Constrained Optimized Dynamic Mode Decomposition with Control for Physically Stable Systems with Exogeneous Inputs

Jacob Rains[1], Yi Wang[1]*, Alec House[2], Andrew L. Kaminsky[2], Nathan A. Tison[3], Vamshi M. Korivi[3]
[1]Mechanical Engineering, University of South Carolina, Columbia, SC
[2]CFD Research Corporation, Huntsville, AL
[3]US Army DEVCOM Ground Vehicle Systems Center, Warren, MI

ABSTRACT

Reduced order models (ROMs) generated by the dynamic mode decomposition (DMD) are sensitive to the selection of DMD hyperparameters and suffer from spurious eigenvalue issues. To address these challenges, this paper presents a novel method named constrained optimized DMD with Control (cOptDMDc) that extends optimized DMD to systems with exogenous inputs and can optionally enforce the stability of the resulting ROM. The variable projection and the exponential data fitting approaches are reformulated to consider temporal effects of exogenous inputs on state evolution. A new process combining iterative exponential data fitting and a linear inequality constraint is proposed to optimally place eigenvalues within the stable region, yielding a sub-optimal spectral structure, which is more robust and effective to mitigate spurious eigenvalue issues and alleviate the brittleness of DMDc. Three case studies are conducted to compare the proposed cOptDMDc with the native Dynamic Mode Decomposition with Control (DMDc) and Sparsity-Promoting DMDc (SPDMDc). The results prove that the cOptDMDc method is accurate with maximum relative error < 0.5% and more robust to the choice of ROM subspace dimensions, thus eliminating the need for a trial-and-error process and additional validation data for ROM construction.

NOTATION

A^*: Hermitian (conjugate) transpose of matrix A
A^T: Transpose (non-conjugate) of matrix A
A^+: Moore-Penrose pseudoinverse of A
$A \odot B$: Component-wise (Hadamard) matrix product of matrices A and B
$A(:)$: Vector obtained by stacking all columns of A on top of each other
$\mathrm{diag}(A)$: Vector obtained from the diagonal elements of A
$\bar{A}$: Component-wise complex conjugate of A
$A(i,:)$: the i^{th} row of A
$A(:,j)$: the j^{th} column of A
$\Re\{A\}$: the real part of A
$\Im\{A\}$: the imaginary part of A

I. INTRODUCTION

The Dynamic Mode Decomposition (DMD) is a data-driven, reduced-order model (ROM) approach to linearly approximate the relationship between two consecutive states of a time-varying system. It has found widespread implications in many engineering applications, in particular, fluid dynamics, and aerodynamics, among others, where it is often used to analyze the dynamic evolution of experimental

* Corresponding Author: Associate Professor, University of South Carolina, yiwang@cec.sc.edu

and computational fluid dynamics (CFD) data [1, 2]. The DMD reduces the complexity and dimensionality of the system through the use of the truncated singular value decomposition (SVD), relating it to proper orthogonal decomposition (POD), another ROM approach that is often used for model simplification or model order reduction [3, 4, 5]. The DMD is grounded in Koopman operator theory by approximating the infinite-dimensional linear Koopman operator with one of finite dimension [6].

The classic DMD is often computed using least squares involving the truncated SVD [7], although there exist a variety of extensions and variants that improve the classic formulation in certain aspects. Hemati et al. developed Total DMD (tlsDMD), which uses total least squares to correct the DMD's bias to sensor noise [8]. Jovanović et al. developed Sparsity-promoting DMD (SPDMD), which uses convex optimization with a regularization term to select the most dominant DMD modes [9] followed by another DMD procedure on the selected modes. Williams et al. developed a method of Extended DMD (EDMD) [10]. It employs an extra set of observables in an attempt to fit highly-nonlinear data [11]. Additionally, they determined an efficient way to evaluate this dictionary using the "kernel trick," which required far fewer operations than direct evaluation. Optimized DMD by Ashkam et al. applies the variable projection algorithms of Golub and LeVeque [12] to solve a nonlinear least-squares exponential data-fitting problem to both de-bias the DMD eigenvalues and fit unevenly-spaced snapshots [13].

When DMD is applied to data generated from the discretization of continuous linear operators, the appearance of "spurious" eigenvalues and eigenmodes pollute the operator spectrum [14]. These eigenvalues often have large growth rates and can cause model predictions to diverge. Colbrook and Townsend developed Residual DMD (ResDMD) to determine if an eigenmode is spurious by computing the squared relative residual of the identified Koopman operator's action on the candidate eigenmode [15].

Many dynamic systems also incorporate exogenous inputs that may affect the evolution of the state, such as the Linearized Navier Stokes (LNS) equations [16, 17, 18], which determine the effect of external perturbations on steady channel flow. The aeroelastic equations of motion [19] also define a relationship between external inputs and control inputs with the system state, and reformulating the aeroelastic equations into a state-space form aids in stability analysis of the system, since the eigenvalues of the system state matrix determine the onset of aeroelastic instability. When the underlying aeroelastic operators are unknown, classic system identification methods such as the N4SID method [20] may be used to extract the operators from known input-output measurements. Currently, multiple system identification methods have been successfully applied to aeroelastic ROMs, such as the Eigensystem realization algorithm (ERA) [21, 22], the AutoRegressive with eXogenous input (ARX) models [23, 24], and the state consistence enforced ARX for parametric ROMs [25, 26] among others.

The DMD has also been extended to dynamic systems with inputs, allowing for modal analysis of the dynamic system along with the identification of operators that may be used to predict and control the future states of the system [27]. The first such extension, DMD with control (DMDc) proposed by Proctor et al., solves for two (possibly low-dimensional) linear operators using least squares on an augmented snapshot matrix [28]. Annoni et al. extended DMDc to systems with input-output pairs and named it Input-output DMD (IODMD). It uses the POD modal coefficients as the state and then utilizes the N4SID method to identify four linear operators that relate the system state to its inputs and outputs [29]. Kou and Zhang improved DMDc for cases where data may be collected without actuation using DMD with exogenous data input (DMDX), which creates a DMD model for the intrinsic (non-actuated) dynamics and another DMDc model for the actuated dynamics using two sets of experiments/simulations [27]. Annoni and Seiler recast the DMDc within the framework of SPDMD and developed a sparsity-promoting DMDc (SPDMDc),

which reformulates the DMDc problem into one amenable to SPDMD, allowing the use of the same convex regularized optimization algorithms proposed in SPDMD to exclude extraneous modes from the final DMDc model [30], hence achieving parsimonious model structure. Extended DMD developed by Williams et al. was also generalized to systems with exogenous inputs by Proctor et al. in [31] to create Koopman with Inputs and Control. This allows for constructing ROMs that exhibit nonlinear characteristics while enabling the ROM to consider the effect of inputs on the system.

One potential issue arising from using DMD to predict future states is the effect of noise on the input data. If the data is too noisy, the eigenvalues computed by DMD or DMDc become biased towards the noise and the less relevant dynamics which can give rise to unexpected system behavior in the subsequent predictions. This problem is further accentuated when using too few snapshots to generate the DMD/DMDc ROMs, because the effect of the noise is more pronounced over the true underlying dynamics. In [13] Ashkam et al. showed the effect of noise bias on the eigenvalues, which causes a non-decaying system to decay, and in some opposite cases, a non-decaying system or a decaying system erroneously grows exponentially with time. The Optimized DMD of Ashkam et al. [13] effectively corrects for this bias by using nonlinear least-squares to fit a set of complex exponentials to the snapshot data. What makes Optimized DMD stand out is its ability to change the eigenvalues and eigenvectors of the identified DMD operator, while others simply choose a subset of the given DMD modes. The result is a model that does not necessarily identify the spectral properties as well as other DMD methods but provides a model that better fits the training data than the other methods.

This paper presents a new approach of constrained optimized DMD with control (named cOptDMDc hereafter) by extending the optimized DMD procedure to systems with exogenous inputs. The variable projection method [12, 13] is adopted as the primary computational framework to enable exponential data fitting and computationally effective determination of underlying spectral structures. Different from the optimized DMD, the regressor matrix of the exponential data fitting in cOptDMDc includes the contribution from the time series of exogenous inputs. Although more numerically challenging to solve, cOptDMDc does yield a new formulation to consider exogenous inputs in optimized DMD and their effects on state evolution. The second essential element in the proposed cOptDMDc is a new process to constrain the eigenvalues within the stability region to mitigate the issue associated with spurious eigenvalues generated by the native DMDc. The locations of the adjusted eigenvalues are found through an optimization process based on iterative exponential data fitting subject to a linear inequality constraint. Essentially, the algorithm produces a sub-optimal spectral structure relative to the conventional DMDc in data-fitting residual, which, however, is more effective in eliminating spurious eigen components and alleviating the brittleness of DMDc to snapshot data selection and configuration hyperparameters while retaining desirable model accuracy. Thus, the trial-and-error process and additional validation data to tune DMDc model construction can be potentially circumvented.

The novelties of the present work include (1) a new mathematical formulation and framework for extending Optimized DMD to systems with exogenous inputs, allowing the use of the variable projection and exponential data fitting methods to extract more robust spectral structures of ROMs for broad application scenarios with control; (2) a new process and algorithm, i.e., cOptDMDc that effectively constrains eigenvalues at optimal locations within the stability region, preventing erroneously exponentially-growing snapshot predictions. The eigenvalue locations are determined through an iterative procedure to achieve a balanced performance in two aspects: guaranteeing the system stability and preserving exponential data fitting accuracy. Furthermore, (3) the present cOptDMDc is developed in a more generic context, and its subprocess, in particular, the constraining procedure and constrained exponential data fitting, can be used for the native Optimized DMD.

It should also be pointed out that the present implementation of cOptDMDc is limited to inherently stable systems that widely exist in the real world, such as thermal and flow applications. Some multi-domain systems, such as aeroelasticity that couples aerodynamics with structural dynamics, could exhibit instability at certain operating regimes. However, the individual physics, when modeled separately, is stable, and thus, the proposed cOptDMDc is still applicable. In addition, similar to DMDc and its variants, when subject to noisy data, our cOptDMDc is most likely not able to reveal the exact dynamics underlying the original physical system. However, it captures the dominant structures with prominent data fitting accuracy for enhanced robustness of ROM construction and prediction.

This paper is organized as follows: Section II will describe derivations and algorithms for the construction of DMDc and SPDMDc followed by the development of the Optimized DMDc and the eigenvalue constraint method. Section III will demonstrate case studies using a linearized Navier-Stokes equation and a 2D and 3D vehicle model undergoing transient heat conduction. The ROM predictions made by cOptDMDc will be compared to DMDc and SPDMDc. Cases where DMDc and SPDMDc fail, will be analyzed in depth, and the associated explanations will be provided. These will be contrasted to the Optimized DMDc, which performs very well in these cases. Section IV will summarize the results obtained from the Optimized DMDc and list potential avenues for future research.

II. METHODS

This section will present the method and the derivation of the proposed cOptDMDc. Section II-A briefly introduces the fundamentals of existing DMDc, which provides the basis of SPDMDc and cOptDMDc, and Section II-B describes SPDMDc, which is one of the main alternative modifications to standard DMDc. Section II-C presents the derivation of our cOptDMDc and shows how the DMDc problem can be recast into a form that allows the use of the same variable projection algorithms used in the construction of Optimized DMD. Section II-D elucidates the modification to the Levenberg-Marquardt update used in Optimized DMD and cOptDMDc to constrain the eigenvalues while maintaining necessary data fitting accuracy and shows how the resulting constrained least-squares problem can be solved in real arithmetic.

A. Dynamic Mode Decomposition with Control

Suppose there exists a dynamic system with a state vector $x_{k-1} \in \mathbb{C}^m$ and a corresponding input vector $u_{k-1} \in \mathbb{C}^p$ in the current time instance $k-1$, and thus, the next state $x_k \in \mathbb{C}^m$ may be determined from a function f of the current state and current input. This relationship may be represented as

$$x_k = f(x_{k-1}, u_{k-1}). \tag{1}$$

Assume that f may be well-approximated by a linear state-space model

$$x_k = Ax_{k-1} + Bu_{k-1}, \tag{2}$$

for n observed snapshots, where $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{m\times p}$ are constant state-space matrices. This model assumes the relationship between two consecutive snapshots is linear, but not that total evolution of the system is linear. It can be summarized for all n observations in a matrix form [28], viz.,

$$X_1 = AX_0 + B\Gamma_0, \tag{3}$$

where $X_0 = [x_1 \quad \cdots \quad x_{n-1}] \in \mathbb{C}^{m\times(n-1)}$, $X_1 = [x_2 \quad \cdots \quad x_n] \in \mathbb{C}^{m\times(n-1)}$, and $\Gamma_0 = [u_1 \quad \cdots \quad u_{n-1}] \in \mathbb{C}^{p\times(n-1)}$. A and B in Eq. (3) may then be determined by solving the least-squares problem [28]

$$A, B = \underset{A,B}{\operatorname{argmin}} \|X_1 - (AX_0 + B\Gamma_0)\|_2 \tag{4}$$

or,

$$[A \quad B] = \underset{[A \quad B]}{\operatorname{argmin}} \left\| X_1 - [A \quad B] \begin{bmatrix} X_0 \\ \Gamma_0 \end{bmatrix} \right\|_2. \tag{5}$$

Once A and B are determined, this model can be used to predict future states or can be analyzed to determine the underlying dynamics of the system.

One of the main benefits of the DMD and its related variants is its ability to reduce the dimensionality of the snapshot data. This can be attained using the truncated SVD, which finds the optimal rank-r approximation of the full-order snapshot matrix with respect to the 2-norm. Assume that the snapshot matrix X_0 is well-approximated by the truncated SVD

$$X_0 \approx U_r \Sigma_r V_r^* \tag{6}$$

with truncation dimension r, where $U_r \in \mathbb{C}^{m\times r}$ and $V_r \in \mathbb{C}^{n\times r}$ are, respectively, the matrices of the r left and right singular vectors of X_0 corresponding to the r largest singular values of X_0, and $\Sigma_r \in \mathbb{R}^{r\times r}$ is the diagonal matrix with the r largest singular values of X_0. Then the full-dimension least-squares problem in Eq. (5) may be approximated by a solution in a low-dimension subspace spanned by the columns of U_r. This is obtained with a new set of lower-dimensional snapshot matrices found by projecting the snapshot matrices X_0 and X_1 onto U_r, giving

$$\mathrm{H}_0 = [\eta_1 \quad \cdots \quad \eta_{n-1}] = U_r^* X_0 \in \mathbb{C}^{r\times(n-1)}, \qquad \mathrm{H}_1 = [\eta_2 \quad \cdots \quad \eta_n] = U_r^* X_1 \in \mathbb{C}^{r\times(n-1)}.$$

The full-dimension snapshot matrices X_0 and X_1 can be approximately reconstructed from H_0 and H_1 using $X_0 \approx U_r \mathrm{H}_0$ and $X_1 \approx U_r \mathrm{H}_1$, and this reconstruction becomes more accurate with increasing r. These relations are then substituted into Eq. (3), yielding

$$\mathrm{H}_1 = (U_r^* A U_r)\mathrm{H}_0 + (U_r^* B)\Gamma_0. \tag{7}$$

Defining $F = U_r^* A U_r \in \mathbb{C}^{r\times r}$ and $G = U_r^* B \in \mathbb{C}^{r\times p}$ as the low-dimensional forms of A and B, the solution to the low-dimension form of Eq. (5) is obtained by

$$[F \quad G] = \underset{[F \quad G]}{\operatorname{argmin}} \left\| U_r^* X_1 - [F \quad G] \begin{bmatrix} \Sigma_r V_r^* \\ \Gamma_0 \end{bmatrix} \right\|_2. \tag{8}$$

Eq. (8) can then be solved using the standard linear least squares approach by

$$[F \quad G] = U_r^* X_1 \begin{bmatrix} \Sigma_r V_r^* \\ \Gamma_0 \end{bmatrix}^+. \tag{9}$$

After obtaining F and G from Eq. (9), the state space representation in the low-dimensional space can then be constructed by,

$$\eta_k = F\eta_{k-1} + Gu_{k-1}, \tag{10}$$

Eq. (10) is essentially a reduced order model (ROM) and evaluating the reduced states η_k is computationally efficient, as all calculations are performed in the r-dimensional subspace. Once η_k is available, the full-order state can be reconstructed using $x_k \approx U_r\eta_k$. Eq. (10) is a recursive relationship and can be rewritten in terms of the initial state η_1 as

$$\eta_k = F^{k-1}\eta_1 + \sum_{j=1}^{k-1} F^{k-j-1}Gu_j. \tag{11}$$

To analyze the system dynamics of the reduced system, eigenvalue decomposition can be applied to F, viz., $F = \mathrm{Z}\mathrm{M}\hat{\mathrm{Z}}^*$, where $\mathrm{Z} = [\zeta_1 \quad \cdots \quad \zeta_r]$, $\mathrm{M} = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_r \end{bmatrix}$, $\hat{\mathrm{Z}} = [\hat{\zeta}_1 \quad \cdots \quad \hat{\zeta}_r] = (\mathrm{Z}^{-1})^*$.

Then Eq. (11) may be rewritten as

$$\eta_k = \sum_{i=1}^{r} \zeta_i \mu_i^{k-1} \hat{\zeta}_i^* \eta_1 + \sum_{j=1}^{k-1}\sum_{i=1}^{r} \zeta_i \mu_i^{k-j-1} \hat{\zeta}_i^* G u_j. \tag{12}$$

If the modal amplitudes $\vec{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_r \end{bmatrix} \in \mathbb{C}^{r\times 1}$ are defined, where $\alpha_i = \hat{\zeta}_i^*\eta_1$, $\beta^* = \begin{bmatrix} \vec{\beta}_1^* \\ \vdots \\ \vec{\beta}_r^* \end{bmatrix} \in \mathbb{C}^{r\times p}$, and $\vec{\beta}_i^* = \hat{\zeta}_i^* G \in \mathbb{C}^{1\times p}$, then

$$\eta_k = \sum_{i=1}^{r} \zeta_i \mu_i^{k-1} \alpha_i + \sum_{j=1}^{k-1}\sum_{i=1}^{r} \zeta_i \mu_i^{k-j-1} \vec{\beta}_i^* u_j. \tag{13}$$

This particular formulation of DMDc [30] will be used for describing SPDMDc and our cOptDMDc in later sections.

B. Sparsity-Promoting Dynamic Mode Decomposition with Control

One difficulty presented by DMDc is the choice of the truncation rank r. If r is too small, then the accuracy of the resultant ROM suffers as it is not capable of representing the dynamics of the system. If r is too large, then the computational acceleration provided by performing computations in the r-dimension subspace is compromised, and the ROM accuracy may even deteriorate as well [29]. SPDMDc aims to solve this problem by augmenting Eq. (8) with a sparsity regularization term that excludes certain extraneous DMDc modes based on the magnitudes of their corresponding amplitudes α_i and $\vec{\beta}_i^*$. If a set of modal amplitudes are determined to be too small, they are set to zero. Following determination of the truncated modes, the data fitting process is resumed on the remaining α_i and $\vec{\beta}_i^*$, and thus, the difference in state predictions by the old and the new models is minimized. SPDMDc begins with the summation

representation of the DMDc model found in Eq. (13). Defining $\theta_i = \begin{bmatrix} \alpha_i \\ \vec{\beta}_i \end{bmatrix} \in \mathbb{C}^{(p+1)\times 1}$, $q_i^*(k) = \begin{bmatrix} \mu_i^k & \sum_{j=1}^{k-1} \mu_i^{k-j-1} u_j^* \end{bmatrix} \in \mathbb{C}^{1\times(p+1)}$, and factoring out ζ_i from Eq. (13) reads

$$\eta_k = \mathrm{Z} \sum_{i=1}^{r} q_i^*(k)\, \theta_i . \tag{14}$$

Stacking the snapshots into one column, i.e., $\widetilde{\mathrm{H}}_0 = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_k \end{bmatrix} \in \mathbb{C}^{kr\times 1}$ yields

$$\widetilde{\mathrm{H}}_0 = Q\vec{\theta}, \tag{15}$$

where

$$Q = \begin{bmatrix} \zeta_1 q_1^*(1) & \cdots & \zeta_r q_r^*(1) \\ \vdots & \ddots & \vdots \\ \zeta_1 q_1^*(k) & \cdots & \zeta_r q_r^*(k) \end{bmatrix} \in \mathbb{C}^{kr\times(p+1)r}, \qquad \vec{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_r \end{bmatrix} \in \mathbb{C}^{(p+1)r\times 1}.$$

The standard DMDc determines $\vec{\theta}$ from the optimization problem $\vec{\theta} = \underset{\vec{\theta}}{\operatorname{argmin}} L(\vec{\theta})$, where $L(\vec{\theta}) = \left\| \widetilde{\mathrm{H}}_0 - Q\vec{\theta} \right\|_2$. In contrast, SPDMDc adds a regularization term to promote the sparsity of $\vec{\theta}$,

$$\vec{\theta}_{SP} = \underset{\vec{\theta}}{\operatorname{argmin}} L(\vec{\theta}) + \lambda \sum_{i=1}^{r} \|\theta_i\|_2 \tag{16}$$

for the user-defined regularization hyperparameter λ. The optimization problem in Eq. (16) is solved using the Alternating Direction Method of Multipliers (ADMM), and the resulting $\vec{\theta}_{SP}$ is used to determine the optimal sparsity pattern of $\vec{\theta}$ based on λ. After the optimal sparsity pattern is determined, the constrained quadratic programming problem

$$\begin{aligned} &\underset{\vec{\theta}}{\operatorname{argmin}} L(\vec{\theta}) \\ &\text{subject to} \quad E^{\mathrm{T}}\vec{\theta} = 0 \end{aligned} \tag{17}$$

is solved for $\vec{\theta}$, where $E^T\vec{\theta}$ yields the components of $\vec{\theta}$ that correspond to the zero entries of $\vec{\theta}_{SP}$. After Eq. (17) is solved, Eq. (14) then can be used to determine the evaluation of the low-dimensional states. SPDMDc determines the optimal number of modes and the values of the modal amplitudes based on a given λ. A larger value of λ promotes a sparser solution, while a smaller value of λ allows a better fit to the data. In other words, SPDMDc enables a parsimonious representation of the DMDc model.

In the presence of clean data, SPDMDc can generate an accurate, low-order ROM that maintains much of the accuracy of DMDc while retaining fewer unnecessary modes. In the presence of noisy data, however, the eigenvalues may be misidentified by DMDc, which may cause the SPDMDc procedure to fail. If a misidentified eigenvalue lies outside the complex unit circle (corresponding to a positive growth rate), the construction of the SPDMDc ROM would fail when it tries to construct the Q matrix from Eq. (15). Raising one of these "spurious" eigenvalues to a large power could cause the bottom rows of Q to take on NaN

or Inf values due to the limits of floating-point computation. Since this error occurs before the ADMM optimization takes place, SPDMDc may not be able to reject it, leading to the failure of ROM generation. On the other hand, DMDc would not fail and at least produces a model, although it would misidentify the spectrum and its predictions would be wrong.

C. Optimized Dynamic Mode Decomposition with Control

In [13] Ashkam et al. developed Optimized DMD, which uses the nonlinear least squares approach to fit complex exponentials to the snapshot data. It aims to address the erroneous DMD mode issues from a new perspective by allowing for the complete modification of the dynamic modes and their eigenvalues, as opposed to only deciding if a DMDc mode (and its eigenvalue) should be retained or not. This allows Optimized DMD to fit unevenly-spaced snapshot data as well. In this context, it is highly desirable to extend the Optimized DMD functionality to DMDc (with exogeneous inputs) and extract accurate DMDc modes and eigenvalues of actuated dynamic systems.

If a DMDc model of the snapshot data already exists, it is possible to begin from Eq. (13) and recast it into a continuous-time form using the identity $\mu_i^k = e^{\gamma_i t_k}$, where $\gamma_i = \log(\mu_i)/\Delta t$, t_k is the time corresponding to the k^{th} snapshot, and Δt is the timestep used to generate the snapshots. This relation may be substituted back into Eq. (13), yielding

$$\eta_k = \sum_{i=1}^{r} \zeta_i e^{\gamma_i t_{k-1}} \alpha_i + \sum_{j=1}^{k-1} \sum_{i=1}^{r} \zeta_i e^{\gamma_i t_{k-j-1}} \vec{\beta}_i^* u_j. \tag{18}$$

Eq. (18) may be further simplified into a matrix form, leading to

$$\eta_k = \mathrm{Z}\mathrm{diag}\big(\chi_{k-1}\Phi_{k-1}(\vec{\gamma})\big), \tag{19}$$

Where $\chi_{k-1} = [\vec{\alpha} \quad \beta^*\Gamma_0(:,1:k-1)] \in \mathbb{C}^{r\times k}$, $\Phi_{k-1}(\vec{\gamma}) = e^{\vec{t}_{k-1}\vec{\gamma}^T} \in \mathbb{C}^{k\times r}$, $\vec{t}_{k-1} = \begin{bmatrix} t_{k-1} \\ \vdots \\ t_1 \\ t_0 \end{bmatrix}$, $t_0 = 0$. The details of deriving Eq. (19) are provided in Appendix A. With Eq. (19) the columns of H_0 may be constructed, which is

$$\mathrm{H}_0 = \big[\mathrm{Z}\mathrm{diag}\big(\chi_1\Phi_1(\vec{\gamma})\big) \quad \cdots \quad \mathrm{Z}\mathrm{diag}\big(\chi_{n-1}\Phi_{n-1}(\vec{\gamma})\big)\big]. \tag{20}$$

Taking the non-conjugate transpose of both sides to find H_0^T yields

$$\mathrm{H}_0^T = \begin{bmatrix} \mathrm{diag}\big(\chi_1\Phi_1(\vec{\gamma})\big)^T \\ \vdots \\ \mathrm{diag}\big(\chi_{n-1}\Phi_{n-1}(\vec{\gamma})\big)^T \end{bmatrix} \mathrm{Z}^T. \tag{21}$$

The purpose of taking the non-conjugate transpose is only to change the shape of the matrix, so that the problem of determining the snapshots H_0^T may be reduced to a minimization problem

$$\min_{\vec{\gamma},\Omega} \left\| \mathrm{H}_0^T - \Psi\left(\vec{\gamma}, \vec{t}_{n-1}\right)\Omega \right\|_F^2 \tag{22}$$

for a matrix-valued function $\Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right)$ and a matrix Ω. This allows the use of variable projection algorithms [12, 13] to find a solution of Eq. (22) by defining $\Omega = \Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right)^{+} \mathrm{H}_0^T$ and solving

$$\min_{\vec{\gamma}} \left\| \mathrm{H}_0^T - \Psi\left(\vec{\gamma}, \vec{t}_{n-1}\right)\Psi\left(\vec{\gamma}, \vec{t}_{n-1}\right)^{+} \mathrm{H}_0^T \right\|_F^2 \tag{23}$$

in $\vec{\gamma}$ alone. Once the optimal $\vec{\gamma}$ is determined, the future snapshots may be predicted by

$$\eta_k = \Omega^T \Psi(\vec{\gamma}, t_k)^T. \tag{24}$$

Eqs (18)-(21) represent DMDc and Eqs. (22) – (24) are the optimized DMD formulation. The combination of both extends the functionality of Optimized DMD to systems with exogenous inputs. Since Eq. (21) is in the desired form, the expression for $\Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right)$ may be obtained as

$$\Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right) = \begin{bmatrix} \mathrm{diag}\left(\chi_1 \Phi_1(\vec{\gamma})\right)^T \\ \vdots \\ \mathrm{diag}\left(\chi_{k-1} \Phi_{k-1}(\vec{\gamma})\right)^T \end{bmatrix} \in \mathbb{C}^{(k-1)\times r}. \tag{25}$$

Then the solution of the optimization problem in Eq. (23) readily follows from [13]. First, calculate $\Omega = \Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right)^{+} \mathrm{H}_0^T \in \mathbb{C}^{\mathrm{r}\times\mathrm{r}}$ and define

$$R = R(\vec{\gamma}) = \mathrm{H}_0^T - \Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right)\Omega,$$

Then, Eq. (23) becomes

$$\min_{\vec{\gamma}} \|R(\vec{\gamma})\|_F^2.$$

It is a nonlinear least-squares optimization problem with multiple right-hand sides similar to those in [12]. It may be solved by first defining $\vec{\rho}(\vec{v}) = R(:)$, and then, the problem becomes

$$\min_{\vec{\gamma}} \|\vec{\rho}(\vec{\gamma})\|_2^2. \tag{26}$$

Eq. (26) can be computed using the nonlinear Levenberg-Marquardt algorithm, which iteratively searches for the update $\vec{\delta}^{(i)}$ so that

$$\vec{\gamma}^{(i+1)} = \vec{\gamma}^{(i)} - \vec{\delta}^{(i)}. \tag{27}$$

The update $\vec{\delta}^{(i)}$ is solved by

$$\vec{\delta}^{(i)} = \underset{\vec{\delta}^{(i)}}{\operatorname{argmin}} \left\| \begin{pmatrix} J\left(\vec{\gamma}^{(i)}\right) \\ \nu^{(i)} M\left(\vec{\gamma}^{(i)}\right) \end{pmatrix} \vec{\delta}^{(i)} - \begin{pmatrix} \vec{\rho}\left(\vec{\gamma}^{(i)}\right) \\ 0 \end{pmatrix} \right\|_2^2, \tag{28}$$

where $J\left(\vec{\gamma}^{(i)}\right) \in \mathbb{C}^{(k-1)r \times r}$ is the Jacobian matrix of $\vec{\rho}(\vec{\gamma})$ evaluated at $\vec{\gamma}^{(i)}$; $M\left(\vec{\gamma}^{(i)}\right)$ is a diagonal matrix with entries corresponding to the 2-norm of the columns of $J(\vec{\gamma}_i)$, viz., $M_{jj} = \left\| J\left(\vec{\gamma}^{(i)}\right)(:, j) \right\|_2$, and $\nu^{(i)}$ is an adjustable parameter used by the Levenberg-Marquardt algorithm to promote faster convergence when possible. The results in [13] may be used to determine $J\left(\vec{\gamma}^{(i)}\right)$, since their derivation was for a general differentiable function $\Phi(\vec{\alpha})$. They also showed that the columns of the Jacobian may be block-calculated, meaning $J\left(\vec{\gamma}^{(i)}\right)(:, j) = J_j^{mat}(:)$, where

$$J_j^{mat} = -\left((I - \Psi\Psi^+) \frac{\partial \Psi}{\partial \gamma_j} \Psi^+ + \left[(I - \Psi\Psi^+) \frac{\partial \Psi}{\partial \gamma_j} \Psi^+ \right]^* \right) \mathrm{H}_0^T \in \mathbb{C}^{(k-1) \times r}. \tag{29}$$

Taking the SVD of Ψ, viz., $\Psi = U\Sigma V^*$, this equation simplifies significantly to

$$J_j^{mat} = -(I - UU^*) D_j \Omega - U\Sigma^{-1} V^* D_j^* R, \tag{30}$$

Where $D_j = \frac{\partial \Psi}{\partial \gamma_j}$. The (p, q) component of D_j may be calculated as

$$D_j(p, q) = \begin{cases} 0, & q \neq j \\ \chi_p(j, :) \left(\vec{t}_p \odot \Phi_p\left(\gamma_j^{(i)}\right) \right), & q = j \end{cases}. \tag{31}$$

Details on deriving this equation can be found in Appendix B. It can be seen from this equation that D_j is a sparse matrix with only one nonzero column which, as in [13], gives the optimal order of computing Eq. (30) to preserve sparsity:

$$J_j^{mat} = -\left(D_j - U\left(U^* D_j\right) \right) \Omega - U\left(\Sigma^{-1} \left(V^* \left(D_j^* R \right) \right) \right). \tag{32}$$

Again, these results match those of [13], with the only difference appearing in the computation of D_j because our Ψ includes the contribution from the exogenous inputs. Once the Jacobian matrix has been constructed, Eq. (28) may be solved for $\vec{\delta}^{(i)}$, followed by updating $\vec{\gamma}^{(i)}$ with $\vec{\gamma}^{(i+1)} = \vec{\gamma}^{(i)} - \vec{\delta}^{(i)}$. This new $\vec{\gamma}^{(i+1)}$ may be evaluated to determine $\nu^{(i+1)}$ depending on the particular implementation of the Levenberg-Marquardt algorithm to improve convergence. Once $\left\| \vec{\rho}\left(\vec{\gamma}^{(i)}\right) \right\|_2^2$ satisfactorily converges, the low-dimensional solutions may be determined from Eq. (24).

It is important to note that the χ matrix is dependent on the particular choice of inputs Γ_0, as shown by Eq. (19), and therefore, the choice of inputs may affect the optimization process. However, in [13] the Jacobian matrix is independent of the training-specific contributions. It would be highly preferrable in our cOptDMDc that the χ matrix can be separated from the Jacobian matrix, and hence, it contains no training data. One such derivation is given in Appendix C. Unfortunately, there are several computational challenges with such a derivation, which are also detailed in Appendix C, precluding us from further development along this direction.

D. Constrained Optimized Dynamic Mode Decomposition with Control (cOptDMDc)

The results from cOptDMDc can be used to find the values of $\vec{\gamma}$ that best fit the snapshots, but do not guarantee that the resulting system is stable. If there is a γ_j that is $\Re\{\gamma_j\} > 0$ in the continuous domain, then the corresponding complex exponential $e^{\gamma_j t_k}$ grows with t_k. This phenomenon occurs for standard DMDc as well when the eigenvalues of the F matrix lie outside of the unit circle in the complex plane. This is because the eigenvalues are raised to successively higher powers when Eq. (13) is evaluated, causing uncontrolled growth in these particular eigenvalues and system instability. SPDMDc also suffers from the issue, as evaluating the Q matrix in Eq. (15) requires raising the eigenvalues to high powers during the construction of the optimization problem. This could potentially run into floating-point precision errors as the bottom rows of Q tend to infinite values.

In contrast, constrained cOptDMDc, i.e., cOptDMDc modifies the above cOptDMDc procedure to prevent these problems from occurring by constraining $\Re\{\vec{\gamma}^{(i)}\}$, the real part of $\hat{\gamma}^{(i)}$, to be less than or equal to 0 (if in the continuous time). Its procedure is given as follows:

- Set $\vec{\gamma}^{(1)} \leftarrow \min\left(\Re\{\vec{\gamma}^{(1)}\}, 0\right) + \Im\{\vec{\gamma}^{(1)}\}i$, where min is performed component-wise. In discrete time, this is equivalent to radially projecting any eigenvalues that lie outside the unit circle in the complex plane back onto the boundary of the unit circle. In continuous time, this is equivalent to shifting any eigenvalues that lie to the right of the imaginary axis to the imaginary axis.
- Solve Eq. (28) for exponential data fitting again with linear inequality constraints on $\vec{\delta}^{(i)}$ so that $\Re\{\vec{\gamma}^{(i)} - \vec{\delta}^{(i)}\} \leq 0$. This gives the constraint $\Re\{\vec{\delta}^{(i)}\} \geq \Re\{\vec{\gamma}^{(i)}\}$, and Eq. (28) becomes

$$\vec{\delta}^{(i)} = \underset{\vec{\delta}^{(i)}}{\operatorname{argmin}} \left\| \begin{pmatrix} J(\vec{\gamma}^{(i)}) \\ \nu^{(i)} M(\vec{\gamma}^{(i)}) \end{pmatrix} \vec{\delta}^{(i)} - \begin{pmatrix} \vec{\rho}(\vec{\gamma}^{(i)}) \\ 0 \end{pmatrix} \right\|_2^2. \quad (33)$$
$$\text{subject to } \Re\{-\vec{\delta}^{(i)}\} \leq \Re\{-\vec{\gamma}^{(i)}\}$$

These two modifications are the foremost innovations of the present cOptDMDc algorithm. First, it prevents $\Re\{\vec{\gamma}^{(i)}\}$ from becoming greater than zero, resulting in a stable model. Second, the iterative exponential data fitting allows eigenvalues relocated at optimal locations to still preserve data fitting accuracy.

Several constrained linear least-squares solvers, like MATLAB's lsqlin function, do not support computations involving complex numbers, and therefore, for generality the optimization problem in Eq. (33) must be recast into a form involving only real numbers. This is performed by defining S and T, i.e., $\begin{pmatrix} J(\vec{\gamma}^{(i)}) \\ \nu^{(i)} M(\vec{\gamma}^{(i)}) \end{pmatrix} = S + Ti$ and defining $\vec{a}$ and $\vec{b}$, so that $\vec{\delta}^{(i)} = \vec{a} + \vec{b}i$. Then Eq. (33) becomes

$$\vec{a}, \vec{b} = \underset{\vec{a},\vec{b}}{\operatorname{argmin}} \left\| (S + Ti)(\vec{a} + \vec{b}i) - \begin{pmatrix} \vec{\rho}(\vec{\gamma}^{(i)}) \\ 0 \end{pmatrix} \right\|_2^2, \quad (34)$$
$$\text{subject to } -\vec{a} \leq \Re\{-\vec{\gamma}^{(i)}\}$$

which simplifies to

$$\begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} = \underset{\begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix}}{\operatorname{argmin}} \left\| \left(\begin{bmatrix} S & -T \end{bmatrix} \begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} + \begin{bmatrix} T & S \end{bmatrix} \begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} i \right) - \begin{pmatrix} \vec{\rho}\left(\vec{\gamma}^{(i)}\right) \\ 0 \end{pmatrix} \right\|_2^2 . \quad \text{subject to } \begin{bmatrix} -I & 0 \end{bmatrix} \begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} \leq \Re\{-\vec{\gamma}^{(i)}\} \tag{35}$$

Grouping together the real and complex parts yields

$$\begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} = \underset{\begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix}}{\operatorname{argmin}} \left\| \begin{bmatrix} S & -T \\ T & S \end{bmatrix} \begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} - \begin{bmatrix} \Re\left\{ \begin{pmatrix} \vec{\rho}\left(\vec{\gamma}^{(i)}\right) \\ 0 \end{pmatrix} \right\} \\ \Im\left\{ \begin{pmatrix} \vec{\rho}\left(\vec{\gamma}^{(i)}\right) \\ 0 \end{pmatrix} \right\} \end{bmatrix} \right\|_2^2 , \quad \text{subject to } \begin{bmatrix} -I & 0 \end{bmatrix} \begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix} \leq \Re\{-\vec{\gamma}^{(i)}\} \tag{36}$$

which can be solved by using any pre-existing constrained linear least-squares solver, such as LSQLIN in MATLAB. After $\vec{\delta}^{(i)} = \vec{a} + \vec{b}i$ is determined, the regular procedure of COptDMDc presented in Section II-C may continue.

Although cOptDMDc may prevent finding a global minimum of Eq. (26) during data fitting, it does guarantee that the model returned is stable, and the new spectral structure obtained through constrained exponential data fitting can be considered sub-optimal. This provides a more robust model, as it is less affected by the number of retained modes compared to other methods and does not need additional validation data or the trial-and-error process to tune the number of retained modes. This particular constraining procedure may be used on Optimized DMD as well since the minimization problem in Eq. (28) is the same for both Optimized DMD and COptDMDc.

Algorithm 1: cOptDMDc

Given: X_0, X_1, Γ_0
Compute $F, G = DMDc(X_0, X_1, \Gamma_0)$
Compute $\mathrm{H}_0 = U^* X_0$
Find $\mu, Z = eig(F)$
Compute $\vec{\gamma}^{(1)} = \log(\mu)/\Delta t$
Constrain $\vec{\gamma}^{(1)} = \min\left(\Re\{\vec{\gamma}^{(1)}\}, 0\right) + \Im\{\vec{\gamma}^{(1)}\} i$
Compute $\Psi\left(\vec{\gamma}^{(1)}, \vec{t}_{k-1}\right)$, $\Omega = \Psi^{+}\mathrm{H}_0^T$, $R = \mathrm{H}_0^T - \Psi\left(\vec{\gamma}^{(1)}, \vec{t}_{k-1}\right)\Omega$, and $\rho = R(:)$
Determine $res_1 = \|\rho\|_2$ for convergence
Set restarts = 0
for $i = 1,2, \ldots, MaxIter$ **do**
  Find $U_\Psi, \Sigma_\Psi, V_\Psi = svd(\Psi)$
  for $j = 1,2, \ldots, r$ **do**
    Compute $J_j^{mat} = -\left(D_j - U_\Psi\left(U_\Psi^* D_j\right)\right)\Omega - U_\Psi\left(\Sigma_\Psi^{-1}\left(V_\Psi^*\left(D_j^* R\right)\right)\right)$
    Assign $J\left(\vec{\gamma}^{(i)}\right)(:, j) \leftarrow J_j^{mat}(:)$
  end
  Solve

$$\vec{\delta}^{(i)} = \underset{\vec{\delta}^{(i)}}{\operatorname{argmin}} \left\| \begin{pmatrix} J(\vec{\gamma}^{(i)}) \\ \nu^{(i)} M(\vec{\gamma}^{(i)}) \end{pmatrix} \vec{\delta}^{(i)} - \begin{pmatrix} \vec{\rho}(\vec{\gamma}^{(i)}) \\ 0 \end{pmatrix} \right\|_2^2$$

$$\text{subject to } \Re\{-\vec{\delta}^{(i)}\} \leq \Re\{-\vec{\gamma}^{(i)}\}$$

Update $\vec{\gamma}^{(i+1)} = \vec{\gamma}^{(i)} - \vec{\delta}^{(i)}$
Compute $\Psi(\vec{\gamma}^{(i+1)}, \vec{t}_{k-1})$, $\Omega = \Psi^{+}\mathrm{H}_0^T$, $R = \mathrm{H}_0^T - \Psi(\vec{\gamma}^{(i+1)}, \vec{t}_{k-1})\Omega$, and $\rho = R(:)$
Find $res_{i+1} = \|\rho\|_2$
if $res_{i+1} > res_i$
  if restarts > $GMAX$
    Break loop, exceeded $GMAX$
  end
  Assign $\nu^{(i)} \leftarrow INCR \cdot \nu^{(i)}$
  Increment restarts=restarts + 1
  restart i-loop without updating i
else
  if $res_{i+1} < Tol$
    Break loop, converged
  end
  if restarts == 0
    Update $\nu^{(i+1)} = \nu^{(i)}/DECR$
  end
  Set restarts = 0
end
end

Algorithm 1 is pseudo-code summarizing the algorithm used to construct a ROM by the cOptDMDc procedure from given snapshot and input data. It also details how the Levenberg-Marquardt nonlinear least squares optimization algorithm is implemented for the presented results. Here MaxIter is the maximum number of iterations before the optimization is terminated regardless of convergence, GMAX is the maximum number of inner iterations dedicated to updating $\nu^{(i)}$, INCR is a positive number greater than 1 used to increase the size of $\nu^{(i)}$, and DECR is a positive number greater than 1 used to decrease the size of $\nu^{(i)}$. Tol is the user-specified tolerance used to determine convergence. For the results presented in all case studies below, GMAX was set to 50, INCR was set to 1.5, and DECR was set to 2. A larger value of $\nu^{(i)}$ makes the optimization process similar to gradient descent, while a smaller value of $\nu^{(i)}$ leads to Gauss-Newton optimization [32].

III. RESULTS AND DISCUSSION

This section presents three numerical case studies and compares the ROMs generated by the proposed cOptDMDc with those by DMDc and SPDMDc for a comprehensive evaluation of the former. Section III-A compares the predictions of the models on clean data that is generated by operators with known eigenvalues and on data produced from those operators subject to noise. In this study, the data used to generate or train the ROMs is also used for validation, essentially testing the accuracy of each ROM's reconstruction of the training data. Section III-B compares the results obtained by the three ROMs using a heat conduction problem on a simple two-dimensional vehicle geometry with a variable engine temperature. Section III-C compares the heat conduction results on a more complicated, three-dimensional Ahmed Body geometry [33] with an engine temperature that also varies with time. In these

two case studies, the time-varying engine temperature profile is treated as the exogenous input. The ROMs will be constructed on one prescribed input profile. The constructed ROMs are then validated using two different engine temperature profiles to assess the generalization capabilities of the ROMs.

To examine the robustness of the ROMs, two scenarios of determining the number of SVD modes are considered. In the first, the dimension of the reduced-order subspace will be manually selected through a trial-and-error process to yield the best prediction results for the validation cases. Then the dimension will then be slightly adjusted to simulate an actual use case where the optimal subspace dimension is unknown in order to show the potential issues with using DMDc or SPDMDc, hence emphasizing the significance of the proposed cOptDMDc.

A. Linearized Navier-Stokes

First, a simplified case study of a linearized Navier-Stokes (LNS) problem for Poiseuille flow [16] is examined to compare results between DMDc, SPDMDc, and cOptDMDc. The same case was also studied in [30], which considered a three-dimensional, steady-state Poiseuille flow between two plates with a prescribed parabolic velocity profile as shown in Figure 1.

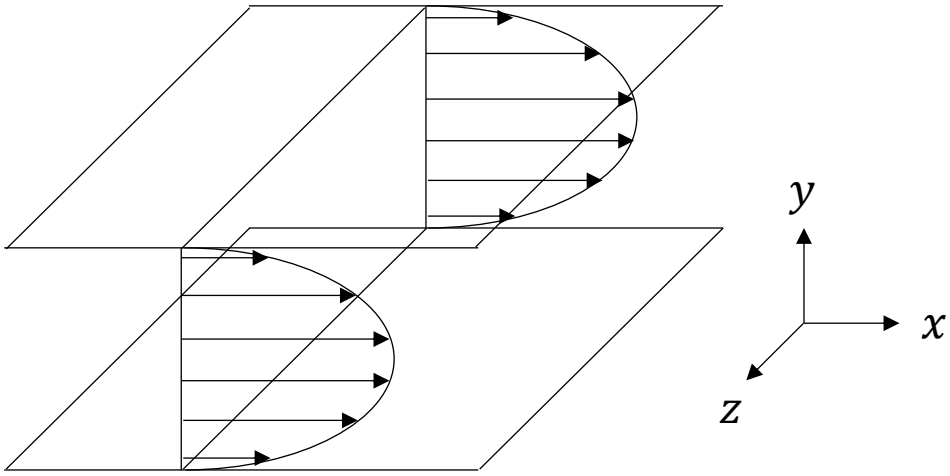


Figure 1. 3-Dimensional Poiseuille flow between two plates

The LNS procedure determines the effect of external perturbations from the prescribed profile, and may be represented by a state-space model [16]

$$\partial_t \widehat{\psi}(k_x, y, k_z, t) = \hat{A}(k_x, y, k_z)\widehat{\psi}(k_x, y, k_z, t) + \hat{B}(k_x, y, k_z)\hat{\Gamma}(k_x, y, k_z, t) \tag{37}$$

where $\psi(x, y, z, t) = \begin{bmatrix} v(x, y, z, t) \\ \omega_y(x, y, z, t) \end{bmatrix}$, with $v(x, y, z, t)$ as the velocity perturbation in the y-direction and ω_y as the vorticity perturbation in the y-direction. $\Gamma(x, y, z, t)$ is the external forcing term exerted on the system. $\widehat{\psi}(k_x, y, k_z, t)$ is the Fourier transformation of ψ in the streamwise (x) and spanwise (z) directions with streamwise and spanwise wavenumbers k_x and k_z, respectively. Similarly, $\hat{\Gamma}(k_x, y, k_z, t)$ is the Fourier transformation of $\Gamma(x, y, z, t)$ in the same directions as ψ. $\hat{A}(k_x, y, k_z)$ and $\hat{B}(k_x, y, k_z)$ are linear, time-invariant operators that determine the evolution of the system with inputs. For details on how $\hat{A}$ and $\hat{B}$ are constructed and how Eq. (37) generates data, the readers are referred to Jovanović and Bamieh's analysis in [16] and to Jovanović's dissertation [34].

The results of this case study are obtained with wavenumbers $k_x = k_z = 1$, the prescribed Poiseuille flow velocity $U(y) = 1 - y^2$, a Reynolds number of $Re = 2000$, and a snapshot-spacing time step of $\Delta t = 1$. The spatial domain of analysis is a box region defined by $0 \leq x \leq 1$, $-1 \leq y \leq 1$, and $0 \leq z \leq 1$. Discrete data is generated from a spatial discretization of 200 points in the y-direction. Similar to [30], the system is excited with a streamwise, temporally-varying chirp force defined by $\Gamma(t) = \sin(2\pi[(1.9 \times 10^{-4})t^2 + 0.01t])$, located in the middle of the channel. The analysis is run for 500 snapshots with a random initial condition. For all cases in this work, the error is measured for each snapshot using two metrics: the ℓ_2 relative error

$$\epsilon_{rel,i} = 100\% \times \frac{\|\psi_i - \tilde{\psi}_i\|_2}{\|\psi_i\|_2} \tag{38}$$

and the maximum absolute error

$$\epsilon_{max,i} = \max_j |\psi_i^j - \tilde{\psi}_i^j| \tag{39}$$

where ψ_i is the i^{th} ground truth snapshot, $\tilde{\psi}_i$ is the i^{th} snapshot estimated by the ROM, and the superscript j in Eq. (39) denotes the j^{th} component of the snapshot vector, i.e., v and ω_y.

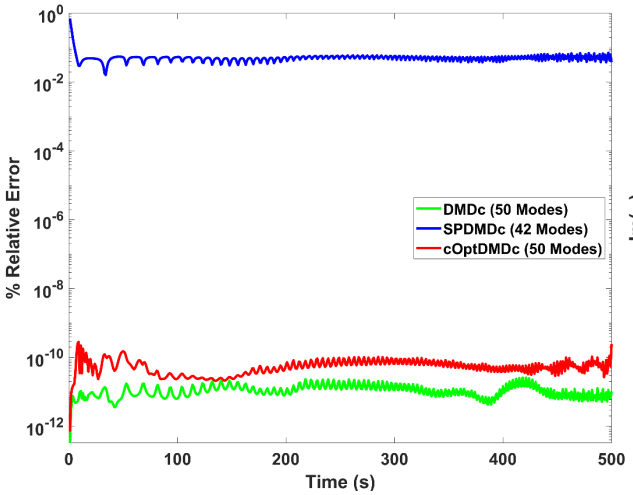


Figure 2: Relative error of flow predictions for DMDc (green), SPDMDc (blue), and cOptDMDc (red) on clean training data.

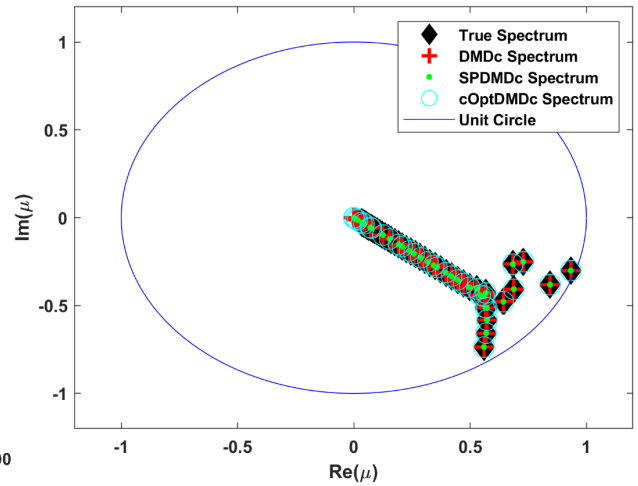


Figure 3. Discrete operator spectra for DMDc (red cross), SPDMDc (green dot), and cOptDMDc (cyan circle) compared to the actual operator spectrum (black diamond)

Figure 2 shows the relative errors of the three methods, where each model is trained and tested on the same data. In this case, the sparsity weight parameter γ is set to 10 and SPDMDc starts to discard eigenvalues to achieve parsimony of the model structure. If an eigenvalue is not discarded, the output of SPDMDc would be identical to that of DMDc. The errors for DMDc and the proposed cOptDMDc are similar and stay stably around 10^{-10} %. The error for SPDMDc is still reasonable at about 0.05%.

Table 1: Errors associated with each DMDc method on clean data of the LNS problem

Model	Max Relative Error (%)	Mean Relative Error (%)	Max Absolute Error	Retained Modes
DMDc	1.75E-10	1.13E-11	4.30E-12	50

SPDMDc	2.515	0.0528324	0.0762	42
cOptDMDc	5.99E-10	2.84E-10	8.72E-12	50

Table 1 enumerates the maximum relative error, the mean relative error, and the maximum absolute error between the methods across all snapshots along with the number of the retained modes. We can see that the error associated with cOptDMDc is slightly higher than DMDc, although they remain on the same order of magnitude, which means that they both reconstruct the data extremely well. While performing well, the SPDMDc ROM is worse than DMDc and our cOptDMDc due to the pruning of those 8 modes for better sparsity. Figure 3 plots the discrete-time eigenvalues of the DMDc operator, SPDMDc operator, and the cOptDMDc operator, and compares them against the spectrum of the continuous LNS operator $\hat{A}(k_x, y, k_z)$ obtained analytically from Eq. (37). The continuous-time eigenvalues determined from the LNS operator and the cOptDMDc procedure are converted to the discrete-time domain through exponentiation for an easier comparison and visualization (as eigenvalues with large negative real part in the continuous-time domain will be concentrated around the origin in the discrete-time domain). This visualization technique is used in the latter case studies as well. All eigenvalues are within the unit circle, which indicates that the system is fully stable. These three methods identify the operator's eigenvalues very well, which is the fundamental reason for the excellent agreement of their results with the training data.

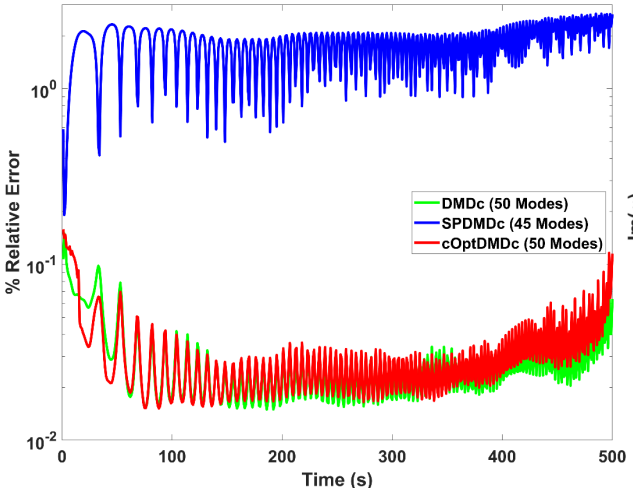


Figure 4. Relative Error of flow predictions for DMDc (green), SPDMDc (blue), and cOptDMDc (red) on noisy training data

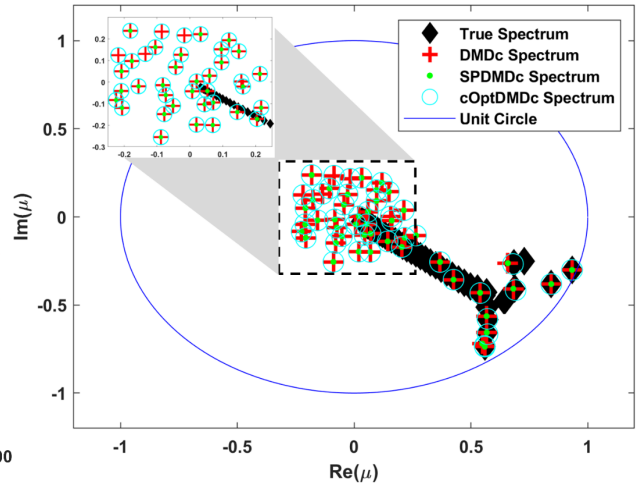


Figure 5. Operator spectra for DMDc (red cross), SPDMDc (green dot), and cOptDMDc (cyan circle) compared to the actual operator spectrum (black diamond). Inset in top left shows eigenvalues in range $\Re(\lambda) \in [-0.3,0.3], \Im(\lambda) \in [-0.3,0.3]$

All three methods perform very well in the presence of clean data. Another study is then conducted where the Gaussian white noise with a mean of zero and a standard deviation of $\sigma_{noise} = 10^{-3}$ is added to the data snapshot matrix to simulate the process noise, while the inputs are not corrupted by the noise. Figure 4 shows the relative errors of the data reconstructions created by the ROMs that are trained on the noisy data, and the comparison is made with the underlying clean noise-free data. DMDc and cOptDMDc perform similarly again and their errors hold stable throughout the time domain. However, due to the data contamination by the noise, their relative errors rise to around 0.03% error on average, which is significantly higher than the previous case with training on clean data (around 10^{-10}). In this study, SPDMDc removes 5 extraneous modes (shown by the inset in Figure 5) and yields a relative error of about 1.79%.

Table 2: Errors associated with each DMDc method on noisy data of the LNS problem

Model	Max Relative Error (%)	Mean Relative Error (%)	Max Absolute Error	Retained Modes
DMDc	0.1499	0.0296	0.00304	50
SPDMDc	2.67	1.785	0.07714	45
cOptDMDc	0.1585	0.0316	0.00366	50

Table 2 lists the results of the three ROMs obtained for the noisy training data. The DMDc and cOptDMDc models are quite accurate, with a maximum relative error of only 0.1499% and 0.1585%, respectively. SPDMDc achieves a maximum relative error of 2.67%, which again is expected because discarding five extraneous modes leads to the loss of data reconstruction accuracy and agrees with the findings in [30]. The effect of the noise on the operator eigenvalues is pronounced. The near-zero eigenvalues of the $\hat{A}$ operator are dispersed randomly around zero due to the added noise of high frequency. Figure 5 also shows that there are some eigenvalues retained by DMDc that are discarded by SPDMDc, as some red crosses (DMDc eigenvalues) are not accompanied by green dots (SPDMDc eigenvalues). Furthermore, the figure shows where cOptDMDc modifies the eigenvalues by noting the difference between the red crosses and blue circles (cOptDMDc eigenvalues). Both DMDc and cOptDMDc also have similar ROM performance with noisy data, as the eigenvalues in cOptDMDc does not change greatly after the optimization process is completed. Furthermore, they also exhibit great accuracy despite the presence of a large amount of noise.

The case study above is performed to show the differences between the three ROM methods, and their different responses to the noise. That is, even in the presence of noises, they still produce reasonable results. SPDMDc trade the model accuracy for the sparsity of the model structure. However, only when it retains all modes and essentially is the original DMDc model, SPDMDc could achieve best performance similar to the other two model.

B. 2-Dimensional Vehicle Mockup

For the second case, the transient heat conduction of a 2-Dimensional vehicle mockup is considered as shown in Figure 6. The vehicle material is aluminum with specific heat capacity $c_p = 903\,J/kg - K$, thermal conductivity $k = 237\,W/m - K$, and density $\rho = 2702\,kg/m^3$. The boundary conditions are also summarized in Figure 6, including an adiabatic floor, a roof experiencing convective heat transfer with a constant heat transfer coefficient and a constant ambient temperature of 300 K, and a uniform engine temperature that varies with time. The initial temperature of the entire vehicle is also 300K. The transient heat distribution is simulated over a period of 96 hours with a time step of 10 s using the implicit unsteady thermal conduction solver from STAR-CCM+, and snapshots are saved every 10 time steps. Note that in this work, STAR-CCM+ simulation results are used as either the training data or the validation data for the ROM generated by different approaches.

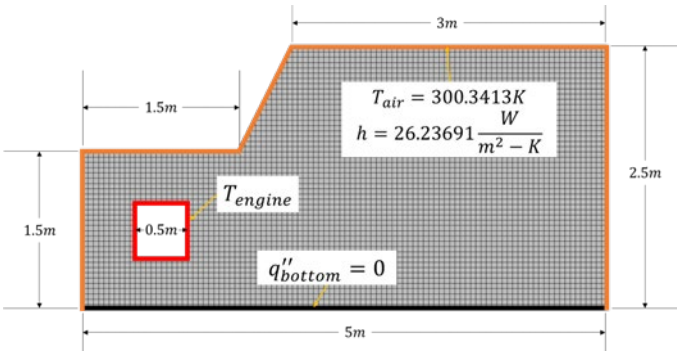


Figure 6. 2-Dimensional vehicle mockup showing the finite volume mesh and boundary conditions

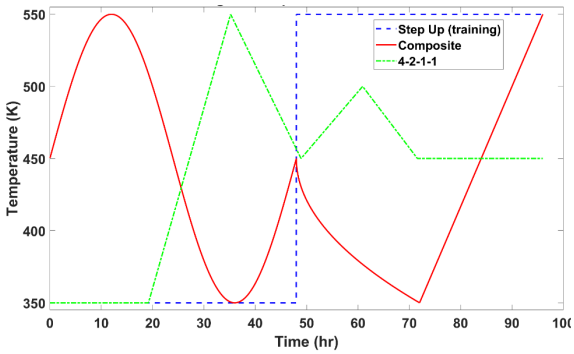


Figure 7. The input profile of the engine temperature over time

Figure 7 shows three engine temperature profiles used in the simulation. Recall that the time-varying engine temperature is treated as the exogeneous input in this case study as discussed above. The first thermal simulation is run using a simple "Step" input profile (colored in dashed blue line), where the engine temperature is held at a constant temperature of 350K until 48 hours have passed, and then increases to 550K for the remaining 48 hours. This simulation data is used to train the ROM. The remaining two simulations are used to collect validation data to evaluate the accuracy of the ROM. The first validation simulation is run using the "Composite" engine temperature profile (colored in red). For the first 48 hours, the profile follows a sinusoidal path with an amplitude of 100K for one full period. Then, the temperature decreases following a square-root path, and after 24 hours, the temperature reaches 350K. Finally, the temperature linearly increases until reaching 550K for the remaining 24 hours. This particular profile is used because of the large overall temperature variation and the existence of sharp corners, both of which would likely be challenging for a predictive model to accurately approximate. The second validation simulation is run using the "4-2-1-1" engine temperature, similar to the 3211 multistep function often used in system identification [35, 36]. Below to demonstrate the performance improvements of cOptDMDc over standard DMDc or SPDMDc, the ROMs are first set to retain 13 SVD modes for their ROM construction, and then are set to retain the 16 SVD modes. The comparison between these two will be used to reveal the susceptibility of DMDc and SPDMDc to the model dimensions relative to cOptDMDc.

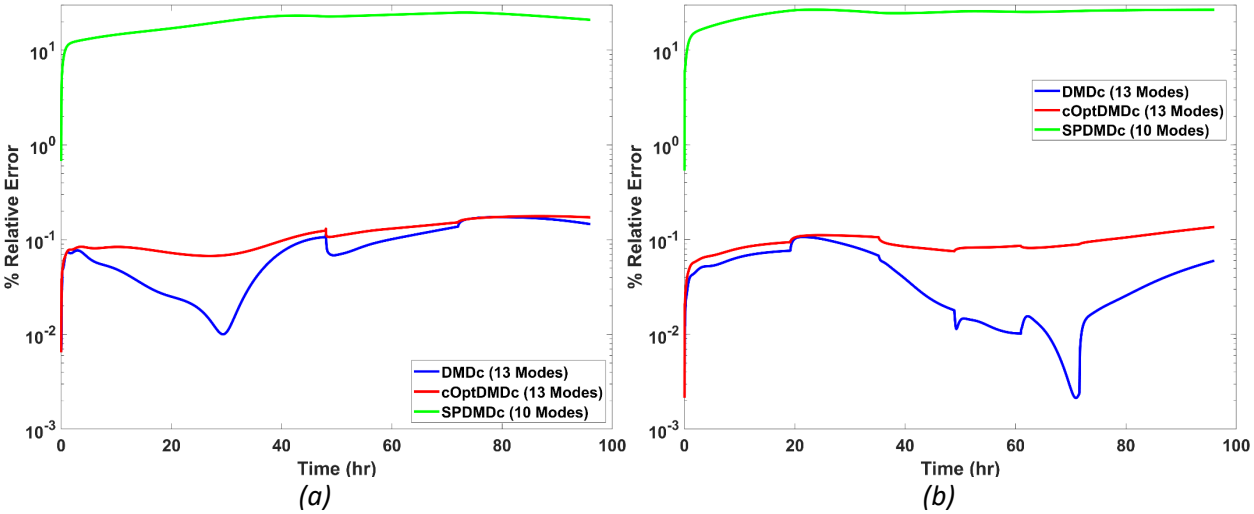


Figure 8. L_2 Relative Errors of DMDc, SPDMDc, and cOptDMDc trained on 13 retained SVD modes over time for (a) the Composite engine temperature profile and (b) the 4-2-1-1 profile

Figure 8 shows that the time-varying relative errors averaged across the entire vehicle domain for (a) the Composite engine temperature profile and (b) the 4-2-1-1 engine temperature profile, and the relative error is defined in the same manner as the first case study above. For the SPDMDc procedure to remove extraneous modes, the sparsity hyperparameter λ is set to 10^4, which is a relatively high value for the sparsity hyperparameter λ and used to force SPDMDc to produce different results from DMDc, yielding a meaningful comparison. If the value is set lower, the SPDMDc predictions are identical to the DMDc. A relatively high λ up to 10^6 was also used in previous research [30] to show its effect on model sparsity and accuracy. Accordingly, three extraneous modes are discarded by the SPDMDc procedure, generating a ROM with 10 dimensions. The DMDc and cOptDMDc relative errors are very small and of the same order of magnitude, with cOptDMDc performing slightly worse for the Composite profile but noticeably different for the 4-2-1-1 profile. Such a minor loss in accuracy may be attributed to overfitting of training data or the slight change of the eigenvalues and eigenvectors during the cOptDMDc procedure. The errors for SPDMDc are quite high and reach 25%, which implies that the three modes thrown out are important for data fitting. The SPDMDc errors appear to remain stable over time, where noticeable discontinuities are observed in the DMDc and cOptDMDc errors corresponding to discontinuities in the input profile.

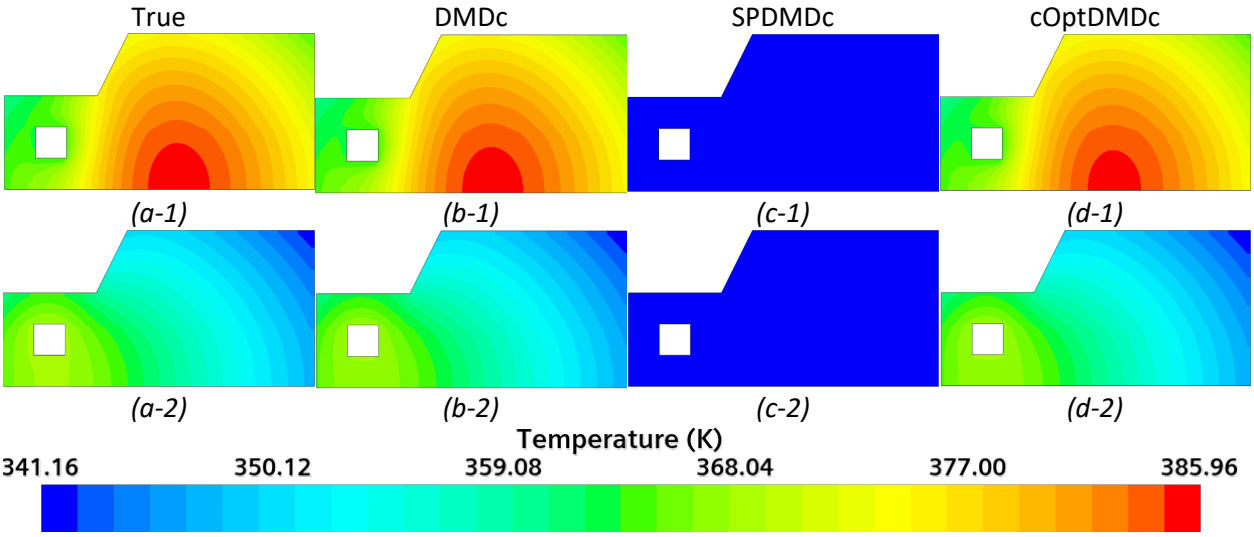


Figure 9. Spatial temperature distribution of the Composite input profile at different time instances: (top) $t = 32$ hr and (bottom) $t = 64$ hr. The temperature distribution from different sources: (a) the true value; (b) ROM generated by DMDc; (c) ROM by SPDMDc, and (d) ROM by cOptDMDc.

Figure 9 shows the true vehicle temperature as generated by STAR-CCM+ along with the ROM results predicted by DMDc, SPDMDc, and cOptDMDc at two time instances for the Composite input profile. It should be noted that the dark blue temperature distribution predicted by SPDMDc in Figure 9(c) does not mean the predicted profile is constant, but that it falls below the minimum bound of the true value shown on the color bar. Apparently, DMDc and cOptDMDc perform very well, as the temperature distributions at both time instances are very similar to the true profiles.

True DMDc SPDMDc cOptDMDc

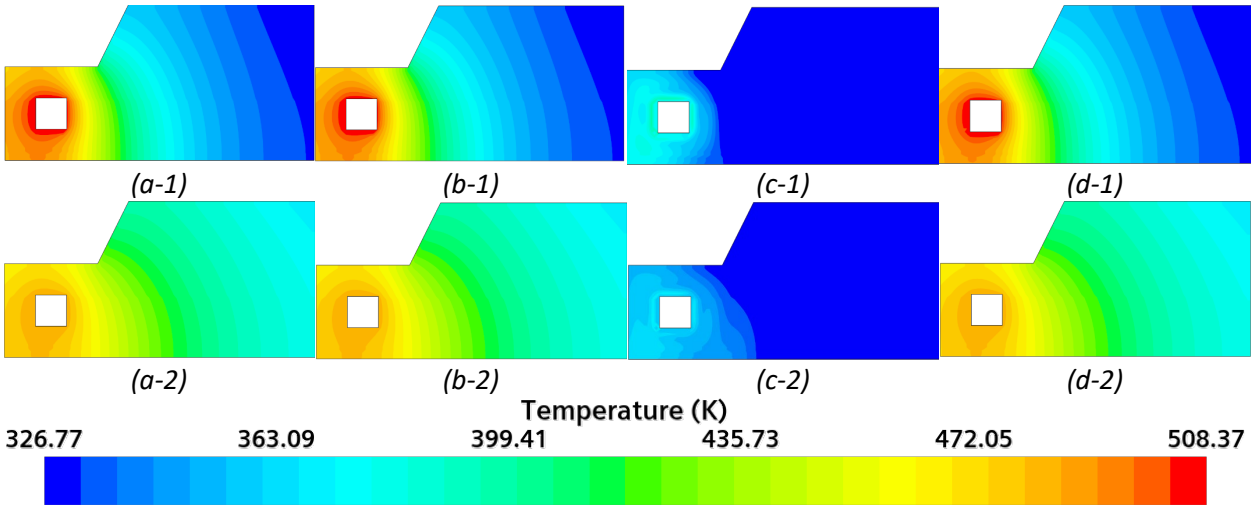
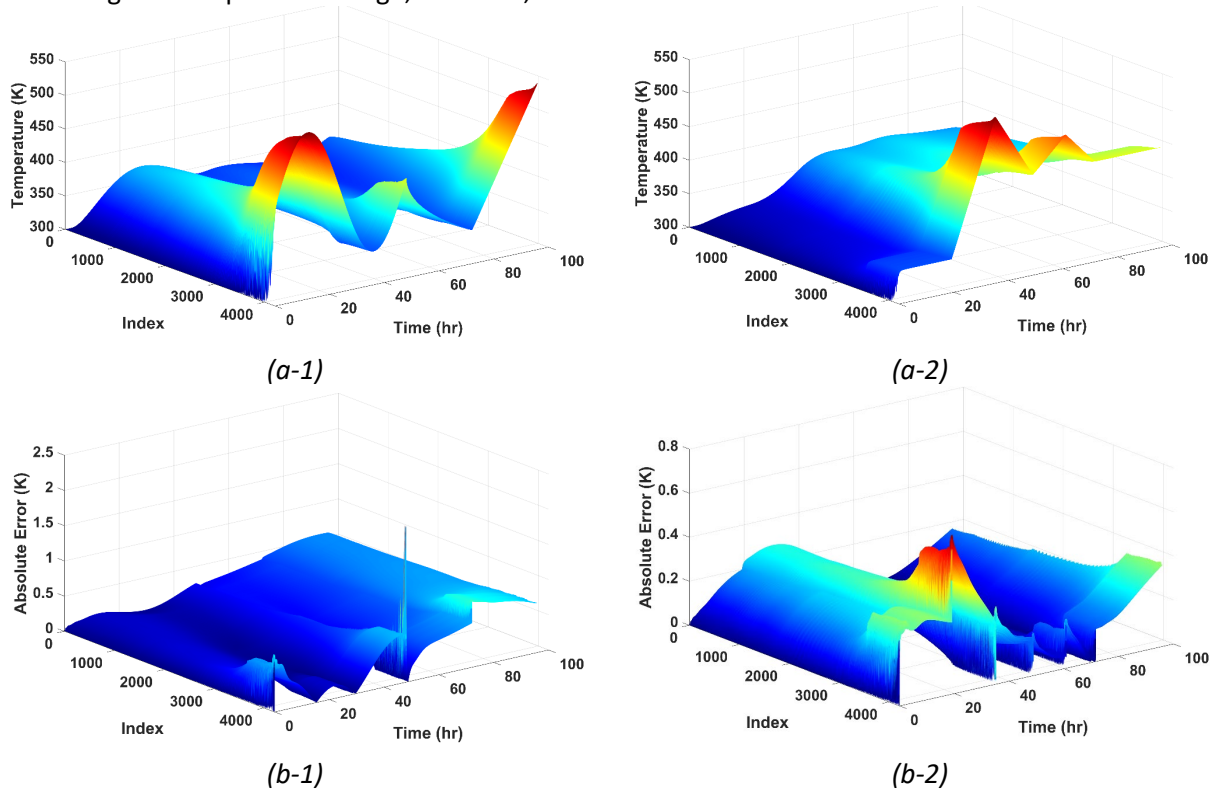
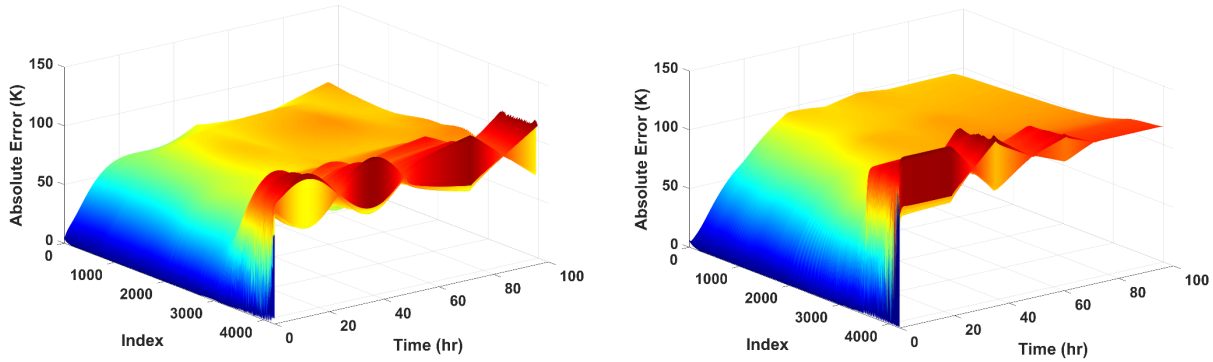


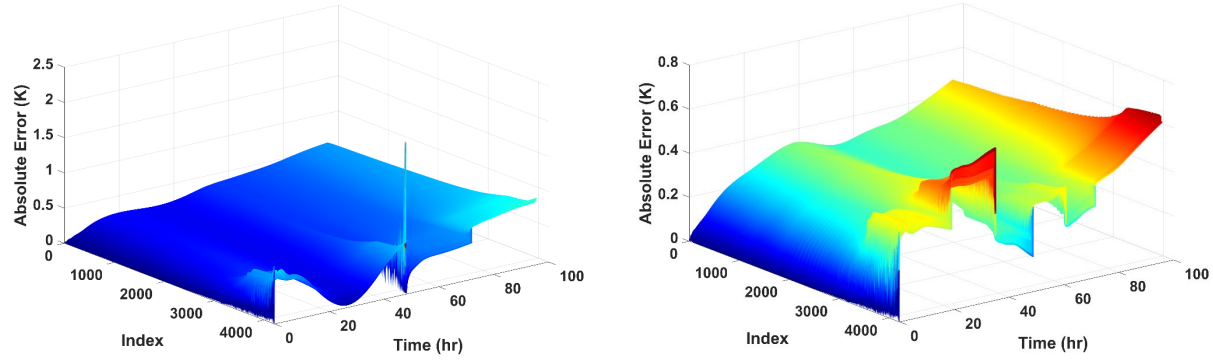
Figure 10. Spatial temperature distribution of the 4-2-1-1 input profile at different time instances: (top) $t = 32$ hr and (bottom) $t = 64$ hr. The temperature distribution from different sources: (a) the true value; (b) ROM generated by DMDc; (c) ROM by SPDMDc, and (d) ROM by cOptDMDc

Similar to Figure 9, Figure 10 shows the vehicle temperatures from the ground truth and all three ROMs for the 4-2-1-1 input profile at the two time instances. The performance of DMDc and cOptDMDc on this case is as salient as the Composite input profile above, since their ROM predictions and the true temperatures are nearly indistinguishable. In this case, SPDMDc does make ROM predictions that fall inside the given temperature range; however, most still fall well below the lower bound.





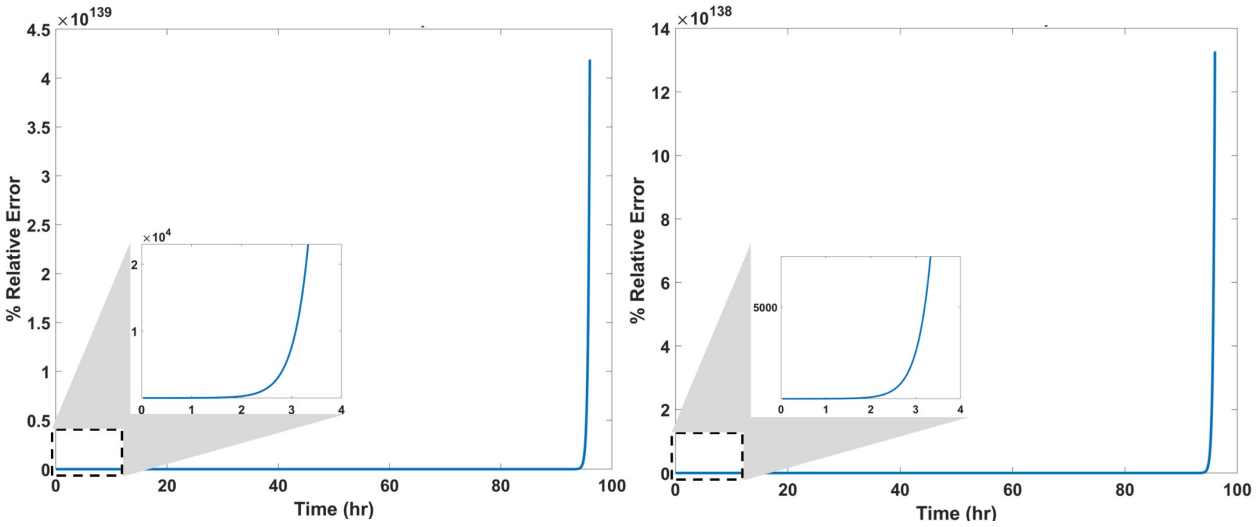
(c-1) *(c-2)*



(d-1) *(d-2)*

Figure 11: Spatiotemporal temperature distribution within the 2D Vehicle mock-up for (left) the Composite and (right) 4-2-1-1 profile. (a) True values; (b) the absolute error of the ROM generated by DMDc; (c) the absolute error of the ROM generated by SPDMDc; and (d) the absolute error of the ROM generated by cOptDMDc.

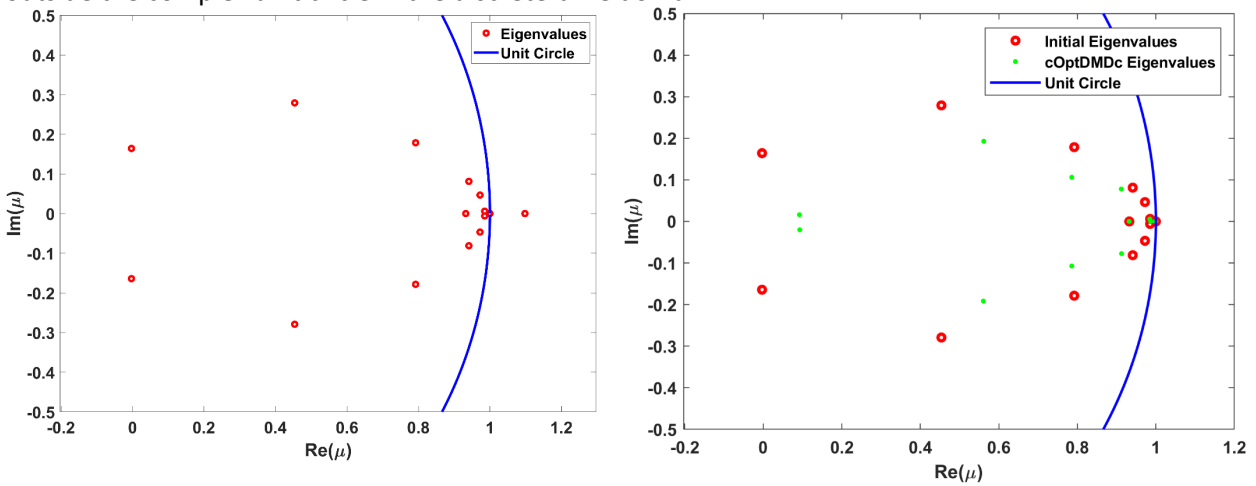
Figure 11 shows the temperature profile of the vehicle throughout the entire computational domain and across all time steps. The temperatures are sorted based on the true temperature at the final timestep to facilitate visualization of all values in the plot. Figure 11(a-1) and (a-2) depict the true temperature distribution within the spatiotemporal domain, where the axis 'index' denotes the indices of all the computational cells. Figure 11(b-1)-(d-1) illustrate the absolute error of the predicted temperatures by DMDc, SPDMDc, and cOptDMDc, respectively, for the Composite profile, and Figure 11(b-2)-(d-2) are those for the 4-2-1-1 profile. The DMDc and cOptDMDc models yield excellent results. The errors are all stable except for a few discontinuous locations corresponding to discontinuities in the input profile. The errors start low and remain low, which show that both procedures identify stable dynamics and would remain stable for a variety of external input profiles. SPDMDc results show that the high prediction error remains relatively constant through time, which indicates that the model is stable but insufficiently complex to account for all the system dynamics hidden in the data.



(a) (b)

Figure 12. L_2 Relative Errors of DMDc with 16 SVD modes over time for (a) the Composite engine temperature profile and (b) the 4-2-1-1 profile. Inset in the plots is a zoomed-in view of the first 4 hours of data predicted.

The choice of 13 modes for all the three models above was obtained through a trial-and-error process by repeatedly changing the number of retained modes and testing the resulting errors to identify the best results for the training and validation cases. However, in a practical use case without validation data this process would be impossible. To illustrate what could happen when a slightly different number of modes is chosen, we also examine another case where 16 SVD modes are retained in the three models instead of 13. Figure 12 shows the relative error of the DMDc model, which grows explosively and essentially diverges with this mode number. Furthermore, the insets in Figure 12 confirm that the explosive predictions happen at the beginning and continue exponentially throughout time. SPDMDc is not able to produce meaningful results either because the optimization process fails to converge from an ADMM residual of around 10^{287}. Both procedures diverge due to an eigenvalue of the F matrix from Eq. (10) lying outside the complex unit circle in the discrete-time domain.



(a) (b)

Figure 13: Spectrum of the F matrix from (a) DMDc and (b) cOptDMDc

Figure 13(a) and (b) illustrate the spectrum of the ROMs with 16 modes generated by DMDc and our cOptDMDc, respectively. An eigenvalue of the DMDc ROM lies outside the complex unit circle. The issue with an eigenvalue with a magnitude greater than 1 is apparent when considering Eq. (13), which, equivalent to Eq. (10), indicates that the DMDc and SPDMDc predictions involve the computation of μ_i^{k-1} for the k^{th} snapshot. For large values of k, an eigenvalue with a magnitude greater than 1 causes μ_i^{k-1} to grow exponentially, leading to divergence. On the other hand, Figure 13(b) reveals that cOptDMDc constrains the eigenvalues to the interior of the complex unit circle and modifies the dynamic modes accordingly mitigating this issue. It should be noted that in this analysis the eigenvalues from the F matrix generated by DMDc are used as initial values for the optimization process to produce the eigenvalues in cOptDMDc. The eigenvalues that lie outside the complex unit circle are radially projected onto its perimeter, and the constrained least squares solver keeps the eigenvalues inside the circle. Another interesting observation is that cOptDMDc alters not only the unstable eigenvalue, but also tune the locations of all eigenvalues, i.e., the green dots in Figure 13(b) to ensure exponential data fitting accuracy when the system is stable.

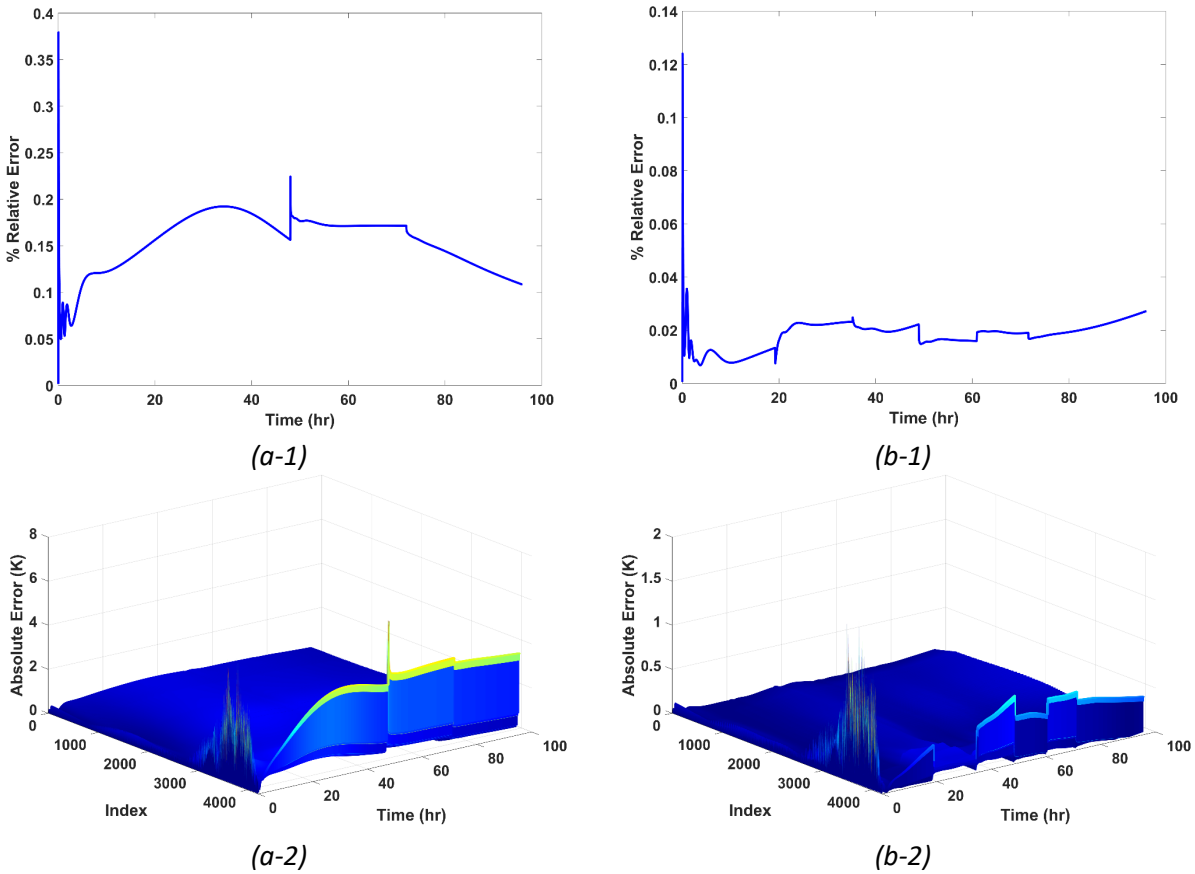


Figure 14. Results of cOptDMDc trained with 16 SVD modes for the (left) Composite and (right) 4-2-1-1 profiles. (Top) the relative errors vs. time; and (bottom) the spatiotemporal distribution of the absolute error.

Figure 14 shows the results of cOptDMDc's ROM predictions. Figure 14(a-1) and (a-2) are the relative errors and the spatiotemporal distribution of the absolute error for the Composite profile, respectively, and Figure 14(b-1)-(b-2) are the results for the 4-2-1-1 profile. The effect of constraining the eigenvalues to the interior of the unit circle by our cOptDMDc is pronounced, and the ROM predictions are still very accurate. The errors somewhat increase for the Composite input profile, with a maximum relative error of 0.3793% as opposed to 0.1772% for the case with 13 modes above. This implies potential slight overfitting, because given a more complex ROM structure, cOptDMDc may start to fit the small noises in the training data.

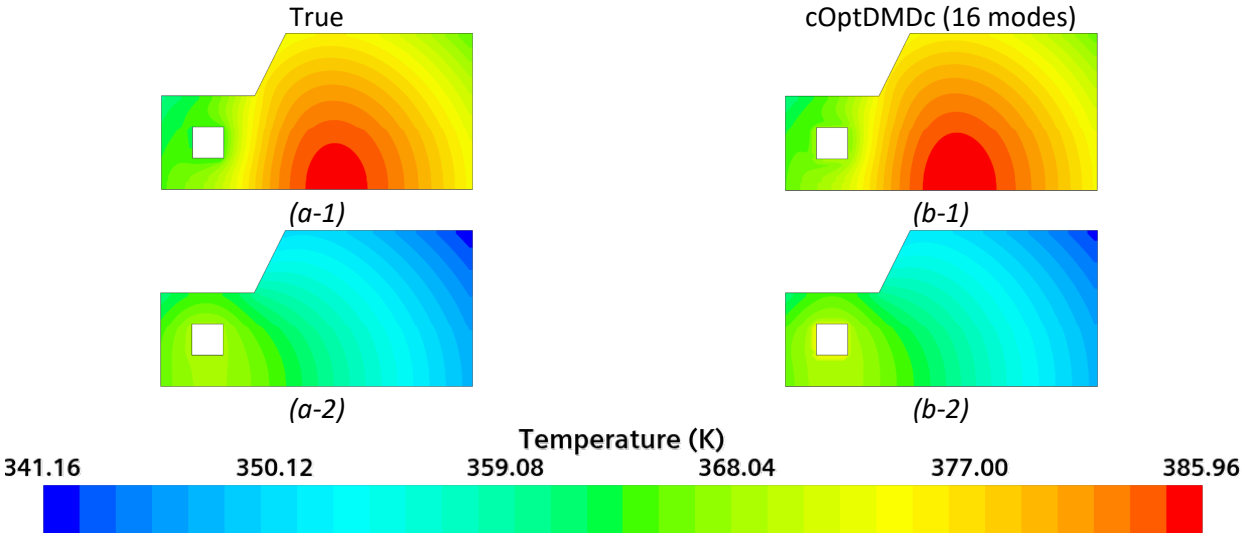


Figure 15. Spatial temperature distribution for the Composite input profile at (top) $t = 32$ hr and (bottom) $t = 64$ hr. The comparison is made between the (left) true temperature and (right) cOptDMDc ROM predictions.

Figure 15 shows the spatial temperature distribution predicted by the ROM generated by cOptDMDc compared to the true temperature values for the Composite input profile. Although the results obtained by other methods (DMDc and SPDMDc) diverge, the predictions made by cOptDMDc are still very close to those produced by STAR-CCM+ with the maximum relative error of 0.38% and the maximum absolute error of 6.52 K.

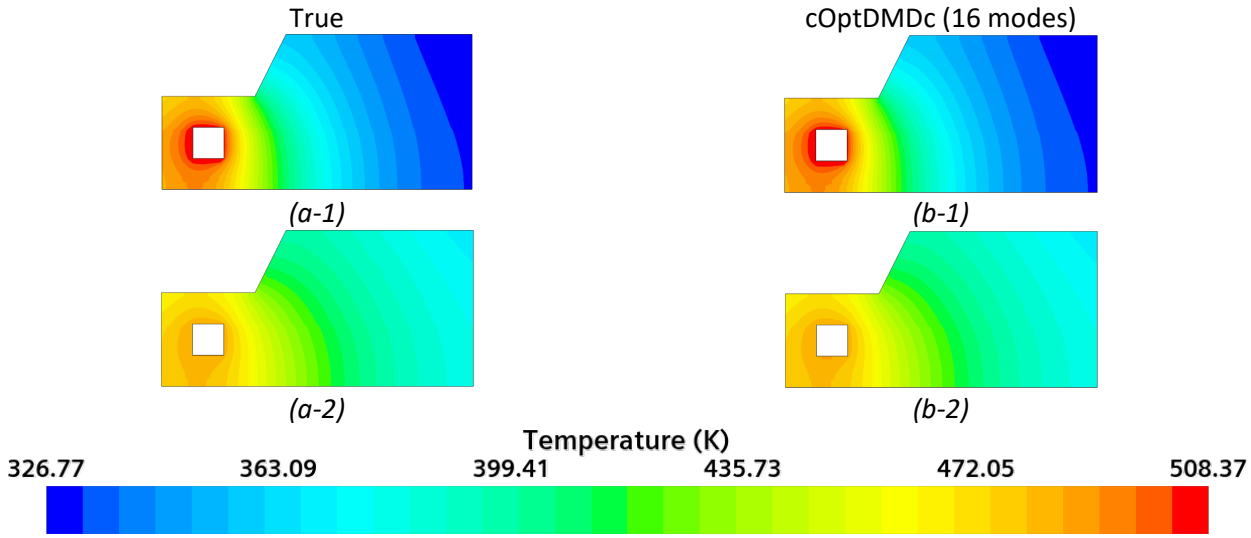


Figure 16. Spatial temperature distribution for the 4-2-1-1 input profile at (top) $t = 32$ hr and (bottom) $t = 64$ hr. The comparison is made between the (left) true temperature and (right) cOptDMDc ROM predictions.

Figure 16 shows the predictions made by cOptDMDc using 16 modes for the 4-2-1-1 engine temperature profile. Similar to the Composite profile, the ROM predictions and the true temperature values are visibly indistinguishable with the maximum relative error of 0.124% and the maximum absolute error of 1.9192

K, which confirms that the ROM's robustness enabled by constraining the eigenvalues within the unit circle of the spectrum is independent of the choice of the input profile.

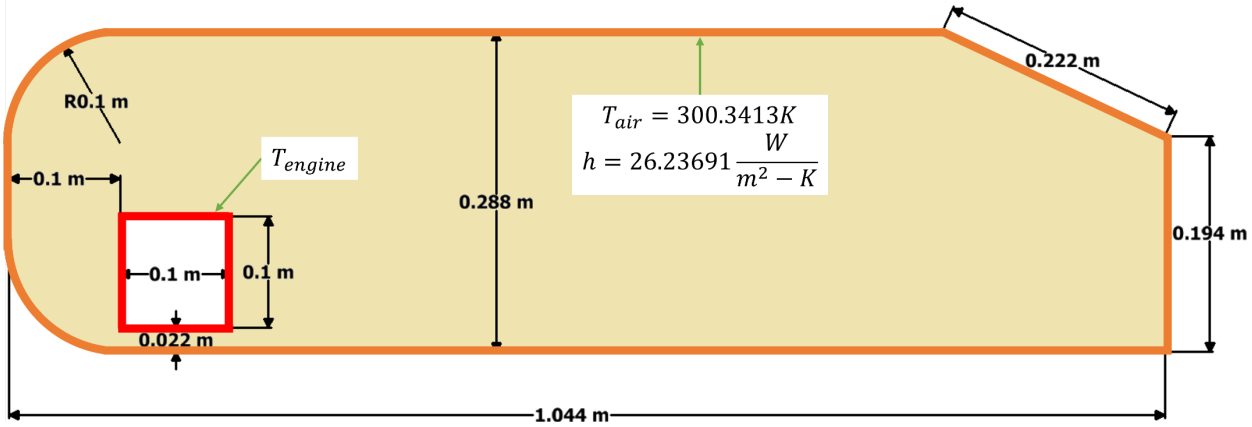
Table 3 shows the comparison in maximum relative (max rel.) and maximum absolute (max abs.) errors between the validation data and the predictions of the three ROMs with 13 and 16 modes for both the Composite and 4-2-1-1 input profiles. There are several interesting observations. First, both ROMs of DMDc and cOptDMDc exhibit very low errors when only 13 SVD modes are used to construct the corresponding ROMs. Although producing larger errors due to the truncation of three additional modes, ROM obtained by SPDMDc is still able to yield temporally stable results. Second, the ROM behaviors are completely different when 16 SVD modes are employed for model construction. It is clear that results of the ROM generated by cOptDMDc still match the true validation data excellently, while those of DMDc/SPDMDc diverge in time with completely erroneous results. Table 3 lists N/A for the errors of SPDMDc with 16 modes because SPDMDc fails to produce a model. Third, comparing ROM generated by cOptDMDc with 13 SVD modes and 16 SVD modes shows that the relative errors appear to have increased with a larger number of modes, which may be attributed to slight model overfitting given the more complex model structure.

Table 3: Maximum relative (max rel.) and maximum absolute (max abs.) errors of the validation data and the predictions of the three ROMs.

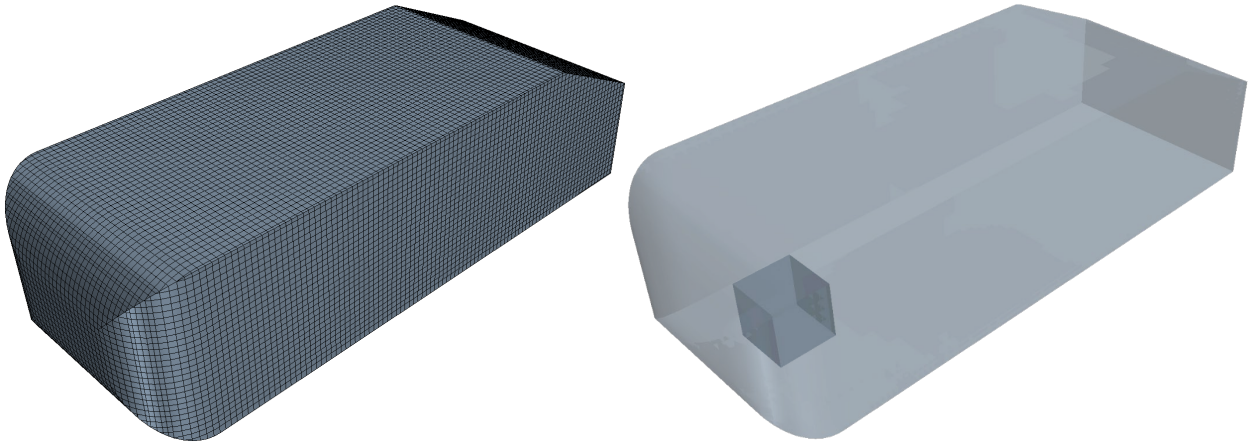
ROM	2D Vehicle							
	13 Modes (DMDc & cOptDMDc) 10 Modes (SPDMDc)				16 Modes			
	Composite		4-2-1-1		Composite		4-2-1-1	
	Max Rel. Error (%)	Max Abs. Error (K)	Max Rel. Error (%)	Max Abs. Error (K)	Max Rel. Error (%)	Max Abs. Error (K)	Max Rel. Error (%)	Max Abs. Error (K)
DMDc	0.1730	2.2179	0.1066	0.7236	4E+139	8E+140	1E+139	3E+140
SPDMDc	24.987	122.68	26.81	144.82	N/A	N/A	N/A	N/A
cOptDMDc	0.1772	2.1234	0.1362	0.7052	0.3793	6.5246	0.124	1.9192

C. 3D Ahmed Body

The third example involves the heat conduction of a 3D Ahmed Body geometry with an engine near the front of the vehicle as shown in Figure 17. The boundary conditions for this vehicle are similar to the 2D case above, with a time-varying engine temperature as the exogeneous input of the model. The exterior boundary experiences convective heat transfer with air at $T_{air} = 300.3413K$ and a constant heat transfer coefficient $h = 26.23691 \frac{W}{m^2-K}$. The material is also aluminum with the same density, heat capacity, and thermal conductivity as the 2D vehicle. The engine temperature is varied for three separate simulations, following the prescribed profiles given in Figure 7. The Step profile is employed by STAR-CCM+ to generate data for ROM training, and the Composite and 4-2-1-1 profiles are used for ROM validation. The initial temperature of the entire vehicle is set to 300K. The transient heat distribution is simulated over a period of 96 hours with a time step of 10 s using the implicit unsteady thermal conduction solver from STAR-CCM+, and the snapshots are saved every 10 time steps. The finite volume mesh is generated by STAR-CCM+'s trimmed cell mesher, yielding 41,130 computational cells.

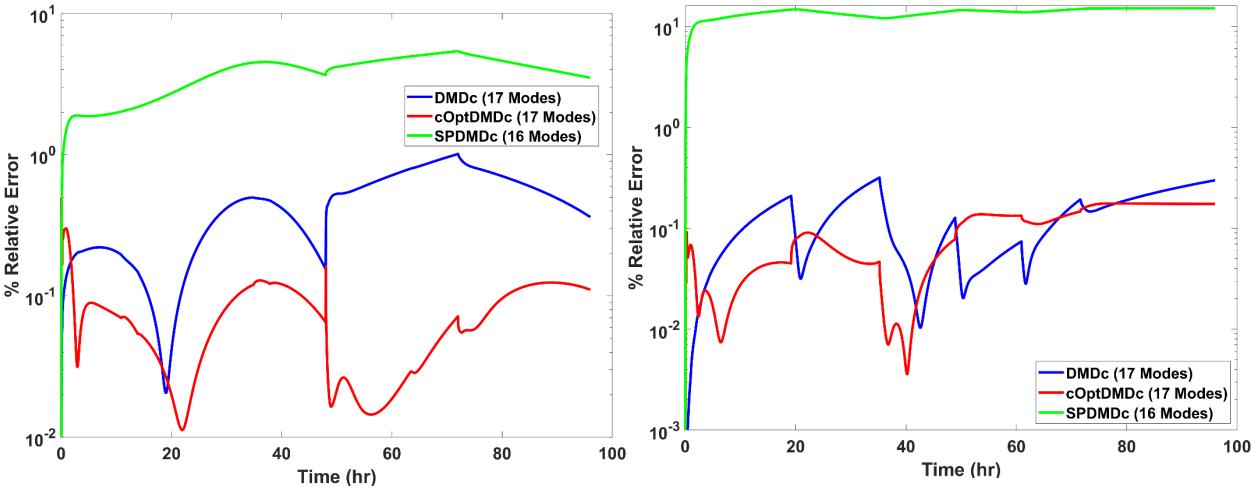


(a)



(b) *(c)*

Figure 17. 3D Ahmed Body Geometry showing (a) the dimensions of the cross-section and boundary conditions, (b) the finite-volume mesh, and (c) a transparent view of the geometry to visualize the placement of the cubical engine



(a) *(b)*

Figure 18. L_2 Relative Errors of DMDc, SPDMDc, and cOptDMDc trained on 17 retained SVD modes over time for (a) the Composite engine temperature profile and (b) the 4-2-1-1 input profile.

Using the same trial-and-error method to find the optimal number of modes as the 2D case above, it is determined that 17 modes would provide the best converged results for the ROMs. Figure 18 shows the relative errors obtained from the ROM predictions. Similar to the 2D vehicle, a value of $\lambda = 10^4$ is chosen for the sparsity hyperparameter to force SPDMDc to truncate a single mode only. In this case, the maximum relative error of DMDc, SPDMDc, and cOptDMDc are, respectively, 1.01%, 5.42%, and 0.49% for the Composite input profile, and 0.32%, 15.13%, and 0.176% for the 4-2-1-1 profile. Therefore, the proposed cOptDMDc outperforms DMDc and SPDMDc for the Composite profile by a significant margin, while performing similar to DMDc for the 4-2-1-1 profile, which again verifies the robustness and consistence of cOptDMDc and its independence of the input profiles.

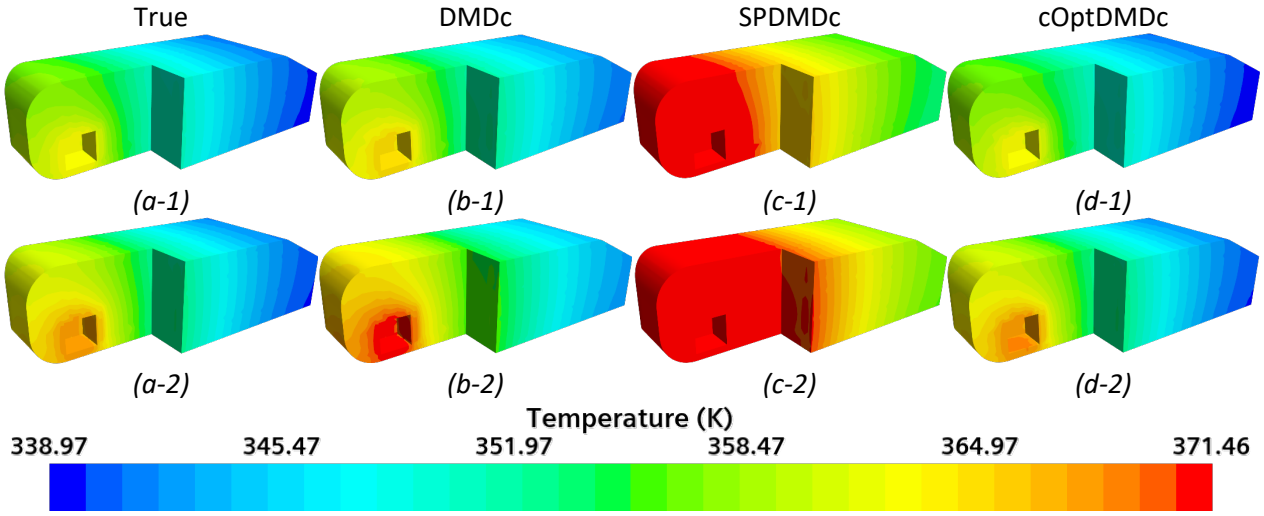


Figure 19. Spatial temperature distribution of the Composite input profile at different time instances: (top) $t = 32$ hr and (bottom) $t = 64$ hr. The temperature distribution from different sources: (a) the true value; (b) ROM generated by DMDc; (c) ROM by SPDMDc, and (d) ROM by cOptDMDc.

Figure 19 shows the comparison between the true temperature distribution generated by STAR-CCM+ with the ROM results of DMDc, SPDMDc, and cOptDMDc at two time instances for the Composite input profile. The ROM predictions offered by cOptDMDc match the values of the true temperature profile very well, as evidenced by comparing Figure 19(d) and (a). The DMDc predictions in Figure 19(b) are not as accurate as the cOptDMDc, and it appears that the DMDc-predicted temperatures are markedly higher than the true values, in particular, around the engine box. The ROM of SPDMDc overestimates the temperature distribution in Figure 19(c) dramatically and causes many temperature values to exceed the upper bound of the truth temperature given in the color bar, which is the reason why the front of the vehicle to take on the maximum color bar value (in dark red).

Figure 20 shows the Ahmed Body temperatures for the 4-2-1-1 input profile. In this case, DMDc and cOptDMDc both perform very well, as their predictions are visibly close to the true distribution. It is also observed that the ROM predictions of DMDc are much closer to the true values than in the Composite input profile case above, while the range of temperature variations is larger than in the Composite case. It appears that the temperature profile predicted by the ROM of SPDMDc is largely underestimated, as many values within the computational domain fall below the minimum temperature.

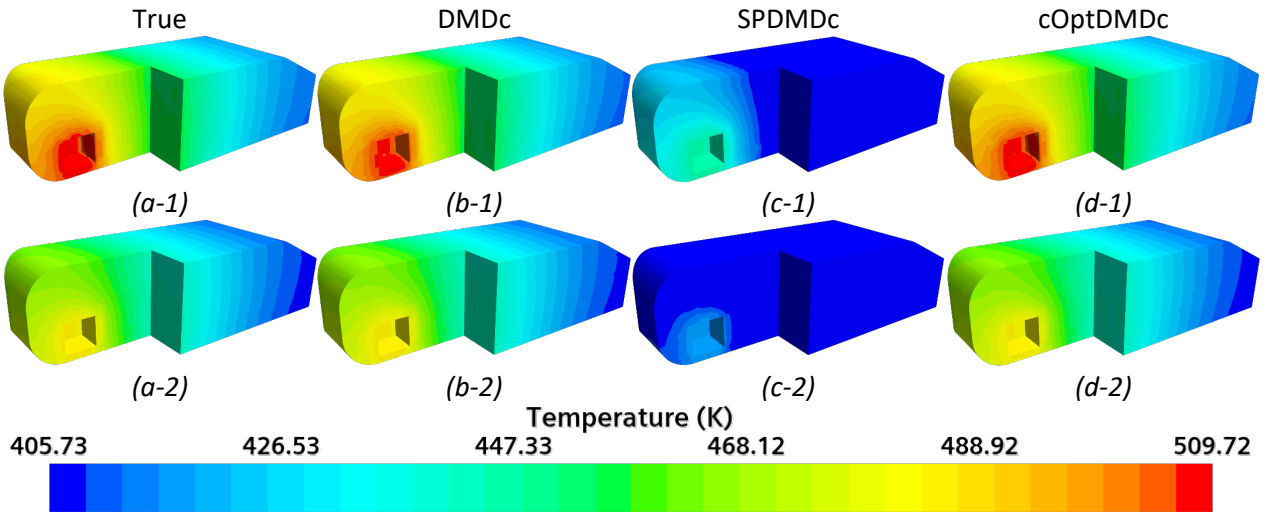
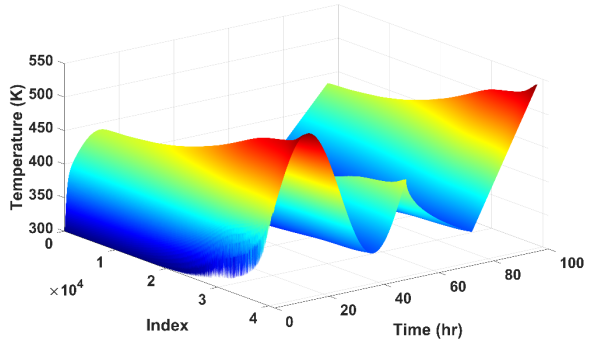
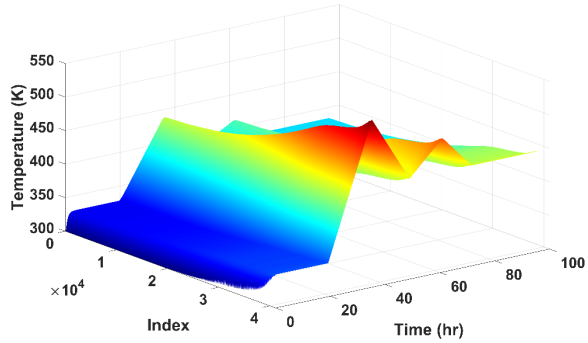


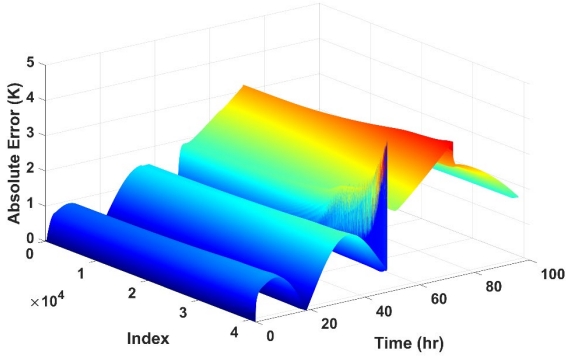
Figure 20. Spatial temperature distribution of the 4-2-1-1 input profile at different time instances: (top) $t = 32$ hr and (bottom) $t = 64$ hr. The temperature distribution from different sources: (a) the true value; (b) ROM generated by DMDc; (c) ROM by SPDMDc, and (d) ROM by cOptDMDc



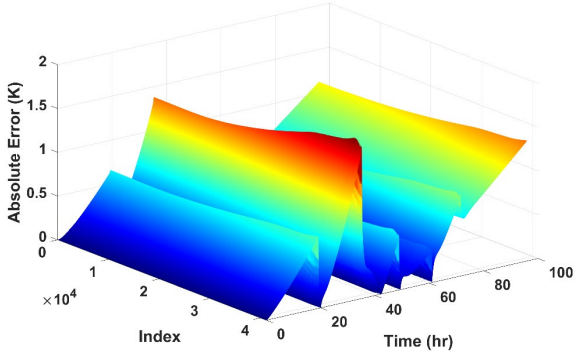
(a-1)



(a-2)



(b-1)



(b-2)

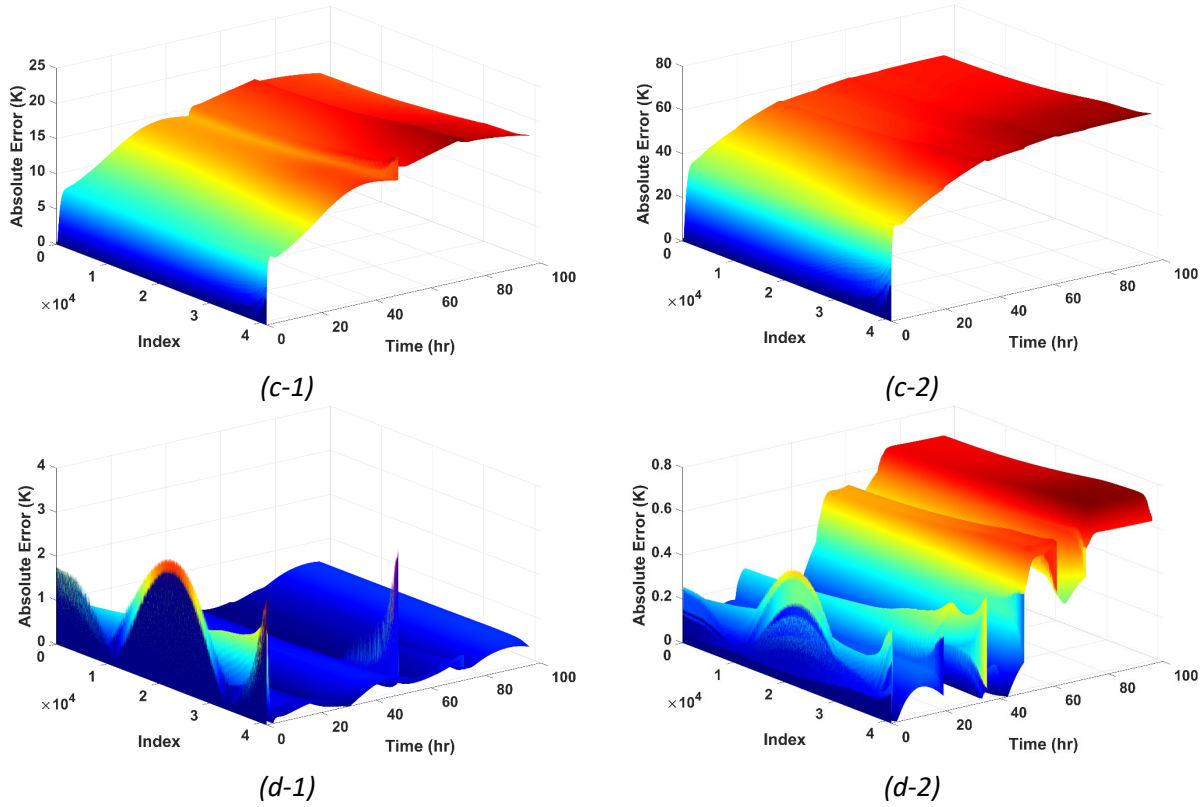


Figure 21. Spatiotemporal temperature distribution within the 3D Ahmed Body for (left) the Composite and (right) 4-2-1-1 profile. (a) True values; (b) the absolute error of the ROM generated by DMDc; (c) the absolute error of the ROM generated by SPDMDc; and (d) the absolute error of the ROM generated by cOptDMDc.

Figure 21 shows the temperature profile of the Ahmed Body throughout the entire computational domain and across all time steps, and the left column and the right column, respectively, show the results for the Composite and the 4-2-1-1 profile. Again, the temperatures are sorted based on the true temperature at the final timestep for enhanced visualization. Figure 21(a) present the true temperature of the domain, and Figure 21 (b)-(d) illustrate the absolute error of the ROM-predicted temperatures obtained by DMDc, SPDMDc, and cOptDMDc. We can clearly see that the proposed cOptDMDc is able to construct a robust ROM that appreciably outperforms the other models in both the Composite and 4-2-1-1 input profile. The maximum absolute errors of DMDc, SPDMDc, and cOptDMDc are, respectively, 4.36 K, 20.22 K, and 3.32 K for the Composite input profile, and 1.80 K, 68.08 K, and 0.77 K for the 4-2-1-1 input profile.

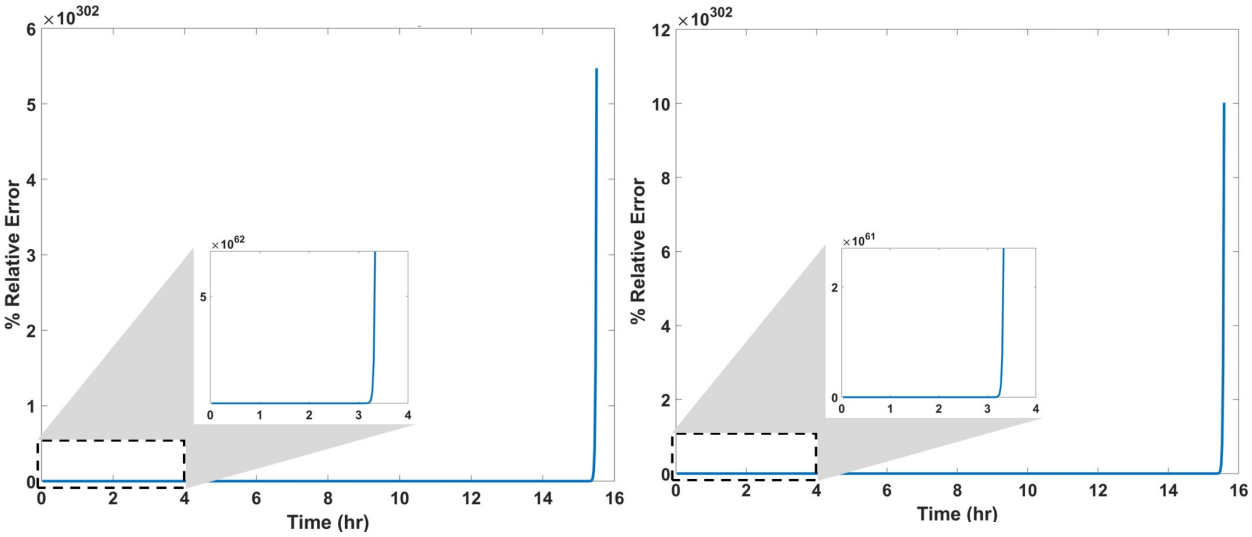


Figure 22: L_2 Relative Errors of DMDc with 16 retained SVD modes over time for (a) the Composite engine temperature profile and (b) the 4-2-1-1 profile. Inset in the plots are a zoomed-in view of the first 4 hours of data predicted.

Once again, the choice of 17 modes was made because there exists validation data to confirm its optimality. Next, we will suppose a scenario that the validation data is not available, and instead, 16 modes are retained for analysis. Figure 22 shows the relative errors of the DMDc predictions over time. Similar to the 2D vehicle case above, the insets show the explosive growth of the relative error at the very beginning, which continues to worsen over time. It is also important to note that the plot only shows relative errors up to about 16 hours because beyond that MATLAB treated the predictions as infinite. Likewise, ROM generated by SPDMDc also fails to produce a model and returns NaN values for the refit modal amplitudes θ_i. The spectrum of the F matrix explains both of these phenomena.

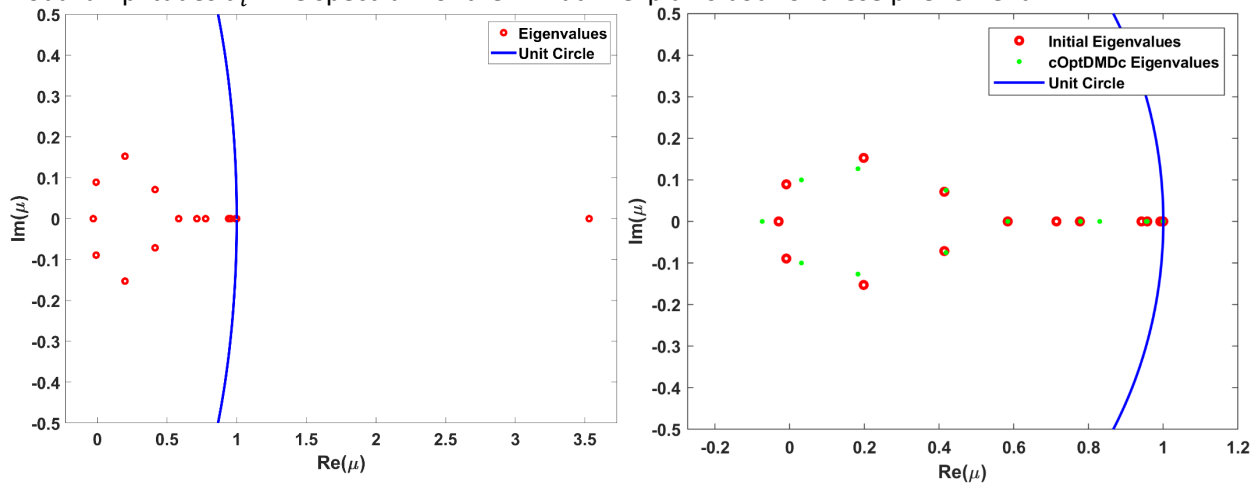


Figure 23. Spectrum of the F matrix from (a) DMDc and (b) cOptDMDc

Figure 23(a) and (b) show the spectrum of the F matrix and the eigenvalues of the DMDc and cOptDMDc models, respectively. The spectrum of DMDc includes an eigenvalue $\mu = 3.53$, which becomes enormous when raised to large values of $k-1$. In this case the maximum value of $k-1$ is 3455, yielding

$\mu^{3455} \sim 3.6 \times 10^{1892}$, which far exceeds the limits of double precision. This eigenvalue is involved in the predictions of DMDc/SPDMDc as well as the construction of the Q matrix in Eq. (15). Furthermore, for SPDMDc, the Q matrix is used in both optimization processes, which explains why NaN values are returned for θ_i. Even if this spurious eigenvalue is removed independently of the optimization process in Eq. (16), the Q matrix would still be involved in computing the refit amplitudes in Eq. (17), yielding the same results. On the other hand, the spectrum of the equivalent F matrix for cOptDMDc in Figure 23(b) clearly shows that the spurious eigenvalue is successfully constrained to the interior of the unit circle. The locations of the initial eigenvalues in Figure 23(b) confirm that the optimization process within cOptDMDc never needs to deal with the one of $\mu = 3.53$ because it is projected onto the unit circle to give $\mu = 1$ for initialization. The optimization process is then able to find an excellent fit with the new eigenvalues subject to the constraint, avoiding the numerical issues that arose in DMDc and SPDMDc above.

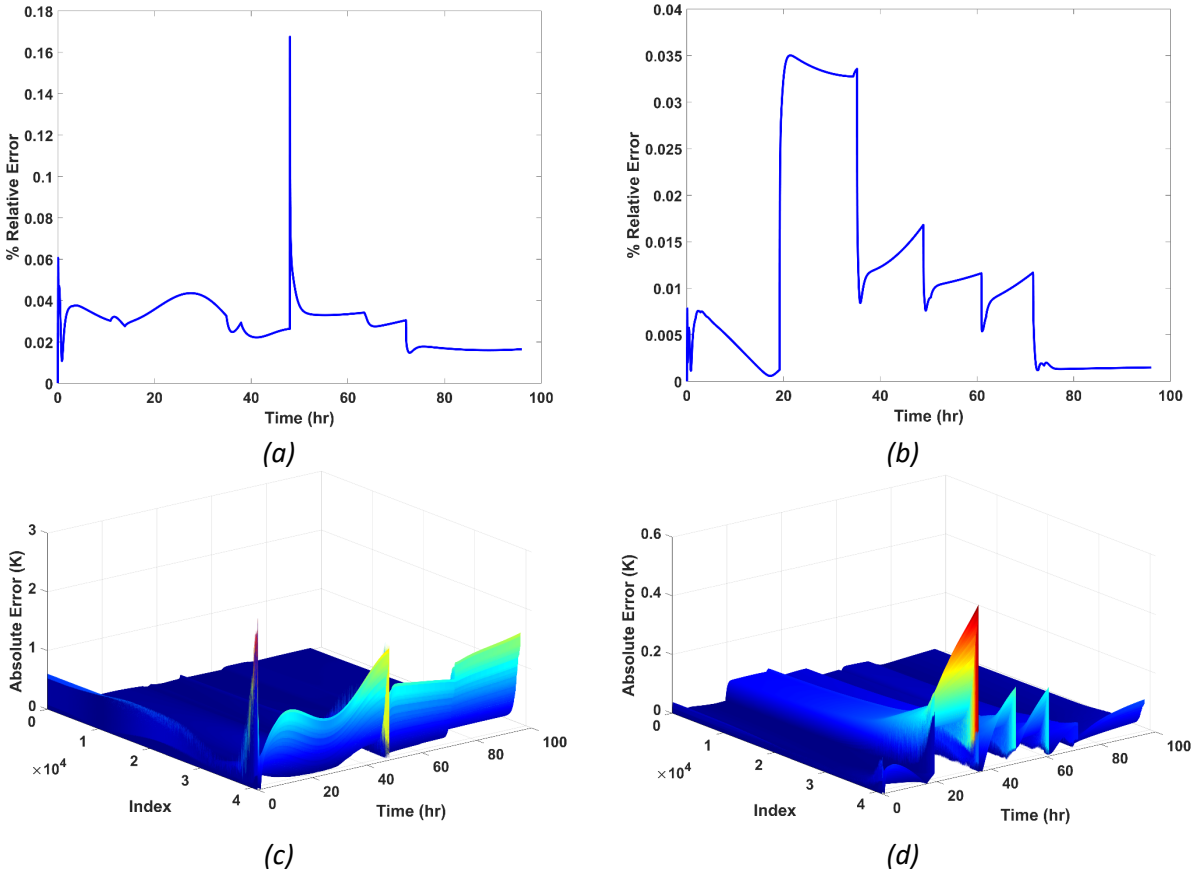


Figure 24. Results of cOptDMDc trained with 16 SVD modes for the (left) Composite and (right) 4-2-1-1 profiles. (Top) the relative errors vs. time; and (bottom) the spatiotemporal distribution of the absolute error.

Figure 24 illustrates the ROM results of cOptDMDc with 16 SVD modes throughout the entire computational domain and across all time steps. The left column and the right column, respectively, show the results for the Composite and the 4-2-1-1 profile. Since ROMs of DMDc/SPDMDc both diverge, they are not included, and only the results of cOptDMDc are present. Comparison of Figure 24 with Figure 18 and the last row in Figure 21 (that takes 17 modes) shows that the ROM accuracy actually improves when 16 SVD modes are used to construct the model. The error for the Composite profile drops from a maximum of 0.4896% with 17 modes to a maximum of 0.1675% with 16 modes, and that for the 4-2-1-1 profile decreases from 0.1761% to 0.035%. The prediction errors generated by cOptDMDc on the 4-2-1-1

input profile tend to decrease as time progresses, as the profile directs the Ahmed Body temperature towards a steady-state. The steady state is strongly affected by the eigenvalues at the proximity of the circumference of the unit circle, and correctly identifying these eigenvalues allows for the model to increase in accuracy as temporal effects start to subside.

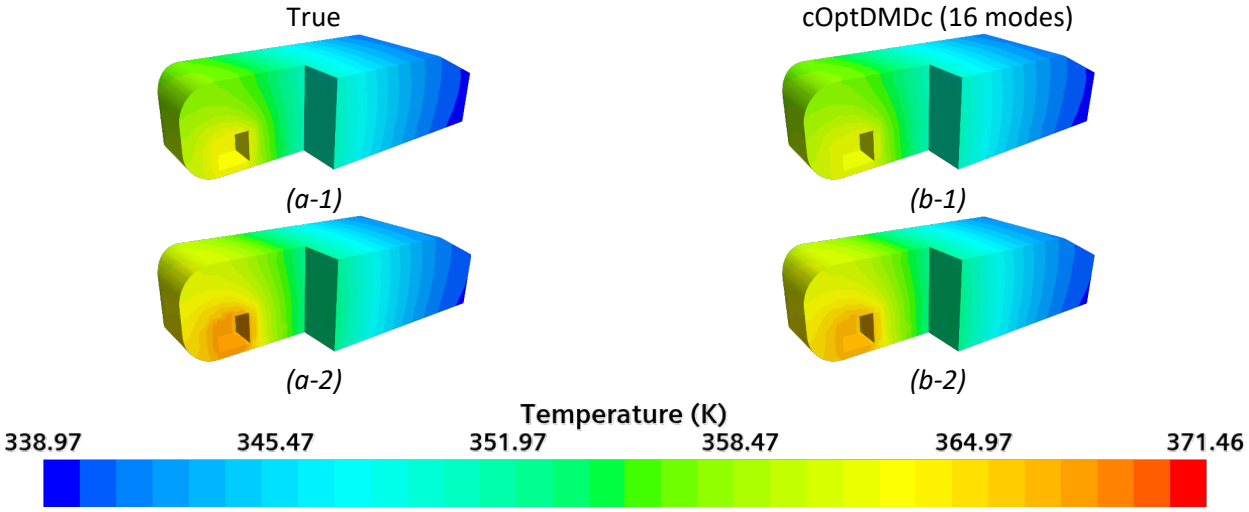


Figure 25. Spatial temperature distribution for the Composite input profile at (top) $t = 32$ hr and (bottom) $t = 64$ hr. The comparison is made between the (left) true temperature and (right) cOptDMDc ROM predictions.

As in the case with the 2D vehicle mockup, the predictions of cOptDMDc are accurate and robust, and are not noticeably affected by the number of modes. In contrast to DMDc/SPDMDc which could not predict reasonable results due to divergence, the accuracy of the cOptDMDc ROM improves when 16 modes are used, which produces a visually almost same temperature distribution as the true values for the Composite input profile as shown in Figure 25. Similarly, Figure 26 displays excellent agreement between cOptDMDc and the true temperature distribution for the 4-2-1-1 profile. The errors are very small, and it is quite difficult to visibly distinguish between the ROM-predicted and true values.

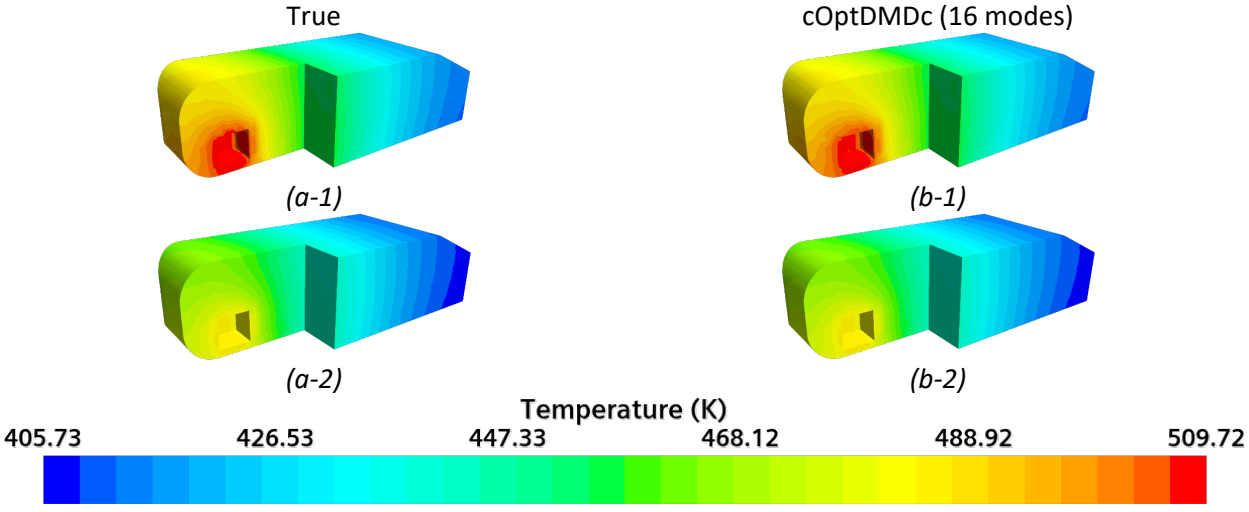


Figure 26. Spatial temperature distribution for the 4-2-1-1 input profile at (top) $t = 32$ hr and (bottom) $t = 64$ hr. The comparison is made between the (left) true temperature and (right) cOptDMDc ROM predictions.

Table 4: Maximum relative (max rel.) and maximum absolute (max abs.) errors between the validation data and the predictions of the three ROMs

Model	3D Ahmed Body							
	16 Modes				17 Modes (DMDc & cOptDMDc) 16 Modes (SPDMDc)			
	Composite		4-2-1-1		Composite		4-2-1-1	
	Max Rel. Error (%)	Max Abs. Error (K)	Max Rel. Error (%)	Max Abs. Error (K)	Max Rel. Error (%)	Max Abs. Error (K)	Max Rel. Error (%)	Max Abs. Error (K)
DMDc	Inf	Inf	Inf	Inf	1.0133	4.3599	0.3182	1.8047
SPDMDc	N/A	N/A	N/A	N/A	5.4209	20.217	15.133	68.075
cOptDMDc	0.1675	3.0894	0.035	0.6784	0.4896	3.3193	0.1761	0.7698

Table 4 shows the comparison in maximum relative (max rel.) and maximum absolute (max abs.) errors between the true validation data and the predictions of the three ROMs with 16 and 17 modes for both the Composite and 4-2-1-1 input profiles. Again, there are several points of note. First, both ROMs of DMDc and cOptDMDc exhibit salient performance and low errors when only 17 SVD modes are used for ROM construction, although cOptDMDc still somewhat exceeds DMDc. However, it should be pointed out that the choice of 17 modes is determined through an iterative trial-and-error process. Due to the truncation of one mode, ROM generated by SPDMDc yields very poor agreement with the validation data, while its results are still temporally stable. Second, when 16 modes are selected, the ROMs generated by DMDc and SPDMDc show divergence in prediction, while our cOptDMDc is robust and still excellently agrees with the true validation data. Third, the comparison of the ROMs generated by cOptDMDc with 16 modes (i.e., Figure 24) and 17 modes (i.e., Figure 18 and Figure 21) indicates the relative errors of the former are around one third of the latter for the Composite profile, and around a fifth for the 4-2-1-1 profile. The maximum absolute errors for both are similar, which may be attributed to the sharp discontinuities in the input profile that are smoothed out by the ROM. The results of all three ROMs for 2D and 3D vehicle test cases show that these models can successfully capture the underlying physics for heat conduction problems, while only cOptDMDc succeeds in every case. This verifies that the proposed eigenvalue constraint process for the optimization makes cOptDMDc a robust model that is not as strongly affected by the number of SVD modes or input profiles.

IV. CONCLUSIONS

In this paper, a new method of constrained optimized DMD with Control (cOptDMDc) is proposed to extend the native optimized DMD to physically stable systems with exogenous inputs and address the spurious eigenvalue issues. It includes two key elements. First, the variable projection approach is modified to allow exponential data fitting with the time series of exogenous inputs, leading to accurate identification of dominant spectral structure. Second, an optimization process based on iterative execution exponential data fitting subject to a linear inequality constraint adjusts the eigenvalue locations of the eigenvalues to preserve the system stability while maximizing data fitting accuracy, which, essentially, converges to a sub-optimal spectral structure that eliminates spurious eigenvalues and prevents the divergent behavior likely occurring for DMDc and SPDMDc. The method is examined against DMDc and SPDMDc to reveal the effect of constrained optimization and iterative exponential data fitting on the solution stability using three case studies of increasing complexity. The technical findings harvested from the case studies and analysis are summarized as follows:

1. cOptDMDc performs similarly to DMDc in terms of accuracy when the latter produces a valid ROM without spurious eigenvalues. Specifically, a maximum relative error < 0.5% in all three case studies is achieved for both cOptDMDc and DMDc. However, the DMDc generates a valid ROM only when the subspace dimension is properly selected. Due to mode truncation, SPDMDc generally underperforms appreciably compared to cOptDMDc and DMDc.
2. DMDc and SPDMDc seem more susceptible to the choice of the ROM dimension. When the dimension is varied, the corresponding ROMs could demonstrate erroneous behavior due to the appearance of the spurious eigenvalues, e.g., system stability changed to unstable or difficulty to construct the ROM. Instead, our cOptDMDc is more robust, and its constrained optimization process of cOptDMDc successfully prevents the divergence of ROM predictions.
3. Although cOptDMDc produces a sub-optimal spectral structure in exchange for preserving system stability during variable project and exponential data fitting, our results in all case studies show that it achieves accuracy similar to DMDc or even exceeds it.
4. Thus, the trial-and-error process and additional validation data used to determine the ROM dimension and tune DMDc-type model construction may not be necessary in the present cOptDMDc framework.

Future work will be focused on accelerating model convergence and computational efficiency and further examining the robustness and independence of the convergence criterion used for the optimization process, as well as other measures to prevent overfitting.

ACKNOWLEDGMENTS

This research was sponsored by the US Army DEVCOM Ground Vehicle Systems Center (GVSC) under contract number W911QX-20-C-0005. The authors would like to acknowledge Dr. Kapil Pant at CFD Research Corporation for providing valuable feedback to the project.

APPENDIX A: Simplification of Eq. (18) to Eq. (19)

This appendix shows how to recast Eq. (18)

$$\eta_k = \sum_{i=1}^{r} \zeta_i e^{\gamma_i t_{k-1}} \alpha_i + \sum_{j=1}^{k-1} \sum_{i=1}^{r} \zeta_i e^{\gamma_i t_{k-j-1}} \vec{\beta}_i^* u_j$$

into the form in Eq. (19)

$$\eta_k = \mathrm{Z}\mathrm{diag}\left(\chi_{k-1} \Phi_{k-1}(\vec{\gamma})\right)$$

Then the optimization problem in Eq. (26) may be solved. First, note that the two sums in the double sum term in Eq. (18) may be interchanged, which yields

$$\eta_k = \sum_{i=1}^{r} \left[\zeta_i e^{\gamma_i t_{k-1}} \alpha_i + \sum_{j=1}^{k-1} \zeta_i e^{\gamma_i t_{k-j-1}} \vec{\beta}_i^* u_j \right].$$

This sum is a matrix vector multiplication between Z and another vector defined by the modal amplitudes α_i and β_i^*, the complex exponentials $e^{\gamma_i t_k}$, and the inputs u_j. Thus, ζ_i can be factored out in the matrix vector multiplication, giving

$$\eta_k = \mathrm{Z}\begin{bmatrix} e^{\gamma_1 t_{k-1}}\alpha_1 + e^{\gamma_1 t_{k-2}}\vec{\beta}_1^* u_1 + e^{\gamma_1 t_{k-3}}\vec{\beta}_1^* u_2 + \cdots + e^{\gamma_1 t_0}\vec{\beta}_1^* u_{k-1} \\ \vdots \\ e^{\gamma_r t_{k-1}}\alpha_r + e^{\gamma_r t_{k-2}}\vec{\beta}_r^* u_1 + e^{\gamma_r t_{k-3}}\vec{\beta}_r^* u_2 + \cdots + e^{\gamma_r t_0}\vec{\beta}_r^* u_{k-1} \end{bmatrix}. \tag{40}$$

Upon rearranging and collecting like terms

$$\eta_k = \mathrm{Z}\begin{bmatrix} e^{\gamma_1 t_{k-1}}\alpha_1 + \vec{\beta}_1^* \left(\sum_{j=1}^{k-1} e^{\gamma_1 t_{k-j-1}} u_j \right) \\ \vdots \\ e^{\gamma_r t_{k-1}}\alpha_r + \vec{\beta}_r^* \left(\sum_{j=1}^{k-1} e^{\gamma_r t_{k-j-1}} u_j \right) \end{bmatrix}. \tag{41}$$

Defining $\Phi_{k-1}(\vec{\gamma}) = e^{\vec{t}_{k-1}\gamma^T}$, which is substituted into Eq. (41) and noting that the sums represent another matrix-vector multiplication, i.e.,

$$\eta_k = \mathrm{Z}\begin{bmatrix} e^{\gamma_1 t_{k-1}}\alpha_1 + \vec{\beta}_1^* \Gamma_0(:,1:k-1)\Phi_{k-2}(\gamma_1) \\ \vdots \\ e^{\gamma_r t_{k-1}}\alpha_r + \vec{\beta}_r^* \Gamma_0(:,1:k-1)\Phi_{k-2}(\gamma_r) \end{bmatrix} = \mathrm{Z}\begin{bmatrix} \left[\alpha_1, \vec{\beta}_1^* \Gamma_0(:,1:k-1)\right]\Phi_{k-1}(\gamma_1) \\ \vdots \\ \left[\alpha_r, \vec{\beta}_r^* \Gamma_0(:,1:k-1)\right]\Phi_{k-1}(\gamma_r) \end{bmatrix}. \tag{42}$$

Defining $\chi_{k-1} = [\vec{\alpha},\ \beta^* \Gamma_0(:,1:k-1)]$, the first component of the vector being multiplied by Z involves only the multiplication of the first row of χ_{k-1} and the first column of $\Phi_{k-1}(\vec{\gamma})$, the second component involves only the multiplication of the second row of χ_{k-1} and the second column of $\Phi_{k-1}(\vec{\gamma})$, and so forth. This yields

$$\eta_k = \mathrm{Z}\mathrm{diag}\left(\begin{bmatrix} \alpha_1 & \vec{\beta}_1^* \Gamma_0(:,1:k-1) \\ & \vdots \\ \alpha_r & \vec{\beta}_r^* \Gamma_0(:,1:k-1) \end{bmatrix} \Phi_{k-1}(\vec{\gamma}) \right) = \mathrm{Z}\mathrm{diag}\big(\chi_{k-1}\Phi_{k-1}(\vec{\gamma})\big). \tag{43}$$

where "diag", similar to MATLAB's command, returns the diagonal terms of a matrix as a column vector. Note that Eq. (43) is Eq. (19) in the main body of the paper, and therefore is the desired result.

APPENDIX B: Derivation of (31)

This appendix shows how to derive Eq. (31), viz.,

$$D_j(p,q) = \begin{cases} 0, & q \neq j \\ \chi_p(j,:)\left(\vec{t}_p \odot \Phi_p(\gamma_j)\right), & q = j \end{cases}$$

and $D_j = \dfrac{\partial \Psi}{\partial \gamma_j}$ is derived to determine the Jacobian matrix of Ψ. First, note that

$$\Psi\big(\vec{\gamma}, \vec{t}_{k-1}\big) = \begin{bmatrix} \mathrm{diag}\big(\chi_1 \Phi_1(\vec{\gamma})\big)^T \\ \vdots \\ \mathrm{diag}\big(\chi_{k-1}\Phi_{k-1}(\vec{\gamma})\big)^T \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_1 e^{\gamma_1 t_0} & \cdots & \alpha_r e^{\gamma_r t_0} \\ \alpha_1 e^{\gamma_1 t_1} + \left(\vec{\beta}_1^* u_1\right) e^{\gamma_1 t_0} & \cdots & \alpha_r e^{\alpha_r t_1} + \left(\vec{\beta}_1^* u_1\right) e^{\gamma_r t_0} \\ \vdots & \ddots & \vdots \\ \alpha_1 e^{\gamma_1 t_{k-1}} + \left(\sum_{m=1}^{k-1} \left(\vec{\beta}_1^* u_m\right) e^{\overline{\gamma}_1 t_{k-m-1}}\right) & \cdots & \alpha_r e^{\gamma_r t_{k-1}} + \left(\sum_{m=1}^{k-1} \left(\vec{\beta}_r^* u_m\right) e^{\gamma_r t_{k-m-1}}\right) \end{bmatrix}. \quad (44)$$

To find the derivative $D_j = \frac{\partial \Psi}{\partial \gamma_j}$, consider the derivative of the i^{th} column of $\Psi\left(\vec{\gamma}, \vec{t}_{k-1}\right)$ with respect to γ_j. If $i \neq j$, then

$$\frac{\partial}{\partial \gamma_j} \begin{bmatrix} \alpha_i e^{\gamma_i t_0} \\ \alpha_i e^{\gamma_i t_1} + \left(\vec{\beta}_i^* u_1\right) e^{\gamma_i t_0} \\ \vdots \\ \alpha_i e^{\gamma_i t_{k-1}} + \left(\sum_{m=1}^{k-1} \left(\vec{\beta}_i^* u_m\right) e^{\gamma_i t_{k-m-1}}\right) \end{bmatrix} = \vec{0}.$$

If $i = j$, then

$$\frac{\partial}{\partial \gamma_j} \begin{bmatrix} \alpha_i e^{\gamma_i t_0} \\ \alpha_i e^{\gamma_i t_1} + \left(\vec{\beta}_i^* u_1\right) e^{\gamma_i t_0} \\ \vdots \\ \alpha_i e^{\gamma_i t_{k-1}} + \left(\sum_{m=1}^{k-1} \left(\vec{\beta}_i^* u_m\right) e^{\gamma_i t_{k-m-1}}\right) \end{bmatrix} = \begin{bmatrix} \alpha_j t_0 e^{\gamma_j t_0} \\ \alpha_j t_1 e^{\gamma_j t_1} + \left(\vec{\beta}_i^* u_1\right) t_0 e^{\gamma_j t_0} \\ \vdots \\ \alpha_j t_{k-1} e^{\gamma_j t_{k-1}} + \left(\sum_{m=1}^{k-1} \left(\vec{\beta}_i^* u_m\right) t_{k-m-1} e^{\gamma_j t_{k-m-1}}\right) \end{bmatrix}. \quad (45)$$

Eq. (45) can be simplified to

$$\begin{bmatrix} \chi_1(j,:)\left(\vec{t}_1 \odot \Phi_1(\gamma_j)\right) \\ \chi_2(j,:)\left(\vec{t}_2 \odot \Phi_2(\gamma_j)\right) \\ \vdots \\ \chi_{k-1}(j,:)\left(\vec{t}_{k-1} \odot \Phi_{k-1}(\gamma_j)\right) \end{bmatrix}. \quad (46)$$

Note that the operator $\odot$ denotes the element-wise (Hadamard) product. This gives the overall expression

$$D_j(p,q) = \begin{cases} 0, & q \neq j \\ \chi_p(j,:)\left(\vec{t}_p \odot \Phi_p(\gamma_j)\right), & q = j \end{cases}$$

which is the desired result.

APPENDIX C: Alternative Definition of Ψ

Starting from Eq. (12)

$$\eta_k = \sum_{i=1}^{r} \zeta_i \mu_i^{k-1} \hat{\zeta}_i^* \eta_1 + \sum_{j=1}^{k-1} \sum_{i=1}^{r} \zeta_i \mu_i^{k-j-1} \hat{\zeta}_i^* G u_j$$

and defining $\Phi_k(\vec{\gamma}) = \Phi(\vec{\gamma}, \vec{t}_k) = e^{\vec{t}_k \vec{\gamma}^T} \in \mathbb{C}^{(k+1)\times r}$, $\vec{t}_k = \begin{bmatrix} t_k \\ \vdots \\ t_1 \\ 0 \end{bmatrix} \in \mathbb{R}^{(k+1)\times 1}$, $\tau_k = [\eta_1 \quad G\Gamma_0(:,1:k)] \in \mathbb{C}^{r\times(k+1)}$, we have

$$\eta_k = [\xi_1 \zeta_1^* \quad \cdots \quad \xi_r \zeta_r^*] \begin{bmatrix} \tau_{k-1}\Phi_{k-1}(\gamma_1) \\ \vdots \\ \tau_{k-1}\Phi_{k-1}(\gamma_r) \end{bmatrix}. \tag{47}$$

Here $W = [\xi_1 \zeta_1^* \quad \cdots \quad \xi_r \zeta_r^*] \in \mathbb{C}^{r\times r^2}$ and $\begin{bmatrix} \tau_{k-1}\Phi_{k-1}(\gamma_1) \\ \vdots \\ \tau_{k-1}\Phi_{k-1}(\gamma_r) \end{bmatrix} \in \mathbb{C}^{r^2\times 1}$. τ_{k-1} may be removed from $\begin{bmatrix} \tau_{k-1}\Phi_{k-1}(\gamma_1) \\ \vdots \\ \tau_{k-1}\Phi_{k-1}(\gamma_r) \end{bmatrix}$ by representing τ_{k-1} left-multiplied with each $\Phi_{k-1}(\gamma_i)$ as a block-diagonal matrix

$$T_{k-1} = \begin{bmatrix} \tau_{k-1} & & \\ & \ddots & \\ & & \tau_{k-1} \end{bmatrix} \in \mathbb{C}^{r^2\times kr},$$

Then,

$$\eta_k = W \begin{bmatrix} \tau_{k-1} & & \\ & \ddots & \\ & & \tau_{k-1} \end{bmatrix} \begin{bmatrix} \Phi_{k-1}(\gamma_1) \\ \vdots \\ \Phi_{k-1}(\gamma_r) \end{bmatrix}. \tag{48}$$

Defining $P_{k-1}(\vec{\gamma}) = \begin{bmatrix} \Phi_{k-1}(\gamma_1) \\ \vdots \\ \Phi_{k-1}(\gamma_r) \end{bmatrix} \in \mathbb{C}^{kr\times 1}$, Eq. (48) becomes $\eta_k = W T_{k-1} P_{k-1}$. As a result,

$$\mathrm{H}_0 = [\eta_1 \quad \eta_2 \quad \cdots \quad \eta_{n-1}] = W[T_1 P_1(\vec{\gamma}) \quad T_2 P_2(\vec{\gamma}) \quad \cdots \quad T_{k-1} P_{n-1}(\vec{\gamma})].$$

A proper expression for $\Psi(\vec{\gamma})$ is required to implement $\mathrm{H}_0^T = \Psi(\vec{\gamma})\Omega$ as that in [13], thereby,

$$\mathrm{H}_0^T = \begin{bmatrix} P_1(\vec{\gamma})^T T_1^T \\ P_2(\vec{\gamma})^T T_2^T \\ \vdots \\ P_{n-1}(\vec{\gamma})^T T_{n-1}^T \end{bmatrix} W^T. \tag{49}$$

T_i^T and $P_i(\vec{\gamma})^T$ may be separated in Eq. (49) with a block-diagonal matrix multiplication, i.e.,

$$\mathrm{H}_0^T = \begin{bmatrix} P_1(\vec{\gamma})^T & & \\ & \ddots & \\ & & P_{n-1}(\vec{\gamma})^T \end{bmatrix} \begin{bmatrix} T_1^T \\ T_2^T \\ \vdots \\ T_{n-1}^T \end{bmatrix} W^T, \tag{50}$$

This would imply $\Psi = \begin{bmatrix} P_1(\vec{\gamma})^T & & \\ & \ddots & \\ & & P_{n-1}(\vec{\gamma})^T \end{bmatrix} \in \mathbb{C}^{(n-1)\times\frac{rn(n-1)}{2}}$, $\begin{bmatrix} T_1^T \\ T_2^T \\ \vdots \\ T_{n-1}^T \end{bmatrix} \in \mathbb{C}^{\frac{rn(n-1)}{2}\times r^2}$, and $\Omega \in \begin{bmatrix} T_1^T \\ T_2^T \\ \vdots \\ T_{n-1}^T \end{bmatrix} W^T = \mathbb{C}^{\frac{rn(n-1)}{2}\times r}$. Since this formulation of Ψ does not include χ, the derivative $\frac{\partial\Psi}{\partial\gamma_j}$ does not either.

However, the Ψ matrix is very large, and very sparse as well. It would be advisable not to construct this matrix, and instead perform matrix-vector multiplication using a function representation of Ψ, where the matrix-vector product Ψx is represented as a function $f(x)$ that returns the product Ψx. This issue lies in computing the pseudoinverse Ψ^+. Since the number of columns $\left(\frac{rn(n-1)}{2}\right)$ of Ψ is much larger than its number of rows $(n-1)$, the pseudoinverse does not have a unique solution. The pseudoinverse calculated is only one of the solutions with the minimized L_2 norm. Furthermore, the calculation of the pseudoinverse would require either the explicit construction of the Ψ matrix or the calculation of its singular value decomposition. The former would potentially require a long time to enumerate each timestep in order to save the matrix in MATLAB's sparse matrix format, while the latter would only require a functional form of the Ψ matrix. The problem then becomes the computational cost of performing the SVD, and the storage of the dense U_Ψ and V_Ψ matrices. When implemented on a computer, MATLAB's sparse randomized SVD algorithm required approximately 6 minutes to evaluate, while the original formulation of Ψ required approximately 50 microseconds on the same computer. This decomposition is required each time a new $\vec{\gamma}$ is determined and a new $\Psi(\vec{\gamma})$ is evaluated, which makes this alternative formulation very computationally intensive.

REFERENCES

[1] J. N. Kutz, S. L. Brunton, B. W. Brunton and J. L. Proctor, "Applications of DMD in fluids," in *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, Philadelphia, Society for Industrial and Applied Mathematics, 2016, pp. 31-38.

[2] P. J. Schmid, "Dynamic Mode Decomposition and Its Variants," *Annual Review of Fluid Mechanics,* vol. 54, no. 1, pp. 225-254, 2022.

[3] G. Berkooz, P. Holmes and J. L. Lumley, "The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows," *Annual Review of Fluid Mechanics,* vol. 25, no. 1, pp. 539-575, 1993.

[4] K. Kunisch and S. Volkwein, "Galerkin Proper Orthogonal Decomposition Methods for a General Equation in Fluid Dynamics," *SIAM Journal on Numerical Analysis,* vol. 40, no. 2, pp. 492-515, 2002.

[5] C. W. Rowley, "Model Reduction for Fluids, Using Balanced Proper Orthogonal Decomposition," *International Journal on Bifurcation and Chaos,* vol. 15, no. 3, pp. 997-1013, 2005.

[6] I. Mezić, "Analysis of Fluid Flows via Spectral Properties of the Koopman Operator," *Annual Review of Fluid Mechanics,* vol. 45, no. 1, pp. 357-378, 2013.

[7] J. N. Kutz, S. L. Brunton, B. W. Brunton and J. L. Proctor, "Applications of DMD in fluids," in *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, Philadelphia, Society for Industrial and Applied Mathematics, 2016, pp. 7-10.

[8] M. S. Hemati, C. W. Rowley, E. A. Deem and L. N. Cattafesta, "De-Biasing the Dynamic Mode Decomposition for Applied Koopman Spectral Analysis," *Theoretical and Computational Fluid Dynamics,* vol. 31, pp. 349-368, 2017.

[9] M. R. Jovanović, P. J. Schmid and J. W. Nichols, "Sparsity-promoting dynamic mode decomposition," *Physics of Fluids,* vol. 26, no. 2, p. 024103, 2014.

[10] M. O. Williams, C. W. Rowley and I. G. Kevrekidis, "A kernel-based method for data-driven koopman spectral analysis," *Journal of Computational Dynamics,* vol. 2, no. 2, pp. 247-265, 2015.

[11] M. O. Williams, I. G. Kevrekidis and C. W. Rowley, "A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition," *Journal of Nonlinear Science,* vol. 25, pp. 1307-1346, 2015.

[12] G. H. Golub and R. J. LeVeque, "Extensions and Uses of the Variable Projection Algorithm for Solving Nonlinear Least Squares Problems," in *Army Numerical Analysis and Computers Conference*, Research Triangle Park, N.C., 1979.

[13] T. Ashkam and J. N. Kutz, "Variable Projection Methods for an Optimized Dynamic Mode Decomposition," *SIAM Journal on Applied Dynamical Systems,* vol. 17, no. 1, pp. 380-416, 2018.

[14] J. P. Boyd, Chebyshev and Fourier Spectral Methods, Mineola, New York: Dover Publications, Inc., 2000.

[15] M. J. Colbrook and A. Townsend, "Rigorous data-driven computation of spectral properties of Koopman operators for dynamical systems," *arXiv:2111.14889,* 2021.

[16] M. R. Jovanović and B. Bamieh, "Componentwise energy amplification in channel flows," *J. Fluid Mech.,* vol. 534, pp. 145-183, 2005.

[17] B. F. Farrell and P. J. Ioannou, "Stochastic forcing of the linearized Navier-Stokes equations," *Physics of Fluids A: Fluid Dynamics,* vol. 5, no. 11, pp. 2600-2609, 1993.

[18] W. M. Orr, "The Stability or Instability of the Steady Motions of a Perfect Liquid and of a Viscous Liquid. Part I: A Perfect Liquid," *Proceedings of the Royal Irish Academy. Section A: Mathetmatical and Physical Sciences,* vol. 27, pp. 9-68, 1907.

[19] J. R. Wright and J. E. Cooper, Introduction to Aircraft Aeroelasticity and Loads, 2nd ed., West Sussex, United Kingdom: Wiley, 2014.

[20] P. V. Overschee and B. de Moor, "N4SID: Numerical Algorithms for State Space Subspace System Identification," *IFAC Proceedings Volumes,* vol. 26, no. 2, Part 5, pp. 55-58, 1993. 12th Triennial World Congress of the International Federation of Automatic control. Volume 5 Associated Technologies and Recent Developments, Sydney, Australia, 18-23 July.

[21] J.-N. Juang, Applied System Identification, Upper Saddle River, NJ: Prentice Hall, 1994.

[22] T. Kim, "Efficient Reduced-Order System Identification for Linear Systems with Multiple Inputs," *AIAA Journal,* vol. 43, no. 7, pp. 1455-1464, 2005.

[23] L. Ljung, System Identification: Theory for the User, 2nd ed., Upper Saddle River, NJ: Prentice-Hall PTR, 1999.

[24] C. R. Marqui, D. D. Bueno, L. C. S. Goes and P. J. P. Gonçalves, "A reduced order state space model for aeroelastic analysis in time domain," *Journal of Fluids and Structures,* vol. 69, pp. 428-440, 2017.

[25] J. I. Shu, Y. Wang, W. Krolick and K. Pant, "Aeroelastic Reduced Order Model with State Consistence Enforcement," *AIAA Journal,* vol. 61, no. 3, pp. 1109-1128, 2023.

[26] J. I. Shu, Y. Wang, A. Brown and A. Kaminsky, "Genetic Algorithm-Guided Development of Parametric Aeroelastic Reduced-Order Models with State-Consistence Enforcement," *AIAA Journal (Accepted),* 2023.

[27] J. Kou and W. Zhang, "Dynamic mode decomposition with exogenous input for data-driven modeling of unsteady flows," *Physics of Fluids,* vol. 31, no. 5, p. 057106, 2019.

[28] J. L. Proctor, S. L. Brunton and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems,* vol. 15, no. 1, pp. 142-161, 2016.

[29] J. Annoni and P. Seiler, "A method to construct reduced-order parameter-varying models," *International Journal of Robust and Nonlinear Control,* vol. 27, no. 4, pp. 582-597, 2017.

[30] J. Annoni, P. Seiler and M. R. Jovanović, "Sparsity-promoting dynamic mode decomposition for systems with inputs," in *2016 IEEE 55th Conference on Decision and Control (CDC), pages 6506–6511*, 2016.

[31] J. L. Proctor, S. L. Brunton and J. N. Kutz, "Generalizing Koopman Theory to Allow for Inputs and Control," *SIAM Journal on Applied Dynamical Systems,* vol. 17, no. 1, pp. 909-930, 2018.

[32] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics,* vol. 11, no. 2, pp. 431-441, 1963.

[33] S. R. Ahmed, G. Ramm and G. Faltin, "Some Salient Features of the Time-Averaged Ground Vehicle Wake," *SAE Transactions,* vol. 93, pp. 473-503, 1984.

[34] M. R. Jovanović, *Modeling, Analysis, and Control of Spatially Distributed Systems,* Ph.D dissertation, Dept. of Mech. Eng., Univ. California, Santa Barbara, 2004.

[35] T. J. Cowan, A. S. Arena and K. K. Gupta, "Acceleration computational fluid dynamics based aeroelastic predictions using system identification," *Journal of Aircraft,* vol. 38, no. 1, pp. 81-87, 2001.

[36] P. G. Hamel and R. V. Jategaonkar, "Evolution of flight vehicle system identification," *Journal of Aircraft,* vol. 33, no. 1, pp. 9-28, 1996.

[37] H. Hesse and R. Palacios, "Reduced-Order Aeroelastic Models for Dynamics of Maneuvering Flexible Aircraft," *AIAA Journal,* vol. 52, no. 8, pp. 1717-1732, 2014.