Multi-trajectory Dynamic Mode Decomposition

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

[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*

(Dated: January 22, 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 with which we can model a dynamical system using multiple *trajectories*, or sets of consecutive snapshots. Here, a trajectory corresponds to an experiment, so the proposed method accepts multiple experimental results. This is in contrast to any of the existing DMD-based methods, that accepts only one set of consecutive snapshots. The incorporation of the multi-trajectory modeling in DMD is beneficial because we can suppress the undesirable effects from the observation noises. Also, we can perform multiple experiments with different experimental conditions to model a complex system with simple experimental set-ups.

We test the proposed method against the numerically generated synthetic data, and show that the MTDMD achieves not only good reconstruction results but also numerically fast multi-trajectory modeling for multi-dimensional time-series data.

The proposed method is quite flexible, and we suggest that we can also implement L2 regularization and equality constraints on the model parameters. The proposed method is expected to serve as an effective method to model large systems using noisy time-series data without elaborate design of experiment.

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]. Among them, the celebrated DMDc developed by Proctor *et al.* enabled us to incorporate 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 deepen the understandings of the mathematical aspects of the DMD framework from the view of Koopman theory [4, 10–12]. In this point of view, we regard the DMD as a numerical method to find an approximation for Koopman mode decomposition.

Throughout this article, we denote $\hom(\mathbf{R}^q, \mathbf{R}^p)$ by $\mathbf{R}^{q\times p}$ for simplicity.

A. DMD as a Numerical Optimization

From the numerical point of view, the DMD can be regarded 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,

$$X = \left[\boldsymbol{x}_{\text{obs},0}, \boldsymbol{x}_{\text{obs},1}, \ldots, \boldsymbol{x}_{\text{obs},m-1}\right], \tag{2}$$
$$X' = \left[\boldsymbol{x}_{\text{obs},1}, \boldsymbol{x}_{\text{obs},2}, \ldots, \boldsymbol{x}_{\text{obs},m-2}\right]. \tag{3}$$

---

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

In what follows, we use the exact DMD as the example, but a similar discussion also apply for the DMDc.

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 predicted *increments*. An increment of the dynamical variables at time i is $\boldsymbol{x}_{i+1} - \boldsymbol{x}_i$, so we have the loss function as follows:

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

Here, note that $A\boldsymbol{x}_{\mathrm{obs},i}$ is the predicted increment at time point i for given observations $\boldsymbol{x}_{\mathrm{obs},i}, \boldsymbol{x}_{\mathrm{obs},i+1}$.

Compared to the other modeling methods such as the least squares method (LSM), which minimizes a loss function defined as the SSE of the dynamical variables $L_{\mathrm{SSE}} := \sum_i |\boldsymbol{x}_{\mathrm{pred},i} - \boldsymbol{x}_{\mathrm{obs},i}|^2$, 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) On the other hand, the simplification makes a long-term prediction difficult.

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

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

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: $L_{\mathrm{DMD}} \propto |\epsilon|^2 T$. 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 $t \gtrsim T^{1/2}\delta^{-1/2}$.

On the other hand, the SSE loss function for $\boldsymbol{x}_{\mathrm{pred}}$ and $\boldsymbol{x}_{\mathrm{obs}}$ is proportional to the *second* power of the total time interval, i.e., $L_{\mathrm{SSE}} \propto |\epsilon|^2 T^2$. 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 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 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 part of noise. In the realm of data analysis, one usually apply 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 dead-time.

Another major contribution to the errors in coefficient estimations is originated to the inappropriate design of experiment (DoE). Suppose we have a large system with the degrees of freedom (DoF) exceeding 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 $\{\boldsymbol{x}_i | i = 0, 1, \ldots, m-1\}$ and let $X = \left[\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m-1}\right]$ and $X' = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\right]$. Let us denote a realization of noise at time point i is $\delta\boldsymbol{x}_i$ and let $\delta X = \left[\delta\boldsymbol{x}_0, \delta\boldsymbol{x}_1, \ldots, \delta\boldsymbol{x}_{m-1}\right]$ and $\delta X' = \left[\delta\boldsymbol{x}_1, \delta\boldsymbol{x}_2, \ldots, \delta\boldsymbol{x}_m\right]$. We assume that the noise is sufficiently small in the sense $\|X^+\delta X\|_\infty < 1$ and, at the same time, $\epsilon := \|\delta X\|_{\mathsf{F}}/\|X\|_{\mathsf{F}} \ll 1$. The ground truth of the coefficient A_0 is as follows:

$$A_0 = X'X^+. \quad (6)$$

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

$$A = (X' + \delta X')(X + \delta X)^+. \quad (7)$$

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

$$(X + \delta X)^+ = (\mathrm{id} + X^+\delta X F_{S_X} \delta X^* (X^+)^*)^{-1}(A^+ - A^+ S_X^+ X^+)(\mathrm{id} + (X^+)^* E_{S_X} X^+)^{-1}, \quad (8)$$

where $S_X = \mathrm{id} + X^+\delta X$ and $E_M = \mathrm{id} - MM^+$ and $F_M = \mathrm{id} - M^+M$ for a matrix M are orthogonal projections onto null spaces of matrix M^* and M, respectively.

Because $\|X^+ \delta X\|_\infty < 1$, the S_X is invertible, and hence we obtain a simple result:

$$(X + \delta X)^+ = X^+ - X^+ \delta X X^+. \tag{9}$$

Thus,

$$A = A_0 + \delta X' X^+ - X' X^+ \delta X X^+ + \mathcal{O}(\epsilon^2). \tag{10}$$

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 the condition number of 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 often fails, especially when the majority of the 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 $\kappa(X)$. Thus, to suppress the errors in the optimized coefficients, we might need huge number of experiments and average the resultant coefficients.

C. Multi-Trajectory Modeling

Another, more elaborate approach to suppress the contributions from noises is the *multi-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 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 nonlinear modeling method with numerical sparse optimization of coefficients of the corresponding nonlinear dynamical systems. It is shown that SINDy works quite well for identification of the underlying dynamics for low-dimensional time-series data.

However, for systems with huge number of degrees of freedom, 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 the optimal parameters are obtained by applying a simple *function* to observed- and 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 above-mentioned difficulty 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 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.

We show that MTDMD is useful in suppressing the effects from noises in the observation data. The method is also shown to be useful to model huge systems, because one can use a set of multiple experimental conditions each of which excites merely small portion of the whole system, meaning that each of the experimental set-ups can be quite simple. This also means that we can design experiments to improve S/N ratio much easier than the case with existing methods, especially when the experimental set-ups has limitations (e.g., total input power, the number of simultaneous nonzero inputs).

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 each model (i.e., existing DMD and proposed methods) are shown in the end 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. 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:

$$\boldsymbol{x}_{i+1} = f(\{\boldsymbol{x}_j\}_{j\le i}, i|\theta), \tag{11}$$

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

$$\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i + B\boldsymbol{u}_i, \tag{12}$$

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. The exact DMD is 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 $A \in \mathbf{R}^{d\times d}$:

$$\arg\min_A \|Y - AX\|_{\mathsf{F}} = YX^{+}, \tag{13}$$

where $\bullet^{+}$ is the Moore-Penrose pseudo-inverse of a matrix, and $\|\bullet\|_{\mathsf{F}}$ 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

$$L(A|X,Y) = \|Y - AX\|_{\mathsf{F}}^2. \tag{14}$$

The function L is the *loss function* of the linear model $Y = AX$ for given data X, Y. Noting that the second power $\mathbf{R}_{\ge 0} \ni x \mapsto x^2$ 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 $f : \mathbf{R}^n \ni \boldsymbol{x} \mapsto f(\boldsymbol{x}) \in \mathbf{R}$:

$$\left[\arg\min_{\boldsymbol{x}} f(\boldsymbol{x})\right]_\alpha = -\sum_\beta \left[\left(\frac{\partial^2 f}{\partial \boldsymbol{x}^2}\right)^{+}\right]^{\beta}_{\alpha} \left.\frac{\partial f}{\partial x_\beta}\right|_{\boldsymbol{x}=0}, \tag{15}$$

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.,

$$\mathsf{A} = \begin{bmatrix} A^0{}_0 \\ A^0{}_1 \\ \vdots \\ A^{d-1}{}_{d-1} \end{bmatrix}, \ \mathsf{B} = \begin{bmatrix} B^0{}_0 \\ B^0{}_1 \\ \vdots \\ B^{d-1}{}_{d_c-1} \end{bmatrix}. \tag{16}$$

We use the similar symbols $\widehat{\mathsf{H}}, \widehat{\mathsf{K}}, \ldots$ to denote the 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:

$$\widehat{\mathsf{H}} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,d-1} \\ h_{10} & h_{11} & \cdots & h_{1,d-1} \\ \vdots & \vdots & & \vdots \\ h_{d-1,0} & h_{d-1,1} & \cdots & h_{d-1,d-1} \end{bmatrix} \in \mathbf{R}^{d^2\times d^2}, \tag{17}$$

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

$$h_{\alpha\beta} = \begin{bmatrix} H^{\alpha,0}{}_{\beta,0} & H^{\alpha,0}{}_{\beta,1} & \cdots & H^{\alpha,0}{}_{\beta,d-1} \\ H^{\alpha,1}{}_{\beta,0} & H^{\alpha,1}{}_{\beta,1} & \cdots & H^{\alpha,0}{}_{\beta,d-1} \\ \vdots & \vdots & & \vdots \\ H^{\alpha,d-1}{}_{\beta,0} & H^{\alpha,d-1}{}_{\beta,1} & \cdots & H^{\alpha,d-1}{}_{\beta,d-1} \end{bmatrix}. \tag{18}$$

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

$$\left[\arg\min_A L(A|X,Y)\right]^i{}_j = -\sum_{k\ell} K^{ik}{}_{j\ell} \left.\frac{\partial L(A|X,Y)}{\partial A^k{}_\ell}\right|_{A=0}, \tag{19}$$

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

$$K = \arg\min_K \sum_{i,j,m,n} \left[K^{ik}{}_{j\ell} \frac{\partial^2 L(A|X,Y)}{\partial A^k{}_\ell \partial A^m{}_n} - \delta^i_j \delta^n_m\right]^2. \tag{20}$$

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

$$L(A|X,Y) = \sum_{i,a} \left(Y^i{}_a - \sum_j A^i{}_j X^j{}_a\right)^2, \tag{21}$$

we obtain

$$\frac{\partial L(A|X,Y)}{\partial A^i{}_j} = -2\sum_a Y^i{}_a X^j{}_a, \quad \frac{\partial^2 L(A|X,Y)}{\partial A^i{}_j \partial A^k{}_\ell} = 2\sum_a X^j{}_a X^\ell{}_a \delta^{ik}. \tag{22}$$

Note that the Hessian is only dependent on two spatial indices (i.e., j, ℓ in Eq. (22)). Using the standard matrix multiplication, the Jacobian and Hessian become,

$$\frac{\partial L(A|X,Y)}{\partial A^i{}_j} = -2(YX^\top)^i{}_j, \quad \frac{\partial^2 L(A|X,Y)}{\partial A^i{}_j \partial A^k{}_\ell} = 2(XX^\top)^{j\ell}\delta^{ik}. \tag{23}$$

Thus, the optimal coefficient A_* becomes as follows:

$$A_* = YX^\top(XX^\top)^+. \tag{24}$$

This is identical to Eq. (13) for left-invertible X (i.e., $X^\top X = \mathrm{id}_d$), because in that case $X^\top(XX^\top)^+ = X^+$, as expected.

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, and obtain the following equation:

$$\begin{bmatrix} A_* & B_* \end{bmatrix} = Y \begin{bmatrix} X^\top & \Upsilon^\top \end{bmatrix} \begin{bmatrix} XX^\top & X\Upsilon^\top \\ \Upsilon X^\top & \Upsilon\Upsilon^\top \end{bmatrix}^+, \tag{25}$$

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

2. *Optimization via Vector Representation*

In this subsection, we express the formula Eq. (25) shown in the previous subsection in a vectorized notation.

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

$$\begin{aligned} &\mathrm{lex}_d(0,0) = 0, \\ &\mathrm{lex}_d(0,1) = 1, \\ &\ldots, \\ &\mathrm{lex}_d(0,d-1) = d-1, \\ &\mathrm{lex}_d(1,0) = d, \\ &\ldots, \\ &\mathrm{lex}_d(d,d) = d^2 - 1. \end{aligned} \tag{26}$$

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

$$\mathsf{A}_* = -\frac{\partial^2 L}{\partial \mathsf{A}^2}^+ \left.\frac{\partial L}{\partial \mathsf{A}}\right|_{\mathsf{A}=0}. \tag{27}$$

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:

$$\widehat{\mathsf{H}} = \begin{array}{c} \begin{bmatrix} h & 0 & 0 & \cdots & 0 \\ 0 & h & 0 & \cdots & 0 \\ 0 & 0 & h & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & \cdots & h \end{bmatrix} \end{array} \begin{array}{l} i=0 \\ i=1 \\ i=2 \\ \vdots \\ i=d-1 \end{array}, \tag{28}$$

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:

$$\widehat{\mathsf{H}}^+ = \begin{bmatrix} h^+ & 0 & 0 & \cdots & 0 \\ 0 & h^+ & 0 & \cdots & 0 \\ 0 & 0 & h^+ & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & \cdots & h^+ \end{bmatrix} \begin{array}{l} i=0 \\ i=1 \\ i=2 \\ \vdots \\ i=d-1 \end{array}. \tag{29}$$

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:

$$\mathsf{A}_* = -\widehat{\mathsf{H}}^+\mathsf{J}; \quad \widehat{\mathsf{H}} = \frac{\partial^2 L}{\partial \mathsf{A}^2}. \tag{30}$$

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

Note that, for a large spatial dimension d, the above-mentioned 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. (13) is still much than our procedure in Eq. (30).

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.

$$\left.\begin{bmatrix} 0 \\ \vdots \\ d+d_c-1 \\ \hline \vdots \\ \hline (d-1)(d+d_c) \\ \vdots \\ d(d+d_c)-1 \end{bmatrix}\right\} d \text{ blocks} \tag{31}$$

Now, let us introduce multi-indices $\alpha = (i,j) =$

$0, 1, 2, \ldots, d(d + d_c) - 1$ using the lexicographical order

$$\begin{aligned}
&\mathrm{lex}_{d+d_c}(0, 0) = 0,\\
&\mathrm{lex}_{d+d_c}(0, 1) = 1,\\
&\ldots,\\
&\mathrm{lex}_{d+d_c}(0, d + d_c - 1) = d + d_c - 1,\\
&\mathrm{lex}_{d+d_c}(1, 0) = d + d_c,\\
&\ldots,\\
&\mathrm{lex}_{d+d_c}(d, d + d_c) = d(d + d_c) - 1.
\end{aligned} \tag{32}$$

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

$$\mathsf{C} = \left.\begin{bmatrix} {A^0}_0 \\ \vdots \\ {A^0}_d \\ {B^0}_0 \\ \vdots \\ {B^0}_{d_c} \\ \hline \vdots \\ \hline {A^d}_0 \\ \vdots \\ {A^d}_d \\ {B^d}_0 \\ \vdots \\ {B^d}_{d_c} \end{bmatrix}\right\} d \text{ blocks} \tag{33}$$

The resultant expression is as follows:

$$\mathsf{C}_* = -\widehat{\mathsf{H}}^+ \mathsf{J}; \quad \widehat{\mathsf{H}} = \frac{\partial^2 L}{\partial \mathsf{C}^2}. \tag{34}$$

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

C. Multi-trajectory modeling in DMD

We can use our result Eq. (24) and Eq. (25) in the previous section for multi-trajectory DMD, or MTDMD. Hereafter we use DMDc framework, but one can get MTDMD without control simply by omitting matrix blocks containing control input Υ.

Let us assume that we have N tuples of data $X_\mu, Y_\mu, \Upsilon_\mu$ indexed by $\mu, \nu, \ldots$ Hereafter, we assume that Y, X, and Υ represents the observation, state, and external input. In the exact DMD case, $Y = [\boldsymbol{x}_1, \ldots \boldsymbol{x}_{m-1}]$ and $X = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{m-2}]$. We denote $\{X, Y, \Upsilon\} = \{(Y_\mu, X_\mu, \Upsilon_\mu) : \mu = 0, 1, 2, \ldots, N - 1\}$ for simplicity. We call each tuple $(Y_\mu, X_\mu, \Upsilon_\mu)$ a *trajectory*. Now let us define the loss function for MTDMD as

$$L(A, B|\{X, Y, \Upsilon\}) = \sum_\mu \|Y_\mu - AX_\mu - B\Upsilon_\mu\|_{\mathsf{F}}^2. \tag{35}$$

Note that the pair of coefficients (A, B) is a constant of trajectory index μ. In MTDMD with control, we think that the model parameters form a $d \times (d + d_c)$ matrix as follows:

$$\begin{bmatrix} A & B \end{bmatrix} \in \mathbf{R}^{d\times(d+d_c)}. \tag{36}$$

1. Optimization via Matrix Product

Noting that the first- and second-order derivatives are linear with respect to the target function, we can readily apply the same procedure we used in the previous section, and obtain the following formula for the optimal coefficients A_*, B_*:

$$\begin{bmatrix} A_* & B_* \end{bmatrix} = \left(\sum_\mu Y_\mu \begin{bmatrix} X_\mu^\top & \Upsilon_\mu^\top \end{bmatrix}\right)\left(\sum_\nu \begin{bmatrix} X_\nu X_\nu^\top & X_\nu \Upsilon_\nu^\top \\ \Upsilon_\nu X_\nu^\top & \Upsilon_\nu \Upsilon_\nu^\top \end{bmatrix}\right)^+. \tag{37}$$

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

$$J = -\sum_\mu Y_\mu \begin{bmatrix} X_\mu^\top & \Upsilon_\mu^\top \end{bmatrix}, \quad H = \sum_\nu \begin{bmatrix} X_\nu X_\nu^\top & X_\nu \Upsilon_\nu^\top \\ \Upsilon_\nu X_\nu^\top & \Upsilon_\nu \Upsilon_\nu^\top \end{bmatrix}. \tag{38}$$

Note that, the numerical optimization in Eq. (37) 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

Now that we get the following expression for the Hessian H:

$$\widehat{\mathsf{H}} = \begin{array}{c} \begin{bmatrix} h & 0 & 0 & \cdots & 0 \\ 0 & h & 0 & \cdots & 0 \\ 0 & 0 & h & & \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & \cdots & h \end{bmatrix} \end{array} \begin{array}{c} i = 0 \\ i = 1 \\ i = 2 \\ \vdots \\ i = d - 1 \end{array}, \tag{39}$$

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

$$h = \begin{bmatrix} X_\nu X_\nu^\top & X_\nu \Upsilon_\nu^\top \\ \Upsilon_\nu X_\nu^\top & \Upsilon_\nu \Upsilon_\nu^\top \end{bmatrix} \tag{40}$$

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

$$d \text{ blocks}\left\{\begin{bmatrix} {A^0}_0 \\ \vdots \\ {A^0}_d \\ {B^0}_0 \\ \vdots \\ {B^0}_{d_c} \\ \hline \vdots \\ \hline {A^d}_0 \\ \vdots \\ {A^d}_d \\ {B^d}_0 \\ \vdots \\ {B^d}_{d_c} \end{bmatrix}\right. = -\widehat{\mathsf{H}}^+\mathsf{J}. \tag{41}$$

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:

$$A_{ij} = 0 \text{ if } P^A_{ij} = 0, \quad B_{ij} = 0 \text{ if } P^B_{ij} = 0, \tag{42}$$

with $P^A \in \{0,1\}^{d\times d}$ and $P^B \in \{0,1\}^{d\times d_c}$ are matrices. Note that, the free parameters are now reduced to $P^A \odot A$, with $\odot$ being the element-wise (Hadamard) product of matrices. Let us define a matrix P as follows:

$$P = \begin{bmatrix} P^A & P^B \end{bmatrix}. \tag{43}$$

To implement this constraint, we use the vectorized formula, as follows: for $i = 0, 1, 2, \ldots, d-1$, let us define

$$(h_i)^{j\ell} = \begin{cases} \frac{\partial^2 L(A|X,Y)}{\partial {A^i}_j \partial {A^i}_\ell} & P_{ij}P_{i\ell} = 1 \\ 0 & P_{ij}P_{j\ell} = 0 \end{cases}. \tag{44}$$

One method to implement this is as follows:

$$(h_i)^{j\ell} = \frac{\partial^2 L(A|X,Y)}{\partial {A^i}_j \partial {A^i}_\ell} {P^i}_j {P^i}_\ell. \tag{45}$$

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 as follows:

$$(J_i)_j = \frac{\partial L(A|X,Y)}{\partial A^{ij}}. \tag{46}$$

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

$$d \text{ blocks}\left\{\begin{bmatrix} {A^0}_0 \\ \vdots \\ {A^0}_d \\ {B^0}_0 \\ \vdots \\ {B^0}_{d_c} \\ \hline \vdots \\ \hline {A^d}_0 \\ \vdots \\ {A^d}_d \\ {B^d}_0 \\ \vdots \\ {B^d}_{d_c} \end{bmatrix}\right. = -\begin{bmatrix} h_0^+ & & & 0 \\ & h_1^+ & & \\ & & \ddots & \\ 0 & & & h_{d-1}^+ \end{bmatrix}\begin{bmatrix} J_0 \\ J_1 \\ \vdots \\ J_{d-1} \end{bmatrix}. \tag{47}$$

This is a numerically advantageous method compared to the brute-force method in which one compute $-\widehat{\mathsf{H}}^+\mathsf{J}$ with the pseudo-inverse 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 constraint to the parameters, such as

$$A_{ij} = \alpha \text{ if } P^A_{\alpha,ij} = 0 \quad \alpha \in \{\alpha_s \in \mathbf{R} | s = 0, 1, 2, \ldots, S\}, \tag{48}$$

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

E. 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 most heavy 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 $M \in \mathbf{n} \times \mathbf{m}$ 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.

One remarkable feature is that the asymptotic order of MTDMD is not dependent on the number of trajectories N. Also, as mentioned above, in the case with constraints on the parameter matrices, our method achieves better reconstructions compared to the trivial approach (the asymptotic order is $\mathcal{O}(d^6)$) in which the Jacobian

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.

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

[a] Applying the existing DMD method for each experimental result and take an average of the estimated coefficients
[b] Constraints of the form shown in Eq. (42)

and Hessian of the loss function becomes n^2 dimensional vector and matrix, respectively.

III. NUMERICAL TESTS

In this section, we perform numerical tests for our proposed methods against numerically generated synthetic data.

A. Synthetic Data

Suppose we have a graph consists of N nodes and several edges. A node has finite heat capacity, and an edge has finite heat transfer constant. Each node has a *heater* and a *thermometer*. Let us denote nodes by $i, j, k, \dots = 0, 1, \dots, N-1$. We denote the temperature and heater power at node i by T_i and P_i, respectively. For a node i, we denote the set of adjacent nodes to i by $\Gamma_i^{(1)}$. Using a graph theoretic statement, we can define more generic case: $\Gamma_i^{(n)}$ is the set of nodes connected to the node i by a path of length n and not connected to i by any paths shorter than n.

The heat energy ith heater generates transfers to adjacent nodes $j \in \Gamma_i^{(1)}$ with heat transfer constants $\kappa_{ji} > 0$. We denote the heat capacity of node i by $C_i > 0$. Note that, the heat transfer constant κ_{ji} is zero for non-adjacent pairs of nodes i, j:

$$\kappa_{ji} = 0 \quad (j \notin \Gamma_i^{(1)}). \tag{49}$$

As the special case of the above equation, we have $\kappa_{ii} = 0$ for any node i.

The heat transfer equation for node i becomes as follows:

$$\frac{\mathrm{d}T_i}{\mathrm{d}t} = \sum_{j \in \Gamma_i^{(1)}} \frac{\kappa_{ij}}{C_i}(T_j - T_i) + \frac{1}{C_i} P_i. \tag{50}$$

Let us denote the state vector of the system by $\boldsymbol{T} = \left[T_0, T_1, \dots, T_{N-1}\right]^\top$ and input vector by $\boldsymbol{P} = \left[P_0, P_1, \dots, P_{N-1}\right]^\top$. The system of the above heat transfer equations for $i = 0, 1, \dots, N-1$ is equivalent to the following linear nonautonomous dynamical system:

$$\frac{\mathrm{d}\boldsymbol{T}}{\mathrm{d}t} = A\boldsymbol{T}(t) + B\boldsymbol{P}(t), \tag{51}$$

with matrices $A = \left[A_{ij}\right] \in \mathbf{R}^{N\times N}$ and $B = \left[B_{ij}\right] \in \mathbf{R}^{N\times N}$ being

$$A_{ij} = \frac{\kappa_{ij} - \delta_{ij}\sum_j \kappa_{ij}}{C_i}, \tag{52}$$

and

$$B_{ij} = \frac{\delta_{ij}}{C_i}. \tag{53}$$

Note that, the matrix $A = [A_{ij}]$ satisfies the following conditions:

$$A_{ij} > 0 \text{ and } \sum_k A_{ik} = 0 \quad (i, j = 0, 1, \dots, N-1). \tag{54}$$

In this set-up, our aim is to determine the matrices A and B from time-series data $\boldsymbol{T}(t)$ and $\boldsymbol{P}(t)$.

We generate a matrix A randomly so that A satisfies the conditions Eq. (54), and simulate the time-evolutions of temperature $\boldsymbol{T}^{(\mu)}$ for $\boldsymbol{P} = 0$ and initial temperature distributions $\boldsymbol{T}_0^{(\mu)}$ for $\mu = 0, 1, \dots, N!$. We define the vectors $\boldsymbol{T}_0^{(\mu)}$ using the permutation function:

$$\boldsymbol{T}_0^{(\mu)} = \mathrm{perm}(\boldsymbol{T}_0^{(0)}, \mu) \quad (\mu = 1, 2, \dots, N!), \tag{55}$$

where $\mathrm{perm}(-, \mu)$ applies the μth permutation to a vector. We assume that the permutation is enumerated by lexicographical order.

We simulate the trajectory for initial condition $\boldsymbol{T}_0^{(\mu)}$ using implicit Euler scheme. The resultant set of trajectories becomes as follows:

$$\mathcal{S} = \{\boldsymbol{T}^{(\mu)} | \mu = 0, 1, \dots, N!\}, \tag{56}$$

where $\boldsymbol{T}^{(\mu)} = \left[\boldsymbol{T}_0^{(\mu)}, \boldsymbol{T}_1^{(\mu)}, \dots, \boldsymbol{T}_{m-1}^{(\mu)}\right]$ is the discrete time-evolution with m time points.

We prepare the test dataset using the similar procedures as follows. For $\boldsymbol{T}_{\mathrm{test}}^{(0)} \in \mathbf{R}^N$, we prepare the initial conditions using the permutation function:

$$\boldsymbol{T}_{\mathrm{test}}^{(\mu)} = \mathrm{perm}(\boldsymbol{T}_{\mathrm{test}}^{(0)}, \mu) \quad (\mu = 1, 2, \dots, N!). \tag{57}$$

We simulate the time-evolutions using the implicit Euler scheme with m time points, and get the set of test trajectories

$$\mathcal{S}_{\mathrm{test}} = \{\boldsymbol{T}_{\mathrm{test}}^{(\mu)} | \mu = 0, 1, \dots, N! - 1\}. \tag{58}$$

We use the test dataset to evaluate the modeling methods below.

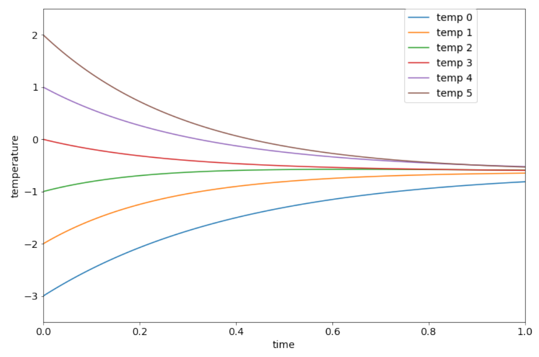

FIG. 2. The generated trajectory $\boldsymbol{T}^{(0)}(t) \in \mathcal{S}$. Each solid line (**temp** i) shows the ith component of $\boldsymbol{T}^{(0)}(t)$.

B. Numerical Set-ups

We set $N = 6$ and $\boldsymbol{T}_0^{(0)} = [-3, -2, -1, 0, 1, 2]^\top$ and $\boldsymbol{T}_{\text{test}}^{(0)} = [-2.5, -1.5, -0.5, 0.5, 1.5, 2.5]^\top$ for training and test dataset, respectively. Consequently, we got a set of $6! = 720$ training trajectories $\mathcal{S}$ and 720 test trajectories $\mathcal{S}_{\text{test}}$. We simulated the time-evolutions from $t = 0$ to $t = 1$ with uniformly discretized $m = 100$ time points. The generated coefficient $A \in \mathbf{R}^{6\times 6}$ is shown in Fig. 1; A has no zero element, so the graph we consider is a complete graph.

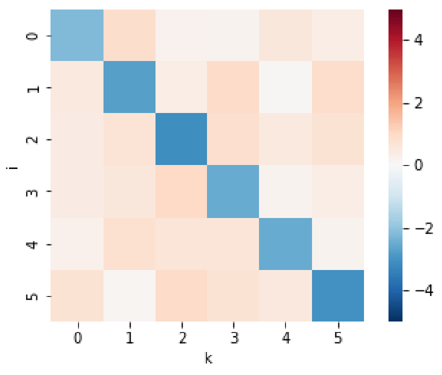

FIG. 1. The generated matrix $A = [A_{ik}]$.

The time-series data $\boldsymbol{T}^{(0)}(t)$ is shown in Fig. 2 below. As one can see, the temperatures approach to the equilibrium as time passes.

We applied the proposed method MTDMD for the increasing sequence of subsets:

$$\mathcal{S}_0 \subset \mathcal{S}_1 \subset \cdots \subset \mathcal{S}, \tag{59}$$

where, for $k = 0, 1, \ldots, N! - 1$,

$$\mathcal{S}_k = \{\boldsymbol{T}^{(\mu)} | \mu = 0, 1, \ldots, k-1\} \subset \mathcal{S}. \tag{60}$$

We adopt DMD and OP-DMD as the existing methods for comparison. As we have seen above, DMD and OP-DMD accept only one trajectory. To compare with the proposed method, we use average over the test trajectories, as follows: for $k = 1, 2, \ldots$ and $\texttt{method} = \texttt{DMD}, \texttt{OP-DMD}$,

$$A = k^{-1} \sum_{\mu=0}^{k-1} A^{(\mu)} \quad A^{(\mu)} = \texttt{fit}(\mathcal{S}_k, \texttt{method}), \tag{61}$$

where $\texttt{fit}(-, \text{method})$ returns the coefficient matrix A using $\texttt{method}$ for given set of trajectories. The resultant matrix A is used for the model reconstruction.

C. Modeling of Noiseless Data

We performed numerical test using noiseless synthetic data. We will show the results for noisy data in the next subsection.

The numerical result is shown in Fig. 3. As we can see, the proposed method achieves smaller RMSE compared to the existing methods (DMD and OP-DMD) for the cases with more than one trajectory.

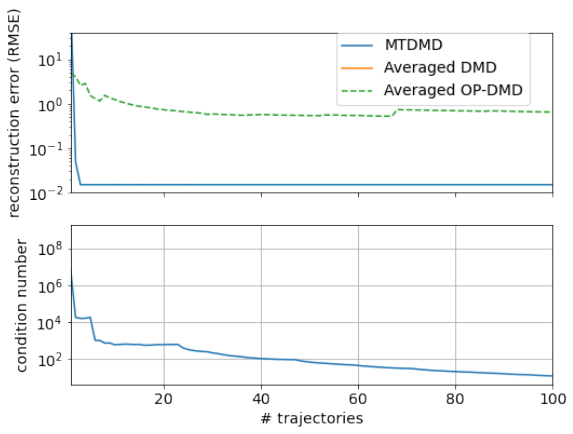

FIG. 3. Upper panel: Comparison of the reconstruction errors for noiseless time-series data. Blue solid line (**MTDMD**): the proposed method applied for increasing sequence of subsets Eq. (59). Green dashed line (**Averaged OP-DMD**): OP-DMD with the averaged coefficient matrix Eq. (61) for the same increasing sequence of subsets Eq. (59). The result of DMD (labeled as **Averaged DMD**) with the same procedure is not shown, because the reconstruction error diverged. The reconstruction error is defined for the test dataset $\mathcal{S}_{\text{test}}$ Eq. (58) and shown in root mean squared error (RMSE) per trajectory per DoF per time point. Lower panel: Condition number of the Hessian matrix in the MTDMD for the noiseless time-series data.

An example for reconstruction is shown in Fig. 4 below.

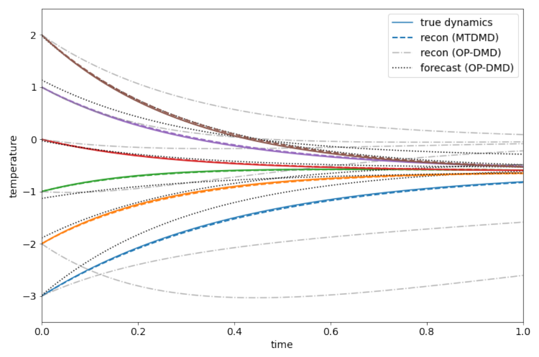

FIG. 4. The reconstruction results for the model identified using noiseless time-series data. Solid lines (**true dynamics**): the ground truth (generated trajectory) $\boldsymbol{T}^{(0)}(t)$. Dashed lines (**recon (MTDMD)**): the reconstructed time-series data using coefficient matrix A modeled by MTDMD. Dash-dotted lines (**recon (OP-DMD)**): the reconstructed time-series data for OP-DMD using the averaged coefficient matrix Eq. (61). Black dotted lines (**forecast (OP-DMD)**): the reconstructed time-series data using the `forecast` method of pyDMD (`svd_rank` = 2) fitted to $\boldsymbol{T}^{(99)}$. The reconstruction results by DMD are not shown, because the time-evolutions diverged. The number of trajectories used for system identification is 100, and the other details of the system identification for OP-DMD and DMD are the same as shown in Fig. 3.

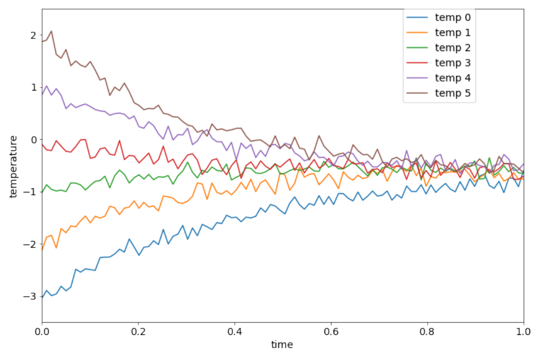

FIG. 5. The generated trajectory $\boldsymbol{T}^{(0)}(t)$ plus the i.i.d. Gaussian noise with 0 mean and standard deviation $\sigma = 0.1$ for each time point. Each solid line (**temp** i) shows the ith component of $\boldsymbol{T}^{(0)}(t)$.

One can see that the reconstruction by MTDMD matches the ground truth, while the reconstruction by OP-DMD fails to mimic the qualitative behaviors of the dynamics. The reason for this improvement is partly explained by the improvements in the condition number of the Hessian matrix.

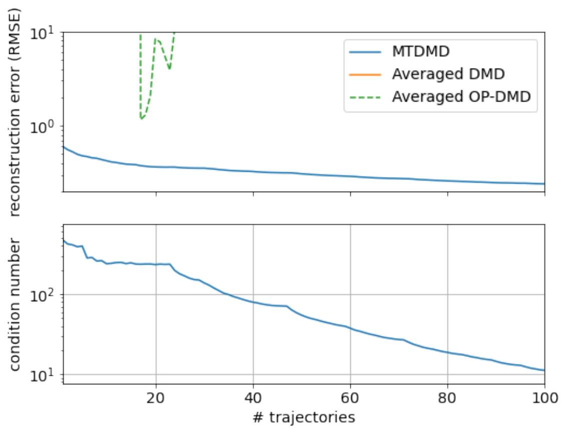

FIG. 6. Upper panel: Comparison of the reconstruction errors for noisy time-series data. Blue solid line (**MTDMD**): the proposed method applied for increasing sequence of subsets Eq. (59). Green dashed line (**Averaged OP-DMD**): OP-DMD with the averaged coefficient matrix Eq. (61) for the same increasing sequence of subsets Eq. (59). The result of DMD (labeled as **Averaged DMD**) with the same procedure is not shown, because the reconstruction error diverged. The reconstruction error is defined for the test dataset $\mathcal{S}_{\text{test}}$ Eq. (58) and shown in root mean squared error (RMSE) per trajectory per DoF per time point. Lower panel: Lower panel: Condition number of the Hessian matrix in the MTDMD for the noisy time-series data.

D. Modeling of Noisy Data

We now show numerical results for noisy data in this subsection. We used the same matrix A and the (noiseless) trajectories shown in the previous subsection, and added independently and identically distributed (i.i.d.) Gaussian noise with zero mean and standard deviation $\sigma = 0.1$ to each time point of the trajectories. In this section, we identify the model coefficient A using noisy data, and evaluate the reconstruction errors for the obtained A against *noiseless* data. All other set-ups are the same as we used in the previous subsection.

The noisy time-series data $\boldsymbol{T}^{(0)}(t)$ is shown in Fig. 5 below.

The numerical result is shown in Fig. 6. As we can see, the proposed method achieved smaller RMSE compared to the existing methods (DMD and OP-DMD) for the cases with more than one trajectory.

An example for reconstruction is shown in Fig. 7 below.

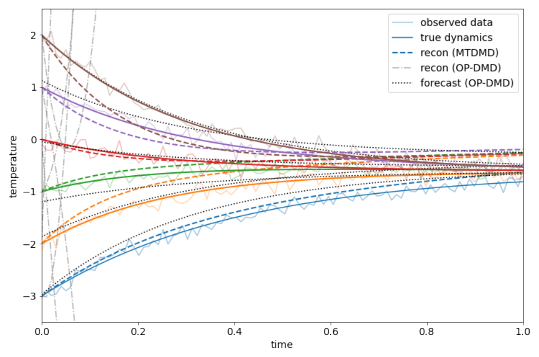

FIG. 7. The reconstruction results for the model identified using noisy time-series data. Solid lines (**true dynamics**): the ground truth (generated trajectory) $\boldsymbol{T}^{(0)}(t)$. Shaded solid lines (**observed data**): noisy, observed data. Dashed lines (**recon (MTDMD)**): the reconstructed time-series data using coefficient matrix A modeled by MTDMD. Dash-dotted lines (**recon (OP-DMD)**): the reconstructed time-series data for OP-DMD using the averaged coefficient matrix Eq. (61). Black dotted lines (**forecast (OP-DMD)**): the reconstructed time-series data using the `forecast` method of pyDMD (`svd_rank` = 2) fitted to $\boldsymbol{T}^{(99)}$. The reconstruction results by DMD are not shown, because the time-evolutions diverged. The number of trajectories used for system identification is 100, and the other details of the system identification for OP-DMD and DMD are the same as shown in Fig. 3.

One can see that the reconstruction by MTDMD approximately coincides with the ground truth, while the reconstruction by OP-DMD fails to reconstruct the physical behavior of the dynamics at all. The reason for this improvement is, again, partly explained by the improvements in the condition number of the Hessian matrix. Interestingly, the condition number declines quicker compared to the case with noiseless data.

E. Computational Cost

The computational time for parameter optimization by MTDMD using 100 trajectories is 0.0127 seconds, while fitting DMD and OP-DMD for 100 trajectories one-by-one and take an average over the resultant 100 coefficient matrices requires 0.0639 seconds and 2.174 seconds, respectively. The computer used in this numerical experiment has Intel® Core$^{\mathrm{TM}}$ i7-10610U processor [17] and 16.0 GB of random access memory. We implemented the numerical code in `Python` and no multi-thread computing used explicitly.

IV. DISCUSSION

The proposed method, MTDMD, is shown to be able to model the linear dynamical system effectively and accurately using multiple trajectories even for noisy time-series data. The reconstruction error (RMSE) for the proposed method with 100 trajectories is approximately 2×10^{-2} for noiseless data and approximately 0.3 for noisy data (noise standard deviation $\sigma = 0.1$). Note that, the initial temperature is of order of unity, and the reconstruction error is calculated against noiseless dynamics with different set of initial temperatures. The reconstruction error for MTDMD is much smaller compared to the existing DMD-based methods (DMD and OP-DMD with averaged coefficient matrices).

Although a careful selection of hyper-parameter of OP-DMD enables us to get a better reconstruction using the built-in `forecast` method of PyDMD for noisy time-series data, the reconstruction result using OP-DMD coefficients (i.e., without coefficient averaging over multiple trajectories and `forecast` method) is worse than the MTDMD results for noiseless time-series data. Note that, we can use the `forecast` method only for time-series data which belong to the set of training trajectories. The fact suggests that although OP-DMD can efficiently suppress the undesirable effects from observation noises, the generalizability of the identified model by OP-DMD is largely dependent on the *closeness* of the training trajectory to the test trajectory.

The improvement of the reconstruction by the MTDMD is partly attributed to the quick decay of the condition number $\kappa(H)$ of the Hessian H with respect to the number of the trajectories. A comparison between MTDMD using the noiseless- and noisy data shows us that the improvement in condition number is larger in noisy data. Another comparison between MTDMD and existing DMD-based methods using the noisy data indicates that even from noisy data one can reconstruct the original, noiseless data using the MTDMD.

It is also shown that, for sufficiently large number of trajectories, the proposed method achieves better reconstruction results compared to the existing methods not only in a sense of reconstruction error, but also in a sense of computational time. Note that, each trajectory is a 6×100 array, so the total amount of data is relatively large.

V. SUMMARY

We propose a numerical method MTDMD, or multi-trajectory DMD, for time-series data analysis. The proposed method is useful to model linear dynamical systems with noisy observations using multiple trajectories. We show that with MTDMD we can successfully identify the system using the set of noiseless- and noisy synthetic time-series data. The computational time for MTDMD is shown to be comparative to the existing DMD-based methods, and the reconstruction results are better in MTDMD, especially for cases with noisy time-series data. The MTDMD achieves not only good reconstruction results, but also numerically fast multi-trajectory modeling. 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, one may perform a qualitative comparison between MTDMD and the existing DMD-based methods for identification of non-autonomous dynamical systems. Another issue to be addressed in the future work is numerical test using large, complex systems.

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 have to 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 can not 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 pseudo-inverse

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

with matrix pseudo-inversion 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).

[17] Intel, the Intel logo, and Intel Core are trademarks of Intel Corporation or its subsidiaries.