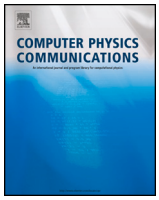
Computer Physics Communications 313 (2025) 109619

Contents lists available at ScienceDirect

Computer Physics Communications

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



Computational Physics

Distributed computing for physics-based data-driven reduced modeling at scale: Application to a rotating detonation rocket engine

Ionuţ-Gabriel Farcaş [a,b], Rayomand P. Gundevia [c], Ramakanth Munipalli [d], Karen E. Willcox [a,*]

[a] Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin TX, USA
[b] Department of Mathematics at Virginia Tech, Blacksburg VA, USA
[c] Amentum, Edwards Air Force Base, Edwards CA, USA
[d] Air Force Research Laboratory, Edwards Air Force Base, Edwards CA, USA

ARTICLE INFO

The review of this paper was arranged by Prof. Andrew Hazel

Keywords:
High-performance computing
Data-driven modeling
Scientific machine learning
Large-scale simulations
Rocket combustion

ABSTRACT

High-performance computing (HPC) has revolutionized our ability to perform detailed simulations of complex real-world processes. A prominent contemporary example is from aerospace propulsion, where HPC is used for rotating detonation rocket engine (RDRE) simulations in support of the design of next-generation rocket engines; however, these simulations take millions of core hours even on powerful supercomputers, which makes them impractical for engineering tasks like design exploration and risk assessment. Data-driven reduced-order models (ROMs) aim to address this limitation by constructing computationally cheap yet sufficiently accurate approximations that serve as surrogates for the high-fidelity model. This paper contributes a distributed memory algorithm that achieves fast and scalable construction of predictive physics-based ROMs trained from sparse datasets of extremely large state dimension. The algorithm learns structured physics-based ROMs that approximate the dynamical systems underlying those datasets. This enables model reduction for problems at a scale and complexity that exceeds the capabilities of standard, serial approaches. We demonstrate our algorithm's scalability using up to 2,048 cores on the Frontera supercomputer at the Texas Advanced Computing Center. We focus on a real-world three-dimensional RDRE for which one millisecond of simulated physical time requires one million core hours on a supercomputer. Using a training dataset of 2,536 snapshots each of state dimension 76 million, our distributed algorithm enables the construction of a predictive data-driven reduced model in just 13 seconds on 2,048 cores on Frontera.

1. Introduction

Scientists and engineers leverage data-driven reduced-order models (ROMs) for efficient predictions and decision-making across a wide range of applications underpinned by complex physical phenomena, such as design of next-generation propulsion devices, assessing the impact of turbulent transport in fusion devices, and real-time control of wind farms. The advent of petascale computing and, more recently, exascale-capable machines is revolutionizing our ability to conduct numerical simulations of complex real-world problems [1]. This has opened doors to previously unimaginable levels of realism, enabling predictive simulations with billions of degrees of freedom in fusion plasmas [10,14] or combustion processes [7,28], yet the dynamics in such simulations occur across multiple spatiotemporal scales and require substantial computational resources that can often exceed millions of core hours on supercomputers. The role of ROMs in enabling rapid yet accurate prediction and simulation-based design thus remains as important as ever; however, scalable methods to construct ROMs have not kept pace with the increased scale and resolution made possible by high-performance computing (HPC) in such applications.

Driven by increased data availability and computing power, various data-driven methods have emerged for scientific applications, including inferring models from data [15,42,48], data-driven discretizations for partial differential equations (PDEs) [5], and physics-informed machine learning methods [30], to name only a few. Moreover, a number of recent efforts were dedicated toward data-driven surrogate modeling [4]

* Corresponding author.
E-mail addresses: ionut.farcas@austin.utexas.edu (I.-G. Farcaş), rayomand.gundevia.ctr@afrl.af.mil (R.P. Gundevia), ramakanth.munipalli@us.af.mil (R. Munipalli), kwillcox@oden.utexas.edu (K.E. Willcox).

https://doi.org/10.1016/j.cpc.2025.109619
Received 13 July 2024; Received in revised form 29 March 2025; Accepted 9 April 2025
Available online 14 April 2025

and model reduction of nonlinear systems [3,11,17,23,25,26,29,40,41], including Operator Inference (OpInf) [32,42]. OpInf incorporates the structure of the governing equations into the learning problem and constructs structure-preserving ROMs for systems with polynomial structure from data. However, standard, serial ROM methods are unable to efficiently handle the large-scale training datasets generated by modern HPC simulations, where the state dimension numbers in the many millions (or even billions) and the training dataset—although sparse in time—consumes terabytes of memory. There is thus a need for novel ROM approaches based on the principles of modern computational science, namely scalable algorithms rooted in physical principles capable of running efficiently on powerful computing systems. The distributed memory (i.e., message-passing based) ROM algorithm proposed in this work aims to fulfill this need.

Algorithms and software tools for model reduction in large-scale applications have advanced considerably in recent years. A prominent class of model reduction methods is based on proper orthogonal decomposition (POD) [13]. Given the large computational cost of computing POD bases using large datasets, several works have focused on parallel POD approximations via either the singular value decomposition (SVD) [24] or the method of snapshots [51]. Beattie et al. [9] proposed an approximate iterative parallel algorithm and Wang, McBee, and Iliescu [56] formulated an approximate distributed-memory approach based on the method of snapshots for approximating the POD basis. The recent work [31] formulated a high-performance solver for computing partial SVD spectra. The randomized SVD [27] provides an alternative approach to approximating the POD basis for large datasets. Reference [47] employed the parallel QR-decomposition algorithm for tall-and-skinny data matrices proposed in [19] to parallelize Dynamic Mode Decomposition (DMD) [33,49,55]. Streaming approaches offer a complementary perspective by avoiding the need to fully load the data matrix into memory. Levy and Lindenbaum [34] proposed an incremental procedure for computing the POD from streaming data. This procedure was leveraged in Ref. [50], which formulated a streaming, greedy approach for nonlinear dimensionality reduction based on quadratic manifolds. Nonetheless, when the snapshot dimension is extremely large, performing a streaming approach without also using distributed memory computing will likely be infeasible on a single computer due to the significant memory requirements of loading and processing the data streams. Complementing methodological improvements, recent efforts have also developed software tools to facilitate efficient model reduction in large-scale applications. Examples include `libROM` [18], a `C++` library for data-driven approaches including POD, DMD, and projection-based and hyper-reduced intrusive ROMs, `Pressio` [45], an open-source project aimed at providing intrusive model reduction capabilities to large-scale application codes, `PyMOR` [39], a `Python` library for model order reduction algorithms such as POD, DMD, reduced basis methods, and system-theoretic methods for linear time-invariant systems, which also supports distributed memory parallelization, and the `PySPOD` library [46] for parallel spectral POD.

In this paper, we propose a distributed OpInf (dOpInf) algorithm that incorporates HPC into the data-driven learning process to enable rapid and scalable learning of structured, physics-based ROMs for problems at a scale and complexity that exceed the capabilities of standard serial processing approaches for model reduction. In contrast to existing parallel intrusive methods that require some level of access to the high-fidelity code or approaches that parallelize only specific stages of the reduced modeling process (such as approximating the POD basis), our formulation represents a fully distributed memory workflow for nonlinear, physics-based model reduction of complex applications. The proposed model reduction workflow enables (1) efficient data transformations and dimensionality reduction of large datasets with extremely large state dimension without explicitly having to compute the POD basis and without introducing approximations, and (2) the non-intrusive learning of structured, physics-based ROMs that approximate the dynamics underlying those datasets. We note that the elements of our distributed algorithm are transferable to other data-driven reduced modeling approaches such as DMD and quadratic manifolds [6,23], and to parametric ROMs that embed parametric dependence using the strategies surveyed in [12]. The scalability and prediction capabilities of the proposed dOpInf algorithm are assessed in a real-world three-dimensional unsteady rotating detonation rocket engine (RDRE) scenario with 76 million degrees of freedom. The training dataset amounts to 1.4 TB, which prohibits using standard serial methods for model reduction. In addition to contributing the dOpInf algorithm, our work also represents a novel contribution to model reduction for RDREs, addressing unprecedented levels of complexity in terms of data size and physical coupling. We demonstrate the scalability of our algorithm using up to $2,048$ cores on the Frontera supercomputer at the Texas Advanced Computing Center (TACC) [53]. On this system, our method constructs a predictive physics-based ROM from the large dataset in just 13 seconds. This is in contrast to many existing data-driven reduced and surrogate modeling approaches that can require significant processing times of large datasets and a large snapshot dimension. The resulting ROM is $90,000$ times faster to evaluate than the original high-fidelity simulation, paving the way toward exploring ROM-based design and quantification of uncertainty. An implementation, including the detailed tutorial from Ref. [21], can be found at https://github.com/ionutfarcas/distributed_Operator_Inference.

The remainder of this paper is organized as follows. Section 2 summarizes the general setup for high-fidelity nonlinear simulations. Section 3 details the proposed dOpInf algorithm that enables fast and scalable learning of physics-based ROMs for complex applications with training datasets with extremely large state dimension. We present the scalability results and prediction capabilities of the proposed distributed algorithm in a large-scale RDRE scenario in Sec. 4. Section 5 concludes the paper.

2. Setup for high-fidelity nonlinear physics-based simulations

Consider a complex physical process whose temporal dynamics over $[t_{\text{init}}, t_{\text{final}}]$ are described by the high-dimensional dynamical system

$$\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}), \quad \mathbf{s}(t_{\text{init}}) = \mathbf{s}_{\text{init}}. \tag{1}$$

Here, $\mathbf{s}(t) \in \mathbb{R}^n$ denotes the spatially discretized vector of physical state variables at time $t \in [t_{\text{init}}, t_{\text{final}}]$, $n = n_x n_s \in \mathbb{N}$ is the dimension of the discretized state space, where $n_x \in \mathbb{N}$ denotes the number of degrees of freedom used to discretize the underlying physical domain and $n_s \in \mathbb{N}$ denotes the number of physical state variables (e.g., pressure, velocity, chemical concentrations etc. in RDRE simulations), $\mathbf{s}_{\text{init}}$ is a specified initial condition, and $\mathbf{f} : [t_{\text{init}}, t_{\text{final}}] \times \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear function that defines the time evolution of $\mathbf{s}$.

Our starting point is a given set of training data that represent observations of the state of the system, where the training data might be generated by experiments or by simulation, or by a combination of the two. In many cases, and in the real-world RDRE scenario considered in this paper, the data come from a simulation that solves the physical equations governing the spatiotemporal evolution of the system state. These governing equations take the form of PDEs encoding physical conservation laws and constitutive relationships. When discretized in space, these equations are written as an n-dimensional system of ordinary differential equations (ODEs) as in (1), representing the system of discretized PDEs over the computational domain denoted by $\Omega \subset \mathbb{R}^d$, where typically $d = 2, 3$. In such cases, n is large (e.g., in $O(10^6) - O(10^9)$), because it scales with the dimension of the PDE spatial discretization, n_x.

We focus on ROMs utilized for predictions over a time horizon $[t_{\text{init}}, t_{\text{final}}]$, with t_{init} denoting the initial time and t_{final} denoting the final time. The training dataset comprises $n_t \in \mathbb{N}$ large-scale state vectors or snapshots at n_t time instants over a training horizon $[t_{\text{init}}, t_{\text{train}}]$ with $t_{\text{train}} < t_{\text{final}}$. The state solution at time instant $t_k \in [t_{\text{init}}, t_{\text{train}}]$ is referred to as the kth snapshot and is denoted by $\mathbf{s}_k$. The n_t snapshots are

Algorithm 1 dOpInf data transformations and dimensionality reduction on p compute cores.

Input: $p \geq 2$, n, n_t, r, *transform*
Output: projected transformed snapshots $\hat{\mathbf{Q}} \in \mathbb{R}^{r \times n_t}$

1: *divide* the snapshot dimension n into p parts $n_1, n_2, \ldots, n_p$ such that $\sum_{i=1}^{p} n_i = n$
2: **for** $i \leftarrow 1$ to p in parallel **do**
3: *load* snapshot components $\mathbf{S}_i \in \mathbb{R}^{n_i \times n_t}$ from disk
4: **if** *transform* **then**
5: apply *data transformations* to $\mathbf{S}_i$ such as lifting, centering, and scaling, to obtain $\mathbf{Q}_i \in \mathbb{R}^{m_i \times n_t}$ such that $\sum_{j=1}^{p} m_j = m \geq n$
6: **else**
7: continue with $\mathbf{Q}_i = \mathbf{S}_i$ and set $m_i = n_i$
8: *compute* $\mathbf{D}_i = \mathbf{Q}_i^{\top}\mathbf{Q}_i \in \mathbb{R}^{n_t \times n_t}$
9: *compute* and *broadcast* $\mathbf{D} = \sum_{j=1}^{p} \mathbf{D}_j$ via a collective reduction
10: *compute* the first r eigenpairs $\{(\lambda_k, \mathbf{u}_k)\}_{k=1}^{r}$ of $\mathbf{D}$ and arrange them s. t. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$
11: *compute* the corresponding global POD singular values $\sigma_k = \sqrt{\lambda_k}$ for $k = 1, 2, \ldots, r$
12: *compute* $\mathbf{T}_r = \mathbf{U}_r \boldsymbol{\Lambda}_r^{-\frac{1}{2}} \in \mathbb{R}^{n_t \times r}$, where $\mathbf{U}_r = \left[\mathbf{u}_1 | \mathbf{u}_2 | \ldots | \mathbf{u}_r\right]$ and $\boldsymbol{\Lambda}_r = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$
13: *compute* the global representation of the transformed data in the low-dimensional space spanned by the POD basis vectors as $\hat{\mathbf{Q}} = \mathbf{T}_r^{\top}\mathbf{D} \in \mathbb{R}^{r \times n_t}$

collected into a large-scale snapshot matrix $\mathbf{S} \in \mathbb{R}^{n \times n_t}$ with $\mathbf{s}_k$ as its kth column:

$$\mathbf{S} = \begin{bmatrix} | & | & & | & & | \\ \mathbf{s}_1 & \mathbf{s}_2 & \ldots & \mathbf{s}_k & \ldots & \mathbf{s}_{n_t} \\ | & | & & | & & | \end{bmatrix}.$$

In large-scale applications, n_x (and therefore $n = n_s n_x$) is orders of magnitude larger than n_t. In other words, the row dimension of $\mathbf{S}$ is significantly larger than its column dimension. Given $\mathbf{S}$ and the governing equations (1), our goal is to efficiently and scalably learn a predictive data-driven ROM over the target time horizon $[t_{\text{init}}, t_{\text{final}}]$. However, doing so via standard serial methods would be computationally expensive and limited by memory storage and bandwidth. Our proposed distributed workflow, in contrast, enables a fast and scalable workflow for constructing physics-based ROMs on distributed memory machines.

3. dOpInf: a new distributed computing algorithm for fast and scalable learning of nonlinear physics-based reduced models

This section presents the proposed distributed Operator Inference (dOpInf) algorithm. dOpInf represents a distributed memory workflow for physics-based model reduction of complex applications with extremely large state dimension. Section 3.1 presents the first step in dOpInf focused on distributed data manipulations and dimensionality reduction, followed by the distributed learning of the reduced model operators in Sec. 3.2, and the distributed postprocessing of the reduced solution in Sec. 3.3. We provide a summary in Sec. 3.4.

3.1. Distributed computation of data transformations and dimensionality reduction

The OpInf methodology requires the generation of a low-dimensional representation of the snapshot data. These low-dimensional data are then used to infer the reduced model operators that define the physics-based ROM. Generating the low-dimensional snapshot representation entails two key steps: data transformations and dimensionality reduction. Data transformations are used to improve numerical conditioning through centering and scaling, and to expose polynomial structure in nonlinear problems [44]. Dimensionality reduction represents the (transformed) snapshot data in a subspace of reduced dimension $r \ll n$. The proposed dOpInf approach introduces a formulation that ensures a scalable distributed computation for each of these steps. In particular, by starting from the POD method of snapshots introduced in the seminal work [51] but now typically replaced in POD implementations by the SVD, we formulate a more scalable algorithm for achieving dimensionality reduction without necessitating the computation of the POD basis.

Algorithm 1 details our distributed memory formulation of data transformations and dimensionality reduction. The inputs are the number of compute cores $p \in \mathbb{N}_{\geq 2}$, the high-dimensional snapshot dimension n, the number of training snapshots n_t, the reduced dimension of the target ROM $r \in \mathbb{N}$, and a variable *transform* that indicates whether data transformations are employed. In the following, we denote by $i = 1, 2, \ldots, p$ the index of the ith core.

The first step is to distribute the snapshot data into p non-overlapping row blocks $\mathbf{S}_i \in \mathbb{R}^{n_i \times n_t}$ with one block per core such that $\sum_{i=1}^{p} n_i = n$. The splitting strategy is user-defined and depends on factors such as the target parallel architecture and necessary data transformations, as we will discuss below. A straightforward approach suitable for homogeneous architectures is to use blocks of size $n_i = \lfloor n/p \rfloor$; when p does not divide n exactly, the remaining $n - \lfloor n/p \rfloor$ rows can be further distributed among the p compute cores. Loading $\mathbf{S}_i$ into the memory of the ith core requires $\mathcal{O}(n_i n_t)$ bytes.

Remark 1. To ensure scalable access to the training dataset across p compute cores, large datasets should be stored in a format like HDF5 or NetCDF, which supports efficient parallel I/O operations, and on high throughput partitions such as the `scratch` partition on a supercomputer. A comprehensive understanding of the underlying file system is also crucial for maximizing performance. We note that saving a large-scale dataset in a single file can hinder scalability when an increasing number of cores attempt to access it simultaneously. This can be mitigated by file partitioning (i.e., the dataset is divided into multiple files, allowing scalable parallel reading operations across different compute cores) or distributed reading and broadcasting (i.e., one core reads the data and broadcasts it to all other cores), keeping in mind that this strategy might be more time consuming than file partitioning for very large datasets.

The application of data transformations is determined by the input *transform*. While often glossed over in the literature, most complex applications of dimensionality reduction via POD require centering and scaling of the snapshot data. The importance of centering and scaling for obtaining accurate OpInf ROMs is particularly observed in problems with multiple physical state variables (e.g., pressure, velocity, chemical concentrations in RDRE simulations), each with differing physical scales [20,43,54]. One strategy, which we also employ in the RDRE scenario under consideration, is to center each discretized state variable in $\mathbf{S}_i$ by its mean over the training time horizon and then scale it by its maximum absolute value to ensure that the scaled variables do not exceed $[-1, 1]$. To efficiently compute centered and scaled snapshots, we distribute the full snapshot dataset over the full computational domain Ω into p non-overlapping subdomains $\Omega_1, \Omega_2, \ldots, \Omega_p \in \mathbb{R}^d$ satisfying $\Omega = \cup_{i=1}^{p} \Omega_i$ such that each compute core gets all discrete state variables for a subdomain Ω_i. Note that this splitting strategy aligns with tradi-

tional non-overlapping domain decomposition methods. This allows for independent centering calculations on each core, eliminating communication needs. Scaling parameters, which are typically global across the training horizon, can be computed locally on each core followed by an inexpensive collective communication step to compute the global results. In general, the exact computational costs of centering and scaling are transformation dependent, but they are usually low compared to the remaining preprocessing costs. In addition, they can be performed inplace on each $\mathbf{S}_i$, requiring no additional memory. The OpInf approach might also employ lifting and other variable transformations [44]. By exploiting the knowledge of a system's governing equations, these transformations seek to find a new coordinate system where the system dynamics exhibit a polynomial structure. One example used in computational fluid dynamics involves expressing the governing equations in terms of specific volume variables instead of the conventional conservative variables, resulting in a quadratic structure in the transformed governing equations. In other examples, the transformations might involve augmenting the system's physical state with additional auxiliary variables, thus lifting the physical variables to a higher dimensional coordinate system. Lifting and other variable transformations are applied entry-wise in each $\mathbf{S}_i$ and do not generally require communication.

If data transformations are necessary, the transformed snapshot matrices for each compute core are denoted as $\mathbf{Q}_i \in \mathbb{R}^{m_i \times n_t}$, where m_i denotes the dimensions of the transformed ith snapshot partition, such that $\sum_{i=1}^{p} m_i = m \geq n$; m exceeds n when employing lifting transformations that introduce auxiliary state variables. In this case, the number of lifted (or transformed) state variables, $m_s \in \mathbb{N}$, exceeds the original number of physical state variables, n_s. The original partitions $\mathbf{S}_i$ can be discarded from memory, and $\mathbf{Q}_i$ are used for the ensuing calculations. If data transformations are not required, we set $\mathbf{Q}_i = \mathbf{S}_i$ and $m_i = n_i$. Under these circumstances, any snapshot partitioning scheme suffices provided that the entire dataset is non-overlappingly distributed among the p cores.

We next compute the representation of the transformed snapshots in the low-dimensional subspace spanned by the rank-r POD basis vectors. We start from the method of snapshots [51], as this provides several computational and memory advantages in our context. We note that the sequential version of the method of snapshots offers, in principle, a path to handle large datasets by processing subsets of at least two snapshots at a time. However, its sequential nature and potential for redundancy (e.g., reloading the full dataset for scaling) limit its efficiency.

We compute the Gram matrices $\mathbf{D}_i = \mathbf{Q}_i^\top \mathbf{Q}_i \in \mathbb{R}^{n_t \times n_t}$ on each compute core, which involves a dense matrix-matrix multiplication requiring $\mathcal{O}(m_i n_t^2)$ operations and $\mathcal{O}(n_t^2)$ bytes of main memory. We then compute their summation $\mathbf{D} = \sum_{i=1}^{p} \mathbf{D}_i$ via a collective parallel reduction that ensures that all cores have $\mathbf{D}$ in their local memory, which requires $\mathcal{O}(n_t^2)$ computations and $\mathcal{O}(n_t^2)$ additional bytes of memory on each core. Since the full snapshot data was distributed non-overlappingly, the following lemma shows that $\mathbf{D}$ is equal to the global Gram matrix $\mathbf{Q}^\top \mathbf{Q}$.

Lemma 1. *Let $\mathbf{A} \in \mathbb{R}^{q \times k}$ be a matrix and let $\mathbf{A}_i \in \mathbb{R}^{q_i \times k}$ with $i = 1, 2, \ldots, p$ be p non-overlapping row blocks such that $\sum_{i=1}^{p} q_i = q$. Then, $\mathbf{A}^\top \mathbf{A} = \sum_{i=1}^{p} \mathbf{A}_i^\top \mathbf{A}_i$.*

Proof. Let $\mathbf{P}_i \in \mathbb{R}^{q \times q_i}$ be prolongation matrices with entries $p_{i,j} \in \{0, 1\}$ such that $\mathbf{A} = \sum_{i=1}^{p} \mathbf{P}_i \mathbf{A}_i$. The conclusion follows from the fact that $\mathbf{P}_i^\top \mathbf{P}_i = \mathbf{I}_{q_i \times q_i}$ and, since the row blocks are non-overlapping, $\mathbf{P}_j^\top \mathbf{P}_i = \mathbf{0}_{q_j \times q_i}$, $\forall j \neq i$. □

Each core proceeds by computing the partial eigendecomposition $\{(\lambda_k, \mathbf{u}_k)\}_{k=1}^{r}$ of the symmetric positive semi-definite matrix $\mathbf{D}$, where λ_k are the real and non-negative eigenvalues, and $\mathbf{u}_k \in \mathbb{R}^{n_t}$ denote the corresponding eigenvectors. The computational cost of this decomposition depends on the value of r and the employed approach. If r is small, efficient methods such as those based on the Lanczos iterative algorithm [24] can be employed. The r eigenpairs necessitate $\mathcal{O}(rn_t)$ bytes of memory on each core. We must ensure that they are arranged such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$.

Consider the thin SVD [24] of $\mathbf{Q} = \mathbf{V}\mathbf{\Sigma}\mathbf{W}^\top$. $\mathbf{V} \in \mathbb{R}^{m \times n_t}$ contains the left singular vectors, $\mathbf{\Sigma} \in \mathbb{R}^{n_t \times n_t}$ is a diagonal matrix containing the singular values of $\mathbf{Q}$ in non-decreasing order $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{n_t}$, where σ_j denotes the jth singular value, and $\mathbf{W} \in \mathbb{R}^{n_t \times n_t}$ contains the right singular vectors. $\mathbf{V}$ and $\mathbf{W}$ are semi-orthogonal, that is, $\mathbf{V}^\top \mathbf{V} = \mathbf{W}^\top \mathbf{W} = \mathbf{I}_{n_t}$, where $\mathbf{I}_{n_t} \in \mathbb{R}^{n_t \times n_t}$ denotes the identity matrix. Since

$$\mathbf{D} = \mathbf{Q}^\top \mathbf{Q} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^\top \mathbf{V}\mathbf{\Sigma}\mathbf{W}^\top = \mathbf{W}\mathbf{\Sigma}^2\mathbf{W}^\top,$$

it follows that $\mathbf{D}\mathbf{W} = \mathbf{W}\mathbf{\Sigma}^2$, where we used the semi-orthogonality of $\mathbf{W}$. This implies that the eigenvalues of $\mathbf{D}$ are the squared singular values of $\mathbf{Q}$ and the eigenvectors of $\mathbf{D}$ are equivalent (in terms of spanned subspaces) to the right singular vectors of $\mathbf{Q}$. Hence, the first r global POD singular values can be computed as $\sigma_k = \sqrt{\lambda_k}$, $\quad k = 1, 2, \ldots, r$, which requires $\mathcal{O}(r)$ operations. Furthermore, from the thin SVD of $\mathbf{Q}$ we have that $\mathbf{V} = \mathbf{Q}\mathbf{W}\mathbf{\Sigma}^{-1}$, which means that the left singular vectors in $\mathbf{V}$ can be equivalently expressed as

$$\mathbf{V} = \mathbf{Q}\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}, \tag{2}$$

where $\mathbf{U} = \left[\mathbf{u}_1 | \mathbf{u}_2 | \ldots | \mathbf{u}_{n_t}\right]$ and $\mathbf{\Lambda}_r = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{n_t})$.

The user can also choose r during the execution of Algorithm 1 based on an energy criterion such as

$$\frac{\sum_{k=1}^{r} \sigma_k^2}{\sum_{k=1}^{n_t} \sigma_k^2} \geq E_{\%}, \tag{3}$$

where $E_{\%}$ is a user-specified energy truncation threshold (e.g., $E_{\%} = 95\%$ or $E_{\%} = 99\%$). In this case, we must compute all eigenpairs of $\mathbf{D}$, arrange them such that the eigenvalues are in increasing order, and choose r based on (3). The first r eigenpairs are then used in the next steps in Algorithm 1. Computing all eigenpairs of a symmetric positive definite matrix requires $\mathcal{O}(n_t^3)$ operations and $\mathcal{O}(n_t^2)$ bytes of memory for the n_t eigenpairs.

In the next step (step 12), each core computes $\mathbf{T}_r = \mathbf{U}_r \mathbf{\Lambda}_r^{-\frac{1}{2}} \in \mathbb{R}^{n_t \times r}$, where $\mathbf{U}_r = \left[\mathbf{u}_1 | \mathbf{u}_2 | \ldots | \mathbf{u}_r\right]$ and $\mathbf{\Lambda}_r = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$, which requires $\mathcal{O}(rn_t)$ computations and $\mathcal{O}(rn_t)$ bytes of memory for the result. At this point we have all the ingredients to compute the global representation of the transformed snapshots in the low-dimensional subspace spanned by the rank-r POD basis vectors without explicitly requiring the basis. The standard, sequential approach to compute the low-dimensional representation $\hat{\mathbf{Q}} \in \mathbb{R}^{r \times n_t}$ of the high-dimensional transformed data in OpInf is

$$\hat{\mathbf{Q}} = \mathbf{V}_r^\top \mathbf{Q}, \tag{4}$$

where $\mathbf{V}_r \in \mathbb{R}^{m \times r}$ denotes the global rank-r POD basis. In the standard OpInf formulation [32,42], $\mathbf{V}_r$ is computed from the thin SVD of the full snapshot matrix $\mathbf{Q}$ by taking the first $r \ll m$ columns of $\mathbf{V}$, that is, the left singular vectors corresponding to the r largest singular values. While (4) is easily parallelizable, it would involve several steps: (i) compute the components $\mathbf{V}_{r,i} \in \mathbb{R}^{m_i \times r}$ of the POD basis on each core (which can be done by leveraging (2) or the thin SVD of $\mathbf{Q}_i$, for example), (ii) compute $\hat{\mathbf{Q}}_i = \mathbf{V}_{r,i}^\top \mathbf{Q}_i$ on each core, which would require $\mathcal{O}(m_i n_t r)$ operations and $\mathcal{O}(rn_t)$ bytes of memory, and (iii) compute $\hat{\mathbf{Q}}$ via a parallel reduction in terms of $\hat{\mathbf{Q}}_1, \hat{\mathbf{Q}}_2, \ldots, \hat{\mathbf{Q}}_p$, which would require communication. In contrast, from Eq. (2), the global rank-r POD basis can be equivalently computed as

$$\mathbf{V}_r = \mathbf{Q}\mathbf{W}_r \mathbf{\Sigma}_r^{-1} = \mathbf{Q}\mathbf{U}_r \mathbf{\Lambda}_r^{-\frac{1}{2}}. \tag{5}$$

This, in turn means that

$$\hat{\mathbf{Q}} = \left(\mathbf{Q}\mathbf{U}_r \mathbf{\Lambda}_r^{-\frac{1}{2}}\right)^\top \mathbf{Q} = \mathbf{T}_r^\top \mathbf{Q}^\top \mathbf{Q} = \mathbf{T}_r^\top \mathbf{D}. \tag{6}$$

Algorithm 2 dOpInf reduced operator learning.

Input: $p \geq 2$, n, n_t, r, *transform*, $\mathcal{B}_1$, $\mathcal{B}_2$, t_{trial}, $\tau \in (0,1)$
Output: reduced operators $\hat{\mathbf{c}} \in \mathbb{R}^r$, $\hat{\mathbf{A}} \in \mathbb{R}^{r\times r}$, $\hat{\mathbf{H}} \in \mathbb{R}^{r\times r^2}$

1: *split* the regularization parameter pairs in $\mathcal{B}_1 \times \mathcal{B}_2$ into p disjoint subsets $\mathcal{B}_1^i \times \mathcal{B}_2^i$ of equal size
2: *use* the reduced data matrix $\hat{\mathbf{Q}} \in \mathbb{R}^{r\times n_t}$ computed using Algorithm 1
3: **for** $i \leftarrow 1$ to p in parallel **do**
4: **for** $(\beta_1, \beta_2) \in \mathcal{B}_1^i \times \mathcal{B}_2^i$ **do**
5: *infer* the reduced operators using $\{\beta_1, \beta_2\}$
6: *compute* the reduced solution over the trial time horizon $[t_{\text{init}}, t_{\text{trial}}]$
7: perform a parallel reduction to *find* $(\beta_1^{\text{opt}}, \beta_2^{\text{opt}})$ that minimize the training error and ensure that the maximum deviation from the training mean of the inferred reduced coefficients over the trial horizon stays within τ of the maximum deviation from the mean over training
8: *determine* the index i^{opt} of the compute core where $(\beta_1^{\text{opt}}, \beta_2^{\text{opt}})$ resides
9: **if** $i = i^{\text{opt}}$ **then**
10: the target reduced operators $\{\hat{\mathbf{c}}, \hat{\mathbf{A}}, \hat{\mathbf{H}}\}$ are the ones *inferred* using $(\beta_1^{\text{opt}}, \beta_2^{\text{opt}})$

Therefore, by starting from the POD method of snapshots, $\hat{\mathbf{Q}}$ can be computed in terms of the small matrices $\mathbf{T}_r$ and $\mathbf{D}$ at a much lower cost of $\mathcal{O}(rn_t^2)$ operations and $\mathcal{O}(rn_t)$ bytes of local main memory without necessitating the computation of the POD basis.

The total cost of Algorithm 1 per core is in $\mathcal{O}(m_i n_t^2 + n_t^2 + rn_t + rn_t^2) = \mathcal{O}((m_i + r + 1)n_t^2) = \mathcal{O}(m_i n_t^2)$ since r is generally smaller than m_i, plus the cost of computing the eigendecomposition of $\mathbf{D}$ and the global POD singular values from the eigenvalues of $\mathbf{D}$. The corresponding memory requirements are in $\mathcal{O}(m_i n_t + 2n_t^2 + 2rn_t)$ bytes plus the memory requirements for holding the eigenpairs of $\mathbf{D}$.

Remark 2. Step 9 in Algorithm 1 computes and broadcasts $\mathbf{D}$ to all other cores via a collective reduction, and subsequently all cores compute the reduced eigendecomposition of $\mathbf{D}$ as well as the matrices $\mathbf{T}_r$ and $\hat{\mathbf{Q}}$ (steps 10–13). This is because the reduced representation of the transformed snapshot data, $\hat{\mathbf{Q}}$, is required by all cores for inferring the reduced model operators via OpInf in the next step, and computing $\hat{\mathbf{Q}}$ depends on $\mathbf{T}_r$ and $\mathbf{D}$ via (6). In addition, as we will discuss in Section 3.3, $\mathbf{T}_r$ is generally also needed by all cores for postprocessing the reduced solution produced by the dOpInf ROM. These computations are cheap when n_t and r are significantly smaller than m, in which case they do not affect the scalability of the algorithm (as demonstrated by our scalability results in Section 4.3). An alternative approach would be to compute $\mathbf{D}$ at step 9 using a standard reduction to one core, perform all subsequent computations on that core and broadcast the results to all other compute cores. However, this strategy requires communication and can create a bottleneck, as all remaining computations must wait for the results, which could reduce the overall efficiency of the data dimensionality reduction procedure.

Remark 3. Algorithm 1 is distributed with respect to the large transformed snapshot dimension, m. All other steps (steps 10–13) are sequential in the current formulation. When n_t is large, the cost of these steps per core will become dominant, which can impact the scalability of Algorithm 1. One solution is to compute $\mathbf{T}_r$ (step 12) and $\hat{\mathbf{Q}}$ (step 13) in parallel on the p cores with respect to n_t. In addition, the full or reduced eigendecomposition of $\mathbf{D}$ at step 10 can be performed in parallel using high-performance linear algebra libraries like ELPA[1] (Eigenvalue Solvers for Petaflop Applications) [2]. Another solution is to leverage streaming methods: instead of processing all n_t snapshots at once, we would process them in streams of size n_ℓ such that $n_\ell < n_t$ [34]. Future work will explore incorporating streaming approaches into Algorithm 1.

3.2. Distributed learning of the reduced model operators

We next use the reduced data matrix $\hat{\mathbf{Q}}$ to learn the reduced operators that specify the reduced model. For example, for a quadratic reduced model

$$\dot{\hat{\mathbf{q}}} = \hat{\mathbf{A}}\hat{\mathbf{q}} + \hat{\mathbf{H}}(\hat{\mathbf{q}} \otimes \hat{\mathbf{q}}) + \hat{\mathbf{c}}, \tag{7}$$

we must determine the constant, linear, and quadratic reduced operators $\hat{\mathbf{c}} \in \mathbb{R}^r, \hat{\mathbf{A}} \in \mathbb{R}^{r\times r}$, and $\hat{\mathbf{H}} \in \mathbb{R}^{r\times r^2}$. OpInf determines the reduced operators that best match the projected snapshot data in a minimum residual sense by solving the linear least-squares minimization

$$\underset{\hat{\mathbf{O}}}{\operatorname{argmin}} \left\| \hat{\mathbf{D}}\hat{\mathbf{O}}^\top - \dot{\hat{\mathbf{Q}}}^\top \right\|_F^2 + \beta_1 \left(\left\| \hat{\mathbf{A}} \right\|_F^2 + \|\hat{\mathbf{c}}\|_F^2 \right) + \beta_2 \left\| \hat{\mathbf{H}} \right\|_F^2, \tag{8}$$

where $\hat{\mathbf{O}} = \left[\hat{\mathbf{A}} \,|\, \hat{\mathbf{H}} \,|\, \hat{\mathbf{c}} \right] \in \mathbb{R}^{r\times(r+r^2+1)}$ denotes the unknown operators, $\hat{\mathbf{D}} = \left[\hat{\mathbf{Q}}^\top \,|\, \hat{\mathbf{Q}}^\top \odot \hat{\mathbf{Q}}^\top \,|\, \hat{\mathbf{1}}_{n_t} \right] \in \mathbb{R}^{n_t\times(r+r^2+1)}$ denotes the OpInf data, $\dot{\hat{\mathbf{Q}}} \in \mathbb{R}^{r\times n_t}$ is the time derivative data, and F denotes the Frobenius norm. When the high-fidelity model does not provide time derivative data, $\dot{\hat{\mathbf{Q}}}$ must be approximated numerically using finite differences, for example. Numerical approximations, however, will likely be inaccurate in applications with severely downsampled snapshots, as it was also observed in the RDRE scenario under consideration. In such cases, we use the time-discrete version of OpInf [20] and learn the corresponding reduced operators analogously to Eq. (8), where instead of using $\dot{\hat{\mathbf{Q}}}$, we shift $\hat{\mathbf{Q}}$ by one column the right as is done in DMD, which learns linear discrete-time systems.

The learning step that can benefit from a distributed formulation is the search for the optimal regularization hyperparameter pair $(\beta_1^{\text{opt}}, \beta_2^{\text{opt}}) \in \mathbb{R}^2_{>0}$. These parameters are introduced to address overfitting and accommodate model misspecification and other sources of error [38]. Following [38,43], the regularization hyperparameters can be found through a grid search involving two nested loops. The regularization pairs are independent of each other, which means that this search is embarrassingly parallel. The optimal hyperparameters minimize the training error under the constraint that the inferred reduced coefficients have bounded growth within a trial time horizon at least as long as the target time horizon [43].

We present the steps to learn the reduced operators using a distributed search for the optimal hyperparameters in Algorithm 2. In addition to the input parameters from Algorithm 1, we have the sets $\mathcal{B}_1$, $\mathcal{B}_2 \subset \mathbb{R}_{>0}$ comprising the candidate regularization parameters $\beta_1 \in \mathcal{B}_1$ and $\beta_2 \in \mathcal{B}_2$, $t_{\text{trial}} \geq t_{\text{final}}$ denoting the end of the trial time horizon, and a percentage $\tau \in (0,1)$ constraining the growth of the inferred reduced coefficients over the trial horizon. For convenience, we choose $\mathcal{B}_1$ and $\mathcal{B}_2$ such that the cardinality $B \in \mathbb{N}$ of $\mathcal{B}_1 \times \mathcal{B}_2$ is divisible by p. We start by splitting the regularization pairs in $\mathcal{B}_1 \times \mathcal{B}_2$ into p disjoint subsets of equal size B/p. For each regularization pair in $\mathcal{B}_1^i \times \mathcal{B}_2^i$, we first infer the reduced operators using Eq. (8) and then compute the reduced solution over the trial time horizon $[t_{\text{init}}, t_{\text{trial}}]$. Since (8) depends on the reduced dimension r and the number of training snapshots n_t, solving it is computationally cheap in general and can be done utilizing standard least-squares solvers [24]. Computing the dOpInf reduced solution is also cheap; in fact, in large-scale applications, this is orders of magnitude faster than the high-fidelity model. The distributed partitioning of the candidate regularization pairs therefore decreases the cost of the

[1] https://elpa.mpcdf.mpg.de/.

grid search from B least-squares solves (8) and reduced solution calculations in the sequential approach to B/p such calculations per core.

The optimal regularization pair $(\beta_1^{\text{opt}}, \beta_2^{\text{opt}})$ is found via a parallel reduction that minimizes the error between the computed reduced solutions and the given projected training data $\hat{\mathbf{Q}}$ (this error can be, for example, the mean-squared error) subject to a constraint. The maximum deviation of the inferred reduced coefficients over the trial horizon from the mean of the reduced-order coefficients over training must stay within τ of the maximum deviation from the mean over training. We then determine the index i^{opt} of the compute core where the reduced operators inferred using the optimal regularization pair reside.

3.3. Distributed postprocessing of the reduced solution

Finally, we postprocess the reduced solution produced by the dOpInf ROM constructed using the optimal regularization parameter pair $(\beta_1^{\text{opt}}, \beta_2^{\text{opt}})$. Let $\tilde{\mathbf{Q}} \in \mathbb{R}^{r \times n_p}$ denote the matrix containing the reduced solution at $n_p \in \mathbb{N}$ target time instants. For example, n_p can represent user-defined, representative time instants or all time instants over the time horizon of interest $[t_{\text{init}}, t_{\text{final}}]$.

In many cases, postprocessing involves computing low-dimensional output quantities of interest that are relevant to the problem at hand (e.g., integrated forces, average field values, state values over a specific region of interest, etc.). In some cases, postprocessing the reduced solution might involve mapping it back to the original high-dimensional space. This would be the case, for example, if the user desired visualization of the full state fields, such as pressure or temperature. In these cases, in order to map from the reduced solution $\tilde{\mathbf{Q}}$ to the desired full-state field, we must compute the corresponding components of the rank-r POD basis (noting that our proposed dOpInf algorithm has thus far avoided computing the POD basis). To achieve this POD basis computation efficiently, we distribute it across cores. Let $\mathbf{V}_{r,i} \in \mathbb{R}^{m_i \times r}$ denote the components of the global POD basis on the ith core, where $i = 1, 2, \ldots, p$. From Eq. (5), we can compute $\mathbf{V}_{r,i}$ as

$$\mathbf{V}_{r,i} = \mathbf{Q}_i \mathbf{T}_r. \tag{9}$$

Equation (9) requires $\mathcal{O}(m_i n_t r)$ operations and $\mathcal{O}(m_i r)$ bytes of memory for $\mathbf{V}_{r,i}$ on the ith core. Lifting the reduced solution back to the high-dimensional space is done by computing $\mathbf{V}_{r,i}\tilde{\mathbf{Q}} \in \mathbb{R}^{m_i \times n_p}$ at a cost of $\mathcal{O}(m_i n_p r)$ operations and $\mathcal{O}(m_i n_p)$ bytes of memory for the result on the ith core. To obtain the approximate solutions in the original coordinates, we apply all inverse data transformations, provided data transformations were used during preprocessing. These transformations are typically applied independently on each core, with a computational cost that varies depending on the specific transformation. This cost, however, is usually low. Finally, based on the postprocessing objective, the computed approximate solutions might be saved to disk or utilized for computing errors or other quantities.

3.4. Algorithm summary

In summary, dOpInf offers a complete distributed workflow for physics-based ROMs tailored to problems with extremely large state dimensions. Key aspects of our approach include efficient data transformations and efficient dimensionality reduction without necessitating the computation of the POD basis. The primary computational steps involve standard linear algebra operations, such as matrix-matrix multiplications and eigenvalue solvers, which can be efficiently implemented using state-of-the-art scientific computing libraries. This, and the fact that our formulation requires few communication steps, results in a scalable, distributed memory approach for learning physics-based ROMs. These operations can be efficiently performed on CPUs as well as on specialized hardware such as graphical processing units or tensor processing units [35].

Elements of our distributed approach are transferable to other approaches such as DMD [33,49,55] and quadratic manifolds [6,23], and also to parametric ROMs [12]. A summary for parametric ROM settings is as follows. Let $\boldsymbol{\mu} \in \mathbb{R}^{n_u}$ denote the n_u-dimensional vector of parameters of interest (e.g., comprising the mass flow, equivalence ratio and other relevant parameters in RDRE simulations). Let $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_z$ denote z instances used for training and denote by $\mathbf{S}_\alpha \in \mathbb{R}^{n \times n_t}$ the corresponding snapshot matrices containing n_t snapshots for $\alpha = 1, 2, \ldots, z$. Following the parametric ROM survey in Ref. [12], two strategies for reducing the high-dimensional data are to use a global reduced basis using all z snapshot matrices or separate reduced basis for each training parameter instance. For either strategy Algorithm 1 can be used for distributed data transformations and dimensionality reduction. These approaches proceed by constructing reduced operators for each training parameter instance which can be done via Algorithm 2. Given a new parameter instance $\boldsymbol{\mu}$, the corresponding reduced operators are typically obtained by interpolating the reduced operators for training; we refer to [12] for more details on interpolation strategies. Finally, the resulting ROM solution can be postprocessed in parallel analogously to Section 3.3.

4. Application to a large-scale real-world combustion scenario

In this section, we assess the scalability and predictive capabilities of the proposed dOpInf algorithm in a complex three-dimensional unsteady RDRE scenario. We focus on RDRE simulations as a representative example of a real-world application for which ROMs are needed to enable real-world engineering tasks that would be intractable with high-fidelity models. Section 4.1 summarizes the considered RDRE scenario, followed by the description of the available high-fidelity dataset in Sec. 4.2. Section 4.3 presents our dOpInf scalability results using up to $2,048$ cores on the Frontera supercomputer. Finally, we assess the prediction capabilities of dOpInf ROMs in the RDRE scenario under consideration in Sec. 4.4.

4.1. Overview of the considered rotating detonation rocket engine scenario

The RDRE concept injects fuel into an axially symmetric chamber, such as an annulus. When ignited under appropriate conditions, this process generates a system of spinning detonation waves [16]. RDREs offer several advantages compared to conventional devices, including mechanical simplicity, which has led to active research in RDRE designs. Designing optimized RDREs, however, remains an open challenge due the difficulty in both simulation-based and experimental design space explorations. Large-eddy simulations (LES) of RDREs have advanced and can now provide valuable data that inform performance, stability, and realizability assessments of a given RDRE design [7,36]. However, the large computational cost of LES, even on large supercomputers, makes it impractical for design optimization purposes—a single LES usually requires millions of core hours [8]. This highlights the need for scalable and predictive ROMs.

The physics of the considered RDRE scenario are modeled using the three-dimensional, reactive, viscous Navier-Stokes equations coupled with a skeletal chemistry mechanism (FFCMy-12) based on the FFCM model [52]. This scenario is based on a design with 72 discrete injector pairs. The high-fidelity simulations for the full RDRE were performed using implicit LES via the AHFM (ALREST High-Fidelity Modeling) large-scale simulation code from the Air Force Research Laboratory (AFRL). For more details about the code and typical setups, we refer the reader to [7,36]. The high-fidelity LES employed a spatial discretization via a multi-block mesh comprising 136 million spatial cells and a temporal discretization via an adaptive timestep $\Delta t \approx 10^{-9}$ seconds in the quasi-limit-cycle regime. Each LES analysis requires approximately one million CPU-hours for one millisecond of simulated physical time on the DoD supercomputer Nautilus using $14,336$ cores across 112 AMD EPYC Milan nodes. This amounts to almost three days of CPU time, excluding queue waiting and postprocessing times.

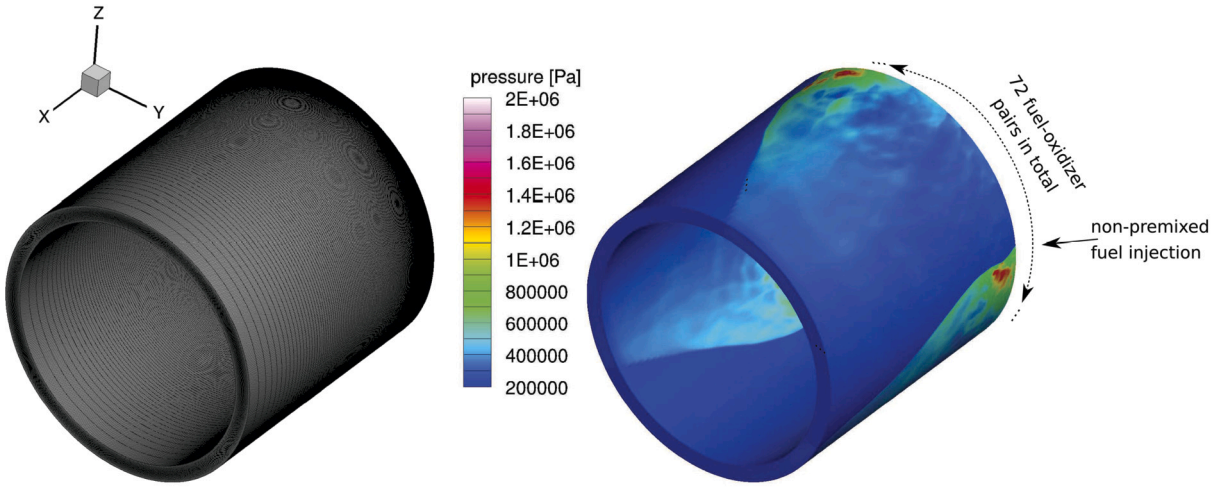


Fig. 1. Combustion chamber domain for the considered RDRE scenario. The left figure plots the structured mesh. The right figure plots an example pressure field showing the three dominant co-rotating waves.

The input values of the mass flow-rate, $\dot{m} = 0.267\ \text{kg} \cdot \text{s}^{-1}$, and equivalence ratio, $\Phi = 1.16$, lead to the formation of three dominant co-rotating waves and no secondary waves in the quasi-limit cycle regime. Fig. 1 plots the combustion chamber of this RDRE, showing the mesh and an example pressure field with the three detonation waves.

We construct ROMs via dOpInf for the RDRE combustion chamber. The three-dimensional computational domain spans from 0.05 to 76.15 millimeters in the x direction and from -37.50 to 37.50 millimeters in both y and z directions. It has a fixed channel height of 4.44 millimeters throughout. The combustion chamber degrees of freedom were extracted from the full RDRE simulation results and interpolated onto a structured grid comprising $n_x = 4,204,200$ spatial degrees of freedom, with clustering of grid points at mid channel and closer toward the injector plane, as shown in Fig. 1.

4.2. Acquiring the large-scale training dataset

The governing equations are not in polynomial form. We follow the work in [54] and represent the transformed governing equations in terms of the following $m_s = 18$ physical state variables:

$$q = [1/\rho \;\; p \;\; v_x \;\; v_y \;\; v_z \;\; w_{\text{CH}_4} \;\; w_{\text{O}_2} \;\; w_{\text{CO}_2} \;\; w_{\text{H}_2\text{O}} \;\; w_{\text{CO}} \;\; w_{\text{H}_2} \;\; w_{\text{OH}} \;\; w_{\text{CH}_2\text{O}} \;\; w_{\text{CH}_4} \;\; w_{\text{HO}_2} \;\; w_{\text{H}} \;\; w_{\text{O}} \;\; T]^\top, \tag{10}$$

where ρ [$\text{kg} \cdot \text{m}^{-3}$] is density (and $1/\rho$ [$\text{m}^3 \cdot \text{kg}^{-1}$] is specific volume), p [Pa] is pressure, v_x, v_y, v_z [$\text{m} \cdot \text{s}^{-1}$] are the velocity components, $w_\cdot$ are the mass fractions of all chemical species (12 in total here), and T [K] is temperature. The $m_s = 18$ variables in Eq. (10) make most terms in the lifted governing equations linear or quadratic. The dimensionality of the transformed snapshots is therefore $m = 18 \times 4,204,200 = 75,675,600$.

The computational cost and size of the resulting simulation datasets limit the number of time instants for which the high-fidelity LES solutions can be saved to disk. We have a total of 3,805 downsampled snapshots from the high-fidelity LES over the time horizon [5.0000, 5.3804] milliseconds (the down-sampling factor was about 100), which represents one of the largest RDRE datasets used for ROM development. The 3,805 downsampled snapshots correspond to about three full quasi-cycles of the three-wave system. Because of the severe downsampling of the snapshots, we employ the discrete formulation of OpInf [20] (cf. Section 3.2). Wave cycles in the quasi-steady state of RDRE simulations usually exhibit near-periodicity but also cycle-to-cycle variations. This lack of self-similarity can present a considerable challenge for accurate modeling. Simulating the full 5.3804 milliseconds of physical time required about 5.4 million core hours on 14,336 compute cores, which means that the 0.3804 milliseconds spanning the target time interval necessitated about 380,000 core hours, corresponding to a simulation time of 27 hours. We use the first $n_t = 2,536$ snapshots (i.e., two thirds of the total number of available snapshots) for training dOpInf ROMs, corresponding to two quasi-cycles of the three-wave system. The remaining 1,269 snapshots are used to assess the prediction capabilities of the dOpInf ROM beyond training. We note that constructing ROMs for RDRE predictions beyond a training time horizon is an essential step for assessing the prediction capabilities of ROMs in these complex applications as well as for enabling predictions over time horizons that would be computationally too expensive to simulate using high-fidelity LES—recall that one millisecond of simulated physical time via LES can require one million core hours on a supercomputer. An initial study showcasing the potential of OpInf (using the standard, serial formulation) for constructing parametric data-driven ROMs for three-dimensional unsteady RDREs can be found in Ref. [20]. As noted in Section 3.4, elements of our dOpInf workflow can be also transferred to the parametric setting.

The large dimension $m = 75,675,600$ of the $n_t = 2,536$ training snapshots means that their total size amounts to 1.4 TB in double precision. As such, performing data transformation and dimensionality reduction to construct ROMs would be computationally expensive and limited by memory storage and bandwidth. Our distributed formulation, in contrast, offers a scalable workflow for constructing physics-based ROMs on distributed memory machines.

4.3. Scalability results on the Frontera supercomputer

We first demonstrate the scalability of the proposed dOpInf approach, from loading the training data in parallel to computing the reduced solution over the target time horizon [5.0000, 5.3804] (i.e., using Algorithms 1 and 2). The number of training snapshots, $n_t = 2,536$, permits a maximum reduced dimension $r = 68$ for a quadratic dOpInf ROM since it sets the maximum number of operator coefficients that can be learned via the regularized regression problem (8). In all our subsequent experiments we construct dOpInf ROMs with reduced dimension $r = 68$. The number of training snapshots (i.e., $n_t = 2,536$) and the target time horizon (i.e., [5.0000, 5.3804] milliseconds) are the same in all experiments. Moreover, the trial time horizon for the optimal regularization parameter search in Algorithm 2 is the same as the target horizon.

Our distributed memory implementation was done using `Python` (`Python 3.7` on Frontera) and the Message Passing Interface (MPI) bindings provided by the `mpi4py` library. We leveraged TACC's scientific software stack to maximize performance.[2] `mpi4py` uses the Intel

[2] https://docs.tacc.utexas.edu/hpc/frontera/#ml.

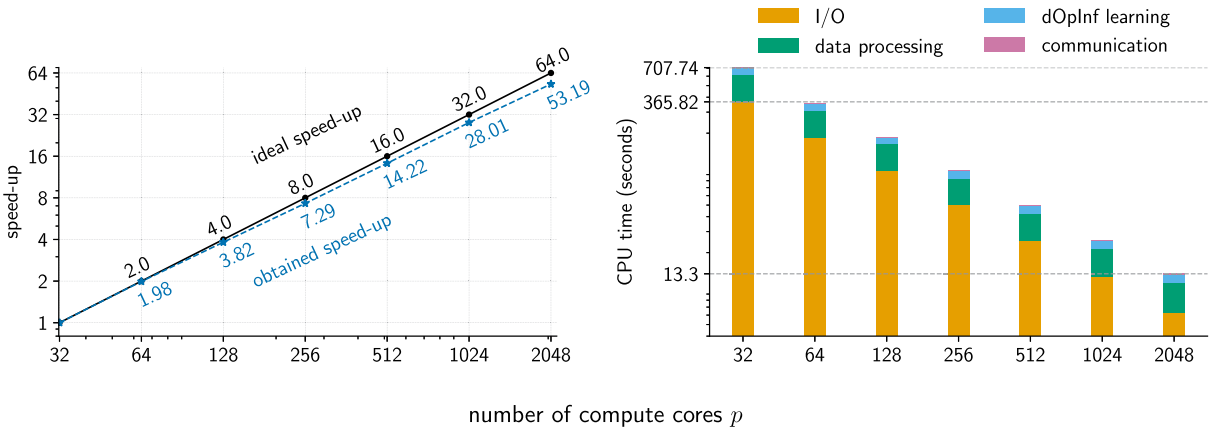


Fig. 2. Strong scaling results using between $p = 32$ and $p = 2,048$ cores on Frontera. The left plot shows the speed-ups and the right plot shows the percentages of the total CPU time corresponding to data loading, all data processing computations, learning the reduced operators with dOpInf, and communication overhead.

MPI implementation tailored to the underlying hardware, thus ensuring fast MPI operations. The dense linear algebra operations are performed using `numpy`'s linear algebra functions `matmul` (for matrix-matrix multiplications) and `scipy`'s `eigsh` function for partial eigendecompositions of real symmetric matrices. `numpy` and `scipy` are linked to the Intel Math Kernel Library (MKL) and compiled to take advantage of CORE-AVX512 instruction set architecture relevant to Frontera nodes. MKL provides access to optimized versions of the low-level BLAS and LAPACK libraries.

To obtain a comprehensive assessment of our algorithm's scalability, we use up to 2,048 cores distributed over Cascade Lake (CLX) compute nodes on the Frontera supercomputer. Each CLX node provides 56 cores across two sockets and 192 GB of main memory. In assessing our algorithm's scalability, we average the CPU times over five measurements for each core count to mitigate the effects of fluctuations.

The RDRE training dataset was saved in HDF5 format and transferred on the `scratch1` partition on the Frontera supercomputer. This employs a Lustre file system,[3] typically used on large-scale computing systems. To mitigate file system bottlenecks when using a large number of compute cores, we distributed the 1.4 TB training dataset across multiple HDF5 files, which represents a common approach for handling large datasets. Each file contains data for the $m_s = 18$ transformed physical state variables within a specific subdomain of the full computational domain. For a comprehensive overview of Frontera's file system and recommended best practices, we refer the reader to Ref. [37]. For loading the training dataset in parallel in our implementation, we used the `h5py` library which is linked to the parallel HDF5 (phdf5) library on Frontera. We note that data reading is independent of the other dOpInf algorithmic steps. Users are thus responsible for ensuring that their training data are stored in a format conducive to efficient I/O operations.

We begin with the results for strong scaling. The goal is to assess the speed-ups obtained with the dOpInf algorithm, from loading the data in parallel to computing the reduced solution using the dOpInf ROM over the target time horizon $[5.0000, 5.3804]$ milliseconds. We keep the problem size fixed, that is, we employ the entire snapshot data of size $75,675,600 \times 2,536$ to construct a predictive dOpInf ROM with reduced dimension $r = 68$ using our distributed workflow, and increase the number of compute cores. Here, we use $p \in \{32; 64; 128; 256; 512; 1,024; 2,048\}$. The regularization hyperparameter search is conducted using a grid of size 32×64 (amounting to $B = 2,048$ candidate pairs in total). The speed-up is computed as $T(32)/T(p)$. Here, $T(p)$ denotes the average CPU time using p cores measured on the compute core that contains the optimal regularization hyperparameters. The ideal speed-up is $p/32$.

Fig. 2 plots the results. The left plot shows the speed-ups relative to $p = 32$ cores and the right plot shows the corresponding CPU time distribution into loading the snapshot data (I/O), all data processing computations in Algorithm 1, learning the reduced operators via Algorithm 2, and the communication overhead in both Algorithms 1 and 2. The total CPU time decreases from 707.74 ± 8.94 seconds for $p = 32$ cores down to 13.30 ± 0.04 seconds for $p = 2,048$ compute cores. Our results closely match the ideal speed-ups, indicating excellent strong scalability. I/O and data processing computations drive the CPU times, but their excellent scalability coupled with a minimal communication overhead leads to the obtained near-ideal speed-ups. Because of this, constructing the dOpInf ROM with dimension $r = 68$ took only 13.30 ± 0.04 seconds using $p = 2,048$ cores. This breaks down to 6.30 ± 0.04 seconds for I/O, 4.89 ± 0.01 seconds for all data processing computations, 1.84 ± 0.01 seconds for dOpInf learning, and 0.26 ± 0.01 seconds for communication. Therefore, in contrast to many existing data-driven reduced and surrogate modeling approaches that require substantial CPU times, our method enables a rapid and efficient construction of physics-based ROMs in problems with large datasets and a large snapshot dimension.

We next assess the weak scaling efficiency of Algorithms 1 and 2. In contrast to the speed-up analysis where we fixed the problem size and scaled the number of cores, here we fix the problem size per core. Our goal is to evaluate how effectively our distributed workflow utilizes resources as both the problem size (i.e., the size of the snapshot matrix and the resolution of the grid used for searching the optimal regularization parameter pair) and the number of cores increase, which provides more insights into its scalability for datasets with extremely large state dimension. We vary the number of cores from $p = 1$ and $p = 2,048$, increasing in powers of two. The largest core count, $p = 2,048$, is used to construct the dOpInf ROM using the full training dataset and a grid of size 32×64 for the regularization hyperparameter search. Note that this setup is the same as the one used for the strong scaling analysis for $p = 2,048$ cores. To maintain a consistent problem size per core for configurations with fewer than 2,048 cores, we adjust the problem size by downsampling the full snapshot dimension and reducing the grid size for the hyperparameter search such that we have $B = p$ candidate pairs for all considered p core counts. The computed weak scaling efficiency is $T(1)/T(p)$, where $T(1)$ corresponds to the serial implementation of Algorithms 1 and 2 using a downsampled snapshot matrix of size $36,951 \times 2,536$ (i.e., with a downsampled snapshot dimension $75,675,600/2,048$), averaged over five runs. We obtain $T(1) = 12.51 \pm 0.20$ seconds (6.16 ± 0.15 seconds for I/O, 4.66 ± 0.06 for all data processing computations, and 1.68 ± 0.02 seconds for dOpInf learning).

A good weak scaling performance would be obtained if our algorithm maintained a similar CPU time for I/O and all computations, and a low communication overhead as p increases. Fig. 3 plots the results.

[3] https://docs.tacc.utexas.edu/hpc/frontera/#files.

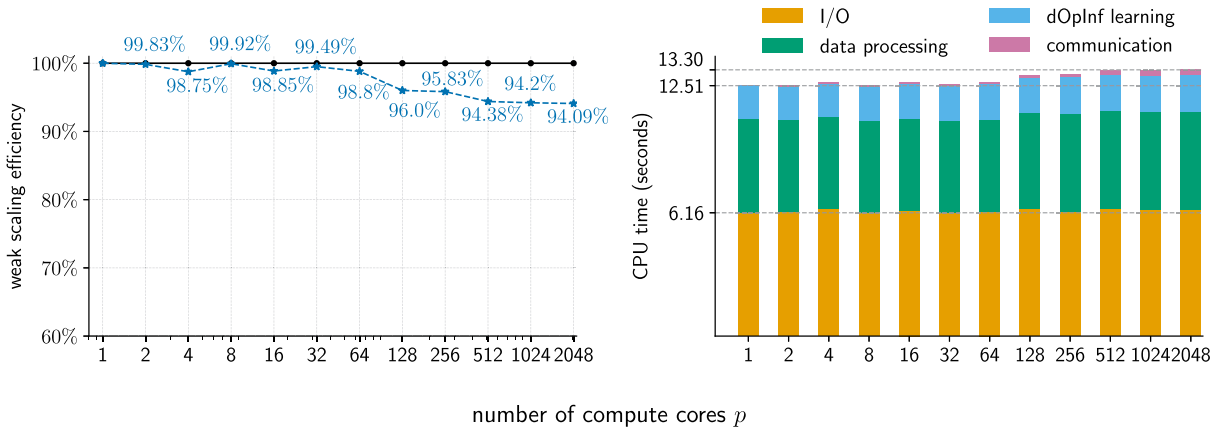


Fig. 3. Weak scaling results using between $p = 1$ and $p = 2,048$ cores on Frontera. The left plot shows the obtained efficiency and the right plot shows the corresponding CPU times for data loading, all data processing computations, learning the reduced operators with dOpInf, and communication overhead.

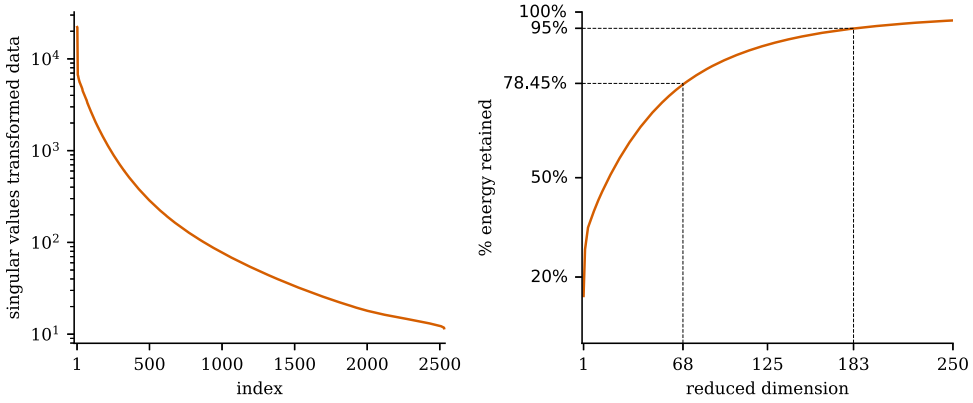


Fig. 4. The left figure plots the POD singular values of the transformed snapshots. The right figure plots the corresponding retained energy.

Our algorithm indeed demonstrates an excellent weak scaling efficiency that exceeds 94% for all values of p, as shown on the left. The right plot in Fig. 3 shows that our algorithm maintains a similar CPU time for data loading and all computations, whereas the communication overhead increases only slowly with p.

Two important elements for the obtained scalability results were the efficient parallel loading of the large training dataset and the low computational cost of the sequential steps in Algorithm 1 (cf. Remarks 2 and 3). Furthermore, the dOpInf efficiency, evident in its ability to complete the entire workflow (parallel data loading, data transformations, dimensionality reduction, and ROM construction via a grid search over $B = 2,048$ candidate pairs) in an average time of just 13.30 seconds on $p = 2,048$ cores was facilitated by the powerful scientific software stack on Frontera.

4.4. Predictions beyond the training time horizon

We now assess the potential of the proposed distributed memory model reduction workflow for constructing predictive structure-preserving, physics-based ROMs for the real-world RDRE scenario under consideration. We use the ROM results obtained for the scalability studies above for $p = 2,048$ cores. Recall that the average CPU time for constructing the dOpInf ROM with reduced dimension $r = 68$ amounted to 13.30 seconds in total.

Remark 4. Due to the complexity of RDRE simulations, we prioritize ROMs that accurately capture large-scale features, including detonation and shockwave fronts. These ROMs should preserve essential time-averaged chamber quantities, wave propagation characteristics, and key engineering quantities like pressure, axial velocity/temperature, and fuel/oxidizer mass fractions. Achieving a ROM that accurately represents these large-scale features provides a practically useful capability for design exploration and analysis tasks, even if it does not capture all small-scale features.

We start by analyzing the decay of the POD singular values. Fig. 4 plots the POD singular values on the left and corresponding retained energy on the right. A slow decay of the singular values is expected due to the complex dynamics in this scenario. In addition, the corresponding POD basis is global, representing all $m_s = 18$ transformed variables in (10), characterized by heterogeneous dynamics. Retaining 95% of the total energy requires a large reduced dimension, $r = 183$. In contrast, $r = 68$ retains 78.45% of the total energy. We will nonetheless show that the dOpInf ROM with reduced dimension $r = 68$ produces predictions that meet the requirements specified in Remark 4.

We then assess how well the ROM solution meets the accuracy criteria presented in Remark 4. We utilize the ROM solution to extract one-dimensional radial profiles for pressure, temperature, and fuel (CH_4) mass fraction at three representative locations near the mid-channel of the combustion chamber. Axially, the first location is close to the injectors, the second location is further way from the injectors but still within the detonation region, and the third location is downstream of the detonation zone. For a more comprehensive evaluation, we also compute the full pressure field. This enables us to assess the ROM's ability to represent the large-scale features of the three detonation waves in the full computational domain. To map the ROM solution back to the original coordinates, we compute the components of the global POD basis

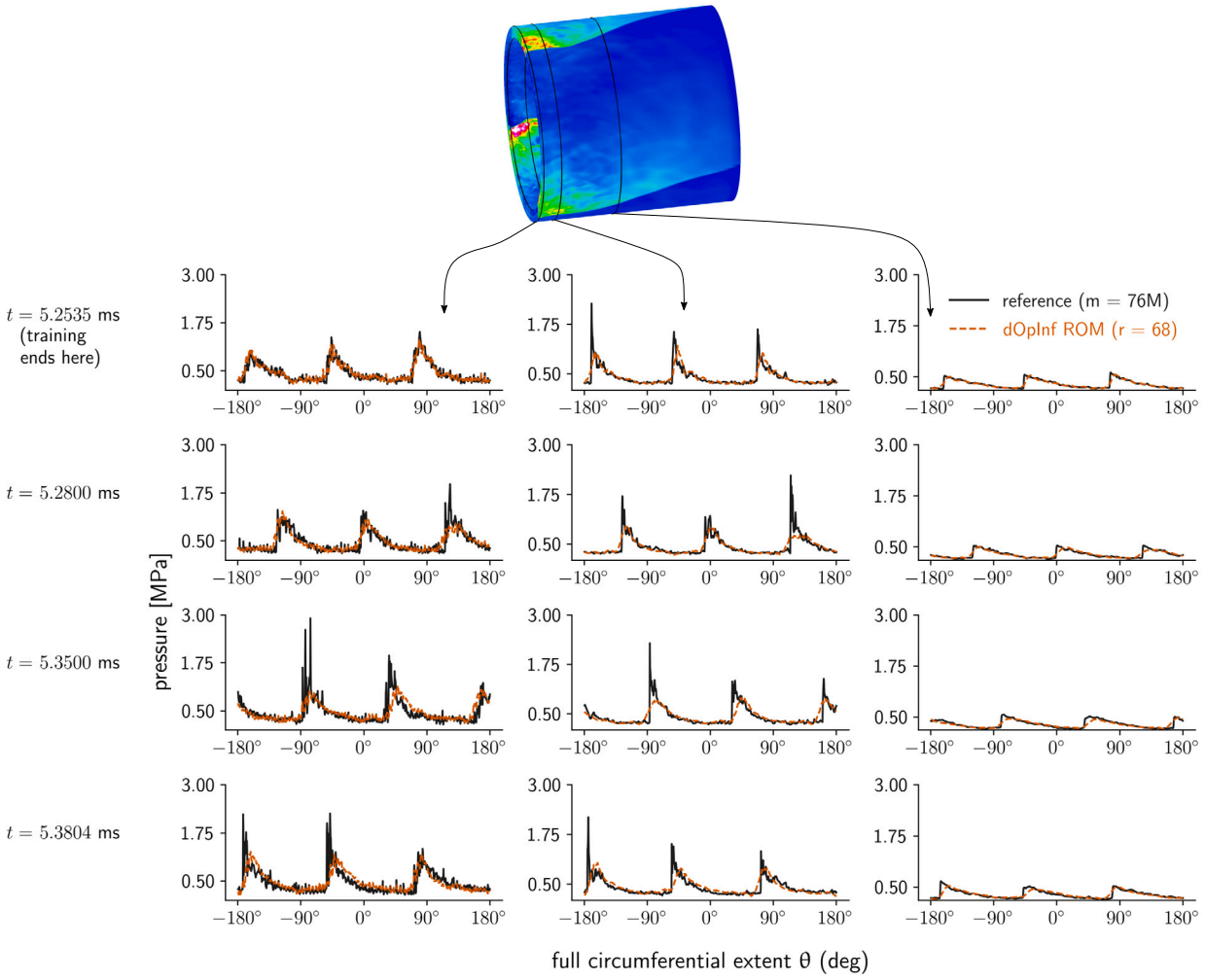


Fig. 5. One-dimensional circumferential profiles for pressure. The columns plot the results at three representative locations close to the mid-channel. The rows plot the profiles at four representative time instants.

of rank $r = 68$ via Eq. (9), which requires an average of 0.24 seconds on $p = 2,048$ cores.

We present first the approximate solutions for pressure. Fig. 5 plots the radial profiles at four representative time instants: the last time instant in the training horizon, $t = 5.2535$ ms, and three time instances in the prediction horizon, $t = 5.2800$ ms (the 265th), $t = 5.3500$ ms (the 965th), and $t = 5.3804$ ms (the last, i.e., 1269th time instant in the prediction horizon). The obtained profiles align well with the corresponding reference results: the ROM approximate solutions capture larger-scale features, including the number of waves, wave fronts, and also the lower amplitudes of the reference solutions, but, as expected, they do not capture the abruptly changing pressure spikes in the reference solution. Nevertheless, pressure spikes are not considered a critical metric in this context, as they can exhibit significant variability among high-fidelity simulations. Fig. 6 plots the corresponding full fields, which shows that the ROM solution adequately captures the key characteristics of the shock waves in the full computational domain.

Figs. 7 and 8 plot the radial profiles for temperature and fuel mass fraction, respectively, at the same time instants as for the pressure profiles in Fig. 5. Note that Fig. 8 plots the profiles at only the first two locations since the fuel mass fraction values at the third location are close to zero. The non-smoothness of these profiles as well as the variations between the three quasi-cycles further evidentiate the complexity of the dynamics in the RDRE scenario under consideration. However, our dOpInf ROM captures well the larger-scale features.

Finally, the average evaluation time of the resulting dOpInf ROM is 1.09 seconds on one CLX core. This represents a computational cost reduction of 90,000 compared to the CPU time of the high- fidelity simulation over the target time horizon.

5. Conclusion

Recent advancements in HPC simulations of complex real-world problems necessitate the development of innovative, parallelizable data-driven model reduction techniques tailored to modern HPC architectures. This paper demonstrated the power of integrating HPC into data-driven reduced modeling. The proposed distributed Operator Inference algorithm allows a fast and scalable processing of extremely large datasets, and the construction of predictive physics-based reduced models that approximate the dynamics underlying these datasets. These capabilities unlock new possibilities for computationally expensive tasks like design optimization, which would otherwise be intractable using high-fidelity models. These developments hold promise for a wide range of fields, including rocket propulsion and the assessment of turbulent transport in fusion devices. An implementation of the distributed Operator Inference algorithm, including a detailed tutorial, is available at https://github.com/ionutfarcas/distributed_Operator_Inference.

The proposed distributed algorithm inherits the limitations of standard Operator Inference, namely, the difficulty of effectively construct-

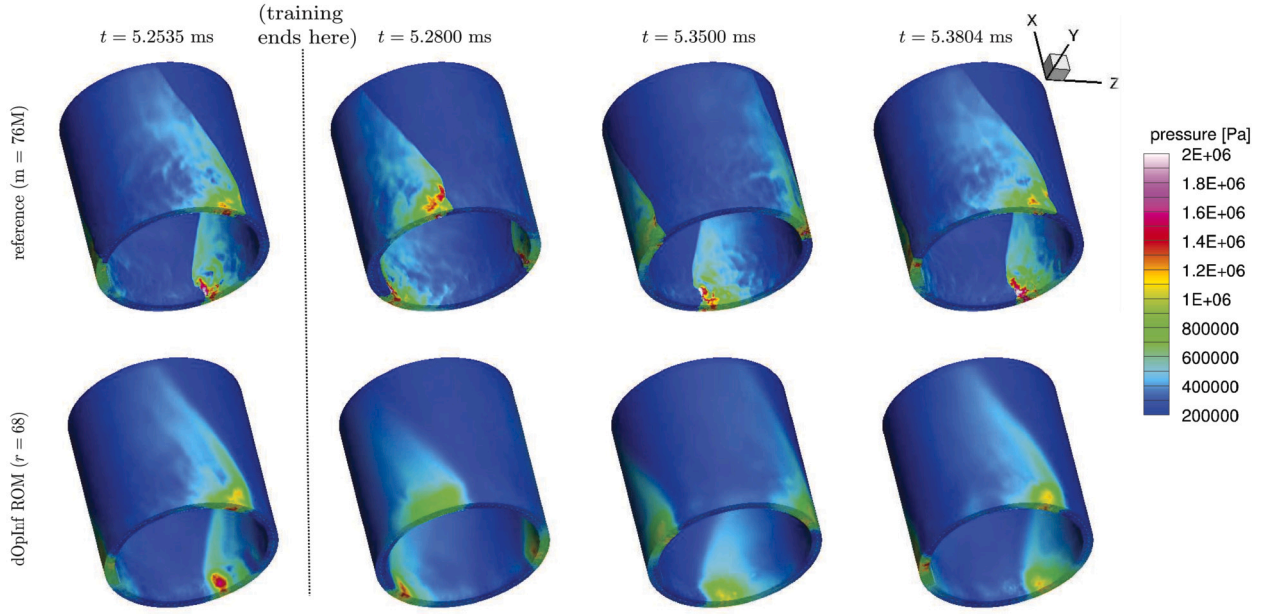


Fig. 6. Pressure fields at four time instants. The top row plots the reference solutions. The bottom row plots the approximate solution obtained with our reduced model.

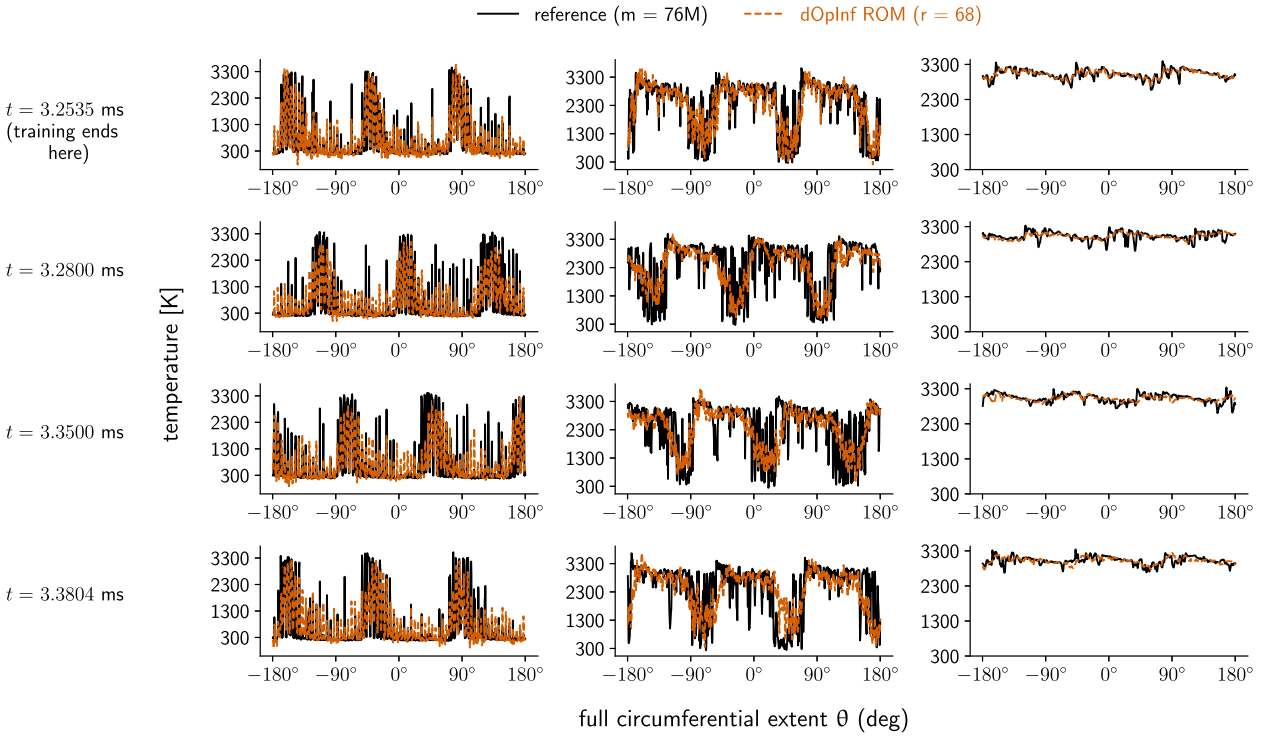


Fig. 7. One-dimensional circumferential profiles for temperature. The columns plot the results at three representative locations close to the mid-channel. The rows plot the profiles at four representative time instants.

ing reduced models for problems characterized by slowly decaying Kolmogorov n-widths. This can be addressed, for example, via quadratic manifolds [6,23], which can be extended to a distributed formulation following a similar approach to that presented here.

Domain decomposition also holds promise for parallel processing of large datasets [22]. However, methods like Operator Inference struggle with a high number of subdomains due to hyperparameter tuning complexity. Finding a balance between subdomain usage and accuracy, potentially through integration with distributed methods, could unlock significant efficiency gains.

CRediT authorship contribution statement

Ionuţ-Gabriel Farcaş: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Rayomand P. Gundevia:** Writing – original draft, Validation, Data curation, Conceptualization. **Ramakanth Munipalli:** Writing – original draft, Validation, Resources, Data curation, Conceptualization. **Karen E. Willcox:** Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Conceptualization.

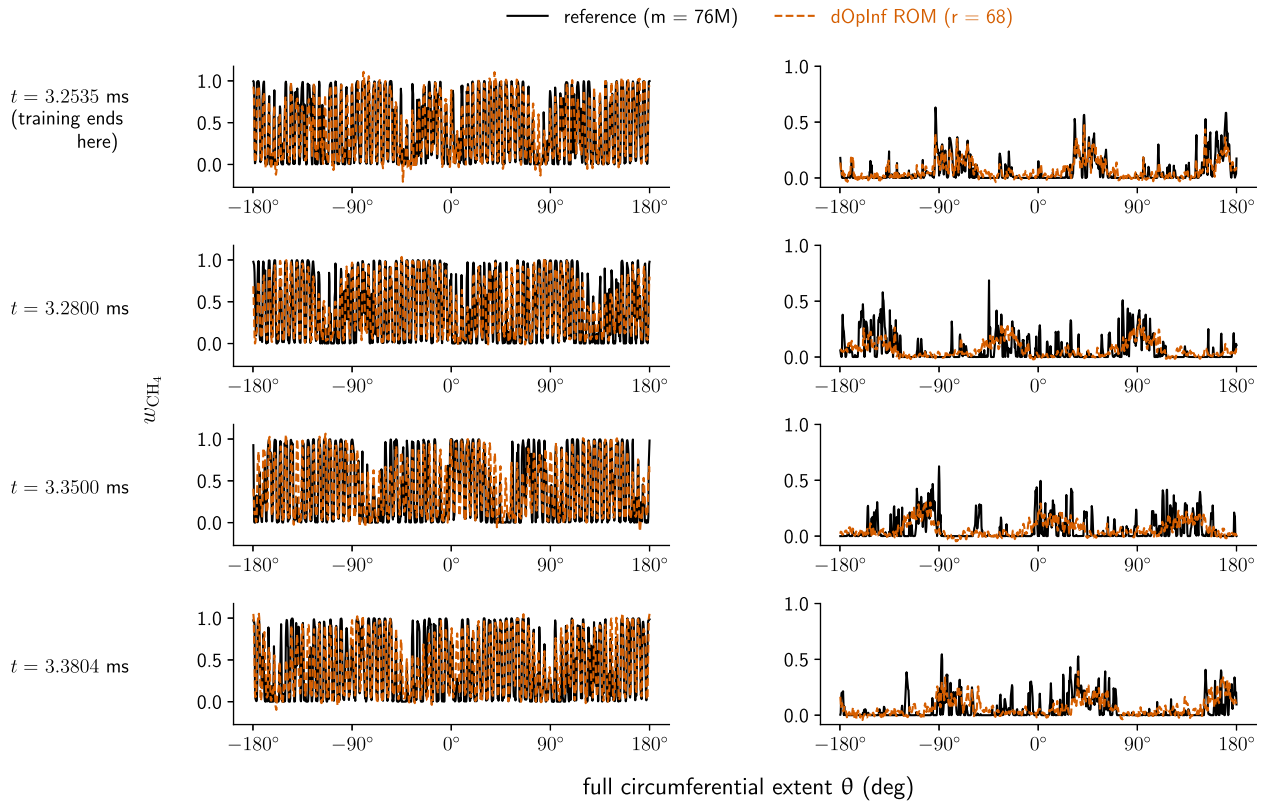


Fig. 8. One-dimensional circumferential profiles for fuel (CH_4) mass fraction. The columns plot the results at two representative locations close to the mid-channel. The rows plot the profiles at four representative time instants.

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 AFRL Grant FA9300-22-1-0001 and the Air Force Center of Excellence on Multifidelity Modeling of Rocket Combustor Dynamics under grant FA9550-17-1-0195. The authors gratefully acknowledge Jonathan Hoy who helped with transferring the high-fidelity dataset to TACC. The authors also gratefully acknowledge the compute and data resources provided by the Texas Advanced Computing Center at The University of Texas at Austin https://www.tacc.utexas.edu and the DoD High Performance Computing Modernization Program (HPCMP). The views expressed are those of the author and do not necessarily reflect the official policy or position of the Department of the Air Force, the Department of Defense, or the U.S. government.

Distribution Statement A: Approved for Public Release; Distribution is Unlimited. PA# AFRL-2024-111.

Data availability

The authors do not have permission to share data.

References

[1] S. Atchley, C. Zimmer, J. Lange, D. Bernholdt, V. Melesse Vergara, T. Beck, M. Brim, R. Budiardja, S. Chandrasekaran, M. Eisenbach, T. Evans, M. Ezell, N. Frontiere, A. Georgiadou, J. Glenski, P. Grete, S. Hamilton, J. Holmen, A. Huebl, D. Jacobson, W. Joubert, K. Mcmahon, E. Merzari, S. Moore, A. Myers, S. Nichols, S. Oral, T. Papatheodore, D. Perez, D.M. Rogers, E. Schneider, J.L. Vay, P.K. Yeung, Frontier: Exploring exascale, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Association for Computing Machinery, New York, NY, USA, 2023.

[2] T. Auckenthaler, V. Blum, H.J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, P. Willems, Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations, in: 6th International Workshop on Parallel Matrix Algorithms and Applications (PMAA'10), Parallel Comput. 37 (2011) 783–794, https://doi.org/10.1016/j.parco.2011.05.002, https://www.sciencedirect.com/science/article/pii/S0167819111000494.

[3] J. Axås, M. Cenedese, G. Haller, Fast data-driven model reduction for nonlinear dynamical systems, Nonlinear Dyn. 111 (2023) 7941–7957, https://doi.org/10.1007/s11071-022-08014-0.

[4] K. Azizzadenesheli, N. Kovachki, Z. Li, M. Liu-Schiaffini, J. Kossaifi, A. Anandkumar, Neural operators for accelerating scientific simulations and design, Nat. Rev. Phys. 6 (2024) 320–328, https://doi.org/10.1038/s42254-024-00712-5.

[5] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. 116 (2019) 15344–15349, https://doi.org/10.1073/pnas.1814058116.

[6] J. Barnett, C. Farhat, Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction, J. Comput. Phys. 464 (2022) 111348.

[7] A. Batista, M. Ross, C. Lietz, W.A. Hargus, Detonation Wave Interaction Classification in a Rotating Detonation Rocket Engine, in: AIAA Propulsion and Energy 2020 Forum, 2020.

[8] A. Batista, M.C. Ross, C. Lietz, W.A. Hargus, Descending Modal Transition Dynamics in a Large Eddy Simulation of a Rotating Detonation Rocket Engine, Energies 14 (2021), https://doi.org/10.3390/en14123387.

[9] C.A. Beattie, J. Borggaard, S. Gugercin, T. Iliescu, A Domain Decomposition Approach to POD, in: Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 6750–6756.

[10] E.A. Belli, J. Candy, I. Sfiligoi, Spectral transition of multiscale turbulence in the tokamak pedestal, Plasma Phys. Control. Fusion 65 (2022) 024001, https://doi.org/10.1088/1361-6587/aca9fa.

[11] P. Benner, P. Goyal, B. Kramer, B. Peherstorfer, K. Willcox, Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms, Comput. Methods Appl. Mech. Eng. 372 (2020) 113433, https://doi.org/10.1016/j.cma.2020.113433.

[12] P. Benner, S. Gugercin, K. Willcox, A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems, SIAM Rev. 57 (2015) 483–531, https://doi.org/10.1137/130932715.

[13] G. Berkooz, P. Holmes, J.L. Lumley, The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows, Annu. Rev. Fluid Mech. 25 (1993) 539–575, https://doi.org/10.1146/annurev.fl.25.010193.002543.

[14] R. Bird, N. Tan, S.V. Luedtke, S.L. Harrell, M. Taufer, B. Albright, VPIC 2.0: Next generation particle-in-cell simulations, IEEE Trans. Parallel Distrib. Syst. 33 (2021) 952–963.

[15] 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. 113 (2016) 3932–3937, https://doi.org/10.1073/pnas.1517384113.

[16] F. Bykovskii, S. Zhdan, E. Vedernikov, Continuous Spin Detonations, J. Propuls. Power 22 (2006) 1204–1216, https://doi.org/10.2514/1.17656.

[17] M. Cenedese, J. Axås, B. Bäuerlein, K. Avila, G. Haller, Data-driven modeling and prediction of non-linearizable dynamics via spectral submanifolds, Nat. Commun. 13 (2022) 872, https://doi.org/10.1038/s41467-022-28518-y.

[18] Y. Choi, W.J. Arrighi, D.M. Copeland, R.W. Anderson, G.M. Oxberry, libROM. [Computer Software], https://doi.org/10.11578/dc.20190408.3, 2019.

[19] J. Demmel, L. Grigori, M. Hoemmen, J. Langou, Communication-optimal Parallel and Sequential QR and LU Factorizations, SIAM J. Sci. Comput. 34 (2012) A206–A239, https://doi.org/10.1137/080731992.

[20] I. Farcas, R. Gundevia, R. Munipalli, K.E. Willcox, Parametric non-intrusive reduced-order models via operator inference for large-scale rotating detonation engine simulations, in: AIAA Scitech 2023 Forum, 2023.

[21] I. Farcas, R.P. Gundevia, R. Munipalli, K.E. Willcox, A Parallel Implementation of Reduced-Order Modeling of Large-Scale Systems, in: AIAA SCITECH 2025 Forum, 2025.

[22] I.G. Farcas, R.P. Gundevia, R. Munipalli, K.E. Willcox, Domain Decomposition for Data-Driven Reduced Modeling of Large-Scale Systems, AIAA J. (2024) 1–16, https://doi.org/10.2514/1.J063715.

[23] R. Geelen, S. Wright, K. Willcox, Operator inference for non-intrusive model reduction with quadratic manifolds, Comput. Methods Appl. Mech. Eng. 403 (2023) 115717, https://doi.org/10.1016/j.cma.2022.115717.

[24] G.H. Golub, C.F. Van Loan, Matrix Computations, Third ed., The Johns Hopkins University Press, 1996.

[25] A. Gouasmi, E.J. Parish, K. Duraisamy, A priori estimation of memory effects in reduced-order models of nonlinear systems using the Mori-Zwanzig formalism, Proc. Royal Soc. A, Math. Phys. Eng. Sci. 473 (2017) 20170385, https://doi.org/10.1098/rspa.2017.0385.

[26] P. Goyal, P. Benner, Generalized quadratic embeddings for nonlinear dynamics using deep learning, Phys. D: Nonlinear Phenom. 463 (2024) 134158, https://doi.org/10.1016/j.physd.2024.134158.

[27] N. Halko, P.G. Martinsson, J.A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, SIAM Rev. 53 (2011) 217–288, https://doi.org/10.1137/090771806.

[28] M.T. Henry de Frahan, L. Esclapez, J. Rood, N.T. Wimer, P. Mullowney, B.A. Perry, L. Owen, H. Sitaraman, S. Yellapantula, M. Hassanaly, M.J. Rahimi, M.J. Martin, O.A. Doronina, S.N. A., M. Rieth, W. Ge, R. Sankaran, A.S. Almgren, W. Zhang, J.B. Bell, R. Grout, M.S. Day, J.H. Chen, The Pele Simulation Suite for Reacting Flows at Exascale, in: Proceedings of the 2024 SIAM Conference on Parallel Processing for Scientific Computing, 2024, pp. 13–25.

[29] S. Hijazi, G. Stabile, A. Mola, G. Rozza, Data-driven POD-Galerkin reduced order model for turbulent flows, J. Comput. Phys. 416 (2020) 109513, https://doi.org/10.1016/j.jcp.2020.109513, https://www.sciencedirect.com/science/article/pii/S0021999120302874.

[30] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440, https://doi.org/10.1038/s42254-021-00314-5.

[31] D. Keyes, H. Ltaief, Y. Nakatsukasa, D. Sukkari, High-Performance SVD Partial Spectrum Computation, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Association for Computing Machinery, New York, NY, USA, 2023.

[32] B. Kramer, B. Peherstorfer, K.E. Willcox, Learning Nonlinear Reduced Models from Data with Operator Inference, Annu. Rev. Fluid Mech. 56 (2024) 521–548, https://doi.org/10.1146/annurev-fluid-121021-025220.

[33] J.N. Kutz, S.L. Brunton, B.W. Brunton, J.L. Proctor, Dynamic Mode Decomposition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

[34] A. Levy, M. Lindenbaum, Sequential Karhunen-Loeve basis extraction and its application to images, in: Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269), 1998, pp. 456–460, vol.2.

[35] A.G.M. Lewis, J. Beall, M. Ganahl, M. Hauru, S.B. Mallick, G. Vidal, Large-scale distributed linear algebra with tensor processing units, Proc. Natl. Acad. Sci. 119 (2022) e2122762119, https://doi.org/10.1073/pnas.2122762119.

[36] C. Lietz, Y. Desai, R. Munipalli, S.A. Schumaker, V. Sankaran, Flowfield analysis of a 3D simulation of a rotating detonation rocket engine, in: AIAA Scitech 2019 Forum, 2019.

[37] Managing I/O on TACC Resources, Texas Advanced Computing Center, 2024, https://docs.tacc.utexas.edu/tutorials/managingio/. (Accessed 15 December 2024).

[38] S.A. McQuarrie, C. Huang, K.E. Willcox, Data-driven reduced-order models via regularised Operator Inference for a single-injector combustion process, J. R. Soc. N.Z. 51 (2021) 194–211, https://doi.org/10.1080/03036758.2020.1863237.

[39] R. Milk, S. Rave, F. Schindler, pyMOR – Generic Algorithms and Interfaces for Model Order Reduction, SIAM J. Sci. Comput. 38 (2016) S194–S216, https://doi.org/10.1137/15M1026614.

[40] D. Papapicco, N. Demo, M. Girfoglio, G. Stabile, G. Rozza, The Neural Network shifted-proper orthogonal decomposition: A machine learning approach for non-linear reduction of hyperbolic equations, Comput. Methods Appl. Mech. Eng. 392 (2022) 114687, https://doi.org/10.1016/j.cma.2022.114687.

[41] B. Peherstorfer, Model Reduction for Transport-Dominated Problems via Online Adaptive Bases and Adaptive Sampling, SIAM J. Sci. Comput. 42 (2020) A2803–A2836, https://doi.org/10.1137/19M1257275.

[42] B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Comput. Methods Appl. Mech. Eng. 306 (2016) 196–215, https://doi.org/10.1016/j.cma.2016.03.025.

[43] E. Qian, I.G. Farcaş, K. Willcox, Reduced Operator Inference for Nonlinear Partial Differential Equations, SIAM J. Sci. Comput. 44 (2022) A1934–A1959, https://doi.org/10.1137/21M1393972.

[44] E. Qian, B. Kramer, B. Peherstorfer, K. Willcox, Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems, Phys. D: Nonlinear Phenom. 406 (2020) 132401, https://doi.org/10.1016/j.physd.2020.132401.

[45] F. Rizzi, P.J. Blonigan, E.J. Parish, K.T. Carlberg, Pressio: Enabling projection-based model reduction for large-scale nonlinear dynamical systems, https://arxiv.org/abs/2003.07798, arXiv:2003.07798, 2021.

[46] M. Rogowski, B.C. Yeung, O.T. Schmidt, R. Maulik, L. Dalcin, M. Parsani, G. Mengaldo, Unlocking massively parallel spectral proper orthogonal decompositions in the PySPOD package, Comput. Phys. Commun. (2024) 109246, https://doi.org/10.1016/j.cpc.2024.109246.

[47] T. Sayadi, P.J. Schmid, Parallel data-driven decomposition algorithm for large-scale datasets: with application to transitional boundary layers, Theor. Comput. Fluid Dyn. 30 (2016) 415–428, https://doi.org/10.1007/s00162-016-0385-x.

[48] H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proc. Royal Soc. A, Math. Phys. Eng. Sci. 473 (2017) 20160446.

[49] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28, https://doi.org/10.1017/S0022112010001217.

[50] P. Schwerdtner, P. Mohan, A. Pachalieva, J. Bessac, D. O'Malley, B. Peherstorfer, Online learning of quadratic manifolds from streaming data for nonlinear dimensionality reduction and nonlinear model reduction, https://arxiv.org/abs/2409.02703, 2024, arXiv:2409.02703.

[51] L. Sirovich, Turbulence and the dynamics of coherent structures part i: Coherent structures, Q. Appl. Math. 45 (1987) 561–571.

[52] Y. Tao, H. Wang, Foundational Fuel Chemistry Model Version 1.0 (FFCM-1), 2016.

[53] D. Stanzione, J. West, R.T. Evans, T. Minyard, O. Ghattas, D.K. Panda, Frontera: The Evolution of Leadership Computing at the National Science Foundation, in: Practice and Experience in Advanced Research Computing, Association for Computing Machinery, New York, NY, USA, 2020, pp. 106–111.

[54] R. Swischuk, B. Kramer, C. Huang, K. Willcox, Learning Physics-Based Reduced-Order Models for a Single-Injector Combustion Process, AIAA J. 58 (2020) 2658–2672, https://doi.org/10.2514/1.J058943.

[55] 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 (2014) 391–421.

[56] Z. Wang, B. McBee, T. Iliescu, Approximate partitioned method of snapshots for POD, J. Comput. Appl. Math. 307 (2016) 374–384, https://doi.org/10.1016/j.cam.2015.11.023.