Journal of Computational Science 30 (2019) 157–173

Contents lists available at ScienceDirect

Journal of Computational Science

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

pyROM: A computational framework for reduced order modeling

Vladimir Puzyrev[a,b,*], Mehdi Ghommem[c], Shiv Meka[b]

[a] School of Earth and Planetary Sciences, Curtin University, Perth, WA 6102, Australia
[b] Curtin Institute for Computation, Curtin University, Perth, WA 6102, Australia
[c] Department of Mechanical Engineering, American University of Sharjah, Sharjah 26666, UAE

ARTICLE INFO

Article history:
Received 12 July 2018
Received in revised form
20 November 2018
Accepted 12 December 2018
Available online 18 December 2018

Keywords:
Model reduction
Projection-based method
Open-source software
Proper orthogonal decomposition
Dynamic mode decomposition

ABSTRACT

Model reduction techniques reduce the overall complexity of dynamic systems and allow to speed up simulations of their behavior several orders of magnitude while retaining good accuracy. Despite being useful to obtain real-time simulations and apply control strategies, only few freely available software implementations of model reduction techniques have been reported in the literature. Furthermore, the use of these tools tends to be only for a limited range of dynamic problems, mostly related to fluid flows, and to deal with relatively small systems and datasets. In this paper, we build a portable, user-friendly, and open source computational framework, namely pyROM, implementing model reduction techniques in the Python programming language. This tool is designed to satisfy the needs of wide range of users to deploy model reduction for reproducing the dynamic response of high-dimensional models with good accuracy while achieving significant computational savings. The framework is designed in an object-oriented way to be easy to use and extend and employs visualization tools from various Python libraries such as Matplotlib, Mayavi, and Bokeh. Several numerical examples using modern spatial discretization methods such as the finite element method, the isogeometric analysis, the meshless point collocation method, and the generalized multiscale finite element method demonstrate the performance of the developed computational tool and the capabilities of model reduction methods to handle different engineering problems.

1. Introduction

The performance analysis of dynamic systems can be conducted at the earliest stages of design through the deployment of computational tools. Recent advances in computer hardware and software allow to reduce the computational burden associated with the numerical integration of the high-dimensional models described by linear and nonlinear partial differential equations (PDEs). Nevertheless, the extensive computational resources and time associated with the use of high-fidelity models usually limit the capability to simulate the large number of configurations required for design purposes. Certainly, the high-fidelity and computationally expensive numerical methods can model the behavior of large dynamic systems with great accuracy, capture all underlying physical features and explain relevant causal mechanisms. However, these systems are often dominated by low-dimensional dynamics that can be described by relatively simple models. For instance, one may describe and analyze fluid flows by extracting the most energetic modes from the raw data [1,45,46,2,16]. This leads to the need for the development of reduced-order modeling (ROM) methods that are capable of embodying dominant physical features and yielding relevant response characteristics within reasonable simulation times. These reduced-order models greatly decrease the computational demand, enable the reproduction of results of high-fidelity simulations with a predetermined accuracy, improve the understanding of the underlying physics and dynamics that govern the system's response, and perform a rapid and reasonably accurate exploration of the available design space, which is rather large in many real applications.

Over the last few decades, several model reduction techniques such as balanced truncation, proper orthogonal decompositions (POD), dynamic mode decomposition (DMD), discrete empirical interpolation method (DEIM), have been efficiently used for model reduction of dynamic systems [19,3,18,9,8,17,16]. Most of these techniques involve projection of the original governing equations onto a set of dominant modes. While the ROM techniques do not change for different applications, the requirements of these applications are often complex to manage in engineering practice. Furthermore, there is a tendency to implement each model reduc-

* Corresponding author.
E-mail addresses: vladimir.puzyrev@gmail.com (V. Puzyrev), mghommem@aus.edu (M. Ghommem), shiv.meka@curtin.edu.au (S. Meka).

https://doi.org/10.1016/j.jocs.2018.12.004
1877-7503/

tion technique for a particular problem. These applications are mostly related to fluid flows and are applied to relatively small systems and datasets. Furthermore, despite being an essential element to obtain real-time simulations and control dynamic systems, few software implementations of model reduction have been developed and reported in the literature [28,5,25,44]. These either have limitations to be applied for large scale problems or lack flexibility to be deployed for a broad range of applications. Milk et al. [28] implemented generic algorithms and interfaces for model reduction. They showed through numerical examples the capability of their tool to be integrated with existing open-source PDE solvers. Belson et al. [5] developed parallel Python library modred that includes the computationally-efficient implementations of POD, balanced POD, and DMD and can interface with existing codes written in C/C++ or Fortran.

In this paper, we present pyROM, a new Python framework for model order reduction. The goal of this framework is to provide an easy to use, efficient, and scalable package for the well-established ROM methods, as well as for the latest developments in the field. pyROM contains implementations of common model reduction methods such as proper orthogonal decomposition and dynamic mode decomposition, and discrete empirical interpolation method suitable for approximating nonlinear functions. The framework is written in a clear and concise way targeting a wide range of users, including those with little or no knowledge of model reduction principles.

The paper is organized as follows. We give a brief overview of ROM and describe the methods implemented in the framework in Section 2. Next, we discuss the main features of pyROM in Section 3. Section 4 presents several numerical examples of different complexity to show the applicability of ROM techniques to various modeling problems. Finally, in Section 5 we give concluding remarks and discuss future research directions.

2. Overview of model reduction

2.1. Problem formulation: large-scale system to reduced-order system

We first present projection-based techniques (other referred to as reduced basis methods) that reduce the dimension of general large-scale ordinary differential equations (ODEs) and parameterized algebraic equations that arise from space discretization of linear and nonlinear PDEs. Space discretization of a nonlinear PDE with well-defined boundary and initial conditions using finite element, finite volume, finite difference or other methods results in a system of nonlinear ODEs of the form

$$\begin{aligned} \frac{d}{dt}\mathbf{y}(t;\mu) &= \mathbf{A}\mathbf{y}(t;\mu) + \mathbf{F}(\mathbf{y}(t;\mu)) + \mathbf{G}, \\ \mathbf{y}(0) &= \mathbf{y}_0, \end{aligned} \tag{1}$$

where $t \in [0;T]$ refers to time, $\mathbf{y}(t;\mu) = [\mathbf{y}_1(t;\mu), \ldots, \mathbf{y}_n(t;\mu)] \in \mathbb{R}^n$ is an unknown function, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a discrete approximation of the linear spatial differential operator, $\mathbf{F} = [F(\mathbf{y}_1(t;\mu)), \ldots, F(\mathbf{y}_n(t;\mu))] \in \mathbb{R}^n$ is a nonlinear function, and $\mathbf{G} \in \mathbb{R}^n$ is a forcing or a source term. The parameter $\mu \in \mathfrak{D} \subset \mathbb{R}^d$, $d = 1, 2, \ldots$ denotes the systems properties (e.g., geometry, material properties, etc.) or any input parameter.

Similarly, space discretization of a steady nonlinear PDEs yields a system of algebraic equations of the form

$$\mathbf{R}(\mathbf{y}(\mu)) = \mathbf{A}\mathbf{y}(\mu) + \mathbf{F}(\mathbf{y}(\mu)) + \mathbf{G} = 0, \tag{2}$$

where $y(\mu) = [\mathbf{y}_1(\mu), \ldots, \mathbf{y}_n(\mu)] \in \mathbb{R}^n$ and $\mathbf{A}$, $\mathbf{F}$, and $\mathbf{G}$ are defined as in Eq. (1). The analysis of Eq. (2) is usually followed by the computation of the Jacobian as required for characterizing the effect of varying the parameter μ on the stability of the system. The Jacobian is given by

$$\mathbf{J}(\mathbf{y}(\mu)) = \frac{\partial \mathbf{R}(\mathbf{y}(\mu))}{\partial \mathbf{y}} = \mathbf{A} + \mathbf{J}_\mathbf{F}(\mathbf{y}(\mu)), \tag{3}$$

where

$$\mathbf{J}_\mathbf{F}(\mathbf{y}(\mu)) = \mathrm{diag}\{\frac{\partial F(\mathbf{y}_1(t))}{\partial \mu}, \ldots, \frac{\partial F(\mathbf{y}_n(t))}{\partial \mu}\}. \tag{4}$$

Projection-based techniques construct reduced-order models of dimension $k \ll n$ that approximate the original system governed by Eq. (1) or (2). We assume a subspace spanned by a reduced basis $\Phi = \{\phi\}_{i=1}^k \in \mathbb{R}^{n\times k}$ and approximate the solution vector as

$$\mathbf{y}(t) \approx \Phi\alpha(t), \tag{5}$$

where $\alpha \in \mathbb{R}^k$. Replacing $\mathbf{y}(t)$ in Eq. (1) and projecting the result onto the reduced set of basis functions Φ, we obtain the reduced-order model of the form

$$\begin{aligned} \frac{d}{dt}\alpha(t) &= \tilde{\mathbf{M}}\ \tilde{\mathbf{A}}\ \alpha(t) + \tilde{\mathbf{M}}\ \tilde{\mathbf{F}}(\Phi\ \alpha(t)) + \tilde{\mathbf{M}}\ \tilde{\mathbf{G}}, \\ \alpha(0) &= \tilde{\mathbf{M}}\ \Phi^T\mathbf{y}_0, \end{aligned} \tag{6}$$

where

$$\tilde{\mathbf{M}} = (\Phi^T\Phi)^{-1} \in \mathbb{R}^{k\times k}, \qquad \tilde{\mathbf{A}} = \Phi^T\mathbf{A}\Phi \in \mathbb{R}^{k\times k},$$

$$\tilde{\mathbf{F}} = \Phi^T\mathbf{F} \in \mathbb{R}^k, \qquad \tilde{\mathbf{G}} = \Phi^T\mathbf{G} \in \mathbb{R}^k. \tag{7}$$

As for the numerical integration of Eq. (6), both explicit or implicit schemes (e.g., Crank–Nicolson method, Euler method, Runge–Kutta method, etc.) can be used to advance in time.

Similarly, the reduced-order system of a steady nonlinear PDE (2) becomes

$$\tilde{M}\ \tilde{A}\ \alpha(\mu) + \tilde{M}\ \tilde{\mathbf{F}}(\Phi\ \alpha(\mu)) + \tilde{M}\ \tilde{\mathbf{G}}(\mu) = 0, \tag{8}$$

and Newton–Raphson method can be used to solve the above set of nonlinear algebraic equations. The corresponding reduced Jacobian is given by

$$\tilde{\mathbf{J}}(\alpha(\mu)) = \tilde{A} + \Phi^T\mathbf{J}_\mathbf{F}(\Phi\ \alpha(\mu))\Phi. \tag{9}$$

Once the reduced-order model is derived, as detailed above, it may be used to approximate the original system for different initial conditions, system's parameters, and input forcing assuming that the reduced basis functions $\Phi = \{\phi\}_{i=1}^k$ contain enough information about the response characteristics under varying operating conditions. Clearly, these basis functions or *modes* should be carefully selected to enable a robust implementation of the reduced-order model that satisfies the following conditions:

- Dynamically-relevant structures are captured in the modes ϕ_i.
- The error $\|\mathbf{y} - \Phi\ \alpha\|$ is small.
- System properties, such as stability, are preserved.
- The procedure is computationally efficient.

The construction of the reduced model from a given set of snapshots (the calculation of Φ) is also refered to as *offline* stage, while the solution of the reduced model (6) is called *online* stage. The offline stage is typically performed once and then is used for accurate and fast online evaluations. The particular mode decomposition methods for computing the set of reduced basis functions Φ are discussed and illustrated with numerical examples in the following sections. The software users have the option to provide the matrices **A**, **F**, and **G** obtained from their PDE solver.

2.2. Mode decomposition methods

Two most common mode decomposition methods for global model reduction of nonlinear dynamical systems are Proper Orthogonal Decomposition (POD) and Dynamic Mode Decomposition (DMD) [37]. Both of them are based on processing a sequence of snapshots (or instantaneous solutions) to compute a low-dimensional set of basis functions. These functions are then used to derive a low-dimensional dynamical system that is typically obtained by Galerkin projection [1,45,46,2,16,10]. POD constitutes a powerful mode decomposition technique for extracting the most energetic structures from a linear or nonlinear dynamical process [4,42,12,6,22,1,45,46,2,21,20,35,15]. Reduced basis constructed by POD is optimal in the sense that a certain approximation error concerning the given snapshots is minimal. In comparison with POD, DMD is intended to accurately extract the coherent and dynamically relevant structures rather than selecting the dominant modes that capture most of the flow's energy. DMD computes from simulation and empirical data the eigenvalues and eigenvectors of a linear model that best describes the underlying dynamics, even if those dynamics are generated from a nonlinear process. As such, this technique has been successfully applied to the analysis of experimental [14,38,32,36,40,27] and numerical [31,39,37,41,29] flow field data and has demonstrated a great capability to capture the relevant associated dynamics.

2.2.1. Proper orthogonal decomposition (POD)

POD enables the selection of the most-energetic modes or basis functions. Consider a set of snapshots $\mathbb{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_{n_s}\} \in \mathbb{R}^{n \times n_s}$, which are dynamic solutions equispaced in time; that is, $\mathbf{y}_i = \mathbf{y}(t_i) = \mathbf{y}(i \times \Delta t)$ where Δt is the time step. A POD basis of dimension $k \leq \min(n, n_s)$ is a set of orthonormal vectors $\{\phi\}_{i=1}^{k}$ such that their linear combination best approximates the set $\mathbb{Y}$ based on the energy content. The basis $\{\phi\}_{i=1}^{k}$ is obtained by solving the following minimization problem [37]

$$\min_{\Phi} \sum_{i=1}^{n_s} \|\mathbf{y}_i - \sum_{j=1}^{k} (\mathbf{y}_i^T \phi_j)\phi_j\|_2^2, \tag{10}$$

where the subscript denotes the L^2 norm. The modes ϕ_j's are orthonormal, that is, $\phi_i^T \phi_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta, and are ranked by their importance in the minimization of the L^2 error given by Eq. (10). For many physical applications, the norm is chosen such that minimizing the error corresponds physically to optimally capturing the kinetic energy (e.g., for fluid flows problems). The solution of the minimization problem is obtained from the set of the left singular vectors of the snapshot matrix $\mathbb{Y}$. Let the singular value decomposition (SVD) of $\mathbb{Y}$ be

$$\mathbb{Y} = \mathbf{V}\Sigma\mathbf{W}^*, \tag{11}$$

where $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_r] \in \mathbb{R}^{n \times r}$ and $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_r] \in \mathbb{R}^{n_s \times r}$ are orthogonal, * denotes the conjugate transpose, $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_r)$ is the set of singular values, and $r = \min(n, n_s)$. The POD modes are the vectors $\{\mathbf{v}_i\}_{i=1}^{k}$. The error between the snapshots and their approximation based on the POD modes is quantified in terms of the remaining singular values as

$$\sum_{i=1}^{n_s} \|\mathbf{y}_i - \sum_{j=1}^{k} (\mathbf{y}_i^T \phi_j)\phi_j\|_2^2 = \sum_{i=k+1}^{r} \sigma_i^2. \tag{12}$$

As such, the square of the singular value represents a measure of the energy content of each POD mode and thus provides guidance for the number of modes that should be considered in order to capture the relevant dynamics of the system.

SVD-based approach may be too computationally expensive, especially when dealing with large-scale problems (when n is large). Alternatively, one could use the method of snapshots [43] which allows for a significant reduction of the large data sets. In this method, an eigen analysis of the correlation matrix $\mathbf{C} = \mathbb{Y}^T \ \mathbb{Y}$ is performed and the POD modes are obtained as follows:

$$\mathbf{C} = \mathbb{Y}^T \ \mathbb{Y} \in \mathbb{R}^{n_s \times n_s} : \mathbf{C}u_i = \sigma_i^2 u_i \quad \text{and} \quad \phi_i = \frac{1}{\sigma_i} \mathbb{Y} \ u_i, \tag{13}$$

where u_i are the eigenvectors of the correlation matrix $\mathbf{C}$. This formulation is well-suited for large-scale problems because the eigenvalue problem does not depend on the dimension of $\mathbb{Y}$ but rather depends on the number of snapshots n_s which is usually much smaller than n. The implementation of the POD technique, as described above, is detailed in **Algorithm 1**. This algorithm is implemented in the developed framework pyROM to enable users to efficiently compute POD modes from the simulation data collected in the snapshot matrix $\mathbb{Y}$.

Algorithm 1. Proper Orthogonal Decomposition (POD)

Input: Sequence of n_s snapshots $\mathbb{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_{n_s}\} \in \mathbb{R}^{n \times n_s}$ sampled equispaced in time

Output: POD modes ϕ_j and associated singular values σ_j, $j = 1, \ldots, r$

1. Collect and store the snapshots in a snapshot matrix

$$\mathbb{Y} = \begin{pmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{n_s} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n \times n_s}.$$

2. Compute the correlation matrix $\mathbf{C} = \mathbb{Y}^T\mathbb{Y} \in \mathbb{R}^{n_s \times n_s}$.

3. Compute the eigenvalues and eigenvectors of $\mathbf{C}$
$\mathbf{C}u_i = \sigma_i^2 u_i$ or $\mathbf{C}\mathbf{U} = \mathbf{U}\Sigma^2$ where $\mathbf{U}^T\mathbf{U} = I_n$.

4. Sort the eigenvalues σ_i and corresponding eigenvectors u_i in descending order ranked by σ_i^2.

5. Select the number of modes k such that the error $\|E\| = \sum_{i=k+1}^{r} \sigma_i^2 < \epsilon$, where $\epsilon \ll 1$.

6. Compute matrices $\mathbf{U}_r$ and Σ_r by keeping the first r rows of $\mathbf{U}$ and the first r rows and columns of Σ^2, respectively

$$\mathbf{U} = \begin{pmatrix} \mathrm{U_r} & \cdots \end{pmatrix} \text{ and } \Sigma^2 = \begin{pmatrix} \Sigma_r^2 & \cdots \\ \vdots & \ddots \end{pmatrix}.$$

7. Compute matrix $\mathbf{M} = \mathrm{U_r} \ \ \Sigma_r^{-1}$.

8. Compute matrix $\Phi = \mathbb{Y}\mathbf{M}$ whose columns are the POD modes.

The selection of POD modes is based on an energy ranking (singular values) of the coherent structures. However, the energy may not be the appropriate measure to detect the most dynamically-relevant modes in all circumstances [11]. As such, other alternatives, like dynamic mode decomposition, can be considered to identify these modes.

2.2.2. Dynamic mode decomposition (DMD)

The DMD method originally proposed by Schmid et al. [37,39] is based on postprocessing a sequence of snapshots to extract the dynamic information. Consider a snapshot sequence $\mathbb{Y}_1^{n_s}$, where the subscript and superscript denote the first and last elements of the snapshot sequence, respectively. The DMD method uses the Arnoldi approach to relate the solution field $\mathbf{y}_i$ to the subsequent solution field $\mathbf{y}_{i+1}$ through a linear mapping A as $\mathbf{y}_{i+1} = A\mathbf{y}_i$. This assumption leads to a representation of the solution field as a Krylov sequence

$$\mathbb{Y}_1^{n_s} = \{\mathbf{y}_1, A\mathbf{y}_1, A^2\mathbf{y}_1, \cdots, A^{n_s-1}\mathbf{y}_1\}. \tag{14}$$

The objective is to determine the main characteristics of the dynamic process represented by the linear mapping A (even if the solution field involves nonlinear aspects). This is performed by computing or approximating the eigenvectors and eigenvalues of the matrix A. For instance, if A represents the Jacobian matrix, this analysis would provide stability information (the complex eigenvalues correspond to growth rates and frequencies). For a very large system, these computations may be numerically intractable. Furthermore, in some cases, the exact form of A may be not given. As

such, an efficient and fast numerical approach that approximates well the relevant dynamics is required. Below we give a brief mathematical description of the DMD-based approach [37,39], whose derivation closely follows arguments used in iterative methods for solution of linear algebraic systems.

The solution vector $\mathbf{y}_{n_s}$ can be represented by a linear combination of the previous solution fields as

$$\mathbf{y}_{n_s} = \mathbb{Y}_1^{n_s-1} \ \mathbf{a} + \mathbf{r}, \tag{15}$$

where $\mathbf{a} = \{a_1, a_2, \cdots, a_{n_s-1}\}^T$ and $\mathbf{r}$ is the residual vector. Combining Eqs. (14) and (15), we obtain

$$A \ \mathbb{Y}_1^{n_s-1} = \mathbb{Y}_2^{n_s} = \mathbb{Y}_1^{n_s-1} \ \mathbf{S} + \mathbf{r} \ \mathbf{e}_{n_s-1}^T, \tag{16}$$

where $\mathbf{e}_{n_s-1}^T = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix}$ is the (n_s-1) unit vector and $\mathbf{S}$ is a companion matrix defined as:

$$\mathbf{S} = \begin{pmatrix} 0 & & & a_1 \\ 1 & 0 & & a_2 \\ & \ddots & \ddots & \vdots \\ & & 1 \ \ 0 & a_{n_s-2} \\ & & 1 & a_{n_s-1} \end{pmatrix}. \tag{17}$$

The unknown matrix $\mathbf{S}$ is determined by minimizing the residual $\mathbf{r}$ which is obtained by expressing the n_sth snapshot $\mathbf{y}_{n_s}$ by a linear combination of $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \cdots, \mathbf{y}_{n_s-1}\}$ in a least-squares sense. The minimization problem is expressed as:

$$\mathbf{S} = \min_{\mathbf{S}} \| \mathbb{Y}_2^{n_s} - \mathbb{Y}_1^{n_s-1} \mathbf{S} \|. \tag{18}$$

To avoid cumbersome notation, we use the same variable for the minimizer as the variable that is being minimized. The solution can be determined either using a QR decomposition of the snapshot matrix $\mathbb{Y}_1^{n_s-1}$ [39]:

$$\mathbf{S} = \mathbf{R}^{-1} \ \mathbf{Q}^T \ \mathbb{Y}_2^{n_s}, \tag{19}$$

or using the Moore–Penrose pseudoinverse of $\mathbb{Y}_1^{n_s-1}$ [11]:

$$\mathbf{S} = ((\mathbb{Y}_1^{n_s-1})^T \ \mathbb{Y}_1^{n_s-1})^{-1} \ (\mathbb{Y}_1^{n_s-1})^T \ \mathbb{Y}_2^{n_s}. \tag{20}$$

Once $\mathbf{S}$ is computed, the next step is to evaluate its eigenvalues and eigenvectors. The DMD spectrum λ_j is obtained by logarithmic transform of the eigenvalues of $\mathbf{S}$ from the time-stepper format to the format more commonly used in stability theory:

$$\lambda_j = \log(\mathbf{D}_{jj})/\Delta t. \tag{21}$$

The dynamic modes ϕ_j are computed by weighting the snapshot matrix by the eigenvectors of $\mathbf{S}$ as:

$$\phi_j = \mathbb{Y}_1^{n_s-1} \ \mathbf{X}_j, \tag{22}$$

where $\mathbf{X}_j$ is the j-th column of the eigen decomposition matrix $\mathbf{X}$. The detailed description of all steps is given in **Algorithm 2**.

In practice, this method for computing the dynamic modes based on the companion matrix $\mathbf{S}$ may be ill-conditioned. As such, Schmid [37] proposed a more robust implementation described below as **Algorithm 3**. First, we perform the singular value decomposition $\mathbb{Y}_1^{n_s-1} = \mathbf{U} \ \Sigma \ \mathbf{W}$. Substituting the SVD representation $\mathbf{U} \ \Sigma \ \mathbf{W}$ into Eq. (16) and multiplying the result by $\mathbf{U}^T$ from the left and by $\mathbf{W}^T \ \Sigma^{-1}$ from the right, we obtain the following matrix

$$\mathbf{U}^T \ A \ \mathbf{U} = \mathbf{U}^T \ \mathbb{Y}_2^{n_s} \ \mathbf{W}^T \ \Sigma^{-1} \equiv \tilde{\mathbf{S}}. \tag{23}$$

For the configuration sets with $\dim(\mathbf{y}_i) = n \gg n_s$, the method of snapshots, as described in Section 2.2.1, is usually used to avoid the computational burden associated with the singular value decomposition of large matrices. The matrix $\mathbf{U}$ contains the POD modes of the sequence of snapshots $\mathbb{Y}_1^{n_s-1}$, and then the matrix $\tilde{\mathbf{S}}$ is obtained from the projection of the linear operator A, which is used to approximate the underlying dynamical process onto a the POD basis. The main advantage of this method is its ability to take into account the rank-deficiency of $\mathbb{Y}_1^{n_s-1}$ by considering a limited basis $\mathbf{U}$ given only by the non-zero singular values of Σ (or by singular values above a threshold that can be determined based the amount of cumulative energy that needs to be captured). The dynamic modes are computed from $\tilde{\mathbf{S}}$ as follows:

$$\phi_j = \mathbf{U} \ \mathbf{v}_j, \tag{24}$$

where $\mathbf{v}_j$ is the jth eigenvector of $\tilde{\mathbf{S}}$, i.e., $\tilde{\mathbf{S}} \ \mathbf{v}_j = \mu_j \mathbf{v}_j$, and $\mathbf{U}$ is the matrix of the right singular vectors of the snapshot sequence $\mathbb{Y}_1^{n_s-1}$.

Algorithm 2. Dynamic Mode Decomposition (DMD). Approach 1

Input: Sequence of n_s snapshots $\mathbb{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_{n_s}\} \in \mathbb{R}^{n \times n_s}$ sampled equispaced in time
Output: DMD modes ϕ_j and associated spectrum λ_j, $j = 1, \dots, r$
1. Collect and store the snapshots in two consecutive sequences
$$\mathbb{Y}_1^{n_s-1} = \begin{pmatrix} | & | & | & \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{n_s-1} \\ | & | & | & \end{pmatrix}, \quad \mathbb{Y}_2^{n_s} = \begin{pmatrix} | & | & | & \\ \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_{n_s} \\ | & | & | & \end{pmatrix}.$$
2. Compute the QR decomposition of $\mathbb{Y}_1^{n_s-1} = \mathbf{Q} \ \mathbf{R}$.
3. Compute $\mathbf{S}$ as $\mathbf{S} = \mathbf{R}^{-1} \ \mathbf{Q}^T \ \mathbb{Y}_2^{n_s}$ or using the Moore-Penrose pseudoinverse of $\mathbb{Y}_1^{n_s-1}$ as $\mathbf{S} = ((\mathbb{Y}_1^{n_s-1})^T \ \mathbb{Y}_1^{n_s-1})^{-1} \ (\mathbb{Y}_1^{n_s-1})^T \ \mathbb{Y}_2^{n_s}$.
4. Compute the eigenvalues and eigenvectors of $\mathbf{S}$ as $\mathbf{S} \ \mathbf{X}_i = \mathbf{D}_{ii} \ \mathbf{X}_i$ or $\mathbf{S} \ \mathbf{X} = \mathbf{X} \ \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix.
5. Compute the dynamic mode spectrum λ as $\lambda_i = \log(\mathbf{D}_{ii})/\Delta t$.
6. Compute the dynamic modes as $\phi_i = \mathbb{Y}_1^{n_s-1} \ \mathbf{X}_i$.

Algorithm 3. Dynamic Mode Decomposition (DMD). Approach 2

Input: Sequence of n_s snapshots $\mathbb{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_{n_s}\} \in \mathbb{R}^{n \times n_s}$ sampled equispaced in time
Output: DMD modes ϕ_j and associated spectrum λ_j, $j = 1, \dots, r$
1. Collect and store the snapshots in two consecutive sequences
$$\mathbb{Y}_1^{n_s-1} = \begin{pmatrix} | & | & | & \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{n_s-1} \\ | & | & | & \end{pmatrix}, \quad \mathbb{Y}_2^{n_s} = \begin{pmatrix} | & | & | & \\ \mathbf{y}_2 & \mathbf{y}_2 & \cdots & \mathbf{y}_{n_s} \\ | & | & | & \end{pmatrix}.$$
2. Perform SVD of the snapshot matrix $\mathbb{Y}_1^{n_s-1} = \mathbf{U} \ \Sigma \ \mathbf{W}$ or compute the correlation matrix $\mathbf{C} = (\mathbb{Y}_1^{n_s-1})^T \mathbb{Y}_1^{n_s-1}$ and proceed as described in Algorithm 1.
3. Compute $\tilde{\mathbf{S}} = \mathbf{U}^T \ \mathbb{Y}_2^{n_s} \ \mathbf{W}^T \ \Sigma^{-1}$.
4. Compute the eigenvalues and eigenvectors of $\tilde{\mathbf{S}}$ as $\tilde{\mathbf{S}} \ \mathbf{X}_i = \mathbf{D}_{ii} \ \mathbf{X}_i$ or $\tilde{\mathbf{S}} \ \mathbf{X} = \mathbf{X} \ \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix.
5. Compute the dynamic mode spectrum λ as $\lambda_i = \log(\mathbf{D}_{ii})/\Delta t$.
6. Compute the dynamic modes as $\phi_i = \mathbf{U} \ \mathbf{X}_i$.

We illustrate in the appendix the robustness of **Algorithm 3** in approximating the eigenvalues with good accuracy and the failure of **Algorithm 2** for some cases. As such, **Algorithm 3** is implemented in pyROM to allow users to compute efficiently the DMD modes.

2.2.3. Discrete empirical interpolation method (DEIM)

When dealing with ROM of nonlinear systems obtained by projecting the governing equations onto a subspace spanned by the POD or DMD modes, evaluating the projected nonlinear term $\mathbf{F}(\Phi \ \alpha(t))$ in Eq. (6) can be costly since it depends on the full dimension n of the original system. It requires about $2 \times n \times k$ flops for matrix multiplications and also involves a full evaluation of $\mathbf{F}$ at the n-dimensional vector $\Phi \ \alpha(t)$. As such, the reduction in the computational cost, when employing mode decomposition methods, can be limited by the full evaluation of the projected nonlinear function. To deal with this issue, Chaturantabut and Sorensen [10] proposed the discrete empirical interpolation method (DEIM) in the context of nonlinear model reduction. DEIM chooses a small set of points

and forms an approximation to the full nonlinear function by interpolation through these points. Hence, function evaluations at only few selected points $m \ll n$ are required, which enables significant reduction in the computational cost. Matrix operations now require $2 \times m \times k$ operations, which is independent of dimension n of the full-order system [10].

DEIM is based on the approximation of the nonlinear terms by means of an interpolatory projection of m selected snapshots of the nonlinear terms. We briefly describe the DEIM following [10]. Let $\mathbf{f}(\tau) \in \mathbb{R}^n$ denote a nonlinear function where τ refers to time t or any control parameter μ. We assume an approximation of the nonlinear function $\mathbf{f}$ obtained by projecting it into a subspace spanned by the basis functions $\mathbf{U}_f = (\mathbf{u}_{f_1}, \ldots, \mathbf{u}_{f_m}) \in \mathbb{R}^{n \times m}$ as

$$\mathbf{f}(\tau) \approx \mathbf{U}_f \mathbf{c}(\tau). \tag{25}$$

To compute the coefficient vector $\mathbf{c}$, we consider the following matrix

$$\mathbf{P} = [\mathbf{e}_{\wp_1}, \ldots, \mathbf{e}_{\wp_m}] \in \mathbb{R}^{n \times m},$$

where $\mathbf{e}_{\wp_i} = [0, \ldots, 0, 1, 0, \ldots, 0]^T \in \mathbb{R}^n$ is the $\wp_i$-th column of the identity matrix $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ for $i = 1, \ldots, m$. Multiplying Eq. (25) by $\mathbf{P}^T$ and assuming that the matrix $\mathbf{P}^T\mathbf{U}_f$ is nonsingular, we obtain

$$\mathbf{f}(\tau) \approx \mathbf{U}_f \mathbf{c}(\tau) = \mathbf{U}_f (\mathbf{P}^T \mathbf{U}_f)^{-1} \mathbf{P}^T \mathbf{f}(\tau). \tag{26}$$

The approximation of the nonlinear function $\mathbf{f}(\tau)$, as given by Eq. (26), involves the computation of the projection basis $\mathbf{U}_f = (\mathbf{u}_{f_1}, \ldots, \mathbf{u}_{f_m})$ and identification of the indices $\{\wp_1, \ldots, \wp_m\}$. To determine the projection basis $\mathbf{U}_f$, we assemble the function evaluations in an $n \times n_s$ matrix $(\mathbf{f}(\tau_1), \ldots, \mathbf{f}(\tau_{n_s}))$ and employ the POD technique (i.e., apply **Algorithm 1**) to compute the POD modes. These modes are used as the projection basis in the approximation given by (25). Interpolation indices $\{\wp_1, \ldots, \wp_m\}$ are selected using the DEIM algorithm described below (**Algorithm 4**). The implementation of DEIM is incorporated in pyROM to enable users to approximate nonlinear functions while reducing the simulation time.

Algorithm 4. Discrete Empirical Interpolation Method (DEIM)

Input: Projection basis $\mathbf{U}_f = (\mathbf{u}_{f_1}, \ldots, \mathbf{u}_{f_m}) \in \mathbb{R}^{n \times m}$ obtained by applying POD on a sequence of n_s function evaluations
Output: Interpolation indices $\vec{\wp} = (\wp_1, \ldots, \wp_m)^T$
1. Set $[|\rho|, \wp_1] = \max\{|\mathbf{u}_{f_1}|\}$.
2. Set $\mathbf{U}_f = (\mathbf{u}_{f_1})$, $\mathbf{P} = (\mathbf{e}_{\rho_1})$, and $\vec{\wp} = (\wp_1, \ldots, \wp_m)$.
3. **for** ℓ=2 t m **do**
Solve $(\mathbf{P}^T\mathbf{U}_f)\mathbf{c} = \mathbf{P}^T\mathbf{u}_{f_\ell}$.
Compute $\mathbf{r} = \mathbf{u}_{f_\ell} - \mathbf{U}_f\mathbf{c}$.
Compute $[|\rho|, \wp_\ell] = \max\{|\mathbf{r}|\}$.
Set $\mathbf{U}_f = (\mathbf{U}_f \quad \mathbf{u}_{f_\ell})$, $\mathbf{P} = (\mathbf{P} \quad \mathbf{e}_{\rho_\ell})$, and $\vec{\wp} = \begin{pmatrix} \vec{\wp} \\ \wp_\ell \end{pmatrix}$.
end for
Here $\mathbf{e}_{\wp_i}$ is the $\wp_i$-th column of the identity matrix $\mathbf{I}_n$,
$[|\rho|, \wp_\ell] = \max\{|\mathbf{r}|\}$ implies $|\rho| = |\mathbf{r}_{\wp_\ell}| = \max_{i=1,\cdots,n}\{|\mathbf{r}_i|\}$

For the application of DEIM to nonlinear reduced-order models given by (6), the approximation of the nonlinear function is written as

$$\begin{aligned} \mathbf{F}(\Phi\alpha(t)) &\approx \mathbf{U}_f (\mathbf{P}^T\mathbf{U}_f)^{-1} \mathbf{P}^T \mathbf{F}(\Phi\alpha(t)) \\ &= \mathbf{U}_f (\mathbf{P}^T\mathbf{U}_f)^{-1} \mathbf{F}(\mathbf{P}^T\Phi\alpha(t)). \end{aligned} \tag{27}$$

Hence

$$\tilde{\mathbf{F}}(\Phi\alpha(t)) \approx \Phi^T\mathbf{U}_f (\mathbf{P}^T\mathbf{U}_f)^{-1} \mathbf{F}(\mathbf{P}^T\Phi\alpha(t)). \tag{28}$$

The term $\Phi^T\mathbf{U}_f(\mathbf{P}^T\mathbf{U}_f)^{-1} \in \mathbb{R}^{k \times m}$ of the last equation is computed once and stored, while the term $(\mathbf{P}^T\Phi)\alpha(t) \in \mathbb{R}^{m \times 1}$ is obtained by extracting the rows $\wp_1, \ldots, \wp_m$ of the modes collected in Φ and then multiplying the result by α. Here comes the computational saving since such operation requires only the evaluation of the nonlinear function at m selected points rather than performing a full evaluation over the whole system domain. Computational speedup of 150–180 has been reported recently for application of DEIM to a high-fidelity model of ≈100, 000 degrees of freedom [30].

3. Computational tool description

High-level programming languages, such as MATLAB or general-purpose Python, provide a flexible environment for a wide range of numerical problems. They enable native interfacing of code written in different low-level languages to leverage the ease of use provided by high-level interpretive languages with the numerical efficiency of compiled languages. pyROM is written in Python that is popular, easy to use, freely available, high-level programming language that supports multiple programming paradigms, including object-oriented construct, and is widely used in the scientific computing community. Python's design philosophy emphasizes code readability and conciseness. Its syntax allows programmers to express concepts in fewer lines of code than might be used in other widely used programming languages. Python also has a large and comprehensive standard library that facilitates the programming of a wide array of tasks and can easily interface with codes in other languages including C/C++ and Fortran. Users can provide data in files or provide functions that generate their data in another way, thus making pyROM compatible with any type of data.

pyROM is based on the following principles:

- Usability and ease of use – the framework can be used by anyone with only basic knowledge of model reduction principles. The details of the implementation of the underlying algorithms are hidden from the user, thus making ROM methods accessible to a wider scientific audience.
- Extensibility and portability – pyROM is designed and implemented in such a way that it is easy to extend and port to different computer architectures. Python is available on a wide range of computer architectures and operating systems.
- Standard libraries and visualization – the framework makes an active use of standard Python libraries and tools, including specialized numerical libraries and visualization routines. Several visualization tools are accessible in pyROM through Python interface.
- Documentation and license – the code is well documented and is freely available under an open-source license that enables its modification and distribution with almost no restrictions.

pyROM is developed using object-oriented methodology to separate the physical problem formulation and the ROM method that is used to build the reduced model. This provides a greater degree of flexibility for the problem formulation and reduced solution allowing to combine several ROM techniques. All pyROM modules mirror the underlying mathematical algorithms described in Section 2 for clarity and easy understanding of the code.

The framework leverages inherent multi-threaded optimization that certain high-level Python libraries such as NumPy and SciPy [24] are configured with. The aforementioned dependencies are compiled using Intel MKL library that is highly optimized for Intel x86-64 architectures. While the framework is built to minimize verbosity in terms of user input, there is, however, a protocol that pyROM conforms to in reading user supplied matrices as briefed in Fig. 1. A single MATLAB/JSON formatted file encompassing data pertaining to several physical variables such as the mass and stiffness matrices (**M**, **K**), the snapshot matrix (**R_enrich**), perturbation (**F**) and boundary conditions is expected as input. The module expects the matrices to be defined in the optimized CCS sparse matrix format. The user can then choose the ROM method between the two

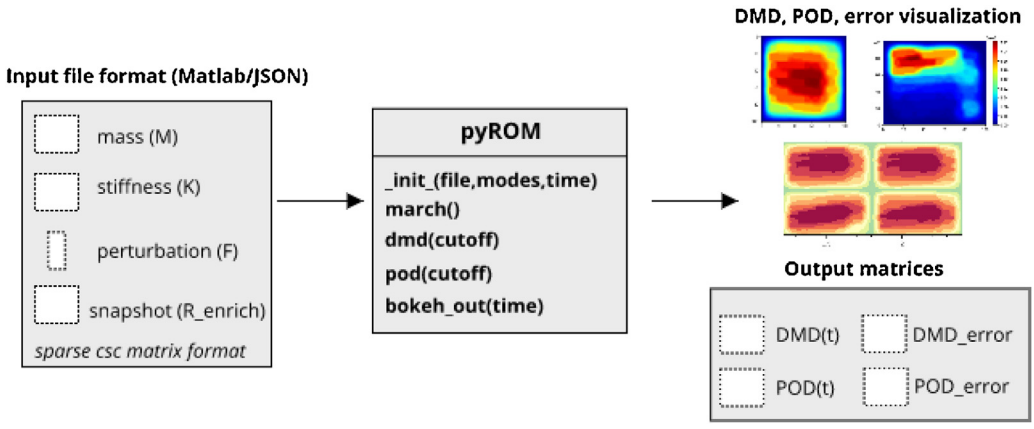


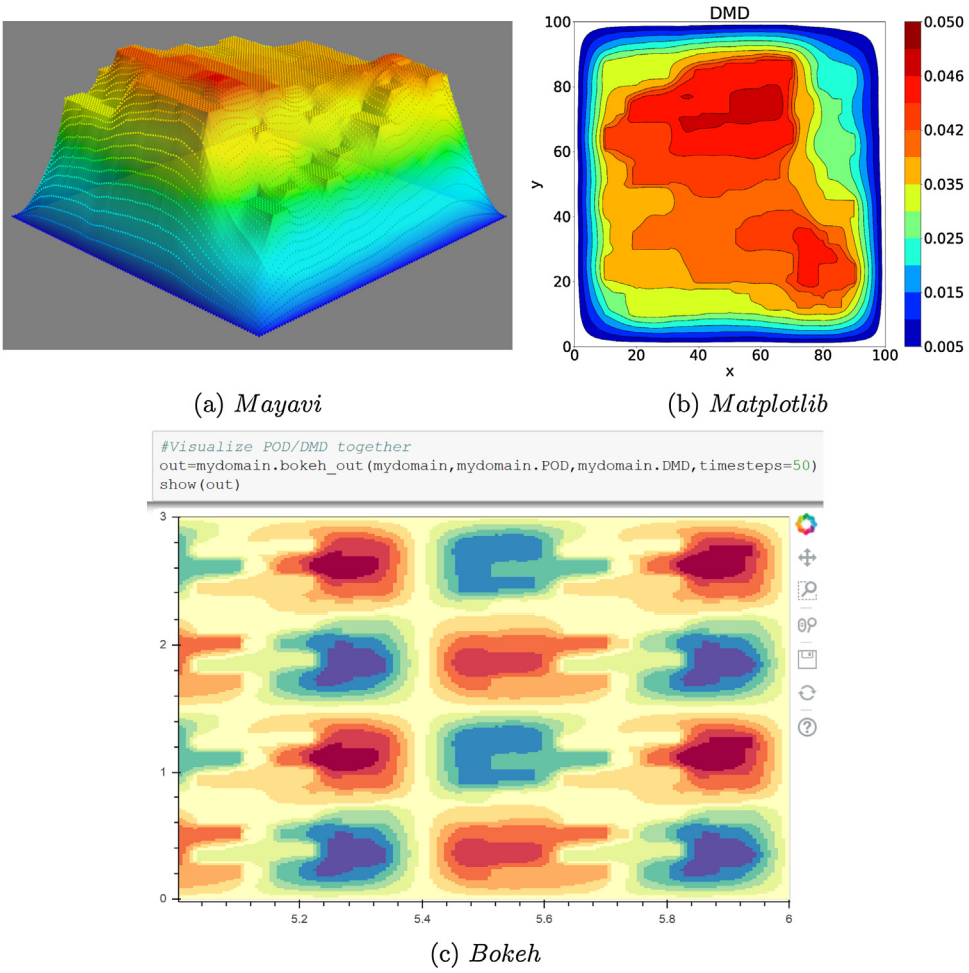
Fig. 1. Structure of the pyROM workflow.

currently available routines - POD and DMD, define the number of modes needed for model reduction and specify timestepping parameters.

The code for the heat equation is presented as an example in *Code Snippet 1*. The module enables users to provide analytical inputs, e.g., perturbation function **F** in Fig. 1, in the form of lambda functions. Timing analytics for each subroutine is also made available to the user. Depending on the problem, and based on the I/O that is local to the underlying system, the computational speed is limited by the execution time of the eigenvalue decomposition and the size of the input matrix.

Snippet 1 Heat Equation

```
#Import pyROM functions
from pyROM import *

#Plotting library
from bokeh.plotting import show, output_notebook

#Perturbation
f=lambda x:np.sin(np.float(2*np.pi*x[0]))
                              *np.sin(2*np.pi*x[1])

#Domain decomposition
domain=Domain('examples/porous.mat',perturbation=f,cutoff=20,
                                  timesteps=2000)

#Run full simulation
domain.march()

#Computes POD and stores the value in domain.POD
domain.pod()
#Computes DMD and stores the value in domain.DMD
domain.dmd()

#Print timing information
print ("Initialization: "+domain.wall
       +"\nPOD: "+domain.POD.wall
       +"\nDMD: "+domain.DMD.wall)


#Single timestep visualization DMD and POD
plt.imshow(domain.DMD[:,1500].reshape([101,101]))
plt.imshow(domain.POD[:,1500].reshape([101,101]))

#Full series POD visualization
output_notebook()  #Opens bokeh notebook
out=domain.bokeh_out(domain,domain.POD,timesteps=300)
show(out)
#Full series DMD visualization
out=domain.bokeh_out(domain,domain.DMD,timesteps=300)
show(out)
#Visualize POD/DMD together
out=domain.bokeh_out(domain,domain.POD,domain.DMD,axis=0,
                                  timesteps=300)
show(out)
```



(a) *Mayavi* (b) *Matplotlib*

(c) *Bokeh*

Fig. 2. Mayavi, Matplotlib, and Bokeh visualization examples.

The module also supports running full-order solution to enable the calculation of ROM errors. POD and DMD solutions (matrices - class.POD, class.DMD) are made available along with their associated errors (matrices class.POD_error, class.DMD_error). Output matrices assume the shape (X*Y $\times$ timesteps), where X and Y are the dimensions of the input matrix.

Regardless of the accuracy of the numerical methods, sharp and highly-resolved snapshots constitute the norm in scientific publications. Having such capability would be helpful in the interpretation of the simulation results and developing a good understanding of the dynamic system behavior. As such, subroutines to plot and export visualizations using frameworks such as Mayavi [34] and Bokeh [7] are also included in pyROM. In Fig. 2, we show examples of visualization of the same data (DMD approximation of the problem described in Section 4.1.2) using 3D visualization in Mayavi, 2D visualization in Matplotlib and Bokeh interactive window. The user has the capability to select any of these visualization tools to analyze the numerical data. Bokeh, in particular, can be used for presentation in web browsers providing high-performance interactivity in very large datasets. For example, in Fig. 2(c), we show an interactive window that allows the user to compare any of the POD and DMD snapshots (using standard scrolling and zooming tools) in the whole snapshot sequence.

4. Numerical examples

In this section, we illustrate the performance of the three model reduction methods for a number of physical problems to demonstrate how a given problem can be introduced to the framework, and how pyROM can be used to benchmark the different model reduction algorithms. When we compare the computational time of full and reduced models, we consider both the calculation of modes (offline stage) and the solution of the reduced model (online stage) as the total time of the reduced-model simulation. We note that the online stage is typically much faster compared to the offline stage and the speedup of using the reduced model for simulations (online solution only) often is in the range of hundreds or thousands when compared to the full-scale simulation.

4.1. POD and DMD: illustrative examples

First, we test the performance of pyROM by simulating two different PDEs, namely the diffusion equation and flow in porous media problem.

4.1.1. Diffusion equation

As the first example, we consider the diffusion equation, which is one of the most common second-order PDEs. It describes the process of diffusion, i.e. equalization of the concentration of some substance in a medium with an initially non-homogeneous distribution of this substance. The diffusion equation has the following form

$$c\frac{du}{dt} - \nabla \cdot (D\nabla u) = 0, \tag{29}$$

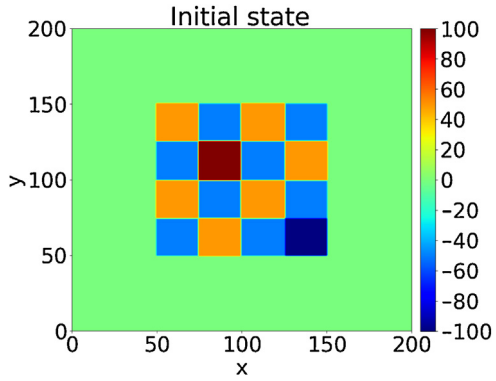


Fig. 3. Initial state ($t = 0$) of the diffusion example.

where $u(\mathbf{x}, t)$ is the unknown concentration of the substance, c is the porosity coefficient and D is the diffusion coefficient. In the following examples, we choose D and c to be equal to unity for simplicity. The initial condition $u(\mathbf{x}, 0) = u_0(\mathbf{x})$ specifies the concentration of the substance at the initial moment of time. In the example below, Equation (29) is discretized using the standard finite element method on a 2D mesh with 200×200 linear elements. Fig. 3 shows the initial state of our checkerboard-type test scenario, which has several contrasts between the "hot" and "cold" zones of the model.

Fig. 4 compares the full-scale solution with POD and DMD approximations that were constructed using 10 modes each. The number of time steps in the full simulation is 2000 and the first 100 of them are used to compute the ROM. The DMD-based approach shows better capabilities to reproduce the solution compared to

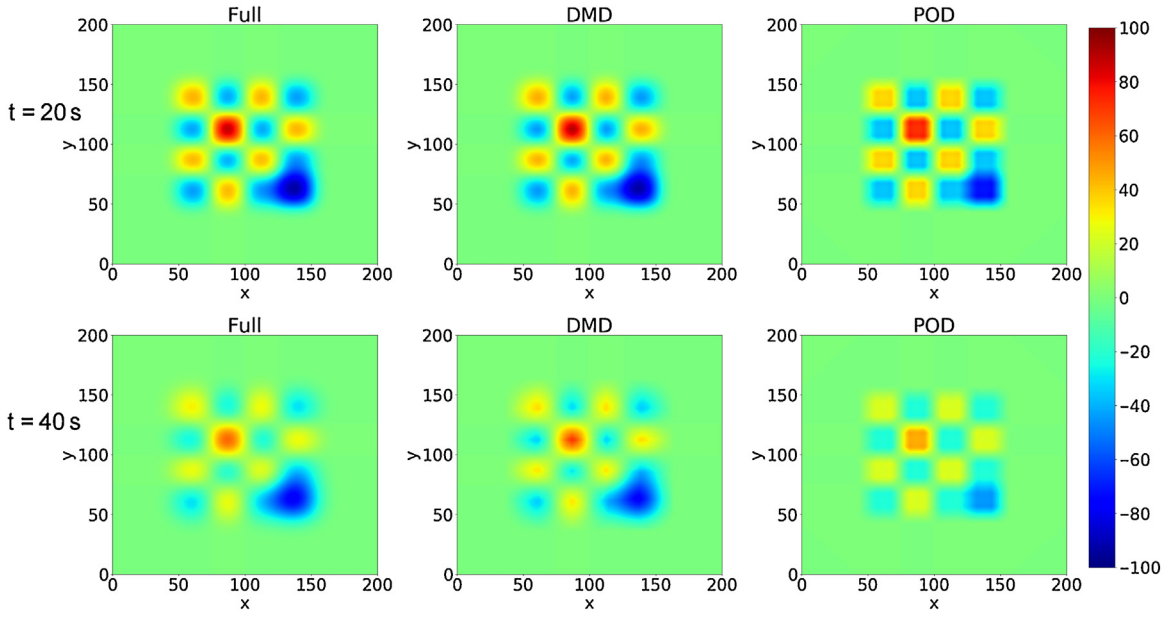


Fig. 4. The full-scale solution of the diffusion example (left column) versus its DMD (middle column) and POD (right column) approximations for $t = 20s$ (top row) and $t = 40s$ (bottom row).

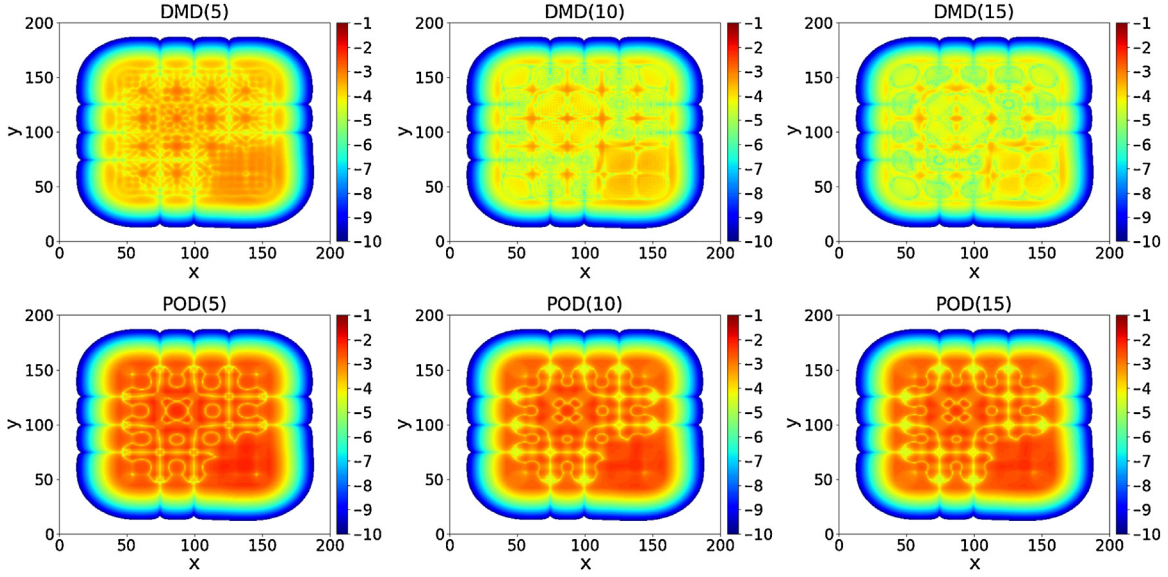


Fig. 5. L^2-norm relative errors for the DMD (top row) and POD (bottom row) approximations using 5, 10, and 15 modes.

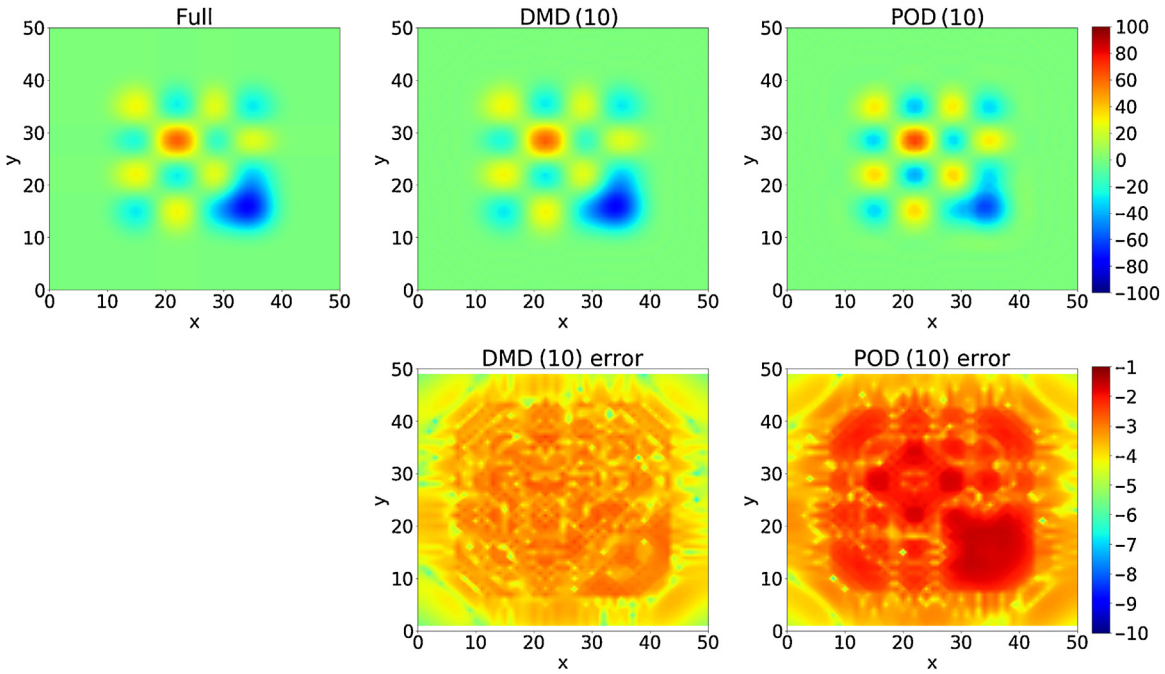


Fig. 6. Top row: the full-scale solution of the diffusion problem (left column) versus its DMD (middle column) and POD (right column) approximations for $t = 40$ s. Bottom row: L^2-norm relative errors of DMD (middle) and POD (right).

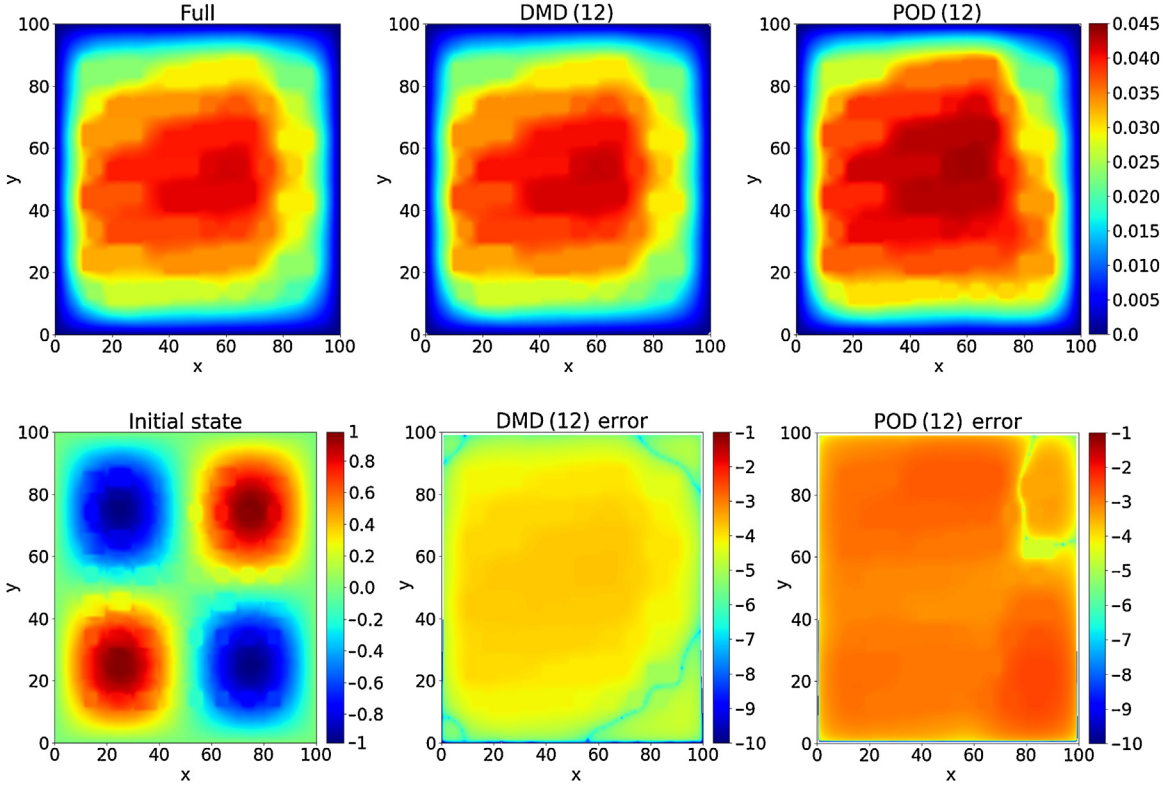


Fig. 7. Top row: the full-scale solution of the flow in porous media problem (left) versus its DMD (middle) and POD (right) approximations using 12 modes. Bottom row: initial state of the problem (left), L^2-norm relative errors of DMD (middle) and POD (right).

POD. The computational speedup of using POD and DMD compared to the full-scale model is 55x and 16x, respectively.

Fig. 5 shows the L^2-norm errors of the final solution between the full solution and its DMD and POD approximations for varying number of modes. DMD delivers significantly more accurate results compared to POD for any number of modes. This can be explained by the fact that DMD has an ability to extract the dynamic information and particularly the modes that govern the long-time dynamics of the process. In this example, sufficient accuracy can be achieved by using approximately 10–15 DMD modes. Using

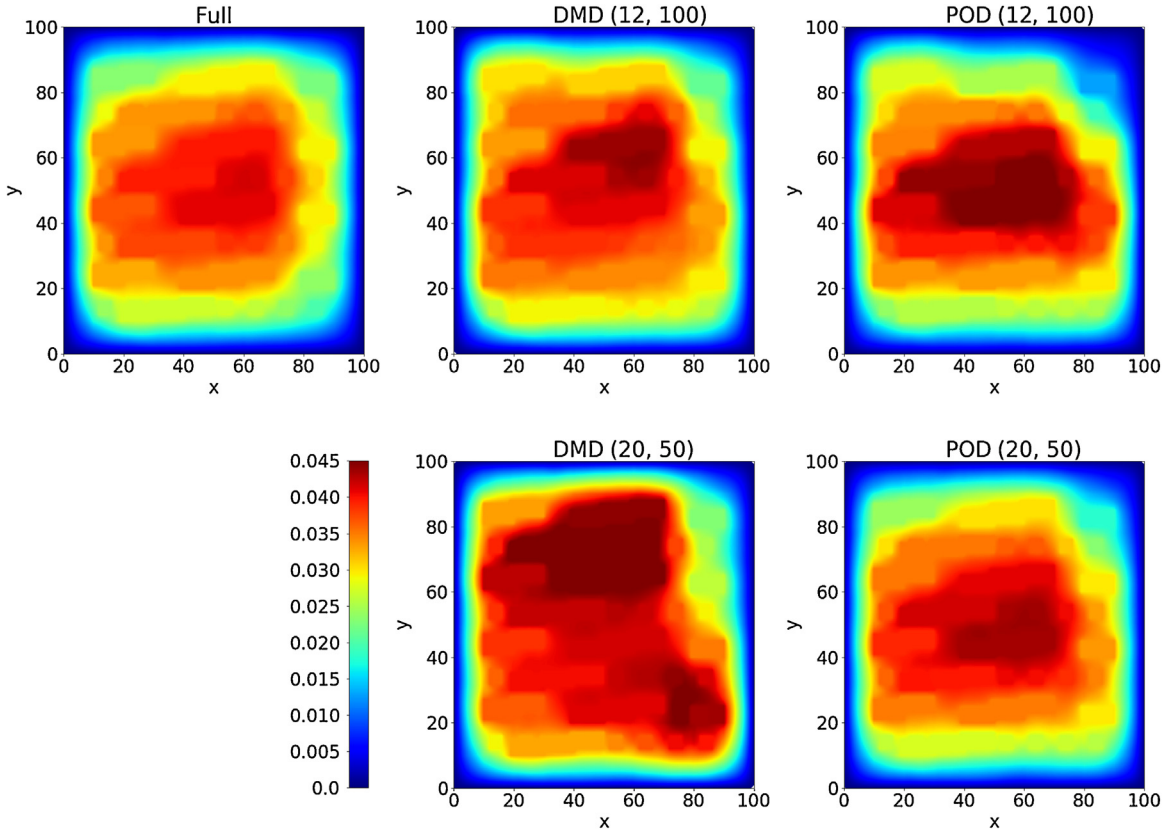


Fig. 8. Simulation results for smaller snapshot matrices: first 100 (top row) and first 50 (bottom row) time steps.

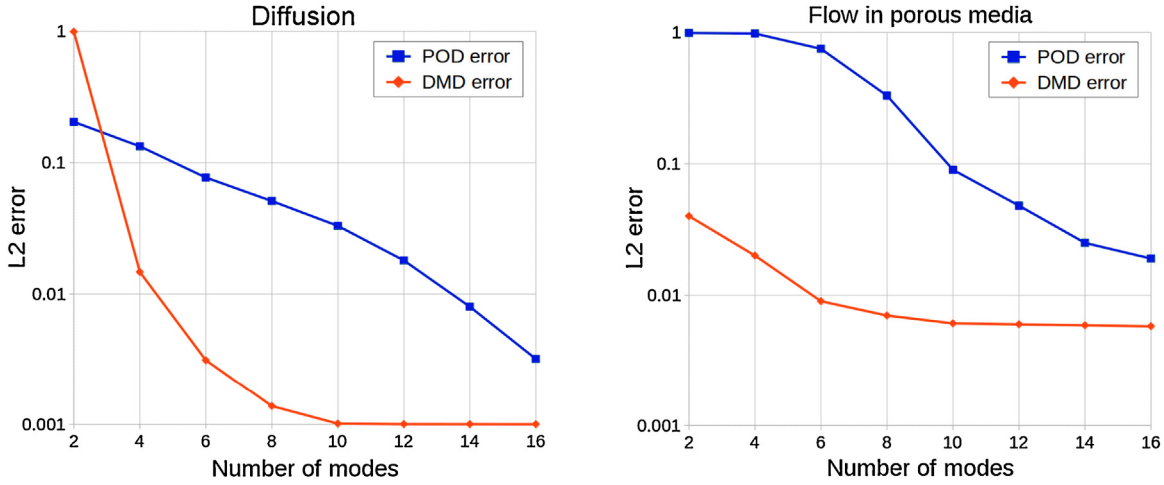


Fig. 9. L^2-norm errors of the POD and DMD solutions of the diffusion (left) and the flow in porous media (right) examples.

more modes does not increase the approximation quality (which is bounded by several factors, including the size of the snapshot matrix $\mathbb{Y}$ and dynamic characteristics of the physical process), but makes the matrix Φ close to singular.

In the example shown above, we use the linear finite element method for discretization of Equation (29). The choice of numerical method for spacial discretization may influence the behavior of ROM techniques. In the next example, we study POD- and DMD-based reduced order models for the same diffusion problem setup while using a different method, namely, the isogeometric analysis formulation [23] with quadratic B-spline basis functions. To improve the accuracy, we invoke the recently developed optimally-blending schemes [33] in the form of dispersion-minimizing quadrature rules [13]. This improves the convergence rate of the errors by two orders when compared against the standard isogeometric analysis. In the following example, we use a much coarser mesh of 50×50 elements and show that the accuracy is similar to the previous example, which has 16 times more grid cells.

Fig. 6 shows the full-scale isogeometric analysis solution versus its POD and DMD approximations using 10 modes. Compared to the results shown in Fig. 4, the POD approximation is of slightly better quality, though it is still less accurate than the DMD result. The computational speedup in this case is even higher and reaches about 200x and 60x for POD and DMD, respectively, when compared to the cost of the full-scale isogeometric solution. This is associated with the fact that the matrices resulting from quadratic

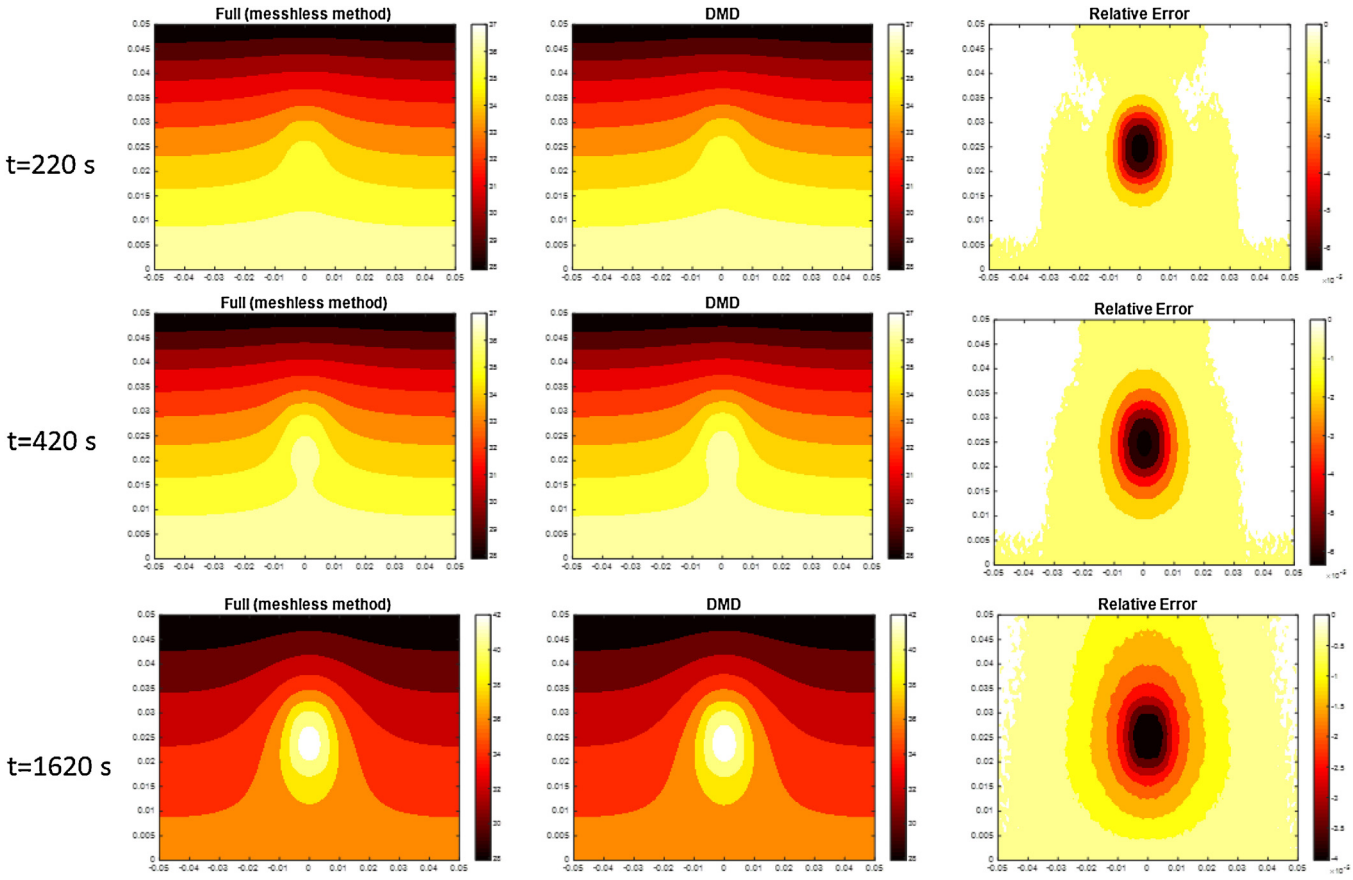


Fig. 10. Plots of the spatial temperature variations at $t = 220$ s (top row), $t = 420$ s (middle row), and $t = 1620$ s (bottom row) obtained from the full simulations using the meshless point collocation method and the DMD-based approximation.

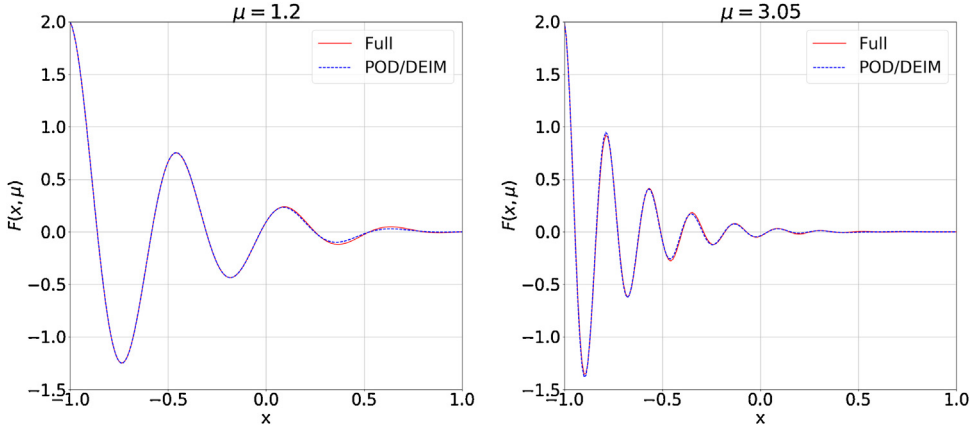


Fig. 11. POD/DEIM approximations using ten modes versus full solution of (33) for $\mu = 1.2$ (left) and $\mu = 3.05$ (right).

isogeometric discretization are denser than the ones obtained from linear finite elements.

4.1.2. Flow in porous media

In this second illustrative example, we consider a flow in porous media problem. The governing equation is discretized using the generalized multiscale finite element method (GMsFEM) developed in [19]. The single-phase fluid flow in porous media is described by the following equation:

$$\frac{du}{dt} - \nabla \cdot (\kappa(x) \nabla u) = f(x), \tag{30}$$

where $\kappa(x)$ represents the ratio of the permeability over the fluid viscosity and is a highly-heterogeneous parameter with high contrasts. Zero Dirichlet boundary conditions are assumed on the boundaries of the domain. Permeability fields contain inclusions of high and low conductivity ranging from $\kappa_{min} = 1$ to $\kappa_{max} = 1.9 \times 10^7$ in the randomly varying high-conductivity channels [19]. The permeability field is described on a fine 100×100 mesh. The problem is solved on a fine-coarse grid over a time interval of $T = [0, 0.2]$ seconds, which is long enough for the steady-state solution to be achieved. We use 2000 time steps of 10^{-4} seconds each.

We note that the PDE given by Eq. (30) is similar to the diffusion equation (29); the main differences between these examples

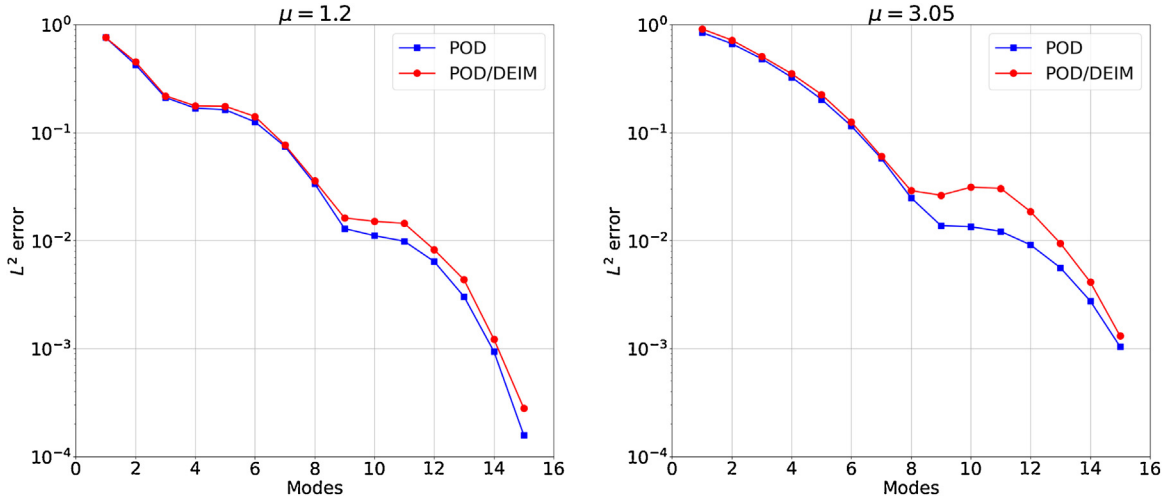


Fig. 12. L^2-norm errors of the POD and POD/DEIM approximations for $\mu = 1.2$ (left) and $\mu = 3.05$ (right).

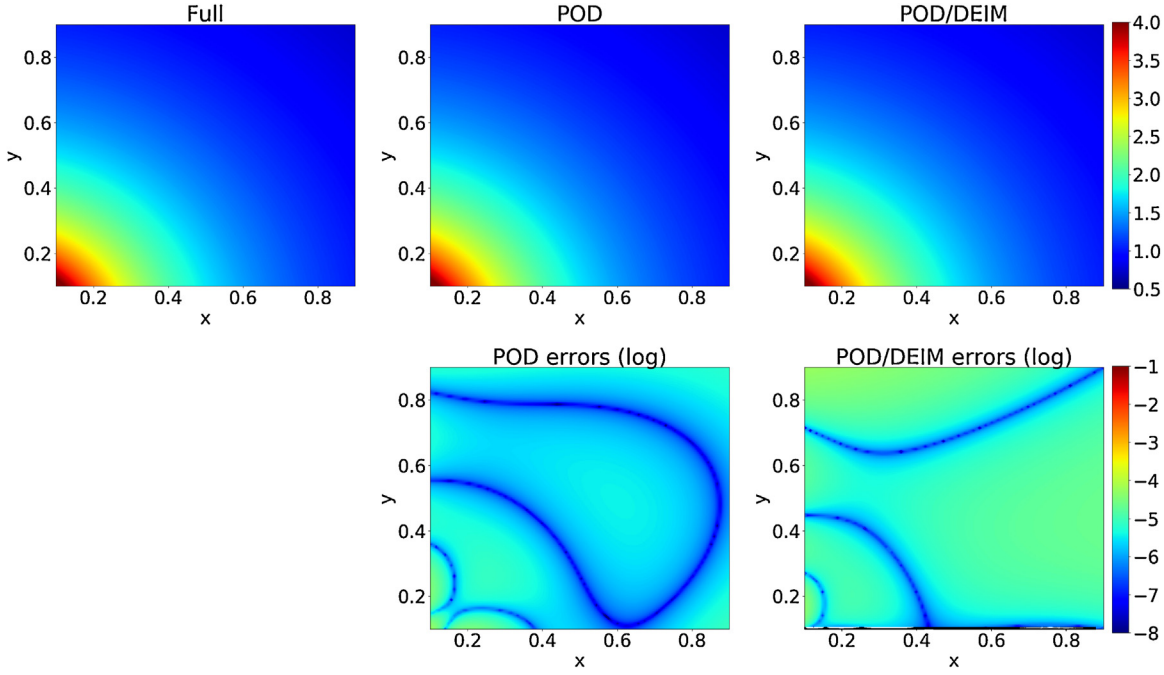


Fig. 13. Top row: the full-scale solution of (37) (left) versus its POD (middle) and POD/DEIM (right) approximations using 10 modes. Bottom row: L^2-norm relative errors of POD (middle) and POD/DEIM (right).

are the non-constant parameter $\kappa(x)$, the source term $f(x)$, different discretization methods (GMsFEM versus classical FEM) and model setup. The fine mesh of the GMsFEM yields a system of size 10201, while the local multiscale approach yields a system of size 850, an order of magnitude smaller than its fully-resolved counterpart. As we will see in the following examples, using DMD allows for another order-of-magnitude decrease in the computational cost.

Fig. 7 compares the full-scale solution of the flow in porous media problem with its POD and DMD approximations. The number of time steps in the full simulation is 2000; the first 200 of them are included in the snapshot matrix and used to compute the POD/DMD reduced solutions. 12 modes are used for both POD- and DMD-based reduced-order models to reach the final solutions shown in the top row of Fig. 7. Again, DMD clearly outperforms POD in terms of accuracy. The main computational gain of using POD and DMD reaches 21x and 11x reduction of the simulation time, respectively, compared to the full-scale model.

Fig. 8 shows the quality of POD and DMD approximations for smaller snapshot matrices, namely including only 100 or 50 first time steps. In the latter case, we also increased the number of modes to 20 as the use of 12 modes was clearly not sufficient in this case. Unlike the previous cases, POD performs better than DMD in this extreme scenario when the snapshot sequence is short and does not fully capture the associated dynamic features of the fluid flow. We note that both solutions are not completely reliable in this example when such small snapshot matrices are used (only 2.5–5% of the total simulation time).

To gain more insight on the capability of the reduced-order models implemented in pyROM based on the chosen ROM technique and the number of modes, we display in Fig. 9 the convergence of the L^2-norm errors of the final solution with increasing num-

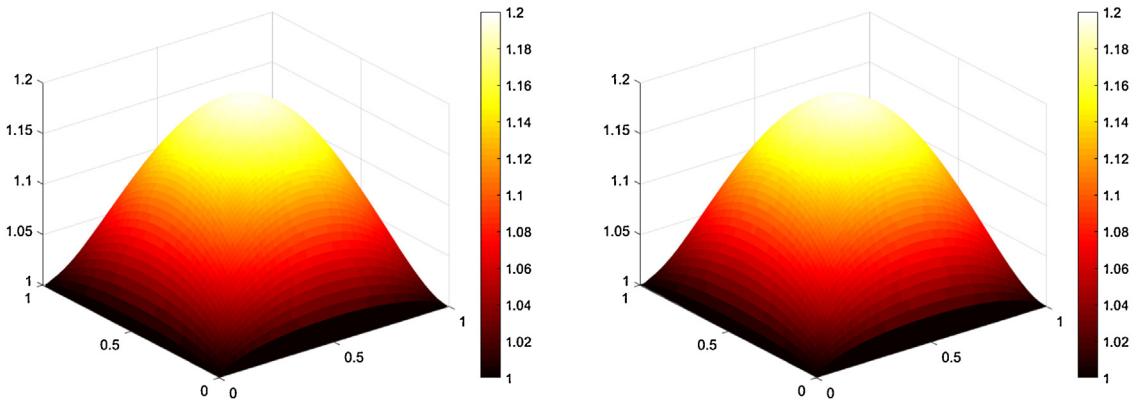


Fig. 14. Comparison between the fully-resolved steady-state temperature with that obtained from the combined POD-DEIM model reduction approach. 10 POD modes and 30 DEIM interpolation points are considered.

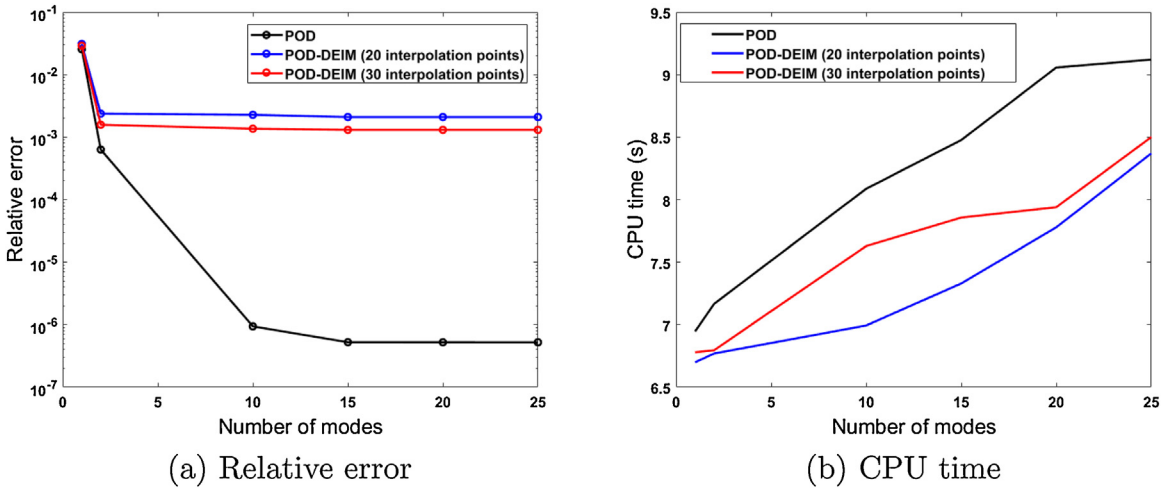


Fig. 15. Variations of the (a) relative error and (b) computational time for the POD- and coupled POD-DEIM based reduced-order models. The total time taken to run the full-order system is 1 min 50 s.

ber of modes in POD and DMD for both diffusion and flow in porous media problems. The snapshot matrices include the information of the first 5% and 10% of the total simulation time for these two problems, respectively, which is observed to be sufficient for capturing the main dynamic characteristics. DMD delivers significantly more accurate results compared to POD for both cases. Sufficient accuracy can be achieved by using as few as 10-12 modes. Increasing further the number of modes is observed not to enhance further the quality of the ROM approximation in these cases.

4.1.3. Bioheat problem: fast simulations of tumor removal

To illustrate the potential use of the computational tool for engineering applications, we consider the two-dimensional Pennes bioheat equation to simulate thermal ablation of localized tumors in body tissues. This equation incorporates the effects of blood perfusion and metabolic heat generation and is expressed as [8]:

$$\rho c \frac{\partial T(x,y,t)}{\partial t} = \nabla(\kappa \nabla T) + \omega_b \rho_b c_b (T_a - T) + Q_m + Q_r(x,y,t), \quad (31)$$

where ρ, c, and κ are the mass density, the specific heat capacity, and thermal conductivity of the body tissue, respectively. ω_b, ρ_b, and c_b denote the blood perfusion, density, and specific heat of blood, respectively. T_a is the arterial temperature and $T(x, y, t)$ is the local temperature of the body tissue. Q_m is the metabolic heat generation and $Q_r(x, y, t)$ is the rate of the external heating source.

We simulate a scenario of a medical practice of the tumor ablation via thermal therapy and show the usefulness of the DMD technique to produce fast and reliable prediction of the treatment progress. A rectangular domain of dimensions 0.1m $\times$ 0.05m comprises a tumor surrounded by a healthy tissue. We assume the following boundary conditions: fixed temperature at the upper side, convection at the lower side, and adiabatic surface sides. The tumor ablation is performed by activating a heat source targeting the tumor having a radius of 0.002 m and located in the middle of the spatial domain. Q_r is set equal to $100t$ Wm^{-3}. Q_m is equal to 4000 W/m^3 and 400 W/m^3 inside and outside the tumor domain, respectively. ω_b is equal to 0.001 kg/s/m^3 and 0.0001 kg/s/m^3 inside and outside the tumor domain, respectively. Typical physical properties of the tumor and healthy tissues are considered; that is, thermal conductivity κ = 0.5 Wm^{-1}K^{-1}, density ρ = 1052 kg·m^{-3}, specific heat capacity c = 3800 J kg^{-1} K^{-1}. At the convection domain boundary, we set the convective heat transfer coefficient h equal to 20 W m^{-2} K^{-1}, and the outer temperature T_f equal to 20°C. At the fixed temperature boundary, we set T_a equal to 28°C. We proceed with the numerical integration of the Pennes equation to evaluate the transient temperature variations of the body tissue when exposing the tumor domain to heat source using the meshless

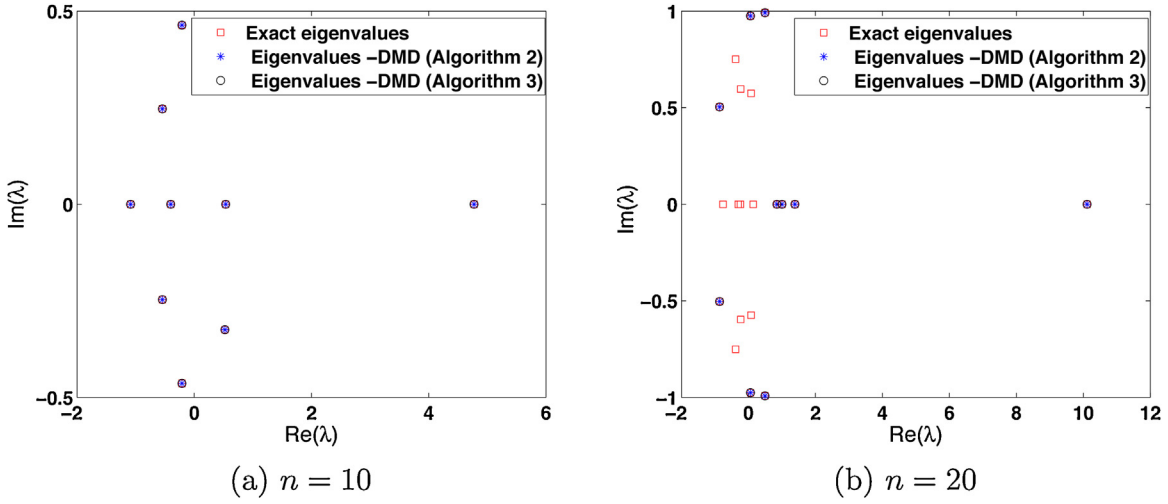


Fig. 16. Exact and DMD-approximated eigenvalues of matrix *A*.

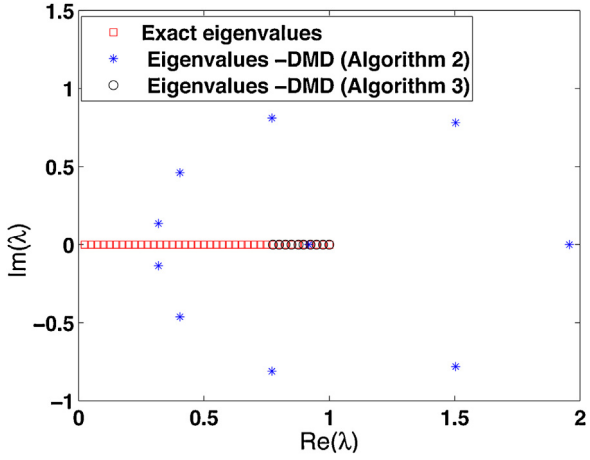


Fig. 17. Exact and DMD-approximated eigenvalues of matrix *A*.

point collocation method [26,8]. The meshless method is based on approximating the unknown field function (temperature field for the present example) and their space derivatives, by using a set of nodes, sprinkled over the spatial domain and functions with compact support. Meshless schemes bypass the use of different mesh configurations of the tumor and the surrounding healthy tissue and require only clouds of points, without any prior knowledge on their connectivity. This would relieve the computational burden associated with mesh generation.

As for the numerical simulation of the case study, the time step is set equal to $\Delta t = 20$ s and the total treatment time is taken 30 minutes resulting in 90 snapshots of the transient variations of the temperature field. The initial temperature distribution is obtained from the steady-state problem. We use the first 40 snapshots to compute the dynamic modes ϕ_i and their spectrum λ_i following the steps given in Algorithm 3 and then approximate the temperature field as

$$T(x, y, t^k) = T(x, y, k\Delta t) \approx \tilde{T}(x, y, k\Delta t) = \sum_{i=1}^{r} \lambda_i^k \phi_i(x, y), \tag{32}$$

where r is the number of modes.

In Fig. 10, we show the temperature variations obtained at different times from the meshless point collocation method and DMD-based approximation given by Eq. (32). Note that only 4 DMD modes are considered while 7381 nodes are used in the meshless method. A good agreement between the two sets of data is obtained. We note that the simulation time of the DMD-based approach is less than 5% of that when using the meshless method. This demonstrates the capability of the DMD modes to accurately detect the dynamics associated with the temperature evolution of the body tissue during the thermal therapy. Therefore, the DMD-based approximation can be deployed to predict the long-time behavior of the body tissue under thermal therapy by processing the first few snapshots that can be extracted from a sophisticated but computationally-demanding numerical tool or from medical imaging tools. This case study is implemented in pyROM to enable users to explore the potential use of the DMD-based technique for fast and reliable simulations of thermal therapies.

4.2. DEIM: illustrative examples

To demonstrate the applicability of DEIM in approximating nonlinear functions, we implement in pyROM two parameterized functions following the examples in [10].

4.2.1. A one-dimensional nonlinear function

First, we consider a one-dimensional nonlinear parameterized function $F : [-1, 1] \times [1, \pi] \mapsto \mathbb{R}$ given by

$$F(x; \mu) = (1 - x)\cos(3\pi\mu(x + 1))e^{-(1+x)\mu}, \tag{33}$$

where $x \in [-1, 1]$ and $\mu \in [1, \pi]$. Vector $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is formed of $n = 200$ equally-spaced points in $[-1, 1]$. For the sake of the subsequent analysis, we introduce the following vector $\mathbf{f}(\mu)$ that collects the function evaluations at the points x_i

$$\mathbf{f}(\mu) = (F(x_1; \mu), \ldots, F(x_n; \mu))^T \in \mathbb{R}^n. \tag{34}$$

To proceed with DEIM, we first construct the snapshot matrix obtained by evaluating the function $\mathbf{f}$ at equally-spaced values $\{\mu^\ell\}_{\ell=1}^{n_s}$ of μ in $[1, \pi]$

$$\begin{pmatrix} | & | & & | \\ \mathbf{f}(\mu^1) & \mathbf{f}(\mu^2) & \cdots & \mathbf{f}(\mu^{n_s}) \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n \times n_s}. \tag{35}$$

Next, we compute the first ten POD modes collected in the matrix $\mathbf{U}_f$. The number of snapshots of the nonlinear function is $n_s = 50$. We select the interpolation indices $\vec{\wp} = (\wp_1, \ldots, \wp_m)^T$ using **Algorithm 4** which enables the determination of the DEIM points only

at which the nonlinear function is evaluated. The approximate nonlinear function is given then by

$$\tilde{\mathbf{f}}(\mu) = \mathbf{U}_f(\mathbf{P}^T\mathbf{U}_f)^{-1}[F(x_{\wp_1};\mu),\ldots,F(x_{\wp_m};\mu)]^T \in \mathbb{R}^n, \tag{36}$$

where $\mathbf{P} = [\mathbf{e}_{\wp_1},\ldots,\mathbf{e}_{\wp_m}]$ and $\mathbf{e}_{\wp_i} = [0,\ldots,0,1,0,\ldots,0]^T \in \mathbb{R}^n$ is the $\wp_i$-th column of the identity matrix $\mathbf{I}_n$ for $i = 1,\ldots,m$. Fig. 11 compares the approximate function obtained from DEIM with $m = 10$ dimensions against the original system of dimension $n = 200$ at different values of μ. The good agreement observed between the two sets of data shows the capability of DEIM to approximate the nonlinear function using few selected points.

The relative error between the nonlinear function and its approximations is defined as $\|E\| = \|F - \tilde{F}\|/\|F\|$, where $\tilde{F}$ is obtained from DEIM approximation and POD L^2 projection as $\tilde{F} = \Phi(\Phi^T\Phi)^{-1}\Phi^T F$. In Fig. 12 we show the change in the relative error with the number of POD modes or DEIM dimension. As expected, POD-based approach (POD-Galerkin in terminology of [10]) yields slightly smaller errors since POD modes form an optimal subspace. However, the latter approach requires function evaluations at all grid points ($n = 200$), while the DEIM-based method requires only the evaluation of the nonlinear function at the interpolation points ($m = 10$) and thus performs much faster.

4.2.2. A two-dimensional nonlinear function

We consider a two-dimensional parameterized function $F : [0.1, 0.9]^2 \times [1,\pi]^2 \mapsto \mathbb{R}$ given by

$$F(x;y;\mu_1;\mu_2) = \frac{1}{\sqrt{(x-\mu_1)^2 + (y-\mu_2)^2 + 0.1^2}}, \tag{37}$$

where $(x, y) \in [0.1, 0.9]^2$ and $\mu = (\mu_1, \mu_2) \in [1,\pi]^2$. We let the vectors $\mathbf{x} = (x_1,\ldots,x_{n_x})^T \in \mathbb{R}^{n_x}$ and $\mathbf{y} = (y_1,\ldots,y_{n_y})^T \in \mathbb{R}^{n_y}$ formed of equally-spaced points in [0.1, 0.9] with $n_x = n_y = 100$. Vector $\mathbf{f}(\mu)$ contains the function evaluations at the grid points (x_i, y_j), namely

$$\mathbf{f}(\mu) = (F(x_1;y_1;\mu), F(x_2;y_1;\mu),\ldots,F(x_{n_x};y_1;\mu),\ldots,F(x_{n_x};y_{n_y};\mu))^T \in \mathbb{R}^n,$$

where $n = n_x \times n_y = 10000$. Similar to the 1D example, we construct the snapshot matrix from evaluations of $\mathbf{f}$ at uniformly selected control parameters $\{\mu^\ell\}_{\ell=1}^{n_s}$ of μ in $[1,\pi]^2$ and then compute its POD modes.

In Fig. 13, we compare the approximate functions obtained from DEIM and POD L^2 projection with $m = 10$ dimensions against the original function of dimension $n = 10000$ for $\mu = (\mu_1, \mu_2) = (-0.05, -0.05)$. As expected, projecting the nonlinear function onto the set of POD modes (optimal subspace) yields a good approximation of the nonlinear function. The L^2-norm errors are about one order of magnitude higher than those of the classical POD-Galerkin approximation but they are still negligible for practical use. This good agreement observed between the original function and its POD/DEIM approximation shows again the capability of this technique to reproduce the original nonlinear function using only a few selected points. The implementation of DEIM in pyROM enables users to approximate nonlinear functions with good accuracy while significantly reducing the simulation time.

4.2.3. A nonlinear unsteady problem: microwave heating of a square slab

As an example of a nonlinear unsteady problem, we consider the heating of a square slab using microwave energy. This heating process is governed by the following nonlinear forced heat equation

$$\begin{aligned} \nabla^2 u &= \frac{\partial u}{\partial t} - \beta u^\nu \quad \text{in} \quad \Omega, \\ u &= 1 \quad \text{on} \quad \partial\Omega, \\ u &= 1 \quad \text{at} \quad t = 0; \end{aligned} \tag{38}$$

where u denotes the temperature, $\Omega = [0;1] \times [0;1]$ is the spatial domain. The parameters introduced to model the microwave radiation through the material are given by $\beta = 2.8$ and $\nu = 3$. The meshless point collocation method [8] is used for the space discretisation of Equation (38) and an implicit θ-weighted scheme is employed for time marching. The time interval is taken as $T = [0;1.9]$ with a time step $\Delta t = 5 \times 10^{-2}$. This time interval is observed to be long enough so that the steady-state solution is achieved. The dimension of the full-order system is $n = 6561$. Furthermore, we develop a reduced-order based on projection-based method, as detailed in Section 2.1 and Section 2.2.3, where we combine POD and DEIM techniques. The result is a reduced-order model in the form of Eq. (6) where the nonlinear term is approximated via interpolation of a set of few selected points as given by Eq. (28). We note that the POD modes are computed from the first 30 snapshots obtained by solving the full-order system at equally-spaced time steps in the interval of [0;0.3] in which the temperature does not reach steady state yet.

In Fig. 14, we plot the steady-state temperature obtained from the full-order system and the reduced-order model based on 10 POD modes and 30 DEIM interpolation points used for the approximation of the nonlinear term. Clearly, the reduced system predicts accurately the temperature while reducing significantly the dimension of the problem.

To gain an insight on the trade-offs between the gain in the computational cost and solution accuracy, we plot the variations of the relative error and the simulation time in seconds for different numbers of POD modes and DEIM interpolation points in Fig. 15. The total time taken to run the full-order system is 1 min 50 s. Clearly, the coupled POD-DEIM model reduction approach yields a significant reduction in the computational cost while reproducing the solutions of the full-order system with a good accuracy. Furthermore, increasing the number of modes when constructing the reduced-order model yields smaller errors, but this comes at the price of an increase in the computational time. Employing the DEIM to approximate the nonlinear term in the reduced-order model enables faster simulations and still leads to a good prediction of the temperature distribution. For instance, a reduced system based on 10 POD modes and 30 DEIM interpolation points reproduces the steady-solution with a relative error of about 0.16% in 7.5 s; that is, 6.8% of the total amount of time required to run the full-scale simulation.

5. Conclusions

Reduced order modeling (ROM) methods are applied in a broad range of fields. They can extract dynamic information from time-varying problems without depending on the availability of a model, but rather from a sequence of snapshots. This work deals with the design and development of a fast and user-friendly implementation of ROM techniques with the capability to cover a wide range of engineering applications, namely pyROM. Users (such as, students, scientists, and application developers) have free access to the framework and can contribute to expand it further. They have the capability to apply model reduction techniques on their orig-

inal model to accelerate their simulations and be able to analyze large raw datasets.

pyROM is designed in an object-oriented way to make it easy to use and possible to extend with new ROM algorithms. The main principles of this framework are usability, ease of use, high extensibility and portability to different computer architectures, active use of standard Python libraries and visualization tools. The code is well documented and is freely available under an open-source license. Several examples describing different physical problems are included in pyROM, in particular, diffusion, flow in porous media, thermal ablation of tumor, nonlinear parameterized functions, and nonlinear microwave heating problem. Spatial discretization of the governing PDEs was performed using several modern methods including the finite element method, the isogeometric analysis, the meshless point collocation method, and the generalized multiscale finite element method. For all of these methods, POD, DMD, and DEIM showed their robustness on various numerical examples implemented in pyROM. DMD had higher accuracy compared to POD results in most of the cases studied.

Acknowledgments

The authors thank three anonymous reviewers for their valuable comments that helped to improve this paper. The authors V. Puzyrev and S. Meka acknowledge the support from the Institute for Geoscience Research (TIGeR), and the Curtin Institute for Computation. The author M. Ghommem would like to thank George Bourantas for his help in implementing some of the test problems reported in the paper.

Appendix A. Assessment of DMD implementations

To illustrate the capability of the DMD-based approach in approximating the eigenvalues of a linear operator defined by A, we consider a test case in which we pick a random matrix $A \in \mathbb{R}^{n\times n}$ and a random vector $\mathbf{y}_1 \in \mathbb{R}^n$ and construct a set of solutions given by Equation (14). We set n_s equal to 10 and n varies from 10 to 20. Fig. 16 shows the exact and approximate eigenvalues λ's of A computed using the DMD-based approaches as given in **Algorithms 2** and **3**. For $n = 10$, the DMD method reproduces exactly all the eigenvalues while for a larger system ($n = 20$) only the first n_s eigenvalues are predicted well by the DMD method. Clearly, DMD does a good job in computing the eigenvalues of A based on a set of snapshots without a prior knowledge of the exact form of A.

A more complex example shows the effectiveness of the second DMD approach (**Algorithm 3**) in computing the dynamic modes and their associated spectrum. We choose A as

$$A = B^T \begin{pmatrix} \frac{1}{n} & 0 & \cdots & 0 \\ 0 & \frac{2}{n} & 0 & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix} B, \quad \text{where} \quad B = \text{rand}(n, n), \tag{39}$$

n_s and n are chosen as 10 and 40, respectively. In Figure we plot 17 the exact and approximate eigenvalues of A. In this case, the eigenvalues predicted from the DMD-based approach using **Algorithm 2** deviate from the exact ones. On the other hand, **Algorithm 3** shows robustness in computing accurately the first n_s eigenvalues of A. Thus, we choose the second DMD approach for implementation in pyROM.

References

[1] I. Akhtar, A.H. Nayfeh, C.J. Ribbens, On the stability and extension of reduced-order Galerkin models in incompressible flows: A numerical study of vortex shedding, Theor. Comput. Fluid Dynam. 23 (2009) 213–237.
[2] I. Akhtar, Z. Wang, J. Borggaard, T. Iliescu, A new closure strategy for proper orthogonal decomposition reduced-order models, J. Comput. Nonlinear Dynam. 7 (2012) 034503.
[3] M. Alotaibi, V.M. Calo, Y. Efendiev, J. Galvis, M. Ghommem, Global-local nonlinear model reduction for flows in heterogeneous porous media, Comput. Methods Appl. Mech. Eng. 292 (2015) 122–137.
[4] H.P. Bakewell, J.L. Lumley, Viscous sublayer and adjacent wall region in turbulent pipe flow, Phys. Fluids 10 (1967) 1880–1889.
[5] B.A. Belson, J.H. Tu, C.W. Rowley, Algoritm 945: modred - a parallelized model reduction library, ACM Trans. Math. Softw. 40 (2014) 1–23.
[6] G. Berkooz, P. Holmes, J.L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annu. Rev. Fluid Mech. 53 (1993) 321–575.
[7] Bokeh Development Team, 2014. Bokeh: Python library for interactive visualization. URL: http://www.bokeh.pydata.org.
[8] G.C. Bourantas, M. Ghommem, G.C. Kagadis, K. Katsanos, V.C. Loukopoulos, V.N. Burganos, G.C. Nikiforidis, Real?time tumor ablation simulation based on the dynamic mode decomposition method, Med. Phys. 41 (2014) 053301.
[9] V.M. Calo, Y. Efendiev, J. Galvis, M. Ghommem, Multiscale empirical interpolation for solving nonlinear PDEs, J. Comput. Phys. 278 (2014) 204–220.
[10] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Scientific Comput. 32 (2010) 2737–2764.
[11] K.K. Chen, J.H. Tu, C.W. Rowley, Variants of dynamic mode decompositon: Boundary conditions, Koopman, and Fourier analysis, J. Nonlinear Sci. (2012), doi:10.1007/s00332-012-9130-9.
[12] A.E. Deane, I.G. Kevrekidis, G.E. Karniadakis, S.A. Orsag, Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinder, Phys. Fluids A 3 (1991) 2337–2354.
[13] Q. Deng, M. Bartoň, V. Puzyrev, V.M. Calo, Dispersion-minimizing quadrature rules for C1 quadratic isogeometric analysis, Comput. Methods Appl. Mech. Eng. 328 (2018) 554–564.
[14] D. Duke, J. Soria, D. Honnery, An error analysis of the dynamic mode decomposition, Exp. Fluids 52 (2012) 529–542.
[15] D. Fink, A. Wagner, W. Ehlers, Application-driven model reduction for the simulation of therapeutic infusion processes in multi-component brain tissue, J. Comput. Sci. 24 (2017) 101–115.
[16] M. Ghommem, I. Akhtar, M. Hajj, A low-dimensional tool for predicting force decomposition coefficients for varying inflow conditions, Progr. Comput. Fluid Dynam. 13 (2013) 368–381.
[17] M. Ghommem, V.M. Calo, Y. Efendiev, Mode decomposition methods for flows in high-contrast porous media. A global approach, J. Comput. Phys. 257 (2014) 400–413.
[18] M. Ghommem, E. Gildin, M. Ghasemi, Complexity reduction of multiphase flows in heterogeneous porous media, SPE J. 21 (2016) 144–151.
[19] M. Ghommem, M. Presho, V.M. Calo, Y. Efendiev, Mode decomposition methods for flows in high-contrast porous media. global-local approach, J. Comput. Phys. 253 (2015) 226–238.
[20] A. Hay, I. Akhtar, J.T. Borggaard, On the use of sensitivity analysis in model reduction to predict flows for varying inflow conditions, Int. J. Numer. Methods Fluids 68 (2012) 122–134.
[21] A. Hay, J. Borggaard, I. Akhtar, D. Pelletier, Reduced-order models for parameter dependent geometries based on shape sensitivity analysis, J. Comput. Phys. 229 (2010) 1327–1352.
[22] P. Holmes, J.L. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, Cambridge, UK, 1996.
[23] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Eng. 194 (2005) 4135–4195.
[24] Jones, E., Oliphant, T., Peterson, P., et al., 2001. SciPy: Open source scientific tools for Python URL: http://www.scipy.org/.
[25] D.J. Knezevic, J.W. Peterson, A high-performance parallel implementation of the certified reduced basis method, Comput. Methods Appl. Mech. Eng. 200 (2011) 1455–1466.
[26] G.R. Liu, Mesh Free Methods: Moving Beyond the Finite Element Method, CRC Press, Florida, 2002, pp. 792.
[27] F. Lusseyran, F. Gueniat, J. Basley, L. Douay, C. Pastur, L.R. Faure, T.M.P. Schmid, Flow coherent structures and frequency signature: application of the dynamic modes decomposition to open cavity flow, J. Phys.: Conf. Ser. 318 (2011) 042036.
[28] R. Milk, S. Rave, F. Schindler, pyMOR - generic algorithms and interfaces for model order reduction, SIAM J. Sci. Comput. 38 (2016) 194–216.
[29] Y. Mizuno, D. Duke, C. Atkinson, J. Soria, Investigation of wall-bounded turbulent flow using dynamic mode decomposition, J. Phys.: Conf. Ser. 318 (2011) 042040.
[30] M. Mordhorst, T. Strecker, D. Wirtz, T. Heidlauf, O. Röhrle, POD-DEIM reduction of computational EMG models, J. Comput. Sci. 19 (2017) 86–96.
[31] T.W. Muld, G. Efraimsson, D.S. Henningson, Flow structures around high-speed train extracted using proper orthogonal decomposition and dynamic mode decomposition, Comput. Struc. 57 (2012) 87–97.

[32] C. Pan, D. Yu, J. Wang, Dynamical mode decomposition of gurney flap wake flow, Theor. Appl. Mech. Lett. 1 (2011) 012002.
[33] V. Puzyrev, Q. Deng, V.M. Calo, Dispersion-optimized quadrature rules for isogeometric analysis: modified inner products, their dispersion properties, and optimally blended schemes, Comput. Methods Appl. Mech. Eng. 320 (2017) 421–443.
[34] P. Ramachandran, G. Varoquaux, Mayavi: 3D visualization of scientific data, IEEE Comput. Sci. Eng. 13 (2011) 40–51.
[35] O. San, T. Iliescu, Proper orthogonal decomposition closure models for fluid flows: Burgers equation, Int. J. Numer. Anal. Model Ser. 1 (2013) 1–18.
[36] P.J. Schmid, Dynamic mode decomposition of experimental data, 8th International Symposium on Particle Image Velocimetry - PIV09 (2009).
[37] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28.
[38] P.J. Schmid, Application of the dynamic mode decomposition to experimental data, Exp. Fluids 50 (2011) 1123–1130.
[39] P.J. Schmid, L. Li, M.P. Juniper, O. Pust, Applications of the dynamic mode decomposition, Theor. Comput. Fluid Dynam. 25 (2011) 249–259.
[40] P.J. Schmid, K.E. Meyer, O. Pust, Dynamic mode decomposition and proper orthogonal decomposition of flow in a lid-driven cylindrical cavity, in: 8th International Symposium on Particle Image Velocimetry - PIV09 (2009).
[41] A. Seena, H.J. Sung, Dynamic mode decomposition of turbulent cavity flows for self-sustained oscillations, Int. J. Heat Fluid Flow 32 (2011) 1098–1110.
[42] L. Sirovich, Turbulence and the dynamics of coherent structures, Quarterly of Applied Mathematics 45 (1987) 561–590.
[43] L. Sirovich, M. Kirby, Low-dimensional procedure for the characterization of human faces, J. Opt. Soc. Am. A 4 (1987) 524–529.
[44] A. Varga, Model reduction software in the SLICOT library. Applied and Computational Control, Signals, and Circuits, The Springer International Series in Engineering and Computer Science. 629 (2001) 239–282.
[45] Z. Wang, I. Akhtar, J. Borggaard, T. Iliescu, Two-level discretizations of nonlinear closure models for proper orthogonal decomposition, J. Comput. Phys. 230 (2011) 126–146.
[46] Z. Wang, I. Akhtar, J. Borggaard, T. Iliescu, Proper orthogonal decomposition closure models for turbulent flows, A numerical comparison. Computer Methods in Applied Mechanics and Engineering 237-240 (2012) 10–26.

Vladimir Puzyrev is Senior Research Fellow at Curtin University, Australia. Prior to taking this position, he worked as a postdoctoral and senior researcher at Barcelona Supercomputing Center. His research interests include numerical methods for PDEs, sparse linear solvers and preconditioners, high-performance computing, deep learning and their applications in geosciences. He has co-authored more than 20 refereed journal papers, and has given more than 40 talks in international conferences, workshops and seminars.

Mehdi Ghommem is Assistant Professor of Mechanical Engineering at American University of Sharjah, UAE (AUS) since August 2016. He earned his Ph.D. degree in Engineering Mechanics from Virginia Tech, USA in December 2011. Prior to joining AUS, he worked as research scientist with Schlumberger for three years. His research interests include nonlinear dynamics, model reduction of large dynamical/energy systems, modeling and simulation of MEMS, and flow in porous media. To date, he has published more than 40 refereed journal papers, presented his research outcomes in several international conferences and holds 4 US patent applications.

Shiv Meka is Senior HPC/AI Scientist at Curtin University, Australia. He received his Master degree in Materials Science and Engineering from Texas A&M University, College Station, USA, in 2009. He has since been involved with research relating computational materials science, quantum transport (especially transport in nanoscale junctions), and multiscale modeling.