)^\dagger \mathcal{F}(t_1)$, and $\omega := \log(\lambda)/\Delta t$. Note that the columns of the matrix $\mathcal{U}_n W$ are an approximation to the eigenvectors of the evolution operator $\tilde{\mathcal{A}}$ and are the so-called DMD modes. These modes can be viewed as an approximation to the so-called Koopman modes of the infinite-dimensional, but linear, Koopman operator [52, 62]. Furthermore, the eigendecomposition $(\mathcal{U}_n W, \lambda)$ is what one refers to as the DMD of the best-fit linear operator $\tilde{\mathcal{A}}$.

3. Learning ROMs for hyperbolic PDEs

This section specializes the DMD approach discussed earlier to hyperbolic PDEs. We start with some preliminaries.

3.1. Preliminaries

A general hyperbolic PDE is given as

$$\begin{aligned} \partial_t u + \nabla \cdot f(u) &= 0, \text{ on } \Omega \times [0, T], & u &= u_0, \text{ on } \Omega \times \{0\}, \\ u &= u_{bc}, \text{ on } \partial\Omega \times [0, T], \end{aligned} \quad (16)$$

where u_0 and u_{bc} represent the initial and the boundary data, respectively. We interpret the boundary conditions in the weak sense of [16]. For simplicity, the solution $u(\cdot, t)$ is assumed to be scalar-valued. An extension to vector-valued functions is straightforward; to corroborate this claim, later, we will consider a numerical example with a vector-valued solution. The time instance $T > 0$ is some finite time horizon, and $\Omega \subset \mathbb{R}^d$ is the spatial domain. The function $f : \mathbb{R} \rightarrow \mathbb{R}^d$ is the so-called flux-function, with ∇ being a gradient operator in \mathbb{R}^d .

We compute a HF approximation of the solution via a second-order finite volume scheme equipped with the local Lax Friedrich flux and the van Leer flux limiter [63]. For time-stepping, we consider an explicit second-order Runge–Kutta scheme with the CFL number set to 0.5. We denote the HF approximation by

$$u(\cdot, t) \approx u_N(\cdot, t) \in \mathcal{X}_N \subset L^2(\Omega). \quad (17)$$

where $\dim(\mathcal{X}_N) = N$. The space \mathcal{X}_N is a finite-volume space of piecewise constant functions defined over a shape-conforming and regular spatial discretization of Ω given as

$$\Omega = \bigcup_{i=1}^N \mathcal{I}_i, \quad (18)$$

where \mathcal{I}_i represents the i -th grid cell. With $\Pi : L^2(\Omega) \rightarrow \mathcal{X}_N$ we represent the orthogonal L^2 projection operator. For the sake of completeness, we have chosen a finite-volume HF solver. In practise, any other HF solver would also suffice.

For later convenience, with

$$\tilde{\mathcal{X}}_N \subset L^2(\Omega) \quad (19)$$

we represent a space of piecewise linear and continuous finite-element functions defined over the spatial mesh. The space $\tilde{\mathcal{X}}_N$ is \tilde{N} dimensional with \tilde{N} being the number of vertices in the spatial mesh. With $\tilde{\Pi} : C^0(\tilde{\Omega}) \rightarrow \tilde{\mathcal{X}}_N$ we represent a projection operator defined as

$$\tilde{\Pi}h(\hat{x}_i) := h(\hat{x}_i), \quad \forall i \in \{1, \dots, \tilde{N}\}, \quad (20)$$

where $\{\hat{x}_i\}_i$ denotes the set of vertices of the spatial mesh. Note that because $\tilde{\mathcal{X}}_N$ contains piecewise linear and continuous functions, the above relation uniquely defines $\tilde{\Pi}h$.

3.2. Decompose then learn approach

As stated earlier, solutions to hyperbolic PDEs exhibit temporal discontinuities and a ROM learned directly for the solution can be inefficient. Therefore, we propose to decompose the HF solution as

$$u_N(\cdot, t) = g_N(\tilde{\varphi}(\cdot, t), t) \quad \text{where} \quad g_N(\cdot, t) := u_N(\varphi(\cdot, t), t). \quad (21)$$

Recall that $\tilde{\varphi}(\cdot, t) = \varphi(\cdot, t)^{-1}$. Our idea was to first learn a ROM for $g_N(\cdot, t)$ and $\tilde{\varphi}(\cdot, t)$, followed by recovery of a ROM for $u_N(\cdot, t)$ via the above relation—refer back to Figure 1 for a summary. Throughout this subsection, we assume that, at the time instances \mathcal{Z}_{train} given in (3), we have access to both, the snapshots of the HF solution $\{u_N(\cdot, t)\}_{t \in \mathcal{Z}_{train}}$ and the snapshots of the spatial transform $\{\varphi(\cdot, t)\}_{t \in \mathcal{Z}_{train}}$. The former results from our HF numerical solver, whereas for the latter, we use the image-registration technique outlined later in Section 3.3.

3.2.1. ROM for g_N and $\tilde{\varphi}$

We specialize the DMD framework from Section 2 to learn a ROM for $g_N(\cdot, t)$ and $\tilde{\varphi}(\cdot, t)$. To proceed further, we need finite dimensional HF representations for $g_N(\cdot, t)$ and $\tilde{\varphi}(\cdot, t)$. We use the projection operator Π to project $g_N(\cdot, t)$ onto \mathcal{X}_N and collect the resulting degrees of freedom in a vector $G(t) \in \mathbb{R}^N$. Likewise, since we expect $\varphi(\cdot, t)$ to be a diffeomorphism (explained later) it is reasonable to project $\tilde{\varphi}(\cdot, t)$ onto the finite-element space $\tilde{\mathcal{X}}_n$ using the projection operator $\tilde{\Pi}$ given in (20). We compute the value $\tilde{\varphi}(\hat{x}_i, t)$ at the i -th vertex \hat{x}_i using the non-linear least-squares problem given as

$$\tilde{\varphi}(\hat{x}_i, t) := \arg \min_{x \in \Omega} \|\varphi(x, t) - \hat{x}_i\|_2^2, \quad (22)$$

which we solve using the `lsqnonlin` function of MATLAB. We collect the degrees of freedom of $\tilde{\Pi}\tilde{\varphi}(\cdot, t)$ in the vector $\tilde{\Phi}(t) \in \mathbb{R}^{\tilde{N} \times d}$.

Recall that the framework in Section 2 provides a ROM for a time-dependent vector $\mathcal{F}(t)$. In this framework, we now choose $\mathcal{F} \equiv G$ and $\mathcal{F} \equiv \tilde{\Phi}$ and recover the following ROMs

$$\begin{aligned} G(t) &\approx G_n(t) := \mathcal{U}_n^G W^G \exp(\omega^G t) b^G, \\ \tilde{\Phi}(t) &\approx \tilde{\Phi}_n(t) := \mathcal{U}_n^{\tilde{\Phi}} W^{\tilde{\Phi}} \exp(\omega^{\tilde{\Phi}} t) b^{\tilde{\Phi}}. \end{aligned} \quad (23)$$

For completeness, we briefly recall the different objects appearing in the expression for $G_n(t)$ —similar definition holds for the terms appearing in $\tilde{\Phi}_n(t)$. The matrices \mathcal{U}_n^G and \mathcal{V}_n^G contain the first n left and right singular vectors, respectively, of the snapshot matrix

$$\mathcal{S}_{1,K-1}(G) := (G(t_1), \dots, G(t_{K-1})). \quad (24)$$

Furthermore, the diagonal matrix Σ_n^G , at its diagonal, contain the first n singular values of the above matrix. The pair (W^G, λ^G) is the eigendecomposition of the POD reduced matrix

$$\mathcal{A}_n^G := (\mathcal{U}_n^G)^* \mathcal{S}_{2,K}(G) \mathcal{V}_n^G (\Sigma_n^G)^{-1}, \quad (25)$$

and $\omega^G := \log(\lambda^G)/\Delta t$. The matrix $\mathcal{U}_n^G W^G$ contains, as its columns, the DMD modes for $G(t)$. Lastly, the amplitude vector b^G results from projecting the initial data onto the DMD modes and reads $b^G := (\mathcal{U}_n^G W^G)^\dagger G(t_1)$.

For further convenience, we express the above ROMs in functional forms. Let $g_n(\cdot, t)$ and $\tilde{\varphi}_n(\cdot, t)$ represent functions in \mathcal{X}_N and $[\tilde{\mathcal{X}}_n]^d$, respectively, whose degrees of freedom are uniquely identified by the vectors $G_n(t)$ and $\tilde{\Phi}_n(t)$, respectively. Then, we have the reduced approximation

$$g_N(\cdot, t) \approx g_n(\cdot, t), \quad \tilde{\varphi}(\cdot, t) \approx \tilde{\varphi}_n(\cdot, t). \quad (26)$$

Remark 1 (POD modes for g_N and $\tilde{\varphi}$) *To balance accuracy with computational cost, one can consider different number of POD modes to approximate $g_N(\cdot, t)$ and $\tilde{\varphi}(\cdot, t)$. Such an approach would require an a-posteriori error indicator, which will be a part of our future investigations.*

3.2.2. ROM for u_N

Replacing the above reduced approximations in the decomposition for $u_N(\cdot, t)$ given in (21), we finally recover a ROM for $u_N(\cdot, t)$ that reads

$$u_N(\cdot, t) \approx u_n(\cdot, t) := g_n(\tilde{\varphi}_n(\cdot, t), t). \quad (27)$$

We now consider the following question: Given that g_n and $\tilde{\varphi}_n$ are accurate approximations (in some sense) of $g_N(\cdot, t)$ and $\tilde{\varphi}(\cdot, t)$, respectively, how well does $u_n(\cdot, t)$ approximates $u_N(\cdot, t)$? The result below provides an answer to some extent. In essence, the result establishes that

$$\|u_N - u_n\|_{L^\infty([0,T]; L^1(\Omega))} = \mathcal{O}(\delta), \quad (28)$$

where δ is the error in approximating g_n and $\tilde{\varphi}_n$. Thus, in terms of scaling with δ , the ROM u_n performs as good as the ROMs g_n and $\tilde{\varphi}_n$. Later, we further elaborate upon the result. First, let us summarize the result below. For notational simplicity, we consider a 1D spatial domain i.e., $d = 1$. A similar result can be obtained for multi-D problems.

Theorem 1 Let $d = 1$. Assume that for all $t \in [0, T]$, $\varphi(\cdot, t)$, $\tilde{\varphi}(\cdot, t)$ and $\tilde{\varphi}_n(\cdot, t)$ are $W^{1,\infty}(\Omega)$ homeomorphisms, where $W^{1,\infty}(\Omega)$ is a Sobolev space of functions with bounded weak-derivatives upto the first-order. Let $n \in \mathbb{N}$ be such that for a $\delta \geq 0$ it holds

$$\|g_N - g_n\|_{L^\infty([0,T];L^1(\Omega))} \leq \delta \quad \text{and} \quad \|\tilde{\varphi} - \tilde{\varphi}_n\|_{L^\infty([0,T];W^{1,\infty}(\Omega))} \leq \delta. \quad (29)$$

Then, the error $\|u_N - u_n\|_{L^\infty([0,T];L^1(\Omega))}$ satisfies

$$\|u_N - u_n\|_{L^\infty([0,T];L^1(\Omega))} \leq C_1(u, \tilde{\varphi})\delta + \delta^2. \quad (30)$$

Above, $C_1(u, \tilde{\varphi}) := (\|u\|_{L^\infty([0,T];BV(\Omega))} + \|\nabla \tilde{\varphi}\|_{L^\infty([0,T] \times \Omega)})$, $|\Omega|$ represents the Lebesgue measure of Ω , and $BV(\Omega)$ is a space of functions with bounded variations.

Proof 1 See [Appendix A](#).

Remarks related to the above result are in order.

1. **Diffeomorphic φ :** Empirically, the image registration technique that we outline later provides a diffeomorphic $\varphi(\cdot, t)$ when the maximum displacement $\sup_{x \in \Omega} |\varphi(x, t) - x|$ is not *too large*—see [2] for further details. In the context of image registration, these are the so-called small-displacement problems. As we demonstrate later, even for this sub-class of problems, our method significantly outperforms the standard DMD approach. However, for problems with large displacements, our registration technique is inadequate and should be replaced with the framework proposed in [5]. We plan to undertake such an approach in the future.
2. **Gradient of $\tilde{\varphi}$:** The size of the Jacobian $\|\nabla \tilde{\varphi}\|_{L^\infty}$ should not be too large. Otherwise, because of the term $C_1(u, \tilde{\varphi})$ appearing in the error bound (30), our ROM for $u_N(\cdot, t)$ might be inaccurate. As shown in [55], $\|\nabla \tilde{\varphi}\|_{L^\infty}$ is small if the spatial discontinuities in $u_N(\cdot, t)$ do not move too far away from the discontinuities in a reference solution $u_N(\cdot, t_{\text{ref}})$, where $t_{\text{ref}} \in [0, T]$. Later in [Section 3.3](#) we will further elaborate on this point.
3. **Homeomorphic $\tilde{\varphi}_n$:** We assume that the ROM $\tilde{\varphi}_n(\cdot, t)$ is a homeomorphism. We acknowledge that this is a difficult property to guarantee. Indeed, all of the previous works in [12, 39, 49, 57, 65], which try to either approximate or generalize (via regression) a spatial transform, cannot guarantee this property. The reason being that we (and all of these previous works) compute $\tilde{\varphi}_n(\cdot, t)$ in a linear reduced space. In our context, the linear reduced space is spanned by the POD modes and is not a subset of all the homeomorphisms over Ω .¹ Therefore, without any additional constraints on the POD coefficients, a POD approximation might not be a homeomorphism over Ω . Nevertheless, for an $n \ll N$, the linear approximation $\tilde{\varphi}_n$ is still accurate in the L^p (or $W^{p,\infty}$) sense. Furthermore, it is cheap and easy to compute, and in our experience, if the condition on the movement of the discontinuities stated in the previous point is satisfied then, it is a homeomorphism with well-behaved derivatives. We will elaborate further with numerical examples.
4. **Generality of the error bound:** The error bound in (30) is general in the sense that it is independent of the technique used to compute snapshots of the spatial transform φ . As a result, it is also valid for a registration technique different from ours—the optimal transport-based [47] or the Lagrangian coordinates based [33, 36] techniques being two noteworthy examples.

3.3. Snapshots of φ

To realize the above technique, we require the snapshots $\{\varphi(\cdot, t)\}_{t \in \mathcal{Z}_{\text{train}}}$. We consider an image registration technique to compute these snapshots. The technique is outlined below, we refer to [54, 57] for additional details. We restrict our discussion (and our numerical examples) to a square spatial domain $\Omega = [0, 1]^2$. The subsequent procedure can be generalized to a class of curved spatial domains by partitioning it into subsets, followed by mapping each of these subsets onto a unit square [60].

¹A span of POD modes contains the zero element which is not a homeomorphism from Ω to Ω .

Consider the splitting

$$\Psi(\cdot, t) := \text{Id} - \varphi(\cdot, t), \quad (31)$$

where $\Psi(\cdot, t)$ is the displacement field, and $\text{Id}(x) := x$. We consider a $\Psi(\cdot, t)$ that belongs to the space $\mathcal{P}_M := [\mathbb{P}_M]^2$ that reads

$$\mathbb{P}_M := \text{span}\{L_{jk}\Upsilon\}_{j,k=1,\dots,M}, \quad (32)$$

where $L_{ij}(x)$ is a product of the i -th and j -th Legendre polynomials $l_i(x_1)$ and $l_j(x_2)$, respectively, and $\Upsilon(x) := \prod_{m=1}^d x_m(1-x_m)$. By the definition of Υ , we find the boundary condition $\Psi(x, t) = 0$. Note that this is a stronger version of the boundary conditions than that proposed in Proposition-2.3 of [57]. At least for the test cases we consider, these boundary conditions provide reasonable results without any additional constraints on the Jacobian of φ . One may also consider periodic boundary conditions with a Fourier series expansion for $\Psi(\cdot, t)$ —see [2].

We compute $\Psi(\cdot, t)$ using the minimization problem

$$\Psi(\cdot, t) = \arg \min_{\Psi^* \in \mathcal{P}_M} (\mathcal{M}(\Psi^*, t, t_{\text{ref}}) + \mathcal{R}(\Psi^*)), \quad (33)$$

where $\mathcal{M}(\cdot, \cdot)$ is a so-called matching criterion, and the regularization term $\mathcal{R}(\cdot)$ penalizes the spatial regularity of Ψ , which promotes a diffeomorphic $\varphi(\cdot, t)$ and provides numerical stability [30, 56]. The matching criterion \mathcal{M} should be such that the transformed solution $g_N(\cdot, t)$, at least ideally, has no discontinuities along the temporal domain and consequently, as compared to $u_N(\cdot, t)$, has much better approximability in a linear reduced space. Following the empirical success reported in [39, 57, 65], we set

$$\mathcal{M}(\Psi^*, t, t_{\text{ref}}) := \|u_N(\text{Id} + \Psi^*, t) - u_N(\cdot, t_{\text{ref}})\|_{L^2(\Omega)}^2, \quad \text{and} \quad \mathcal{R}(\Psi^*) := \epsilon \|\Delta \Psi^*\|_{L^2(\Omega; \mathbb{R}^2)}^2. \quad (34)$$

Following the previous works in [54, 57, 59], we set $\epsilon = 10^{-3}$. We compute M iteratively using the technique outlined in Section-3.2.1 of [54]—we do not repeat the details here for brevity. Furthermore, the initial guess is set using the re-ordering scheme proposed in Section-3.1.2 of [57]. To perform optimization, we resort to the `fmincon` routine of MATLAB.

The choice of the reference time-instance $t_{\text{ref}} \in [0, T]$ is key in computing an accurate spatial transform. We consider a fixed time-invariant t_{ref} and set $t_{\text{ref}} = T/2$. We expect such a choice to be accurate for problems where: (i) the discontinuity topology i.e., the number and the relative orientation of the discontinuities, does not change; and (ii) relative to the discontinuities in $u_N(\cdot, t_{\text{ref}})$, the discontinuities in any other solution do not move too close or too far away from each other. In case these conditions are violated, one should try to satisfy them locally by partitioning the time domain into subsets, followed by a local-in-time choice for t_{ref} —see [55, 64]. Akin to the framework in [3], this procedure will result in a local-in-time reduced space, which will adapt to the changing discontinuity topology. We plan to develop such an approach in the future. For now—similar to [39, 65]—we assume that the above two conditions are satisfied. Also note that these two conditions ensure that the gradients of φ and $\tilde{\varphi}$ are well behaved, which, as noted in Theorem 1, is desirable.

Remark 2 (Approximability of $\tilde{\varphi}(\cdot, t)$ in the POD basis) *For our ROM to be accurate, both the transformed solution $g_N(\cdot, t)$ and the de-transformer $\tilde{\varphi}(\cdot, t)$ should be well approximable in the POD basis. Assuming that our registration technique works well, we expect $g_N(\cdot, t)$ to be well approximable in the POD basis. For $\tilde{\varphi}(\cdot, t)$ to be sufficiently regular in time and thus (hopefully) well approximable in the POD basis, the spatial discontinuities in the solution $u(\cdot, t)$ should be sufficiently regular in time. Precisely, if $u(\cdot, t)$ is discontinuous along the curve $\mathcal{D}(t) \subset \Omega$ then, we want $\mathcal{D}(\cdot)$ to be sufficiently regular—the precise quantification of the required regularity and the condition that makes $\mathcal{D}(\cdot)$ regular is given in Corollary-3.4 of [55]. We insist that for several problems of practical interest, and particularly for our numerical examples, this condition is indeed satisfied.*

3.4. Why learn a dynamical system for $\tilde{\varphi}$?

Assume that the solution $u(\cdot, t)$ satisfies a dynamical system of the form $\partial_t u = \mathcal{L}(u)$, with \mathcal{L} being some differential operator in space. Then, one can check that the transformed solution $g(\cdot, t)$ satisfies $\partial_t g = \mathcal{L}_\varphi(g)$, with \mathcal{L}_φ being a transformed differential operator that depends upon φ —see [57, 59] for some examples. This justifies our motivation behind learning a dynamical system (followed by learning a ROM) for g . However, it is not obvious why $\tilde{\varphi}$ would satisfy a dynamical system. Actually, if $\tilde{\varphi}(\cdot, t)$ is a diffeomorphism and if for all $x \in \Omega$, $\tilde{\varphi}(x, \cdot) \in C^1([0, T])$ then, one can show that it satisfies a dynamical system of the form

$$\partial_t \tilde{\varphi}(x, t) = v(\varphi(x, t), t) \quad \forall (x, t) \in \Omega \times [0, T], \quad (35)$$

where $v \in C^1([0, T] \times \bar{\Omega}, \mathbb{R}^d)$ is a velocity field induced by $\tilde{\varphi}(x, t)$. The above result follows from the Cauchy–Lipschitz theorem and is well studied in the image registration literature—see [5, 35], for instance.

A few comments related to the above equation are in order.

1. **Relation to image registration:** For a certain class of image registration techniques, the velocity field results from an optimization problem, and the snapshots of the spatial transform are computed using the above ODE [5]. Our set-up is the exact opposite. Using the snapshots of $\tilde{\varphi}$, we learn (and then reduce) the dynamics of the above ODE.
2. **Relation to Lagrangian DMD:** Choosing $v(x, t)$ as the characteristic velocity of the hyperbolic PDE and computing $\tilde{\varphi}$ using the above ODE, we recover the Lagrangian DMD approach developed in [33]. However, as explained earlier in the introduction, the resulting ODE is well-posed only for non-intersecting characteristic curves, which, for non-linear problems, limits the applicability of the Lagrangian DMD approach to space-time smooth solutions. Furthermore, for both linear and non-linear problems, to derive an explicit expression for the characteristic velocity, we necessarily require the explicit form of the underlying PDE, which might be unavailable in a data-driven setting.

3.5. The standard DMD approach

We briefly recall the standard DMD approach since we use it in our numerical experiments. Let $U(t) \in \mathbb{R}^N$ represent a vector that contains the degrees of freedom of $u_N(\cdot, t)$. In the standard approach, one does not perform any solution decomposition and directly chooses $\mathcal{F} \equiv U$ in the framework developed earlier in Section 2. This results in a ROM that reads [1]

$$U(t) \approx U_n(t) := \mathcal{U}_n^U W^U \exp(\omega^U t) b^U, \quad (36)$$

where all the matrices are as defined earlier in Section 2. For further discussion, we recall that \mathcal{U}_n^U contains the first n left singular vectors of a snapshot matrix given as

$$\mathcal{S}_{1, K-1}(U) := (U(t_1), \dots, U(t_{K-1})). \quad (37)$$

3.5.1. Interpolation and extrapolation regime

The above ROM can either be used in the interpolation regime $t \in [0, T]$ or in the extrapolation regime $t > T$. For hyperbolic PDEs, extrapolation is a much more difficult task than interpolation. One can easily find examples where a ROM that is convergent (with n) in the interpolation regime might fail to converge in the extrapolation regime. Let us elaborate further.

Assuming that the training set \mathcal{Z}_{train} is dense in $[0, T]$, in the interpolation regime, we expect to find an $n \in \mathbb{N}$ to meet any given tolerance $\delta > 0$ on the error $\sup_{t \in [0, T]} \|U(t) - U_n(t)\|_{l^2}$. Of course, if $u(\cdot, t)$ has temporal discontinuities, which is usually the case, the value of n might scale poorly with δ ; for instance, for the 1D advection equation with a unit step function as the initial data, n scales as $\mathcal{O}(1/\delta^2)$ —see [9] for a proof. Nevertheless, at least there exists an n —no matter how large—such that one can expect to meet a given error tolerance. Equivalently, we expect the reduced approximation to converge in the interpolation regime—empirical results in [1] further corroborate our claim.

In the extrapolation regime, irrespective of the number of snapshots collected, our reduced approximation might not converge. There are a plethora of examples where the solution is orthogonal to all other solutions at previous time instances—see [33, 44, 48, 49]. Equivalently, for $T < t_0 \leq t$ and for all $t^* \in [0, T]$, we have $\langle U(t), U(t^*) \rangle_{L^2} = 0$. As a result, for all $n \in \mathbb{N}$ and for all $t_0 < t$, the solution $U(t)$ is orthogonal to the approximation space $\text{range}(\mathcal{U}_n)$, resulting in no convergence. We emphasize that the orthogonality condition can also hold for space-time smooth solutions, which, we expect, are well approximable in the interpolation regime—[33] provides an example. Furthermore, even if the solution is not orthogonal to $\text{range}(\mathcal{U}_n)$, an approximation in $\text{range}(\mathcal{U}_n)$ can wrongly predict the shock locations in $u_N(\cdot, t)$. This usually results in large error values. In our numerical experiments, the standard DMD approach will perform poorly in the extrapolation regime. The above discussion explains our findings to some extent.

In contrast to the solution, the transformed solution g_N exhibits minimal transport-dominated behaviour. As a result, it is well approximable both in the interpolation and the extrapolation regime—a similar comment holds for the de-transformer $\tilde{\varphi}$. However, for both the standard DMD and our new approach, as shown in the next section, the error in the extrapolation regime grows monotonically with time. Thus, despite the de and re-composition, for hyperbolic PDEs, accurate long-time prediction remains a challenging task—authors in [33] make similar observations. Let us recall that a DMD-based ROM for non-linear parabolic and elliptic problems faces similar challenges [34].

4. Summary

In Algorithm 2 and Algorithm 3, we summarize the offline and the online stages of the algorithm, respectively. The offline phase relies on the `get_DMD_matrices` routine that we summarize in Algorithm 1. This routine provides the different matrices required to compute a DMD-based ROM.

Algorithm 1. `get_DMD_matrices` routine

- 1: **Input:** $n, \mathcal{S}_{1,K}, \Delta t$
 - 2: **Output:** $\mathcal{U}_n, W, \omega, b$
 - 3: $[\mathcal{U}_n, \Sigma_n, \mathcal{V}_n] \leftarrow \text{svds}(\mathcal{S}_{1,K-1}, n)$ {svds is the in-built MATLAB routine.}
 - 4: $\mathcal{A}_n \leftarrow \mathcal{U}_n^* \mathcal{S}_{2,K} \mathcal{V}_n \Sigma_n^{-1}$ { \mathcal{A}_n is the POD reduced matrix.}
 - 5: $[W, \lambda] \leftarrow \text{eig}(\mathcal{A}_n)$ {eig is the in-built MATLAB routine.}
 - 6: $\omega \leftarrow \log(\lambda)/\Delta t, b \leftarrow (\mathcal{U}_n W)^\dagger s_1$ { s_1 is the first columns of $\mathcal{S}_{1,K}$.}
-

Algorithm 2. Summary: Offline phase

- 1: **Input:** $n, \mathcal{S}_{1,K}(G), \mathcal{S}_{1,K}(\tilde{\Phi}), \Delta t$
 - 2: **Output:** $\{\mathcal{U}_n^G, W^G, \omega^G, b^G\}, \{\mathcal{U}_n^{\tilde{\Phi}}, W^{\tilde{\Phi}}, \omega^{\tilde{\Phi}}, b^{\tilde{\Phi}}\}$
 - 3: $[\mathcal{U}_n^G, W^G, \omega^G, b^G] \leftarrow \text{get_DMD_matrices}(n, \mathcal{S}_{1,K}(G), \Delta t)$
 - 4: $[\mathcal{U}_n^{\tilde{\Phi}}, W^{\tilde{\Phi}}, \omega^{\tilde{\Phi}}, b^{\tilde{\Phi}}] \leftarrow \text{get_DMD_matrices}(n, \mathcal{S}_{1,K}(\tilde{\Phi}), \Delta t)$
-

Algorithm 3. Summary: Online phase

- 1: **Input:** $\{\mathcal{U}_n^G, W^G, \omega^G, b^G\}, \{\mathcal{U}_n^{\tilde{\Phi}}, W^{\tilde{\Phi}}, \omega^{\tilde{\Phi}}, b^{\tilde{\Phi}}\}, t$
 - 2: **Output:** $u_n(\cdot, t)$
 - 3: $G_n(t) \leftarrow \mathcal{U}_n^G W^G \exp(\omega^G t) b^G$
 - 4: $\tilde{\Phi}_n(t) \leftarrow \mathcal{U}_n^{\tilde{\Phi}} W^{\tilde{\Phi}} \exp(\omega^{\tilde{\Phi}} t) b^{\tilde{\Phi}}$
 - 5: $g_n(\cdot, t) \leftarrow G_n(t), \tilde{\varphi}_n(\cdot, t) \leftarrow \tilde{\Phi}_n(t)$
 - 6: $u_n(\cdot, t) = g_n(\tilde{\varphi}_n(\cdot, t), t)$
-

5. Numerical Results

We abbreviate our DMD approach as TS–DMD, where TS stands for transformed snapshots. The standard approach (outlined in [Section 3.5](#)) is abbreviated as DMD. Also recall that we abbreviate the high-fidelity solver as HF.

5.1. Description of test case

Following is a description of the test cases we consider.

1. **Test-1 (1D advection)** We consider the 1D advection equation

$$\partial_t u + \partial_x u = 0, \text{ on } \Omega \times [0, T], \quad u = u_0, \text{ on } \Omega \times \{0\}. \quad (38)$$

We set $\Omega = (-0.2, 2)$ and $T = 0.8$. The initial data u_0 reads

$$u_0 = \chi_{[\delta, \delta+0.5]}, \quad (39)$$

where χ_A represents a characteristic function of the set $A \subset \mathbb{R}$. We set $\delta := -0.2$. Along the boundary $\partial\Omega \times [0, T]$, we set $u = 0$.

2. **Test-2 (1D wave equation):** We consider the 1D wave equation (rewritten as a first order system)

$$\partial_t u + A \partial_x u = 0, \text{ on } \Omega \times [0, T], \quad (40)$$

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

$$A := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (41)$$

We choose $\Omega = (-0.3, 3)$, and $T = 0.6$. The initial data is given by the linear superposition

$$\begin{aligned} u_1(x, t = 0) &= \frac{1}{\sqrt{2}} (w_1(x) + w_2(x)), \\ u_2(x, t = 0) &= \frac{1}{\sqrt{2}} (-w_1(x) + w_2(x)), \end{aligned} \quad (42)$$

where w_1 and w_2 read

$$\begin{aligned} w_1(x) &:= (\sin(2\pi(x + 0.2)) + 1) \chi_{[\delta_1 - 0.5, \delta_1]}(x), \\ w_2(x) &:= (\sin(2\pi(x - 2.3)) + 1) \chi_{[\delta_2 - 0.5, \delta_2]}(x). \end{aligned} \quad (43)$$

We set $\delta_1 := 0.3$, and $\delta_2 := 2.8$. Along the boundary $\partial\Omega \times [0, T]$, we prescribe $u = 0$.

3. **Test-3 (2D Burgers' equation):** We consider the 2D Burgers' equation given as

$$\partial_t u(x, t) + \left(\frac{1}{2}, \frac{1}{2} \right)^T \cdot \nabla u(x, t)^2 = 0, \quad \forall (x, t) \in \Omega \times [0, T]. \quad (44)$$

The initial data u_0 reads

$$u_0 = \chi_{[0, 0.5]^2}. \quad (45)$$

We set $\Omega = (-0.1, 1.4)^2$, and $T = 1$. Along the boundary $\partial\Omega \times \mathcal{Z}$, we prescribe $u = 0$.

5.2. Error quantification

For a given time-instance $t \in [0, \infty)$, we define the relative $L^1(\Omega)$ error as

$$E(n, t) := \frac{1}{\|u_N(\cdot, t)\|_{L^1(\Omega)}} (\|u_N(\cdot, t) - u_n(\cdot, t)\|_{L^1(\Omega)}), \quad (46)$$

where $u_N(\cdot, t)$ is the HF solution, and the reduced solution $u_n(\cdot, t)$ can either result from DMD or TS-DMD. The average error inside a time-interval $[t_0, t_1]$ is given by

$$E_a(n, t_0, t_1) := \frac{1}{\#\mathcal{Z}_{test}} \sum_{t \in \mathcal{Z}_{test}} E(n, t), \quad (47)$$

where $\mathcal{Z}_{test} \subset [t_0, t_1]$ contains 100 uniformly and independently sampled parameters from the time interval $[t_0, t_1]$. The average error defined as such will allow us to separately study the error in the interpolation $t \in [0, T]$ and the extrapolation $t > T$ regime. Recall that we expect DMD to perform poorly in the extrapolation regime—see [Section 3.5](#). The following results will further corroborate our expectations.

5.3. Test-1

We partition Ω into 4×10^3 uniform elements, and collect snapshots at 500 uniformly placed time instances inside $[0, 0.8]$. We approximate the displacement field $\Psi(\cdot, t)$ (defined in [\(31\)](#)) in the polynomial space $\mathcal{P}_{M=4}$, where the value of M results from the procedure outlined in [Section 3.3](#).

5.3.1. Singular value decay

For further convenience, we recall that:

1. $\mathcal{S}_{1,K}(U)$ contains the (untransformed) snapshots $\{u_N(\cdot, t)\}_{t \in t_{train}}$;
2. $\mathcal{S}_{1,K}(G)$ contains the transformed snapshots $\{g_N(\cdot, t)\}_{t \in t_{train}}$;
3. $\mathcal{S}_{1,K}(\tilde{\Phi})$ contains the de-transformer's snapshots $\{\tilde{\varphi}(\cdot, t)\}_{t \in t_{train}}$.

For these three snapshot matrices, we compare the singular value decay. We denote the n -th singular value by σ_n , and study the scaled singular value σ_n/σ_1 .

[Figure 2a](#) compares the singular value decay. The matrix $\mathcal{S}_{1,K}(U)$ has a slow singular value decay, indicating that a large number of POD modes (i.e., the value of n in [\(36\)](#)) might be required to well approximate the solution. This slow decay is triggered by the temporal discontinuities in the solution—see [Figure 2b](#). In contrast, as shown in [Figure 2c](#), spatial discontinuities in the transformed solution move very little in the time domain. This induces a fast singular value decay in the matrix $\mathcal{S}_{1,K}(G)$ and improves the approximability of a transformed solution in the POD basis. Similarly, the de-transformer $\tilde{\varphi}(\cdot, t)$ appears to be smooth in the space-time domain, which results in a fast singular value decay in $\mathcal{S}_{1,K}(\tilde{\Phi})$ —see [Figure 2d](#). Also note that $\tilde{\varphi}(\cdot, t)$ does not exhibit any steep gradients, which, as noted in [Theorem 1](#), is desirable.

Observe that the singular values of $\mathcal{S}_{1,K}(G)$ appear to stagnate. The error introduced by this stagnation is of little practical relevance because it is of the same order of magnitude as the error introduced by the HF solver. We refer to [\[55\]](#) for further details.

5.3.2. Approximation error

Following the discussion in [Section 3.5](#), we study the approximation error separately for the interpolation $t \in [0, 0.8]$ and extrapolation $t > 0.8$ regime. We first consider the interpolation regime. [Figure 3a](#) compares the average error. TS-DMD outperforms DMD. The difference is the most pronounced for $n = 11$, for which TS-DMD results in an error of 0.42%. Compare this to DMD which provides an error of 18.4%. For $n \geq 9$, the error from TS-DMD appears to stagnate because of the stagnation in the singular values reported earlier.

We now consider the extrapolation regime $t > 0.8$ —recall that we only collect snapshots inside $[0, 0.8]$. [Figure 3b](#) presents the average error. Error from DMD oscillates around 100%, and

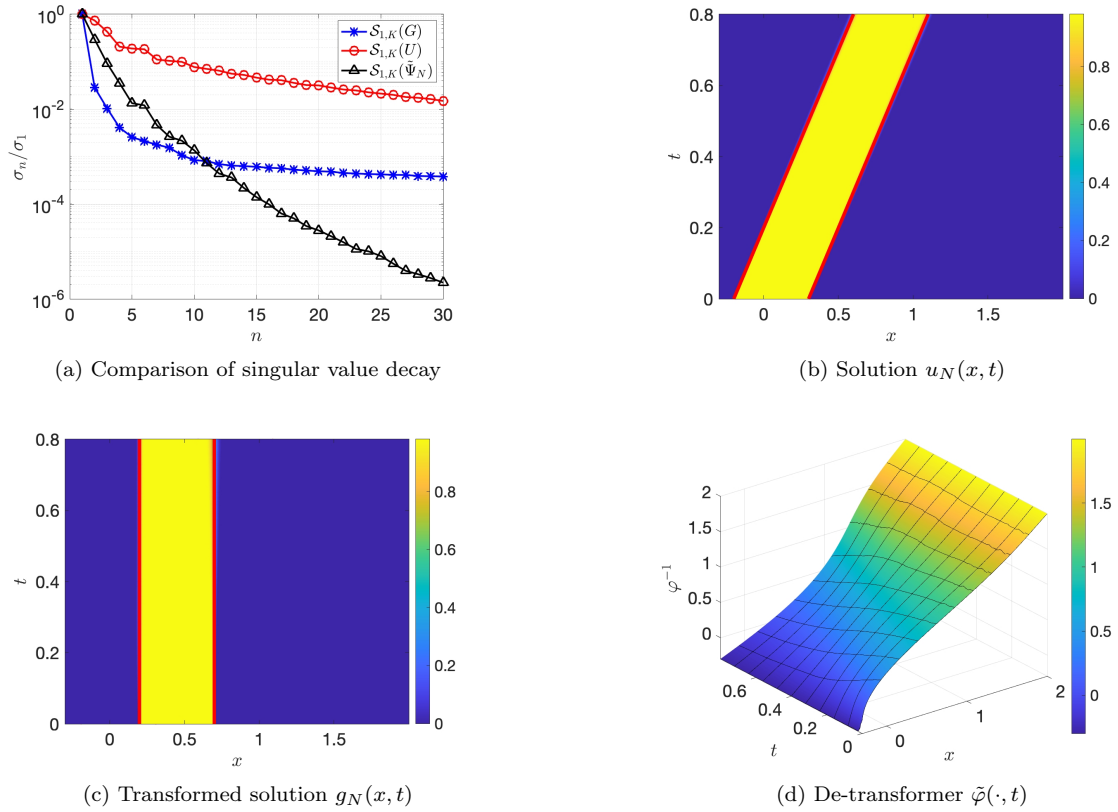


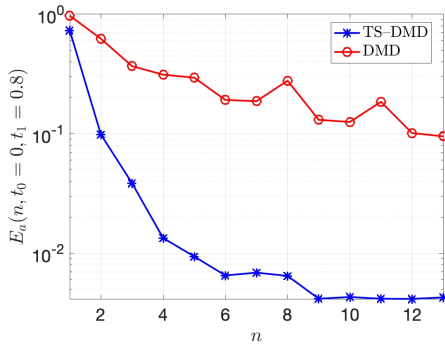
Figure 2.: Results for test-1.

it appears that increasing the value of n does not lead to convergence. TS-DMD performs much better. For all $n \geq 5$, it results in a maximum error of 9%, which is almost two orders of magnitude smaller than that resulting from DMD.

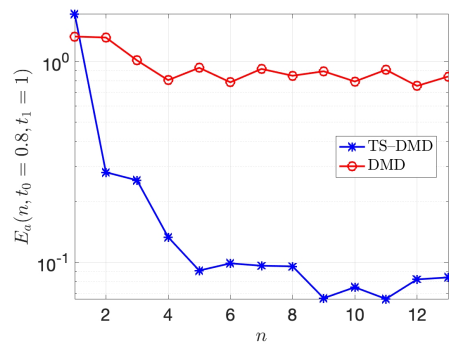
Let us elaborate on the poor performance of DMD for the extrapolation regime—Section 3.5 provides further elaboration. Ignoring the errors introduced by the HF solver, one can check that support of the solution $u_N(\cdot, t)$ is given via $K_t := [-0.2 + t, 0.3 + t] \subset \Omega$. Furthermore, the inner-product $\langle u_N(\cdot, t), u_N(\cdot, t^*) \rangle_{L^2(\Omega)}$ reads $|K_t \cap K_{t^*}|$. For $t > 0.8$, as the gap $t - 0.8$ increases, the maximum overlap $\sup_{t^* \in [0, 0.8]} |K_t \cap K_{t^*}|$ decreases. Consequently, in the L^2 -sense, as time $t > 0.8$ increases, $u_N(\cdot, t)$ points more and more in a direction that is orthogonal to the previously collected snapshots. This decreases the approximability of $u_N(\cdot, t)$ in the span of snapshots collected from $[0, 0.8]$ (and thus, also in the span of the POD modes) and thus, results in a poor DMD approximation. We emphasize that increasing the number of solution snapshots does not improve the accuracy.

On the other hand, the support of the transformed solution $g_N(\cdot, t)$ is time-invariant—ignoring the slight discontinuity misalignment reported above—and is given by $[-0.2 + t_{\text{ref}}, 0.3 + t_{\text{ref}}]$, where $t_{\text{ref}} = 0.4$. As a result, even for $t > 0.8$, $g_N(\cdot, t)$ is well-approximable in the span of the transformed snapshots collected from $[0, 0.8]$. Similar observation holds for the de-transformer $\tilde{\varphi}(\cdot, t)$. An accurate approximation of $g_N(\cdot, t)$ and $\tilde{\varphi}(\cdot, t)$, eventually, results in an accurate TS-DMD approximation of $u_N(\cdot, t)$.

For $n = 13$, time variation of the L^1 error is shown in Figure 4a. At all times, the error from TS-DMD is almost an order of magnitude smaller than that from DMD. Observe that in the extrapolation regime—i.e., for $t > 0.8$ —the error from both the methods grows monotonically with t ; our findings here are consistent with that in [33, 34]. Nevertheless, TS-DMD provides a reasonable extrapolation with a maximum error of 16%.

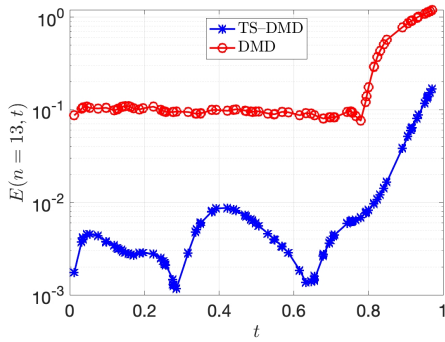
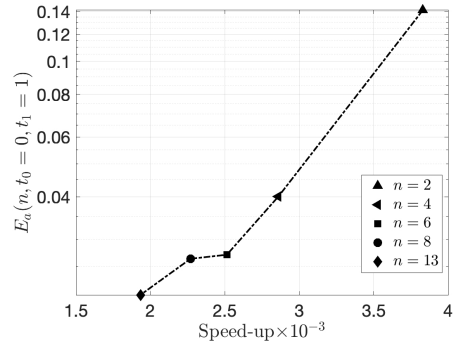


(a) Average error: Interpolation regime



(b) Average error: Extrapolation regime

Figure 3.: Results for test-1.


 (a) Temporal behaviour of the L^1 error: $n = 13$


(b) Speed-up vs. error

Figure 4.: Results for test-1.

5.3.3. Solution comparison

A major drawback of the DMD approach is its highly oscillatory solution—see [Figure 5](#). These oscillations are triggered by the temporal discontinuities in the solution and get stronger inside the extrapolation regime. In contrast, the TS-DMD solution exhibits no such oscillations. Note that oscillations in a ROM for advection-dominated problems are not just limited to the DMD approach. Indeed, as explained earlier, they are a shortcoming of using a linear reduced space and for that matter, also the works in [\[13, 21, 33, 36\]](#) report similar oscillations.

5.3.4. Study of $\tilde{\varphi}_n$

In [Theorem 1](#), we assumed that the ROM $\tilde{\varphi}_n(\cdot, t)$ for $\tilde{\varphi}(\cdot, t)$ is a homeomorphism. Here, we justify this assumption empirically. We first define a Jacobian

$$\tilde{\mathcal{J}}_n(x, t) := \partial_x \tilde{\varphi}_n(x, t),$$

where we interpret the derivative in a weak sense. For all $t \in [0, T]$, we want the Jacobian to be positive. Since, for all $t \in [0, T]$, $\tilde{\varphi}_n(\partial\Omega, t) = \partial\Omega$, this would then imply that $\tilde{\varphi}_n(\cdot, t)$ is a homeomorphism [\[57\]](#). We restrict our study to two limiting values of n : $n = 1$ and $n = 10$. Qualitatively, the results remain similar for all the other intermediate values. We study the Jacobian at 100 uniformly sampled points from $[0, T]$.

For $n = 1$, [Figure 6a](#) plots $\tilde{\varphi}_n$ over the space-time domain. With little deviations, $\tilde{\varphi}_n$ is almost the same as the identity mapping. Furthermore, the minimum value of the Jacobian $\inf_{x \in \Omega} \tilde{\mathcal{J}}_n(x, t)$ stays well above zero, which is desirable—see [Figure 6b](#). Note that because the value of n is small, $\tilde{\varphi}_n$ is a crude approximation of the true $\tilde{\varphi}(\cdot, t)$ shown in [Figure 2d](#). Nevertheless, increasing n to

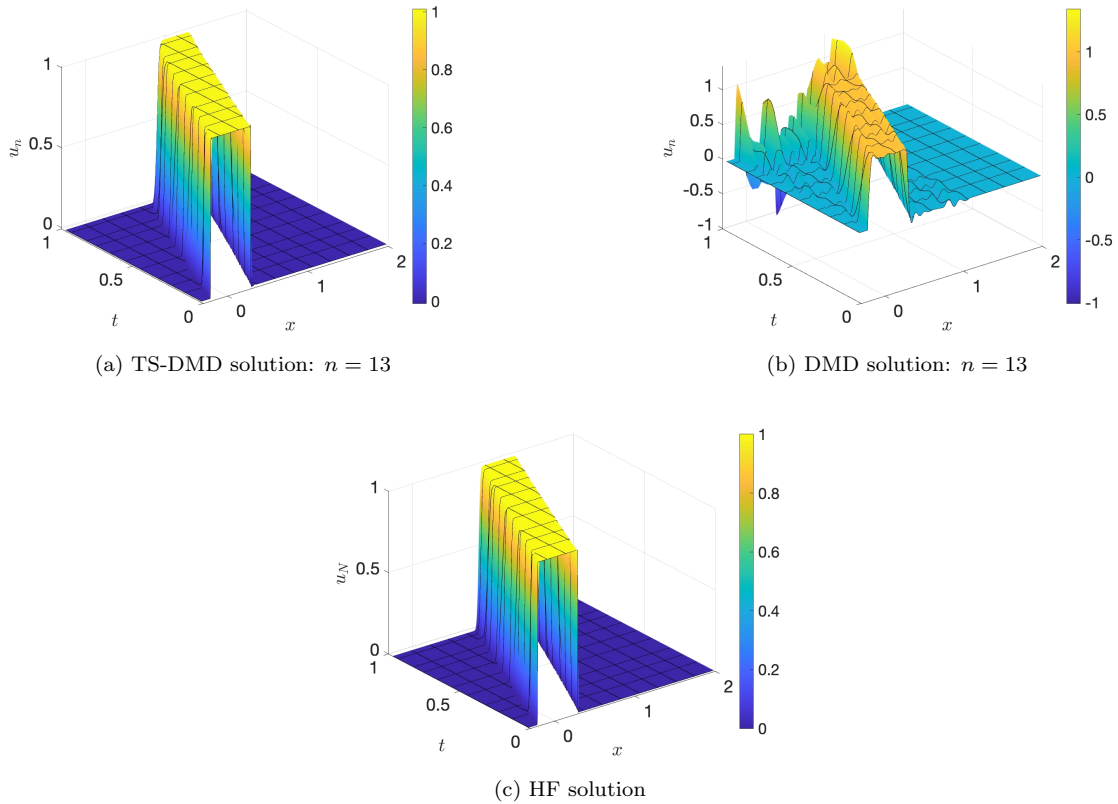


Figure 5.: Results for test-1.

10, improves the approximation quality—see Figure 6c. Also, as shown in Figure 6d, for $n = 10$, the Jacobian is still positive.

5.3.5. Speed-up vs. accuracy

We study the speed-up offered by the TS–DMD approach. We define the speed-up as

$$\kappa := \frac{\sum_{t \in \mathcal{Z}_{test}} \tau_{HF}(t)}{\sum_{t \in \mathcal{Z}_{test}} \tau_{ROM}(t)}, \quad (48)$$

where $\tau_{ROM}(t)$ and $\tau_{HF}(t)$ denote the CPU-time required by the TS–DMD and HF solver, respectively, to compute a solution at time $t \in [0, \infty)$. The set \mathcal{Z}_{test} contains 100 uniformly and independently sampled points from $[0, 1]$. The CPU-time is computed using the `tic-toc` function of MATLAB.

Recall that TS–DMD is an equation-free approach. To compute a solution at any $t \in [0, \infty)$, we just need to compute the sum in (15). The HF solver, on the other hand, depending upon the value of t , requires some finite number of time steps to compute the solution. Therefore, we expect TS–DMD to be much more efficient than the HF solver.

Figure 4b presents the results. For clarity, we plot only a handful of values. As anticipated, in general, both the error and speed-up decrease upon increasing n . Furthermore, the speed-up stays well above 10^3 . The minimum speed-up of 1.9×10^3 results from $n = 13$ and corresponds to an average error of 1.7%.

5.4. Test-2

We discretize Ω with 4×10^3 uniform elements, and collect snapshots at 500 uniformly placed time instances inside $[0, 0.6]$. We approximate $\Psi(\cdot, t)$ in the polynomial space $\mathcal{P}_{M=5}$, where the value of

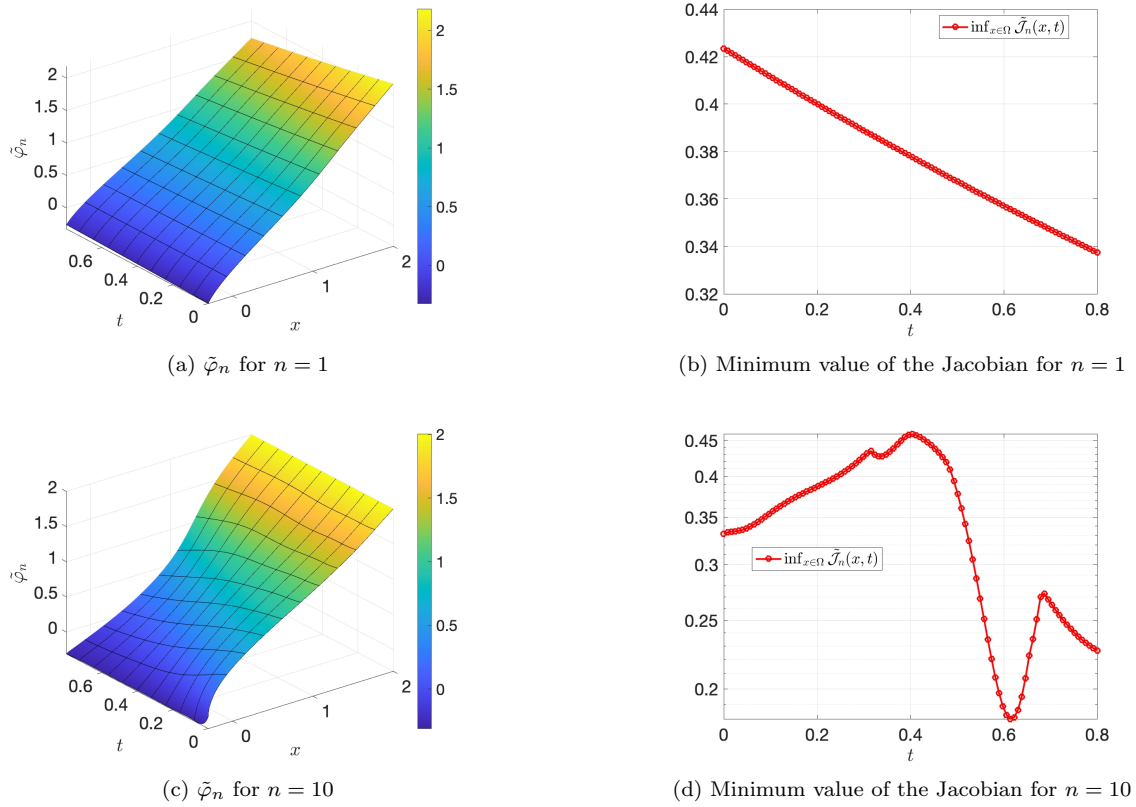


Figure 6.: Results for test-1.

M results from the procedure outlined in Section 3.3. We report the results for the first solution component u_1 . Results for u_2 are similar and not discussed for brevity.

5.4.1. Singular value decay

Figure 7a compares the singular value decay for the three snapshot matrix described earlier. Similar to the previous test case, because of the temporal discontinuities in the solution, singular values of the matrix $\mathcal{S}_{1,K}(U)$ decay slowly. This is indicative of the poor approximability of the solution in a linear reduced space. Comparatively, the snapshot matrices $\mathcal{S}_{1,K}(G)$ and $\mathcal{S}_{1,K}(\tilde{\Phi})$ exhibit a fast singular value decay. For instance, for $n=4$, we find

$$\begin{aligned} \mathcal{S}_{1,K}(U) : \frac{\sigma_n}{\sigma_1} &= 14.7 \times 10^{-2}, & \mathcal{S}_{1,K}(G) : \frac{\sigma_n}{\sigma_1} &= 0.3 \times 10^{-2}, \\ \mathcal{S}_{1,K}(\tilde{\Phi}) : \frac{\sigma_n}{\sigma_1} &= 1.3 \times 10^{-2}. \end{aligned} \quad (49)$$

Our results indicate that it is better to first approximate the transformed solution and the de-transformer in the POD basis, followed by a re-composition based approximation for the solution. The average approximation errors reported below will further corroborate our claim.

5.4.2. Approximation error

Figure 8a presents the average error in the interpolation regime $t \in [0, 0.6]$. Our observations remain similar to the previous test case. The TS-DMD technique outperforms DMD. For $n=10$, it results in an error of 0.35%, which is almost fifteen times smaller than the error of 5.2% resulting from the DMD approach. Notice that because of the stagnation in the singular value decay reported earlier, the error from TS-DMD stagnates for $n \gtrsim 6$.

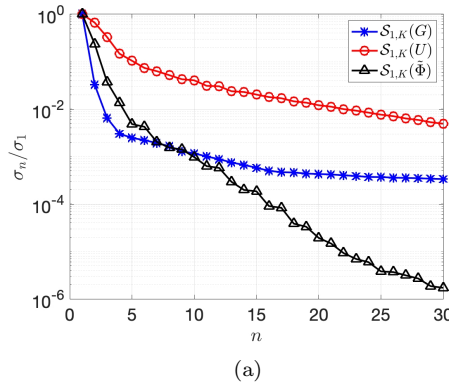
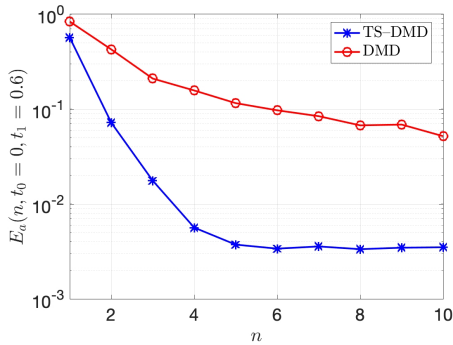


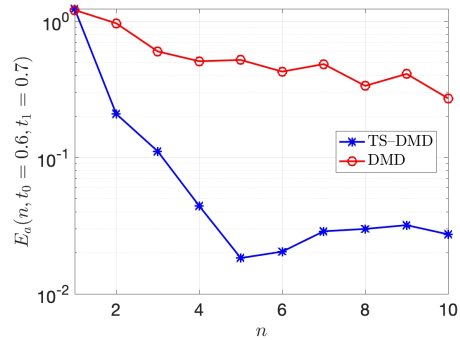
Figure 7.: Results for test-2. Comparison of singular value decay.

As one might anticipate from our earlier discussion, DMD performs poorly in the extrapolation regime $t \in [0.6, 0.7]$ —see Figure 8b. For $n = 10$, it results in an error of $\approx 27\%$, which is almost ten times larger than the error of 2.6% resulting from TS–DMD. Same as the interpolation regime, the error from TS–DMD almost stagnates for $n \gtrsim 7$.

For $n = 10$, time variation of the relative L^1 error is shown in Figure 9a. For both the methods, in the extrapolation regime, the error increases monotonically with time. Furthermore, as compared to DMD, at all time instances, TS–DMD provides an error that is at least an order of magnitude smaller. At the final time $t = 0.7$, both the methods result in the maximum error with the error being 4.4% and 37.1% for TS–DMD and DMD, respectively.



(a) Average error: interpolation regime



(b) Average error: extrapolation regime

Figure 8.: Results for test-2.

5.4.3. Solution comparison

For $n = 5$, Figure 10 compares the different solutions. As expected, DMD provides an oscillatory solution. These oscillations are particularly strong in the extrapolation regime $t > 0.6$. Notice that in the extrapolation regime, DMD largely misrepresents the solution. It gets both the shape of the sin bump and the location of discontinuities wrong. Furthermore, it exhibits oscillations outside of the support of the HF solution. In contrast, TS–DMD accurately approximates the solution both in the interpolation and the extrapolation regime. In the *eye-ball* norm, it is almost indistinguishable from the HF solution. Upon a closer inspection, next to the spatial discontinuities, one can find minor over and undershoots in the TS–DMD solution. These artifacts are a result of the minor misalignment in discontinuities reported earlier.

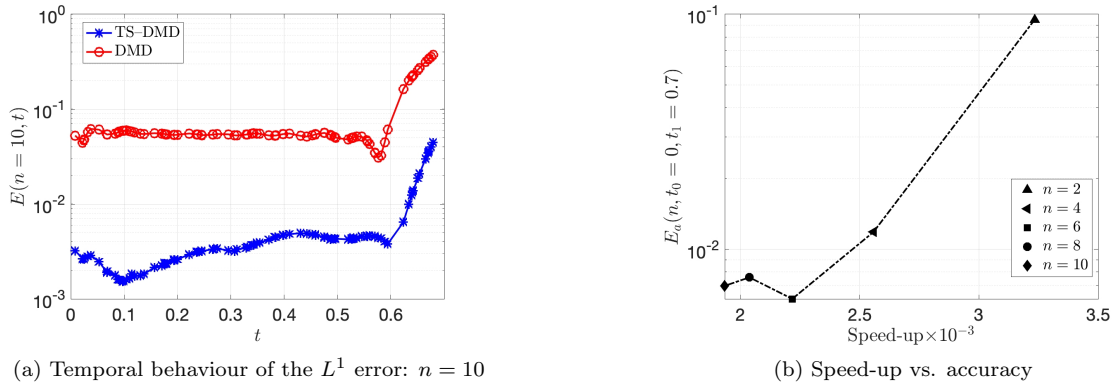


Figure 9.: Results for test-2.

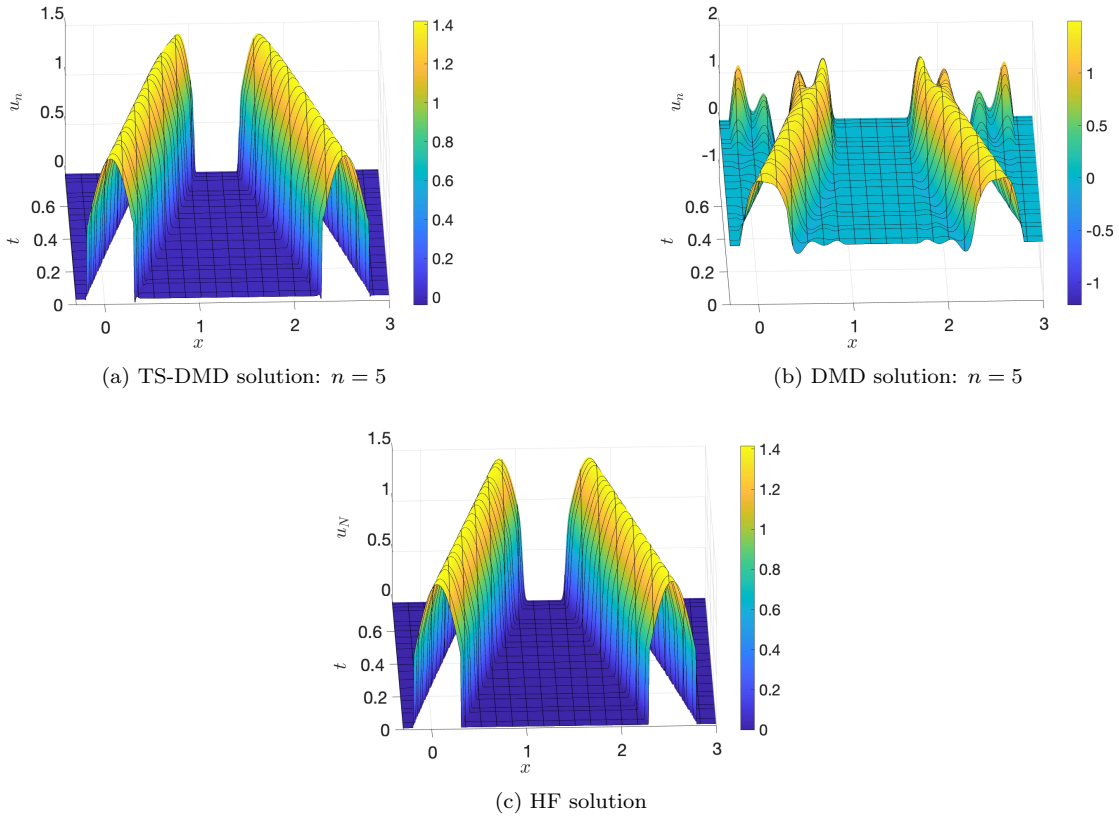


Figure 10.: Results for test-1.

5.4.4. Speed-up vs. accuracy

Figure 9b compares the speed-up defined in (48) to the average error $E_a(n, t_0, t_1)$. Our findings remain similar to the previous test case. In general, both the error and the speed-up decrease upon increasing n . For $n = 10$, we recorded a minimum speed-up of 1.9×10^3 with a corresponding average error of 0.6%.

5.5. Test-3

We discretize Ω with 300×300 grid cells, and collect snapshots at 50 uniformly placed temporal points inside $[0, 0.8]$. Note that we collect much fewer snapshots than in the previous two test cases. For multi-dimensional problems, snapshot collection is usually expensive and we demonstrate that our technique works well with fewer snapshots. We approximate the displacement field $\Psi(\cdot, t)$ in $\mathcal{P}_{M=6}$.

5.5.1. Singular value decay

Figure 11 compares the singular value decay of the three matrices: $\mathcal{S}_{1,K}(U)$, $\mathcal{S}_{1,K}(G)$ and $\mathcal{S}_{1,K}(\tilde{\Phi})$. Our observations are similar to the previous test cases. Singular values of the matrix $\mathcal{S}_{1,K}(U)$ decay slowly, indicating that the solution has poor approximability in the POD basis. Comparatively, the singular values of $\mathcal{S}_{1,K}(G)$ and $\mathcal{S}_{1,K}(\tilde{\Phi})$ decay faster. For instance, consider $n = 4$, for which we find

$$\begin{aligned} \mathcal{S}_{1,K}(U) : \frac{\sigma_n}{\sigma_1} &= 18.8 \times 10^{-2}, & \mathcal{S}_{1,K}(G) : \frac{\sigma_n}{\sigma_1} &= 4.7 \times 10^{-2}, \\ \mathcal{S}_{1,K}(\tilde{\Phi}) : \frac{\sigma_n}{\sigma_1} &= 3.5 \times 10^{-3}. \end{aligned} \quad (50)$$

Notice that the singular value of $\mathcal{S}_{1,K}(G)$ and $\mathcal{S}_{1,K}(\tilde{\Phi})$ is almost four and seventy times smaller, respectively, to that of $\mathcal{S}_{1,K}(U)$.

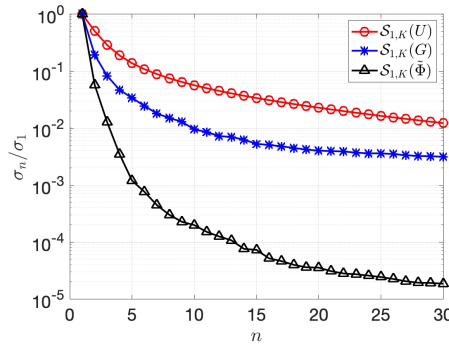


Figure 11.: Results for test-3. Comparison of singular value decay.

5.5.2. Approximation error, solution comparison and speed-up

Results for the convergence study remain similar to the previous test cases and are not repeated again for brevity. Rather, we set $n = 5$ and study the approximation error. First, we consider the interpolation regime $t \in [0, 1]$, for which we find the following average errors

$$\text{Interpolation regime} \begin{cases} \text{TS-DMD} : & E_a(n = 5, 0, 1) = 1.7 \times 10^{-2}, \\ \text{DMD} : & E_a(n = 5, 0, 1) = 14.9 \times 10^{-2}. \end{cases} \quad (51)$$

Observe that the error from TS-DMD is almost nine times smaller than that from DMD. Figure 12 compares the solution at $t = 1$. Same as earlier, the DMD solution exhibits strong oscillations and provides a poor approximation. In contrast, the TS-DMD solution is accurate and (almost) oscillation-free. Let us recall that $n = 5$ is just 5.5×10^{-3} percent of the HF dimension N . For such a small value of n , a relative error of 1.7% can be considered reasonable.

Now consider the extrapolation regime $t \in [1, 1.1]$, for which we find the errors

$$\text{Extrapolation regime} \begin{cases} \text{TS-DMD} : & E_a(n = 5, 1, 1.1) = 8.0 \times 10^{-2}, \\ \text{DMD} : & E_a(n = 5, 1, 1.1) = 30.1 \times 10^{-2}. \end{cases} \quad (52)$$

Notice that both the methods perform worse than in the interpolation regime. Nevertheless, TS-DMD provides a much better solution with an average that is almost four times smaller than that

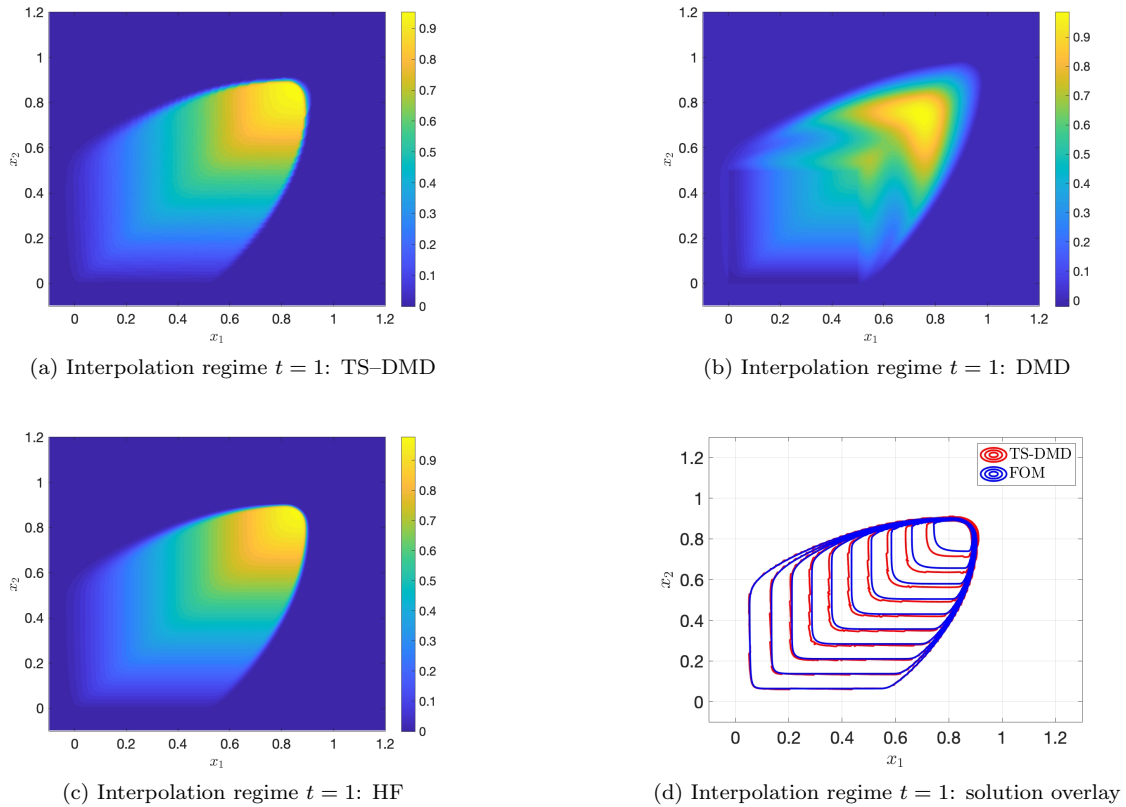


Figure 12.: Results for test-3. Solutions for TS–DMD and DMD computed with $n = 5$.

from DMD. For most practical applications, an error of 30% resulting from DMD is undesirable. The current test case, and the previous ones, indicate that for hyperbolic PDEs, DMD is largely inadequate in the extrapolation regime. Our claim is further corroborated by Figure 13 that compares the solutions at $t = 1.1$. Compared to the interpolation regime, the DMD solution exhibits stronger oscillations. In contrast, the TS–DMD solution is (almost) oscillation-free and provides an accurate approximation of the HF solution.

Although TS–DMD provides a much better solution than DMD, it has two inadequacies: (i) close to the leading edge, it under predicts the shock location by almost 2%, and (ii) it fails to accurately capture the shape of the rarefaction fan—see Figure 13d. The linearity of the best-fit evolution operator (i.e., the matrix \tilde{A} in (5)) could be a possible reason behind these inadequacies. Note that the Burgers’ equation is non-linear. Consequently, it exhibits a solution-dependent shock location and a rarefaction fan; recall that linear hyperbolic PDEs (with a sufficiently regular velocity field) do not exhibit rarefaction fans and have a solution independent shock-location. The action of the best-fit linear operator may be incapable of accurately predicting these non-linear transport-dominated effects. At least for the Burgers’ equation, introducing quadratic nonlinearities via the operator inference (see [42]) approach might improve the results. Further investigation is required to establish anything conclusive.

Figure 14 depicts the temporal behaviour of the error $E(n = 5, t)$. TS-DMD provides a much better approximation with error values almost an order of magnitude smaller than in DMD. Similar to the previous test cases, in the extrapolation regime, the error from both the methods increases monotonically with time. Notice that for TS–DMD, the error is particularly large close to $t = 0$. This is because at $t = 0$, the discontinuity-topology of the solution suddenly changes—see Section 3.3 and Section-6 of [54] for further details. As a result, compared to the other time instances, the image registration technique provides a slightly inaccurate spatial transform.

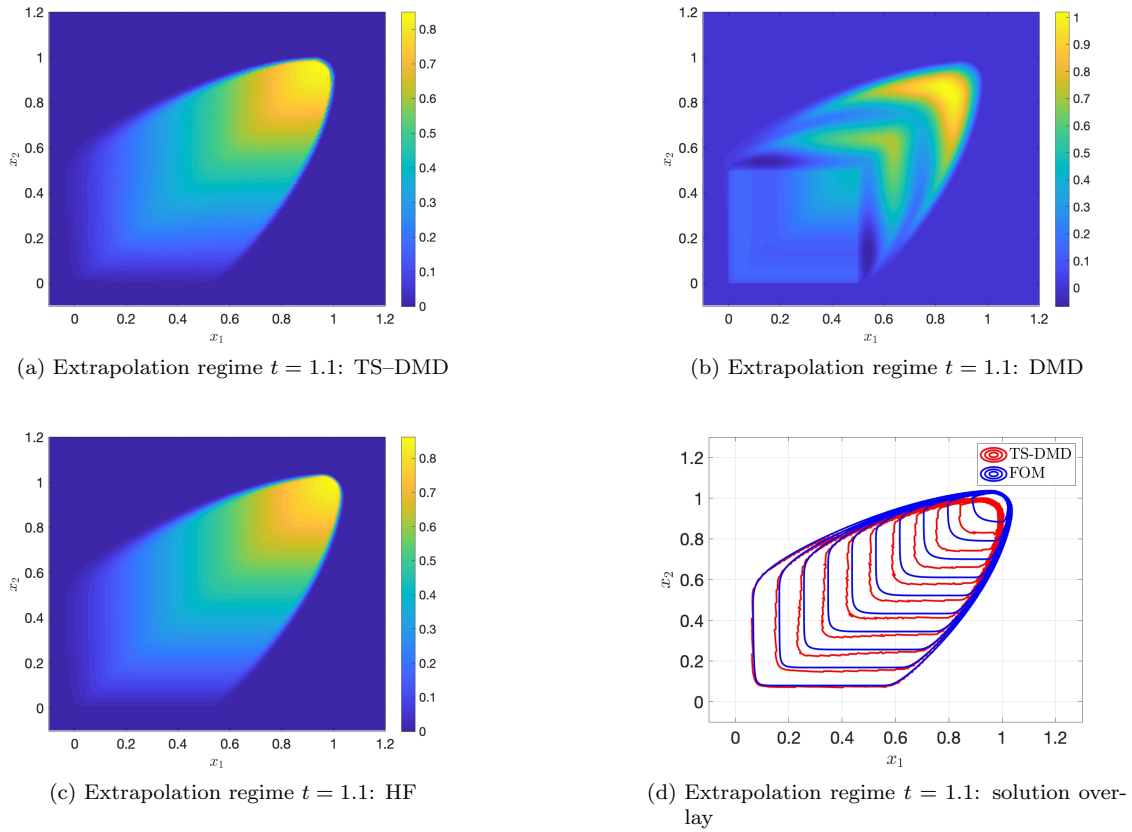


Figure 13.: Results for test-3. Solutions for TS-DMD and DMD computed with $n = 5$.

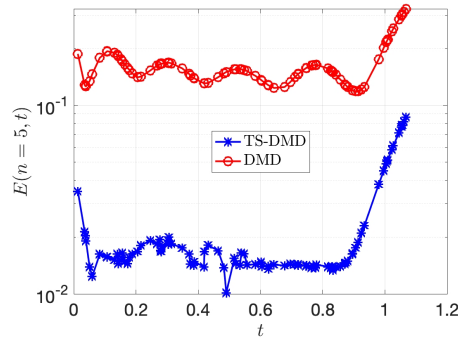


Figure 14.: Results for test-3. Temporal behaviour of the L1-error defined in (46). Simulations performed with $n = 5$.

5.5.3. Speed-up vs. accuracy

Consider the speed-up defined in (48). The test set \mathcal{Z}_{test} consists of 100 uniformly and independently sampled points from $[0, 1.1]$. For $n = 5$, TS-DMD results in a speed-up of 1.7×10^3 with an average error of 2.3%.

6. Conclusions

Solutions to hyperbolic PDEs exhibit temporal discontinuities which hinders their approximability in a linear reduced space. This makes the standard approach of learning a ROM directly for the

solution ineffective. We resolve this problem by introducing a de and re-composition step. In the decomposition step, we express the solution as a composition of a transformed solution and a de-transformer. As compared to the solution, both these objects have good approximability in a linear reduced space and allow for efficient learnability via standard ROM learning techniques. We then recover a ROM for the solution via a re-composition.

Our de and re-composition architecture has mainly three appealing properties. Firstly, it does not require an explicit form of the underlying hyperbolic PDE. Secondly, it is detached from any pre-existing numerical implementation of a ROM learning toolbox. Lastly, de and re-composition can easily be deactivated for problems that are tractable via standard, previously developed, learning techniques. This way, one can avoid the extra cost associated with these two steps.

We used DMD to learn the ROMs and performed numerical experiments involving linear, non-linear, and multi-D (in space) hyperbolic PDEs. Consistent with the previous works, we identified the following drawbacks of a standard DMD approach that learns a ROM directly for the solution. Firstly, both in the interpolation and the extrapolation regime, the ROM is plagued with oscillations. Secondly, in the extrapolation regime, the ROM largely misrepresents the solution and results in an error of fifty to hundred percent. Furthermore, at least empirically, even with a large reduced space, in the extrapolation regime, the ROM does not appear to converge to the true solution.

To a significant extent, our approach overcomes the aforementioned inadequacies. Both inside the interpolation and the extrapolation regime, it results in an error that is at least five to ten times smaller than in the standard approach. Furthermore, owing to the de-transformation step, it results in a ROM that is (almost) oscillation free. However, similar to the standard approach, it results in an error that increases monotonically with time in the extrapolation regime. As a result, accurate extrapolation was only possible for short time intervals.

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).

A. Error bound

By triangle's inequality, we find

$$\begin{aligned} \|u_N(\cdot, t) - u_n(\cdot, t)\|_{L^1(\Omega)} &\leq \|g_N(\tilde{\varphi}(\cdot, t), t) - g_N(\tilde{\varphi}_n(\cdot, t), t)\|_{L^1(\Omega)} \\ &\quad + \|g_N(\tilde{\varphi}_n(\cdot, t), t) - g_n(\tilde{\varphi}_n(\cdot, t), t)\|_{L^1(\Omega)} \\ &=: A_1 + A_2. \end{aligned} \quad (53)$$

We first bound A_1 . Provided our HF solver is total variation diminishing (TVD), we have

$$|u_N(\cdot, t)|_{BV(\Omega)} \leq |u(\cdot, t)|_{BV(\Omega)}, \quad (54)$$

where $BV(\Omega)$ is a space of functions with bounded variations. Note that the TVD property is an immediate consequence of the CFL condition in Lemma-20.2 of [19]. Since $\varphi(\cdot, t)$ is a $W^{1,\infty}(\Omega, \mathbb{R}^d)$ homeomorphism, we find

$$|g_N(\cdot, t)|_{BV(\Omega)} = |u_N(\cdot, t)|_{BV(\Omega)}. \quad (55)$$

Then, using the assumptions on $\tilde{\varphi}$ and $\tilde{\varphi}_n$, Lemma-3.1 of [65] and the above two relations (54) and (55), we bound A_1 via

$$A_1 \leq |g_N(\cdot, t)|_{BV(\Omega)} \|\tilde{\varphi}(\cdot, t) - \tilde{\varphi}_n(\cdot, t)\|_{L^\infty(\Omega)} = |u_N(\cdot, t)|_{BV(\Omega)} \delta \leq |u(\cdot, t)|_{BV(\Omega)} \delta. \quad (56)$$

An integral transform provides a bound for A_2 that reads

$$\begin{aligned} A_2 &\leq \|\nabla \tilde{\varphi}_n(\cdot, t)\|_{L^\infty(\Omega)} \|g_N(\cdot, t) - g_n(\cdot, t)\|_{L^1(\Omega)} \\ &\leq (\|\nabla \tilde{\varphi}(\cdot, t)\|_{L^\infty(\Omega)} + \delta) \|g_N(\cdot, t) - g_n(\cdot, t)\|_{L^1(\Omega)}. \end{aligned} \quad (57)$$

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] Y. Amit. A nonlinear variational problem for image matching. *SIAM Journal on Scientific Computing*, 15(1):207–224, 1994.
- [3] D. Amsallem, M. J. Zahr, and C. Farhat. Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, 92(10):891–916, 2012.
- [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. *Comput. Methods Appl. Mech. Engrg.*, 372:113433, 17, 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, C. Himpe, and T. Mitchell. On reduced input-output dynamic mode decomposition. *Advances in Computational Mathematics*, 44(6):1751–1768, 2018.
- [9] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox. *Model Reduction and Approximation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.
- [10] W.-J. Beyn and V. Thümmler. Freezing solutions of equivariant evolution equations. *SIAM J. Appl. Dyn. Syst.*, 3(2):85–116, 2004.
- [11] D. A. Bistrián and I. M. Navon. Randomized dynamic mode decomposition for nonintrusive reduced order modelling. *Internat. J. Numer. Methods Engrg.*, 112(1):3–25, 2017.
- [12] 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.
- [13] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem. The GNAT method for non-linear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.
- [14] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.
- [15] P. Constantine and G. Iaccarino. Reduced order models for parameterized hyperbolic conservation laws with shock reconstruction. *Annual Brief: Center for Turbulence Research*, 6(1), 2012.
- [16] F. Dubois and P. Le Floch. Boundary conditions for nonlinear hyperbolic systems of conservation laws. *Journal of Differential Equations*, 71(1):93–122, 1988.
- [17] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

-
- [18] 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 Math. Model. Numer. Anal.*, 54(6):2159–2197, 2020.
- [19] R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. In *Solution of Equation in R^n (Part 3), Techniques of Scientific Computing (Part 3)*, volume 7 of *Handbook of Numerical Analysis*, pages 713–1018. Elsevier, 2000.
- [20] N. R. Franco, A. Manzoni, and P. Zunino. A deep learning approach to reduced order modelling of parameter dependent partial differential equations. *arXiv:2103.06183*, 2021.
- [21] S. Fresca, L. Dede’, and A. Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *Journal of Scientific Computing*, 87(2):61, 2021.
- [22] C. Greif and K. Urban. Decay of the Kolmogorov N-width for wave problems. *Applied Mathematics Letters*, 96:216–222, 2019.
- [23] 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.
- [24] B. Haasdonk and M. Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *M2AN. Mathematical Modelling and Numerical Analysis*, 42(2):277–302, 2008.
- [25] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, Cham, 2016.
- [26] 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.
- [27] L. Jiang and M. Li. Model reduction for nonlinear multiscale parabolic problems using dynamic mode decomposition. *International Journal for Numerical Methods in Engineering*, 121(16):3680–3701, 2020.
- [28] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014.
- [29] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. Efficient nonlinear manifold reduced order model. *arXiv:2011.07727*, 2020.
- [30] 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.
- [31] J. N. Kutz, X. Fu, and S. L. Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.
- [32] K. Lee and K. T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 32, 2020.
- [33] H. Lu and D. M. Tartakovsky. Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena. *Journal of Computational Physics*, 407:109229, 17, 2020.
- [34] H. Lu and D. M. Tartakovsky. Prediction accuracy of dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 42(3):A1639–A1662, 2020.
- [35] M. I. Miller, A. Trouvé, and L. Younes. On the metrics and Euler-Lagrange equations of computational anatomy. *Annual Review of Biomedical Engineering*, 4(1):375–405, 2002.

- [36] R. Mojjani and M. Balajewicz. Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows. *arXiv:1701.04343*, 2017.
- [37] R. Mojjani and M. Balajewicz. Physics-aware registration based auto-encoder for convection dominated PDEs. *arXiv:2006.15655*, 2020.
- [38] S. Mowlavi and T. P. Sapsis. Model order reduction for stochastic dynamical systems with continuous symmetries. *SIAM Journal on Scientific Computing*, 40(3):A1669–A1695, 2018.
- [39] 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.
- [40] M. Ohlberger and S. Rave. Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing. *Comptes Rendus Mathematique*, 351(23):901 – 906, 2013.
- [41] B. Peherstorfer. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *SIAM Journal on Scientific Computing*, 42(5):A2803–A2836, 2020.
- [42] 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.
- [43] 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.
- [44] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*. Springer International Publishing, 2016.
- [45] D. Rafiq and M. A. Bazaz. Nonlinear model order reduction via nonlinear moment matching with dynamic mode decomposition. *International Journal of Non-Linear Mechanics*, 128:103625, 2021.
- [46] S. A. Renganathan, R. Maulik, and V. Rao. Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil. *Physics of Fluids*, 32(4):047110, 2020.
- [47] D. Rim and K. T. Mandli. Displacement interpolation using monotone rearrangement. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4):1503–1531, 2018.
- [48] D. Rim, S. Moe, and R. J. LeVeque. Transport reversal for model reduction of hyperbolic partial differential equations. *SIAM/ASA Journal on Uncertainty Quantification*, 6(1):118–150, 2018.
- [49] D. Rim, B. Peherstorfer, and K. T. Mandli. Manifold approximations via transported subspaces: Model reduction for transport-dominated problems. *arXiv:1912.13024*, 2019.
- [50] C. W. Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1):115–129, 2004.
- [51] C. W. Rowley and J. E. Marsden. Reconstruction equations and the Karhunen-Loève expansion for systems with symmetry. *Physica D. Nonlinear Phenomena*, 142(1-2):1–19, 2000.
- [52] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.
- [53] G. Rozza, D. B. P. Huynh, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):229, 2008.
- [54] N. Sarna and P. Benner. Data-driven model order reduction for problems with parameter-dependent jump-discontinuities. *arXiv:2105.00547*, 2021.

- [55] N. Sarna, J. Giesselmann, and P. Benner. Data-driven snapshot calibration via monotonic feature matching. *arXiv:2009.08414*, 2020.
- [56] A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *IEEE Transactions on Medical Imaging*, 32(7):1153–1190, 2013.
- [57] T. Taddei. A registration method for model order reduction: Data compression and geometry reduction. *SIAM Journal on Scientific Computing*, 42(2):A997–A1027, 2020.
- [58] T. Taddei, S. Perotto, and A. Quarteroni. Reduced basis techniques for nonlinear conservation laws. *ESAIM: M2AN*, 49(3):787–814, 2015.
- [59] T. Taddei and L. Zhang. Space-time registration-based model reduction of parameterized one-dimensional hyperbolic PDEs. *arXiv:2004.06693*, 2020.
- [60] T. Taddei and L. Zhang. Registration-based model reduction in complex two-dimensional geometries. *arXiv:2101.10259*, 2021.
- [61] G. Tissot, L. Cordier, N. Benard, and B. R. Noack. Model reduction using dynamic mode decomposition. *Comptes Rendus Mécanique*, 342(6):410–416, 2014.
- [62] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.
- [63] B. van Leer. Towards the ultimate conservative difference scheme. ii. Monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 14(4):361–370, 1974.
- [64] G. Welper. h and hp -adaptive interpolation by transformed snapshots for parametric and stochastic hyperbolic PDEs. *arXiv:1710.11481*, 2017.
- [65] G. Welper. Interpolation of functions with parameter dependent jumps by transformed snapshots. *SIAM Journal on Scientific Computing*, 39(4):A1225–A1250, 2017.
- [66] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.
- [67] 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.