ISSN 2066-6594 Ann. Acad. Rom. Sci. Ser. Math. Appl. Vol. 15, No. 1-2/2023

ONLINE DYNAMIC MODE DECOMPOSITION: AN ALTERNATIVE APPROACH FOR LOW RANK DATASETS[*]

G.H. Nedzhibov[†]

DOI https://doi.org/10.56082/annalsarscimath.2023.1-2.229

Dedicated to Dr. Dan Tiba on the occasion of his 70^{th} anniversary

Abstract

In this study, we provide an alternative approach for computing the dynamic mode decomposition (DMD) in real-time for streaming datasets. It is a low-storage method that updates the DMD approximation of a given dynamic as new data becomes available. Unlike the standard online DMD method, which is applicable only to overconstrained and full-rank datasets, the new method is applicable for both overconstrained and underconstrained datasets. The method is equation-free in the sense that it does not require knowledge of the underlying governing equations and is entirely data-driven. Several numerical examples are presented to demonstrate the performance of the method.

MSC: 65P99, 37M02, 37L65

keywords: DMD method, Online Dynamic mode decomposition, Koopman operator, Singular value decomposition, Equation-free.

[*]Accepted for publication in revised form on April 2-nd, 2023

[†]g.nedzhibov@shu.bg Faculty of Mathematics and Informatics, Shumen University, Shumen 9700, Bulgaria; Paper written with financial support of Shumen University under Grant RD-08-35/18.01.2023.

1 Introduction

The characterization of complex systems finds application in various fields of the physical, biological, and engineering sciences. As technology advances, data storage capabilities and data transfer speeds increase, which results in the generation of massive data sets. This requires the development of new quantitative methods and data-driven analysis approaches. One such method is the recently developed Dynamic Mode Decomposition (DMD) method, which has been established as a leading technique for identifying spatiotemporal coherent structures from high-dimensional data. Since it was introduced for the first time by Schmid [1] in the fluid mechanics community, it has gained popularity and has been applied in many different fields, such as video processing [2], epidemiology [3], neuroscience [4], financial trading [5, 6, 7], robotics [8], cavity flows [9, 10] and various jets [11, 12]. For a review of the DMD literature, we refer the reader to [13, 14, 15]. For some recent modifications of DMD for non-uniformly sampled data, the higher order DMD method, parallel implementations of DMD and some derivative DMD techniques, we we refer to [19, 20, 21, 22, 23, 24, 25, 26].

In this work, we are interested in a recently developed modification of the DMD method called *Online Dynamic Mode Decomposition* (*Online DMD*) [27], see also [28, 29]. Our goal is to introduce an alternative approach for implementing the Online DMD method. The aim is to propose a scheme that extends the standard approach and overcomes the main drawbacks of the online DMD method.

Among the shortcomings of the standard algorithm are the following assumptions: the number of snapshots has to be much larger than the state dimensions, and the data matrix has to be of full rank.

The scheme we will propose here will have the following advantages over the standard approach:

- The method is applicable to both overconstrained and underconstrained datasets;
- The approach is applicable to full-rank data as well as to low-rank datasets;
- In the case of low-rank data sets, it is possible to predefine a maximum order for truncated or compact SVDs that are performed at each iteration of the process.

We should note that in the particular case of overconstrained and full-rank data, the standard online DMD algorithm would be more cost-effective than the new scheme.

Some advantages of the new scheme compared to the classical DMD method are:

- The DMD operator can be updated incrementally as new snapshots become available without storing previous snapshots;
- The SVD performed at each iteration step is of much lower order, especially for low-rank data.

The outline of the paper is as follows: in the rest of this section, we give a brief summary of the standard DMD and Online DMD methods. In Section 2, we introduce and discuss the new approach to Online DMD method. We then present, in Section 3, numerical examples demonstrating the new scheme. Conclusions are given in Section 4.

1.1 *Dynamic mode decomposition (DMD)*

There are many conceptually equivalent but mathematically different definitions of DMD. In this subsection, we outline the so-called *exact Dynamic Mode Decomposition*, following the descriptions in [13, 14].

Given a data set of snapshot pairs

$$\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{m}, \quad \text{where} \quad \mathbf{x}_i, \mathbf{y}_i \in R^n, \tag{1}$$

spaced a fixed time-interval apart, such that

$$\mathbf{y}_i = F(\mathbf{x}_i), \tag{2}$$

where F is a map associated with the evolution of a given dynamical system. In the standard DMD definition the data is given as a single time series, then for a given snapshot $\mathbf{x}_i$, $\mathbf{y}_i = \mathbf{x}_{i+1}$ is the next snapshot in the time series. Then, we stack the snapshots as a pair of data sets

$$X = [\mathbf{x}_0, \dots, \mathbf{x}_{m-1}] \quad \text{and} \quad Y = [\mathbf{x}_1, \dots, \mathbf{x}_m]. \tag{3}$$

The DMD algorithm seeks the leading eigendecomposition of the best fit linear operator $A \in R^{n\times n}$ such that

$$Y = AX. \tag{4}$$

Equation (4) has a well-known analytical solution in the form

$$A = YX^{\dagger}, \tag{5}$$

where $X^{\dagger}$ is the Moore-Penrose pseudoinverse of X. One possible way to perform the pseudo-inverse is via SVD

$$X = U\Sigma V^{*}, \tag{6}$$

where $U \in \mathbf{R}^{n\times n}$ is unitary, $\Sigma \in \mathbf{R}^{n\times m}$ is diagonal, $V \in \mathbf{R}^{m\times m}$ is unitary. Note that the left singular vectors U represent the modes of *proper orthogonal decomposition* (POD) of X [30].

If there is no A that exactly satisfies (4), then the choice (5) minimizes $\|AX - Y\|_F$, where $\|.\|_F$ is the Frobenius norm. However, it is usually difficult to derive DMD operator A and compute its eigendecomposition, because n describes a large number of states ($n \gg 1$). Therefore, the DMD algorithm attempts to perform a low-rank truncation of the data.

To represent A in a low-dimensional way, matrices U, V and Σ are truncated from the first r modes. Then A is projected to the first r POD modes of X

$$\tilde{A} = U_r^{*} Y V_r \Sigma_r^{-1}, \tag{7}$$

where $U_r \in \mathbf{R}^{n\times r}$, $\Sigma_r \in \mathbf{R}^{r\times r}$ and $V_r \in \mathbf{R}^{r\times m}$. Therefore, we can reconstruct from $\tilde{A}$ the leading nonzero eigenvalues and eigenvectors of A without explicitly computing A. The eigenvectors of $\tilde{A}$ are called *DMD modes*, and the eigenvalues of A are the *DMD eigenvalues*.

Once the DMD modes and eigenvalues are calculated, it is possible to represent the state of the system in terms of DMD modes and eigenvalues. As a result, the time evolution of the system can also be predicted. For more on DMD expansion, different variants of the DMD algorithm and the connections between DMD and the Koopman operator, see [13, 14].

1.2 *Online Dynamic mode decomposition (Online DMD)*

Dynamic mode decomposition for streaming datasets is firstly proposed by Hemati et al. [28], and later in [27] Zhang proposes so-called online DMD. Another related method using an incremental SVD algorithm is presented in [29]. Unlike the standard DMD, where matrix A is determined once, in online DMD, we seek a matrix A_k that varies in time, giving a local linear model for the dynamics. Given snapshot pairs $(\mathbf{x}_i, \mathbf{y}_i)$, defined by (1), for $j = 1, \ldots, k$, we form matrices

$$X_k = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \quad \text{and} \quad Y_k = [\mathbf{y}_1, \ldots, \mathbf{y}_k] \tag{8}$$

which both have dimension $n \times k$. The following assumptions are made:

- The number of snapshots k is large compared with the state dimension n, i.e., we consider the overconstrained case of the dataset ($k > n$).
- The matrix X_k has full row rank, i.e. $rank(X_k) = n$.

The objective of the online DMD method is to provide an alternative way of computing DMD operator such that it can be updated incrementally as new snapshots become available without storing previous snapshots. The approach is based on the following representation:

$$X_k^\dagger = X_k^T (X_k X_k^T)^{-1},$$

which is a direct consequence of the upper assumptions.

The algorithm of online DMD proceeds on each iteration as follows:

Algorithm 1: Online DMD method [27]

1. Collect k snapshot pairs $(\mathbf{x}_j, \mathbf{y}_j)$, $j = 1, \ldots, k$, where $k > n$ is large enough so that $rank(X_k) = n$, where X_k is given by (8).

2. Compute A_k and P_k from:

$$A_k = Y_k X_k^\dagger \quad \text{and} \quad P_k = (X_k X_k^T)^{-1}$$

3. When a new snapshot pair $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ becomes available, update

$$A_{k+1} = A_k + \gamma_{k+1}(\mathbf{y}_{k+1} - A_{k+1}\mathbf{x}_{k+1})\mathbf{x}_{k+1}^T P_k$$

and

$$P_{k+1} = P_k - \gamma_{k+1} P_k \mathbf{x}_{k+1}\mathbf{x}_{k+1}^T P_k,$$

where

$$\gamma_{k+1} = \frac{1}{1 + \mathbf{x}_{k+1}^T P_k \mathbf{x}_{k+1}}.$$

2 Alternative Online DMD

DMD was originally used in the study of large fluid flow fields, where typically $n \gg m$, i.e., the number of state dimension n is large compared with the snapshots m. It is worth emphasizing that the online DMD algorithm relies on an important assumptions: the number of snapshots is much larger than the state dimension. Besides, the main drawback to the online DMD is that it requires the data matrix X_k to be of full rank.

To overcome these shortcomings, we propose a different approach for updating the DMD operator in real time. By analogy to online DMD, given the first k snapshot pairs $(\mathbf{x}_j, \mathbf{y}_j)$ for $j = 1, \ldots, k$, we form matrices

$$X_k = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \text{ and } Y_k = [\mathbf{y}_1, \ldots, \mathbf{y}_k] \tag{9}$$

which both have dimension $n \times k$. We make no assumptions about the relation between n and k or about the rank of the matrix X_k here.

We wish to find an $n \times n$ matrix A_k such that $A_k X_k = Y_k$ approximately holds. We first determine the truncated SVD of X_k

$$X_k = U_k \Sigma_k V_k^*, \tag{10}$$

where $U_k \in R^{n\times r}, V_k \in R^{k\times r}, \Sigma_k \in R^{r\times r}$ and $r = \min(rank(X_k), r_{max})$. The parameter r_{max} is a predefined threshold of truncation $r_{max} \leq rank(X_k)$.

Then, using the identity $X_k^\dagger = V_k \Sigma_k^{-1} U_k^*$, we get

$$A_k = Y_k X_k^\dagger = Y_k V_k \Sigma_k^{-1} U_k^*. \tag{11}$$

As with the standard DMD approach, we can approximate A_k using the projected DMD operator

$$\tilde{A}_k = U_k^* A_k U_k = U_k^* Y_k V_k \Sigma_k^{-1}, \tag{12}$$

which is of dimension $r \times r$. As we know, it is possible to perform the leading eigendecomposition of A_k through the eigendecomposition of $\tilde{A}_k$.

Suppose we have already computed $\tilde{A}_k$ (or A_k) for a given dataset. As time progresses and a new pair of snapshots $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ becomes available, we want to compute the matrix A_{k+1} in a less expensive manner than the standard method (11).

Let us denote the augmented matrices

$$X_{k+1} = [X_k \mid \mathbf{x}_{k+1}] \text{ and } Y_{k+1} = [Y_k \mid \mathbf{y}_{k+1}]. \tag{13}$$

Suppose that the singular value decomposition of X_{k+1} has the form

$$X_{k+1} = U_{k+1}\Sigma_{k+1}V_{k+1}^{*}. \tag{14}$$

From (14), we can express V_{k+1} as

$$V_{k+1} = X_{k+1}^{*}U_{k+1}\Sigma_{k+1}^{-*}. \tag{15}$$

The updated DMD operator is then given by

$$\begin{aligned} A_{k+1} &= Y_{k+1}X_{k+1}^{\dagger} = Y_{k+1}V_{k+1}\Sigma_{k+1}^{-1}U_{k+1}^{*} \\ &= Y_{k+1}X_{k+1}^{*}U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^{*}. \end{aligned} \tag{16}$$

Substituting (13) into (16), we obtain

$$\begin{aligned} A_{k+1} &= [Y_k \mid \mathbf{y}_{k+1}] \begin{bmatrix} X_k^{*} \\ \mathbf{x}_{k+1}^{*} \end{bmatrix} U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^{*} \\ &= \left(Y_kX_k^{*} + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^{*}\right) U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^{*}. \end{aligned} \tag{17}$$

Using the SVD of X_k in (10), we get

$$A_{k+1} = \left(Y_kV_k\Sigma_k^{*}U_k^{*} + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^{*}\right) U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^{*}. \tag{18}$$

Then, applying (11) and after some manipulation, the formula for A_{k+1} becomes

$$A_{k+1} = \left(A_kU_k\Sigma_k^{2}U_k^{*} + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^{*}\right) U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^{*}. \tag{19}$$

The above expression gives a rule for computing A_{k+1} given $U_k, \Sigma_k, U_{k+1}, \Sigma_{k+1}$ and the new snapshot pair $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$.

In what follows, we will show that matrices U_{k+1} and Σ_{k+1} can be expressed from matrices U_k and Σ_k. Based on this and the expression in (12), we can also formulate a way to update $\tilde{A}_k$.

First, it is easy to verify the following identity

$$X_{k+1} = [X_k \mid \mathbf{x}_{k+1}] = [U_k \mid \mathbf{x}_{k+1}] \begin{bmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} V_k^{*} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{20}$$

where $\mathbf{0} \in R^r$ is a zero vector. To obtain an alternative representation of the SVD of X_{k+1}, we will consider two scenarios, depending on whether $\mathbf{x}_{k+1} \in \mathit{range}(U_k)$.

1) If $\mathbf{x}_{k+1} \in range(U_k)$, then the expression

$$\mathbf{x}_{k+1} = U_k U_k^* \mathbf{x}_{k+1} \tag{21}$$

is valid. Therefore, X_{k+1} can be written

$$X_{k+1} = [X_k \mid \mathbf{x}_{k+1}] = [U_k \Sigma_k \mid U_k U_k^* \mathbf{x}_{k+1}] \left[\begin{array}{cc} V_k^* & \mathbf{0} \\ \mathbf{0}^T & 1 \end{array} \right] = U_k B_k \bar{V}_k^*, \tag{22}$$

where we define

$$B_k = [\Sigma_k \mid U_k^* \mathbf{x}_{k+1}] \ \text{ and } \ \bar{V}_k = \left[\begin{array}{cc} V_k & \mathbf{0} \\ \mathbf{0}^T & 1 \end{array} \right]. \tag{23}$$

Obviously, the columns of $\bar{V}_k$ are orthonormal, and $\bar{V}_k^* \bar{V}_k = I$.

Assume the compact SVD of the $r \times (r+1)$ matrix B_k has the form

$$B_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^*, \tag{24}$$

where $\tilde{U}_k \in R^{r\times r}$ is unitary, $\tilde{V}_k \in R^{(r+1)\times r}$ is semi-unitary, and $\tilde{\Sigma}_k \in R^{r\times r}$ is diagonal. Then, from (22) and (24), it follows

$$U_{k+1} = U_k \tilde{U}_k, \quad \Sigma_{k+1} = \tilde{\Sigma}_k \quad (\text{and} \quad V_{k+1} = \bar{V}_k \tilde{V}_k). \tag{25}$$

Now, we can express the reduced order matrix $\tilde{A}_{k+1} = U_{k+1}^* A_k U_{k+1}$ as

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\tilde{A}_k \Sigma_k^2 + U_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* U_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}. \tag{26}$$

2) If $\mathbf{x}_{k+1} \notin range(U_k)$.

Let us define

$$\mathbf{z}_{k+1} = (I - U_k U_k^*) \mathbf{x}_{k+1}$$

and

$$\mathbf{u}_{r+1} = \frac{\mathbf{z}_{k+1}}{\|\mathbf{z}_{k+1}\|}.$$

By using the identity

$$U_k U_k^* \mathbf{x}_{k+1} + \mathbf{u}_{r+1} \|\mathbf{z}_{k+1}\| = \mathbf{x}_{k+1},$$

it follows the expression

$$X_{k+1} = [U_k \mid \mathbf{u}_{r+1}] \left[\begin{array}{cc} \Sigma_k & U_k^* \mathbf{x}_{k+1} \\ \mathbf{0}^T & \|\mathbf{z}_{k+1}\| \end{array} \right] \left[\begin{array}{cc} V_k^* & \mathbf{0} \\ \mathbf{0}^T & 1 \end{array} \right] = \bar{U}_k \bar{B}_k \bar{V}_k^*, \tag{27}$$

where we define

$$\bar{U}_k = [U_k \mid \mathbf{u}_{r+1}] \quad \text{and} \quad \bar{B}_k = \begin{bmatrix} \Sigma_k & U_k^* \mathbf{x}_{k+1} \\ \mathbf{0}^T & \|\mathbf{z}_{k+1}\| \end{bmatrix}. \tag{28}$$

Assume the compact SVD of the $(r+1) \times (r+1)$ matrix $\bar{B}_k$ has the form

$$\bar{B}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^*, \tag{29}$$

where $\tilde{U}_k, \tilde{V}_k \in R^{(r+1)\times(r+1)}$ are unitary, and $\tilde{\Sigma}_k \in R^{(r+1)\times(r+1)}$ is diagonal. Then from (27) and (29) it follows that

$$U_{k+1} = \bar{U}_k \tilde{U}_k, \quad \Sigma_{k+1} = \tilde{\Sigma}_k \quad (\text{and} \quad V_{k+1} = \bar{V}_k \tilde{V}_k). \tag{30}$$

We can express the reduced order matrix $\tilde{A}_{k+1} = U_{k+1}^* A_{k+1} U_{k+1}$ as

$$\tilde{A}_{k+1} = U_{k+1}^* \left(A_k U_k \Sigma_k^2 U_k^* + \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \right) U_{k+1} \Sigma_{k+1}^{-2}, \tag{31}$$

and hence

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\bar{U}_k^* A_k U_k \Sigma_k^2 U_k^* \bar{U}_k + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}, \tag{32}$$

which is an $(r+1) \times (r+1)$ matrix.

An equivalent representation of $\tilde{A}_{k+1}$ using the expression (28) is:

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} U_k^* A_k U_k \Sigma_k^2 \\ \mathbf{u}_{r+1}^* A_k U_k \Sigma_k^2 \end{bmatrix} U_k^* \bar{U}_k + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}, \tag{33}$$

which yields

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} \tilde{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{u}_{r+1}^* Y_k V_k \Sigma_k & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}, \tag{34}$$

where we have used (11) and the identity $U_k^* \bar{U}_k = [I \mid \mathbf{0}]$.

In the particular case of $\mathbf{y}_i = \mathbf{x}_{i+1}$ for $i = 1, \ldots, k$, from the orthogonality of $\mathbf{u}_{r+1}$ and $\mathbf{x}_2, \ldots, \mathbf{x}_k$, it follows

$$\mathbf{u}_{r+1}^* Y_k = [0, \ldots, 0, \mathbf{u}_{r+1}^* \mathbf{x}_{k+1}].$$

Therefore

$$\mathbf{u}_{r+1}^* Y_k V_k = (\mathbf{u}_{r+1}^* \mathbf{x}_{k+1}) \mathbf{v}_k^{lr},$$

where $\mathbf{v}_k^{lr}$ is the last row of matrix V_k. From the SVD of X_k, it follows

$$V_k^* = \Sigma_k^{-1} U_k^* X_k,$$

which yields the last column of V_k^*

$$\left(\mathbf{v}_k^{lr}\right)^* = \Sigma_k^{-1} U_k^* \mathbf{x}_k.$$

Then the expression (34) takes the form

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} \tilde{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{b}_k^* & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}, \tag{35}$$

where

$$\mathbf{b}_k^* = (\mathbf{u}_{r+1}^* \mathbf{x}_{k+1}) \mathbf{x}_k^* U_k \Sigma_k^{-*} \Sigma_k \quad \text{and} \quad \mathbf{y}_{k+1} = \mathbf{x}_{k+2}.$$

Algorithm for an alternative online DMD

For convenience, we consider a time series of data $\mathbf{x}_i$ with $k+1$ snapshots organized in the following two data matrices

$$X_k = [\mathbf{x}_1, \dots, \mathbf{x}_k] \quad \text{and} \quad Y_k = [\mathbf{x}_2, \dots, \mathbf{x}_{k+1}]. \tag{36}$$

Then, we compute the truncated SVD of $X_k = U_k \Sigma_k V_k^*$ as in (10), where $r = rank(X_k)$ is the truncation value, and compute the matrix $\tilde{A}_k = U_k^* Y_k V_k \Sigma_k^{-1}$ as in (12). For each new data point $\mathbf{x}_{k+2}$ the first task is to determine whether the basis contained in U_k should be expanded. To do so, the residual $\mathbf{z}_{k+1} = \mathbf{x}_{k+1} - U_k U_k^* \mathbf{x}_{k+1}$ is computed, and if $\|\mathbf{z}_{k+1}\| / \|\mathbf{x}_{k+1}\|$ is greater than some pre-specified tolerance ϵ, then we expand U_k by appending $\mathbf{z}_{k+1}/\|\mathbf{z}_{k+1}\|$. A single iteration of the algorithm can be summarized, as follows:

Algorithm 2: Alternative online DMD method

Input: Matrices $\tilde{A}_k, U_k, \Sigma_k$, scalar r_{max}, last 3 snapshots $\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{x}_{k+2}$.

Output: Matrices $\tilde{A}_{k+1}, U_{k+1}, \Sigma_{k+1}$.

Compute $\mathbf{z}_{k+1} = (I - U_k U_k^*) \mathbf{x}_{k+1}$ **and proceed**:

If $\|\mathbf{z}_{k+1}\| / \|\mathbf{x}_{k+1}\| < \epsilon$ **:**

1. Construct the matrix

$$B_k = [\Sigma_k \mid U_k^* \mathbf{x}_{k+1}].$$

2. Compute the compact SVD

$$B_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^*.$$

3. Compute the left singular vectors and singular values of X_{k+1}

$$U_{k+1} = U_k \tilde{U}_k \text{ and } \Sigma_{k+1} = \tilde{\Sigma}_k.$$

4. If the DMD modes are required, compute spectral decomposition of

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\tilde{A}_k \Sigma_k^2 + U_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* U_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}.$$

If $\mathbf{w}_j$ is the j-th eigenvector of $\tilde{A}$, then $U_{k+1}\mathbf{w}_j$ is j-th DMD mode.

Else If $\|\mathbf{z}_{k+1}\| / \|\mathbf{x}_{k+1}\| \geq \epsilon$ **:**

1. Construct the matrix

$$\bar{B}_k = \begin{bmatrix} \Sigma_k & U_k^* \mathbf{x}_{k+1} \\ \mathbf{0}^T & \|\mathbf{z}_{k+1}\| \end{bmatrix}.$$

2. Compute the truncated SVD

$$\bar{B}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^*,$$

with truncation value: $min\left(rank(\bar{B}_k), r_{max}\right)$.

3. Compute the left singular vectors and singular values of X_{k+1}

$$U_{k+1} = \bar{U}_k \tilde{U}_k, \quad \text{and} \quad \Sigma_{k+1} = \tilde{\Sigma}_k \ ,$$

where

$$\bar{U}_k = \left[U_k \mid \mathbf{u}_{r+1} \right], \quad \text{and} \quad \mathbf{u}_{r+1} = \frac{\mathbf{z}_{k+1}}{\|\mathbf{z}_{k+1}\|}.$$

4. If the DMD modes are required, compute spectral decomposition of

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} \tilde{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{b}_k^* & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2},$$

where

$$\mathbf{b}_k^* = (\mathbf{u}_{r+1}^* \mathbf{x}_{k+1}) \mathbf{x}_k^* U_k \Sigma_k^{-*} \Sigma_k \quad \text{and} \quad \mathbf{y}_{k+1} = \mathbf{x}_{k+2}.$$

If $\mathbf{w}_j$ is the j-th eigenvector of $\tilde{A}$, then $U_{k+1}\mathbf{w}_j$ is j-th DMD mode.

In practice, the snapshot data is often corrupted by noise. Therefore, in many cases, the data matrix X can be decomposed into a low-rank component that contains the signal and a high-rank component that contains the noise, which results in $rank(X) \sim min(n, m)$. Thus, during the execution of the algorithm, the dimensionality of the basis U_k can grow progressively (at Step 3), even if the streaming data is from low-rank dynamics. In these cases, we can use a predefined threshold for truncation of r_{max} (i.e., the maximum number of retained POD modes). Parameter r_{max} is set as an input parameter in Algorithm 2. In this way, at each iteration step of the process, only the leading r_{max} number of basis vectors and their corresponding eigenvalues are preserved. This is achieved by retaining the first r_{max} columns of $\tilde{U}_k$ and the first r_{max} diagonal elements of $\tilde{\Sigma}_k$, at Step 2 of Algorithm 2.

Computational complexity of Algorithm 2.

In the more complex case (second part of Algorithm 2), the computational cost of each iteration is dominated by the computation of truncated SVD of the $(r+1) \times (r+1)$ matrix $\bar{B}_k$, with a computational cost of $\mathcal{O}(r^3)$. In addition, at each iteration, a matrix multiplication is required to calculate $U_{k+1} = \bar{U}_k \tilde{U}_k$, which costs $\mathcal{O}(nr(r+1))$. Also, at each step, a vector $\mathbf{z}_{k+1} = \mathbf{x}_{k+1} - U_k(U_k^* \mathbf{x}_{k+1})$ is calculated, which requires two multiplications of a vector with a matrix, with a computational cost of $\mathcal{O}(2nr)$. Therefore, the computational cost of each iteration is $\mathcal{O}(nr(r+3) + r^3)$ when the DMD modes and eigenvalues are not required.

If DMD modes are required, then the $r \times r$ matrix $\tilde{A}_{k+1}$ and its eigendecomposition are calculated. After a suitable conversion, matrix $\tilde{A}_{k+1}$ requires three matrix multiplications of square $(r+1) \times (r+1)$ matrices, one matrix multiplication of square $r \times r$ matrices, seven vector matrix multiplications (including the computation of $\mathbf{b}_k^*$) and a rank-one matrix of order $r+1$, with computational cost: $\mathcal{O}((r+1)^3)$, $\mathcal{O}(r^3)$, $\mathcal{O}(n(r+1))$ and $\mathcal{O}((r+1)^2)$, respectively. The computational cost of eigendecomposition of matrix $\tilde{A}_{k+1}$ is $\mathcal{O}((r+1)^3)$. Finally, calculating DMD modes $U_{k+1}\mathbf{w}_j$ requires $\mathcal{O}(n(r+1)^2)$. Therefore, the total complexity for the DMD calculation is $\sim \mathcal{O}(n(r+1)(r+2) + 3r^3)$, where the low degree terms are omitted.

In total, if the DMD modes and eigenvalues are desired after every iteration, the computational cost of Algorithm 2 is $O(nr(2r+7) + 4r^3)$ per iteration, where r is the order of the effective rank of X.

In terms of storage, the algorithm requires the matrices $\tilde{A}_k, Uk, \Sigma_k$, scalar r_{max} and last three snapshot vectors, with a total of $O(n(r+3)+r^2+r+1)$ entries. From what has been stated so far, it follows that Algorithm 2 will be computationally and memory efficient when $r \ll n$ and $r \ll m$. In addition, it does not require previous snapshots to be stored, thus making it useful for applications with large datasets or data streams with large m.

3 Numerical illustrations

In this section, we illustrate the introduced algorithm for calculating alternative online DMD. We also compare alternative online DMD against results from standard DMD. The standard online DMD method (Algorithm 1) is not applicable to the examples considered because $m > n$ (i.e. the systems are overdetermined). The simulations are performed in MATLAB (R2013a) on a personal computer equipped with a 2.2 GHz Intel Core i3 processor.

Example 1. *An illustrative example of a toy problem*

We consider a simple example of incrementally updated DMD on a toy problem to demonstrate the online DMD algorithm introduced above. In this example, we have an arbitrary n-dimensional dynamical system with two characteristic frequencies. A total of m snapshot pairs are measured sequentially and are subject to additive zero-mean Gaussian noise

$$\begin{aligned} \mathbf{x}_j = \quad & \mathbf{v}_1 \cos(2\pi f_1 \triangle tj) + \mathbf{v}_2 \cos(2\pi f_2 \triangle tj) + \\ & \mathbf{v}_3 \sin(2\pi f_1 \triangle tj) + \mathbf{v}_4 \sin(2\pi f_2 \triangle tj) + \mathcal{N}(c), \end{aligned} \tag{37}$$

where $\mathbf{v}_i \in \mathbf{R}^n$ are random state directions, $f_1, f_2 \in \mathbf{R}$ are the characteristic

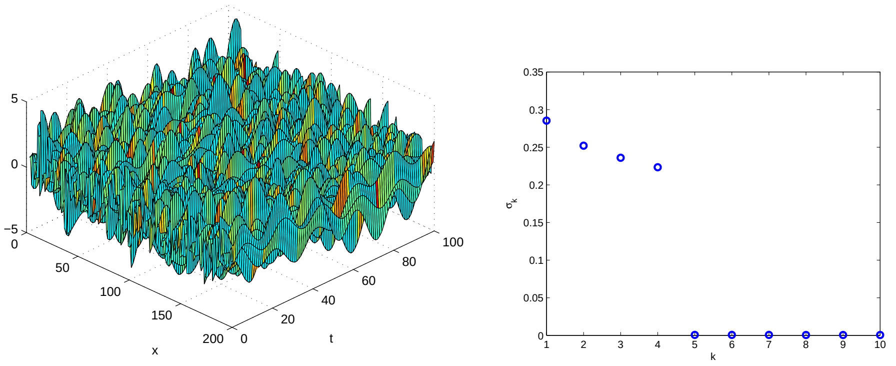

Figure 1: *Spatiotemporal dynamics of signal defined by (39) (left panel) and first 10 singular values of the generated data (right panel).*

frequencies, $\triangle t$ is the time sampling and $\mathcal{N}(c) \in \mathbf{R}^n$ is the independent identically distributed zero-mean Gaussian noise with covariance $c \in \mathbf{R}$.

For the simulation, we set the parameters as follows: $n = 200$, $m = 100$, $f_1 = 5.2$, $f_2 = 1$, $\triangle t = 0.01$ and $c = 0.01$. We define the vectors $\mathbf{v}_i$ and $\mathcal{N}(c)$ using the built-in function *randn* in MATLAB using the following syntax

$$\mathbf{v}_i = \mathrm{randn}(n, 1) \quad \text{and} \quad \mathcal{N}(c) = c * \mathrm{randn}(n, 1). \tag{38}$$

The 3D visualization of the dynamics is illustrated in Fig. 1.

The singular values of data matrix X, illustrated in Fig. 1, show that the data can be adequately represented by the rank-four ($r = 4$) approximation. We initialize the alternative online DMD algorithm by the first three snapshots ($k = 2$), i.e., the initial matrices X_k and Y_k have the form:

$$X_k = [\mathbf{x}_1, \mathbf{x}_2] \quad \text{and} \quad Y_k = [\mathbf{x}_2, \mathbf{x}_3].$$

We performed both methods with a rank reduction of $r = 4$ and with different numbers of snapshots in the interval of 50 to 100. All runs of both algorithms yield the same frequencies and DMD eigenvalues shown in Fig.2.

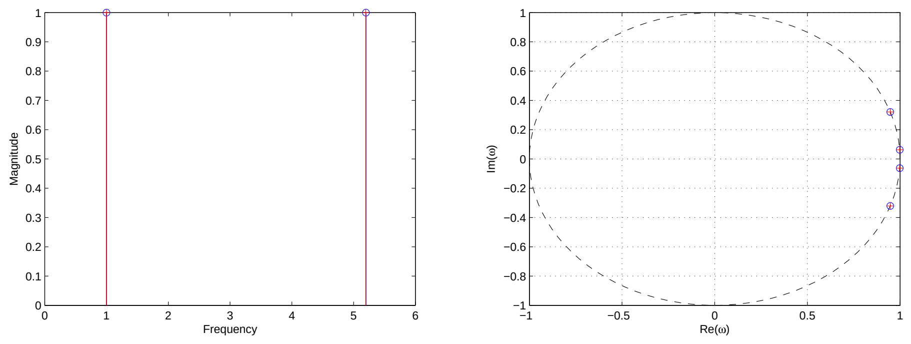

Figure 2: *The two characteristic frequencies computed by standard DMD and alternative online DMD (left panel). The DMD eigenvalues computed by standard DMD ('o') and alternative online DMD ('+'). (right panel).*

To make quantitative comparisons among alternative online DMD and standard DMD, we compute the l_2-norm of the difference between the spatial modes extracted by the two algorithms. We compute the residual errors as follows: $err_i = \|\phi_{DMD}(i) - \phi_{altDMD}(i)\|$ for $i = 1, 2, 3, 4$, where ϕ_{DMD} and ϕ_{altDMD} denotes the DMD modes computed by both algorithms. The error values for two cases, at 50 and 100 snapshots, are shown in Table 1.

♯ snapshots	err_1	err_2	err_3	err_4
$m = 50$	0.0052	0.0052	0.0042	0.0042
$m = 100$	0.0024	0.0024	0.0010	0.0010

Table 1: *Residual errors* $err_i = \|\phi_{DMD}(i) - \phi_{altDMD}(i)\|$.

Example 2. *Spatiotemporal dynamics of two signals*

To demonstrate the alternative online DMD algorithm, we consider an example of two mixed spatio-temporal signals. The signal of interest is

$$X(t) = f_1(\mathbf{x}, t) + f_2(\mathbf{x}, t), \tag{39}$$

where the individual spatiotemporal signals are

$$f_1(\mathbf{x}, t) = sech(\mathbf{x} + 3)e^{i2.3t} \quad \text{and} \quad f_2(\mathbf{x}, t) = 2sech(\mathbf{x})tanh(\mathbf{x})e^{i2.8t}.$$

The mixed signal and the individual spatiotemporal signals are illustrated in Fig. 3 (a)-(c). The two frequencies present are $\omega_1 = 2.3$ and $\omega_2 = 2.8$, which have distinct spatial structures. From the singular value depiction of data matrix X in Fig. 3(d), it follows that the rank-two ($r = 2$) approximation can adequately represent the data.

For the simulation, we use $n = 400$ spatial coordinates in the interval $[-10, 10]$ and $m = 200$ temporal ones in $[0, 4\pi]$. The first three snapshots were used to initialize the alternative online DMD algorithm, i.e., the initial values of the matrices X_k and Y_k in (36) each contain two columns ($k = 2$).

The two algorithms, standard DMD and alternative online DMD, are applied to different numbers of snapshots in the interval of 50 to 200. All runs of both algorithms yield the same DMD modes and DMD eigenvalues shown in Fig.4.

We compute the discrete-time DMD eigenvalues with both algorithms, as shown in Fig. 4

$$\lambda_1 = 0.9844 + 0.1759i \quad \text{and} \quad \lambda_2 = 0.9895 + 0.1447i.$$

Observe that the eigenvalues computed by the alternative online DMD algorithm agree with those identified by standard DMD. As we know, continuous-time DMD eigenvalues are related to discrete-time eigenvalues by

$$\omega_i = ln(\lambda_i)/\triangle t,$$

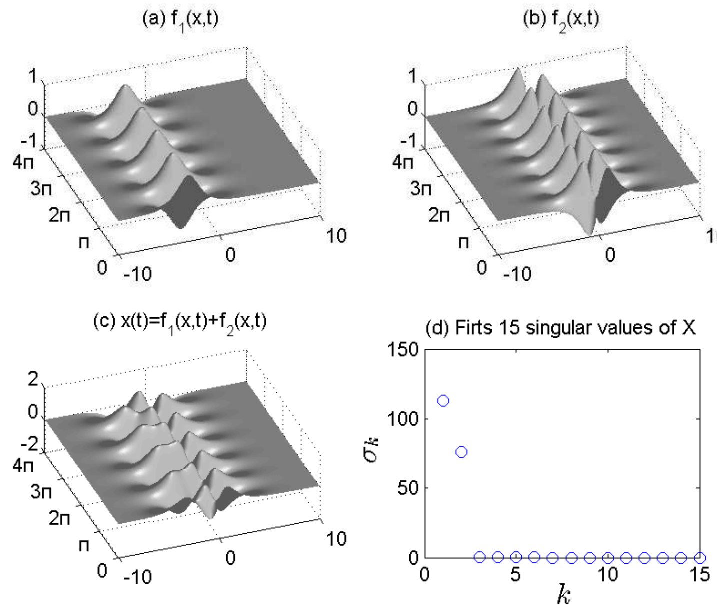

Figure 3: *Spatiotemporal dynamics of two signals (**a**) $f_1(x,t)$, (**b**) $f_2(x,t)$, and mixed signal in (**c**) $X = f_1 + f_2$. Singular values of X are shown in (**d**).*

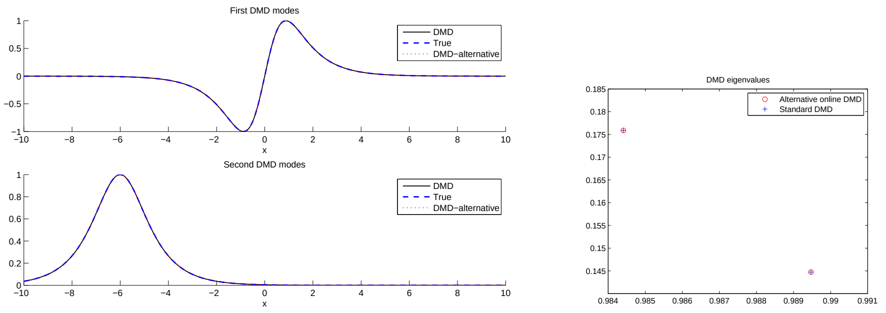

Figure 4: *Firts two DMD modes: true modes, modes extracted by standard DMD and modes extracted by Alternative online DMD (left panel). Discrete-time DMD eigenvalues (right panel)*

where $\triangle t$ is the time spacing between snapshot pairs, in which we get the exact frequencies of oscillation $\omega_1 = 2.3$ and $\omega_2 = 2.8$.

The rank-2 approximations by alternative online DMD and standard

DMD for three different numbers of snapshots are shown in Figures 5-7. It can be seen that the DMD modes obtained by both methods coincide perfectly with the exact DMD modes.

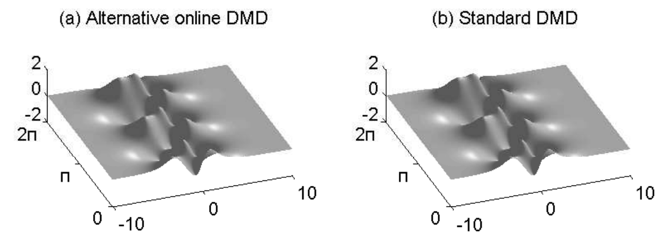

Figure 5: *Reconstruction with* $m = 100$ *snapshots.*

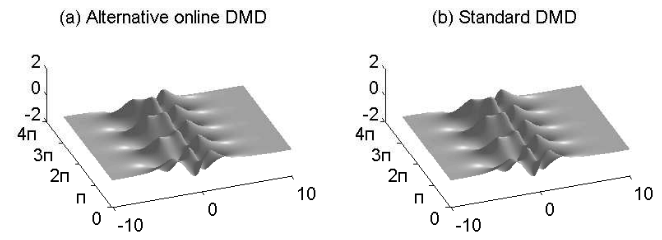

Figure 6: *Reconstruction with* $m = 150$ *snapshots.*

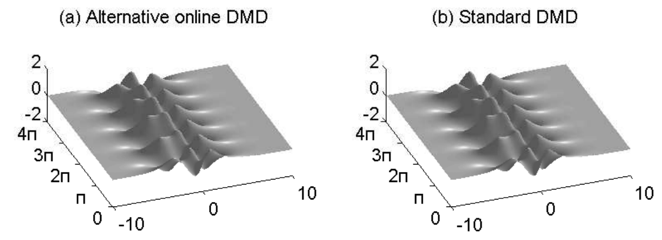

Figure 7: *Reconstruction with* $m = 200$ *snapshots.*

4 Conclusion

The aim of this study was to present a new alternative approach for executing online dynamic mode decomposition. We have introduced and analyzed a new algorithm, an alternative procedure that is applicable in the case of low-rank data. We have demonstrated the performance of the presented algorithms with numerical examples. From the obtained results, we can conclude that the introduced approach gives identical results with those of the *standard online DMD method*. The presented results show that the introduced algorithm can be used for both unconstrained and overconstrained data. The approach does not require the data to be of full rank, suggesting possibilities in various application areas.

References

[1] P. J. Schmid and J. Sesterhenn. Dynamic mode decomposition of numerical and experimental data. In 61st Annual Meeting of the APS Division of Fluid Dynamics. American Physical Society, November 2008.

[2] J. Grosek, J. Nathan Kutz, Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video, arXiv: 1404.7592.

[3] J. L. Proctor and P. A. Eckhoff, Discovering dynamic patterns from infectious disease data using dynamic mode decomposition, International health, 7 (2015), pp. 139-145.

[4] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz, Extracting spatialtemporal coherent patterns in large-scale neural recordings using dynamic mode decomposition, Journal of Neuroscience Methods, 258 (2016), pp. 1-15.

[5] J. Mann and J. N. Kutz, Dynamic mode decomposition for financial trading strategies, Quantitative Finance, (2016), pp. 1-13.

[6] Ling-xiao Cui, Wen Long, 2016. "Trading strategy based on dynamic mode decomposition: Tested in Chinese stock market," Physica A: Statistical Mechanics and its Applications, Elsevier, vol. 461(C), pages 498-508.

[7] D. P. Kuttichira, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using dynamic mode decomposition," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 2017, pp. 55-60, doi: 10.1109/ICACCI.2017.8125816.

[8] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. B. Amor, Estimation of perturbations in robotic behavior using dynamic mode decomposition, Journal of Advanced Robotics, 29 (2015), pp. 331-343.

[9] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. J. Fluid Mech., 656:5-28, August 2010.

[10] A. Seena and H. J. Sung. Dynamic mode decomposition of turbulent cavity ows for selfsustained oscillations. Int. J. Heat Fluid Fl., 32(6):1098-1110, December 2011.

[11] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. J. Fluid Mech., 641:115-127, December 2009.

[12] P. J. Schmid. Application of the dynamic mode decomposition to experimental data. Exp. Fluids, 50(4):1123-1130, April 2011.

[13] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. Journal of Computational Dynamics, 2014, 1 (2) : 391-421. doi: 10.3934/jcd.2014.1.391

[14] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua Proctor (2016). Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, SIAM 2016, ISBN 978-1-611-97449-2, pp. 1-234.

[15] Zhe Bai, Eurika Kaiser, Joshua L. Proctor, J. Nathan Kutz, Steven L. Brunton, Dynamic Mode Decomposition for CompressiveSystem Identification, AIAA Journal, Vol. 58, No. 2, 2020, pp. 561-574.

[16] S.L. Brunton , M.Budišić , E.Kaiser , J.N. Kutz, Modern Koopman Theory for Dynamical Systems, SIAM Review, Vol. 64, Issue 2, 2020, pp. 229-340.

[17] J.L. Proctor, S.L. Brunton, and N.Kutz, Dynamic mode decomposition with control, SIAM Journal on Applied Dynamical Systems, vol. 15, no. 1, 2016, pp. 142161.

[18] 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, 17 (2018), pp. 909930.

[19] S. Le Clainche, J.M.Vega, J. Soria, Higher order dynamic mode decomposition of noisy experimental data: The flow structure of a zero-net-mass-flux jet, , Experimental Thermal and Fluid Science, vol. 88, 2017, https://doi.org/10.1016/j.expthermflusci.2017.06.011

[20] S. Anantharamu, K. Mahesh, A parallel and streaming Dynamic Mode Decomposition algorithm with finite precision error analysis for large data, Journal of Computational Physics, Volume 380, 2019, https://doi.org/10.1016/j.jcp.2018.12.012

[21] T. Sayadi, P.J. Schmid, Parallel data-driven decomposition algorithm for large-scale datasets: with application to transitional boundary layers. Theor. Comput. Fluid Dyn. 30, 415428 (2016). https://doi.org/10.1007/s00162-016-0385-x

[22] K.R. Maryada, S.E. Norris, Reduced-communication parallel dynamic mode decomposition, Journal of Computational Science, Volume 61, 2022. https://doi.org/10.1016/j.jocs.2022.101599

[23] B. Li, J. Garicano-Menaab, E. Valero, A dynamic mode decomposition technique for the analysis of nonuniformly sampled flow data, Journal of Computational Physics, Volume 468, 2022, https://doi.org/10.1016/j.jcp.2022.111495

[24] E. Smith, I. Variansyah, R. McClarren, Variable Dynamic Mode Decomposition for Estimating Time Eigenvalues in Nuclear Systems, https://arxiv.org/abs/2208.10942v1

[25] M.R. Jovanovic, P.J. Schmid, J.W. Nichols, Sparsity-promoting dynamic mode decomposition, Physics of Fluids 26, 2014, https://doi.org/10.1063/1.4863670

[26] F. Guniat, L. Mathelin, L.R. Pastur, A dynamic mode decomposition approach for large and arbitrarily sampled systems, Phys Fluids 27, 2014 https://doi.org/10.1063/1.4908073

[27] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, Online Dynamic Mode Decomposition for Time-Varying Systems, SIAM Journal on Applied Dynamical Systems, 18 (2019), pp. 15861609, https://doi.org/10.1137/18M1192329.

[28] M. S. Hemati, M. O. Williams, and C. W. Rowley, Dynamic mode decomposition for large and streaming datasets, Phy. Fluids, 26 (2014), 111701, https://doi.org/10.1063/1.4901016.

[29] D. Matsumoto and T. Indinger, On-the-Fly Algorithm for Dynamic Mode Decomposition Using Incremental Singular Value Decomposition and Total Least Squares, preprint, arXiv:1703.11004, 2017

[30] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, 2nd ed., Cambridge Monographs on Mechanics (Cambridge University Press, Cambridge, 2012).