computation

MDPI

Article

Extended Online DMD and Weighted Modifications for Streaming Data Analysis

Gyurhan Nedzhibov

Faculty of Mathematics and Informatics, Shumen University, 9700 Shumen, Bulgaria; g.nedzhibov@shu.bg

Abstract: We present novel methods for computing the online dynamic mode decomposition (online DMD) for streaming datasets. We propose a framework that allows incremental updates to the DMD operator as data become available. Due to its ability to work on datasets with lower ranks, the proposed method is more advantageous than existing ones. A noteworthy feature of the method is that it is entirely data-driven and does not require knowledge of any underlying governing equations. Additionally, we present a modified version of our proposed approach that utilizes a weighted alternative to online DMD. The suggested techniques are demonstrated using several numerical examples.

Keywords: DMD method; online dynamic mode decomposition; Koopman operator; singular value decomposition; equation-free

Citation: Nedzhibov, G. Extended Online DMD and Weighted Modifications for Streaming Data Analysis. *Computation* **2023**, *11*, 114. https://doi.org/10.3390/computation11060114

Academic Editor: Alexander Pchelintsev

Received: 5 May 2023
Revised: 1 June 2023
Accepted: 6 June 2023
Published: 9 June 2023

1. Introduction

Dynamic mode decomposition (DMD) has become increasingly popular in studying complex dynamical systems. Since it was introduced for the first time by Schmid [1], it has been successfully applied to a wide range of problems, such as video processing [2], epidemiology [3], neuroscience [4], financial trading [5–7], robotics [8], cavity flows [9,10], and various jets [11,12]. For a review of the DMD literature, we refer the reader to [13–17]. 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 recommend [18–29].

In this work, we are interested in a recently developed modification of the DMD method called *online dynamic mode decomposition* (*online DMD*) [30] (see also [31,32]). We introduce an alternative online DMD method that is applicable for both overconstrained and underconstrained datasets.

The outline of this paper is as follows. In the rest of this section, we give a brief summary of the online DMD and alternative online DMD methods. In Section 2, we introduce and discuss the new approach of the online DMD method. In Section 3, we present examples demonstrating the new algorithm. The conclusion is in Section 4.

1.1. Online Dynamic Mode Decomposition (Online DMD)

In this subsection, the Algorithm 1 of so-called online dynamic mode decomposition (online DMD) is introduced, following the description in [30].

It requires a dataset of snapshot pairs

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

spaced a fixed time interval apart. Then, the snapshots are stacked into the following pair of matrices:

$$X_k = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \quad \text{and} \quad Y_k = [\mathbf{y}_1, \ldots, \mathbf{y}_k]. \tag{2}$$

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, where $k > n$).
- 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 the DMD operator such that it can be updated incrementally as new snapshots become available. The algorithm of online DMD proceeds as follows [30]:

Algorithm 1 Online DMD Method [30]

1. Collect k snapshot pairs $(\mathbf{x}_j, \mathbf{y}_j)$, $j = 1, \ldots, k$, where $k > n$ is large enough that $rank(X_k) = n$, where X_k is given by (2).
2. Compute A_k and P_k from
$$A_k = Y_k X_k^{\dagger} \text{ and } P_k = (X_k X_k^T)^{-1}$$
3. When a new snapshot pair $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$ becomes available, update A_k and P_k:
$$A_{k+1} = A_k + \gamma_{k+1}(\mathbf{y}_{k+1} - A_{k+1}\mathbf{x}_{k+1})\mathbf{x}_{k+1}^T P_k$$
and
$$P_{k+1} = P_k - \gamma_{k+1} P_k \mathbf{x}_{k+1} \mathbf{x}_{k+1}^T P_k,$$
where
$$\gamma_{k+1} = \frac{1}{1 + \mathbf{x}_{k+1}^T P_k x_{k+1}}.$$

1.2. Alternative Online DMD

A novel approach was introduced in [33] to overcome the main shortcomings of online DMD. The new scheme is designed to work with low-rank data, and the condition $n < k$ has been relaxed.

Consider a time series of data $\mathbf{x}_i$ with $k+1$ snapshots organized into the following two data matrices:
$$X_k = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \text{ and } Y_k = [\mathbf{x}_2, \ldots, \mathbf{x}_{k+1}].$$

Then, the truncated SVD $X_k = U_k \Sigma_k V_k^*$ is performed, where $U_k \in R^{n \times r}$, $V_k \in R^{k \times r}$, $\Sigma_k \in R^{r \times r}$, and $r = rank(X_k)$ is the truncation value. The low-dimensional DMD operator is calculated using the formula $\tilde{A}_k = U_k^* Y_k V_k \Sigma_k^{-1}$.

A single iteration of the modified algorithm can be summarized as follows [33].

In this study, we aim to improve the alternative online DMD method (Algorithm 2) from a computational perspective. The next section shows how matrix $\tilde{A}_{k+1}$ can be expressed by using a *diagonal-plus-rank-one matrix* (DPR1 matrix) instead of matrix B_k (or $\bar{B}_k$) in Step 1 of Algorithm 2. As a result, the eigendecomposition of a DPR1 matrix can be used instead of the SVD of a matrix of the same dimension.

Algorithm 2 Alternative Online DMD Method

Input: Matrices $\tilde{A}_k, U_k, \Sigma_k$, scalar r_{max}, last 3 snapshots $\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{x}_{k+2}$.
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, then 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 jth eigenvector of $\tilde{A}$, then $U_{k+1}\mathbf{w}_j$ is jth 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: $\bar{B}_k = \bar{U}_k \tilde{\Sigma}_k \tilde{V}_k^*$, with truncation value $min(rank(B_k), r_{max})$.
3. Compute the left singular vectors and singular values of X_{k+1}:

$$U_{k+1} = \bar{U}_k \tilde{U}_k, \ \Sigma_{k+1} = \tilde{\Sigma}_k, \text{ where } \bar{U}_k = [U_k \mid \mathbf{u}_{r+1}] \text{ and } \mathbf{u}_{r+1} = \frac{\mathbf{z}_{k+1}}{\|\mathbf{z}_{k+1}\|}.$$

4. If the DMD modes are required, then compute the 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

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

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

2. Improved Online DMD

Through analogy with the online DMD, let us consider the k snapshot pairs $(\mathbf{x}_j, \mathbf{y}_j)$ for $j = 1, \ldots, k$ as in Equation (1). We form matrices

$$X_k = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \text{ and } Y_k = [\mathbf{y}_1, \ldots, \mathbf{y}_k], \tag{3}$$

which both have dimensions of $n \times k$. Then, the truncated SVD

$$X_k = U_k \Sigma_k V_k^*, \tag{4}$$

is performed, where $U_k \in R^{n \times r}$, $V_k \in R^{k \times r}$, and $\Sigma_k \in R^{r \times r}$. The truncation value r is such that $r = \min(rank(X_k), r_{max})$, and it is a predefined value. From the pseudoinverse $X_k^\dagger = V_k \Sigma_k^{-1} U_k^*$, we express the DMD operator

$$A_k = Y_k X_k^\dagger = Y_k V_k \Sigma_k^{-1} U_k^*. \tag{5}$$

The projected DMD operator has the form

$$\tilde{A}_k = U_k^* A_k U_k = U_k^* Y_k V_k \Sigma_k^{-1}, \tag{6}$$

which is of dimension $r \times r$. We can perform the leading eigendecomposition of A_k by using the eigendecomposition of $\tilde{A}_k$.

Suppose now that we have already computed $\tilde{A}_k$ (or A_k) for a given dataset. Whenever a new pair of snapshots becomes available $(\mathbf{x}_{k+1}, \mathbf{y}_{k+1})$, we would like to compute matrix A_{k+1} more efficiently than how it is usually carried out (Equation (6)).

Let us denote the augmented matrices

$$X_{k+1} = [X_k \mid \mathbf{x}_{k+1}] \text{ and } Y_{k+1} = [Y_k \mid \mathbf{y}_{k+1}]. \tag{7}$$

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

$$X_{k+1} = U_{k+1}\Sigma_{k+1}V_{k+1}^*. \tag{8}$$

From Equation (8), we express V_{k+1} as

$$V_{k+1} = X_{k+1}^* U_{k+1}\Sigma_{k+1}^{-*}. \tag{9}$$

Then, the updated DMD operator has the form

$$\begin{aligned} A_{k+1} &= Y_{k+1}X_{k+1}^\dagger = Y_{k+1}V_{k+1}\Sigma_{k+1}^{-1}U_{k+1}^* \\ &= Y_{k+1}X_{k+1}^* U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^*. \end{aligned} \tag{10}$$

By substituting Equation (7) into the last expression, we obtain

$$\begin{aligned} A_{k+1} &= [Y_k \mid \mathbf{y}_{k+1}]\begin{bmatrix} X_k^* \\ \mathbf{x}_{k+1}^* \end{bmatrix} U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^* \\ &= \left(Y_k X_k^* + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^*\right)U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^*. \end{aligned} \tag{11}$$

Using the SVD of X_k, we obtain

$$A_{k+1} = \left(Y_k V_k \Sigma_k^* U_k^* + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^*\right)U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^*. \tag{12}$$

By applying Equation (5), the last expression yields

$$A_{k+1} = \left(A_k U_k \Sigma_k^2 U_k^* + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^*\right)U_{k+1}\Sigma_{k+1}^{-2}U_{k+1}^* \tag{13}$$

Equation (13) 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. For this purpose, we consider the covariance matrix

$$C_{k+1} = X_{k+1}^* X_{k+1}, \tag{14}$$

which has the following presentation:

$$C_{k+1} = X_k X_k^* + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^* = U_k \Sigma_k^2 U_k^* + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^*. \tag{15}$$

by using Equation (7) and the SVD of X_k. It is well known that the leading left singular vectors and singular values of X_{k+1} can be computed by the eigendecomposition of C_{k+1}. We will consider two scenarios in order to obtain an alternative representation of the SVD of X_{k+1}, depending on whether $\mathbf{x}_{k+1} \in range(U_k)$:

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

$$\mathbf{x}_{k+1} = U_k U_k^* \mathbf{x}_{k+1} \tag{16}$$

is valid, and the covariance matrix C_{k+1} can be presented as

$$C_{k+1} = U_k\left(\Sigma_k^2 + U_k^*\mathbf{x}_{k+1}\mathbf{x}_{k+1}^*U_k\right)U_k^* = U_kB_kU_k^*, \tag{17}$$

where we define

$$B_k = \Sigma_k^2 + U_k^*\mathbf{x}_{k+1}\mathbf{x}_{k+1}^*U_k \tag{18}$$

as a diagonal-plus-rank-one (DPR1) matrix of a dimension $r \times r$. In this case, the eigendecomposition of C_{k+1} can be reduced to that of a DPR1 matrix B_k plus two matrix multiplications. Assume that the eigendecomposition of the $r \times r$ matrix B_k has the form

$$B_k = \tilde{U}_k\tilde{\Sigma}_k\tilde{U}_k^* \tag{19}$$

where $\tilde{U}_k \in R^{r\times r}$ is unitary and $\tilde{\Sigma}_k \in R^{r\times r}$ is diagonal. Then, for the left singular vectors and singular values of X_{k+1}, we obtain

$$U_{k+1} = U_k\tilde{U}_k, \text{ and } \Sigma_{k+1} = \left(\tilde{\Sigma}_k\right)^{1/2}. \tag{20}$$

Now, from Equation (13), we can express the matrix $\tilde{A}_{k+1} = U_{k+1}^*A_{k+1}U_{k+1}$ as

$$\tilde{A}_{k+1} = \tilde{U}_k^*\left(\tilde{A}_k\Sigma_k^2 + U_k^*\mathbf{y}_{k+1}\mathbf{x}_{k+1}^*U_k\right)\tilde{U}_k\Sigma_{k+1}^{-2}. \tag{21}$$

(2) If $\mathbf{x}_{k+1} \notin range(U_k)$, then let us define

$$\mathbf{z}_{k+1} = (I - U_kU_k^*)\mathbf{x}_{k+1} \tag{22}$$

and

$$\mathbf{u}_{r+1} = \frac{\mathbf{z}_{k+1}}{\|\mathbf{z}_{k+1}\|}. \tag{23}$$

We denote the matrix

$$\bar{U}_k = [U_k \mid \mathbf{u}_{r+1}]. \tag{24}$$

It is straightforward that

$$\bar{U}_k\bar{U}_k^*U_k = U_k \text{ and } \bar{U}_k\bar{U}_k^*\mathbf{x}_{k+1} = \mathbf{x}_{k+1}.$$

By applying the last two expressions in Equation (15), the formula for C_{k+1} becomes

$$C_{k+1} = \bar{U}_k\left(\bar{U}_k^*U_k\Sigma_k^2U_k^*\bar{U}_k + \bar{U}_k^*\mathbf{x}_{k+1}\mathbf{x}_{k+1}^*\bar{U}_k\right)\bar{U}_k^*, \tag{25}$$

which is an $n \times n$ hermitian matrix. From Equation (24) and the definition of $\mathbf{u}_{r+1}$ follows the identity

$$U_k^*\bar{U}_k = U_k^*[U_k \mid \mathbf{u}_{r+1}] = [I \mid \mathbf{0}], \tag{26}$$

where $\mathbf{0} \in R^r$ is a zero vector and $I \in R^{r\times r}$ is the identity matrix.

By substituting Equation (26) into Equation (25), we obtain

$$C_{k+1} = \bar{U}_k\left(\begin{bmatrix} \Sigma_k^2 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \bar{U}_k^*\mathbf{x}_{k+1}\mathbf{x}_{k+1}^*\bar{U}_k\right)\bar{U}_k^* = \bar{U}_k\bar{B}_k\bar{U}_k^*, \tag{27}$$

where

$$\bar{B}_k = \begin{bmatrix} \Sigma_k^2 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \bar{U}_k^*\mathbf{x}_{k+1}\mathbf{x}_{k+1}^*\bar{U}_k \tag{28}$$

is a diagonal-plus-rank-one (DPR1) matrix of the dimensions $(r+1) \times (r+1)$. Assume that the eigendecomposition of B_k is

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

where $\tilde{U}_k \in R^{(r+1)\times(r+1)}$ is unitary and $\tilde{\Sigma}_k \in R^{(r+1)\times(r+1)}$ is diagonal. Then, for the left singular vectors of X_{k+1} (or the eigenvectors of C_{k+1}) and singular values of X_{k+1}, we obtain

$$U_{k+1} = \bar{U}_k \tilde{U}_k \text{ and } \Sigma_{k+1} = (\tilde{\Sigma}_k)^{1/2}. \tag{30}$$

Furthermore, we can compute the reduced-order approximation of A_{k+1}:

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

which is an $(r+1) \times (r+1)$ matrix. An equivalent representation of $\tilde{A}_{k+1}$ using the expression in Equation (26) and the definition of $\tilde{U}_k$ is

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} \tilde{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{u}_{r+1}^* A_k U_k \Sigma_k^2 & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}, \tag{32}$$

where the following expression holds:

$$\mathbf{u}_{r+1}^* A_k U_k \Sigma_k^2 = \mathbf{u}_{r+1}^* Y_k V_k \Sigma_k.$$

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 that

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

Therefore, we have

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

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

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

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 in Equation (32) takes the form

$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} \tilde{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{b}_k^* & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{y}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2}, \tag{33}$$

where

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

The next paragraph summarizes the obtained results as an improved variant of Algorithm 2.

Algorithm for Improved Alternative Online DMD

Let us consider a time series of data $\mathbf{x}_i$ with $k+1$ snapshots organized into the following two data matrices:

$$X_k = [\mathbf{x}_1, \ldots, \mathbf{x}_k] \text{ and } Y_k = [\mathbf{x}_2, \ldots, \mathbf{x}_{k+1}]. \tag{34}$$

Then, we compute the truncated SVD of $X_k = U_k \Sigma_k V_k^*$ as in Equation (4), 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 Equation (6).

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 accomplish this, the residual $\mathbf{z}_{k+1} = \mathbf{x}_{k+1} - U_k U_k^* \mathbf{x}_{k+1}$ is computed, and if $\|\mathbf{z}_{k+1}\|$ 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 updated algorithm can be summarized as follows.

The main advantages of the presented algorithm over Algorithm 2 are in Step 1 and Step 2. Algorithm 3 uses a matrix with a simpler structure in Step 1 (in both parts), namely the DPR1 matrix B_k (or $\bar{B}_k$, respectively). In addition, at Step 2 in Algorithm 3, spectral decomposition of a DPR1 matrix is used instead of SVD of a regular matrix.

Algorithm 3 Improved 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 $\|\mathbf{z}_{k+1}\| < \epsilon$ **:**

1. Construct the DPR1 matrix:
$$B_k = \Sigma_k^2 + U_k^* \mathbf{x}_{k+1} \mathbf{x}_{k+1}^* U_k$$
2. Compute the spectral decomposition:
$$B_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{U}_k^*.$$
3. Compute the left singular vectors and singular values of X_{k+1}
$$U_{k+1} = U_k \tilde{U}_k \text{ and } \Sigma_{k+1} = \left(\tilde{\Sigma}_k\right)^{1/2}.$$
4. If the DMD modes are required, then compute the spectral decomposition of
$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\tilde{A}_k \Sigma_k^2 + U_k^* y_{k+1} x_{k+1}^* U_k\right) \tilde{U}_k \Sigma_{k+1}^{-2}.$$
If $\mathbf{w}_j$ is the jth eigenvector of $\tilde{A}$, then $U_{k+1}\mathbf{w}_j$ is the jth DMD mode.

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

1. Construct the DPR1 matrix:
$$\bar{B}_k = \begin{bmatrix} \Sigma_k^2 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{x}_{k+1} \mathbf{x}_{k+1}^* \bar{U}_k$$
2. Compute the truncated SVD:
$$\bar{B}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{U}_k^*,$$
with truncation value: $min(rank(\bar{B}_k), r_{max})$.
3. Compute the left singular vectors and singular values of X_{k+1}:
$$U_{k+1} = \bar{U}_k \tilde{U}_k \text{ and } \Sigma_{k+1} = \left(\tilde{\Sigma}_k\right)^{1/2},$$
where
$$\bar{U}_k = [U_k \mid \mathbf{u}_{r+1}] \text{ and } \mathbf{u}_{r+1} = \frac{\mathbf{z}_{k+1}}{\|\mathbf{z}_{k+1}\|}.$$
4. If the DMD modes are required, then compute the spectral decomposition of
$$\tilde{A}_{k+1} = \tilde{U}_k^* \left(\begin{bmatrix} \tilde{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{b}_k^* & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{x}_{k+2} \mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2},$$
where
$$\mathbf{b}_k^* = (\mathbf{u}_{r+1}^* \mathbf{x}_{k+1}) \mathbf{x}_k^* U_k \Sigma_k^{-*} \Sigma_k.$$
If $\mathbf{w}_j$ is the jth eigenvector of $\tilde{A}$, then $U_{k+1}\mathbf{w}_j$ is the jth DMD mode.

The main advantages of Algorithm 3 (as well as Algorithm 2) over Algorithm 1 are that it does not require full rank data matrix, and unlike Algorithm 1, Algorithm 3 (and Algorithm 2) does not require the number of snapshots to be greater than the state dimension.

In the next section, we present the framework with which Algorithm 3 (and Algorithm 2) can be modified to a weighted alternative.

3. Weighted Modifications to Online DMD

The algorithms described above are suitable for time-varying systems. They allow the DMD matrix A_k to be updated in real time. In such cases, we may want to give more weight to recent snapshots and less weight to older snapshots. This can be achieved by using a modified cost function. As is known in the standard DMD method, for the given data matrices X_k and Y_k in Equation (3), the DMD operator A_k is found by minimizing the following cost function:

$$P_k = \sum_{i=1}^{k} \|\mathbf{y}_i - A_k \mathbf{x}_i\|^2 = \|Y_k - A_k X_k\|_F^2, \tag{35}$$

where $\|.\|$ denotes the Euclidean norm on the vectors and $\|.\|_F$ denotes the Frobenius norm on the matrices.

In cases where the system changes over time to give more weight to newer snapshots than older snapshots, we can use the modified cost function

$$\hat{P}_k = \sum_{i=1}^{k} \rho^{k-i} \|\mathbf{y}_i - A_k \mathbf{x}_i\|^2, \tag{36}$$

for some constant ρ where $0 < \rho \leq 1$. In fact, for $\rho = 1$, the two formulas match, and for $\rho < 1$, errors in past snapshots are discounted.

We will show below that Algorithm 3 can easily be modified to use this weighting scheme (i.e., to minimize a cost function of the form in Equation (36) instead of the original cost function in Equation (35)). For convenience, let us denote $\rho = \sigma^2$, where $0 < \sigma \leq 1$, and write the cost function (2) as

$$\hat{P}_k = \sum_{i=1}^{k} \|\sigma^{k-i} y_i - A_k \sigma^{k-i} x_i\|^2. \tag{37}$$

Let us define the matrices based on scaled versions of the snapshots as

$$\hat{X}_k = [\sigma^{k-1}\mathbf{x}_1, \sigma^{k-2}\mathbf{x}_2, \ldots, \mathbf{x}_k] \text{ and } \hat{Y}_k = [\sigma^{k-1}\mathbf{y}_1, \sigma^{k-2}\mathbf{y}_2, \ldots, \mathbf{y}_k]. \tag{38}$$

Then, the cost function in Equation (37) can be written as

$$\hat{P}_k = \|\hat{Y}_k - A_k \hat{X}_k\|_F^2. \tag{39}$$

The DMD operator A_k, which is a unique solution that minimizes this cost function, is given by

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

where the truncated SVD $\hat{X}_k = U_k \Sigma_k V_k^*$ is used. Then, the reduced-order DMD operator will have the form

$$\check{A}_k = U_k^* Y_k V_k \Sigma_k^{-1}.$$

On the next $((k+1)$th) step, we want to compute A_{k+1}. Let us denote

$$\hat{X}_{k+1} = [\sigma^k \mathbf{x}_1, \sigma^{k-1}\mathbf{x}_2, \ldots, \sigma\mathbf{x}_k, \mathbf{x}_{k+1}] = [\sigma\hat{X}_k \ \mathbf{x}_{k+1}]$$

and

$$\hat{Y}_{k+1} = [\sigma^k \mathbf{y}_1, \sigma^{k-1}\mathbf{y}_2, \ldots, \sigma \mathbf{y}_k, \mathbf{y}_{k+1}] = [\sigma \hat{Y}_k \ \mathbf{y}_{k+1}].$$

By analogy with Section 2, we can repeat the expressions from Equation (8) to Equation (12) to obtain

$$A_{k+1} = \left(\rho A_k U_k \Sigma_k^2 U_k^* + \mathbf{y}_{k+1}\mathbf{x}_{k+1}^*\right) U_{k+1}\Sigma_{k+1}^{-2} U_{k+1}^* \tag{40}$$

for the updated DMD matrix, where U_{k+1} and Σ_{k+1} are the singular vectors and value matrices of $\hat{X}_{k+1}$, respectively.

In a similar way to that in Equations (14) and (15), we can extract the singular vectors and singular values of $\hat{X}_{k+1}$ through the eigendecomposition of

$$\hat{C}_{k+1} = \hat{X}_{k+1}^* \hat{X}_{k+1} = \sigma^2 \hat{X}_k \hat{X}_k^* + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^* = \rho U_k \Sigma_k^2 U_k^* + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^*. \tag{41}$$

Matrix $\hat{C}_{k+1}$ differs from C_{k+1} in Equation (15) only by a factor of ρ. Again, we will have two scenarios, depending on whether $\mathbf{x}_{k+1} \in range(U_k)$. At each step, we have to factorize one of the following two matrices:

$$\hat{B}_k = \rho \Sigma_k^2 + U_k^* \mathbf{x}_{k+1}\mathbf{x}_{k+1}^* U_k \tag{42}$$

or

$$\check{B}_k = \begin{bmatrix} \rho\Sigma_k^2 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{x}_{k+1}\mathbf{x}_{k+1}^* \bar{U}_k, \tag{43}$$

where $\bar{U}_k = [U_k \mid \mathbf{u}_{r+1}]$, $\mathbf{u}_{r+1} = \frac{\mathbf{z}_{k+1}}{\|\mathbf{z}_{k+1}\|}$, and $\mathbf{z}_{k+1} = (I - U_k U_k^*)\mathbf{x}_{k+1}$. The matrices $\hat{B}_k$ and $\check{B}_k$ differ only by a factor of ρ from the corresponding matrices B_k and $\bar{B}_k$ in Equations (18) and (28), respectively. Then, the singular vectors and values of $\hat{X}_{k+1}$ are given by

$$U_{k+1} = U_k \tilde{U}_k, \ \ \Sigma_{k+1} = \left(\tilde{\Sigma}_k\right)^{1/2} \ \text{ or } \ U_{k+1} = \bar{U}_k \tilde{U}_k, \Sigma_{k+1} = \left(\tilde{\Sigma}_k\right)^{1/2}, \tag{44}$$

where $\tilde{U}_k$ and $\tilde{\Sigma}_k$ are from the eigendecomposition of $\hat{B}_k$ or $\check{B}_k$.

The reduced order DMD matrix is then given by

$$\check{A}_{k+1} = \tilde{U}_k^* \left(\rho \check{A}_k \Sigma_k^2 + U_k^* \mathbf{y}_{k+1}\mathbf{x}_{k+1}^* U_k\right) \tilde{U}_k \Sigma_{k+1}^{-2} \tag{45}$$

or

$$\check{A}_{k+1} = \tilde{U}_k^* \left(\rho \begin{bmatrix} \check{A}_k \Sigma_k^2 & \mathbf{0} \\ \mathbf{u}_{r+1}^* \hat{A}_k U_k \Sigma_k^2 & 0 \end{bmatrix} + \bar{U}_k^* \mathbf{y}_{k+1}\mathbf{x}_{k+1}^* \bar{U}_k \right) \tilde{U}_k \Sigma_{k+1}^{-2} \tag{46}$$

for the two considered scenarios. Observe that the updates in Equations (45) and (46) for the reduced order approximation $\check{A}_{k+1}$ are similar to the updates in Equations (31) and (32). The only difference is the factor of ρ.

Therefore, the algorithm for weighted alternative online DMD follows the same steps as in Algorithm 3. The only difference is the calculation of matrices $\hat{B}_k$ (or $\check{B}_k$) and $\check{A}_{k+1}$, where Equations (42), (43) and (45), (46) are used, respectively. We have to note that the choice of the weighting factor ρ matters. Therefore, for example, choosing $\rho^{-1/m}$ will ensure that snapshots have a *half-life* of m samples [30]. In practice, the choice of ρ depends on the rate of change of the dynamics. Choosing a smaller ρ value will result in faster tracking while making the model more sensitive to noise in the data.

4. Numerical Illustrations

The new algorithm (Algorithm 3), introduced in Section 2, offers the advantage of being more cost-effective than the alternative algorithm (Algorithm 2). Considering that Algorithms 2 and 3 produce identical results, in this part, we will mainly present the results of the application of Algorithm 3 and its comparison with standard methods.

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

We consider a simple example of incrementally updated DMD in a toy problem. 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 with covariance:

$$\begin{aligned} \mathbf{x}_j = \quad & \mathbf{v}_1 \cos(2\pi f_1 \triangle t j) + \mathbf{v}_2 \cos(2\pi f_2 \triangle t j) + \\ & \mathbf{v}_3 \sin(2\pi f_1 \triangle t j) + \mathbf{v}_4 \sin(2\pi f_2 \triangle t j) + \mathcal{N}(c), \end{aligned} \tag{47}$$

where $\mathbf{v}_i \in \mathbf{R}^n$ are random state directions, $f_1, f_2 \in \mathbf{R}$ are the characteristic 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 $c = 0.01$. By using the built-in function randn in Matlab, we defined the vectors $\mathbf{v}_i$ and $\mathcal{N}(c)$ via the following syntax:

$$\mathbf{v}_i = \text{randn}(n, 1) \text{ and } \mathcal{N}(c) = c * \text{randn}(n, 1). \tag{48}$$

The 3D visualization of the dynamics is illustrated in Figure 1. The singular values of data matrix X, illustrated in Figure 1, show that the data can be adequately represented by the rank-four ($r = 4$) approximation.

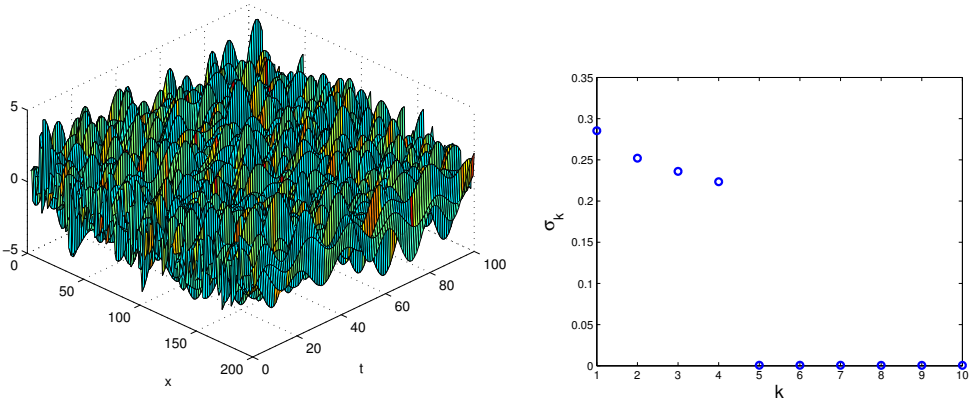

Figure 1. Spatiotemporal dynamics of signal defined by Equation (47) (**left panel**) and first 10 singular values of the generated data (**right panel**).

The standard online DMD method (Algorithm 1) is not applicable in this case because $m > n$. Therefore, here we will compare the standard DMD method and improved online DMD method (Algorithm 3). We initialized the improved online DMD algorithm with the first three snapshots ($k = 2$) (i.e., the initial matrices X_k and Y_k have the form $X_k = [\mathbf{x}_1, \mathbf{x}_2]$ and $Y_k = [\mathbf{x}_2, \mathbf{x}_3]$, respectively).

We performed both methods with rank reduction of $r = 4$. The two algorithms, standard DMD and Algorithm 3, were applied to different numbers of snapshots in the interval from 50 to 100. All runs of both algorithms yielded the same DMD eigenvalues shown in Figure 2.

To make quantitative comparisons among alternative online DMD and standard DMD, we computed the l_2 norm of the difference between the spatial modes extracted by the two algorithms. We computed 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} denote the DMD modes computed by both algorithms. The error values for two cases at 50 and 100 snapshots are shown in Table 1.

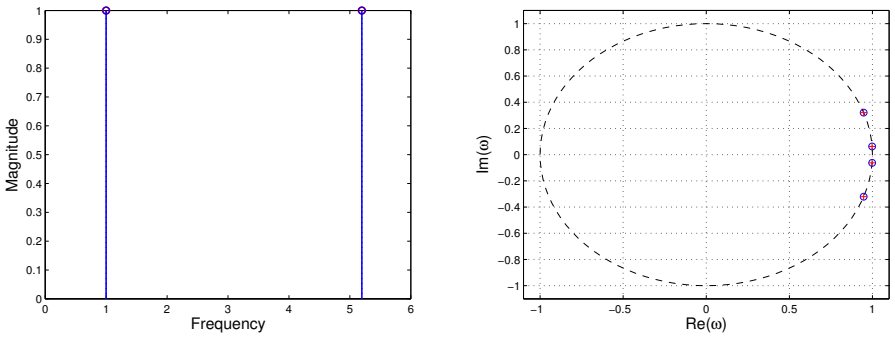

Figure 2. The two characteristic frequencies computed by Algorithms 1 and 3 (**left panel**). The DMD eigenvalues computed by online DMD ('o') and improved online DMD ('+') (**right panel**).

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

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

Example 2. *Flow around a cylinder wake (*$Re = 100$*).*

In this example, we consider a dataset representing a time series of fluid vorticity fields for the wake behind a circular cylinder at Reynolds number $Re = 100$. The data for this example are publicly available at www.siam.org/books/ot149/flowdata (see [15]). The collected data consisted of $m = 150$ snapshots at regular intervals in time $10\triangle t$, where $\triangle t = 0.02$, sampling five periods of vortex shedding.

Each vorticity field snapshot from the finest computational domain was reshaped into a large vector $\mathbf{x}_k$, and these vectors comprised columns of matrices X and Y, as described in Equation (34). A snapshot example of the cylinder wake data is shown in Figure 3.

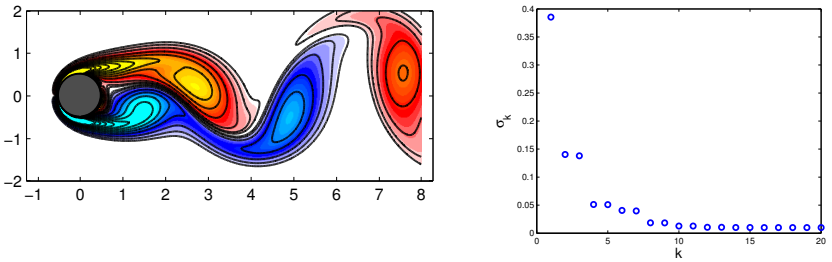

Figure 3. An example of a vorticity field (**left panel**) and first 20 singular values of data matrix X (**right panel**).

Here, we compare the standard DMD method and the improved online DMD method, since online DMD (Algorithm 1) is not applicable. In order to capture the low-dimensional dynamic, we performed rank-seven truncation. We used Algorithm 3 to obtain the DMD reconstruction of the data. The two algorithms reproduced the same DMD eigenvalues and modes. We compared the results with the standard DMD algorithm for the same rank-seven truncation. The DMD eigenvalues computed by the improved alternative online DMD and standard DMD algorithms are shown in Figure 4.

The first 6 DMD modes computed by the improved online DMD and standard DMD algorithms are shown in Figure 5 and Figure 6, respectively.

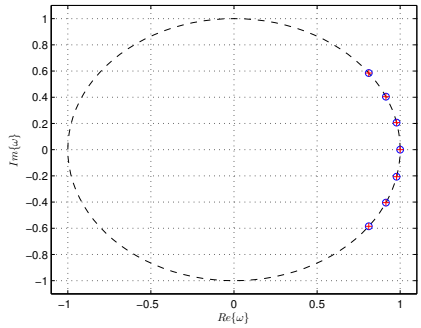

Figure 4. DMD eigenvalues computed by standard DMD ('o') and improved online DMD ('+').

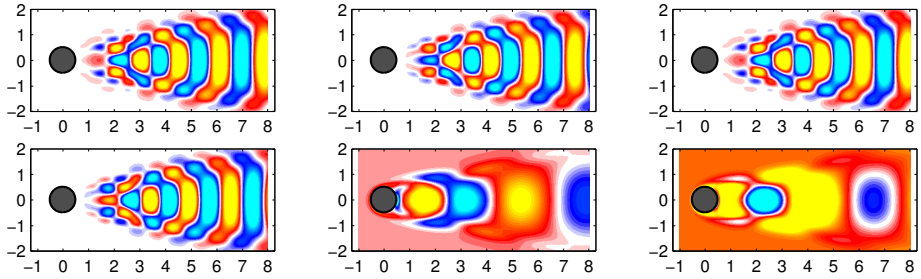

Figure 5. First 6 DMD modes computed by the improved online DMD algorithm (Algorithm 3).

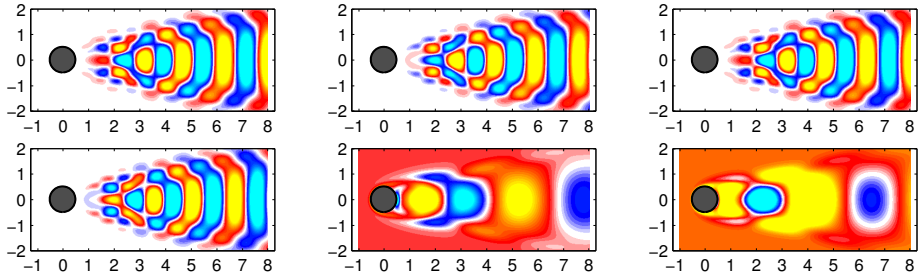

Figure 6. First 6 DMD modes computed by the standard DMD algorithm.

In Figure 7, some examples of data reconstruction by the standard DMD and improved online DMD methods are shown.

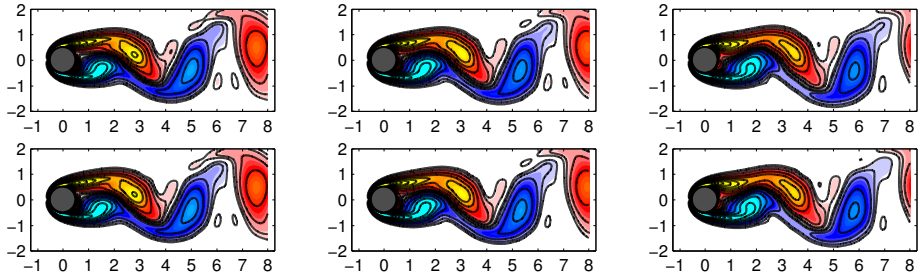

Figure 7. Reconstructed data (three example snapshots) computed by improved online DMD in the top row and by standard DMD in the bottom row.

There was no distinction between the DMD modes and eigenvalues generated by the standard DMD algorithm (Algorithm 1) and Algorithm 3.

Example 3. *Flow around a cylinder wake with added noise.*

It is well known that the DMD method's performance depends on the cleanliness of the data collected. The noise in the collected data can affect the discovered model's

accuracy. In this example, we consider the same fluid vorticity fields for the wake behind a circular cylinder at Reynolds number $Re = 100$, as in Example 2. Measurement noise was generated by adding Gaussian white noise to the data matrix.

As in Example 2, the data-set consisted of $m = 150$ snapshots $\mathbf{x}_k$, where each snapshot had $n = 89,351$ coordinates (i.e., the data matrix $X \in R^{89,351 \times 150}$).

The updated snapshots were generated by

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k + \mu_k(\varepsilon),$$

where $\mu_k(\varepsilon)$ is a zero-mean Gaussian white noise process with unit standard deviation. In this example, we kept the noise variance fixed at $\varepsilon = 0.2$.

Here, we compare the standard DMD method and the improved online DMD method. As in Example 2, we performed rank-seven truncation. We used Algorithm 3 to obtain the DMD reconstruction of the data. The two algorithms reproduced the same DMD eigenvalues and modes. Figure 8 shows some examples of data reconstruction by the standard DMD and improved online DMD methods for the noisy flow data.

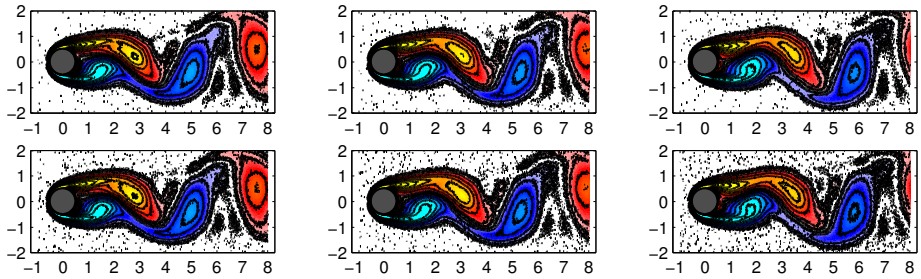

Figure 8. Reconstructed data (three example snapshots) computed by improved online DMD in the top row and by standard DMD in the bottom row for the noisy flow data.

The first four DMD modes computed by the improved online DMD and standard DMD algorithms are shown in Figure 9 and Figure 10, respectively.

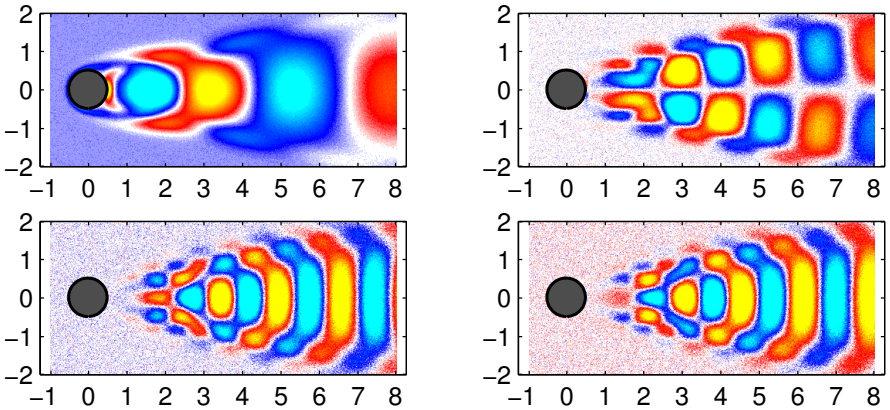

Figure 9. Four consecutive DMD modes computed by the improved online DMD algorithm (Algorithm 3) for the noisy flow data.

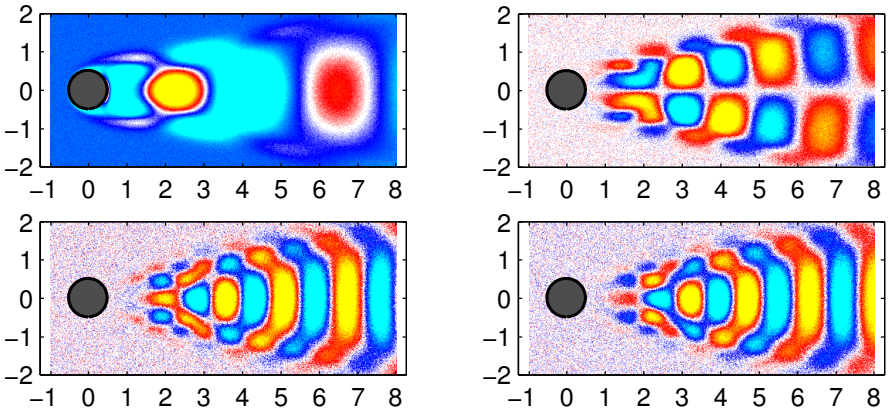

Figure 10. Four consecutive DMD modes computed by the standard DMD algorithm for the noisy flow data.

Note again that there was no distinction between the modes generated by the standard DMD algorithm and Algorithm 3. The visualizations of the DMD eigenvalues calculated by the improved alternative online DMD method and the standard DMD method were omitted as they matched those shown in Figure 4.

Example 4. *Linear time-varying system.*

We now illustrate the improved alternative online DMD algorithm in a simple linear system that slowly varies over time:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t), \tag{49}$$

where $\mathbf{x}(t) \in R^2$, and the time-varying matrix $A(t)$ is given by

$$A(t) = \begin{pmatrix} 0 & \omega(t) \\ -\omega(t) & 0 \end{pmatrix},$$

where $\omega(t) = 1 + 0.1t$ (see [30]). The eigenvalues of $A(t)$ are $\pm iw(t)$, and $\|x(t)\|$ is constant in t. We simulated the system for $0 < t < 10$ from the initial condition $\mathbf{x}(0) = (1, 0)^T$, and the snapshots were taken with a time step of $\triangle t = 0.1$, as shown in Figure 11.

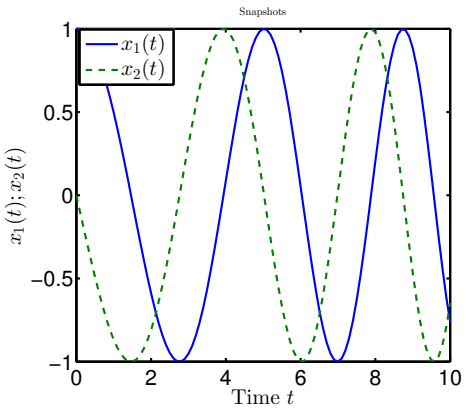

Figure 11. Linear time-varying system defined by Equation (49).

With the snapshots as input, we applied standard DMD, online DMD (Algorithm 1), and improved alternative online DMD (Algorithm 3) and compared the results. The three methods used the first $k = 10$ snapshot pairs to initialize and start iterating from time

$k+1$. The discrete-time eigenvalues μ_{DMD} computed by the algorithms were converted to continuous-time DMD eigenvalues λ_{DMD} by the formula

$$\mu_{DMD} = e^{\lambda_{DMD} \triangle t},$$

where $\triangle t = 0.1$ is the time spacing between snapshot pairs. Observe from Figure 12 that the eigenvalues computed by the alternative online DMD algorithm agreed with those identified by the standard DMD and online DMD algorithms. The true eigenvalues are also shown for comparison. The DMD modes calculated by the three algorithms were also identical, as can be seen from Figure 12.

The alternative weighted online DMD algorithm was applied under the same conditions. A comparison was made between the results of the weighted variant of Algorithm 3 and the weighted online DMD algorithm introduced in [30].

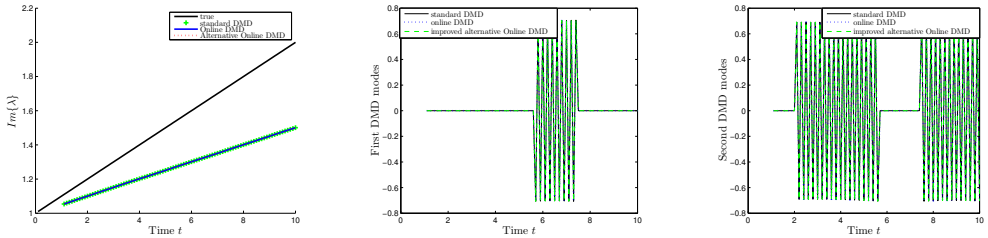

Figure 12. DMD eigenvalues (**left panel**) and DMD modes (**right two panels**) computed by standard DMD, online DMD and improved online DMD.

As before, the results show identical DMD modes and DMD eigenvalues for both algorithms. Figure 13 shows the eigenvalues and modes computed by the weighted online DMD and alternative weighted online DMD methods with a factor $\rho = 0.9$.

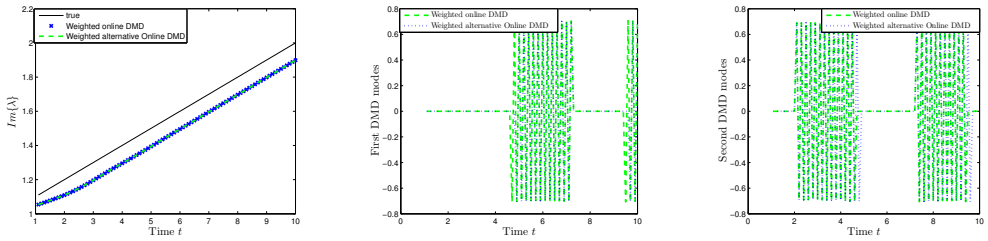

Figure 13. DMD eigenvalues (**left panel**) and modes (**right two panels**) computed by weighted online DMD and alternative weighted online DMD with $\rho = 0.9$.

5. Discussion

The introduced variants of online dynamic mode decomposition approximate the Koopman operator as well as its eigenvalues and modes by processing continuous data streams. Online DMD has applications in various fields, including control systems, sensor networks, weather forecasting, and monitoring dynamic systems in real time. By providing up-to-date decomposition of the data, it allows for the identification of evolving patterns, prediction of future behavior, and monitoring system health in real time.

Online DMD offers several advantages over the standard DMD method when analyzing streaming or real-time data:

- *Computational efficiency and reduced memory requirements*: By processing data incrementally, online DMD avoids redundant computations of previously analyzed data, which results in improved efficiency. It only requires storage for the most recent data rather than the entire dataset, which reduces the memory requirements.
- *Continuous monitoring and prediction*: Online DMD allows for continuous monitoring of evolving systems and the ability to make predictions based on the estimated modes.

This is valuable for applications where forecasting or early detection of anomalies is essential, such as in weather forecasting or fault detection.

- *Flexibility in data input*: Online DMD can handle data streams with missing or irregularly spaced samples, accommodating the challenges often encountered in real-time data acquisition and processing.
- *Adaptability to changing dynamics*: Streaming data often exhibit time-varying or non-stationary behavior. Online DMD and the weighted online DMD modifications continuously update the decomposition using the most recent data, allowing them to capture and adapt to changes in the underlying dynamics.

In this paper, we introduced extended modifications of the standard online DMD method introduced by Zhang et al. in [30]. The main advantages of the proposed modifications over the standard online DMD scheme are that they are applicable regardless of whether the data system is full-rank or not. In addition, the requirement that the number of snapshots be larger than the state dimension is dropped.

The ability of the introduced techniques to handle streaming data and provide real-time insights make them valuable in various scientific domains where continuous monitoring, prediction, and analysis of dynamic systems are essential. Such areas of application include structural health monitoring, biomedical research, weather forecasting, and robotics and control systems. Additionally, these techniques can be useful in scenarios where there are large snapshots and real-time analysis is required. Such areas of application include image and video processing, computational biology and genomics, financial markets, and social media and web analytics. Some related papers include [34–37].

We believe that there is a scope for further research and applications of the presented methods for online DMD. Potential directions for further research and development include adaptive and data-driven parameter selection, online DMD for multi-modal and heterogeneous data, and integration of online DMD with machine learning techniques. Some related papers include [38–41].

6. Conclusions

This study presents efficient alternative algorithms for computing online DMD and weighted online DMD. In contrast to conventional DMD algorithms, the algorithms under consideration use much less memory. They are helpful for engineering applications requiring a large amount of data, as they can be run with a small amount of memory and without storing the flow field data on a storage device. Both of the proposed algorithms work for either over-constrained or under-constrained problems (i.e., regardless of the ratio between the snapshots and state dimensions). The proposed techniques are also applicable to low-rank systems, unlike standard online DMD or weighted online DMD methods. The algorithms were also demonstrated on a variety of examples, such as a linear time-varying system and wake flow data. The obtained results indicate that the proposed approach produces identical results to those obtained by the *online DMD method* as well as the *alternative online DMD method*. As a result, we can conclude that the algorithms introduced can capture dynamics that are time-varying effectively.

Funding: This research received no external funding.

Data Availability Statement: No publicly archived data.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Schmid, P.J.; Sesterhenn, J. Dynamic mode decomposition of numerical and experimental data. In Proceedings of the 61st Annual Meeting of the APS Division of Fluid Dynamics, San Antonio, CA, USA, 23–25 November 2008.
2. Grosek, J.; Nathan Kutz, J. Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video. *arXiv* **2014**, arXiv:1404.7592.
3. Proctor, J.L.; Eckhoff, P.A. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *Int. Health* **2015**, *7*, 139–145. [CrossRef] [PubMed]

4. Brunton, B.W.; Johnson, L.A.; Ojemann, J.G.; Kutz, J.N. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *J. Neurosci. Methods* **2016**, *258*, 1–15. [CrossRef] [PubMed]
5. Mann, J.; Kutz, J.N. Dynamic mode decomposition for financial trading strategies. *Quant. Financ.* **2016**, *16*, 1643–1655. [CrossRef]
6. Cui, L.-X.; Long, W. Trading strategy based on dynamic mode decomposition: Tested in Chinese stock market. *Phys. Stat. Mech. Its Appl.* **2016**, *461*, 498–508. [CrossRef]
7. Kuttichira, D.P.; Gopalakrishnan, E.A.; Menon, V.K.; Soman, K.P. Stock price prediction using dynamic mode decomposition. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 55–60. [CrossRef]
8. Berger, E.; Sastuba, M.; Vogt, D.; Jung, B.; Amor, H.B. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *J. Adv. Robot.* **2015**, *29*, 331–343. [CrossRef]
9. Schmid, P.J. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **2010**, *656*, 5–28. [CrossRef]
10. Seena, A.; Sung, H.J. Dynamic mode decomposition of turbulent cavity ows for selfsustained oscillations. *Int. J. Heat Fluid Fl.* **2011**, *32*, 1098–1110. [CrossRef]
11. Schmid, P.J. Application of the dynamic mode decomposition to experimental data. *Exp. Fluids* **2011**, *50*, 1123–1130. [CrossRef]
12. Rowley, C.W.; Mezić, I.; Bagheri, S.; Schlatter, P.; Henningson, D.S. Spectral analysis of nonlinear flows. *J. Fluid Mech.* **2009**, *641*, 115–127. [CrossRef]
13. Mezić, I. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlin. Dynam.* **2005**, *41*, 309–325. [CrossRef]
14. Tu, J.H.; Rowley, C.W.; Luchtenburg, D.M.; Brunton, S.L.; Kutz, J.N. On dynamic mode decomposition: Theory and applications. *J. Comput. Dyn.* **2014**, *1*, 391–421. [CrossRef]
15. Kutz, J.N.; Brunton, S.L.; Brunton, B.W.; Proctor, J. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*; SIAM: Philadelphia, PA, USA, 2016; pp. 1–234. ISBN 978-1-611-97449-2.
16. Bai, Z.; Kaiser, E.; Proctor, J.L.; Kutz, J.N.; Brunton, S.L. Dynamic Mode Decomposition for CompressiveSystem Identification. *AIAA J.* **2020**, *58*, 561–574. [CrossRef]
17. Brunton , S.L.; Budišić, M.; Kaiser, E.; Kutz, J.N. Modern Koopman Theory for Dynamical Systems. *SIAM Rev.* **2020**, *64*, 229–340. [CrossRef]
18. Proctor, J.L.; Brunton, S.L.; Kutz, N. Dynamic mode decomposition with control. *SIAM J. Appl. Dyn. Syst.* **2016**, *15*, 142–161. [CrossRef]
19. Proctor, J.L.; Brunton, S.L.; Kutz, J.N. Generalizing Koopman theory to allow for inputs and control. *SIAM J. Appl. Dyn. Syst.* **2018**, *17*, 909–930. [CrossRef] [PubMed]
20. Brunton, S.L.; Brunton, B.W.; Proctor, J.L.; Kutz, J.N. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLoS ONE* **2016**, *11*, e0150171. [CrossRef]
21. Proctor, J.L.; Brunton, S.L.; Kutz, J.N. Including inputs and control within equation-free architectures for complex systems. *Eur. Phys. J. Spec. Top.* **2016**, *225*, 2413–2434. [CrossRef]
22. Le Clainche, S.; Vega, J.M.; Soria, J. Higher order dynamic mode decomposition of noisy experimental data: The flow structure of a zero-net-mass-flux jet. *Exp. Therm. Fluid Sci.* **2017**, *88*, 336–353. . [CrossRef]
23. Anantharamu, S.; Mahesh, K. A parallel and streaming Dynamic Mode Decomposition algorithm with finite precision error analysis for large data. *J. Comput. Physics*, **2019**, *380*, 355–377.. [CrossRef]
24. Sayadi, T.; Schmid, P.J. Parallel data-driven decomposition algorithm for large-scale datasets: with application to transitional boundary layers. *Theor. Comput. Fluid Dyn.* **2016**, *30*, 415–428. [CrossRef]
25. Maryada, K.R.; Norris, S.E. Reduced-communication parallel dynamic mode decomposition. *J. Comput. Sci.* **2022**, *61*, 101599. [CrossRef]
26. Li, B.; Garicano-Menaab, J.; Valero, E. A dynamic mode decomposition technique for the analysis of non–uniformly sampled flow data. *J. Comput. Phys.* **2022**, *468*, 111495. [CrossRef]
27. Smith, E.; Variansyah, I.; McClarren, R. Variable Dynamic Mode Decomposition for Estimating Time Eigenvalues in Nuclear Systems. *arXiv* **2022**, arXiv:2208.10942v1.
28. Jovanovic, M.R.; Schmid, P.J.; Nichols, J.W. Sparsity-promoting dynamic mode decomposition. *Phys. Fluids* **2014**, *26*, 024103. [CrossRef]
29. Guéniat, F.; Mathelin, L.; Pastur, L.R. A dynamic mode decomposition approach for large and arbitrarily sampled systems. *Phys. Fluids* **2014**, *27*, 025113. [CrossRef]
30. Zhang, H.; Rowley, C.W.; Deem, E.A.; Cattafesta, L.N. Online Dynamic Mode Decomposition for Time-Varying Systems. *SIAM J. Appl. Dyn. Syst.* **2019**, *18*, 1586–1609. [CrossRef]
31. Hemati, M.S.; Williams, M.O.; Rowley, C.W. Dynamic mode decomposition for large and streaming datasets. *Phy. Fluids* **2014**, *26*, 111701. [CrossRef]
32. Matsumoto, D.; Indinger, T. On-the-Fly Algorithm for Dynamic Mode Decomposition Using Incremental Singular Value Decomposition and Total Least Squares. *arXiv* **2017**, arXiv:1703.11004.
33. Nedzhibov, G. Online Dynamic Mode Decomposition: an alternative approach for low rank datasets. *Ann. Acad. Rom. Sci. Ser. Math. Appl.* **2023**, *in press*.

34. Cao, J.; Jiang, Z.; Gao, L.; Liu, Y.; Bao, C. Continuous crack detection using the combination of dynamic mode decomposition and connected component-based filtering method. *Structures* **2023**, *49*, 640–654. [CrossRef]
35. Barbulescu, R.; Mestre, G.; Oliveira, A.L. Learning the dynamics of realistic models of C. elegans nervous system with recurrent neural networks. *Sci. Rep.* **2023**, *13*, 467. [CrossRef] [PubMed]
36. Semlitsch, B.; Cuppoletti, D.R.; Gutmark, E.J.; Mihaescu, M. Transforming the Shock Pattern of Supersonic Jets Using Fluidic Injection. *AIAA J.* **2019**, *57*, 1851–1861. [CrossRef]
37. Liu, T.; Ma, X.; Li, S.; Li, X.; Zhang, C. A stock price prediction method based on meta-learning and variational mode decomposition. *Knowl.-Based Syst.* **2022**, *252*, 109324. [CrossRef]
38. Holmes, P.; Lumley, J.L.; Berkooz, G.; Rowley, C.W. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, 2nd ed.; Cambridge Monographs on Mechanics; Cambridge University Press: Cambridge, UK, 2012.
39. Chen, K.; Brunton, S.L. Data-driven selection of dynamic mode decomposition parameters. *J. Fluid Mech.* **2020**, *892*, A24.
40. Kutz, J.; Fu, X.; Brunton, S. Multiresolution dynamic mode decomposition. *SIAM J. Appl. Dyn. Syst.* **2016**, *15*, 713–735. [CrossRef]
41. Chakraborty, S.; Rowley, C.; Koumoutsakos, P. Dynamic mode decomposition for machine learning in fluids. *J. Fluid Mech.* **2021**, *907*, A15.