Journal of Computational Physics 444 (2021) 110550

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Extended dynamic mode decomposition for inhomogeneous problems

Hannah Lu, Daniel M. Tartakovsky *

Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Available online 9 July 2021

Keywords:
Reduced-order model
Data-driven
Learning
Inhomogeneous PDE
Residual network

ABSTRACT

Dynamic mode decomposition (DMD) is a powerful data-driven technique for construction of reduced-order models of complex dynamical systems. Multiple numerical tests have demonstrated the accuracy and efficiency of DMD, but mostly for systems described by homogeneous partial differential equations (PDEs) with homogeneous boundary conditions. We propose an extended dynamic mode decomposition (xDMD) approach to cope with the potential unknown sources/sinks in PDEs. Motivated by similar ideas in deep neural networks, we equip our xDMD with two new features. First, it has a bias term, which accounts for inhomogeneity of PDEs and/or boundary conditions. Second, instead of learning a flow map, xDMD learns the residual increment by subtracting the identity operator. Our theoretical error analysis demonstrates the improved accuracy of xDMD relative to standard DMD. Several numerical examples are presented to illustrate this result.

1. Introduction

Complexity of many, if not most, physical and biological phenomena and paucity of measurements undermine the reliability of purely statistical descriptors. Instead, models of such systems are inferred or "learned" from both observational and simulated data and reflect the fundamental laws of nature (e.g., conservation of mass and energy). Various sparse regression techniques [1–4] use a proposed dictionary to "discover" the governing equations from data. The dictionary, comprising plausible spatial and/or temporal derivatives of a state variable, provides functional approximations of different physical laws. Dynamic mode decomposition (DMD) was used to compute the modes of the Koopman operator approximated from a preselected dictionary basis [5]; subsequent studies provided a theoretical analysis of convergence [6], practical guidelines for efficient construction of the dictionary [7], and other aspects of this approach. The data for sparse regression are allowed to be noisy [8], corrupted [9], and limited [10]. Various flavors of deep neural networks (DNN) provide a related dictionary-based approach to PDE learning [11,12]. These and other techniques for equation discovery are as good as a dictionary on which they are based.

A conceptually different, dictionary-free, framework for data-informed predictions is to construct a surrogate (aka reduced-order) model, instead of learning a governing PDE. This framework is often classified as "unsupervised learning" and "equation-free". Much of the research in this field deals with dynamical systems, for which training data are generated by either ordinary differential equations (ODEs) or partial differential equations (PDEs) after spatial discretization. In this context, DMD can be used to construct an optimal linear approximation model for the unknown system [13] and to learn

* Corresponding author.
E-mail addresses: hannahlu@stanford.edu (H. Lu), tartakovsky@stanford.edu (D.M. Tartakovsky).

https://doi.org/10.1016/j.jcp.2021.110550
0021-9991/

the unknown dynamics of chosen observables, rather than of the state itself [14]. The latter is accomplished by utilizing the Koopman theory [15] in order to construct linear models on the observable space, instead of identifying nonlinear models on the state space [16]. Physics-guided selection of observables provides not only better accuracy [17–19], but also a bridge between the understanding of data and physics. Likewise, DNN can be used to build nonlinear surrogate models for ODEs [20,21] and PDEs [22,23]. DNN-based surrogates and reduced-order models (ROMs) [24,25] are invaluable in applications that require a large number of model solves, such as inverse modeling [26,27] and uncertainty quantification [28–31].

Our study contributes to this second framework by extending the range of applicability of DMD-based ROMs to dynamical systems described by inhomogeneous PDEs with inhomogeneous boundary conditions. In various application scenarios, many variants of the standard DMD algorithms were introduced to improve the performance of model reduction. The low-rank approximators are determined from optimization problems that are adapted to control inputs [32], time embeddings [33], higher-order approximations [34], optimal approximation error [35], etc. We adopt a similar methodology by modifying the optimization problem in a way that allows us to cope with a problem's inhomogeneity. A major benefit of our new algorithm is a theoretically guaranteed accuracy improvement relative to the standard DMD algorithm with awareness and identification of the inhomogeneity at almost no extra computational cost. Our extended dynamic mode decomposition (xDMD) borrows ideas from the recent work on residual neural networks (ResNet) to provide an optimal linear approximation model for such systems. Our generalization of the standard DMD includes two ingredients: an added bias term and residual learning. The first builds upon the generalized ResNet [36] that introduces a bias term to model the dynamics described by underlying inhomogeneous ODEs. We extend this idea to systems described by inhomogeneous PDEs and prove the accuracy improvement induced by the added bias term. The second ingredient of xDMD is the learning of effective increments (i.e., the residual of subtracting identity from a flow map) rather than the flow map itself. Although mathematically equivalent to flow map learning, this strategy proved to be highly advantageous in practice and gained traction in the deep-learning community [37], including in its applications to equation recovery [20]. To the best of our knowledge, xDMD is first to fuse these two features and to provide a theoretical estimate of its performance.

In section 2, we provide a problem setup and a detailed formulation of xDMD. A formal proof of the accuracy improvements induced by the added bias term is presented in section 3. A number of numerical experiments are collated in section 4 to evaluate the learning performance of xDMD in terms of representation, extrapolation, interpolation and generalizability. Key results, their implication for applications, and challenges and future work are summarized in section 5.

2. Problem formulation and extended DMD

We consider a real-valued state variable $u(\boldsymbol{\xi}, t)$, whose dynamics is described by a boundary-value problem

$$\begin{cases} \dfrac{\partial u}{\partial t} = \mathcal{L}(u) + S(\boldsymbol{\xi}), & (\boldsymbol{\xi}, t) \in \mathcal{D} \times \mathbb{R}^+; \\ \mathcal{B}(u) = b(\boldsymbol{\xi}), & (\boldsymbol{\xi}, t) \in \partial\mathcal{D} \times \mathbb{R}^+; \\ u(\boldsymbol{\xi}, 0) = u_0(\boldsymbol{\xi}), & \boldsymbol{\xi} \in \mathcal{D}. \end{cases} \tag{2.1}$$

Here, t denotes time; $\boldsymbol{\xi}$ is the spatial coordinate; $\mathcal{D} \subset \mathbb{R}^d$ is the d-dimensional simulation domain bounded by the surface $\partial\mathcal{D}$; $\mathcal{L}$ is a (linear or nonlinear) differential operator that involves spatial derivatives; $\mathcal{B}$ is the boundary differential operator describing Dirichlet, Neumann, and/or Robin boundary conditions; $S(\boldsymbol{\xi})$ and $b(\boldsymbol{\xi})$ represent sources/sinks and boundary functions, respectively; and $u_0(\boldsymbol{\xi})$ is the initial state.

The simulation domain is discretized with a mesh consisting of N elements. A suitable numerical approximation of (2.1) yields a system of (coupled, nonlinear) ODEs,

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}, \mathbf{s}), & \mathbf{u}, \mathbf{s} \in \mathbb{R}^N, \\ \mathbf{u}(0) = \mathbf{u}_0, & \mathbf{u}_0 \in \mathbb{R}^N, \end{cases} \tag{2.2}$$

where $\mathbf{s}$ comes from both $S(\boldsymbol{\xi})$ and $b(\boldsymbol{\xi})$. Let $\boldsymbol{\Phi}_{\Delta t} : \mathbb{R}^N \to \mathbb{R}^N$ denote a flow map, which relates the discretized system state $\mathbf{u}$ at time t to that at time $t + \Delta t$, where Δt is a (sufficiently small) time increment. Since $\mathbf{s}$ is independent of t and acts as a set of parameters, the system (2.2) is time-invariant. Consequently, there exists a flow map $\boldsymbol{\Phi}$, depending only on the time difference $t - t_0$, which represents the solution to (2.2) as $\mathbf{u}(t; \mathbf{u}_0, t_0, \mathbf{s}) = \boldsymbol{\Phi}_{t-t_0}(\mathbf{u}_0; \mathbf{s})$.

Our goal is to learn the dynamic system $\mathbf{f}$, or, more precisely, its reduced-order surrogate, using M temporal snapshots of the solutions. Let $\mathbf{x}^k \equiv \mathbf{u}(t_k)$ and $\mathbf{y}^k \equiv \mathbf{u}(t_k + \Delta t)$ with $k = 1, \ldots, M$, where the time lag between the input and output states, Δt, is assumed to be independent of k for the sake of convenience. The simulation data consist of M pairs $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1}^M$, such that

$$\mathbf{y}^k = \boldsymbol{\Phi}_{\Delta t}(\mathbf{x}^k; \mathbf{s}), \qquad k = 1, \ldots, M. \tag{2.3}$$

Lemma 2.1. *Assume* $\mathbf{f}$ *to be Lipschitz continuous with a Lipschitz constant* L *on a solution manifold* $\mathcal{M} \subset \mathbb{R}^N$. *Define*

$$\mathcal{M}_{\Delta t} = \{\mathbf{x} \in \mathcal{M} : \boldsymbol{\Phi}_{\Delta t}(\mathbf{x}; \mathbf{s}) \in \mathcal{M}\}. \tag{2.4}$$

Then, $\mathbf{\Phi}_{\Delta t}$ is Lipschitz continuous on $\mathcal{M}_{\Delta t}$. Specifically, for any $\mathbf{z}, \tilde{\mathbf{z}} \in \mathcal{M}_{\Delta t}$,

$$\|\mathbf{\Phi}_{\Delta t}(\mathbf{z}; \mathbf{s}) - \mathbf{\Phi}_{\Delta t}(\tilde{\mathbf{z}}; \mathbf{s})\| \le \mathrm{e}^{L\tau} \|\mathbf{z} - \tilde{\mathbf{z}}\|, \qquad t \le \tau \le t + \Delta t. \tag{2.5}$$

Proof. The proof follows directly from the classical numerical analysis results in, e.g., [38, p. 109]. □

Lemma 2.1 imposes requirements on the snapshots data pairs $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1}^M$: the number of data pairs M should be sufficiently large, and the data should be sufficiently rich for the data space to cover the solution space of interest. These requirements are consistent with the core of the Koopman operator theory, which underpins the DMD algorithm, e.g., [39, p. 47] and others [5,14,40]. The error analyses of the DMD algorithms [6,18] also verify the impact of the selection of observables on the success of Koopman methods.

2.1. Standard DMD

Given a dataset of snapshots, $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1}^M$, DMD constructs a best-fit linear operator $\mathbf{A} \in \mathbb{R}^{N\times N}$ such that

$$\mathbf{y}^k \approx \mathbf{A}\mathbf{x}^k, \qquad k = 1, \dots, M. \tag{2.6}$$

Therefore, the matrix $\mathbf{A}$ is determined in a least square sense

$$\mathbf{A} = \underset{\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}}{\operatorname{argmin}} \frac{1}{M} \sum_{k=1}^{M} \|\mathbf{y}^k - \hat{\mathbf{A}}\mathbf{x}^k\|^2. \tag{2.7}$$

Typically, one rewrites the dataset $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1}^M$ in a matrix form,

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^M \\ | & | & & | \end{bmatrix}_{N\times M} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}^1 & \mathbf{y}^2 & \cdots & \mathbf{y}^M \\ | & | & & | \end{bmatrix}_{N\times M}. \tag{2.8}$$

Then, $\mathbf{A}$ is computed as

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^{\dagger} \qquad \text{(standard DMD)}, \tag{2.9}$$

where † denotes the Moore-Penrose inverse.

Remark 2.1. The Moore-Penrose inverse is computed via singular value decomposition (SVD), which requires certain truncation criteria to maintain computational stability. In all our numerical tests, we use the default truncation in the `pinv` command of Matlab.

Remark 2.2. In a typical DMD algorithm, e.g., [39, p. 7], a reduced-order model $\tilde{\mathbf{A}}$ is derived by projecting $\mathbf{A}$ onto the proper orthogonal decomposition (POD) modes. Since the major goal of our study is to obtain a linear approximation model of inhomogeneous PDEs, for which standard DMD algorithms fail, we omit the order-reduction procedure for simplicity.

2.2. Generalized DMD

In order to cope with potential inhomogeneity of the underlying dynamics, the following modification is made in [36]:

$$\mathbf{y}^k \approx \mathbf{A}_{\mathrm{g}}\mathbf{x}^k + \mathbf{b}, \qquad k = 1, \dots, M. \tag{2.10}$$

The matrix $\mathbf{A}_{\mathrm{g}}$ and the vector $\mathbf{b} \in \mathbb{R}^N$ are computed by solving the optimization problem

$$(\mathbf{A}_{\mathrm{g}}, \mathbf{b}) = \underset{\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}, \hat{\mathbf{b}} \in \mathbb{R}^N}{\operatorname{argmin}} \frac{1}{M} \sum_{k=1}^{M} \|\mathbf{y}^k - \hat{\mathbf{A}}\mathbf{x}^k - \hat{\mathbf{b}}\|^2. \tag{2.11}$$

Let us introduce

$$\tilde{\mathbf{X}} := \begin{bmatrix} \mathbf{X} \\ \mathbf{1} \end{bmatrix}_{(N+1)\times M} \tag{2.12}$$

where $\mathbf{1} := [1, 1, \cdots, 1]$ is a vector of size $1 \times M$. Then $\mathbf{A}_{\mathrm{g}}$ and $\mathbf{b}$ are obtained by

$$[\mathbf{A}_{\mathrm{g}}, \mathbf{b}] = \mathbf{Y}\tilde{\mathbf{X}}^{\dagger} \qquad \text{(generalized DMD or gDMD)}. \tag{2.13}$$

Remark 2.3. gDMD can be regarded as a special arrangement of DMD with control (DMDc) [32]. In the latter, the augmented data matrix (2.12) is constructed by stacking the state variable snapshots $\mathbf{x}^k$ and the control input snapshots, which in our context are set to 1 since there is no control. Applications of DMD and the Koopman theory in control are an active research area [41–44], but lie outside the scope of our analysis that focuses on improving the performances of DMD in learning unknown dynamical systems. Of more direct relevance is a connection of gDMD and DMD to dictionary learning (e.g., [7,6,5]): the dictionary composition is treated as the state variable itself and set to 1 in the augmented matrix. However, the computational cost of identifying relevant terms from a proposed dictionary can be prohibitively large for high-dimensional dynamical systems and discretized PDEs. Instead, our framework includes only the bias term, which has physical interpretations in inhomogeneous PDEs.

2.3. Residual DMD

The residual DMD or rDMD borrows a key idea behind ResNet. The latter explicitly introduces the identity operator in a neural network and forces the network to approximate the "residual" of the input-output map. Although mathematically equivalent, this simple transformation proved to improve network performance and became increasingly popular in the machine learning community.

Writing $\mathbf{A} = \mathbf{I} + \mathbf{B}$, where $\mathbf{I}$ is the $(N \times N)$ identity matrix and $\mathbf{B}$ is the remainder, recasts (2.6) as

$$\mathbf{y}^k \approx \mathbf{x}^k + \mathbf{B}\mathbf{x}^k. \tag{2.14}$$

The matrix $\mathbf{B}$ is determined by

$$\mathbf{B} = (\mathbf{Y} - \mathbf{X})\mathbf{X}^\dagger \qquad \text{(residual DMD or rDMD)}. \tag{2.15}$$

It provides an approximation of the "effective increment" [20, definition 3.1], $\boldsymbol{\varphi}_{\Delta t}$, that is defined as

$$\boldsymbol{\varphi}_{\Delta t}(\mathbf{u}; \mathbf{f}, \mathbf{s}) = \Delta t\, \mathbf{f}(\boldsymbol{\Phi}_\tau(\mathbf{u}; \mathbf{s})) \tag{2.16}$$

for some $t \le \tau \le t + \Delta t$ such that

$$\mathbf{u}(t + \Delta t) = \mathbf{u}(t) + \boldsymbol{\varphi}_{\Delta t}(\mathbf{u}; \mathbf{f}, \mathbf{s}). \tag{2.17}$$

2.4. Extended DMD

Combining the modifications introduced in the previous two subsections, we arrive at our extended DMD or xDMD,

$$\mathbf{y}^k \approx \mathbf{x}^k + \mathbf{B}_\mathrm{g}\mathbf{x}^k + \mathbf{b}, \tag{2.18}$$

where $\mathbf{B}_\mathrm{g}$ and $\mathbf{b}$ are computed as

$$[\mathbf{B}_\mathrm{g}, \mathbf{b}] = (\mathbf{Y} - \mathbf{X})\tilde{\mathbf{X}}^\dagger \qquad \text{(extended DMD or xDMD)}. \tag{2.19}$$

3. Relative performance of different DMD formulations

Theorem 3.1. *In the least square sense, gDMD in section 2.2 fits the M snapshots data* $\mathbf{X}$ *and* $\mathbf{Y}$ *better than the standard DMD from section 2.1 does, i.e.,*

$$\frac{1}{M}\sum_{k=1}^{M} \|\mathbf{y}^k - \mathbf{A}_\mathrm{g}\mathbf{x}^k - \mathbf{b}\|^2 \le \frac{1}{M}\sum_{k=1}^{M} \|\mathbf{y}^k - \mathbf{A}\mathbf{x}^k\|^2. \tag{3.1}$$

Proof. The optimization problem (2.11) gives rise to

$$\begin{aligned}
&\frac{1}{M}\sum_{k=1}^{M}\|\mathbf{y}^k-\mathbf{A}_g\mathbf{x}^k-\mathbf{b}\|^2\\
=&\min_{\hat{\mathbf{A}}\in\mathbb{R}^{N\times N},\hat{\mathbf{b}}\in\mathbb{R}^N}\frac{1}{M}\sum_{k=1}^{M}\|\mathbf{y}^k-\hat{\mathbf{A}}\mathbf{x}^k-\hat{\mathbf{b}}\|^2\\
\le&\min_{\hat{\mathbf{A}}\in\mathbb{R}^{N\times N},\hat{\mathbf{b}}\in\mathbb{R}^N}\frac{1}{M}\sum_{k=1}^{M}\left(\|\mathbf{y}^k-\hat{\mathbf{A}}\mathbf{x}^k\|^2+\|\hat{\mathbf{b}}\|^2\right)\\
=&\min_{\hat{\mathbf{A}}\in\mathbb{R}^{N\times N}}\left(\frac{1}{M}\sum_{k=1}^{M}\|\mathbf{y}^k-\hat{\mathbf{A}}\mathbf{x}^k\|^2\right)+\min_{\hat{\mathbf{b}}\in\mathbb{R}^N}\|\hat{\mathbf{b}}\|^2\\
=&\frac{1}{M}\sum_{k=1}^{M}\|\mathbf{y}^k-\mathbf{A}\mathbf{x}^k\|^2+\min_{\hat{\mathbf{b}}\in\mathbb{R}^N}\|\hat{\mathbf{b}}\|^2.
\end{aligned}\tag{3.2}$$

The inequality is derived by triangle inequality and the last equality is achieved by (2.7). Since the equality is achieved with $\hat{\mathbf{b}}=0$, gDMD is equivalent to the standard DMD only when the bias term $\mathbf{b}=0$. □

Remark 3.1. Theorem 3.1 implies that xDMD from section 2.4 fits the M snapshots data $\mathbf{X}$ and $\mathbf{Y}$ better than rDMD from section 2.3 in the least square sense, i.e.,

$$\frac{1}{M}\sum_{k=1}^{M}\|\mathbf{y}^k-\mathbf{x}^k-\mathbf{B}_g\mathbf{x}^k-\mathbf{b}\|^2\le\frac{1}{M}\sum_{k=1}^{M}\|\mathbf{y}^k-\mathbf{x}^k-\mathbf{B}\mathbf{x}^k\|^2.\tag{3.3}$$

Corollary 3.1.1. *Let μ_M be an empirical measure defined on a given dataset $\{\mathbf{x}^1,\cdots,\mathbf{x}^M\}$ by*

$$\mu_M=\frac{1}{M}\sum_{k=1}^{M}\delta_{\mathbf{x}^k},\tag{3.4}$$

where $\delta_{\mathbf{x}^k}$ denotes the Dirac measure at $\mathbf{x}^k$. Then, for any $\mathbf{x}\in\mathcal{M}_{\Delta t}$,

$$\|\mathbf{\Phi}_{\Delta t}(\mathbf{x})-\mathbf{A}_g\mathbf{x}-\mathbf{b}\|^2\le\|\mathbf{\Phi}_{\Delta t}(\mathbf{x})-\mathbf{A}\mathbf{x}\|^2\ \text{ a.s.},\tag{3.5}$$

i.e., the inequality (3.5) holds in the sense of distribution.

Proof. The integral of a test function g with respect to μ_M is given by

$$\int_{\mathcal{M}}g(\mathbf{x})\mathrm{d}\mu_M(\mathbf{x})=\frac{1}{M}\sum_{k=1}^{M}g(\mathbf{x}^k).\tag{3.6}$$

It follows from (3.1) and the definition of $\mathbf{y}^k$ in (2.3) that

$$\frac{1}{M}\sum_{k=1}^{M}\left(\|\mathbf{\Phi}_{\Delta t}(\mathbf{x}^k)-\mathbf{A}\mathbf{x}^k\|^2-\|\mathbf{\Phi}_{\Delta t}(\mathbf{x}^k)-\mathbf{A}_g\mathbf{x}^k-\mathbf{b}\|^2\right)\ge 0.\tag{3.7}$$

Thus, by virtue of (3.6),

$$\int_{\mathcal{M}}\left(\|\mathbf{\Phi}_{\Delta t}(\mathbf{x})-\mathbf{A}\mathbf{x}\|^2-\|\mathbf{\Phi}_{\Delta t}(\mathbf{x})-\mathbf{A}_g\mathbf{x}-\mathbf{b}\|^2\right)\mathrm{d}\mu_M(\mathbf{x})\ge 0.\tag{3.8}$$

Hence, the inequality (3.5) holds in the sense of distributions. □

Remark 3.2. By the same token,

$$\|\mathbf{\Phi}_{\Delta t}(\mathbf{x})-\mathbf{B}_g\mathbf{x}-\mathbf{b}\|^2\le\|\mathbf{\Phi}_{\Delta t}(\mathbf{x})-\mathbf{B}\mathbf{x}\|^2,\ \text{a.s.}\tag{3.9}$$

Theorem 3.2. *Suppose that the assumptions of Lemma 2.1 hold, and further assume that*

1. $\|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\mathbf{x}\|_{L^\infty(\mathcal{M}_{\Delta t})} < +\infty$ *and* $\|\mathbf{\Phi}_{\Delta t} - \mathbf{A}_g\mathbf{x} - \mathbf{b}\|_{L^\infty(\mathcal{M}_{\Delta t})} < +\infty$;
2. $\mathbf{x}^k, \mathbf{y}^k \in \mathcal{M}_{\Delta t}$ *for* $k = 1, \dots, M$.

Let $\mathbf{u}^n_{\text{DMD}}$ *and* $\mathbf{u}^n_{\text{gDMD}}$ *denote solutions, at time* $t^n \equiv t_0 + n\Delta t$, *of the DMD and gDMD models, respectively. Let* $\mathbf{u}^n$ *denote the true solution at time* t^n, *induced by the flow map* $\mathbf{\Phi}_{\Delta t}$. *Then errors of the DMD and gDMD models at time* t^n,

$$\mathcal{E}^n_{\text{DMD}} = \|\mathbf{u}^n - \mathbf{u}^n_{\text{DMD}}\|^2 \qquad \text{and} \qquad \mathcal{E}^n_{\text{gDMD}} = \|\mathbf{u}^n - \mathbf{u}^n_{\text{gDMD}}\|^2, \tag{3.10}$$

satisfy inequalities

$$\begin{aligned}
\mathcal{E}^n_{\text{DMD}} &\le (1 + \mathrm{e}^{L\Delta t})^n \mathcal{E}^0_{\text{DMD}} + \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|_{L^\infty(\mathcal{M})} \frac{(1 + e^{L\Delta t})^n - 1}{\mathrm{e}^{L\Delta t}}, \\
\mathcal{E}^n_{\text{gDMD}} &\le (1 + \mathrm{e}^{L\Delta t})^n \mathcal{E}^0_{\text{gDMD}} + \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}_g - \mathbf{b}\|_{L^\infty(\mathcal{M})} \frac{(1 + \mathrm{e}^{L\Delta t})^n - 1}{\mathrm{e}^{L\Delta t}},
\end{aligned} \tag{3.11}$$

Proof. The proof follows similar derivations as Theorem 4.3 in [20] using triangle inequality:

$$\begin{aligned}
\mathcal{E}^n_{\text{DMD}} &= \|\mathbf{u}^{n-1} + \mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1}) - \mathbf{u}^{n-1}_{\text{DMD}} - \mathbf{B}\mathbf{u}^{n-1}_{\text{DMD}}\|^2 \\
&\le \|\mathbf{u}^{n-1} - \mathbf{u}^{n-1}_{\text{DMD}}\| + \|\mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1}) - \mathbf{B}\mathbf{u}^{n-1}_{\text{DMD}}\|^2 \\
&\le \|\mathbf{u}^{n-1} - \mathbf{u}^{n-1}_{\text{DMD}}\| + \|\mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1}_{\text{DMD}}) - \mathbf{B}\mathbf{u}^{n-1}_{\text{DMD}}\|^2 + \|\mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1}_{\text{DMD}}) - \mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1})\|^2 \\
&= \|\mathbf{u}^{n-1} - \mathbf{u}^{n-1}_{\text{DMD}}\| + \|\mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1}_{\text{DMD}}) - \mathbf{A}\mathbf{u}^{n-1}_{\text{DMD}}\|^2 + \|\mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1}_{\text{DMD}}) - \mathbf{\Phi}_{\Delta t}(\mathbf{u}^{n-1})\|^2 \\
&\le \|\mathbf{u}^{n-1} - \mathbf{u}^{n-1}_{\text{DMD}}\| + \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|^2_{L^\infty(\mathcal{M}_{\Delta t})} + \mathrm{e}^{L\Delta t}\|\mathbf{u}^{n-1}_{\text{DMD}} - \mathbf{u}^{n-1}\|^2 \\
&= (1 + \mathrm{e}^{L\Delta t})\mathcal{E}^{n-1}_{\text{DMD}} + \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|^2_{L^\infty(\mathcal{M}_{\Delta t})} \\
&\le (1 + \mathrm{e}^{L\Delta t})\mathcal{E}^{n-2}_{\text{DMD}} + \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|^2_{L^\infty(\mathcal{M}_{\Delta t})}(1 + (1 + \mathrm{e}^{L\Delta t})) \\
&\le \cdots \\
&\le (1 + \mathrm{e}^{L\Delta t})^n \mathcal{E}^0_{\text{DMD}} + \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|^2_{L^\infty(\mathcal{M}_{\Delta t})} \sum_{k=0}^{n-1} (1 + \mathrm{e}^{L\Delta t})^k
\end{aligned} \tag{3.12}$$

A proof for the error bound for $\mathcal{E}^n_{\text{gDMD}}$ is similar. □

Remark 3.3. The above error estimates indicate that gDMD has a tighter error bound than DMD a.s. because Corollary 3.1.1 indicates $\|\mathbf{\Phi}_{\Delta t} - \mathbf{A}_g - \mathbf{b}\|^2_{L^\infty(\mathcal{M}_{\Delta t})} \le \|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|^2_{L^\infty(\mathcal{M}_{\Delta t})}$ a.s.

Remark 3.4. Similarly, xDMD has a tighter error bound than rDMD a.s.

Remark 3.5. This error bound provides a general guideline for the error growth, in order to compare the DMD and gDMD models. The magnitude of the errors depends on the specific dynamics of the flow map $\mathbf{\Phi}_{\Delta t}$. Many DMD studies (e.g., [18]) have showed that the linear operator $\mathbf{A}$ is not guaranteed to be a good approximator of the general flow map $\mathbf{\Phi}_{\Delta t}$, especially when the latter is highly nonlinear. In another word, $\|\mathbf{\Phi}_{\Delta t} - \mathbf{A}\|^2_{L^\infty(\mathcal{M}_{\Delta t})}$ and, similarly, $\|\mathbf{\Phi}_{\Delta t} - \mathbf{A} - \mathbf{b}\|^2_{L^\infty(\mathcal{M}_{\Delta t})}$ can be large in the error bound estimate. A way to construct ROMs in these highly nonlinear scenarios is to approximate the so-called "Koopman operator" via mapping the state variables onto observables. The discussion is beyond the scope of this work; we refer the interested reader to [15,39].

4. Numerical experiments

We use a series of numerical experiments to demonstrate that xDMD outperforms other DMD variants and to validate our error estimates.[1] Snapshots (training data) are obtained from reference solutions during time $[0, T]$, with input-output time-lag Δt, i.e.,

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^0 & \mathbf{u}^1 & \cdots & \mathbf{u}^M \\ | & | & & | \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \cdots & \mathbf{u}^{M+1} \\ | & | & & | \end{bmatrix}, \qquad T = (M+1)\Delta t. \tag{4.1}$$

[1] Additional numerical experiments are reported in the Supplemental Material.

These datasets are assumed to be sufficiently large and rich to satisfy Lemma 2.1. We construct DMD and xDMD (and the other intermediate variants) by finding the best fit $\mathbf{A}$ or $(\mathbf{B}_g, \mathbf{b})$, which yields a set of linear approximation models for the Δt time-lag input and output. The ability to learn the unknown dynamics is tested in terms of the following characteristics.

- *Representation*: Compare the difference between $\mathbf{u}^k$ and $\mathbf{A}^k\mathbf{u}^0$, or between $\mathbf{u}^k$ and $(\mathbf{I}+\mathbf{B}_g)^k\mathbf{u}^0 + \sum_{i=0}^{k-1}(\mathbf{I}+\mathbf{B}_g)^k\mathbf{b}$, for $k = 1, \dots, M+1$. The error is essentially the least square fitting error, aka "training error" in machine learning. The training error reflects the accuracy of a trained ROM on the available M training data points in $\mathbf{Y}$ of (4.1). It is usually embedded in the target "loss function", e.g., (2.7) for DMD, which is optimized in the process of training. The trained model is then validated on another, different than $\mathbf{Y}$, dataset (i.e., test dataset); the resulting discrepancy is referred to as "test error". The test error reflects the accuracy and robustness of the trained model; small test errors indicate that the obtained ROMs reflect physics rather than simply fit the data. We use this metric to investigate our ROM's capacity for extrapolation, interpolation and generalizability.
- *Extrapolation*: Draw another set of the reference solution $\{\mathbf{u}^k\}_{k=M+1}^{2(M+1)}$ from time interval $[T, 2T]$ following the same Δt time-lag trajectory for the convenience of testing. Compare the difference between $\mathbf{u}^k$ and $\mathbf{A}^k\mathbf{u}^0$ (DMD extrapolation to $[T, 2T]$), and between $\mathbf{u}^k$ and $(\mathbf{I}+\mathbf{B}_g)^k\mathbf{u}^0 + \sum_{i=0}^{k-1}(\mathbf{I}+\mathbf{B}_g)^k\mathbf{b}$ (xDMD extrapolation to $[T, 2T]$), for $k = M+1, \dots, 2(M+1)$.
- *Interpolation*: Select a random subset of the dataset, i.e.,

$$\mathbf{X}_s = \begin{bmatrix} | & | & & | \\ \mathbf{u}^{s_0} & \mathbf{u}^{s_1} & \cdots & \mathbf{u}^{s_m} \\ | & | & & | \end{bmatrix}, \qquad \mathbf{Y}_s = \begin{bmatrix} | & | & & | \\ \mathbf{u}^{s_0+1} & \mathbf{u}^{s_1+1} & \cdots & \mathbf{u}^{s_m+1} \\ | & | & & | \end{bmatrix}, \tag{4.2}$$

  where $s_0 = 0$, $\{s_1, \dots, s_m\} \subset \{1, \cdots, M\}$, with $m < M$. Then determine $\mathbf{A}$ and $(\mathbf{B}_g, \mathbf{b})$ based on the selected dataset $\mathbf{X}_s$ and $\mathbf{Y}_s$. Compare the difference between $\mathbf{u}^k$ and $\mathbf{A}^k\mathbf{u}^0$ (DMD interpolation to $[0, T]$), and between $\mathbf{u}^k$ and $(\mathbf{I}+\mathbf{B}_g)^k\mathbf{u}^0 + \sum_{i=0}^{k-1}(\mathbf{I}+\mathbf{B}_g)^k\mathbf{b}$ (xDMD interpolation to $[0, T]$), for $k = 1, \dots, M+1$. In our examples, the selected number of snapshots, m, is smaller than $M/2$.
- *Generalizability*: Determine $\mathbf{A}$ and $(\mathbf{B}_g, \mathbf{b})$ from the datasets $\mathbf{X}$ and $\mathbf{Y}$, and obtain a linear approximation model of the discretized PDE. Compute another set of reference solutions $\{\mathbf{v}^k\}_{k=1}^{M+1}$ from a different initial input $\mathbf{v}^0 \neq \mathbf{u}^0$ and the same boundary condition and source. Compare the difference between $\mathbf{v}^k$ and $\mathbf{A}^k\mathbf{v}^0$, and between $\mathbf{v}^k$ and $(\mathbf{I}+\mathbf{B}_m)^k\mathbf{v}^0 + \sum_{i=0}^{k-1}(\mathbf{I}+\mathbf{B}_m)^k\mathbf{b}$, for $k = 1, \dots, M+1$. In our examples, the input $\mathbf{v}^0$ has completely different features than the training $\mathbf{u}^0$.
- *Accuracy*: The accuracy is compared in terms of the log relative errors,

$$\varepsilon^n_{\text{DMD}} := \lg\left(\frac{\|\mathbf{u}^n - \mathbf{u}^n_{\text{DMD}}\|_2^2}{\|\mathbf{u}^n\|_2^2}\right), \qquad \varepsilon^n_{\text{xDMD}} := \lg\left(\frac{\|\mathbf{u}^n - \mathbf{u}^n_{\text{xDMD}}\|_2^2}{\|\mathbf{u}^n\|_2^2}\right), \tag{4.3}$$

  where $\|\cdot\|_2$ denotes the L_2 norm.

All comparisons between DMD and xDMD are made using the same dataset and the same SVD truncation criteria in the pseudo-inverse part (using the default truncation criteria in Matlab).

4.1. Inhomogeneous PDEs

We start by examining the performance of the aforementioned DMD variants in learning a PDE with inhomogeneous source terms. Consider a one-dimensional diffusion equation with a source and homogeneous boundary conditions,

$$\begin{cases} \dfrac{\partial u}{\partial t} = 0.1\dfrac{\partial^2 u}{\partial x^2} + S(x), \qquad x \in (0,1), \quad t > 0; \\ u(x,0) = \exp[-20(x-0.5)^2]; \\ u_x(0,t) = 0, \quad u_x(1,t) = 0. \end{cases} \tag{4.4}$$

The reference solution is obtained by an implicit finite-difference scheme with $\Delta x = 0.01$ and $\Delta t = 0.01$. Training datasets consist of $M = 80$ snapshots collected from $t = 0$ to $t = 0.8$. The extrapolation is tested from $t = 0.8$ to $t = 1.6$. The interpolation training set consists of $m = 20$ snapshots randomly selected from the $M = 80$ snapshots.

The left column of Fig. 1 provides a comparison between the reference solution and its DMD and xDMD approximations in the three modes: representation, extrapolation and interpolation. The DMD and xDMD models have a reduced rank of 17. As predicted by the theory, DMD fails in all three modes. For a fixed time, the DMD error grows with x, which is to be expected since standard DMD algorithms are not designed to handle inhomogeneous PDEs, such as (4.4) in which the source term is $S(x) = x$. If a source term lies outside the span of the training data, as happens in this test, then it cannot be represented as a linear combination of the available snapshots. The DMD model always lies within the span of the training data, while the true solution grows out of that subspace because of the source. On the other hand, the xDMD

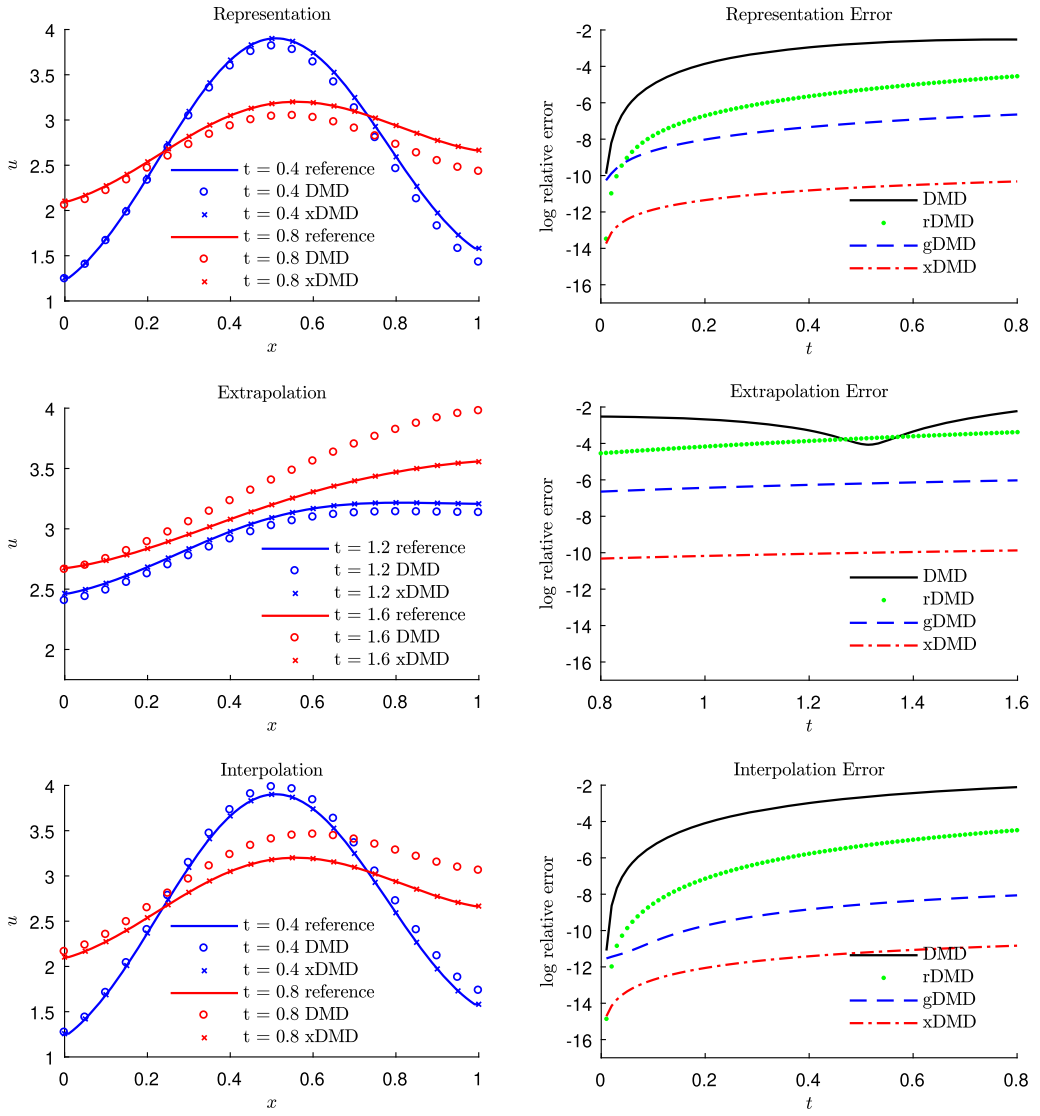

Fig. 1. Reference solution of (4.4) and its DMD (2.9) and xDMD (2.19) approximations in the representation, extrapolation, and interpolation regimes (left column). Also shown are the log relative errors of the DMD (2.9), gDMD (2.13), rDMD (2.15), and xDMD (2.19) models (right column). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

model captures the true solution in all modes thanks to the bias term that accounts for the solution expansion outside the training data span.

The right column of Fig. 1 shows the accuracy of the DMD (2.9), gDMD (2.13), rDMD (2.15), and xDMD (2.19) models. Although DMD and rDMD are mathematically equivalent, the identity subtraction in rDMD reduces the solution error in all three modes (representation, extrapolation, and interpolation). Addition of the bias term in xDMD contributes to further orders-of-magnitude reduction in the error, consistent with the theoretical proof in section 3. In all modes, the proposed xDMD outperforms the other DMD variants by several orders of magnitude, achieving almost machine accuracy.

Fig. 2 shows the performance of different ROMs in long-time extrapolation, up to $t = 100$. Exhibiting errors that grow slowly in time, the gDMD and xDMD predictions accurately capture the underlying dynamics for very long time due to the advantageous role of the bias term. That is in contrast to the DMD and rDMD models, which make poor unphysical predictions without awareness of the inhomogeneous source term.

An added benefit of gDMD and xDMD is their ability to infer a source function, $S(x)$, in an inhomogeneous PDE from temporal snapshots of the solution (Fig. 3). Both methods recover $S(x)$, regardless of whether it is linear ($S = x$) or nonlinear ($S = e^x$), and have comparable errors. While DMD lumps together the differential operator and the source, gDMD and xDMD treat them separately. This endows them with the ability to learn both the operator (the system itself) and the source

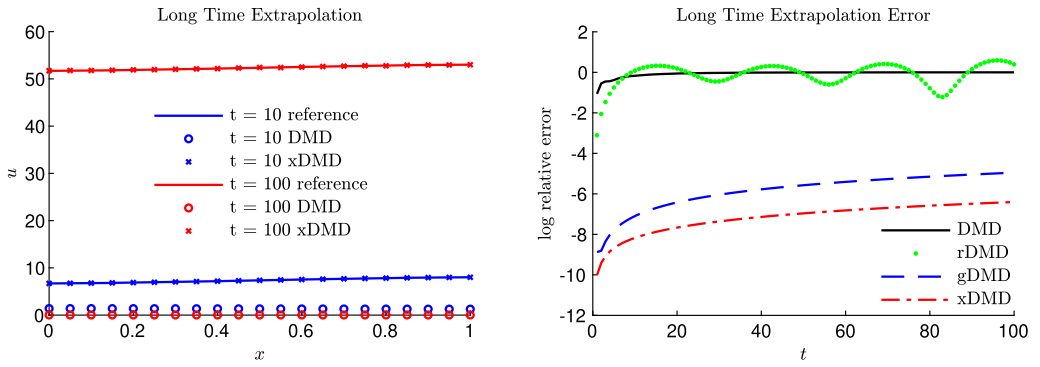

Fig. 2. Reference solution of (4.4) and its DMD (2.9) and xDMD (2.19) approximations (left); and the log relative error of the DMD (2.9), gDMD (2.13), rDMD (2.15), and xDMD (2.19) models (right) for long time extrapolation.

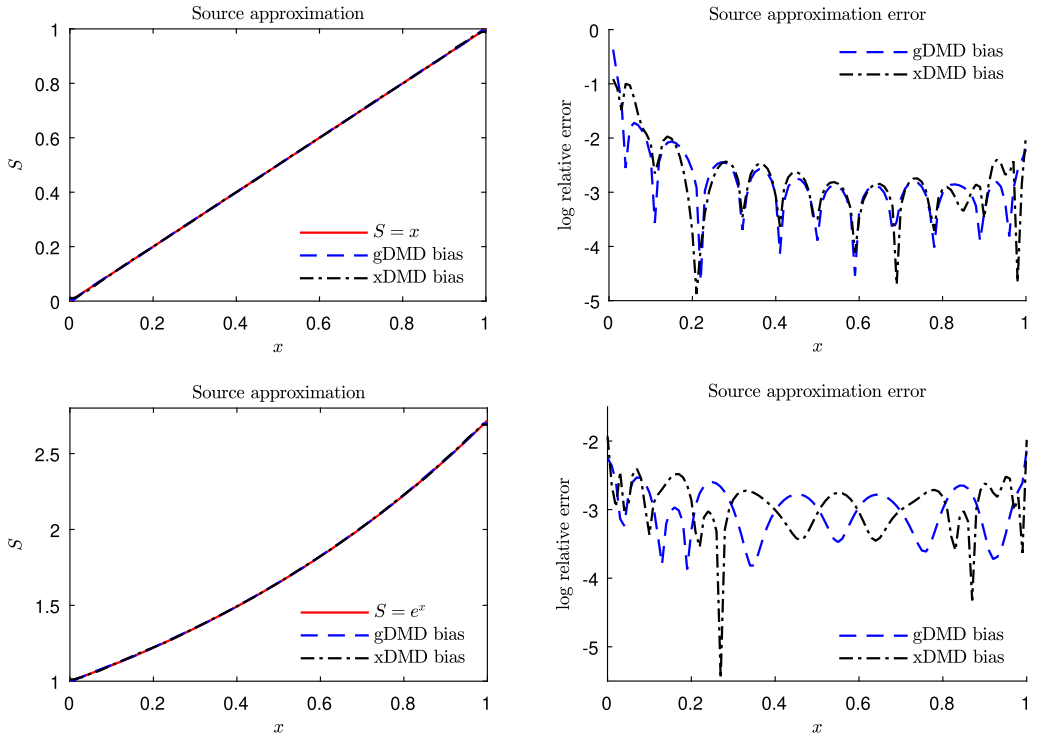

Fig. 3. Estimation of the source term $S(x) = x$ and e^x in (4.4) by gDMD and xDMD: the eyeball measure (left) and the log relative error (right).

(external forces acting on the system), as long as the latter does not vary with time. This self-learning feature carries almost no extra computational cost.

4.2. Inhomogeneous boundary conditions and data errors

Next, we examine the ability of DMD and xDMD to handle inhomogeneous boundary conditions and data errors. Consider a two-dimensional diffusion equation in a multi-connected domain $\mathcal{D}$ with inhomogeneous boundary conditions,

$$\begin{cases} \dfrac{\partial u}{\partial t} = \nabla^2 u, \qquad (x, y) \in \mathcal{D}, \quad t \in (0, 10000]; \\ u(x, y, 0) = 0; \\ u(0, y, t) = 3, \quad u(800, y, t) = 1, \quad \dfrac{\partial u}{\partial y}(x, 0, t) = \dfrac{\partial u}{\partial y}(x, 800, t) = 0, \quad u(x, y, t) = 2 \ \text{ on } \partial\mathcal{S}\,(\text{red}). \end{cases} \tag{4.5}$$

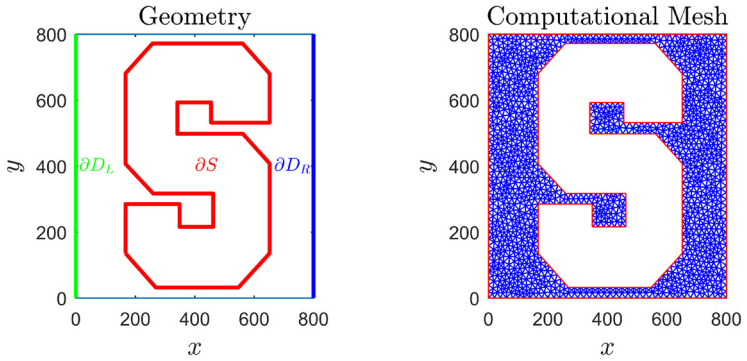

Fig. 4. Multi-connected simulation domain $\mathcal{D}$ (left) and the mesh used in the finite-element solution of (4.5).

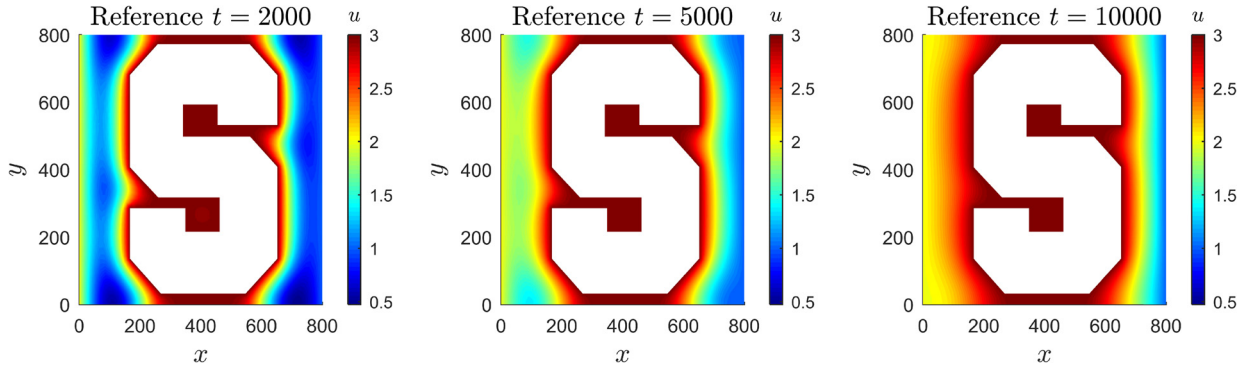

Fig. 5. The reference solution of (4.5), $u(x, y, t)$, at times $t = 2000$, $t = 5000$ and $t = 10000$.

The domain $\mathcal{D}$ is the 800×800 square with an S-shaped cavity (Fig. 4). The Dirichlet boundary conditions are imposed on the left and right sides of the square and the cavity surface. The top and bottom of the square are impermeable. The reference solution is obtained via Matlab PDE toolbox on the finite-element mesh with 1633 elements shown in Fig. 4. The solution from early transient time ($t = 2000$) until steady state ($t = 10000$) is presented in Fig. 5.

With the total simulation time (time sufficient to reach steady state) $t = 10000$, we generate snapshots spaced by $\Delta t = 5$ and use those to conduct four tests. First, the leading $M = 1200$ snapshots are used to inform DMD and xDMD and to ascertain their representation errors. Second, the DMD and xDMD models are deployed to extrapolate until $t = 10000$ and compare the extrapolation error of the two models. Third, randomly selected $M = 600$ snapshots from the first 1200 snapshots are used for interpolation and to compare the interpolation error of DMD and xDMD. Finally, we repeat these representation/extrapolation/interpolation tests on data corrupted by addition of zero-mean white noise whose strengths at any (x, t) is 0.1% of the nominal value of $u(x, t)$ at that point.

Fig. 6 reveals that, for noiseless data, the accuracy of xDMD is orders of magnitude higher than that of DMD in the representation and interpolation modes with the same reduced rank of 30; in the extrapolation mode, the error is dominated by the extrapolation error, which increases with time, but xDMD is still about 9% more accurate than DMD at later times. DMD has a good performance in this case because the inhomogeneity from the boundary conditions happens to lie inside the span of the training data (i.e., can be approximated by a linear combination of the available snapshots), which is not guaranteed for all inhomogeneous boundary conditions (see a counterexample of Figure S1 in Supplemental Material). However, in the presence of measurement noise, xDMD has no better performance than DMD; it is even less accurate in the extrapolation and interpolation regimes. This sensitivity to noise mirrors the over-fitting issue in machine learning: models with more parameters fit the limited number of available data (solution snapshots) too closely and, consequently, fail to fit additional data or to reliably predict future observations. Since xDMD has more parameters than DMD due to the bias term, one should expect the former to be more sensitive to noise than the latter.

4.3. Coupled nonlinear PDEs

Common sense suggests that the success of linear models, such as DMD and xDMD, to approximate nonlinear dynamics is not guaranteed. In machine learning, data augmentation by feature map is widely used to deal with the nonlinearity. Similarly, judiciously chosen observables play a crucial role in the success of data-driven (DMD) modeling [6,19,45]. The selection of observables requires prior knowledge of the underlying process, which is out of scope of this study. Instead, we assume no prior knowledge and apply no data augmentation, i.e., our observables are the state itself. To satisfy the

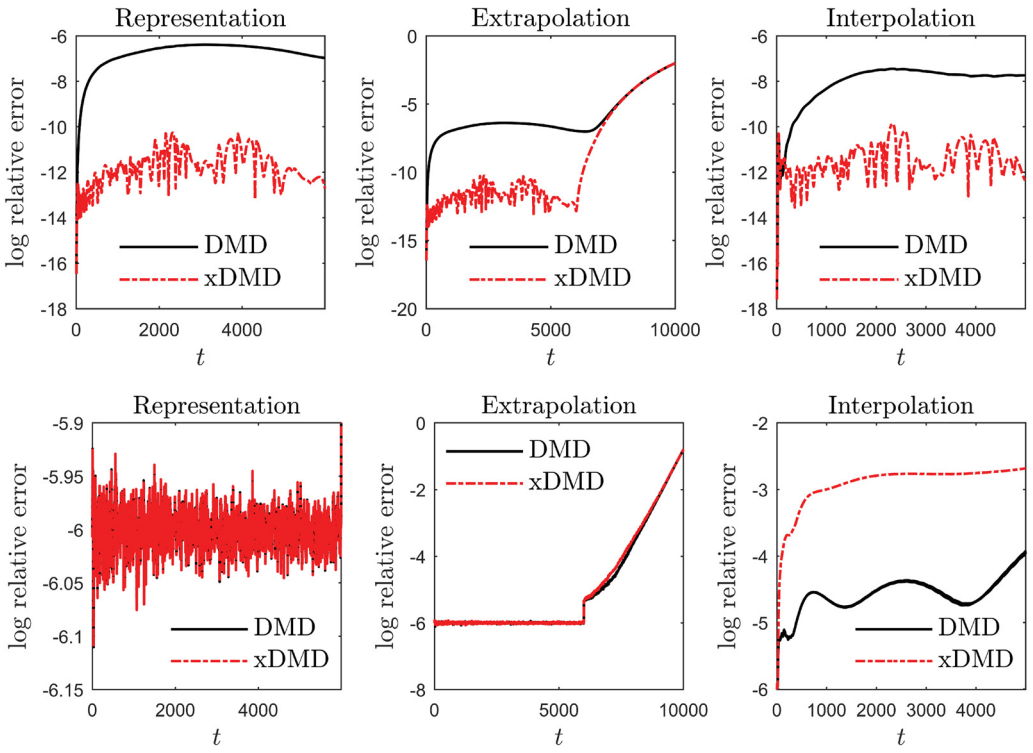

Fig. 6. Dependence of the log relative error of the DMD and xDMD models on time in the representation, extrapolation and interpolation modes. These errors are reported for noiseless data (top row) and data corrupted by addition of zero-mean white noise whose strengths at any (x, t) is 0.1% of the nominal value of $u(x, t)$ at that point (bottom row).

assumptions in Lemma 2.1, we restrict our attention to nonlinear PDEs whose solutions are confined in certain subspace $\mathcal{M}$. Our numerical experiments deal with the two-dimensional viscous Burgers' equation (reported in the Supplemental Material) and the two-dimensional Navier-Stokes equations. The goal of these tests is to assess the ability of DMD and xDMD to learn complex flow maps.

We consider two-dimensional flow of an incompressible fluid with density $\rho = 1$ and dynamic viscosity $\nu = 1/600$ (these and other quantities are reported in consistent units) around an impermeable circle of diameter $D = 0.1$. The flow, which takes place inside a rectangular domain $\mathcal{D} = \{\mathbf{x} = (x, y)^\top : (x, y) \in [0, 2] \times [0, 1]\}$, is driven by an externally imposed pressure gradient; the center of the circular inclusion is $\mathbf{x}_{\text{circ}} = (0.3, 0.5)^\top$. Dynamics of the three state variables, flow velocity $\mathbf{u}(\mathbf{x}, t) = (u, v)^\top$ and fluid pressure $p(\mathbf{x}, t)$, is described by the two-dimensional Navier-Stokes equations,

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = 0; \\ \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} = -\dfrac{1}{\rho}\dfrac{\partial p}{\partial x} + \nu\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right), \qquad \mathbf{x} \in \mathcal{D}, \quad t > 0; \\ \dfrac{\partial v}{\partial t} + u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} = -\dfrac{1}{\rho}\dfrac{\partial p}{\partial y} + \nu\left(\dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2}\right); \end{cases} \tag{4.6}$$

subject to initial conditions $\mathbf{u}(x, y, 0) = (0, 0)^\top$ and $p(x, y, 0) = 0$; and boundary conditions

$$p(2, y, t) = 0, \quad \frac{\partial p}{\partial \mathbf{n}}|_{\partial\mathcal{D}\setminus\{x=2\}} = 0, \quad \mathbf{u}(0, y, t) = (1, 0)^\top, \quad \frac{\partial \mathbf{u}(2, y, t)}{\partial \mathbf{n}} = 0, \quad \mathbf{u}(x, 0, t) = \mathbf{u}(x, 1, t) = \mathbf{0}.$$

Here, $\mathbf{n}$ denotes the unit normal vector. This combination of parameters results in the Reynolds number $\text{Re} = 60$.

The reference solution is obtained with the Matlab code [46], which implements a finite-difference scheme on the staggered grid with $\Delta x = \Delta y = 0.02$ and $\Delta t = 0.0015$. Our observable (quantity of interest) is the magnitude of the flow velocity, $U(x, y, t) = \sqrt{u^2 + v^2}$. Visual examination of the solution $U(x, y, t)$ reveals it to be periodic from $t = 7.5$ to $t = 15$ (the simulation horizon), i.e., the solution is confined in a fixed subspace $\mathcal{M}$. We collect $M = 2500$ snapshots of U from $t = 7.5$ to $t = 11.25$ into a training dataset, from which DMD and xDMD learn the nonlinear dynamics. The discrepancy between the reference solution and its fitting with the DMD and xDMD models is the representation error.

The first row of Fig. 7 depicts the spatial distribution of the flow speed U, at times $t = 9.38$ and $t = 11.25$, computed with the (reference) solution of the Navier-Stokes equations (4.6). The DMD and xDMD models have a reduced rank of 75.

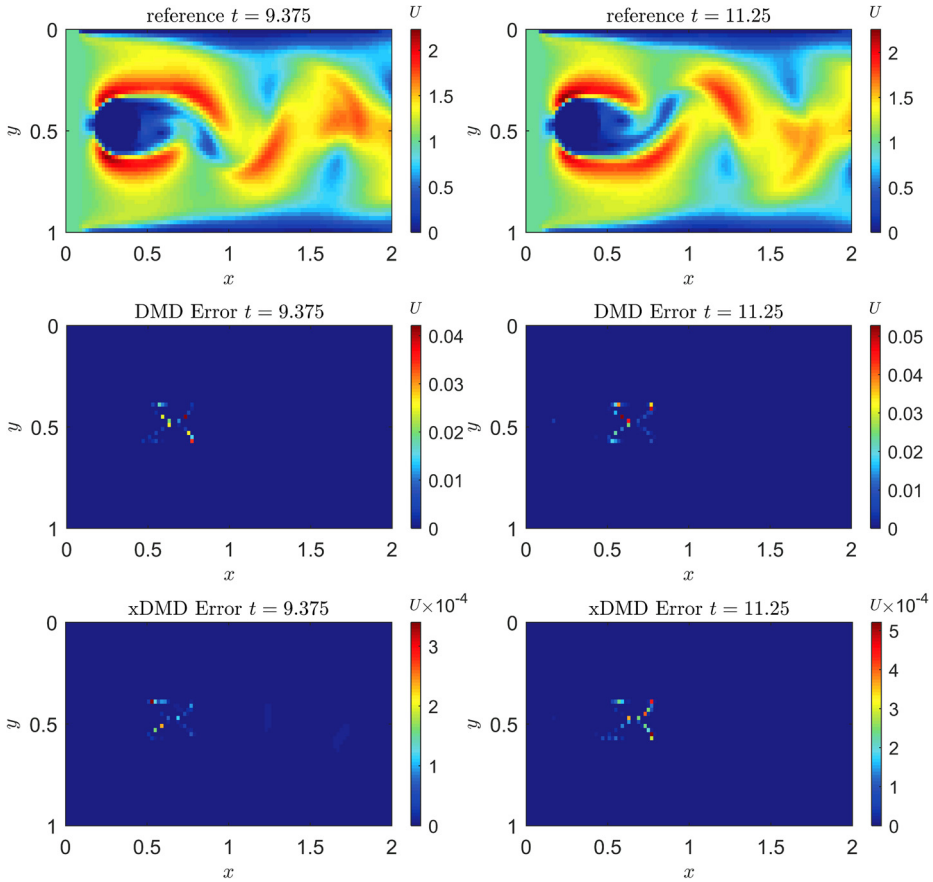

Fig. 7. Velocity magnitude $U = \sqrt{u^2 + v^2}$ of incompressible flow with Reynolds number $\mathrm{Re} = 60$ around an impermeable circle, predicted by solving numerically the two-dimensional Navier-Stokes equations (4.6) (top row) and by using the DMD and xDMD models (middle and bottom rows, respectively) in the representation mode. The representation errors (4.3) for these two approximations are displayed in the second and third rows, respectively.

Both DMD and xDMD fit the nonlinear flow data using a linear approximation with satisfactory accuracy (the last two rows of Fig. 7). The errors are confined to the circle's wake, with xDMD being two orders of magnitude more accurate than DMD.

Next, we use the learned DMD and xDMD models in the extrapolation mode, i.e., to predict $U(x, y, t)$ within the time interval from $t = 11.25$ to $t = 15$. As shown in Fig. 8, both DMD and xDMD yield accurate extrapolation, which should be expected due to the periodic behavior of the solution. Although the accuracy in extrapolation is diminished for both methods, xDMD remains more accurate than DMD at different extrapolation times.

Finally, Fig. 9 exhibits the log relative error of the two methods as function of time. In the representation mode, both DMD and xDMD have nearly steady small fitting error, fluctuating about 10^{-10} for xDMD and 10^{-6} for DMD. The observation of xDMD's higher accuracy in fitting the data is consistent with Theorem 3.1. Similarly, the extrapolation error of DMD and xDMD validates Theorem 3.2. Although both extrapolation errors grow slowly, the xDMD error exhibits a periodic pattern (consistent with the periodic pattern of the solution U), indicating that the xDMD linear model is able to capture the detailed periodic feature of the true flow better. Once an accurate linear representation of the nonlinear flow is available, one can conduct spatiotemporal mode analysis, reduced-order modeling and accelerated simulations. Table 1 collates computational times of simulating the reference solution and the linear approximation models. Further reduction in computation cost can be achieved by constructing reduced-order models using eigen-decomposition in DMD and xDMD.

4.4. Generalizability to new inputs

Generalizability refers to a model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model. With validated generalizability, a DMD or xDMD model can be employed as a surrogate to accelerate, e.g., expensive Markov Chain Monte Carlo (MCMC) sampling used in inverse problems. A typical setting for this type of problems is solute transport in groundwater flow, whose steady-state Darcy velocity (flux) $\mathbf{q}(\mathbf{x}) = -K\nabla h$ is computed from the groundwater flow equation

$$\nabla \cdot (K \nabla h) = 0. \tag{4.7}$$

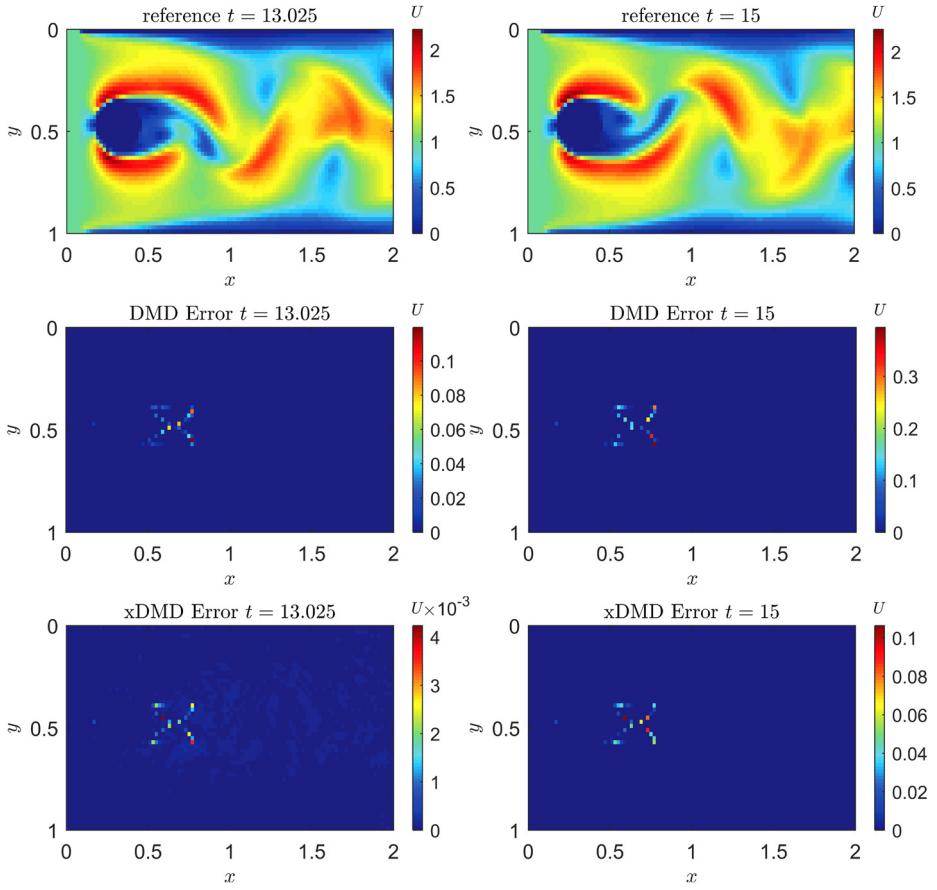

Fig. 8. Velocity magnitude $U = \sqrt{u^2 + v^2}$ of incompressible flow with Reynolds number $\mathrm{Re} = 60$ around an impermeable circle, predicted by solving numerically the two-dimensional Navier-Stokes equations (4.6) (top row) and by using the DMD and xDMD models (middle and bottom rows, respectively) in the extrapolation mode. The extrapolation errors (4.3) for these two approximations are displayed in the second and third rows, respectively.

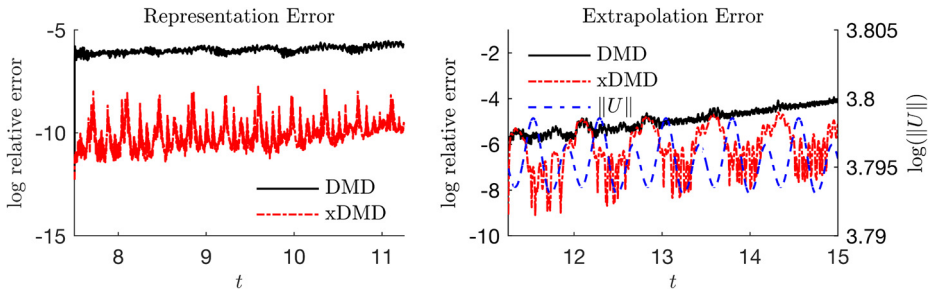

Fig. 9. Temporal evolution of log relative error of the DMD and xDMD models in the representation and extrapolation modes.

Table 1
Computational time and relative error for the reference solution and the DMD and xDMD models.

	Simulation	DMD	xDMD
Computational time (sec)	29.0776	2.1352	2.1654
Relative error	–	2.0515×10^{-5}	3.1193×10^{-6}

Here, $h(\mathbf{x})$ is the hydraulic head, and $K(\mathbf{x})$ is the hydraulic conductivity of a heterogeneous subsurface environment; in our simulations we use a rectangular simulation domain $\mathcal{D} = \{\mathbf{x} = (x, y)^\top : (x, y) \in [0, 128] \times [0, 64]\}$ and the $K(\mathbf{x})$ field in Fig. 10 (these and other quantities are expressed in consistent units). The boundary conditions are $h(x = 0, y) = 1$, $h(x = 128, y) = 0$, and impermeable on $y = 0$ and $y = 64$.

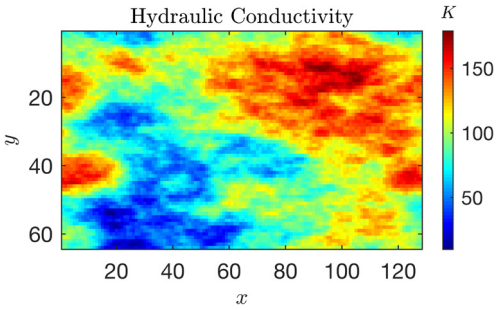

Fig. 10. Spatial distribution of hydraulic conductivity $K(\mathbf{x})$ used in our simulations.

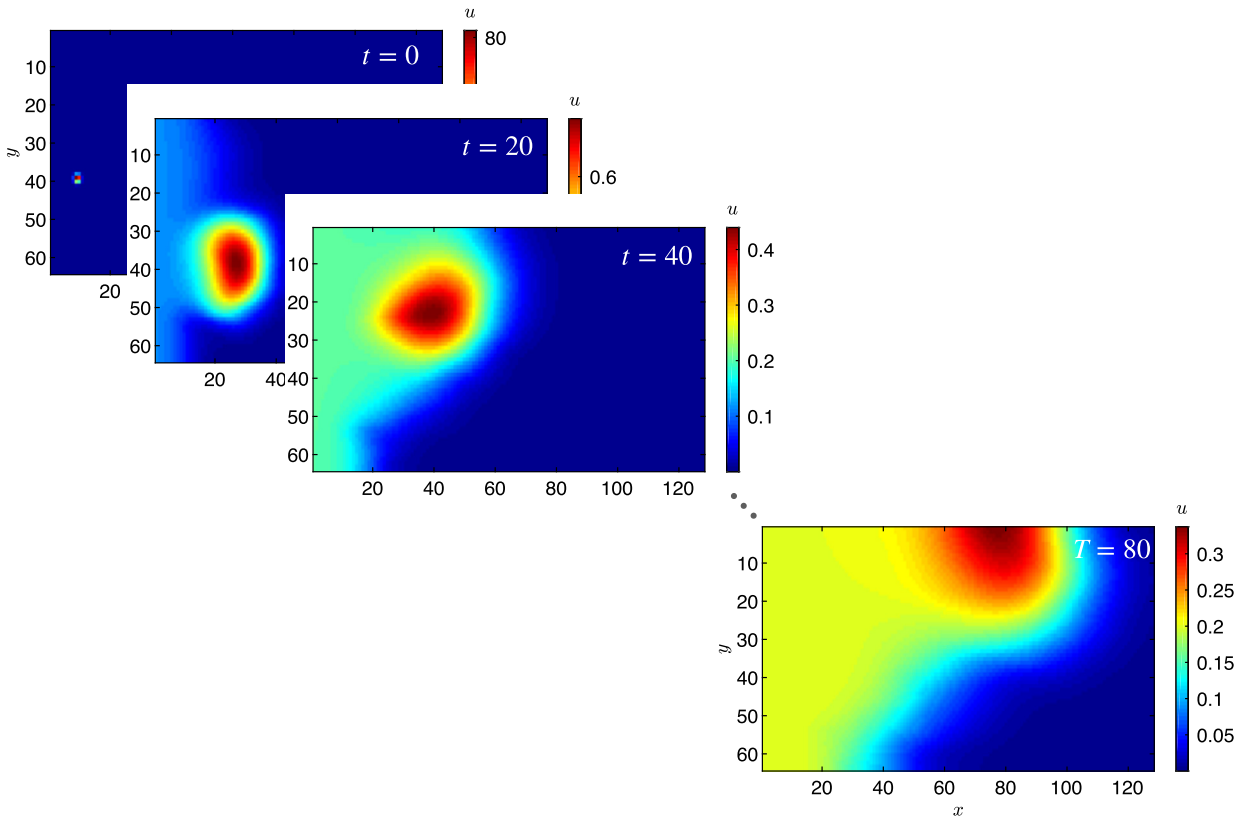

Fig. 11. Representative snapshots of solute concentration, $u(\mathbf{x}, t)$, in the training dataset with initial condition (4.9).

The resulting macroscopic velocity $\mathbf{v}(\mathbf{x}) = \mathbf{q}/\omega$, with ω denoting the porosity, is then used in the advection-dispersion equation to predict the contaminant concentration $u(x, y, t)$:

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = \nabla \cdot (\mathbf{D} \nabla u), \qquad \mathbf{x} \in \mathcal{D}, \quad t \in (0, T], \tag{4.8}$$

with $T = 80$. In general, the dispersion coefficient $\mathbf{D}$ is a second-rank semi-positive definite tensor, whose components depend on the magnitude of the flow velocity, $|\mathbf{u}|$. Here, for illustrative purposes, we treat it as the identity matrix, $\mathbf{D} = \mathbf{I}$. The boundary conditions for (4.8) are $u(0, y, t) = 0.2$ and $\partial_x u(128, y, t) = \partial_y u(x, 0, t) = \partial_y u(x, 64, t) = 0$. The training is done for the initial condition $u(x, y, 0) = u_{\text{in}}(x, y)$ with

$$u_{\text{in}} = s \exp[-(x - x_s)^2 - (y - y_s)^2], \tag{4.9}$$

where $s = 100$ and the coordinates of the plume's center of mass, (x_s, y_s) are treated as independent random variables with uniform distributions, $x_s \sim \mathcal{U}[0, 25]$ and $y_s \sim \mathcal{U}[0, 64]$. An example of the training dynamics is presented in Fig. 11. We generate $N_{\text{MC}} = 2000$ realizations of the pairs (x_s, y_s) and evaluate the corresponding initial conditions $u_{\text{in}}^{(n)}(\mathbf{x})$ for

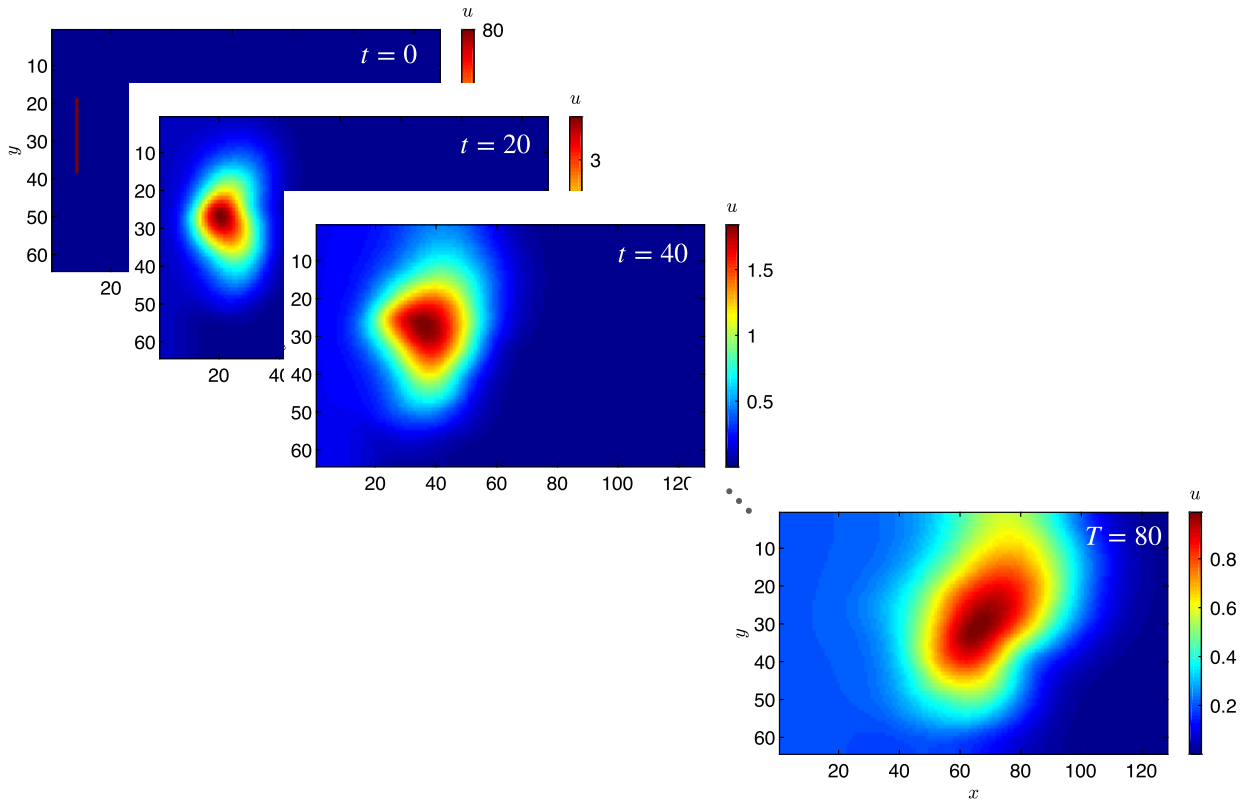

Fig. 12. Representative snapshots of solute concentration, $u(\mathbf{x}, t)$, in the test case with initial condition (4.11).

$n = 1, \dots, N_{\mathrm{MC}}$. For each of these realizations, (4.8) is solved[2] to compute our quantity of interest, the concentration field $u_T^{(n)}(\mathbf{x}) \equiv u^{(n)}(\mathbf{x}, T)$. The matrix pairs $\{u_{\mathrm{in}}^{(n)}, u_T^{(n)}\}_{n=1}^{N_{\mathrm{MC}}}$ are arranged into vectors

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^{N_{\mathrm{MC}}} \\ | & | & & | \end{bmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}^1 & \mathbf{y}^2 & \cdots & \mathbf{y}^{N_{\mathrm{MC}}} \\ | & | & & | \end{bmatrix} \tag{4.10}$$

where $\mathbf{x}^n$ is vectorized $u_{\mathrm{in}}^{(n)}$ and $\mathbf{y}^n$ is vectorized $u_T^{(n)}$. Finally, the DMD and xDMD models are deployed to learn the flow map $\mathbf{\Phi}_{\Delta t}$ with the time lag $\Delta t = T$.

Our goal here is to test the ability of these models to predict $u(\mathbf{x}, T)$ for other initial conditions, such as the line source

$$u_{\mathrm{in}} = \begin{cases} 80 & x = 10,\ y \in [20, 40], \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Fig. 12 exhibits the reference dynamics generated from this line source. The quantity of interest is the solute concentration map at $T = 80$.

In Fig. 13, we compare the ability of DMD and xDMD to predict a quantity of interest, i.e., $u(\mathbf{x}, T)$, for an initial condition that is qualitatively different from that for which they were trained.[3] The DMD and xDMD models have the reduced rank of 1648. While xDMD performs well in this generalizability test, DMD yields a wrong output concentration map because of its failure to handle inhomogeneity. The prediction error is largest in the vicinity of the left boundary, along which the inhomogeneous Dirichlet boundary condition is prescribed.

Fig. 14 demonstrates the DMD and xDMD performance for the same task as before but when noisy data are used for training. The training data are corrupted by addition of zero-mean white noise whose strength is 0.1% of the nominal value. Although the predictions from both models are disturbed by the white noise, xDMD still captures the features of the concentration map. In contrast to Fig. 6, the correction effects from the bias term in xDMD are more dominant than

[2] The reference solutions are obtained with the groundwater flow simulator MODFLOW and the solute transport simulator MT3DMS, both ran on a uniform mesh $\Delta x = \Delta y = 1$.

[3] The results for the initial condition given by a linear combination of two Gaussians are presented in Supplemental Material.

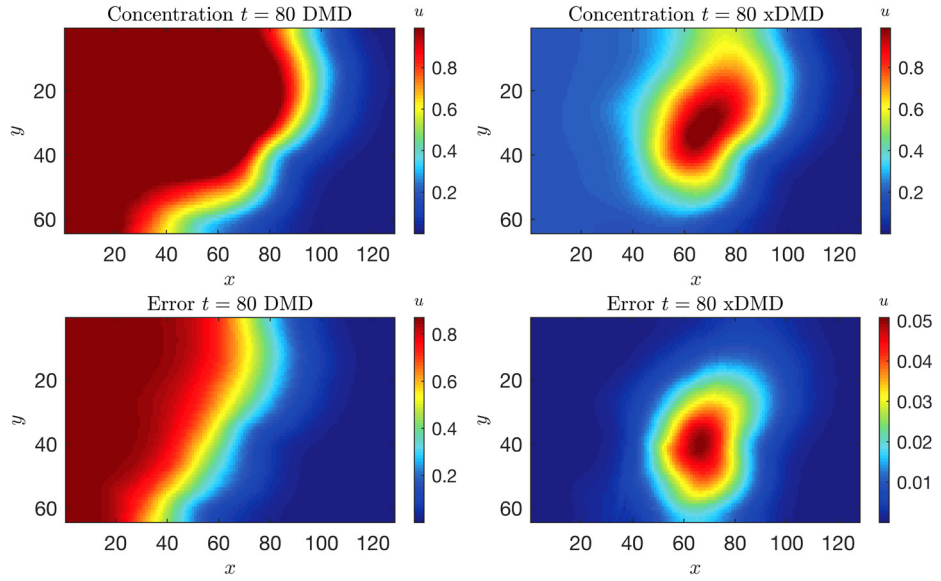

Fig. 13. Solute concentration predicted with the DMD and xDMD models for the initial condition not seen during training. Also shown are the absolute errors of DMD and xDMD.

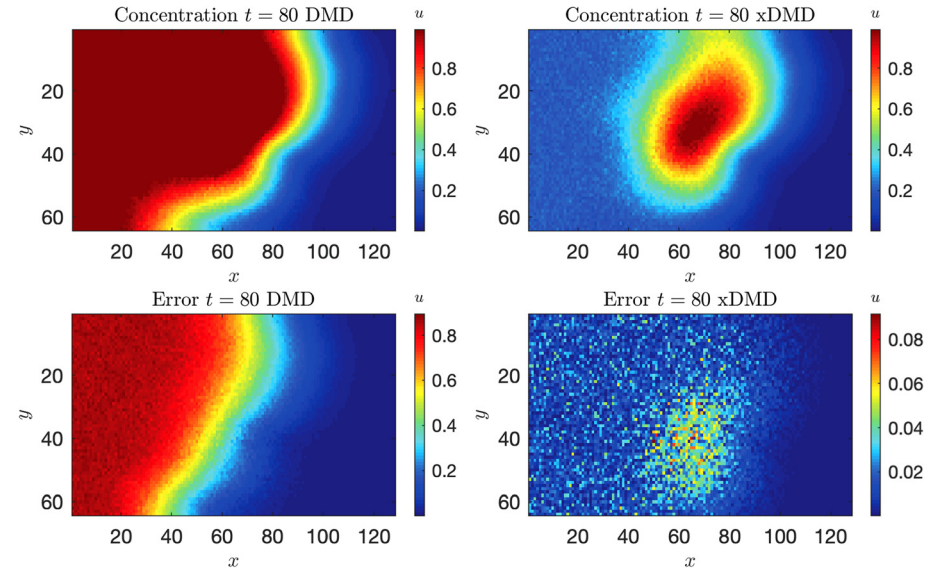

Fig. 14. Solute concentration predicted with the DMD and xDMD models using 0.1% noisy data for the initial condition not seen during training. Also shown are the absolute errors of DMD and xDMD.

over-fitting the noise. The prediction error of DMD is still largest in the vicinity of the left boundary due to its failure to handle the inhomogeneous Dirichlet boundary condition.

5. Conclusions and future work

We presented an extended DMD (xDMD) framework for representation of (linear or nonlinear) inhomogeneous PDEs. Our xDMD borrows from residual learning and bias identification ideas, which originated in the deep neural networks community. It shows high accuracy in learning the underlying dynamics, especially in inhomogeneous systems for which standard DMD fails. The inhomogeneous source can be accurately learned from the bias term at no extra computational cost. We conducted a number of numerical experiments to demonstrate that xDMD is an effective data-driven modeling tool and offers better accuracy than the standard DMD.

Although xDMD provides an optimal linear approximation of the unknown dynamics, data-driven modeling for highly nonlinear PDE in general remains a challenging task. Judiciously chosen observables are needed in order to approximate

the corresponding Koopman operator, which requires either prior knowledge about the dynamics or dictionary learning. Developments and experiences from deep learning may again bring potential solutions and vice versa.

In the follow-up work, we plan to use xDMD to construct surrogates, e.g., for Markov Chain Monte Carlo solutions of inverse problems and for uncertainty quantifications. The verified generalizability will allow us to replace the expensive simulation with xDMD surrogates in each Monte Carlo run. Further model reduction can be carried out to improve efficiency as well.

CRediT authorship contribution statement

Hannah Lu: Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft. **Daniel M. Tartakovsky:** Conceptualization, Funding acquisition, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

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

Acknowledgements

This work was supported in part by Air Force Office of Scientific Research under award number FA9550-18-1-0474 and by the gift from Total.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcp.2021.110550.

References

[1] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (5923) (2009) 81–85.
[2] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA 113 (15) (2016) 3932–3937.
[3] H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proc. R. Soc. A 473 (2197) (2017) 20160446.
[4] J. Bakarji, D.M. Tartakovsky, Data-driven discovery of coarse-grained equations, J. Comput. Phys. 434 (2021) 110219, https://doi.org/10.1016/j.jcp.2021.110219.
[5] M.O. Williams, I.G. Kevrekidis, C.W. Rowley, A data–driven approximation of the Koopman operator: extending dynamic mode decomposition, J. Nonlinear Sci. 25 (6) (2015) 1307–1346.
[6] M. Korda, I. Mezić, On convergence of extended dynamic mode decomposition to the Koopman operator, J. Nonlinear Sci. 28 (2) (2018) 687–710.
[7] Q. Li, F. Dietrich, E.M. Bollt, I.G. Kevrekidis, Extended dynamic mode decomposition with dictionary learning: a data-driven adaptive spectral decomposition of the Koopman operator, Chaos 27 (10) (2017) 103111.
[8] H. Schaeffer, S.G. McCalla, Sparse model selection via integral terms, Phys. Rev. E 96 (2) (2017) 023302.
[9] G. Tran, R. Ward, Exact recovery of chaotic systems from highly corrupted data, Multiscale Model. Simul. 15 (3) (2017) 1108–1129.
[10] H. Schaeffer, G. Tran, R. Ward, Extracting sparse high-dimensional dynamics from limited data, SIAM J. Appl. Math. 78 (6) (2018) 3279–3295.
[11] A.M. Tartakovsky, C.O. Marrero, P. Perdikaris, G.D. Tartakovsky, D. Barajas-Solano, Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems, Water Resour. Res. (2020), https://doi.org/10.1029/2019WR026731.
[12] N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, J. Comput. Phys. 403 (2020) 109056.
[13] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28.
[14] J.H. Tu, C.W. Rowley, D.M. Luchtenburg, S.L. Brunton, J.N. Kutz, On dynamic mode decomposition: theory and applications, J. Comput. Dyn. 1 (2) (2014).
[15] B.O. Koopman, Hamiltonian systems and transformation in Hilbert space, Proc. Natl. Acad. Sci. USA 17 (5) (1931) 315.
[16] S.L. Brunton, B.W. Brunton, J.L. Proctor, E. Kaiser, J.N. Kutz, Chaos as an intermittently forced linear system, Nat. Commun. 8 (1) (2017) 1–9.
[17] J.N. Kutz, J.L. Proctor, S.L. Brunton, Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems, Complexity (2018) 2018.
[18] H. Lu, D.M. Tartakovsky, Prediction accuracy of dynamic mode decomposition, SIAM J. Sci. Comput. 42 (3) (2020) A1639–A1662.
[19] H. Lu, D.M. Tartakovsky, Lagrangian dynamic mode decomposition for construction of reduced-order models of advection-dominated phenomena, J. Comput. Phys. (2020) 109229.
[20] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[21] S.H. Rudy, J.N. Kutz, S.L. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, J. Comput. Phys. 396 (2019) 483–506.
[22] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[23] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. (2020) 109307.
[24] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.
[25] S. Pawar, S. Rahman, H. Vaddireddy, O. San, A. Rasheed, P. Vedula, A deep learning enabler for nonintrusive reduced order modeling of fluid flows, Phys. Fluids 31 (8) (2019) 085101.
[26] S. Mo, N. Zabaras, X. Shi, J. Wu, Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification, Water Resour. Res. 55 (5) (2019) 3856–3881.
[27] Y. Liu, W. Sun, L.J. Durlofsky, A deep-learning-based geological parameterization for history matching complex models, Math. Geosci. 51 (6) (2019) 725–766.

[28] S. Chan, A.H. Elsheikh, A machine learning approach for efficient uncertainty quantification using multiscale methods, J. Comput. Phys. 354 (2018) 493–511.
[29] S. Karumuri, R. Tripathy, I. Bilionis, J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, J. Comput. Phys. 404 (2020) 109120.
[30] R.K. Tripathy, I. Bilionis, Deep UQ: learning deep neural network surrogate models for high dimensional uncertainty quantification, J. Comput. Phys. 375 (2018) 565–588.
[31] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[32] J.L. Proctor, S.L. Brunton, J.N. Kutz, Dynamic mode decomposition with control, SIAM J. Appl. Dyn. Syst. 15 (1) (2016) 142–161.
[33] J.H. Tu, C.W. Rowley, D.M. Luchtenburg, S.L. Brunton, J.N. Kutz, On dynamic mode decomposition: theory and applications, arXiv preprint, arXiv: 1312.0041, 2013.
[34] S. Le Clainche, J.M. Vega, Higher order dynamic mode decomposition, SIAM J. Appl. Dyn. Syst. 16 (2) (2017) 882–925.
[35] P. Héas, C. Herzet, Low-rank dynamic mode decomposition: optimal solution in polynomial-time, arXiv preprint, arXiv:1610.02962, 2016.
[36] Z. Chen, D. Xiu, On generalized residue network for deep learning of unknown dynamical systems, arXiv preprint, arXiv:2002.02528, 2020.
[37] M. Hardt, T. Ma, Identity matters in deep learning, arXiv preprint, arXiv:1611.04231, 2018.
[38] A. Stuart, A.R. Humphries, Dynamical Systems and Numerical Analysis, vol. 2, Cambridge University Press, 1998.
[39] J.N. Kutz, S.L. Brunton, B.W. Brunton, J.L. Proctor, Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, SIAM, Philadelphia, PA, 2016.
[40] C.W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, D.S. Henningson, Spectral analysis of nonlinear flows, J. Fluid Mech. 641 (2009) 115–127.
[41] C.W. Rowley, S.T. Dawson, Model reduction for flow analysis and control, Annu. Rev. Fluid Mech. 49 (2017) 387–417.
[42] M. Korda, I. Mezić, Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control, Automatica 93 (2018) 149–160.
[43] J.L. Proctor, S.L. Brunton, J.N. Kutz, Generalizing Koopman theory to allow for inputs and control, SIAM J. Appl. Dyn. Syst. 17 (1) (2018) 909–930.
[44] S. Peitz, S.E. Otto, C.W. Rowley, Data-driven model predictive control using interpolated Koopman generators, SIAM J. Appl. Dyn. Syst. 19 (3) (2020) 2162–2193.
[45] I. Mezić, Analysis of fluid flows via spectral properties of the Koopman operator, Annu. Rev. Fluid Mech. 45 (2013) 357–378.
[46] J. Johns, A Matlab code for numerical solution of Navier-Stokes equations for two-dimensional incompressible flow (velocity-pressure formulation) along with ability for importing custom scenarios for the fluid flow, https://github.com/JamieMJohns/Navier-stokes-2D-numerical-solve-incompressible-flow-with-custom-scenarios-MATLAB-, 2018.

SUPPLEMENTAL MATERIAL

We provide a few additional tests used to demonstrate the relative performance of DMD and xDMD.

Boundary Conditions and Noisy Data

We study the non-homogeneity driven by boundary conditions. Consider a one-dimensional diffusion equation,

$$\partial_t u = D\partial_{xx} u, \qquad x \in [0,1], \qquad t > 0, \tag{S1}$$

with $D = 0.1$. Three different cases are tested to compare DMD and gDMD:

- Case 1: Dirichlet boundary conditions,

$$\begin{cases} u(x,0) = 1, \\ u(0,t) = 3, \qquad u(1,t) = 2. \end{cases} \tag{S2}$$

- Case 2: Neumann boundary conditions,

$$\begin{cases} u(x,0) = \mathrm{e}^{-20(x-0.5)^2}, \\ u_x(0,t) = 0, \qquad u_x(1,t) = 0. \end{cases} \tag{S3}$$

- Case 3: Contaminant training data. The initial and boundary conditions are the same as in Case 1. The training data are solution to (S1) with 0.1% measurement noise.

The same spatiotemporal discretization as in Test 4.1 is used, with the same number of training data. The solution behavior is trivial and, thus, not shown here. The relative accuracy of DMD and xDMD is compared in Figure S1 in terms of representation, extrapolation and interpolation.

In Case 1, xDMD exhibits the higher-order accuracy than DMD in all three regimes of representation, extrapolation and interpolation. DMD captures the overall solution behavior because the diffusion effect dominates the dynamics in comparison with the non-homogeneity driven by the boundaries. The DMD error is mostly distributed near the two boundaries, and this error accumulates with time. On the other hand, xDMD has a flat error distribution in the physical domain with the much smaller error magnitude.

In Case 2, which is a homogeneous case, the accuracy of xDMD than that of DMD. The improvements are mostly due to the modification in rDMD, but also indicate that no sacrifice of accuracy is made by adding the bias. This test guarantees better performance of xDMD without knowledge of homogeneity.

In Case 3, both DMD and xDMD lose several orders of accuracy and behave almost the same in the presence of noise. In the interpolation test, xDMD is even less accurate than DMD. This behavior is reminiscent of the over-fitting issue in machine learning: models with more parameters fit too closely to

the limited number of contaminant data and therefore fail to fit additional data or reliably predict future observations. Obviously, xDMD has more parameters than DMD due to the bias term.

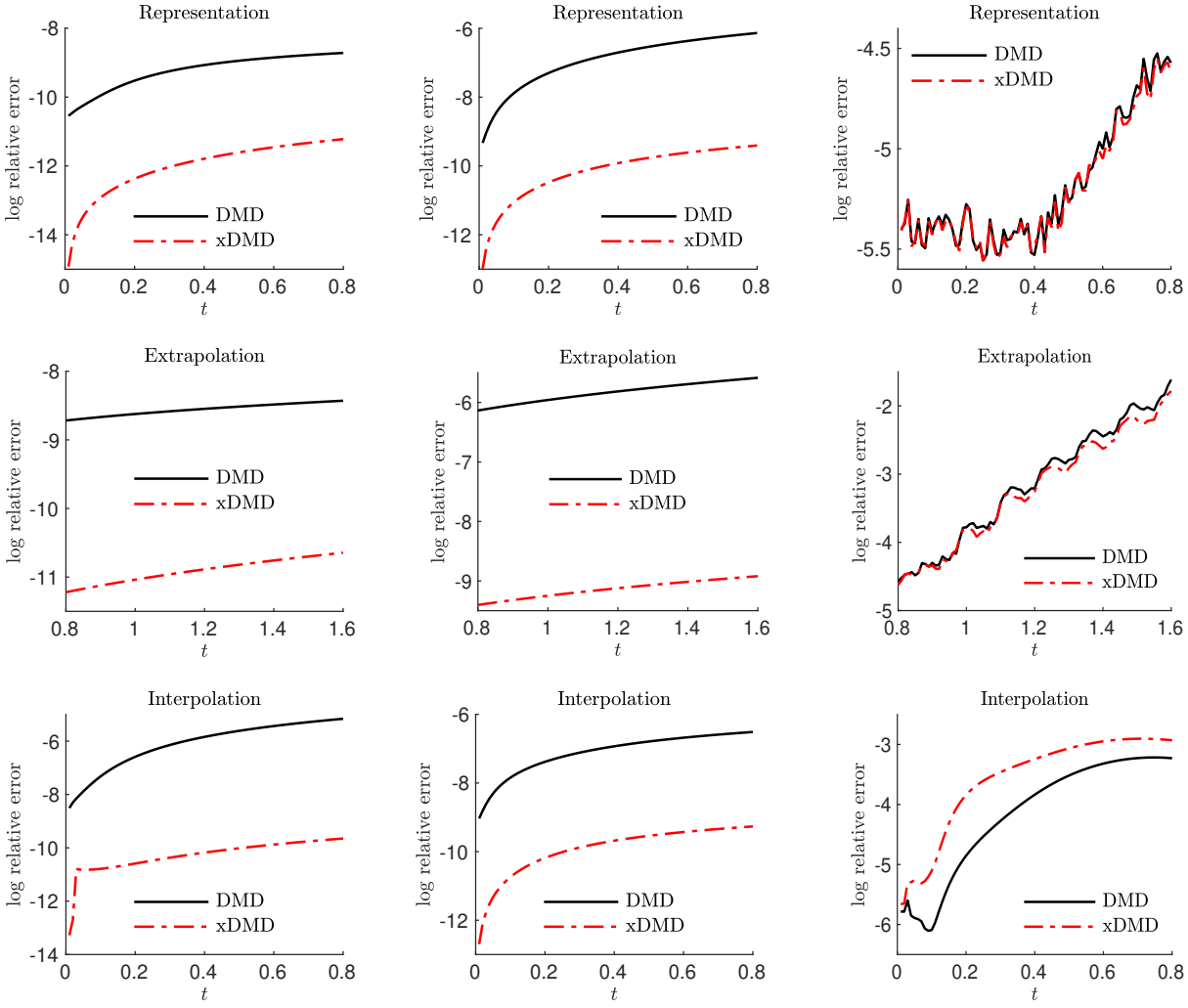

Figure S1: Solution accuracy in Cases 1 (left column), 2 (middle column) and 3 (right column) in the representation (top row), extrapolation (middle row) and interpolation (bottom row) regimes.

Two-dimensional Viscous Burgers' Equation

Consider the two-dimensional viscous Burgers' equation,

$$\begin{cases} \partial_t u + u\partial_x u + v\partial_y u = \nu(\partial_{xx} u + \partial_{yy} u) \\ \partial_t v + u\partial_x v + v\partial_y v = \nu(\partial_{xx} v + \partial_{yy} v). \end{cases} \tag{S4}$$

This equation, with $\nu = 0.05$, is defined for $(x, y) \in [0, 2] \times [0, 2]$ and $t \in [0, 2]$; and is subject to no-flux boundary conditions and the initial condition

$$u(x, y, 0) = v(x, y, 0) = \begin{cases} 1 & (x, y) \in [0.5, 1] \times [0.5, 1], \\ 0 & \text{otherwise.} \end{cases} \tag{S5}$$

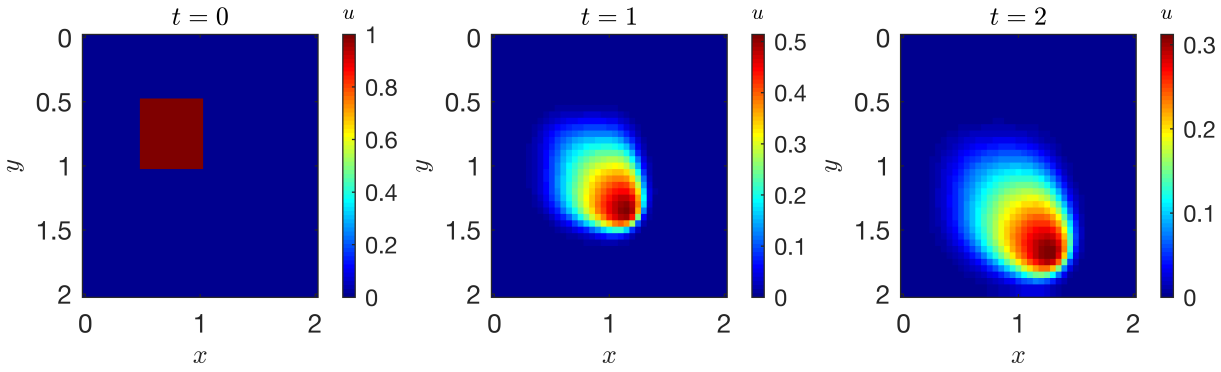

Figure S2: Temporal snapshots of the solution, $u(\mathbf{x}, t) = v(\mathbf{x}, t)$, to the 2D viscous Burgers' equation.

The reference solution is computed via a finite-difference scheme on a uniform mesh with $\Delta x = \Delta y = 0.05$ and $\Delta t = 0.001$. The snapshot solution needs to be reshaped into a vectorized form. We randomly select 500 snapshots out of the 2000 reference solutions to form the training data. Due to the viscosity ν, the solution exhibits weak nonlinearity and smooth diffusive profiles (Figure S2). Both DMD and xDMD capture the solution with satisfactory accuracy. We plot the relative error of xDMD is two orders of magnitude smaller than DMD (Figure S3).

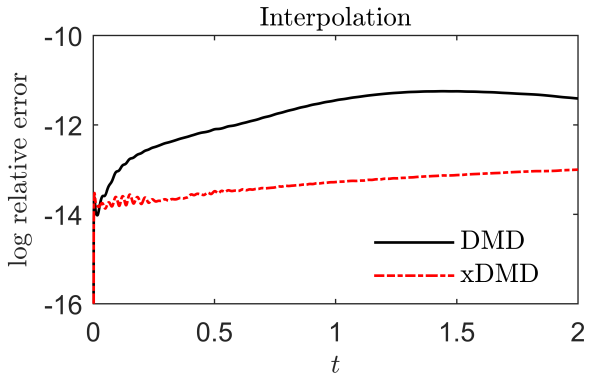

Figure S3: Temporal evolution of the log relative errors of the DMD and xDMD solutions, $\mathbf{u}(\mathbf{x}, t) = (u, v)^\top$, to the viscous Burgers' equation in the interpolation regime.

One-dimensional Advection-Diffusion Equation

Consider a one-dimensional advection-diffusion equation with a time-invariant source,

$$\begin{cases} \partial_t u + v\partial_x u = D\partial_{xx} u + S(x), & x \in [-4, 4], \quad t \in [0, 4], \\ S(x) = \exp(-x^2/0.2). \end{cases} \tag{S6}$$

We set $v = 1$ and $D = 0.1$. The training is conducted using the initial and boundary conditions

$$\begin{cases} u(x, 0) = \exp(-(x + 2)^2/0.1), \\ u_x(-4, t) = 0, \quad u_x(4, t) = 0. \end{cases} \tag{S7}$$

The initial condition mimics a localized source at point $x = -2$ with strength 1 and width $\sqrt{0.1}$.

The training data should be carefully chosen such that its traveling wave can cover the whole domain of interest and the training time should be sufficiently long. In this case, one should choose a training dataset with active pulses all over the domain $[-4, 4]$. Otherwise, the data-driven modeling will receive no signal in parts of the domain and, thus, fail to learn the global dynamics. This issue has been discussed in [19] for advection-dominant phenomena.

The training data are collected from reference solutions using a finite-difference scheme with $\Delta x = 0.04$ and $\Delta t = 0.04$. Figure S4 shows that both DMD and xDMD represent the training data with satisfactory accuracy. As in the previous tests, xDMD achieves higher-order accuracy than DMD.

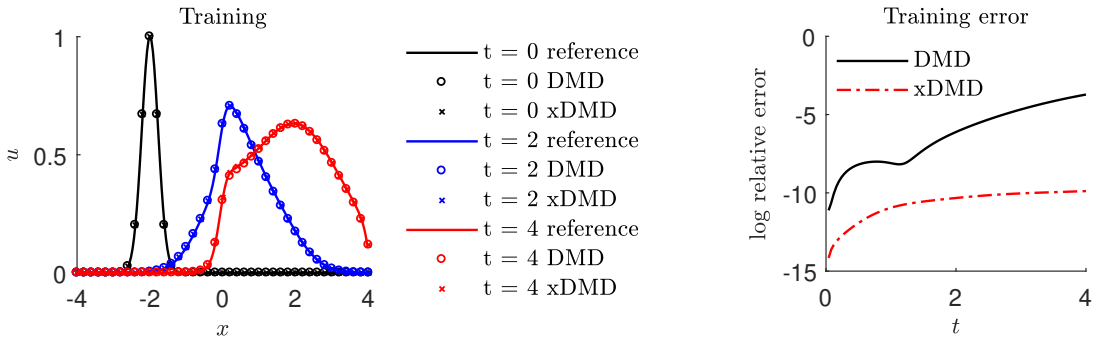

Figure S4: DMD and xDMD solutions to (S6) on the training data. Left: the DMD and xDMD solution profiles compared with the reference solution at different times; Right: temporal evolution of the log relative error.

Essentially, we want the DMD and xDMD models to learn, from the training data, the advection-diffusion operator with a fixed source. If the models are accurate, then for a different input (e.g., a point source with different strength, location and width), one can use the DMD and xDMD approximations to obtain solutions directly, without solving the governing equation. We test this generalizability on two types of inputs. In Test 1, the input data are generated for a single point source $u(x, 0) = s\exp(-(x - x^0)^2/\sigma^2)$, where $s \sim \mathcal{U}[1, 11]$, $x^0 \sim \mathcal{U}[-2, 1]$, and $\sigma^2 \sim \mathcal{U}[1/15, 1/10]$. In Test 2, the input data are

generated from a two-point source $u(x,0) = s_1 \exp(-(x-x_1^0)^2/\sigma_1^2) + s_2 \exp(-(x-x_2^0)^2/\sigma_2^2)$, where $s_1, s_2 \sim \mathcal{U}[1,11]$, $x_1^0, x_2^0 \sim \mathcal{U}[-2,1]$, and $\sigma_1^2, \sigma_2^2 \sim \mathcal{U}[1/15, 1/10]$.

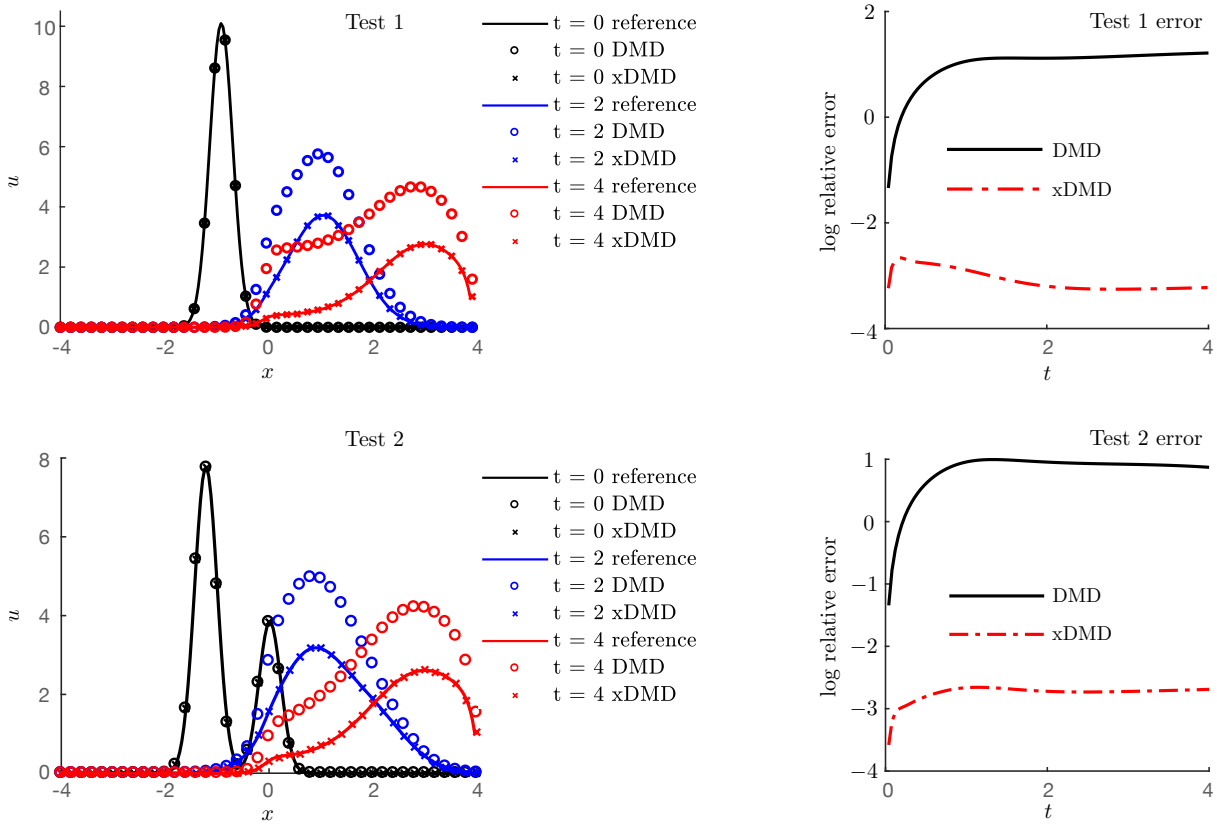

Figure S5: DMD and xDMD solutions to (S6) on the test data. Left: the DMD and xDMD solution profiles compared with the reference solution at different times; Right: temporal evolution of the log relative error.

Figure S5 shows that xDMD has superior performance in generalizing the learned model to new, previously unseen inputs. The modeling errors in the two tests are well controlled under reasonable magnitude. On the other hand, DMD has poor performance in generalization due to the lack of source term identification. The nature of (S6) implies that a good model should consist of two parts: one part accounts for the advection-diffusion operator, which is sensitive to the variation of the initial inputs; the other part accounts for the inhomogeneous source term, which is invariant to the initial inputs. This intuition is explicitly accounted for in the xDMD framework.

Two-dimensional Advection-Diffusion Equation

Next, we consider a two-dimensional advection-diffusion equation

$$\begin{cases} \partial_t u + \mathbf{v} \cdot \nabla u = \nabla \cdot (\mathbf{D} \nabla u) + S(x, y), & (x, y) \in [0, 20] \times [0, 10], \qquad t \in [0, T], \\ S(x, y) = s \exp\left[-\dfrac{(x-5)^2 + (y-5)^2}{2\sigma^2} \right]. \end{cases} \tag{S8}$$

We set

$$\mathbf{v} = \begin{pmatrix} -2.75 \\ 0.0 \end{pmatrix}, \qquad \mathbf{D} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}, \qquad s = 100.0, \qquad \sigma^2 = 0.05.$$

This equation is subject to no-flux boundary conditions and the initial condition $u(\mathbf{x}, 0) = u_{\text{in}}(\mathbf{x})$. The problem describes, e.g., the spatiotemporal evolution of the concentration $u(\mathbf{x}, t)$ of a groundwater contaminant that is advected by flow velocity $\mathbf{v}$, while undergoing hydrodynamic dispersion. The training data are generated for

$$u_{\text{in}}(\mathbf{x}) \equiv s \exp\left[-\frac{(x - x_s)^2 + (y - y_s)^2}{2\sigma^2} \right], \tag{S9}$$

wherein the coordinates of the plume's center of mass, (x_s, y_s) are treated as independent random variables with uniform distributions, $x_s \sim \mathcal{U}[0, 10]$ and $y_s \sim \mathcal{U}[0, 10]$. We generate $N_{\text{MC}} = 4000$ realizations of the pairs (x_s, y_s) and evaluate the corresponding initial conditions $u_{\text{in}}^{(n)}(\mathbf{x})$ for $n = 1, \ldots, N_{\text{MC}}$. For each of these realizations, (S8) is solved to compute $u_T^{(n)}(\mathbf{x}) \equiv u^{(n)}(\mathbf{x}, T)$ with $T = 4$ using a finite-difference scheme with $\Delta x = \Delta y = 0.25$. The matrix pairs $\{u_{\text{in}}^{(n)}, u_T^{(n)}\}_{n=1}^{N_{\text{MC}}}$ are arranged into data matrices $\mathbf{X}$ and $\mathbf{Y}$, as in (4.10). Finally, the DMD and xDMD models are deployed to learn the flow map $\mathbf{\Phi}_{\Delta t}$ with the time lag $\Delta t = T$.

We test the generalizability of the DMD and xDMD models by considering the following three tests.

- Test 1: The initial input is the same as in (S9) but the single point source is now allowed to have different strength and width: $s \sim \mathcal{U}(50, 100)$ and $\sigma^2 \sim \mathcal{U}(0.02, 0.1)$.

- Test 2: The initial input is a two-point source with different strengths, locations and widths:

$$u_{\text{in}} = s_1 \exp\left[-\frac{(x - x_{s_1})^2 + (y - y_{s_1})^2}{2\sigma_1^2} \right] + s_2 \exp\left[-\frac{(x - x_{s_2})^2 + (y - y_{s_2})^2}{2\sigma_2^2} \right], \tag{S10}$$

  where $s_1, s_2 = \mathcal{U}(50, 100)$, $\sigma_1^2, \sigma_2^2 = \mathcal{U}(0.02, 0.1)$, $x_{s_1}, x_{s_2} \sim \mathcal{U}[0, 10]$, and $y_{s_1}, y_{s_2} \sim \mathcal{U}[0, 10]$.

- Test 3: The initial input is a fixed-strength line source,

$$u_{\text{in}} = \begin{cases} 75 & x = 5, \quad y \in [3, 6], \\ 0 & \text{otherwise.} \end{cases} \tag{S11}$$

For Test 1, Figure S6 demonstrates the xDMD model's ability to accurately predict the solution for an initial condition not represented in the training set. At the same time, the DMD model fails this

relatively weak generalization test due to the reason given in section 5. The DMD error map has a peak centered at $(5, 5)$, which is the location of the source $S(\mathbf{x})$ in (S8). This further verifies that the loss of accuracy is caused by the shortcoming of DMD in identifying the inhomogeneous source term.

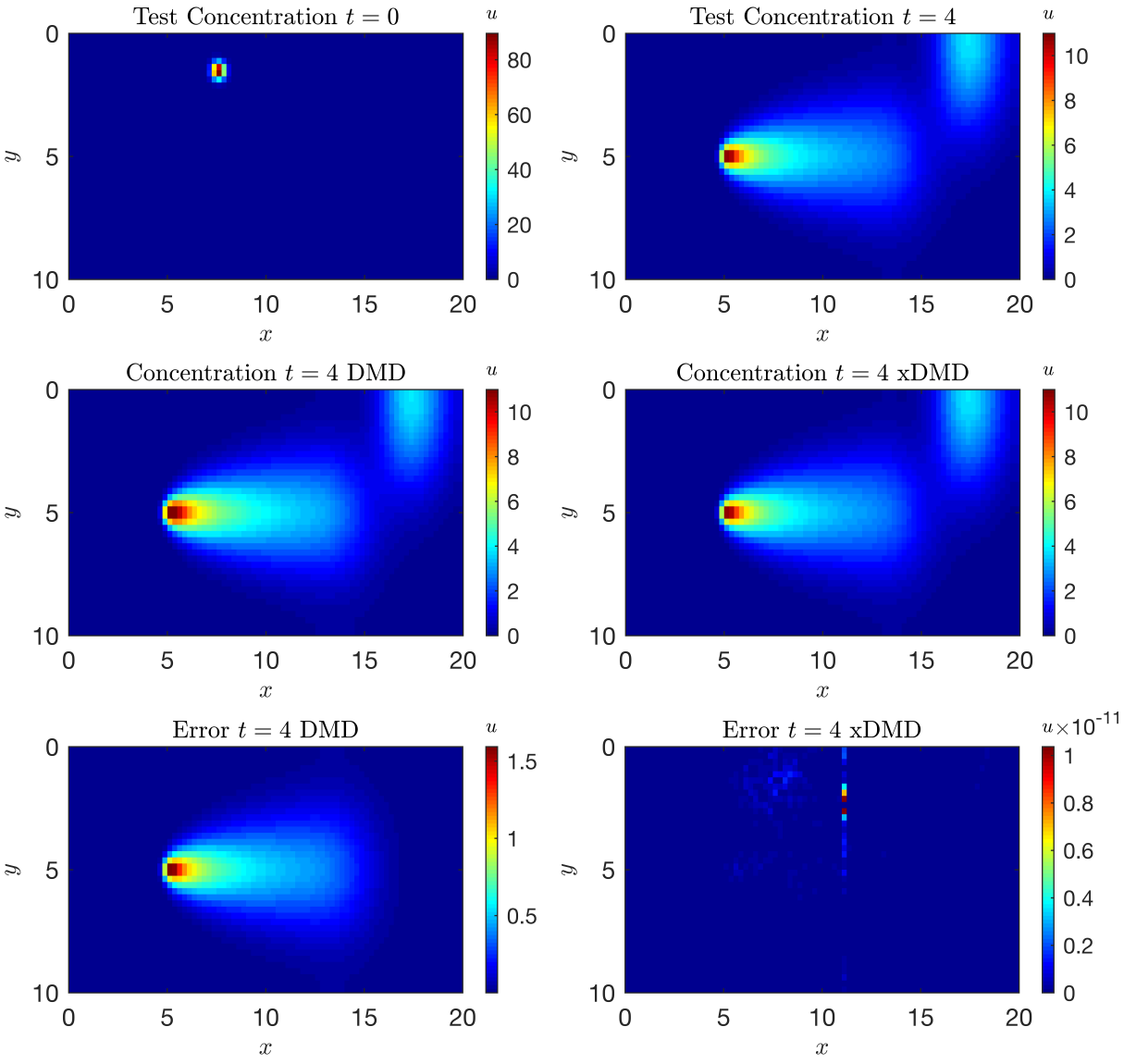

Figure S6: Test 1: The ground truth at the end of simulation time (top row) and its DMD and xDMD approximations (middle row), accompanied by the corresponding absolute error maps (bottom row).

Figure S7 reveals a similar performance of DMD and xDMD in the more challenging Test 2. As before, the xDMD model accurately predicts the solution at $T = 4$ corresponding to the two-point initial input not seen during the training. The right corner concentration tail is mostly caused by the advection-

diffusion effect on the north-east point source. This pure advection-diffusion dynamic is well captured by DMD, as shown by the flat low error concentration in the DMD error map. The error peak is at $(5, 5)$ again, showing the significant effect of identifying the source.

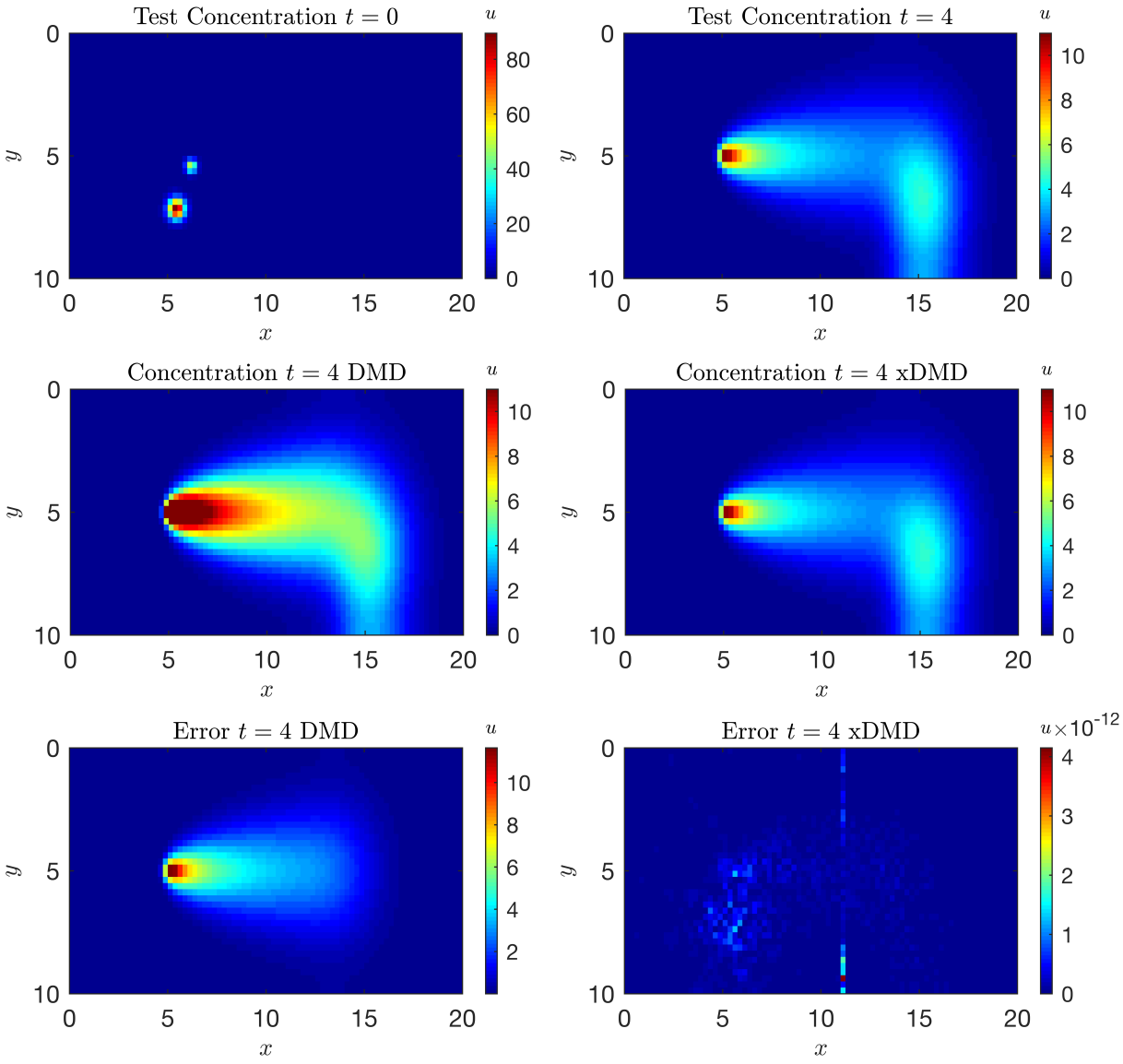

Figure S7: Test 2: The ground truth at the end of simulation time (top row) and its DMD and xDMD approximations (middle row), accompanied by the corresponding absolute error maps (bottom row).

Figure S8 demonstrates a similar behavior of the DMD and xDMD solutions for a line source initial input (Test 3). Although the solutions to (S8) with the single-point source, two-point source and line source exhibit quite different features, all of them can be thought of as a linear superposition of the

training single-point sources. Therefore, all the three types of the initial input can be regarded as drawn from the same distribution. As before, xDMD again achieves satisfactory accuracy in this generalizability test and DMD appears similar error map pattern centered at $(5, 5)$.

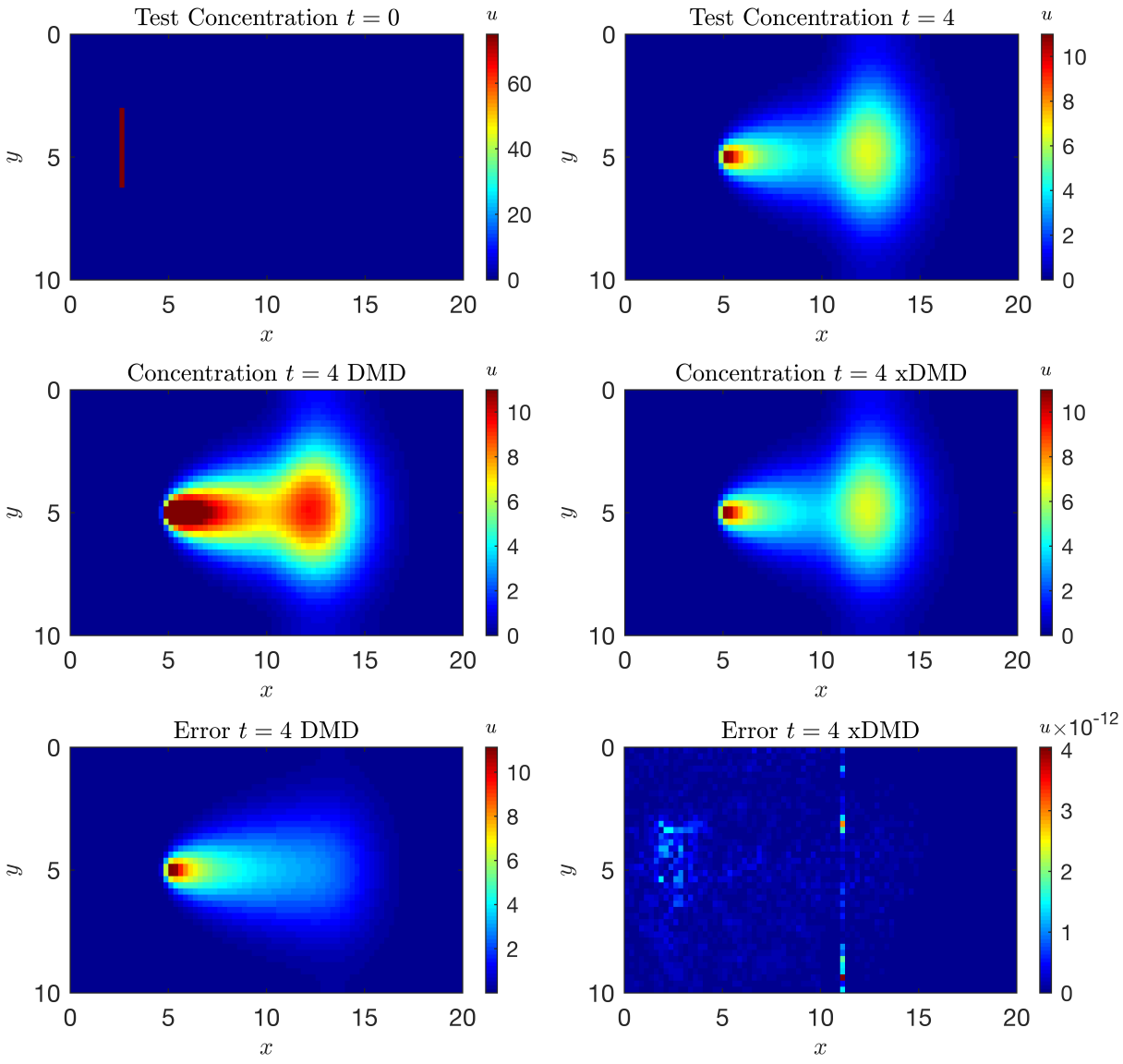

Figure S8: Test 3: The ground truth at the end of simulation time (top row) and its DMD and xDMD approximations (middle row), accompanied by the corresponding absolute error maps (bottom row).

Transport in Heterogeneous Media: Generalizability to New Inputs

The setting is identical to that in Section 4.4. Our goal here is to test the ability of DMD and xDMD models to predict $u(\mathbf{x}, T)$ for initial conditions not seen in training, such as a two-point source with

different strength and locations:

$$u_{\text{in}}(\mathbf{x}) = s_1 \exp(-(x - x_{s_1})^2 + (y - y_{s_1})^2) + s_2 \exp(-(x - x_{s_2})^2 + (y - y_{s_2})^2, \tag{S12}$$

where $s_1 = 50,\ s_2 = 80,\ (x_{s_1}, y_{s_1}) = (10, 40),\ (x_{s_2}, y_{s_2}) = (20, 20)$.

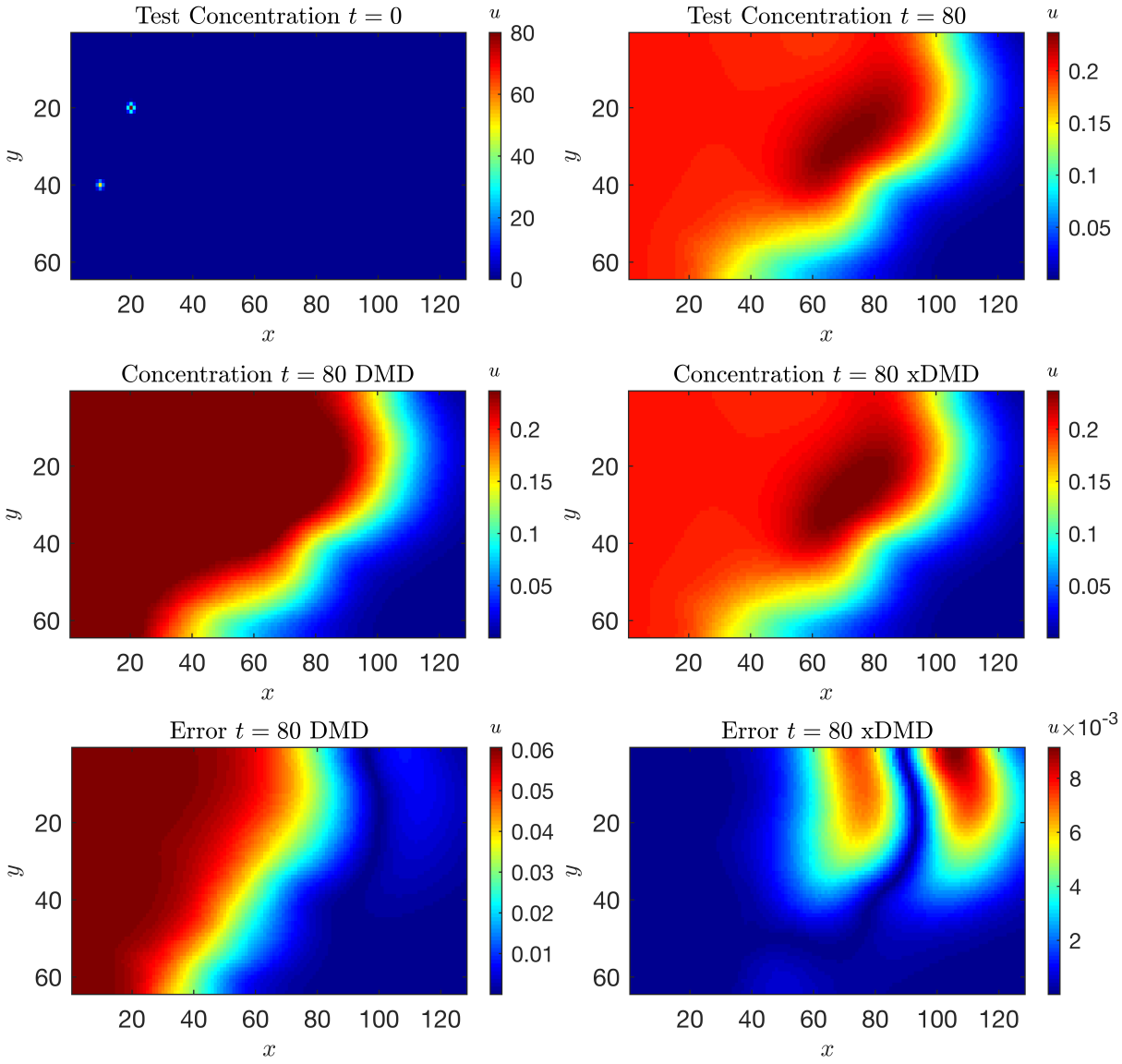

Figure S9: Section 4.4: The ground truth at the end of simulation time (top row) and its DMD and xDMD approximations (middle row), accompanied by the corresponding absolute error maps (bottom row).

Figure S9 shows the success of xDMD in learning the solution for an initial data not seen the training data. The xDMD error map has very small magnitude, indicating the high accuracy of xDMD in this

generalized test. On the other hand, DMD predicts a very different concentration map, failing the generalization test. As before, the DMD error is highest close to the left boundary, where the Dirichlet boundary condition is imposed. This visualization again addresses the significant role of the bias term added in the new xDMD framework.