Multi-trajectory Dynamic Mode Decomposition

Ryoji Anzaki,[1, *] Shota Yamada,[2, †] Takuro Tsutsui,[3, ‡] and Takahito Matsuzawa[4, §]

[1] *Advanced Engineering 1st Department, Digital Design Center, Tokyo Electron Ltd.,*
Akasaka Biz Tower, 3-1 Akasaka 5-chome, Minato-ku, Tokyo 107-6325, Japan
[2] *Advanced Engineering 1st Department, Digital Design Center,*
Tokyo Electron Ltd., Daido Seimei Sapporo Bldg, 1, Kita 3-jo,
Nishi 3-chome, Chuo-ku, Sapporo city, 060-0003, Japan
[3] *Advanced Engineering 2nd Department, Digital Design Center,*
Tokyo Electron Ltd., Daido Seimei Sapporo Bldg, 1, Kita 3-jo,
Nishi 3-chome, Chuo-ku, Sapporo city, 060-0003, Japan
[4] *Global Sales Division, Tokyo Electron Ltd., Akasaka Biz Tower,*
3-1 Akasaka 5-chome, Minato-ku, Tokyo 107-6325, Japan

(Dated: October 7, 2024)

We propose a new interpretation of the parameter optimization in dynamic mode decomposition (DMD), and show a rigorous application in multi-trajectory modeling. The proposed method, multi-trajectory DMD (MTDMD) is a numerical method using which we can model a dynamical system using multiple *trajectories*, or multiple sets of consecutive snapshots. Here, a trajectory corresponds to an experiment, so the proposed method accepts multiple experimental results. This is in a clear contrast to the existing DMD-based methods that accept only one set of consecutive snapshots. Abovementioned *multi-trajectory modeling* is quite useful in applications, because we can suppress the undesirable effects from the observation noises and employ multiple experiments with different experimental conditions to model a complex system.

The proposed method is quite flexible, and we suggest that we can also implement L2 regularization and element-wise equality constraints on the model parameters. We expect that the proposed method is useful to model large complex systems in the industrial applications.

Keywords: Dynamic Mode Decomposition, Multi-trajectory Modeling, Time-series Data Analysis

I. INTRODUCTION

Dynamic mode decomposition (DMD) [1–3] is a widely-used numerical method to analyze multi-dimensional time-series data. Since the publication of its original paper [1] by Schmid in 2010, researchers developed many variants [3, 4] including DMD with control (DMDc) [5], optimized DMD (OP-DMD) [6], bagging-optimized DMD (BOP-DMD) [7], residual DMD [8], and DMD with memory (DMDm) [9]. The well-known DMDc developed by Proctor *et al.* enabled us to incorporate effects from exogeneous control inputs to the DMD framework (i.e., we can model nonautonomous dynamical systems) [5]. DMDm is another type of variant, in which Anzaki *et al.* replaced the time difference operator implicitly applied to the time-series data in the DMD framework by a wider class of matrix to include memory effects [9]. The BOP-DMD is a DMD-based method equipped with variable projection and statistical bagging methods. The OP-DMD corresponds to the BOP-DMD with no statistical bagging [7].

Along with the developments in the abovementioned variants, researchers also have deepened the understandings of the mathematical aspects of the DMD framework from the view of Koopman theory [4, 10–12].

Throughout this article, we denote $\hom(\mathbf{R}^q, \mathbf{R}^p)$ by $\mathbf{R}^{q\times p}$ for simplicity. We also denote the identity matrix by id.

A. DMD as a Numerical Optimization

We can interpret the exact DMD [2] as a framework to optimize the coefficient $A \in \mathbf{R}^{n\times n}$ of a linear, constant coefficient dynamical system $\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i$ for time-dependent variable $\boldsymbol{x} \in \mathbf{R}^n$ and discrete time variable $i = 0, 1, \ldots$ analytically so that the model prediction $\boldsymbol{x}_{\text{pred}}$ *fits* to the observed data $\boldsymbol{x}_{\text{obs}}$. In the exact DMD [2], we can get the optimized coefficient as follows:

$$A = X'X^+, \tag{1}$$

where $(-)^+$ is the Moore-Penrose pseudoinverse of a matrix where $-$ is a placeholder, and

$$X = \left[\boldsymbol{x}_{\text{obs},0}, \boldsymbol{x}_{\text{obs},1}, \ldots, \boldsymbol{x}_{\text{obs},m-1}\right], \tag{2}$$

$$X' = \left[\boldsymbol{x}_{\text{obs},1}, \boldsymbol{x}_{\text{obs},2}, \ldots, \boldsymbol{x}_{\text{obs},m-2}\right]. \tag{3}$$

In what follows, we use the exact DMD as the example, but a similar discussion also applies for the DMDc. Note that, in practice, we use singular-value decomposition (SVD) to calculate Moore-Penrose pseudoinverse. We can reduce the model dimensions to r by keeping r largest singular values and ignoring the smaller ones.

[*] Corresponding author, ryoji.anzaki@tel.com
[†] shota.yamada@tel.com
[‡] takuro.tsutsui@tel.com
[§] matty.matsuzawa@tel.com

In this context, the most important characteristics of the DMD is that we define the loss function L_{DMD} as the sum of squared errors (SSE) for the observed- and one-step ahead predicted variables. The one-step ahead prediction of the dynamical variables at time $i+1$ is $A\boldsymbol{x}_i$, so we have the loss function as follows:

$$L_{\text{DMD}} = \sum_{i=0}^{m-1} |\boldsymbol{x}_{\text{obs},i+1} - A\boldsymbol{x}_{\text{obs},i}|^2. \tag{4}$$

Compared to the other modeling methods such as the least squares method (LSM), in which we minimize a loss function $L_{\text{SSE}} := \sum_i |\boldsymbol{x}_{\text{pred},i} - \boldsymbol{x}_{\text{obs},i}|^2$ defined as the SSE between the observed- and multi-step ahead predicted dynamical variables $\boldsymbol{x}_{\text{pred},i}$, the DMD is advantageous because the loss function L_{DMD} becomes quadratic in the coefficient A of the dynamical system. This simplification in the loss function in the DMD has two aspects: (a) it enables us to use the analytical optimization method represented by a matrix pseudoinversion and matrix products, instead of numerically costly iterative methods, (b) the simplification makes a long-term prediction difficult at the same time.

To understand the latter aspect, one can imagine the following situations in which the error $\boldsymbol{x}_{\text{pred},i+1} - \boldsymbol{x}_{\text{obs},i+1}$ at time point $i+1$ is (approximately) proportional to time t_{i+1} with small constant of proportionality:

$$\begin{cases} \boldsymbol{x}_{\text{obs},i+1} - A\boldsymbol{x}_{\text{obs},i} = \epsilon & \text{(DMD)} \\ \boldsymbol{x}_{\text{obs},i+1} - \boldsymbol{x}_{\text{pred},i+1} = \epsilon t_{i+1} & \text{(LSM)} \end{cases}, \tag{5}$$

with $\epsilon \in \mathbf{R}^n$. Let us assume that the numerically achieved minimum for L_{DMD} and L_{LSM} is $\delta > 0$. In the DMD, the loss function L_{DMD} becomes a small value proportional to the total time interval $T > 0$ of the observed data: $L_{\text{DMD}} \propto |\epsilon|^2 T$. The minimum $|\epsilon|$ DMD can numerically achieve is $|\epsilon| = \sqrt{\delta/T}$. Thus, the DMD is not sensitive to this type of errors, resulting in a large model prediction errors for $t \gtrsim T^{1/2}\delta^{-1/2}$. In other words, in DMD the estimated dynamical system $\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i$ tend to accumulate prediction errors step by step.

On the other hand, the SSE loss function for $\boldsymbol{x}_{\text{pred}}$ and $\boldsymbol{x}_{\text{obs}}$ is proportional to the *second* power of the total time interval, i.e., $L_{\text{SSE}} \propto |\epsilon|^2 T^2$. Let us assume we have a time-evolution model $\boldsymbol{x}_{i+1} = f(\{\boldsymbol{x}_j\}_{j \le i}, i|\theta)$ with adjustable parameters $\theta \in \Theta$. Using an appropriate numerical method, one can suppress the SSE for the given observation data to δ, provided the family of the model $\{f(-,-|\theta)|\theta \in \Theta\}$ includes the true model for the target dynamics. In those cases, one can show that, LSM prediction has large errors only for $t \gtrsim T\delta^{-1/2}$.

B. Errors in Coefficient Optimization in DMD

To overcome the abovementioned difficulty, one need to improve the accuracy of the estimated coefficients for the DMD. One of the major contributions to the errors in the coefficient estimation is from the observation noise. As we have seen above, in the DMD framework we fit the increments of the predicted dynamical variables to the observed ones, meaning that it is vulnerable to high-frequency parts of noise. In the realm of data analysis, one usually applies preprocesses e.g., decimation and low-pass filters (e.g., moving average) to mitigate the effects from the observation noise [13]. However, a preprocess inevitably worsens the dynamic characteristics of the original data, resulting in phase delay and prolonged deadtime.

Another major contribution to the errors in coefficient estimations is originated from the inappropriate design of experiment (DoE). Suppose we have a large system with the degrees of freedom (DoF) more than around one hundred. To model such system precisely, we need to excite all the modes sufficiently strongly so that the signal to noise ratio (S/N ratio) becomes sufficiently large. However, as the number of modes (roughly proportional to the DoF) becomes large, DoEs to achieve a good S/N ratio often becomes hard to realize, due to the limitations on the system and observation instruments.

We can understand the abovementioned two contributions in a unified manner: suppose we have a set of noiseless snapshots from a linear autonomous dynamical system $\{\boldsymbol{x}_i \in \mathbf{R}^d | i = 0, 1, \ldots, m\}$ and let $X = \begin{bmatrix}\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-1}\end{bmatrix}$ and $X' = \begin{bmatrix}\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\end{bmatrix}$. Let us denote a realization of noise at time point i is $\delta\boldsymbol{x}_i \in \mathbf{R}^d$ for $i = 1, \ldots, m$ and let $\delta X = \begin{bmatrix}\delta\boldsymbol{x}_0, \delta\boldsymbol{x}_1, \ldots, \delta\boldsymbol{x}_{m-1}\end{bmatrix}$ and $\delta X' = \begin{bmatrix}\delta\boldsymbol{x}_1, \delta\boldsymbol{x}_2, \ldots, \delta\boldsymbol{x}_m\end{bmatrix}$. We assume that the noise is sufficiently small in the sense $\epsilon := \|\delta X\|_2 / \|X\|_2 \ll 1$, with $\| - \|_2$ being the operator norm of matrix. The ground truth of the coefficient A_0 is as follows:

$$A_0 = X'X^+. \tag{6}$$

The coefficient estimated using the observed data A is as follows:

$$A = (X' + \delta X')(X + \delta X)^+. \tag{7}$$

Here we use a special version of the generalization of Sherman–Morrison–Woodbury theorem (Theorem 3.2 in [14]): if $\text{range}(\delta X) \subseteq \text{range}(X)$, then

$$(X + \delta X)^+ = X^+ - X^+ S_X^+ X^+, \tag{8}$$

where, $S_X = \text{id} + X^+\delta X$ is invertible because we have assumed $\|X^+\delta X\|_2 < 1$, and we use the facts that (a) the orthogonal projections onto null spaces of S_X vanish, and (b) $\text{range}(\delta X^*) \subseteq \mathbf{R}^d = \text{range}(S_X)$.

Thus,

$$A = A_0 + \delta X' X^+ - X' X^+ \delta X X^+ + \mathcal{O}(\epsilon^2). \tag{9}$$

Let us denote the first-order part of the error by δA as follows:

$$\delta A = \delta X' X^+ - X' X^+ \delta X X^+. \tag{10}$$

The magnification ratio ρ of the relative error is

$$\begin{aligned}
\rho &:= \frac{\|\delta A\|_2 / \|A_0\|_2}{\|\delta X\|_2 / \|X\|_2} && (11)\\
&\leq \frac{(\|\delta X' X^+\|_2 + \|X' X^+ \delta X X^+\|_2)/\|X' X^+\|_2}{\|\delta X\|_{\mathsf{F}} / \|X\|_2} && (12)\\
&\simeq \frac{(\|\delta X X^+\|_2 + \|X' X^+ \delta X X^+\|_2)/\|X' X^+\|_2}{\|\delta X\|_2 / \|X\|_2} && (13)\\
&= \frac{\|\delta X X^+\|_2 / \|\delta X\|_2}{\|X X^+\|_2 / \|X\|_2} + \frac{(\|X' X^+ \delta X X^+\|_2)/\|X' X^+\|_2}{\|\delta X\|_2 / \|X\|_2} && (14)\\
&\leq \frac{\|\delta X X^+\|_2 / \|\delta X\|_2}{\|X X^+\|_2 / \|X\|_2} + \|X^+\|_2 \|X\|_2. && (15)
\end{aligned}$$

Here, we use an approximation $\|\delta X'\|_2 \simeq \|\delta X\|_2$, and in the last line we use the submultiplicativity of operator norm $\|X' X^+ \delta X X^+\|_2 \leq \|X' X^+\|_2 \|\delta X\|_2 \|X^+\|_2$. Thus, the upper bound of the magnification ratio ρ is estimated as follows:

$$\rho \leq 2\|X^+\|_2 \|X\|_2 = 2\kappa(X), \tag{16}$$

where $\kappa(X)$ is the condition number of X. This gives important insights on the errors of coefficient optimizations in DMD: (1) the error is proportional to the magnitude of the noises, (2) the errors in the optimized coefficient is magnified by up to twice of the condition number $2\kappa(X)$. Thus, to suppress the errors in the coefficient optimization, we need to suppress the noise and the condition number $\kappa(X)$ of the observed data X.

The most trivial approach to this aim is to prepare a set of experimental conditions $\{\mu = 1, 2, 3, \dots\}$ and take an average over the set of optimized coefficients $A^{(\mu)}$. However, this approach fails when some experimental conditions do not excite the entire system sufficiently. In those cases, the optimized coefficients suffer from a bad condition number $\kappa(X) \gg 1$ of the observed data X and the contribution from the noises are magnified by the factor of up to $2\kappa(X)$. Thus, to suppress the errors in the optimized coefficients, we need a lot of experiments and must average the resultant coefficients over the huge number of ensembles.

C. Multi-Trajectory Modeling

Another, more elaborate approach to suppress the contributions from noises is the *multi-trajectory modeling* (or, multiple-trajectory modeling). In a multi-trajectory modeling procedure, one uses a set of *multiple* trajectories to get a *single* dynamical system which best fits to the set of given trajectories.

The multi-trajectory modeling is already implemented in PySINDy [15], a python library for sparse identification of nonlinear dynamical systems (SINDy) [16]. SINDy is a successful nonlinear modeling method with sparsity-promoting numerical optimization of coefficients of the nonlinear dynamical systems for library functions specified by the user. It is shown that SINDy works quite well for identification of the underlying dynamics for low-dimensional (up to tens of spatial dimensions) time-series data.

However, for systems with huge number of DoF, SINDy has severe limitations. One of such limitations is the problem of numerical costs, especially when applying constraints on the parameters. This is because the optimization is performed numerically by using iterative methods, in contrast to DMD-based methods, in which we obtain the optimal parameters by applying a small number of matrix operations to the observed- and control input data. For example, for a large linear system with DoF $>$ 100, SINDy becomes prohibitively heavy even with small number of library functions.

D. Aim of This Research

We propose a new approach to overcome the abovementioned difficulty in modeling huge dimensional linear dynamical systems using *a priori* constraints (knowledge) on the coefficient matrices by introducing the idea of multi-trajectory modeling to the DMD framework.

Multi-trajectory modeling methods enable us to model a system using *a set of multiple experimental conditions*, rather than *averaging* the model parameters for each experimental condition. However, there is no known efficient way to perform multi-trajectory modeling using existing DMD methods.

In this paper, we propose a numerical method MT-DMD (multi-trajectory DMD), that is a multi-trajectory modeling method based on DMD. In what follows, we re-interpret the existing DMD method as a numerical optimization of the loss function defined as the square of matrix Frobenius norm of the modeling error, and we re-write the optimization scheme by using the Jacobian (gradient) and Hessian of the loss function. Noting that the derivative operators are linear on the target functions, we easily extend DMD to multi-trajectory framework.

II. THEORY

In this section, we first show an alternative way to interpret the optimization of model parameters in DMD framework, and propose a DMD-based, multi-trajectory modeling method using the alternative interpretation.

The numerical complexities for the existing DMD and the proposed method are shown in the last part of this section.

A. Optimization in DMD

Suppose we have a pair of data (Y, X) where $Y, X \in \mathbf{R}^{d\times m}$ are matrices. We call the row indices of X, Y the *spatial* direction, while the column indices the *temporal* direction. We call (X, Y) a trajectory. In the exact DMD, we use the matrix Y constructed by shifting the matrix X towards the *future* by one time step. In the DMDm, we use a memory kernel to construct Y from X [9]. For a matrix Y that satisfies the causality property [9], one can construct a time-evolution model of the form:

$$\boldsymbol{x}_{i+1} = f(\{\boldsymbol{x}_j\}_{j\le i}, i|\theta), \tag{17}$$

where θ is the adjustable parameters. Hereafter, we focus on the constant coefficient time-evolution model of the form:

$$\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i + B\boldsymbol{u}_i, \tag{18}$$

where $A \in \mathbf{R}^{d\times d}$ and $B \in \mathbf{R}^{d\times d_c}$ are constant matrices, and $\boldsymbol{u} \in \mathbf{R}^{d_c}$ is the exogeneous external input. In the exact DMD we deal with the case $B = 0$.

In the exact DMD, we use the following well-known theorem to optimize the model parameter assuming a linear dependency $Y = AX$ for $A \in \mathbf{R}^{d\times d}$:

$$\arg\min_A \|Y - AX\|_{\mathsf{F}} = YX^+, \tag{19}$$

where $\|\bullet\|_{\mathsf{F}}$ is the Frobenius norm of a matrix.

The above optimization problem can also be seen as the optimization of a multivariate quadratic function. Let us denote

$$L(A|X, Y) = \|Y - AX\|_{\mathsf{F}}^2. \tag{20}$$

The function L is the *loss function* of the linear model $Y = AX$ for given data X, Y and coefficient matrix A. Noting that the second power of nonnegative numbers $\mathbf{R}_{\ge 0} \ni x \mapsto x^2$ is a strictly monotonically increasing function, one can see that the optimal coefficient A_* of $L(A|X, Y)$ also satisfies $A_* = YX^+$.

B. An Alternative Interpretation of DMD

In this section, we re-interpret the DMD by showing a different way to optimize the loss function. Because L is quadratic in elements of A, we can use the following analytical formula for a quadratic function $f : \mathbf{R}^n \ni \boldsymbol{x} \mapsto f(\boldsymbol{x}) \in \mathbf{R}$:

$$\left[\arg\min_{\boldsymbol{\xi}} f(\boldsymbol{\xi})\right]_\alpha = -\sum_\beta \left[\left(\frac{\partial^2 f}{\partial \boldsymbol{\xi}^2}\right)^+\right]^\beta_\alpha \left.\frac{\partial f}{\partial \xi_\beta}\right|_{\boldsymbol{\xi}=0}, \tag{21}$$

where, α and β are indices of vectors and matrices.

Before proceeding to application of the above formula to our loss function L, let us introduce the tensor notation of a vector $\boldsymbol{a} \in \mathbf{R}^d$, a matrix $F \in \mathbf{R}^{d\times m}$, and a matrix $G \in \mathbf{R}^{d\times d}$ as follows: $\boldsymbol{a} = [a^i]$, $F = [F^i{}_a]$, and $G = [G^i{}_j]$. Note that, we use upper and lower suffices to discriminate the contravariant and covariant indices, but we do not adopt Einstein's convention. In what follows, we assume that the metric is the unit tensor δ_{ij} ($\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \ne j$) unless otherwise stated.

We use $i, j, k, \ell, m, n, \ldots$ for the *spatial* indices, and $a, b, c, \ldots$ for *temporal* indices. We use A, B to denote the vectorized quantities for matrices $A \in \mathbf{R}^{d\times d}$ and $B \in \mathbf{R}^{d\times d_c}$, i.e.,

$$\mathsf{A} = \begin{bmatrix} A^0{}_0 \\ A^0{}_1 \\ \vdots \\ A^{d-1}{}_{d-1} \end{bmatrix}, \ \mathsf{B} = \begin{bmatrix} B^0{}_0 \\ B^0{}_1 \\ \vdots \\ B^{d-1}{}_{d_c-1} \end{bmatrix}. \tag{22}$$

We use the similar symbols $\widehat{\mathsf{H}}, \widehat{\mathsf{K}}, \ldots$ to denote the matrix representation of a linear maps from matrices to matrices. For instance, for a tensor $H = [H^{ij}{}_{kl}] \in \hom(\mathbf{R}^{d\times d}, \mathbf{R}^{d\times d})$ with i, j being the contravariant indices and k, l being the covariant indices, the corresponding matrix is expressed as follows:

$$\widehat{\mathsf{H}} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,d-1} \\ h_{10} & h_{11} & \cdots & h_{1,d-1} \\ \vdots & \vdots & & \vdots \\ h_{d-1,0} & h_{d-1,1} & \cdots & h_{d-1,d-1} \end{bmatrix} \in \mathbf{R}^{d^2\times d^2}, \tag{23}$$

where submatrices $h_{\alpha\beta} \in \mathbf{R}^{d\times d}$ is expressed as follows:

$$h_{\alpha\beta} = \begin{bmatrix} H^{\alpha,0}{}_{\beta,0} & H^{\alpha,0}{}_{\beta,1} & \cdots & H^{\alpha,0}{}_{\beta,d-1} \\ H^{\alpha,1}{}_{\beta,0} & H^{\alpha,1}{}_{\beta,1} & \cdots & H^{\alpha,0}{}_{\beta,d-1} \\ \vdots & \vdots & & \vdots \\ H^{\alpha,d-1}{}_{\beta,0} & H^{\alpha,d-1}{}_{\beta,1} & \cdots & H^{\alpha,d-1}{}_{\beta,d-1} \end{bmatrix}. \tag{24}$$

Note that, in the vectorized notation, a matrix becomes a vector, while a tensor that maps matrices to matrices becomes a matrix. Also note that a vectorization corresponds to a *multi-indexing*, in which two or more indices of a tensor are grouped and treated as one index. We used the lexicographical order to construct the multi-indices in the above example.

1. *Optimization via Matrix Product*

Now that we can apply the general formula to our loss function, to get

$$\left[\arg\min_A L(A|X, Y)\right]^i{}_j = -\sum_{k\ell} K^{ik}{}_{j\ell} \left.\frac{\partial L(A|X, Y)}{\partial A^k{}_\ell}\right|_{A=0}, \tag{25}$$

where K is the Moore-Penrose inverse of the Hessian of L,

$$K = \arg\min_{\mathcal{K}} \sum_{i,j,m,n} \left[\mathcal{K}^{ik}{}_{j\ell} \frac{\partial^2 L(A|X,Y)}{\partial A^k{}_\ell \partial A^m{}_n} - \delta^i_j \delta^n_m \right]^2 . \quad (26)$$

The Jacobian and Hessian of the loss function can be derived analytically, as follows: noting that

$$L(A|X,Y) = \sum_{i,a} \left(Y^i{}_a - \sum_j A^i{}_j X^j{}_a \right)^2 , \quad (27)$$

we obtain

$$\frac{\partial L(A|X,Y)}{\partial A^i{}_j} = -2 \sum_a Y^i{}_a X^j{}_a, \quad (28)$$

$$\frac{\partial^2 L(A|X,Y)}{\partial A^i{}_j \partial A^k{}_\ell} = 2 \sum_a X^j{}_a X^\ell{}_a \delta^{ik}. \quad (29)$$

Note that the Hessian is only dependent on two spatial indices (i.e., j, ℓ in Eq. (28)). Using the standard matrix multiplication, the Jacobian and Hessian become,

$$\frac{\partial L(A|X,Y)}{\partial A^i{}_j} = -2 (Y X^\top)^i{}_j, \quad (30)$$

$$\frac{\partial^2 L(A|X,Y)}{\partial A^i{}_j \partial A^k{}_\ell} = 2 (X X^\top)^{j\ell} \delta^{ik}. \quad (31)$$

Thus, the optimal coefficient A_* becomes as follows:

$$A_* = Y X^\top (X X^\top)^+. \quad (32)$$

This is identical to Eq. (19) as expected, because $X^\top (X X^\top)^+ = X^+$.

For the DMDc, we can use a similar discussion for a data tuple (Y, X, Υ) with additional Υ being a $d_c \times m$ matrix corresponding to the control input. In this case, we call (Y, X, Υ) a trajectory. We obtain the following equation:

$$\begin{bmatrix} A_* & B_* \end{bmatrix} = Y \begin{bmatrix} X^\top & \Upsilon^\top \end{bmatrix} \begin{bmatrix} X X^\top & X \Upsilon^\top \\ \Upsilon X^\top & \Upsilon \Upsilon^\top \end{bmatrix}^+ , \quad (33)$$

where, we assume $Y = AX + B\Upsilon$ with $B \in \mathbf{R}^{d \times d_c}$ being a matrix. This is identical to DMDc without dimension reduction for invertible X and Υ.

2. *Optimization via Vector Representation*

In this subsection, we express the formula Eq. (33) shown in the previous subsection in another representation called vector representation.

Let us introduce multi-indices $\alpha, \beta = 0, 1, 2, \ldots, d^2 - 1$ using the lexicographical order $\mathrm{lex}_d : \mathbf{Z}_{\geq 0} \times \mathbf{Z}_{\geq 0} \to \mathbf{Z}_{\geq 0}$

$$\begin{aligned} &\mathrm{lex}_d(0,0) = 0, \\ &\mathrm{lex}_d(0,1) = 1, \\ &\ldots, \\ &\mathrm{lex}_d(0,d-1) = d-1, \\ &\mathrm{lex}_d(1,0) = d, \\ &\ldots, \\ &\mathrm{lex}_d(d,d) = d^2 - 1. \end{aligned} \quad (34)$$

The vectorization of A is introduced as $\mathsf{A} = [\mathsf{A}^\alpha] \in \mathbf{R}^{d^2}$, and the equation for the optimal coefficient A_* becomes a matrix-vector product, as follows

$$\mathsf{A}_* = - \frac{\partial L}{\partial \mathsf{A}} \left(\frac{\partial^2 L}{\partial \mathsf{A}^2} \right)^+ \Bigg|_{\mathsf{A}=0} . \quad (35)$$

In this basis, the Hessian $\widehat{\mathsf{H}} = \frac{\partial^2 L}{\partial \mathsf{A}^2} \in \mathbf{R}^{d^2 \times d^2}$ is a block matrix:

$$\widehat{\mathsf{H}} = \begin{array}{cc} \begin{bmatrix} h & 0 & 0 & \cdots & 0 \\ 0 & h & 0 & \cdots & 0 \\ 0 & 0 & h & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & \cdots & h \end{bmatrix} & \begin{array}{c} i=0 \\ i=1 \\ i=2 \\ \vdots \\ i=d-1 \end{array} \end{array} , \quad (36)$$

where each submatrix $h = [h_{j\ell}] = 2XX^\top$ is placed in a diagonal block (i,k) for $\widehat{\mathsf{H}} = [H_{(i,j),(k,\ell)}]$. We therefore can circumvent a numerically costly inversion operation of $d^2 \times d^2$ matrix, by applying the block-wise inversion, as follows:

$$\widehat{\mathsf{H}}^+ = \begin{array}{cc} \begin{bmatrix} h^+ & 0 & 0 & \cdots & 0 \\ 0 & h^+ & 0 & \cdots & 0 \\ 0 & 0 & h^+ & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & \cdots & h^+ \end{bmatrix} & \begin{array}{c} i=0 \\ i=1 \\ i=2 \\ \vdots \\ i=d-1 \end{array} \end{array} . \quad (37)$$

Note that, the Jacobian $\mathsf{J} = [J_{(i,j)}] = \frac{\partial L}{\partial \mathsf{A}}$ does *not* have a block-wise structure. The vectorized optimal coefficient A_* is obtained as follows:

$$\mathsf{A}_* = -\mathsf{J}\widehat{\mathsf{H}}^+; \quad \widehat{\mathsf{H}} = \frac{\partial^2 L}{\partial \mathsf{A}^2}. \quad (38)$$

This is the vectorized notation for DMD in the alternative interpretation of the numerical optimization.

Note that, for a large spatial dimension d, the abovementioned method is numerically very advantageous compared to a numerical convex optimization of the loss function L. However, at this point, the standard procedure of DMD Eq. (19) is still much better than our procedure in Eq. (38).

We can follow a similar procedure for the DMD with control. The number of model parameters is $d(d + d_c)$, so we construct the vectorization so that we have d subspaces each of which is $(d+d_c)$-dimensional space, as shown below.

$$\left.\begin{bmatrix} 0 \\ \vdots \\ d+d_c-1 \\ \hline \vdots \\ \hline (d-1)(d+d_c) \\ \vdots \\ d(d+d_c)-1 \end{bmatrix}\right\} d \text{ blocks.} \tag{39}$$

Now, let us introduce multi-indices $\alpha = (i,j) = 0,1,2,\dots,d(d+d_c)-1$ using the lexicographical order

$$\begin{aligned} &\mathrm{lex}_{d,d+d_c}(0,0) = 0, \\ &\mathrm{lex}_{d,d+d_c}(0,1) = 1, \\ &\dots, \\ &\mathrm{lex}_{d,d+d_c}(0,d+d_c-1) = d+d_c-1, \\ &\mathrm{lex}_{d,d+d_c}(1,0) = d+d_c, \\ &\dots, \\ &\mathrm{lex}_{d,d+d_c}(d,d+d_c) = d(d+d_c)-1. \end{aligned} \tag{40}$$

We can now derive similar expression for DMDc using the concatenated, vectorized coefficient matrix C instead of A:

$$\mathsf{C} = \left.\begin{bmatrix} {A^0}_0 \\ \vdots \\ {A^0}_{d-1} \\ {B^0}_0 \\ \vdots \\ {B^0}_{d_c-1} \\ \hline \vdots \\ \hline {A^{d-1}}_0 \\ \vdots \\ {A^{d-1}}_{d-1} \\ {B^{d-1}}_0 \\ \vdots \\ {B^{d-1}}_{d_c-1} \end{bmatrix}\right\} d \text{ blocks.} \tag{41}$$

The resultant expression is as follows:

$$\mathsf{C}_* = -\mathsf{J}\widehat{\mathsf{H}}^+; \quad \widehat{\mathsf{H}} = \frac{\partial^2 L}{\partial \mathsf{C}^2}. \tag{42}$$

This is the vectorized notation of DMDc in the alternative interpretation.

C. Multi-trajectory modeling in DMD

We can use our result Eq. (32) and Eq. (33) in the previous section for multi-trajectory DMD, or MTDMD. Hereafter we use DMDc framework, but one can get MT-DMD without control simply by omitting matrix blocks containing control input Υ.

Let us assume that we have N trajectories $(X_\mu, Y_\mu, \Upsilon_\mu)$ indexed by $\mu, \nu, \dots$. In the exact DMD case, $Y = [\boldsymbol{x}_1, \dots \boldsymbol{x}_{m-1}]$ and $X = [\boldsymbol{x}_0, \dots, \boldsymbol{x}_{m-2}]$. We denote $\{(X,Y,\Upsilon)\} = \{(Y_\mu, X_\mu, \Upsilon_\mu) : \mu = 0,1,2,\dots,N-1\}$ for simplicity. Note that the pair of coefficients (A,B) is independent on the trajectory index μ. In MTDMD with control, we define the model parameter C by:

$$C \coloneqq \begin{bmatrix} A \ B \end{bmatrix} \in \mathbf{R}^{d\times(d+d_c)}. \tag{43}$$

Our goal is to find the best fit matrix C for a given set of trajectories. Now let us define the loss function for MTDMD as

$$L(C|\{(X,Y,\Upsilon)\}) = \sum_\mu \|Y_\mu - AX_\mu - B\Upsilon_\mu\|^2_{\mathsf{F}}. \tag{44}$$

1. Optimization via Matrix Product

Noting that the first- and second-order derivative operators are linear operators, we can readily apply the same procedure we used in the previous section, and obtain the following formula for the optimal coefficients C_*:

$$\begin{aligned} &C_* \\ &= \left(\sum_\mu Y_\mu \begin{bmatrix} X_\mu^\top & \Upsilon_\mu^\top \end{bmatrix}\right)\left(\sum_\nu \begin{bmatrix} X_\nu X_\nu^\top & X_\nu \Upsilon_\nu^\top \\ \Upsilon_\nu X_\nu^\top & \Upsilon_\nu \Upsilon_\nu^\top \end{bmatrix}\right)^+. \end{aligned} \tag{45}$$

This is the MTDMD with control. In this expression, the Jacobian J and Hessian H appears as matrices:

$$J = -2\sum_\mu Y_\mu \begin{bmatrix} X_\mu^\top & \Upsilon_\mu^\top \end{bmatrix}, \tag{46}$$

$$H = 2\sum_\nu \begin{bmatrix} X_\nu X_\nu^\top & X_\nu \Upsilon_\nu^\top \\ \Upsilon_\nu X_\nu^\top & \Upsilon_\nu \Upsilon_\nu^\top \end{bmatrix}. \tag{47}$$

Note that, the numerical optimization in Eq. (45) is robust against noises if the condition number of the Hessian $\kappa(H) \geq 1$ is small enough. Our proposed method is advantageous to existing methods, because it is almost as fast as exact DMD, and we can incorporate multi-trajectory modeling.

2. Optimization via Vector Representation

In this subsection, we show a vector representation of MTDMD. For the multi-trajectory case, we get the following expression for the Hessian of the loss function H:

$$\widehat{\mathsf{H}} = \begin{array}{cc} \begin{bmatrix} h & 0 & 0 & \cdots & 0 \\ 0 & h & 0 & \cdots & 0 \\ 0 & 0 & h & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & \cdots & h \end{bmatrix} & \begin{array}{c} i=0 \\ i=1 \\ i=2 \\ \vdots \\ i=d-1 \end{array} \end{array}, \tag{48}$$

where the matrix $h \in \mathbf{R}^{(d+d_c)\times(d+d_c)}$ is expressed as follows:

$$h = \begin{bmatrix} X_\nu X_\nu^\top & X_\nu \Upsilon_\nu^\top \\ \Upsilon_\nu X_\nu^\top & \Upsilon_\nu \Upsilon_\nu^\top \end{bmatrix}. \tag{49}$$

The optimal coefficients are obtained as follows, using the Jacobian J:

$$d \text{ blocks} \left\{ \left[\begin{array}{c} C^{0}{}_{0} \\ \vdots \\ C^{0}{}_{d+d_c-1} \\ \hline \vdots \\ \hline C^{d-1}{}_{0} \\ \vdots \\ C^{d-1}{}_{d+d_c-1} \end{array} \right] \right. = -\mathsf{J}\widehat{\mathsf{H}}^{+}, \tag{50}$$

Note that, the proposed method in this vectorized notation is not advantageous compared to the one in using the matrix product.

D. Application of Constraints

In this section, we impose constraints of the following form to the MTDMD:

$$C^{i}{}_{j} = 0 \text{ if } P^{i}{}_{j} = 0, \tag{51}$$

with $P \in \{0,1\}^{d\times(d+d_c)}$ being given matrices. Note that, the free parameters are now reduced to satisfy the element-wise constraint $C = P \odot C$, with $\odot$ being the element-wise (Hadamard) product of matrices. To implement this constraint, we use the vectorized formula, as follows: for $i = 0, 1, 2, \ldots, d-1$, let us define

$$(h_i)_{j\ell} = \begin{cases} \frac{\partial^2 L(C|X,Y)}{\partial C^{i}{}_{j} \partial C^{i}{}_{\ell}} & P^{i}{}_{j} P^{i}{}_{\ell} = 1 \\ 0 & P^{i}{}_{j} P^{j}{}_{\ell} = 0 \end{cases}. \tag{52}$$

One method to implement this is as follows:

$$(h_i)_{j\ell} = \frac{\partial^2 L(C|X,Y)}{\partial C^{i}{}_{j} \partial C^{i}{}_{\ell}} P^{i}{}_{j} P^{i}{}_{\ell}. \tag{53}$$

Note that the repeated indices do not mean summations (i.e., we do not use Einstein's convention). We also define the i-th block of Jacobian $J_i \in \mathbf{R}^d$ as follows:

$$(J_i)_j = \frac{\partial L(C|X,Y)}{\partial C^{i}{}_{j}}. \tag{54}$$

For the i-th row of the concatenated parameter matrix, we use the following formula:

$$\begin{aligned} & d \text{ blocks} \left\{ \left[\begin{array}{c} C^{0}{}_{0} \\ \vdots \\ C^{0}{}_{d+d_c-1} \\ \hline \vdots \\ \hline C^{d-1}{}_{0} \\ \vdots \\ C^{d-1}{}_{d+d_c-1} \end{array} \right] \right. \\ & = -\begin{bmatrix} J_0^\top & J_1^\top & \cdots & J_{d-1}^\top \end{bmatrix} \begin{bmatrix} h_0^+ & & & 0 \\ & h_1^+ & & \\ & & \ddots & \\ 0 & & & h_{d-1}^+ \end{bmatrix}. \end{aligned} \tag{55}$$

This is a numerically advantageous method compared to the brute-force method in which one compute $-\widehat{\mathsf{H}}^{+}\mathsf{J}$ with the pseudoinverse being performed for entire $\widehat{\mathsf{H}}$. Because the matrix inversion of n-dimensional matrix requires $\mathcal{O}(n^3)$ steps, we can reduce the computational cost by the factor $1/d^3 \times d = 1/d^2$ by applying our method, compared to the brute-force numerical optimization of multivariate quadratic function.

Note that, we can also impose wider class of equality constraints to the parameters, such as

$$A^{i}{}_{j} = \alpha \text{ if } (P_\alpha)^{i}{}_{j} = 0 \quad \alpha \in \{\alpha_s \in \mathbf{R} | s = 0, 1, \ldots, S-1\}, \tag{56}$$

for $P_\alpha \in \{0,1\}^{d\times d}$. In that case, we have to modify the Jacobian, because the gradient with respect to the *free* part of the parameter $P \odot C$ at the origin is affected by the *fixed* parameters.

E. Fine-tuning of Models using Additional Set of Trajectories

In the industrial applications, the target systems often have *variations*, i.e., small deviations in model parameters. Such variations are often caused by e.g., machine difference, aging, and change of the external environment of the system. In some cases, the identical system exhibit different characteristics for different region of the dynamical variable $\boldsymbol{x}$, the control inputs $\boldsymbol{u}$, and initial state $\boldsymbol{x}(0)$. For example, a thermostat system exhibits different characteristics for low-temperature region $\boldsymbol{x} < \boldsymbol{x}_0$, middle-temperature region $\boldsymbol{x}_0 < \boldsymbol{x} < \boldsymbol{x}_1$ and high-temperature region $\boldsymbol{x}_1 < \boldsymbol{x}$, where the inequalities are applied element-wise.

To cope with the variations in the target systems, we can *fine-tune* the coefficient matrices obtained by MTDMD using additional set of trajectories after the original modeling: we model the *reference* coefficient matrices A, B using the reference train data, and then fine-tune the model using additional train trajectories

$(X_\mu^{(p)}, Y_\mu^{(p)}, U_\mu^{(p)})$ for $\mu = 0, 1, 2, ..., N_p - 1$ from variation $p = 0, 1, 2, ...$ of the target system, as follows: first, from the state variable $Y_\mu^{(p)}$ we subtract the model prediction by the reference model:

$$\delta Y_\mu^{(p)} = Y_\mu^{(p)} - \left(AX_\mu^{(p)} + BU_\mu^{(p)} \right). \tag{57}$$

Then we model the *increments* of the model coefficient matrices $\delta A^{(p)} \in \mathbf{R}^{d\times d}, \delta B^{(p)} \in \mathbf{R}^{d\times d_c}$ using MTDMD assuming

$$\delta Y^{(p)} = \delta A^{(p)} X + \delta B^{(p)} U \tag{58}$$

for $(X, Y, U) = (X_\mu^{(p)}, Y_\mu^{(p)}, U_\mu^{(p)})$ with $\mu = 0, 1, 2, ..., N_p - 1$.

We can also use MTDMD with reference for partial modeling a nonlinear system; one can train a reference model using all the experimental data and fine-tune the model using a subset of train data corresponding to certain region of state variables.

F. Numerical Complexity of Proposed Methods

In this section, we show the asymptotic order of the numerical costs for the proposed method. Because the SVD in the Moore-Penrose pseudoinverse is the heaviest operation in MTDMD, the asymptotic order of MTDMD equals to that of SVD of the Hessian matrix. Note that the asymptotic order of the SVD of a matrix $M \in \mathbf{R}^{n\times m}$ is $n \times m^2$.

In table I below, we compare the proposed method against the existing method (exact DMD and DMDc). The dimension n is $n = d$ for exact DMD and $n = d + d_c$ for DMDc.

TABLE I. Asymptotic order of the numerical costs: N is the number of trajectories, $n = d$ for exact DMD and $n = d + d_c$ for DMDc. In this table, N/A means not applicable.

	DMD	MTDMD	
		matrix	vector
single-trajectory	$\mathcal{O}(nm^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3 d)$
multi-trajectory	N/A	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3 d)$
with constraint[a]	N/A	N/A	$\mathcal{O}(n^3 d)$

[a] Constraints of the form shown in Eq. (56)

One remarkable feature is that the asymptotic order of MTDMD is not dependent on the number of trajectories N.

III. CONCLUDING REMARKS

We proposed a numerical method MTDMD, or multi-trajectory DMD. We can use MTDMD to model a linear dynamical system using a set of multiple trajectories. Here, a *trajectory* is a set of consecutive snapshots corresponding to observed data. The proposed method emploies Moore-Penrose pseudoinverse of a matrix to model the dynamical system. Although this procedure resembles DMD, the proposed method has remarkable feature that using it we can model a dynamical system even if each trajectory has insufficient information or is too noisy to model the entire system. The proposed method is expected to be useful to model linear dynamical systems with noisy observations using multiple trajectories. Because the multi-trajectory modeling framework enables us to simplify the design of experiments for large, complex systems, MTDMD is expected to serve as an efficient method to identify the dynamical systems.

As a future work, we would like to add rigorous numerical experiments. Another issue to be addressed in the future work is inclusion of generic constraints such as inequality constraint.

Appendix A: Implementation of L2 Regularization

In some cases, we want to suppress the values of elements in the parameter matrices A, B. We can implement a L2 regularization by using the following replacement of Hessian: In the matrix notation,

$$H \to H + 2\lambda \mathrm{id}_{d+d_c}, \tag{A1}$$

while in the vectorized notation,

$$\widehat{\mathsf{H}} \to \widehat{\mathsf{H}} + 2\lambda \begin{bmatrix} \mathrm{id}_{d+d_c} & & 0 \\ & \ddots & \\ 0 & & \mathrm{id}_{d+d_c} \end{bmatrix}, \tag{A2}$$

where id_q is the identity matrix of order q, and $\lambda > 0$ is the coefficient of the regularization.

Appendix B: Dimension Reduction in MTDMD

We use singular-value decomposition (SVD) to reduce the dimensions in DMD. In MTDMD, we can also perform the dimension reduction using SVD. For MTDMD without control, we use the SVD

$$U\Sigma V^* = \sum_\mu X_\mu X_\mu^\top, \tag{B1}$$

where $U, \Sigma, V \in \mathbf{R}^{d\times d}$, and U, V are unitary matrices. By keeping the r largest singular values, we introduce the reduced SVD as follows:

$$U\Sigma V^* \to U_r \Sigma_r V_r^*, \tag{B2}$$

where $U_r, V_r \in \mathbf{R}^{d\times r}$ and $\Sigma_r \in \mathbf{R}^{r\times r}$.

For MTDMD with control, we must use more elaborate method to ensure that the any block in the Hessian is not totally suppressed. To understand this, one may think of a case in which the Hessian is so suppressed that only upper (lower) diagonal block survives. In that case, the reduced model cannot incorporate the effects from the input (state). To circumvent this, we can use the formula for inverse matrix of a 2×2 block matrix: for $A, B, C, D \in \mathbf{A}^{n\times n}$, provided A and $S = D - CA^{-1}B$ are invertible,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{bmatrix}. \tag{B3}$$

We can apply a similar expression for pseudoinverse

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{+} \simeq \begin{bmatrix} A^{+} + A^{+}BS^{+}CA^{+} & -A^{+}BS^{+} \\ -S^{+}CA^{+} & S^{+} \end{bmatrix}. \tag{B4}$$

with matrix pseudoinversion operation for A, B, C, D and S are calculated using SVDs of A, B, C, D and S by keeping e.g., $r, \max(r, p), \max(r, p), p$ and p largest singular values, respectively.

[1] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of fluid mechanics **656**, 5 (2010).

[2] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, On dynamic mode decomposition: Theory and applications, Journal of Computational Dynamics **1**, 391 (2014).

[3] P. J. Schmid, Dynamic mode decomposition and its variants, Annual Review of Fluid Mechanics **54**, 225 (2022).

[4] I. Mezic, Analysis of fluid flows via spectral properties of the koopman operator, Annual Review of Fluid Mechanics **45**, 357 (2013).

[5] J. L. Proctor, S. L. Brunton, and J. N. Kutz, Dynamic mode decomposition with control, SIAM Journal on Applied Dynamical Systems **15**, 142 (2016).

[6] T. Askham and J. N. Kutz, Variable projection methods for an optimized dynamic mode decomposition, SIAM Journal on Applied Dynamical Systems **17**, 380 (2018).

[7] D. Sashidhar and J. N. Kutz, Bagging, optimized dynamic mode decomposition for robust, stable forecasting with spatial and temporal uncertainty quantification, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **380**, 20210199 (2022).

[8] E. Rodrigues, B. Zadrozny, C. Watson, and D. Gold, Decadal forecasts with resdmd: a residual dmd neural network, arXiv preprint arXiv:2106.11111 (2021).

[9] R. Anzaki, K. Sano, T. Tsutsui, M. Kazui, and T. Matsuzawa, Dynamic mode decomposition with memory, Physical Review E **108**, 034216 (2023).

[10] I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dynamics **41**, 309 (2005).

[11] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, Spectral analysis of nonlinear flows, Journal of fluid mechanics **641**, 115 (2009).

[12] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, Modern koopman theory for dynamical systems, arXiv preprint arXiv:2102.12086 (2021).

[13] A. Sano and H. Tsuji, Optimal sampling rate for system identification based on decimation and interpolation, IFAC Proceedings Volumes **26**, 297 (1993).

[14] X. Xu, Generalization of the sherman–morrison–woodbury formula involving the schur complement, Applied Mathematics and Computation **309**, 183 (2017).

[15] A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, Pysindy: A comprehensive python package for robust sparse system identification, Journal

of Open Source Software **7**, 3994 (2022).

[16] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences **113**, 3932 (2016).