A LAPACK implementation of the Dynamic Mode Decomposition I.[*]

Zlatko Drmač[†]

October 5, 2022

Abstract

The Dynamic Mode Decomposition (DMD) is a method for computational analysis of nonlinear dynamical systems in data driven scenarios. Based on the high fidelity numerical simulations or experimental data obtained using e.g. Particle Image Velocimetry/Thermometry technology, or recording new cases of a infectious disease in a territory, the DMD can be used to reveal the latent structures in the dynamics or as a forecasting or a model order reduction tool. The theoretical underpinning of the DMD is the Koopman composition operator on a Hilbert space of observables of the dynamics under study. The numerical realization of the method is in the framework of dense numerical linear algebra. This working note describes a LAPACK implementation of a variant of the DMD, and it shows that the state of the art dense numerical linear algebra is the tool of the trade for computational analysis of complex nonlinear dynamics, in particular in data driven scenarios. Fine numerical issues are discussed in detail. The material presented here is a basis for high performance implementations for large scale problems (e.g. for computational fluid dynamics) in the frameworks of ScaLAPACK, MAGMA and SLATE.

1 Introduction

In this report we propose a LAPACK [6] based implementation of the Dynamic Mode Decomposition (DMD). The DMD is a computational tool for analysis of the structure of nonlinear dynamical systems, introduced by Schmid [44] in the context of computational fluid dynamics (CFD). The theoretical underpinning of the DMD is the Koopman (composition) operator associated with the nonlinear dynamical system under study, which provides a particular global infinite dimensional linearization; see [9], [54], [52], [8].

The DMD and the computational Koopman operator framework are used in plethora of applications in CFD, e.g. to reveal the structure of the dynamics using numerically identified

[*]LAPACK Working Note 298. AIMdyn technical report AIMdyn-DARPA-SBIR-2022, Oct 4 2022.

[†]University of Zagreb, Faculty of Science, Mathematics Department, Croatia, in cooperation with AIMdyn Inc. Santa Barbara CA. This material is based upon work supported by the DARPA Small Business Innovation Research Program (SBIR) Program Office under Contract No. W31P4Q-21-C-0007 to AIMdyn, Inc. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the DARPA SBIR Program Office. Part of this work was supported by the DARPA PAI project "Physics-Informed Machine Learning Methodologies" Contract No: HR0011-18-9-0033 and the DARPA MoDyL project "A Data-Driven, Operator-Theoretic Framework for Space-Time Analysis of Process Dynamics" Contract No: HR0011-16-C-0116. Distribution Statement A: Approved for Public Release, Distribution Unlimited.

coherent structures of the flow [45], [46], [47], [37], [48], [28], [29], [43], [26] as a model order reduction method [4], or e.g. for forecasting and control. Other applications areas include robotics, aeroacustic, epidemiology, algorithmic trading on financial markets, video processing, neural networks and many others; for an overview see [2], [8]. The methodology is in particular useful in data driven scenarios. For an introduction to DMD see [33].

In practical computation, the infinite dimensional operator defined on a suitable Hilbert space of functions (observables) is compressed onto a finite dimensional subspace, and the entire process of the DMD based analysis of the underlying nonlinear dynamics is in the framework of numerical linear algebra. The matrix representation of the compression is derived in a data driven scenario, based on a collection of data snapshots that are represented as vectors in $\mathbb{R}^n$ or $\mathbb{C}^n$. The data acquisition is based on measurements (e.g. Particle Image Velocimetry/Thermometry for measuring velocity and, with suitable thermosensitive tracers, temperature in a flow), computer simulations (numerical solutions of partial differential equations), or e.g. collecting the reported covid infection cases from territorial units in a state.

The two main computational tasks in DMD analysis are: *(i)* Rayleigh-Ritz extraction of eigenvalues and eigenvectors using the subspace spanned by the data snapshots; *(ii)* spatio-temporal representation of the snapshots using a subset of the computed eigenpairs, which amounts to solving a structured least squares problem. The solutions of *(i)*, *(ii)* allow for an analysis of the structure of the dynamics, forecasting and control.

The data matrices are in general dense. Hence, for a state of the art implementation of the DMD, the LAPACK library is a natural computing platform that already contains all necessary subroutines. An advantage of a LAPACK-based implementation of the DMD is that the numerical robustness, run time performance and adaptation to new multi-core hardware and software computing platforms are derived from the development of LAPACK and LAPACK-based software such as ScaLAPACK [7]. Furthermore, the structure of the implementation is such that it can be used as a prototype for porting to GPU and multi-core architectures using MAGMA [50], [51], [16] and SLATE [1].[1]

The particular software solution of the DMD task *(i)*, presented in this report, is based on our earlier work [22], [17] and our main goal is to provide a robust computational tool for practitioners in aplied sciences and engineering. In addition, we offer a numerical analysis with insights that allow for a better understanding of the accuracy of the method and of its limits.

This note is organized as follows. In Section 2 we review the Koopman operator framework and the DMD. The material of this section is a brief tutorial to this subject and a user's guide not only for the software but for the DMD method in general. We believe that some details and new insights will be useful for the experts as well. A modification of the DMD introduced in [22] is reviewed in §3, where we provide an additional analysis and new insights, illustrated using numerical examples, that contribute to a better understanding of numerical issues and to the design of the software implementation. We briefly discuss a challenging problem of shadowing theory for the DMD. In Section 4 we describe the details of the implementations of the two proposed DMD subroutines. Additional numerical examples in Section 5 are selected from our test diary to illustrate the limits of the numerical accuracy.

[1]These ScaLAPACK, MAGMA and SLATE based versions of the code presented in this report will be developed in a separate work.

2 Preliminaries

The main goal of this section is to describe the framework in which the software presented in this note is used. For more details of the theory we refer to [49], [36], [31], [9], [25], [32].

We set the stage in §2.1, with the introduction of the Koopman operator on the space of the observables. Operator compression to a finite dimensional subspace in a data driven scenario and approximations of eigenvalues and eigenfunctions are described in §2.2. For the sake of completeness and for better understanding of the proposed software design, in §2.3 and §2.4.3 we briefly outline the spatio-temporal modal representation of the data snapshots.[2] The DMD and its connection to the Koopman operator are described in §2.4; in §2.4.2, we review the original Schmid's DMD algorithm.

2.1 The space of observables and the Koopman operator

Consider an autonomous continuous dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \equiv \begin{pmatrix} \mathbf{F}_1(\mathbf{x}(t)) \\ \vdots \\ \mathbf{F}_N(\mathbf{x}(t)) \end{pmatrix}, \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{1}$$

with a state space[3] $\mathcal{X} \subset \mathbb{R}^N$ and vector-valued nonlinear function $\mathbf{F} : \mathcal{X} \to \mathbb{R}^N$. The corresponding flow map $\boldsymbol{\varphi}^t$ of the system, that advances an initial state $\mathbf{x}(t_0)$ to $\mathbf{x}(t_0 + t)$ is

$$\mathbf{x}(t_0 + t) = \boldsymbol{\varphi}^t(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{F}(\mathbf{x}(\tau))d\tau. \tag{2}$$

The key idea is to consider observables of the systems, i.e. functions of the states, $f : \mathcal{X} \to \mathbb{C}$, $f \in \mathcal{F}$, where the function space is selected as e.g.[4] $\mathcal{F} = L^p(\mathcal{X}, \mu)$ $(1 \leq p \leq \infty)$.

The values of the observables along trajectories of the system can be described using the Koopman operator semigroup $(\mathcal{U}_{\boldsymbol{\varphi}^t})_{t\geq 0}$ defined by

$$\mathcal{U}_{\boldsymbol{\varphi}^t} f = f \circ \boldsymbol{\varphi}^t, \quad f \in \mathcal{F}. \tag{3}$$

Clearly, $\mathcal{U}_{\boldsymbol{\varphi}^t}$ is linear operator, and the semigroup property follows from $\boldsymbol{\varphi}^t \circ \boldsymbol{\varphi}^s = \boldsymbol{\varphi}^{t+s}$. It is an infinite dimensional linearization of (1) that takes the action into the space $\mathcal{F}$ of observables.

An analogous construction of the Koopman operator applies to a discrete dynamical system

$$\mathbf{z}_{i+1} = \mathbf{T}(\mathbf{z}_i), \tag{4}$$

where $\mathbf{T} : \mathcal{X} \longrightarrow \mathcal{X}$ is a measurable nonlinear map on a state space $\mathcal{X}$ and $i \in \mathbb{Z}$. The Koopman operator $\mathcal{U} \equiv \mathcal{U}_{\mathbf{T}}$ for the discrete system is defined analogously by

$$\mathcal{U} f = f \circ \mathbf{T}, \quad f \in \mathcal{F}. \tag{5}$$

In a practical computation, continuous systems are always approximated using discrete systems. Indeed, if we run a numerical simulation of the ODE's (1) in a time interval $[t_0, t_*]$, the numerical solution is obtained on a discrete equidistant grid with fixed time lag Δt:

$$t_0, \; t_1 = t_0 + \Delta t, \; \ldots, \; t_{i-1} = t_{i-2} + \Delta t, \; t_i = t_{i-1} + \Delta t, \; \ldots \tag{6}$$

[2]This is the computational task *(ii)* from §1 and it will not be further analyzed in this report.
[3]In more general case, $\mathcal{X}$ is smooth N-dimensional compact manifold, with Borel σ algebra $\mathcal{B}$.
[4]Choosing the space $\mathcal{F}$ properly is a separate issue, not considered here.

In this case, a software toolbox acts as a discrete dynamical system $\mathbf{z}_i = \mathbf{T}(\mathbf{z}_{i-1})$ that produces the discrete sequence of $\mathbf{z}_i \approx \mathbf{x}(t_i)$; this can be described as sampling the original system with noise. For $t_i = t_0 + i\Delta t$ we have (using the definitions of $\boldsymbol{\varphi}^{\Delta t}$, $\mathcal{U}_{\boldsymbol{\varphi}^{\Delta t}}$ and the semigroup property)

$$f(\mathbf{x}(t_0 + i\Delta t)) = (f \circ \boldsymbol{\varphi}^{i\Delta t})(\mathbf{x}(t_0)) = (\mathcal{U}_{\boldsymbol{\varphi}^{i\Delta t}} f)(\mathbf{x}(t_0)) = (\mathcal{U}^i_{\boldsymbol{\varphi}^{\Delta t}} f)(\mathbf{x}(t_0)), \tag{7}$$

where $\mathcal{U}^i_{\boldsymbol{\varphi}^{\Delta t}} = \mathcal{U}_{\boldsymbol{\varphi}^{\Delta t}} \circ \ldots \circ \mathcal{U}_{\boldsymbol{\varphi}^{\Delta t}}$. On the other hand, using $\mathcal{U}f = f \circ \mathbf{T}$, we have along the trajectory $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_i$,

$$f(\mathbf{z}_i) = f(\mathbf{T}(\mathbf{z}_{i-1})) = \ldots = f(\mathbf{T}^i(\mathbf{z}_0)) = (\mathcal{U}^i f)(\mathbf{z}_0), \tag{8}$$

where $\mathbf{T}^2 = \mathbf{T} \circ \mathbf{T}$, $\mathbf{T}^i = \mathbf{T} \circ \mathbf{T}^{i-1}$. Hence, in a software simulation of (1) with the initial condition $\mathbf{z}_0 = \mathbf{x}(t_0)$, we have the approximations

$$(\mathcal{U}^i f)(\mathbf{z}_0) \approx (\mathcal{U}^i_{\boldsymbol{\varphi}^{\Delta t}} f)(\mathbf{z}_0), \;\; f \in \mathcal{F},\; \mathbf{z}_0 \in \mathcal{X}, \;\; i = 0, 1, 2, \ldots \tag{9}$$

with accuracy that depends on the numerical scheme deployed in the software. The problem of numerical approximation of trajectories of dynamical systems is studied in the shadowing theory, see e.g. [39], [40].

For a proper analysis of the underlying phenomena the sampling frequency must be set appropriately relative to the Nyquist frequency [46]. Here we do not consider those issues that depend on a particular application; our main goal is to provide robust computational tools for using DMD in a variety of applications. Therefore, in the sequel we consider only the discrete systems (4), (5).

2.1.1 Vector valued observables

The observables can be physical quantities such as pressure, temperature, velocity, obtained by meassurements or numerical simulations and mathematical constructs using suitable classes of functions (e.g. multivariate Hermite polynomials, radial basis functions), or embeding in higher dimensions using time delays. Thus, it is natural to simultaneously consider several observables along a trajectory of the system, i.e. to consider vector valued observables. For a vector valued $\mathbf{f} = (f_1, \ldots, f_d) : \mathcal{X} \longrightarrow \mathbb{C}^d$, the composition operator is defined component-wise as

$$\mathcal{U}_d \mathbf{f} = \begin{pmatrix} f_1 \circ \mathbf{T} \\ \vdots \\ f_d \circ \mathbf{T} \end{pmatrix} = \begin{pmatrix} \mathcal{U} f_1 \\ \vdots \\ \mathcal{U} f_d \end{pmatrix}. \tag{10}$$

In particular, if we set $d = N$, $f_k(\mathbf{z}) = e_k^T \mathbf{z}$, where $\mathbf{z} \in \mathbb{C}^N$, $e_k = (\boldsymbol{\delta}_{jk})_{j=1}^N$, $k = 1, \ldots, N$, then $\mathbf{f}(\mathbf{z}) = \mathbf{z}$ is full state observable and $(\mathcal{U}_d \mathbf{f})(\mathbf{z}_i) = \mathbf{z}_{i+1}$.

In a more general setting of the Extended DMD (EDMD) the dictionary of best fitting observables can be dynamically learned e.g. using neural networks, see e.g. [3], [34].

2.1.2 Data snapshots and data driven framework

A (data) snapshot is a numerical value of a scalar or vector valued observable at a specific instance in time. For example, snapshots may be obtained as

- quantification of images obtained from high speed camera recordings of a combustion process in a turbine
- new cases of covid 19 infections, reported daily
- wind tunnel PIV measurements

- numerical simulation of (1) represented by (7), (8), (9), where we can feed an initial $\mathbf{z}_0$ to a software toolbox (representing $\mathbf{T}$) to obtain the sequence
$$\mathbf{f}(\mathbf{z}_0) = (\mathcal{U}_d^0\mathbf{f})(\mathbf{z}_0),\ \mathbf{f}(\mathbf{z}_1) = (\mathcal{U}_d\mathbf{f})(\mathbf{z}_0),\ \mathbf{f}(\mathbf{z}_2) = (\mathcal{U}_d^2\mathbf{f})(\mathbf{z}_0), \ldots, \mathbf{f}(\mathbf{z}_{M+1}) = (\mathcal{U}_d^{M+1}\mathbf{f})(\mathbf{z}_0), \tag{11}$$
where $\mathbf{f} = (f_1, \ldots, f_d)^T$ is a vector valued ($d > 1$) observable with the action of $\mathcal{U}_d$ defined component-wise (10).

Although genereted in general by a nonlinear system, the snapshots are a Krylov sequence $\mathbf{f}$, $\mathcal{U}_d\mathbf{f}$, $\mathcal{U}_d^2\mathbf{f}, \ldots$, driven by the linear operator $\mathcal{U}_d$ and evaluated along a trajectory initialized at $\mathbf{z}_0$. We will conveniently arrange the data snapshots in the snapshot matrix $\mathbb{F}$ with columns $\mathbf{f}(\mathbf{z}_0), \mathbf{f}(\mathbf{z}_{k+1}) = (\mathcal{U}_d\mathbf{f})(\mathbf{z}_k)$, where $\mathbf{z}_{k+1} = \mathbf{T}(\mathbf{z}_k)$:
$$\mathbb{F} = \begin{pmatrix}\mathbf{f}(\mathbf{z}_0) & \mathbf{f}(\mathbf{z}_1) & \ldots & \mathbf{f}(\mathbf{z}_M) & \mathbf{f}(\mathbf{z}_{M+1})\end{pmatrix} = \begin{pmatrix} f_1(\mathbf{z}_0) & f_1(\mathbf{z}_1) & \ldots & f_1(\mathbf{z}_M) & f_1(\mathbf{z}_{M+1}) \\ f_2(\mathbf{z}_0) & f_2(\mathbf{z}_1) & \ldots & f_2(\mathbf{z}_M) & f_2(\mathbf{z}_{M+1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_d(\mathbf{z}_0) & f_d(\mathbf{z}_1) & \ldots & f_d(\mathbf{z}_M) & f_d(\mathbf{z}_{M+1}) \end{pmatrix} \in \mathbb{C}^{d\times(M+2)}.$$
Thus, if we set $\mathbf{X} = \mathbb{F}(1:d, 1:M+1)$, $\mathbf{Y} = \mathbb{F}(1:d, 2:M+2)$, then $\mathbf{x}_k = \mathbf{f}(\mathbf{z}_k)$, $\mathbf{y}_k = \mathbf{f}(\mathbf{T}(\mathbf{z}_k))$.

In fact, $\mathbf{X}$ and $\mathbf{Y}$ are not necessarily extracted from a single trajectory. The data may consist of several short bursts with different initial conditions, arranged as a sequence of column vector pairs of snapshots $(\mathbf{x}_k, \mathbf{y}_k)$, where $\mathbf{x}_k = \mathbf{f}(\mathbf{z}_k)$, $\mathbf{y}_k = \mathbf{f}(\mathbf{T}(\mathbf{z}_k))$ column-wise so that a kth column in $\mathbf{Y}$ corresponds to the value of the observable in the kth column of $\mathbf{X}$ through the action of $\mathcal{U}_d$.

In this paper, we work under a tacit assumption that $d \gg M$, i.e. the snapshots are from a high dimensional space and the total number of pairs $(\mathbf{x}_i, \mathbf{y}_i)$ is much smaller than the state space dimension.

2.2 Compression of $\mathcal{U}$ in a data driven scenario

In practical computation with $\mathcal{U}$ we face two difficulties: First, computing with infinite dimension is not feasible and we have to compress $\mathcal{U}$ onto a finite dimensional subspace of $\mathcal{F}$. Secondly, in a data driven scenario we are given a certain number of data snapshots and at the moment of numerical computation we may not have a luxury of requesting and receiving more information. In an application, we should be aware of that limitation, in particular that the supplied data may not contain the desired information. For some related numerical details we refer to [23].

For a finite dimensional compression, we first choose a subspace $\mathcal{F}_\mathcal{D} \subset \mathcal{F}$ spanned by a dictionary of scalar functions $\mathcal{D} = \{f_1, \ldots, f_d\}$; this includes the case of full state observables and $d = N$. Then, we compute a matrix representation $\mathbb{U}$ of the compression $\mathbf{\Psi}_{\mathcal{F}_\mathcal{D}}\mathcal{U}_{|\mathcal{F}_\mathcal{D}} : \mathcal{F}_\mathcal{D} \longrightarrow \mathcal{F}_\mathcal{D}$, where $\mathbf{\Psi}_{\mathcal{F}_\mathcal{D}}$ is a suitable projection with the range $\mathcal{F}_\mathcal{D}$. This is the standard construction of matrix representation of linear operator: we need a representation of $\mathcal{U}f_i$ of the form
$$(\mathcal{U}f_i)(s) = f_i(\mathbf{T}(s)) = \sum_{j=1}^{d} \mathbf{u}_{ji} f_j(s) + \rho_i(s), \quad i = 1, \ldots, d, \quad s \in \mathcal{X}. \tag{12}$$
Projecting $\mathcal{U}f_i$ back onto $\mathcal{F}_\mathcal{D}$ is feasible only in a very limited sense because the functions can be evaluated only at the provided data snapshots. Thus, the best we can do is to project in the algebraic least squares sense: we can define the matrix $\mathbb{U} = (\mathbf{u}_{ji}) \in \mathbb{C}^{d\times d}$ column-wise by minimizing the residual $\rho_i(s)$ in (12) over the states $s = \mathbf{z}_k$, using the values
$$(\mathcal{U}f_i)(\mathbf{z}_k) = f_i(\mathbf{T}(\mathbf{z}_k)), \quad i = 1, \ldots, d; \quad k = 0, \ldots, M. \tag{13}$$

To that end, write the least squares residual

$$\frac{1}{M+1}\sum_{k=0}^{M}|\rho_i(\mathbf{z}_k)|^2 = \frac{1}{M+1}\sum_{k=0}^{M}|\sum_{j=1}^{d}\mathbf{u}_{ji}f_j(\mathbf{z}_k) - f_i(\mathbf{T}(\mathbf{z}_k))|^2, \tag{14}$$

which is the L^2 residual with respect to the empirical measure defined as the sum of the Dirac measures concentrated at the $\mathbf{z}_k$'s, $\boldsymbol{\delta}_{M+1} = (1/(M+1))\sum_{k=0}^{M}\boldsymbol{\delta}_{\mathbf{z}_k}$. Hence, the columns of the matrix representation $\mathbb{U}$ are defined as the solutions of the least squares problems

$$\int\left|\sum_{j=1}^{d}\mathbf{u}_{ji}f_j - f_i\circ\mathbf{T}\right|^2 d\boldsymbol{\delta}_{M+1} = \gamma_M\left\|\left[\begin{pmatrix} f_1(\mathbf{z}_0) & \dots & f_d(\mathbf{z}_0)\\ \vdots & \dots & \vdots\\ f_1(\mathbf{z}_M) & \dots & f_d(\mathbf{z}_M)\end{pmatrix}\begin{pmatrix}\mathbf{u}_{1i}\\ \vdots\\ \mathbf{u}_{di}\end{pmatrix} - \begin{pmatrix} f_i(\mathbf{T}(\mathbf{z}_0))\\ \vdots\\ f_i(\mathbf{T}(\mathbf{z}_M))\end{pmatrix}\right]\right\|_2^2 \longrightarrow \min_{\mathbf{u}_{1i},\dots,\mathbf{u}_{di}},$$

for $i = 1,\dots,d$; $\gamma_M = 1/(M+1)$. The solutions of the above algebraic least squares problems for all $i = 1,\dots,d$ are compactly written as the matrix $\mathbb{U}\in\mathbb{C}^{d\times d}$ that minimizes $\|\mathbf{X}^T\mathbb{U} - \mathbf{Y}^T\|_F$. To ensure well defined projection $\boldsymbol{\Psi}_{\mathcal{F}_{\mathcal{D}}}$, uniquely determined $\mathbb{U}$ is usually obtained with the additional constraint that $\|\mathbb{U}\|_F$ is minimal. In that case, the solution $\mathbb{U}$ can be written using the Moore-Penrose generalized inverse as

$$\mathbb{U} = (\mathbf{X}^T)^{\dagger}\mathbf{Y}^T \equiv (\mathbf{Y}\mathbf{X}^{\dagger})^T, \tag{15}$$

and the action of $\mathcal{U}$ can be represented, using (12), as

$$\mathcal{U}\begin{pmatrix} f_1(s) & \dots & f_d(s)\end{pmatrix} \overset{def}{=} \begin{pmatrix}\mathcal{U}f_1(s) & \dots & \mathcal{U}f_d(s)\end{pmatrix} = \begin{pmatrix} f_1(s) & \dots & f_d(s)\end{pmatrix}\mathbb{U} + \begin{pmatrix}\rho_1(s) & \dots & \rho_d(s)\end{pmatrix}. \tag{16}$$

Remark 2.1 The condition of minimality of $\|\mathbb{U}\|_F$ that fixes a uniquely determined point in the linear manifold of the solutions of $\|\mathbf{X}^T\mathbb{U} - \mathbf{Y}^T\|_F \to \min$ and yields the formula (15) is convenient, but somewhat arbitrary. With any $\Delta\mathbb{U}$ such that $\mathbf{X}^T\Delta\mathbb{U} = \mathbf{0}$, the matrix $\mathbb{U}+\Delta\mathbb{U}$ is also a minimizer. We will see later in §2.4.1 that, although convenient, the explicit formula such as (15) is not needed in the DMD computation, in fact it might be misleading, and that the ambiguity in choosing a particular $\mathbb{U}$ is immaterial in the presented algorithms.

2.2.1 Approximate eigenfunctions

The key for the application of the Koopman operator is representation of the snapshots in terms of the eigenfunctions. The spectral theory of the Koopman operator is complicated, see [31]. Once we have a finite dimensional compression we are bound to approximate only the eigenvalues and their eigenspaces; the consideration of the continuous spectrum are out of scope of this paper; interested reader is referred to [8, §4.4]. What we can do is to use (16) to extract numerical information on approximate eigenfunctions. For a convergence theory we refer to [30].

Assume for simplicity that $\mathbb{U}$ has full set of eigenvectors, $\mathbb{U}\mathbf{q}_i = \lambda_i\mathbf{q}_i$, so that $\mathbb{U} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$, with $\Lambda = \mathrm{diag}(\lambda_i)_{i=1}^{d}$, $\mathbf{Q} = (\mathbf{q}_1,\dots,\mathbf{q}_d)$. Following (16), we have for $s\in\mathcal{X}$,

$$\mathcal{U}\begin{pmatrix} f_1(s) & \dots & f_d(s)\end{pmatrix}\mathbf{Q} = \begin{pmatrix} f_1(s) & \dots & f_d(s)\end{pmatrix}\mathbf{Q}\Lambda + \begin{pmatrix}\rho_1(s) & \dots & \rho_d(s)\end{pmatrix}\mathbf{Q},$$

and the approximate eigenfunctions of $\mathcal{U}$, extracted from the span of $f_1,\dots,f_d$, are

$$\begin{pmatrix}\phi_1(s)\dots\phi_d(s)\end{pmatrix} = \begin{pmatrix} f_1(s)\dots f_d(s)\end{pmatrix}\mathbf{Q},\quad (\mathcal{U}\phi_i)(s) = \lambda_i\phi_i(s) + \sum_{j=1}^{d}\rho_j(s)\mathbf{Q}_{ji}. \tag{17}$$

In a numerical simulation, these eigenfunctions are accessible, as well as the observables, only as the tabulated values for $s \in \{\mathbf{z}_0, \ldots, \mathbf{z}_M\}$:

$$\begin{pmatrix} \phi_1(\mathbf{z}_0) & \phi_2(\mathbf{z}_0) & \dots & \phi_d(\mathbf{z}_0) \\ \phi_1(\mathbf{z}_1) & \phi_2(\mathbf{z}_1) & \dots & \phi_d(\mathbf{z}_1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(\mathbf{z}_{M+1}) & \phi_2(\mathbf{z}_{M+1}) & \dots & \phi_d(\mathbf{z}_{M+1}) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{z}_0) & f_2(\mathbf{z}_0) & \dots & f_d(\mathbf{z}_0) \\ f_1(\mathbf{z}_1) & f_2(\mathbf{z}_1) & \dots & f_d(\mathbf{z}_1) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(\mathbf{z}_{M+1}) & f_2(\mathbf{z}_{M+1}) & \dots & f_d(\mathbf{z}_{M+1}) \end{pmatrix} \mathbf{Q} = \mathbb{F}^T \mathbf{Q}. \qquad (18)$$

2.3 A spectral representation of the snapshots

Now that we have (17) and (18), we can try to represent a vector valued observable in terms of approximate numerical eigenfunctions. Let $\mathbf{g}(s)^T = (g_1(s), \ldots, g_d(s))$ be a vector valued observable such that $\mathbf{g}(s)^T = (f_1(s), \ldots, f_d(s))\Gamma$ with some coefficients $\Gamma = (\gamma_{ji}) \in \mathbb{C}^{d\times d}$. If $g_i = f_i$, then $\Gamma = \mathbb{I}_d$. In terms of the ϕ_i's, $\mathbf{g}(z)$ can be expressed as

$$\mathbf{g}(z)^T = \begin{pmatrix} f_1(s) & \ldots & f_d(s) \end{pmatrix} \mathbf{Q}\mathbf{Q}^{-1}\Gamma = \begin{pmatrix} \phi_1(s) & \ldots & \phi_d(s) \end{pmatrix} \mathbf{Q}^{-1}\Gamma, \quad z \in \mathcal{X}.$$

Set $\mathbf{Z} = \Gamma^T \mathbf{Q}^{-T} = \begin{pmatrix} \mathbf{z}_1 & \ldots & \mathbf{z}_d \end{pmatrix}$, where $\mathbf{z}_i$ is the ith column. Then

$$\begin{pmatrix} g_1(s) \\ \vdots \\ g_d(s) \end{pmatrix} = \underbrace{\Gamma^T \mathbf{Q}^{-T}}_{\mathbf{V}} \begin{pmatrix} \phi_1(s) \\ \vdots \\ \phi_d(s) \end{pmatrix} = \sum_{i=1}^{d} \mathbf{z}_i \phi_i(s).$$

Since $(\mathcal{U}\phi_i)(s) \approx \lambda_i \phi_i(s)$, we have the *Koopman mode decomposition*

$$(\mathcal{U}_d^k \mathbf{g})(s) = \begin{pmatrix} (\mathcal{U}^k g_1)(s) \\ \vdots \\ (\mathcal{U}^k g_d)(s) \end{pmatrix} \approx \sum_{i=1}^{d} \mathbf{z}_i \phi_i(s) \lambda_i^k, \quad k = 0, 1, 2, \ldots. \qquad (19)$$

From this representation of $\mathbf{g}$, that can be evaluated at the snapshots using (18), we see that:

(i) Forecasting the future values of $\mathbf{g}$, beyond the last received snapshots, ammounts to applying $\mathcal{U}_d$, which reduces to raising the powers of the eigenvalues.

(ii) If a reasonably good approximation can be obtained only with a small subset of of the modes $\mathbf{z}_i$ (with suitably computed coefficients in the linear combination), then the decomposition discovers a latent structure of the dynamics.

The matrix of the modes $\mathbf{Z} = \Gamma^T \mathbf{Q}^{-T}$ requires $\mathbf{Q}^{-T}$. This matrix can be computed directly if we diagonalize $\mathbb{A} = \mathbb{U}^T$, since $\mathbb{A}^T = \mathbb{U} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$ implies $\mathbb{A}\mathbf{Q}^{-T} = \mathbf{Q}^{-T}\Lambda$, i.e. the columns of $\mathbf{Q}^{-T}$ are the (right) eigenvectors of $\mathbb{A}$. Hence, for computing the Koopman modes, we can proceed with computing the eigenvectors of $\mathbb{A}$. When doing that we should keep in mind Remark 2.1.

2.4 The DMD

The matrix $\mathbb{A} = \mathbb{U}^T = \mathbf{Y}\mathbf{X}^\dagger$ is in the previous section introduced as an auxiliary object to compute the matrix $\mathbf{Q}^{-T}$ directly as the eigenvector matrix of $\mathbb{A}$, instead of first computing $\mathbf{Q}$ as the eigenvector matrix of $\mathbb{U}$. But, $\mathbb{A}$ has a direct link to the Koopman operator of its own. Namely, we can rewrite (16) as $\mathcal{U}_d \mathbf{f}(s) = \mathbb{A}\mathbf{f}(s) + \boldsymbol{\rho}(s)$, i.e.

$$\begin{pmatrix} \mathcal{U} f_1(s) \\ \vdots \\ \mathcal{U} f_d(s) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{T}(s)) \\ \vdots \\ f_d(\mathbf{T}(s)) \end{pmatrix} = \mathbb{A} \begin{pmatrix} f_1(s) \\ \vdots \\ f_d(s) \end{pmatrix} + \begin{pmatrix} \rho_1(s) \\ \vdots \\ \rho_d(s) \end{pmatrix}. \qquad (20)$$

Hence, using the powers of $\mathbb{A}$ mimics the Krylov sequence of $\mathcal{U}_d$ as in (11). More generaly, if we have a sequence of snapshot pairs $(\mathbf{x}_k, \mathbf{y}_k)_{k\geq 0}$ such that $\mathbf{x}_k = \mathbf{f}(\mathbf{z}_k)$, $\mathbf{y}_k = \mathbf{f}(\mathbf{T}(\mathbf{z}_k))$, then at $s = \mathbf{z}_k$ the relation (20) reads $\mathbf{y}_k = \mathbb{A}\mathbf{x}_k + \boldsymbol{\rho}(\mathbf{z}_k)$. In matrix notation, this is compactly written as $\mathbf{Y} = \mathbb{A}\mathbf{X} + R$, where $R(:,k) = \boldsymbol{\rho}(\mathbf{z}_k)$ and $\mathbb{A}$ is determined so that $\|R\|_F$ is minimal.

2.4.1 The DMD matrix

The DMD matrix $\mathbb{A}$ is thus defined as the solution of the least squares problem $\|\mathbf{Y} - A\mathbf{X}\|_F \to \min_A$. Clearly, if $\mathbf{X}^T$ has a nontrivial null-space, $\mathbb{A}$ is not unique; in that case we can choose B so that $B\mathbf{X} = \mathbf{0}$ and thus $(\mathbb{A} + B)\mathbf{X} = \mathbb{A}\mathbf{X}$. That is, adding to any row of $\mathbb{A}$ an arbitrary vector from the left null-space of $\mathbf{X}$ does not change the optimality. In fact, since $\mathbf{X}$ is assumed tall and skinny, it has high-dimensional left null-space (since $\mathrm{Ker}(\mathbf{X}^T) = \mathrm{Range}(\mathbf{X})^{\perp}$) and the least squares solution is not unique.

Independent of the choice of $\mathbb{A} \in \arg\min_A \|\mathbf{Y} - A\mathbf{X}\|_F$, it holds that $\mathbb{A}\mathbf{X} = \mathbf{Y}\mathbf{P}_{\mathbf{X}^T}$, where $\mathbf{P}_{\mathbf{X}^T}$ is the orthogonal projector onto the range of $\mathbf{X}^T$. In the DMD theory, the specifications for $\mathbb{A}$ is strenghtened with a constraint of minimality of $\|\mathbb{A}\|_F$, which yields $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$, expressed using the Moore-Penrose pseudoinverse $\mathbf{X}^{\dagger}$ of $\mathbf{X}$. As we discussed in Remark 2.1, the interpretability of such a constraint, besides ensuring unique least squares solution, is rather vague. In our opinion, the only information contained in the data is that $\mathbb{A}\mathbf{X} = \mathbf{Y}\mathbf{P}_{\mathbf{X}^T}$ so that $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$ is just a particular element in from the linear manifold

$$[\mathbb{A}] = \{\mathbf{Y}\mathbf{X}^{\dagger} + B \ : \ B\mathbf{X} = \mathbf{0}\}. \tag{21}$$

Using the particular choice $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$ can be useful in some estimates if one can exploit the fact that in that case $\|\mathbb{A}\|_F$ is minimal. This issue is further discussed in Remark 2.2, Remark 3.1 and §3.1.1.

2.4.2 Schmid's DMD method

The Schmid method is in essence a data driven Rayleigh-Ritz extraction of spectral information of $\mathbb{A}$, where the subspace used to compress $\mathbb{A}$ is defined using the leading left singular vectors of $\mathbf{X}$. Let $\mathbf{X} = U\Sigma V^*$ be the SVD of $\mathbf{X}$ with the singular values $\sigma_1 \geq \cdots \geq \sigma_m$, and let r be the rank of $\mathbf{X}$. Due to the Krylov seqence structure of its columns, it is expected that $\mathbf{X}$ is ill-conditioned. In addition, if $\mathbf{X}$ contains snapshots from several trajectories with different initial conditions and if the dynamics contains rapid changes, it is possible that the column norms of $\mathbf{X}$ span several orders of magnitude.

Let $U_k = U(:, 1:k)$, $V_k = V(:, 1:k)$, $\Sigma_k = \Sigma(1:k, 1:k)$, for $1 \leq k \leq r$. Now, since $\mathbf{X} = U_r\Sigma_r V_r^*$, $\mathbf{X}^{\dagger} = V_r\Sigma_r^{-1}U_r^*$, the formula $\mathbb{A}\mathbf{X} = \mathbf{Y}\mathbf{P}_{\mathbf{X}^T}$ implies $\mathbb{A}U_r\Sigma_r V_r^* = \mathbf{Y}V_rV_r^*$, then $\mathbb{A}U_r\Sigma_r = \mathbf{Y}V_r$, and it is easily checked that, with any $k \in \{1, \ldots, r\}$, $\mathbb{A}U_k = \mathbf{Y}V_k\Sigma_k^{-1}$. Hence, the Rayleigh quotient $S_k = U_k^*\mathbb{A}U_k$ with respect to the range of U_k can be expressed as

$$S_k = U_k^*\mathbf{Y}_m V_k \Sigma_k^{-1}, \tag{22}$$

which is suitable for data driven setting. Each eigenpair (λ_i, w_i) of S_k yields an approximate eigenpair $(\lambda_i, U_k w_i)$, i.e. $\mathbb{A}(U_k w_i) \approx \lambda_i(U_k w_i)$. Note that $\mathbb{A}U_k = U_k S_k + (I - U_k U_k^*)\mathbb{A}U_k$. If $S_k w_i = \lambda_i w_i$ with $\|w_i\|_2 = 1$, then $\mathbb{A}(U_k w_i) = \lambda_i(U_k w_i) + (\mathbb{I}_n - U_k U_k^*)\mathbb{A}U_k w_i$.

The index k is usually determined as

$$k = \max\{i \ : \ \sigma_i \geq \sigma_1 \boldsymbol{\tau}\}, \tag{23}$$

where the tolerance level is usually a multiple of the round-off unit $\boldsymbol{\varepsilon}$, e.g. $\boldsymbol{\tau} = n\boldsymbol{\varepsilon}$. We discuss this in detail in §3.2 and §3.3 from a point of view that is, to the best of our knowledge, new.

Remark 2.2 Note that the above computation of the Ritz pairs we never used $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$; instead we used that $\mathbb{A}\mathbf{X} = \mathbf{Y}V_k V_k^*$, which is equivalent to say that $\mathbb{A}$ is a solution to $\|\mathbf{Y} - \mathbb{A}\mathbf{X}\|_F \to \min$. The same S_k is obtained if we use SVD for the best rank k approximation $\mathbf{X} \approx U_k\Sigma_k V_k^*$, $\mathbf{X}^\dagger \approx V_k\Sigma_k^{-1}U_k^*$, and then use $\mathbb{A}_k = \mathbf{Y}V_k\Sigma_k^{-1}U_k^*$. Further, nothing is gained if we try to use the non-uniqueness and replace $\mathbb{A}$ with some $\widetilde{\mathbb{A}} = \mathbb{A} + B$ such that $B\mathbf{X} = \mathbf{0}$ (i.e. $\widetilde{\mathbb{A}} \in [\mathbb{A}]$). Then $BU_k = \mathbf{0}$, $\widetilde{\mathbb{A}}U_k = \mathbb{A}U_k = \mathbf{Y}V_k\Sigma_k^{-1}$ and $\widetilde{S}_k = U_k^*\widetilde{\mathbb{A}}U_k = U_k^*\mathbb{A}U_k = S_k$.

Algorithm 1 $[Z_k, \Lambda_k] = \mathrm{DMD}(\mathbf{X}, \mathbf{Y}, \boldsymbol{\tau})$

```
Input:
    $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m), \mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m) \in \mathbb{C}^{n\times m}$ that define a sequence of snapshots
    pairs $(\mathbf{x}_i, \mathbf{y}_i)$. (Tacit assumption is that $n$ is large and that $m \ll n$.)
    Tolerance $\boldsymbol{\tau}$ for the truncation (23).
1: $[U, \Sigma, V] = svd(\mathbf{X})$ ; {The thin SVD: $\mathbf{X} = U\Sigma V^*$, $U \in \mathbb{C}^{n\times m}$, $\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^m$, $V \in \mathbb{C}^{m\times m}$.}
2: Determine numerical rank $k$ using (23) with the threshold $\boldsymbol{\tau}$.
3: Set $U_k = U(:, 1:k)$, $V_k = V(:, 1:k)$, $\Sigma_k = \Sigma(1:k, 1:k)$
4: $S_k = ((U_k^*\mathbf{Y}_m)V_k)\Sigma_k^{-1}$; {Schmid's formula for the Rayleigh quotient $U_k^*\mathbb{A}U_k$.}
5: $[W_k, \Lambda_k] = \mathrm{eig}(S_k)$ {$\Lambda_k = \mathrm{diag}(\lambda_i)_{i=1}^k$; $S_k W_k(:, i) = \lambda_i W_k(:, i)$; $\|W_k(:, i)\|_2 = 1$.}
6: $Z_k = U_k W_k$ {Ritz vectors.}
Output: $Z_k$, $\Lambda_k$
```

2.4.3 The main tasks of the DMD

In applications of the Dynamic Mode Decomposition, the two main computational tasks are

1. Identify approximate eigenpairs (λ_j, z_j) such that

$$\mathbb{A}\mathbf{z}_j \approx \lambda_j \mathbf{z}_j, \;\; \lambda_j = |\lambda_j| e^{\mathrm{i}\omega_j \Delta t}, \;\; j = 1, \ldots, k; \;\; k \le m. \tag{24}$$

2. Derive a spectral spatio–temporal representation of the snapshots $\mathbf{f}_i$:

$$\mathbf{f}_i \approx \sum_{j=1}^{\ell} \mathbf{z}_{\varsigma_j}\alpha_j \lambda_{\varsigma_j}^{i-1} \equiv \sum_{j=1}^{\ell} \mathbf{z}_{\varsigma_j}\alpha_j |\lambda_{\varsigma_j}|^{i-1} e^{\mathrm{i}\omega_{\varsigma_j}(i-1)\Delta t}, \;\; i = 1, \ldots, m. \tag{25}$$

This involves solving the least squares problem

$$\sum_{i=1}^{m} w_i^2 \|\mathbf{f}_i - \sum_{j=1}^{\ell} \mathbf{z}_{\varsigma_j}\alpha_j \lambda_{\varsigma_j}^{i-1}\|_2^2 \longrightarrow \min_{\alpha_j}, \tag{26}$$

for some suitable selection of the modes $\mathbf{z}_{\varsigma_j}$ and weights $w_i \ge 0$.

In the rest of the paper we describe the details of the proposed solution to the task 1. The software is designed so that the subroutines return the results in a way that allows for an efficient software solution to the task 2.

3 Discussion, numerical analysis and modifications

Clearly, using the truncated SVD and compressing $\mathbb{A}$ to the range of U_k means only partial usage of the information in the data, and we should use as large k as numerically feasible. We will see later (see e.g. examples in §3.5) that the choice of k and the quality of the results depend on the accuracy of the SVD.

In general, not all Ritz pairs computed in Algorithm 1 are satisfactory good approximations of some eigenpairs of $\mathbb{A}$. Since we do not have access to $\mathbb{A}$, we need a way to assess the errors; this is discussed in §3.1. In §3.1.1 we discuss and clarify some issues related to the Exact DMD [52]. The value of k, often related with the numerical rank of $\mathbf{X}$ is usually determined by inspecting the singular values of $\mathbf{X}$, and in practical computation it is determined as in (23), where $\boldsymbol{\tau}$ is user supplied tolerance. The rationale behind this is discussed in §3.2 and in §3.3, §3.4 we explore possibilities for more numerical robustness in apparently ill-conditioned cases. Here we adopt and extend the discussions and analysis from [22], [17]. Numerical examples in §3.5 illustrate the theory and confirm that the analysis is sharp.

3.1 Residuals and the refined Ritz vectors

A way to test the quality of the approximation $\mathbb{A}(U_k w_i) \approx \lambda_i (U_k w_i)$, where $w_i = W_k(:,i)$, is to compute the norm of the residual $r_i = \mathbb{A}(U_k w_i) - \lambda_i(U_k w_i)$. Even though we do not have access to $\mathbb{A}$, the residuals can be computed in the data driven framework as

$$r_i = (\mathbb{I}_n - U_k U_k^*)\mathbb{A}(U_k w_i) = \mathbf{Y} V_k \Sigma_k^{-1} w_i - \lambda_i (U_k w_i). \tag{27}$$

This formula is first introduced and successfully used in [22]. Note that it is independent of the choice of $\widetilde{\mathbb{A}} \in [\mathbb{A}]$, so using it is in compliance with the discussion in §2.4.1 and Remark 2.2. In §3.4 we discuss numerical issues related to the usage of (27) in floating point arithmetic.

Remark 3.1 It should be clear from the very beginning that we may expect good spectral approximation (small residuals) only if it is warranted by the data, i.e. if the range of $\mathbf{X}$ contains a nearly $\mathbb{A}$-invariant subspace. Here we see another argument that we should not use $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$, except possibly in some estimates as mentioned in §2.4.1. Namely, in that case we would have $\mathbb{A}\mathbf{Y} = \mathbf{Y}(\mathbf{X}^\dagger \mathbf{Y})$ (information which is not contained in the data) and this would mean that the range of $\mathbf{Y}$ is $\mathbb{A}$-invariant and we could recover all eigenvalues and eigenvectors of $\mathbb{A}$ exactly (with zero residuals, barring rounding errors of the computer arithmetic) but under the false premise that $\mathbb{A}\mathbf{Y}$ is given in the data.

3.1.1 A remark on the Exact Dynamic Mode Decomposition (Exact DMD)

A variant of the DMD, proposed in [52, §2.2, §2.3] and designated as the Exact Dynamic Mode Decomposition (Exact DMD) is entirely built on the computation of exact eigenvalues and eigenvectors of $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$. Since $\mathbf{Y}$ is $\mathbb{A}$-invariant this is possible. The algorithm ([52, Algorithm 2]) follows the lines 1.–5. of Algorithm 1 and in the last step, instead of $Z_k = U_k W_k$, for a computed nonzero eigenvalue λ_i, the corresponding eigenvector is returned as $Z_k^{(ex)}(:,i) = (1/\lambda_i)\mathbf{Y} V_k \Sigma_k^{-1} W_k(:,i)$.

Note that $Z_k^{(ex)}(:,i) = (1/\lambda_i)\mathbb{A} U_k W_k(:,i) = (1/\lambda_i)\mathbb{A} Z_k(:,i)$ and this modification can be understood/interpreted using the following fact: If v is a unit eigenvector belonging to a nonzero eigenvalue μ of a matrix M, then $\widetilde{v} = (1/\mu)Mv = v$. If v is only an approximate eigenvector, then Mv is one step of the power method that may contribute (without guarantee) to improving v in the direction of the dominant eigenvalue.

Proposition 3.1 *The output of the Exact DMD is independent of the particular choice $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$, and it is the same for any $\widetilde{\mathbb{A}} \in [\mathbb{A}] = \arg\min_A \|\mathbf{Y} - A\mathbf{X}\|_F$. The exactness of the computed spectral information (barring finite precision limitations) holds only for $\mathbb{A}$.*

To see this, note that for any $\widetilde{\mathbb{A}} \in [\mathbb{A}]$ (see Remark 2.2)

$$Z_k^{(ex)}(:,i) = (1/\lambda_i)\mathbf{Y}V_k\Sigma_k^{-1}W_k(:,i) = (1/\lambda_i)\mathbb{A}U_kW_k(:,i) = (1/\lambda_i)\widetilde{\mathbb{A}}U_kW_k(:,i).$$

Also, note that the vectors $\lambda_i Z_k^{(ex)}(:,i)$, $i = 1,\ldots,k$ are computed if the residuals (27) for the pairs $(\lambda_i, U_kW_k(:,i))$ are requested.

Remark 3.2 The choice of scaling by $1/\lambda_i$ in the definition of $Z_k^{(ex)}(:,i)$ does not make $Z_k^{(ex)}(:,i)$ unit vector. Indeed,

$$U_kU_k^*\mathbf{Y}V_k\Sigma_k^{-1}W_k(:,i) = U_kS_kW_k(:,i) = \lambda_iU_kW_k(:,i), \quad \|W_k(:,i)\|_2 = 1,$$

so that $|\lambda_i|$ is the norm of the orthogonal projection of $\mathbf{Y}V_k\Sigma_k^{-1}W_k(:,i)$ onto the range of U_k. Hence, $\|Z_k^{(ex)}(:,i)\|_2 \geq 1$, and this should be taken into account in the latter use of $Z_k^{(ex)}(:,i)$, e.g. when computing the residuals or in the modal analysis of the data snapshots.

3.2 Computation of the SVD

The numerical computation of the SVD $\mathbf{X} \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$ is, strictly speaking, *mixed stable*: a slightly changed computed decomposition is an exact SVD of $\mathbf{X} + \delta\mathbf{X}$ with small $\delta\mathbf{X}$. More precisely, there exist unitary matrices $\widehat{U} = \widetilde{U} + \delta\widetilde{U}$, $\widehat{V} = \widetilde{V} + \delta\widetilde{V}$, and a perturbation $\delta\mathbf{X}$ (*backward error*) such that $\|\widehat{U} - \widetilde{U}\|_2 \leq \epsilon_u$, $\|\widehat{V} - \widetilde{V}\|_2 \leq \epsilon_v$, and

$$\mathbf{X} + \delta\mathbf{X} = \widehat{U}\widetilde{\Sigma}\widehat{V}^*, \quad \|\delta\mathbf{X}\|_2 \leq \epsilon\|\mathbf{X}\|_2. \tag{28}$$

Here $\epsilon_u, \epsilon_v, \epsilon$ depend on the details of a particular algorithm and its software implementation. This kind of backward error is typical for the QR SVD and the divide an conquer SVD algorithms, implemented in LAPACK in `xGESVD` and `xGESDD`, respectively. In Matlab, based on some numerical experiments, it seems that the function `svd` uses `xGESVD` if only the singular values are requested, and `xGESDD` otherwise.

Usually, we estimate $\|\delta\mathbf{X}\|_F$ relative to $\|\mathbf{X}\|_F$ ($\|\delta\mathbf{X}\|_F \leq \epsilon_F\|\mathbf{X}\|_F$), and the worst case bounds are modestly growing polynomials in matrix dimensions times the round-off unit $\boldsymbol{\varepsilon}$. The Frobenius norm is more practical in a technically tedious error analysis, and the spectral norm is in some cases more elegant for theoretical study. Since $\|\mathbf{X}\|_2 \leq \|\mathbf{X}\|_F \leq \sqrt{m}\|\mathbf{X}\|_2$, one can easily write the results in either norm.

Because in finite precision arithmetic we cannot compute exactly unitary matrices, $\widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$ is in general not an exact SVD of any matrix, but changes of the order of roundoff in $\widetilde{U}$ and $\widetilde{V}$ will establish an exact SVD (28) of $\mathbf{X} + \delta\mathbf{X}$. Hence, the computed singular values $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_m$ are the exact singular values of $\mathbf{X} + \delta\mathbf{X}$. How much can they differ from the exact singular values of $\mathbf{X}$?

Theorem 3.3 *(Weyl's theorem) Let the singular values of $\mathbf{X}$ and $\mathbf{X} + \delta\mathbf{X}$ be $\sigma_1 \geq \cdots \geq \sigma_m$ and $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_m$, respectively. Then $\max_i |\widetilde{\sigma}_i - \sigma_i| \leq \|\delta\mathbf{X}\|_2$.*

In (28), we have $\widetilde{\Sigma} = \mathrm{diag}(\widetilde{\sigma}_i)_{i=1}^m$ and each computed singular value $\widetilde{\sigma}_i = \sigma_i + \delta\sigma_i$ satisfies

$$|\delta\sigma_i| \leq \|\delta\mathbf{X}\|_2 \leq \epsilon\|\mathbf{X}\|_2 = \epsilon\sigma_1. \tag{29}$$

If we want to estimate the relative error in a $\sigma_i \neq 0$, then

$$\frac{|\widetilde{\sigma}_i-\sigma_i|}{\sigma_i} \le \epsilon\frac{\sigma_1}{\sigma_i} \le \epsilon\|\mathbf{X}\|_2\|\mathbf{X}^\dagger\|_2 \equiv \epsilon\kappa_2(\mathbf{X}),\quad \frac{|\delta\sigma_i|}{\widetilde{\sigma}_i} \le \epsilon\frac{\|\mathbf{X}\|_2}{\widetilde{\sigma}_i} \approx \epsilon\frac{\widetilde{\sigma}_1}{\widetilde{\sigma}_i}. \tag{30}$$

(Here $\kappa_2(\mathbf{X}) = \|\mathbf{X}\|_2\|\mathbf{X}^\dagger\|_2$ is the condition number.) Hence, if $\widetilde{\sigma}_i < \boldsymbol{\varepsilon}\widetilde{\sigma}_1$ we cannot guarantee any accuracy of $\widetilde{\sigma}_i$, and using its inverse in the formula (22) is not advisable. In fact, since the extremely small singular values may be computed with an upward bias, if we have $\widetilde{\sigma}_i < \boldsymbol{\varepsilon}\widetilde{\sigma}_1$ then the true singular value σ_i may be much (possibly many orders of magnitude) smaller than $\widetilde{\sigma}_i$.

3.2.1 On errors in the matrices used in the DMD

To understand the structure of the error, consider the following Gedankenexperiment. Assume that $\delta\mathbf{X}$ is the only error so that $\widetilde{U} = \widehat{U}$ and $\widetilde{V} = \widehat{V}$ are exactly unitary and $\mathbf{X} + \delta\mathbf{X} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$ is the exact SVD. Assume further that $\mathbf{X} = U\Sigma V^*$ is of full column rank. If we use $\widetilde{U}$, $\widetilde{\Sigma}$, $\widetilde{V}$ as the ingredients for the next step of the algorithm and compute $\widetilde{S}_k = (\widetilde{U}_k^*\mathbf{Y})\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$, we accept changing $\mathbf{X}$ to $\widetilde{\mathbf{X}} = \mathbf{X} + \delta\mathbf{X}$ and computing the DMD matrix for the data $(\mathbf{X} + \delta\mathbf{X}, \mathbf{Y})$. This means that we use

$$\mathbb{A}\widetilde{U}\widetilde{\Sigma}\widetilde{V}^* \approx \mathbf{Y} \Longrightarrow \mathbb{A}\widetilde{U}_k \approx \mathbf{Y}\widetilde{V}_k\widetilde{\Sigma_k}^{-1} \Longrightarrow \widetilde{U}_k^*\mathbb{A}\widetilde{U}_k \approx (\widetilde{U}_k^*\mathbf{Y})\widetilde{V}_k\widetilde{\Sigma}_k^{-1} = \widetilde{S}_k.$$

On the other hand, the functional relationship between $\mathbf{X}$ and $\mathbf{Y}$ is lost and we have

$$\mathbb{A}\widetilde{U}\widetilde{\Sigma}\widetilde{V}^* = \mathbf{Y} + \mathbb{A}\delta\mathbf{X} \Longrightarrow \mathbb{A}\widetilde{U}_k = \mathbf{Y}\widetilde{V}_k\widetilde{\Sigma}_k^{-1} + \mathbb{A}\delta\mathbf{X}\widetilde{V}_k\widetilde{\Sigma}_k^{-1} \Longrightarrow \widetilde{U}_k^*\mathbb{A}\widetilde{U}_k = \widetilde{S}_k + \widetilde{U}_k^*\mathbb{A}\delta\mathbf{X}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}.$$

Set $\delta\mathbf{Y} = \mathbb{A}\delta\mathbf{X}$, $E_k = \mathbb{A}\delta\mathbf{X}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$, $\delta\widetilde{S}_k = \widetilde{U}_k^*E_k$. If (28) is the only information on the size of $\delta\mathbf{X}$, then $\|\delta\mathbf{Y}\|_2 \le \epsilon\|\mathbb{A}\|_2\|\mathbf{X}\|_2$, and if we specify $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$ then

$$\|\delta\mathbf{Y}\|_2 = \|\mathbf{Y}\mathbf{X}^\dagger\delta\mathbf{X}\|_2 \le \|\mathbf{Y}\|_2\epsilon\|\mathbf{X}^\dagger\|_2\|\mathbf{X}\|_2 = \epsilon\kappa_2(\mathbf{X})\|\mathbf{Y}\|_2. \tag{31}$$

Further,

$$\|E_k\|_2 \le \|\mathbb{A}\|_2\epsilon\frac{\|\mathbf{X}\|_2}{\widetilde{\sigma}_k} \le \|\mathbb{A}\|_2\frac{\epsilon}{1-\epsilon}\frac{\widetilde{\sigma}_1}{\widetilde{\sigma}_k},\quad \|\delta\widetilde{S}_k\|_2 \le \|E_k\|_2 \le \|\mathbb{A}\|_2\frac{\epsilon}{1-\epsilon}\frac{\widetilde{\sigma}_1}{\widetilde{\sigma}_k}.$$

Note here that, since $k \ll n$, $\delta S_k = \widetilde{U}_k^*E_k \in \mathbb{C}^{k\times k}$ is potentially much smaller that $E_k \in \mathbb{C}^{n\times k}$.

Example 3.4 *In this synthetic example*[5] *we illustrate the relation* $\mathbb{A}\widetilde{U}_k \approx \mathbf{Y}\widetilde{V}_k\widetilde{\Sigma_k}^{-1}$*, using explicitly the matrix* $\mathbb{A}$ *that generated the data. In Figure 1, we show column-wise relative errors* $\|\mathbb{A}\widetilde{U}_k(:,i) - \mathbf{Y}\widetilde{V}_k\widetilde{\Sigma_k}^{-1}(:,i)\|_2/\|\mathbb{A}\widetilde{U}_k(:,i)\|_2$*. Note how the larger errors in the smallest singular values degrade the accuracy with the increased index* i*.*

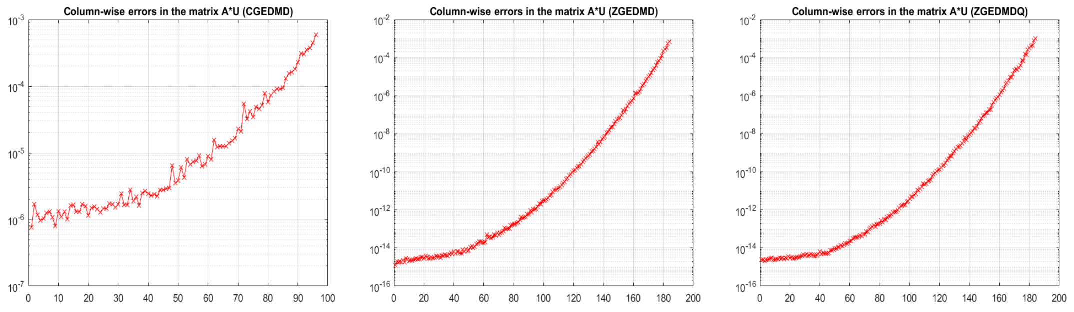

Figure 1: **Example 3.4** The relative errors $\|\mathbb{A}\widetilde{U}_k(:,i) - \mathbf{Y}\widetilde{V}_k\widetilde{\Sigma_k}^{-1}(:,i)\|_2/\|\mathbb{A}\widetilde{U}_k(:,i)\|_2$.

[5]See §3.5 for details how we generate synthetic examples.

Without going into further details, it is clear that the validity of the formulas used in any software implementation of the DMD depends on the accuracy of the computed SVD. This motivates the discussion in §3.3 and §3.4.

3.3 Computation with snapshot-wise small backward error

The backward stability of the SVD computation (28) is a necessary requirement for a SVD algorithm. In the DMD context, where the columns of $\mathbf{X}$ are data snapshots this may not be good enough. Namely, if the snapshots vary in norm, the backward error (28) does not guarantee that the backward error $\delta\mathbf{X}(:,i)$ in a particular column $\mathbf{X}(:,i)$ is small relative to $\|\mathbf{X}(:,i)\|_2$. Since $\mathbf{X}$ may contain snapshots from several trajectories with different initial conditions and with possible rapid changes in the dynamics, in general we could have $\mathbf{X}$ with the column norms that span several orders of magnitude. If we can only guarantee that $\|\delta\mathbf{X}\|_2 \leq \epsilon\|\mathbf{X}\|_2$, then we cannot say that the results computed in finite precision arithmetic correspond to slightly changed (individual) data snapshots.

However, for a robust software implementation of the DMD that can handle different data scenarios, this snapshot-wise backward stability is certainly a desirable feature. In this section, we explore possibilities for such strong form of backward stability. Further, we are also interested in computing more singular values, even the smallest ones, with satisfactory accuracy.

First, we recall that the SVD of $\mathbf{X}$ can be computed with a backward error $\delta\mathbf{X}$ that can be estimated as

$$\|\delta\mathbf{X}(:,i)\|_2 \leq \epsilon_c\|\mathbf{X}(:,i)\|_2, \quad i = 1,\ldots,m, \tag{32}$$

where ϵ_c is bounded, similarly as ϵ, by the round-off times a factor of the dimensions. This more structured backward error, that preserves small columns of $\mathbf{X}$, is ensured in a preconditioned Jacobi SVD algorithm [19], [20], that is implemented in the LAPACK subroutine `xGEJSV`.

Assume that $\widetilde{\mathbf{X}} = \mathbf{X} + \delta\mathbf{X}$ remains of full column rank. As before, we asume that $\delta\mathbf{X}$ is the only error and that $\widetilde{\mathbf{X}} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$ is the exact SVD. Otherwise, we have $\mathbf{X}+\delta\mathbf{X} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^* = \widehat{U}(I+E_1)\widehat{\Sigma}(I+E_2)\widehat{V}^*$ with small $\|E_1\|_2$, $\|E_2\|_2$ and we have to deal with more technical details that are not important for the main thread of the discussion in this section. For those details of the analysis, we refer to [19], [20], [14].

To see the difference in the accuracy of the computed singular values, we use the following theorem:

Theorem 3.5 *(Eisenstat and Ipsen, [24]) Let $\sigma_1 \geq \cdots \geq \sigma_n$ and $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_n$ be the singular values of $\mathbf{X}$ and $\mathbf{X}+\delta\mathbf{X}$, respectively. Assume that $\mathbf{X}+\delta\mathbf{X}$ can be written in the form of multiplicative perturbation $\mathbf{X}+\delta\mathbf{X} = \Xi_1\mathbf{X}\Xi_2$ and let $\xi = \max\{\|\Xi_1\Xi_1^T - \mathbb{I}\|_2, \|\Xi_2^T\Xi_2 - \mathbb{I}\|_2\}$. Then*

$$|\widetilde{\sigma}_i - \sigma_i| \leq \xi\sigma_i, \quad i = 1,\ldots,n.$$

In our case, since we assume full column rank of $\mathbf{X}$, the additive perturbation $\delta\mathbf{X}$ can be recast into a multiplicative one as $\mathbf{X}+\delta\mathbf{X} = (\mathbb{I}+\delta\mathbf{X}\mathbf{X}^\dagger)\mathbf{X}$, with $\Xi_1 = \mathbb{I}+\delta\mathbf{X}\mathbf{X}^\dagger$, $\Xi_2 = \mathbb{I}$. It is easily estimated that

$$\max\{\|\Xi_1\Xi_1^T - \mathbb{I}\|_2, \|\Xi_2^T\Xi_2 - \mathbb{I}\|_2\} \leq 2\|\delta\mathbf{X}\mathbf{X}^\dagger\|_2 + \|\delta\mathbf{X}\mathbf{X}^\dagger\|_2^2$$

and thus

$$\max_i \frac{|\sigma_i - \widetilde{\sigma}_i|}{\sigma_i} \leq 2\|\delta\mathbf{X}\mathbf{X}^\dagger\|_2 + \|\delta\mathbf{X}\mathbf{X}^\dagger\|_2^2 = \widehat{\xi}. \tag{33}$$

Let $D_{\mathbf{X}} = \mathrm{diag}(\|\mathbf{X}(:,i)\|_2)_{i=1}^m$, $\mathbf{X}_c = \mathbf{X}D_{\mathbf{X}}^{-1}$, $\widetilde{\mathbf{X}}_c = \widetilde{\mathbf{X}}D_{\mathbf{X}}^{-1} = \mathbf{X}_c + \delta\mathbf{X}_c$, where

$$\delta\mathbf{X}_c = \begin{pmatrix} \frac{\delta\mathbf{X}(:,1)}{(D_{\mathbf{X}})_{11}} & \dots & \frac{\delta\mathbf{X}(:,m)}{(D_{\mathbf{X}})_{mm}} \end{pmatrix}, \ \max_i \|\delta\mathbf{X}_c(:,i)\|_2 \leq \epsilon_c, \ \ \|\delta\mathbf{X}_c\|_F \leq \sqrt{m}\epsilon_c.$$

This estimate of $\delta\mathbf{X}_c$, that follows from (32), is not possible if $\delta\mathbf{X}$ is bounded only as in (28). Now in (33) we can futher estimate

$$\|\delta\mathbf{X}\mathbf{X}^\dagger\|_2 = \|\delta\mathbf{X}_c\mathbf{X}_c^\dagger\|_2 \leq \frac{\|\delta\mathbf{X}_c\|_2}{\|\mathbf{X}_c\|_2}\kappa_2(\mathbf{X}_c) \leq \sqrt{m}\epsilon_c\kappa_2(\mathbf{X}_c)$$

and $\widehat{\xi} \leq 2\sqrt{m}\epsilon_c\|\mathbf{X}_c^\dagger\|_2 + m(\epsilon_c\|\mathbf{X}_c^\dagger\|_2)^2$, i.e.

$$\frac{|\widetilde{\sigma}_i - \sigma_i|}{\sigma_i} \leq 2\sqrt{m}\epsilon_c\|\mathbf{X}_c^\dagger\|_2 + m(\epsilon_c\|\mathbf{X}_c^\dagger\|_2)^2. \tag{34}$$

This bound is better than (30) because the condition number $\kappa_2(\mathbf{X}_c)$ behaves better than $\kappa_2(\mathbf{X})$ – it can be much smaller than $\kappa_2(\mathbf{X})$ and it is never much bigger. This is a consequence of the following theorem.

Theorem 3.6 *(Van der Sluis [53]) Let $\mathbf{X} \in \mathbb{C}^{n\times m}$ be of full column rank and let $D_{\mathbf{X}} = \mathrm{diag}(\|\mathbf{X}(:,i)\|_2)_{i=1}^m$ and $\mathbf{X}_c = \mathbf{X}D_{\mathbf{X}}^{-1}$. Then $\kappa_2(\mathbf{X}_c) \leq \sqrt{m}\min_{D=\mathrm{diag}}\kappa_2(\mathbf{X}D)$.*

Remark 3.7 The backward error (28) with the estimated accuracy of the computed singular values (30) is typical for the LAPACK SVD driver subroutines xGESVD and xGESDD. The backward error (32) with the error in the singular values bounded as in (34) is guaranteed in the LAPACK SVD driver subroutine xGEJSV.

Remark 3.8 The better accuracy of the Jacobi method is analyzed in [15], [14], [19], [20] and it is well understood. The LAPACK subroutine xGESVDQ, proposed in [21] in most cases achieves the accuracy (34), but it has somewhat weaker theoretical understanding.

3.4 How this impacts the DMD

We now revisit the error analysis from §3.2.1, but with the assumption that the SVD of $\mathbf{X}$ is computed with the backward error as in (32). Under the additional assumption that $\|\delta\mathbf{X}\mathbf{X}^\dagger\|_2 < 1$, $\widetilde{\mathbf{X}}$ is of full column rank and for any $k \in \{1,\dots,n\}$ we have

$$\|E_k\|_2 \leq \|\mathbb{A}\delta\mathbf{X}(\widetilde{\mathbf{X}}^\dagger\widetilde{\mathbf{X}})\widetilde{V}\widetilde{\Sigma}^{-1}\|_2 \leq \|\mathbb{A}\|_2\|\delta\mathbf{X}\widetilde{\mathbf{X}}^\dagger\|_2.$$

Using the diagonal scaling matrix $D_{\mathbf{X}}$, we can estimate

$$\|\delta\mathbf{X}D_{\mathbf{X}}^{-1}D_{\mathbf{X}}\widetilde{\mathbf{X}}^\dagger\|_F \leq \|\delta\mathbf{X}_c\|_F\|\widetilde{\mathbf{X}}_c^\dagger\|_2 \leq \sqrt{m}\epsilon_c\|\widetilde{\mathbf{X}}_c^\dagger\|_2. \tag{35}$$

The key for this estimate is (32)

$$\begin{aligned}\sigma_{\min}(\widetilde{\mathbf{X}}_c) &= \min_{v\neq\mathbf{0}}\frac{\|(\mathbb{I}+\delta\mathbf{X}_c\mathbf{X}_c^\dagger)\mathbf{X}_c v\|_2}{\|v\|_2} \geq \min_{v\neq\mathbf{0}}\frac{\sigma_{\min}(\mathbb{I}+\delta\mathbf{X}_c\mathbf{X}_c^\dagger)\|\mathbf{X}_c v\|_2}{\|v\|_2} \\ &\geq (1-\|\delta\mathbf{X}_c\mathbf{X}_c^\dagger\|_2)\min_{v\neq\mathbf{0}}\frac{\|\mathbf{X}_c\|_2}{\|v\|_2} = (1-\|\delta\mathbf{X}_c\mathbf{X}_c^\dagger\|_2)\sigma_{\min}(\mathbf{X}_c).\end{aligned}$$

Thus

$$\|E_k\|_2 \leq \|\mathbb{A}\|_2\frac{\|\delta\mathbf{X}_c\|_2}{\|\mathbf{X}_c\|_2}\frac{\kappa_2(\mathbf{X}_c^\dagger)}{1-\|\delta\mathbf{X}_c\mathbf{X}_c^\dagger\|_2}, \tag{36}$$

and similarly

$$\|E_k\|_F \leq \|\mathbb{A}\|_2\frac{\|\delta\mathbf{X}_c\|_F}{\|\mathbf{X}_c\|_F}\frac{\kappa_{2,F}(\mathbf{X}_c^\dagger)}{1-\|\delta\mathbf{X}_c\mathbf{X}_c^\dagger\|_2}, \ \ \kappa_{2,F}(\mathbf{X}_c^\dagger) = \|\mathbf{X}_c^\dagger\|_2\|\mathbf{X}_c\|_F. \tag{37}$$

Remark 3.9 The above bound is not optimal because on the right-hand side we have $1/\sigma_{\min}(\mathbf{X}_c)$ for all k. Of course, if $1/\sigma_{\min}(\mathbf{X}_c)$ is moderate, then we can take the maximal k. It would be ideal to have potentially much smaller factor $1/\sigma_k(\mathbf{X}_c)$, but we were not able to derive such a bound. Fortunately, a simple modification introduced in [22] and outlined in §3.6 below will make that possible.

Remark 3.10 With initial noise in the data and numerically induced perturbations of the type (32), instead of $\|\delta\mathbf{X}\|_2 \leq \epsilon\|\mathbf{X}\|_2$ as in (28), there is a better chance to apply the shadowing theory [39], [40]. Such an analysis is out of scope of this note and we leave the theme of shadowing theory of the DMD (developed in the framework of the Koopman operator theory) as a challenging open problem for the future work.

3.5 Numerical examples

The simplest way to test the above theory is to generate random data so that for the purpose of the test we can perform operations that are not feasible in a real application. To that end, we set $n = 2000$, $m = 400$ and use `DLATMR` to generate a real double precision $n \times n$ matrix $\mathbf{A}$ and then we scale $\mathbf{A}$ to control the spectral radius, which is a simple way to generate different levels of numerical difficulties. Then, starting with an initial randomly generated $\mathbf{f}_1$, we build the Krylov sequence $\mathbf{f}_{i+1} = \mathbf{A}\mathbf{f}_i$, $i = 1, \ldots, m$. The input data is $\mathbb{F} = \big(\mathbf{f}_1, \ldots, \mathbf{f}_{m+1}\big)$, i.e. $\mathbf{X} = \big(\mathbf{f}_1, \ldots, \mathbf{f}_m\big)$, $\mathbf{Y} = \big(\mathbf{f}_2, \ldots, \mathbf{f}_{m+1}\big)$. The tested methods have no access to $\mathbf{A}$.

The DMD for real data in IEEE double precision is implemented in a subroutine designated as `DGEDMD`, that is designed following [22] and using the LAPACK library; see Algorithm 2 and §4 for more details. In the three examples in this subsection, the computation is by suitable job parameters set to be as in Algorithm 1, and we optionally can choose four different LAPACK driver routines to compute the SVD: `DGESVD`, `DGESDD`, `DGESVDQ` and `DGEJSV`. In addition, we request that `DGEDMD` returns the norms of the residuals of all computed Ritz pairs, using (27). Further, for the purpose of numerical case study, in addition to the truncation strategy (23) we also test the case with $k = m$.

Since (for the purpose of the test) we know $\mathbf{A}$, we have its eigenvalues, computed using `DGEEV`, and for each Ritz pair returned by `DGEDMD`, we can compute the residuals explicitly, using $\mathbf{A}$, and validate the formula (27). For the purpose of graphical presentation of the results, the computed Ritz pairs are indexed so that the corresponding residuals (as returned by `DGEDMD`) are non-decreasing.[6]

Example 3.11 In the first test, the `DGEDMD` runs four times with the SVD computed using, respectively, `DGESVD`, `DGESDD`, `DGESVDQ`, `DGEJSV`. In all four cases, the results are nearly identical; k is determined (using (23)) as $k = 50$ and 27 Ritz pairs are selected using the residual threshold 10^{-2}. In Figure 2 we show the results for `DGESVD`; the graphs for the other three cases are visually indistinguishable from the ones given in Figure 2.

[6]The apparent staircase-like graph of the residuals is due to the complex conjugate Ritz pairs, that have the same residual. Also note that computing residuals involves cancellations so that we do not expect matching the residuals to high relative accuracy – such an accuracy in this context is not needed, we only need the right order of magnitude.

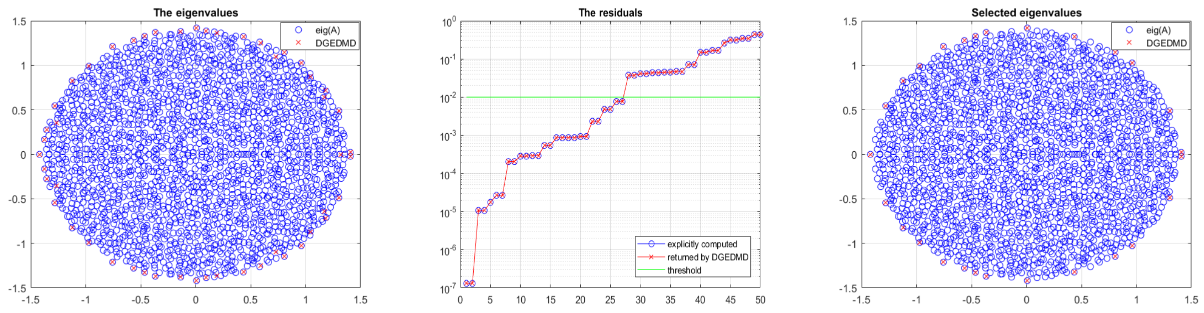

Figure 2: **Example 3.11** (DGEDMD using DGESVD) Left panel: the 50 computed Ritz values (x) and the eigenvalues of $\mathbb{A}$ computed by DGEEV (o). Middle panel: the data driven residuals computed by the subroutine (x) and the residuals computed explicitly using $\mathbf{A}$ (o). Right panel: The 27 selected eigenvalues, based on the residuals below the threshold 10^{-2}. The SVD is computed using DGESVD and k is determined using (23).

Example 3.12 In the second test, we switch the truncation device off, i.e. we enforce computation of all 400 Ritz pairs. We run DGEDMD twice, with the SVD of $\mathbf{X}$ computed using DGESVD and then using DGESDD. The results are shown in Figure 3, where the panels in the first row correspond to DGESVD and in the second to DGESDD. In both cases, the data driven residuals have large errors, severe underestimates resulting in selection of all Ritz pairs as acceptable for the given threshold value 10^{-2}. Many Ritz values are computed in clusters around zero. In this example $\kappa_2(\mathbf{X}) > 10^{60}$ and the computation is done in IEEE double precision arithmetic with $\boldsymbol{\varepsilon} \approx O(10^{-16})$, so according to the analysis in §3.2, we do not expect accurate results.

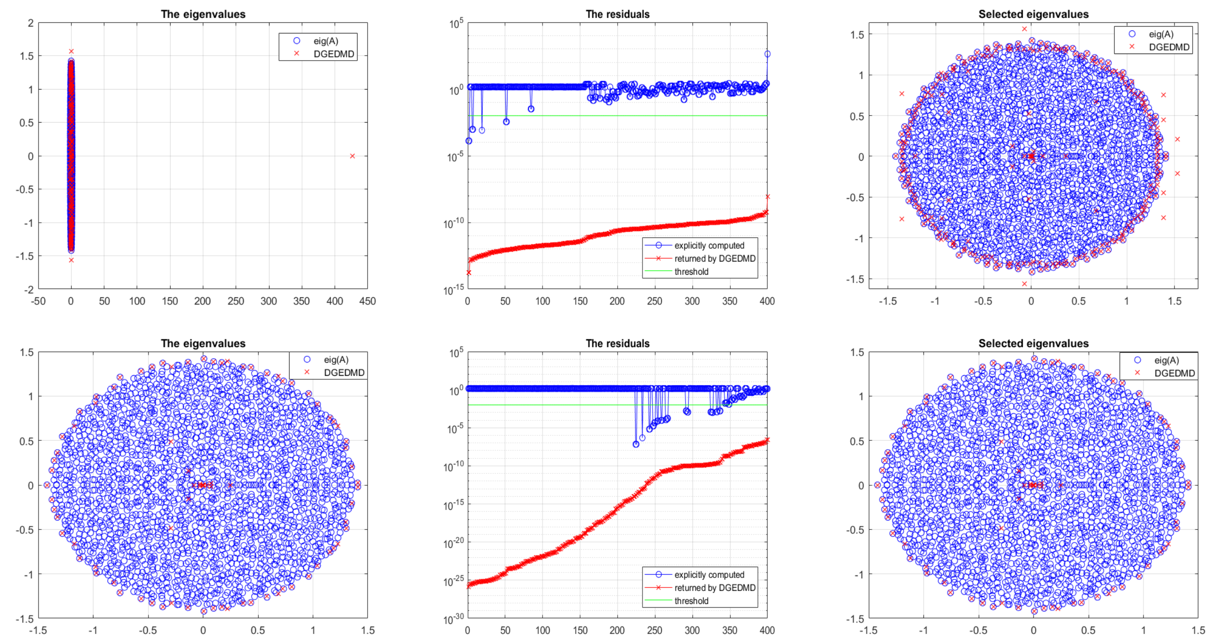

Figure 3: **Example 3.12** (DGEDMD: first row using DGESVD, second row using DGESDD) Left panels: the computed Ritz values (x) and the eigenvalues of **A** computed by DGEEV (o). Middle panels: the data driven residuals computed by the subroutines (x) and the residuals computed explicitly using **A** (o). Right panels: The 400 selected eigenvalues, based on the residuals below the threshold 10^{-2}. (In the case of DGESVD, for better visual inspection, we zoomed in on the disc that contains most of the eigenvalues, so that the real eigenvalue from the interval $[400, 450]$ is not seen in the figure, although it is selected as well.) In both cases, many Ritz values are computed in a cluster in the vicinity of the origin. Compare with Figure 2 and Figure 4.

Example 3.13 Now, we repeat the computation with the same data, again enforcing computation of all 400 Ritz pairs, but using DGESVDQ and DGEJSV. The results are shown in Figure 4; the panels in the first row correspond to DGESVDQ and in the second to DGEJSV. The advantage of computation with the accuracy described in §3.3 is obvious. It is instructive to compare these results with Figure 2 and Figure 3.

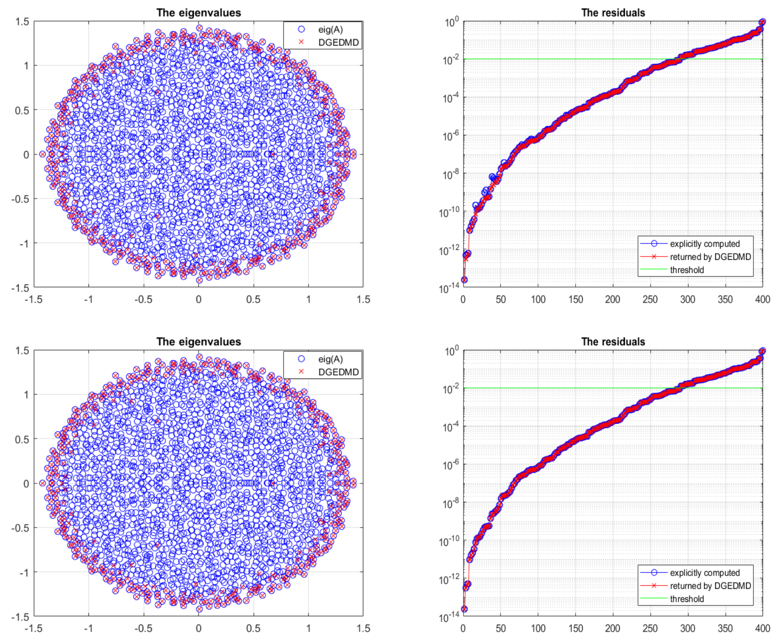

Figure 4: **Example 3.13** (DGEDMD: first row using DGESVDQ, second row using DGEJSV) Left panels: the 400 computed Ritz values (x) and the eigenvalues of $\mathbf{A}$ computed by DGEEV (o). Middle panels: the data driven residuals computed by the subroutines (x) and the residuals computed explicitly using $\mathbf{A}$ (o). Right panels: The 289 selected eigenvalues, based on the residuals below the threshold 10^{-2}. Compare with Figure 2 and Figure 3. Note that the results with DGEJSV are slightly better; see Remark 3.8.

The difference in the accuracy is in accordance with the theory outlined in §3.3 and §3.4. The singular values and the relevant condition numbers in this example are shown in Figure 5. Here $\kappa_2(\mathbf{X}_c) \approx 7.59 \cdot 10^{13}$, which is below $1/\boldsymbol{\varepsilon}$. Once $\kappa_2(\mathbf{X}_c)$ exceeds $1/\boldsymbol{\varepsilon}$, then using all singular values is not advisable; this is illustrated in the examples in §5.1.1.

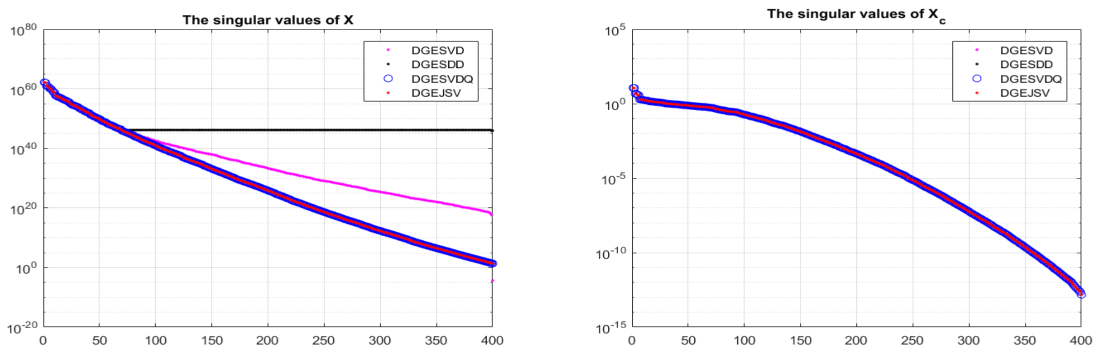

Figure 5: Left panel: the singular values of $\mathbf{X}$, as computed by DGESVD, DGESDD, DGESVDQ and DGEJSV. Right panel: the singular values of $\mathbf{X}_c$ computed by four LAPACK SVD subroutines. Here $\kappa_2(\mathbf{X}) > 10^{60}$ and $\kappa_2(\mathbf{X}_c) \approx 7.59 \cdot 10^{13}$. Note that in the case of $\mathbf{X}_c$ all four subroutines track all singular values equally well. In the case of $\mathbf{X}$, $\kappa_2(\mathbf{X}) > 10^{60}$. Note the importance of Theorem 3.6.

3.6 DMD with scaled data

To exploit the insights from §3.3, we go back to square one. The matrix $\mathbb{A}$ mimics the action of $\mathcal{U}_d$, i.e. (see (20)) $\mathcal{U}_d\mathbf{f}(s) = \mathbb{A}\mathbf{f}(s) + \boldsymbol{\rho}(s)$ for some state s and $\mathbb{A}$ is such that the residuals are minimal in the least squares sense over all available states $\mathbf{z}_i$. To make all data snapshots equally important (independent of the size of $\|\mathbf{f}(\mathbf{z}_i)\|_2$), we can introduce scaling factors $d_i \neq 0$ and consider

$$\mathbb{A} d_i \mathbf{f}(\mathbf{z}_i) \approx \mathcal{U}_d d_i \mathbf{f}(\mathbf{z}_i) = d_i \mathcal{U}_d \mathbf{f}(\mathbf{z}_i). \tag{38}$$

In terms of the data matrices, we replace the pair $(\mathbf{X}, \mathbf{Y})$ with $(\mathbf{X}D, \mathbf{Y}D)$, where $D = \operatorname{diag}(d_i)_{i=1}^m$. If $\mathbf{X}$ is of full column rank, this scaling is hidden in the expression of the DMD matrix $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger = (\mathbf{Y}D)(\mathbf{X}D)^\dagger$ and it does not change $[\mathbb{A}]$.

This invariance under scaling becomes a game changer if we set $D = D_{\mathbf{X}}^{-1}$ and consider the initial data to be $(\mathbf{X}_c, \mathbf{Y}_c) = (\mathbf{X}D_{\mathbf{X}}^{-1}, \mathbf{Y}D_{\mathbf{X}}^{-1})$, as a realisation of (38). This operation is (barring underflows and overflows) both backward and forward stable at all positions (j,k) of both data matrices. Now, if we work with $(\mathbf{X}_c, \mathbf{Y}_c)$, then the condition number $\kappa_2(\mathbf{X}_c)$ replaces $\kappa_2(\mathbf{X})$, and since $\|X_c(:,i)\|_2 = 1$ and $\|X_c\|_2 \le \sqrt{m}$, then the backward error of the type (28) can also be expressed column-wise ($\|\delta\mathbf{X}_c\|_2 \le \epsilon\|X_c\|_2 \Rightarrow \|\delta\mathbf{X}_c(:,i)\|_2 \le \sqrt{m}\epsilon\|X_c(:,i)\|_2$ for all i).

Algorithm 2 $(Z_k, \Lambda_k, r_k, [B_k], [Z_k^{(ex)}]) = \text{xGEDMD}(\mathbf{X}, \mathbf{Y}; \boldsymbol{\tau})$

```
Input:
    $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m), \mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m) \in \mathbb{C}^{n\times m}$ that define a sequence of snapshots
    pairs $(\mathbf{x}_i, \mathbf{y}_i)$. (Tacit assumption is that $n$ is large and that $m \ll n$.)
    Tolerance $\boldsymbol{\tau}$ for the truncation (23).
 1: $D_{\mathbf{X}} = \operatorname{diag}(\|\mathbf{X}(:,i)\|_2)_{i=1}^m$; $\mathbf{X}_c = \mathbf{X}D_{\mathbf{X}}^\dagger$; $\mathbf{Y}_c = \mathbf{Y}D_{\mathbf{X}}^\dagger$.
 2: $[U, \Sigma, V] = svd(\mathbf{X}_c)$ ; {*The thin SVD:* $\mathbf{X}_c = U\Sigma V^*$, $U \in \mathbb{C}^{n\times m}$, $\Sigma = \operatorname{diag}(\sigma_i)_{i=1}^m$}
 3: Determine numerical rank $k$, using (23) with the threshold $\boldsymbol{\tau}$.
 4: Set $U_k = U(:,1:k)$, $V_k = V(:,1:k)$, $\Sigma_k = \Sigma(1:k, 1:k)$
 5: $B_k = \mathbf{Y}_m^{(1)}(V_k\Sigma_k^{-1})$; {*Schmid's data driven formula for* $\mathbb{A}U_k$. *[optional output]*}
 6: $S_k = U_k^* B_k$ {$S_k = U_k^*\mathbb{A}U_k$ *is the Rayleigh quotient.*}
 7: $[W_k, \Lambda_k] = \operatorname{eig}(S_k)$ {$\Lambda_k = \operatorname{diag}(\lambda_i)_{i=1}^k$; $S_k W_k(:,i) = \lambda_i W_k(:,i)$; $\|W_k(:,i)\|_2 = 1$}
 8: $Z_k = U_k W_k$ {*The Ritz vectors*}
 9: $Z_k^{(ex)} = B_k W_k$ {*The (unscaled) Exact DMD vectors [optional].*}
10: $r_k(i) = \|B_k W_k(:,i) - \lambda_i Z_k(:,i)\|_2$, $i = 1, \ldots, k$. {*The residuals (27).*}
Output: $Z_k$, $\Lambda_k$, $r_k$, $[B_k]$, $[Z_k^{(ex)}]$.
```

Remark 3.14 The use of column scaling applies to the Exact DMD as well, so that all improvements of the DMD due to column scaling have a similar effect for the Exact DMD. An interesting observation is that in the case $\kappa_2(\mathbf{X}_c) < 1/\boldsymbol{\tau}$ (e.g. $\boldsymbol{\tau} = n\boldsymbol{\varepsilon}$) the truncation device (23) will select $k = n$, i.e. no information in the subspace determined by the data is wasted.

Example 3.15 We now take the same data as in §3.5, but use the column scaling as in Algorithm 2. In Figure 6, we show the results with the SVD of $\mathbf{X}$ computed using `DGESVD`. Almost identical figures are obtained with the other three SVD subroutines.

3.7 QR compressed DMD

In high resolution simulations, the data snapshots are from high dimensional space (say, over 10^6) and the number of snapshots processed at given time is much smaller, i.e. the data snapshot matrix is tall and skinny. The SVD of a tall and skinny matrix is usually computed by first computing the QR factorization and then, in the second step, the SVD of the square upper triangular factor is computed. Assembling the SVD of the initial matrix is straightforward.

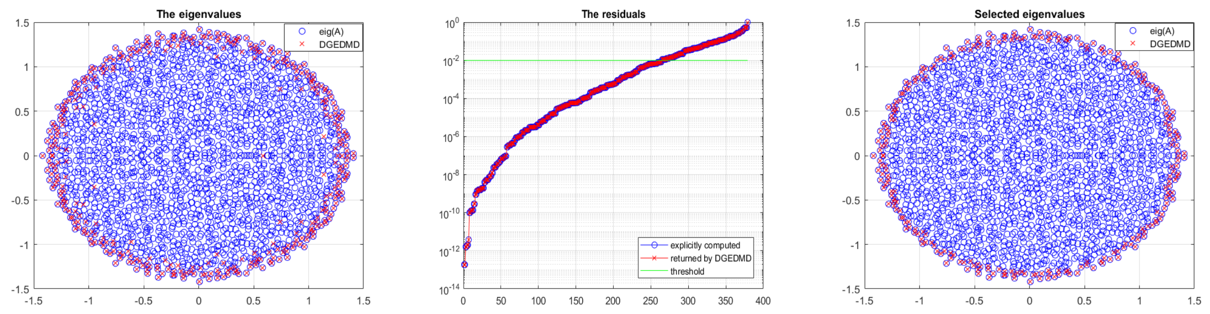

Figure 6: **Example 3.15** (DGEDMD using DGESVD) Left panel: the 379 computed Ritz values (x) and the true eigenvalues of $\mathbf{A}$ (o). Middle panel: the data driven residuals computed by the subroutine (x) and the residuals computed explicitly using $\mathbf{A}$ (o). Right panel: The 264 selected eigenvalues, based on the residuals below the threshold 10^{-2}. The 2000×400 data snapshots are scaled as in Algorithm 2. Compare with Figure 3 and Figure 4.

This strategy is built in the LAPACK subroutines xGESVD, xGESDD with a crossover point (the value of the ratio of the number of columns and the number of rows) that can be in general tuned for best performance. Hence, a modification of the DMD algorithm that first computes the QR factorization of $\mathbf{X}$ (as e.g. in [5]) is already implicitly in the direct LAPACK-based application of the algorithm. A useful extension of this idea, proposed in [22], can be turned into a useful software tool that we describe next.

3.7.1 General non-sequential data

In [22], the QR factorization is used to generate an orthonormal basis in the subspace that contains all data snapshots (both $\mathbf{X}$ and $\mathbf{Y}$). Then, the DMD is applied to a new representation of the original data. More precisely, the QR factorization

$$\begin{pmatrix}\mathbf{X} & \mathbf{Y}\end{pmatrix} = Q\begin{pmatrix} R_{[11]} & R_{[12]} \\ \mathbf{0} & R_{[22]} \\ \mathbf{0} & \mathbf{0}\end{pmatrix} = \widehat{Q}\begin{pmatrix} R_{[11]} & R_{[12]} \\ \mathbf{0} & R_{[22]}\end{pmatrix}, \quad \widehat{Q} = \begin{pmatrix}\widehat{Q}_1 & \widehat{Q}_2\end{pmatrix}, \quad \widehat{Q}_1 = \widehat{Q}(:,1:m), \tag{39}$$

is interpreted as a (unitary[7]) change of coordinates, so that

$$\mathbf{X} = \widehat{Q}\begin{pmatrix} R_{[11]} \\ \mathbf{0}\end{pmatrix} = \widehat{Q}_1 R_{[11]}, \quad \mathbf{Y} = \widehat{Q}\begin{pmatrix} R_{[12]} \\ R_{[22]}\end{pmatrix} = \widehat{Q}_1 R_{[12]} + \widehat{Q}_2 R_{[22]}.$$

Now, if $R_x = U_x\Sigma_x V_x^*$ is the SVD of the $m \times m$ matrix $R_x = R_{[11]}$, then

$$\mathbf{X} = \widehat{Q}_1 U_x \Sigma_x V_x^* = U\Sigma V^*, \quad U = \widehat{Q}_1 U_x, \quad \Sigma = \Sigma_x, \quad V = V_x, \tag{40}$$

is the SVD of $\mathbf{X}$. When we select k (as described in §3.2.1, but applied to $R_{[11]}$ instead of $\mathbf{X}$) then $U_k = \widehat{Q}_1 U_{xk}$, where $U_{xk} = U_x(:,1:k)$ and

$$\begin{aligned} S_k &= U_k^*\mathbf{Y}V_k\Sigma_k^{-1} = U_{xk}^*\widehat{Q}_1^*(\widehat{Q}_1 R_{[12]} + \widehat{Q}_2 R_{[22]})V_{xk}\Sigma_x(1:k,1:k)^{-1} && (41)\\ &= U_{xk}^* R_{[12]} V_{xk}\Sigma_{xk}^{-1}. && (42)\end{aligned}$$

Note that here $\mathbf{X} = \widehat{Q}_1 R_{[11]}$ is the QR factorization of $\mathbf{X}$ and that the same S_k is obtained by computing $S_k = (U_k^*\mathbf{Y})V_k\Sigma_k^{-1}$, i.e. the factorization of $(\mathbf{X}, \mathbf{Y})$ is not necessary. How do we then justify the extra effort to compute the QR factorization (39) of $(\mathbf{X}, \mathbf{Y})$?

[7]Hence, the Euclidean lengths and the angles between the snapshots are preserved.

- The matrix S_k is available with no extra computation, as we can use $R_{[12]}$.
- The residuals are an important information and the cost of computing the residuals is reduced because
$$r_i = \mathbf{Y}V_k\Sigma_k^{-1}w_i - \lambda_i U_k w_i = \widehat{Q}(R_{[:2]}V_{xk}\Sigma_k^{-1}w_i - \lambda_i U_{xk}w_i)$$
so that $\|r_i\|_2 = \|R_{[:2]}V_{xk}\Sigma_k^{-1}w_i - \lambda_i U_{xk}w_i\|_2$ is computed more efficiently and the Ritz pairs can be selected using the computation in the $2m$-dimensional subspace. This avoids computation of the $n \times k$ matrix $\mathbf{Y}V_k$ ($(2m-1)nk$ flops) and for each w_i the norm $\|r_i\|_2$ is computed at a cost that does not involve n.
- The refinement of the Ritz vectors can also be done in the $2m$-dimensional subspace.
- Another argument is the spatio-temporal representation of the snapshots (25) that is accomplished by solving the structured least squares problem (26). Due to the unitary invariance of the norm $\|\cdot\|_2$, the optimization can be done (by keeping the modes in factored form) in the $2m$-dimensional (instead of n-dimensional) space.
- The forward-backward DMD [12] applies DMD twice – first with the data $(\mathbf{X}, \mathbf{Y})$ and then, backward in time, with $(\mathbf{Y}, \mathbf{X})$ so that both the SVD of $\mathbf{X}$ and of $\mathbf{Y}$ are computed. With the factorization (39), this means computing the SVD of $R_x = R_{[11]} \in \mathbb{C}^{m\times m}$ and of $R_y = R_{[:2]} \in \mathbb{C}^{2m\times m}$, which is much more efficient if $m \ll n$.
- In the case of extremely large dimension n, when the memory capacity and the cost of memory traffic become major issues, after computing the out-of-core QR factorization, we can compute the DMD in $2m$-dimensional subspace. On modern multi-core hardware, highly optimized implementations of the QR factorization of tall and skinny matrices is available [13], [38].

In this report, we do not describe this approach any further and its software implementation will be available as a part of another work. Instead, in §3.7.2 we focus to the case of data taken from a single trajectory.

Remark 3.16 The QR factorization of $(\mathbf{X}, \mathbf{Y})$ is used in [52, Algorithm 3], but for different purposes and after computing the SVD of $\mathbf{X}$. There, in our notation, $\widehat{Q}$ is used to compress $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$ and $\widehat{Q}^*\mathbb{A}\widehat{Q}$ is computed explicitly using the SVD of $\mathbf{X}$.

3.7.2 Single trajectory data

In the case of data from a single trajectory $\mathbb{F} = (\mathbf{z}_1, \ldots, \mathbf{z}_m, \mathbf{z}_{m+1})$, we have $\mathbf{X} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$, $\mathbf{Y} = (\mathbf{z}_2, \ldots, \mathbf{z}_{m+1})$, and the QR compression is simpler and more efficient. It suffices to compute the QR factorization

$$(\mathbf{z}_1, \ldots, \mathbf{z}_m, \mathbf{z}_{m+1}) = Q\begin{pmatrix} R \\ \mathbf{0} \end{pmatrix} = \widehat{Q}R, \;\; \text{where } \; Q^*Q = \mathbb{I}_n, \;\; \widehat{Q} = Q(:, 1:m+1), \tag{43}$$

$\mathrm{range}(\widehat{Q}) \supseteq \mathrm{range}(\mathbb{F})$, and R is $(m+1)\times(m+1)$ upper triangular with columns that contain the coordinates of the data snapshots in the basis $\widehat{Q}$. More precisely, $\mathbf{X} = \widehat{Q}R_x$, $\mathbf{Y} = \widehat{Q}R_y$, where

$$R = \begin{pmatrix} \times & \divideontimes & \divideontimes & \divideontimes & \div \\ & \divideontimes & \divideontimes & \divideontimes & \div \\ & & \divideontimes & \divideontimes & \div \\ & & & \divideontimes & \div \\ & & & & \div \end{pmatrix}, \;\; R_x = R(:, 1:m) = \begin{pmatrix} \times & \divideontimes & \divideontimes & \divideontimes \\ & \divideontimes & \divideontimes & \divideontimes \\ & & \divideontimes & \divideontimes \\ & & & \divideontimes \\ & & & 0 \end{pmatrix}, \;\; R_y = R(:, 2:m+1) = \begin{pmatrix} \divideontimes & \divideontimes & \divideontimes & \div \\ \divideontimes & \divideontimes & \divideontimes & \div \\ & \divideontimes & \divideontimes & \div \\ & & \divideontimes & \div \\ & & & \div \end{pmatrix}. \tag{44}$$

Note that this factorization costs nearly as the QR factorization of $\mathbf{X}$, and it is already payed off in the SVD of $\mathbf{X}$ as in (40); there is no need for additional justifications as in §3.7.1. The computation proceeds by applying the DMD algorithm to (R_x, R_y) and then lifting the Ritz vector to the original space by multiplication with $\widehat{Q}$. The computation of the residual, optional refinement of the Ritz vectors and the spatio-temporal representation of the snapshots are done in the $(m+1)$-dimensional subspace, thus more efficiently. The QR-compressed version of the DMD is outlined in Algorithm 3. Note that, for the sake of simplicity, we did not take advantage of the fact that the last row in R_x is zero. Also, in the current implementation we use general SVD driver subroutines, since the specialized SVD subroutines for triangular matrices are not provided in LAPACK.

Algorithm 3 $(Z_k, \Lambda_k, r_k, [\widehat{Q}, R, \widehat{Z}_k], [\widehat{B}_k], [\widehat{Z}_k^{(ex)}]) = \mathrm{xGEDMDQ}(\mathbb{F}, \boldsymbol{\tau})$

```
Input:
    $\mathbb{F} = (\mathbf{z}_1, \ldots, \mathbf{z}_m, \mathbf{z}_{m+1})$ that defines a sequence of snapshots from a single trajectory initial-
    ized with $\mathbf{z}_1 \in \mathbb{C}^n$. (Tacit assumption is that $n$ is large and that $m \ll n$.)
    Tolerance level $\boldsymbol{\tau}$ for the numerical rank determination.
1: $[\widehat{Q}, R] = qr(\mathbb{F}, 0)$ ; {Thin QR factorization.}
2: $R_x = R(1:m+1, 1:m)$, $R_y = R(1:m+1, 2:m+1)$ ;
3: $(\widehat{Z}_k, \Lambda_k, r_k, [\widehat{B}_k], [\widehat{Z}_k^{(ex)}]) = \mathrm{xGEDMD}(R_x, R_y, \boldsymbol{\tau})$; {Algorithm 2 in $(m+1)$-dimensional state
   space.}
4: $Z_k = \widehat{Q}\widehat{Z}_k$
Output: $\mathbf{Z}_k$, $\Lambda_k$, $r_k$, $[\widehat{Q}, R, \widehat{Z}_k]$, $[\widehat{B}_k]$, $[\widehat{Z}_k^{(ex)}]$
```

Remark 3.17 In light of the discussion in §3.3, it should be noted that the backward error in the QR factorization is of type (32), so that so long $\kappa_2(\mathbf{X}_c)$ is moderate, this factorization, as a preprocessor for the DMD, preserves the accuracy described in §3.3.

Example 3.18 We use the same data as in the previous examples. The scaling is switched off and we request the full set of 400 Ritz pairs. If we run `DGEDMDQ` with `DGESVD` or `DGESDD` as SVD solver, the results are similar as in Figure 3. On the other hand, if we use `DGESVDQ` or `DGEJSV`, the results are almost identical as in Figure 4 and we show them in Figure 7 and Figure 8. See also Remark 3.17.

As another justification for the development of `xGEDMDQ`, in §3.7.3 we outline its application to the streaming DMD [27], [55], [35]. With the goal of providing a robust computational tool in various applications of the DMD, and in view of Remark 3.17, our implementation of `xGEDMDQ` is designed to allow for a streaming DMD that we describe next.

3.7.3 Adaptation to sliding data window (streaming DMD)

If we track in time high dimensional data enclosed in a window that slides in discrete time steps, i.e. keeps receiving new and, optionally, discarding old data snapshots (individually or in blocks with dynamically changing widths), then we compute a sequence of the DMD's on a sequence of data windows that dynamically change their widths. In a high dimensional space, this repeated DMD computation may be costly (in particular if the forward-backward DMD scheme is used) and more efficient updating scheme is desirable. One simple idea that we mention here as another motivation for developing `xGEDMDQ` is to keep updating the QR compressed representation and explore the fact that the actual DMD can be done inside a lower dimensional subspace. Updating algorithms for the QR decomposition are known, see e.g. [11], [42]. Our implementation of the `xGEDMDQ` computes the DMD of the current data window

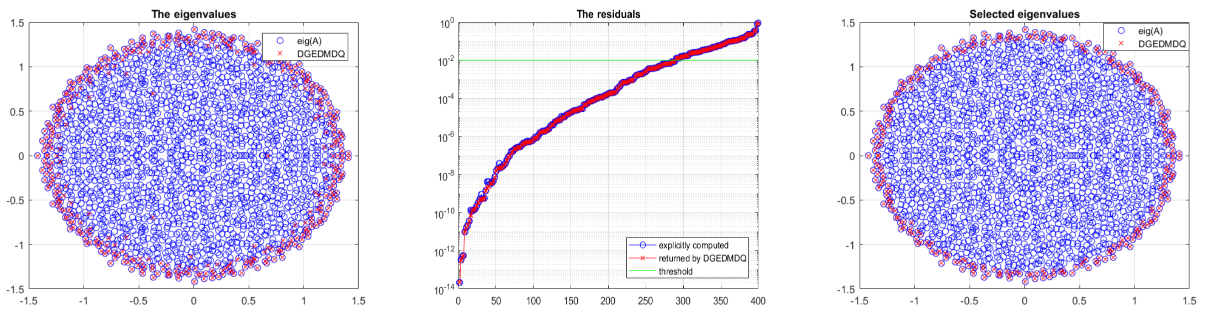

Figure 7: **Example 3.18** (`DGEDMDQ` using `DGESVDQ`) Left panel: the computed Ritz values (x) and the true eigenvalues of $\mathbf{A}$ (o). Middle panel: the data driven residuals computed by the subroutines (x) and the residuals computed explicitly using $\mathbf{A}$ (o). Right panel: The 289 selected eigenvalues, based on the residuals below the threshold 10^{-2}. The data snapshots have not been scaled and the SVD is computed using `DGESVDQ`. The data snapshots were 2000×400 and 400 Ritz pairs have been computed. Compare with Figure 4.

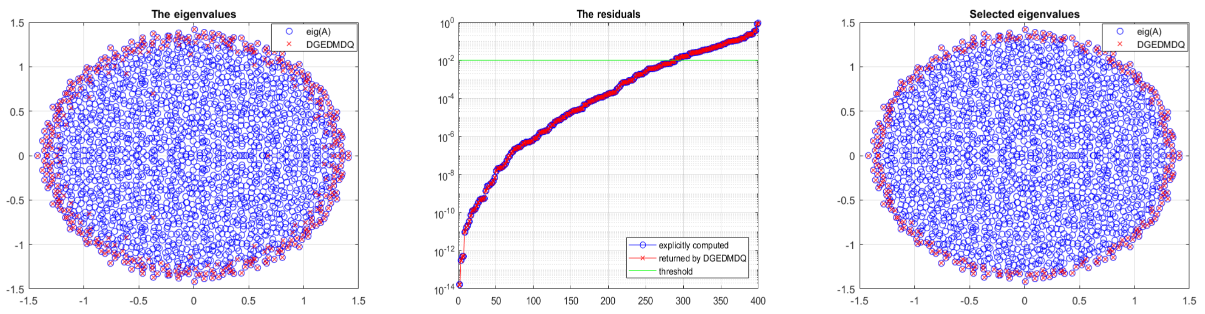

Figure 8: **Example 3.18** (`DGEDMDQ` using `DGEJSV`) Left panel: the computed Ritz values (x) and the true eigenvalues of $\mathbf{A}$ (o). Middle panel: the data driven residuals computed by the subroutines (x) and the residuals computed explicitly using $\mathbf{A}$ (o). Right panel: The 289 selected eigenvalues, based on the residuals below the threshold 10^{-2}. The data snapshots have not been scaled and the SVD is computed using `DGEJSV`. The data snapshots were 2000×400 and 400 Ritz pairs have been computed. Compare with Figure 4.

and returns the current QR factorization that can be easily updated/downdated, depending on the input signal. For the readers convenience and for better understanding of the rationale behind the implementation of `xGEDMDQ`, in the next two paragraphs we outline the procedures of adding (new) and discarding (oldest) arbitrary number of snapshots.

Adding new snapshots. Suppose a block $\mathbf{f}$ of $k \geq 1$ snapshots has been received and the QR compressed representation $\mathbf{F} = QR$ needs to be updated. This is well known procedure based on the relation

$$\mathbf{F}_{new} = (\mathbf{F}, \mathbf{f}) = \begin{pmatrix} Q & (\mathbb{I}_n - QQ^*)\mathbf{f} \end{pmatrix} \begin{pmatrix} R & Q^*\mathbf{f} \\ 0 & \mathbb{I}_k \end{pmatrix}. \tag{45}$$

Note that $\mathbf{f} - QQ^*\mathbf{f}$ is the Gram-Schmidt orthogonalization and that in floating point computation this step should be done with reorthogonalization. If $\mathbf{f} - Q(Q^*\mathbf{f}) = Q_1R_1$ is the thin QR factorization, then

$$\mathbf{F}_{new} = \begin{pmatrix} Q & Q_1 \end{pmatrix} \begin{pmatrix} R & Q^*\mathbf{f} \\ 0 & R_1 \end{pmatrix} = Q_{new} R_{new},$$

which allows for continuous use of the QR compressed DMD. In general, k is small, e.g. $k = 1$, so that this step is computationally inexpensive. If $k = 1$, then $R_{x,new} = R = (R_x, R(:, m+1))$ so that the new SVD is computed for the matrix

$$R_{x,new} = \begin{pmatrix} R_x & R(1:m, m+1) \\ \mathbf{0} & R_{m+1,m+1} \end{pmatrix}.$$

Of course, at this point we may consider updating the SVD, instead computing the SVD of $R_{x,new}$ from scratch. However, since such a procedure is not available in LAPACK and since the main savings is in replacing the dimension n with much smaller m, computing the SVD of $R_{x,new}$ from scratch is acceptable and in the future work we will consider using fast updates of the SVD of $R_{x,new}$.

Discarding old snapshots Now suppose we want to discard $\ell \geq 1$ oldest snapshots, i.e.

$$\mathbf{F}_{new} = \mathbf{F}(:, \ell+1 : end) = QR(:, \ell+1 : end) = Q \begin{pmatrix} x & x & x \\ \bullet & x & x \\ \bullet & \bullet & x \\ & \bullet & \bullet \\ & & \bullet \end{pmatrix} \quad \text{(here for illustration } \ell = 2\text{)}.$$

Restoring the triangular factor amounts to systematical annihilation of the positions $\bullet$, using elementary unitary/orthogonal matrices. We illustrate the process using the above small dimensional example. Start with a unitary H_1 (Householder reflector) such that

$$H_1 \begin{pmatrix} x & x & x \\ \bullet & x & x \\ \bullet & \bullet & x \\ & \bullet & \bullet \\ & & \bullet \end{pmatrix} = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & \bullet & x \\ & \bullet & \bullet \\ & & \bullet \end{pmatrix}; \quad \mathbf{F}_{new} = QH_1^* H_1 \begin{pmatrix} x & x & x \\ \bullet & x & x \\ \bullet & \bullet & x \\ & \bullet & \bullet \\ & & \bullet \end{pmatrix} = QH_1^* \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & \bullet & x \\ & \bullet & \bullet \\ & & \bullet \end{pmatrix},$$

and proceed in a similar fashion with unitary H_2, H_3 such that

$$H_2 \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & \bullet & x \\ & \bullet & \bullet \\ & & \bullet \end{pmatrix} = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ & 0 & \bullet \\ & & \bullet \end{pmatrix}, \quad H_3 \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ & 0 & \bullet \\ & & \bullet \end{pmatrix} = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ & 0 & 0 \\ & & 0 \end{pmatrix}, \quad \mathbf{F}_{new} = Q(H_1^* H_2^* H_3^*) \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ & 0 & 0 \\ & & 0 \end{pmatrix}.$$

Altogether, we have

$$\mathbf{F}_{new} = Q_{new} R_{new}, \quad Q_{new} = Q[(H_1^* H_2^* H_3^*)]\,(:, end - \ell).$$

Clearly, the product $H_1^* H_2^* H_3^* \cdots$ is first accumulated (exploiting the smaller matrix dimensions), and then applied as a BLAS 3 operation to Q.

4 Implementation details

Both the DMD (Algorithm 2) and the QR compressed DMD (Algorithm 3) are implemented for all four LAPACK data types, with generic names `xGEDMD` and `xGEDMDQ`, with `x` $\in$ {`S`, `D`, `C`, `Z`}. The code is written following the LAPACK style and documented in detail. In this section, we give a brief overview of the arguments, and relate them with the descriptions of the algorithms in the previous sections. For more details we refer to the source codes.

The code is designed to return the eigenvectors as defined as in the original DMD and, optionally, as in the Exact DMD. Furthermore, if requested by a job parameter, the necessary ingredients for the refinement of the Ritz vectors are computed.

4.1 xGEDMD

We describe in detail the structure of DGEDMD. The structures of SGEDMD and the two complex subroutines (CGEDMD and ZGEDMD) are, *mutatis mutandis*, analogous to DGEDMD.

```
SUBROUTINE DGEDMD( JOBS, JOBZ, JOBR, JOBF,   WHTSVD,  &
                   M, N, X, LDX, Y, LDY, NRNK, TOL,   &
                   K, REIG,  IMEIG,   Z, LDZ,   RES,   &
                   B, LDB, W,  LDW,   S, LDS,          &
                   WORK, LWORK, IWORK, LIWORK, INFO )
```

4.1.1 Brief description of the arguments of DGEDMD and SGEDMD

JOBS, JOBZ, JOBR, JOBF

$\boxed{\text{JOBS}}$ determines whether the initial data snapshots should be scaled with a diagonal matrix that normalizes the columns of **X** or **Y**. The scaling is determined according the value of S as follows:

'S' The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that X*D has unit nonzero columns (in the Euclidean 2-norm)

'C' The snapshots are scaled as with the 'S' option. If it is found that an i-th column of X is zero vector and the corresponding i-th column of Y is non-zero, then the i-th column of Y is set to zero and a warning flag is raised.

'Y' The data snapshots matrices X and Y are multiplied by a diagonal matrix D so that Y*D has unit nonzero columns (in the Euclidean 2-norm)

'N' No data scaling is used.

$\boxed{\text{JOBZ}}$ Determines whether the eigenvectors (Koopman modes) will be computed.

'V' The eigenvectors (Koopman modes) will be computed and returned in the matrix Z. See the description of Z.

'F' The eigenvectors (Koopman modes) will be returned in factored form as the product X*W, where X contains a POD basis (leading left singular vectors of the data matrix) and W contains the eigenvectors of the corresponding Rayleigh quotient. See the descriptions of X, W, Z.

'N' The eigenvectors are not computed.

$\boxed{\text{JOBR}}$ Determines whether to compute the residuals.

'R' The residuals for the computed eigenpairs will be computed and stored in the array RES. See the description of RES. For this option to be legal, JOBZ must be 'V'.

'N' The residuals are not computed.

$\boxed{\text{JOBF}}$ specifies whether to store information needed for post-processing (e.g. computing refined Ritz vectors)

'R' The matrix needed for the refinement of the Ritz vectors is computed and stored in the array B. See the description of B.

'E' The unscaled eigenvectors of the Exact DMD are computed and returned in the array B. See the description of B.

'N' No eigenvector refinement data is computed.

WHTSVD The SVD decomposition of $\mathbf{X}$ can be computed by one of the following LAPACK subroutines: *(i)* DGESVD (QR SVD) ; *(ii)* DGESDD (divide and conquer SVD); *(iii)* DGESVDQ (preconditioned QR SVD); *(iv)* DGEJSV (preconditioned Jacobi SVD). The concrete choice is specified in a job parameter (WHTSVD). This step of the algorithm provides a low rank approximation $\mathbf{X} \approx U_k \Sigma_k V_k^T$, and in future modifications of the code we can include large scale partial SVD solver.

If the job parameters specify that the information for computing the refined computed Ritz pairs is requested, the matrix $\mathbf{Y} V_k \Sigma_k^{-1}$ is computed and on exit returned in the array $\mathbf{Y}$.

M, N, X, LDX, Y, LDY On entry, the real arrays X and Y contain the data matrices $\mathbf{X}$, $\mathbf{Y}$ with M columns and N rows. On exit, X contains the left singular vectors of $\mathbf{X}$. If the residuals are requested, then Y contains the residual vectors; otherwise the content of Y is not changed.

NRNK, TOL These parameter specify how to compute the numerical rank, i.e. how to truncate singular values $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_n$ of the input matrix X. On input, if NRNK equals

-1 $\widetilde{\sigma}_i$ is truncated if $\widetilde{\sigma}_i \leq$ TOL$\star\widetilde{\sigma}_1$

-2 $\widetilde{\sigma}_i$ is truncated if $i \geq 2$ and $\widetilde{\sigma}_i \leq$ TOL$\star\widetilde{\sigma}_{i-1}$

>0 The numerical rank can be enforced by using positive value of NRNK as follows: If 0<NRNK<=N, then at most NRNK largest singular values will be used. If the number of the computed nonzero singular values is less than NRNK, then only those nonzero values will be used and the actually used dimension is less than NRNK. The actual number of the nonzero singular values is returned in the variable K.

K, REIG, IMEIG, Z, LDZ The dimension of the Rayleigh quotient, determined following the specifications in NRNK, TOL is returned in the output variable K. The eigenvalues of the Rayleigh quotient S_k are computed using DGEEV, which computes the eigenvalues and eigenvectors in real realization and using only real computer arithmetic. As a result, DGEDMD computes the Koopman Ritz values and the modes using only real arithmetic. Following the structure of DGEEV, the real and the imaginary parts of the eigenvalues are returned in two real arrays REIG and IMEIG, complex conjugate pairs of eigenvalues are listed with consecutive indices with the positive imaginary part listed first. If the eigenvectors are requested in the output array Z then Z contains real Ritz vectors as follows:

If IMEIG(j) is zero, then Z(:,j) is an eigenvector of the j-th Ritz value REIG(j). If IMEIG(j) > 0 (and IMEIG(j+1) < 0) then Z(:,j) and Z(:,j+1) span a nearly invariant subspace (depending ion the residual) and the Ritz values extracted from this subspace are REIG(j) $+\,\mathfrak{i}\cdot$ IMEIG(j) and REIG(j) $-\,\mathfrak{i}\cdot$ IMEIG(j). The corresponding complex conjugate pair of approximate eigenvectors are Z(:,j) $+\,\mathfrak{i}\cdot$ Z(:,j+1) and Z(:,j) $-\,\mathfrak{i}\cdot$ Z(:,j+1), respectively. If JOBZ=='F', then the above descriptions hold for the columns of X(:,1:K)*W, where the columns of W are the eigenvectors of the K-by-K Rayleigh quotient.

RES On exit, RES(1:K) contains the residuals for the K computed Ritz pairs as follows. If IMEIG(j) $= 0$, then RES(j) $= \|\mathbb{A}\star$Z(:,j) $-$REIG(j)$\star$Z(:,j))$\|_2$. If IMEIG(j)>0 then

$$\texttt{RES(j)} = \texttt{RES(j+1)} = \|\mathbb{A}\star\texttt{Z(:,j:j+1)} - \texttt{Z(:,j:j+1)}\star\texttt{B}\|_F, \quad B = \begin{pmatrix} \texttt{REIG(j)} & \texttt{IMEIG(j)} \\ \texttt{-IMEIG(j)} & \texttt{REIG(j)} \end{pmatrix}.$$

This is a simple way to compute in real arithmetic

$$\texttt{RES(j)} = \|\mathbb{A}Z_c(:,j) - \lambda_j Z_c(:,j)\|_2, \quad \texttt{RES(j+1)} = \|\mathbb{A}Z_c(:,j+1) - \lambda_{j+1} Z_c(:,j+1)\|_2,$$

where $\lambda_j =$ REIG(j) $+\,\mathfrak{i}\cdot$ IMEIG(j), $Z_c(:,j) = Z(:,j) + \mathfrak{i}\cdot Z(:,j+1)$, $\lambda_{j+1} =$ REIG(j) $-\,\mathfrak{i}\cdot$ IMEIG(j), $Z_c(:,j+1) = Z(:,j) - \mathfrak{i}\cdot Z(:,j+1)$.

B, W, LDW, S, LDS S is the K-by-K Rayleigh quotient, i.e. the matrix $S_k = U_k^* \mathbb{A} U_k$ described in §2.4.2. The array W is used to temporarily hold the right singular values of **X**, and on return it contains the eigenvectors of S, as computed by DGEEV. The array B is used only if the data for computing the refined Ritz vectors (JOBF='R') or the Exact DMD eigenvectors (JOBF='E') are requested.

WORK, LWORK, IWORK, LIWORK If on entry LWORK==-1, or LIWORK==-1 then a workspace query is assumed and the procedure only computes the minimal and the optimal workspace lengths for both WORK and IWORK. In that case, on exit, WORK(1) contains the minimal and WORK(2) is the optimal length of WORK. Similarly IWORK(1) contains the minimal length of IWORK. Otherwise, WORK and IWORK are used as workspace and to return some useful information. IWORK is not used if WHTSVD equals 1. On exit, WORK(1:N) contain the computed singular values of **X**.

INFO On exit, INFO contains status information on the DGEDMD run.

-i<0 On entry, the i-th argument had an illegal value
0 Successful return.
1 Void input. Quick exit (M=0 or N=0).
2 The SVD computation of X did not converge. Suggestion: Check the input data and/or repeat with different WHTSVD.
3 The computation of the eigenvalues did not converge.
4 If data scaling was requested on input and the procedure found inconsistency in the data such that for some column index i, X(:,i) = 0 but Y(:,i) /= 0, then Y(:,i) is set to zero if JOBS=='C'. The computation proceeds with original or modified data and a warning flag is set with INFO=4.

Remark 4.1 Recall that DGEEV scales the eigenvectors to unit Euclidean length. In the case of complex conjugate pairs, this means that the Frobenius norm of the two-column matrix containing the real and the imaginary parts is one. Furthermore, in this case the two columns are post-multiplied by a Givens rotation that makes the largest entry in the real part (and the corresponding imaginary part is zero).

4.2 xGEDMDQ

```
SUBROUTINE DGEDMDQ( JOBS, JOBZ, JOBR, JOBQ, JOBT, JOBF, WHTSVD,    &
                    M, N, F, LDF, X, LDX, Y, LDY, NRNK, TOL,       &
                    K,  REIG, IMEIG, Z, LDZ, RES, B, LDB, W, LDW,  &
                    S, LDS, WORK, LWORK, IWORK, LIWORK, INFO )
```

4.2.1 Brief description of the arguments of DGEDMDQ and SGEDMDQ

JOBS, JOBZ, JOBR, JOBF These arguments are the same as in DGESVD.

JOBQ, JOBT
`JOBQ` specifies whether to explicitly compute and return the orthogonal matrix from the QR factorization.
`'Q'` The matrix Q of the QR factorization of the data snapshot matrix is computed and stored in the array `F`. See the description of `F`.
`'N'` The matrix Q is not explicitly computed.

`JOBT` Specifies whether to return the upper triangular factor from the QR factorization.
`'R'` The matrix R of the QR factorization of the data snapshot matrix `F` is returned in the array `Y`. See the description of `Y`.
`'N'` The matrix R is not returned.

WHTSVD This argument is the same as in `DGEDMD`.

M, N, F, LDF On entry, the columns of `F` are the sequence of data snapshots from a single trajectory, taken at equidistant discrete times. It is assumed that the column norms of `F` are in the range of the normalized floating point numbers. On exit, if `JOBQ` equals:
`'Q'` the array `F` contains the orthogonal matrix/factor of the QR factorization of the initial data snapshots matrix `F`. See the description of `JOBQ`.
`'N'` the entries in `F` strictly below the main diagonal contain, column-wise, the information on the Householder vectors, as returned by `DGEQRF`. The remaining information to restore the orthogonal matrix of the initial QR factorization is stored in `WORK(1:N)`. See the description of `WORK`.

X, LDX, Y, LDY `X` is a `MIN(M,N)`-by-`(N-1)` array that is used as worskpace to hold representations of the leading `N-1` snapshots in the orthonormal basis computed in the QR factorization of the input array `F`. On exit, the leading `K` columns of `X` contain the leading `K` left singular vectors of the above described content of `X`. See the descriptions of `K`, `V` and `Z`.
`Y` is a `MIN(M,N)`-by-`(N-1)` array that is used as worskpace to hold representations of the trailing trailingN-1 snapshots in the orthonormal basis computed in the QR factorization of the inoput array `F`. On exit, if `JOBT == 'R'`, `Y` contains the `MIN(M,N)`-by-`N` upper triangular factor from the QR factorization of the input data snapshot matrix `F`.

NRNK, TOL, K, REIG, IMEIG, Z, LDZ, RES, W, LDW, S, LDS These parameters are defined as in `DGEDMD`.

B, LDB are defined as in `dgedmd`, but the content of `B` is in the lover dimensional space. If needed, it can be lifted in the original space by pre-multiplication with the orthogonal factor from the initial QR factorization.

WORK, LWORK, IWORK, LIWORK These workspace parameters are defined as in `DGEDMD`. The difference is in the value of `WORK` on exit: `WORK(1:MIN(M,N))` contain the scalar factors of the elementary reflectors as returned by `DGEQRF` of the `M`-by-`N` input matrix `F`. `WORK(N+1:2*N-1)` contains the singular values of the input submatrix `F(1:M,1:N-1)`.

INFO On exit, `INFO` contains status information on the `DGEDMDQ` run.

`-i<0` On entry, the `i`-th argument had an illegal value

`0` Successful return.

`1` Void input. Quick exit (`M=0` or `N=0`).

`2` The SVD computation of `X` did not converge. (See the usage of `X` as workspace.) Suggestion: Check the input data and/or repeat with different `WHTSVD`.

`3` The computation of the eigenvalues did not converge.

`4` If data scaling was requested on input and the procedure found inconsistency in the data such that for some column index `i`, `X(:,i) = 0` but `Y(:,i) /= 0`, then `Y(:,i)` is set to zero if `JOBS=='C'`. The computation proceeds with original or modified data and a warning flag is set with `INFO=4`.

5 Numerical examples

We have tested both `xGEDMD` and `xGEDMDQ` for all four data types, `x` $\in$ {`S`, `D`, `C`, `Z`}. In addition to the examples shown in §3.5 and §3.7.2, in this section we provide selected examples to illustrate and explain some details that may be useful to practitioners/users of the code.

5.1 Numerical tests and case studies with synthetic examples

We continue the experiments with the data set that is generated in the same way as in §3.5 Since we know $\mathbf{A}$ explicitly, we can conveniently use it to check the accuracy of the computed outputs for the subroutines under examination. In §5.1.1, we test and illustrate the limits of accuracy that are a priori known from the numerical analysis outlined in §3.3.

5.1.1 The limits of accuracy discussed in §3.3

The purpose of this example is to illustrate the limits of accuracy based on the perturbation theory outlined in §3.3. The data matrices are generated as in §3.5, and we set $\mathbf{A}$ to be 1000×1000 and $\mathbf{X}$ and $\mathbf{Y}$ are 1000×700. The matrix $\mathbf{X}$ is ill-conditioned so that $\kappa_2(\mathbf{X}) > 10^{100}$, $\kappa_2(\mathbf{X}_c)$ closely exceeds $1/\boldsymbol{\varepsilon}$ and the perturbation theory outlined in §3.3 does not apply. (Here $\boldsymbol{\varepsilon}$ is the round-off in the IEEE 64 bit double precision.) The computed singular values of $\mathbf{X}$ and $\mathbf{X}_c$ are shown in Figure 9.

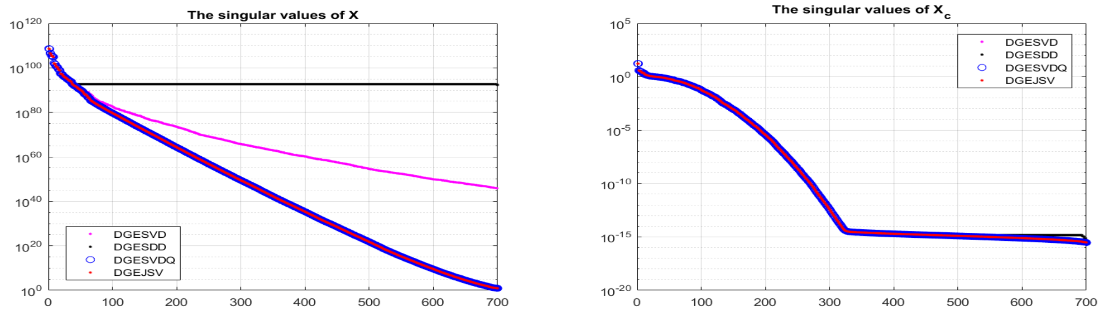

Figure 9: The singular values of $\mathbf{X}$ and $\mathbf{X}_c$, as computed by the LAPACK SVD subroutines `DGESVD`, `DGESDD`, `DGESVDQ`, `DGEJSV`. The estimated condition numbers are $\kappa_2(\mathbf{X}) > 10^{100}$, $\kappa_2(\mathbf{X}_c) \approx 5.6038\cdot10^{16}$. Compare with Figure 5.

Example 5.1 In the first test, we require $k = 700$, i.e. the truncation device is switched off. If the SVD is computed using DGESVD or DGESDD, then in both DGEDMD and DGEDMDQ the results are with large errors, similarly as in Example 3.12. The formula (27) for the residuals failed numerically because the SVD was inaccurate. As a result, severely underestimated residuals resulted in accepting all computed Ritz pairs, see Figure 10 and Figure 11.

Now, we repeat the test with $k = 700$, but using DGESVDQ (Figure 12) and DGEJSV (Figure 13). Since $\kappa_2(\mathbf{X}_c) \approx 5.6038 \cdot 10^{16}$ there is no guarantee for the accuracy and we do not expect good performance. However, few details are worth noticing. The results of DGEDMDQ are in both cases noticeably worse than of DGEDMD. This is because DGEDMDQ starts with the QR factorization and in DGEDMD both DGESVDQ and DGEJSV start with the QR factorization with column pivoting combined with sorting the rows (thus mimicking the Powel-Reid complete pivoting [41], [10]) which to certain extent reduces the dependence on $\kappa_2(\mathbf{X}_c)$, see [18], [19], [20], [21]. Further, comparing the results of DGEDMD, we see that DGEJSV provided more accurate SVD that DGESVDQ; see Remark 3.17 and Remark 3.8.

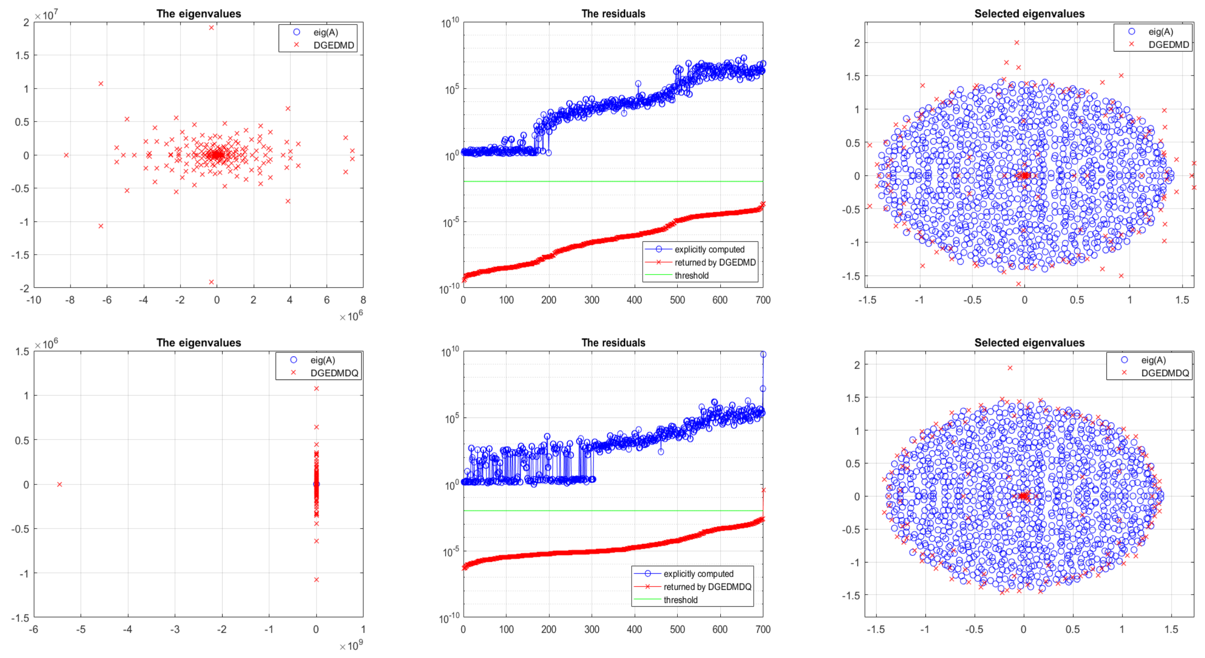

Figure 10: **Example 5.1** (DGEDMD (first row) and DGEDMDQ (second row) using DGESVD of $\mathbf{X}$) The number of Ritz pairs $k = 700$ is set as input parameter.

Example 5.2 Now, with the same data as in Example 5.1, we repeat the computation using the truncation device (23). As expected, both DGEDMD and DGEDMDQ return nearly identical results independent of the particular choice of the SVD subroutine. In all cases 24 Ritz pairs are returned and 16 are selected with the residual threshold set to 10^{-2}. Since the figures in this case are visually identical, for illustration we only show the results of DGEDMD using DGESVD, see Figure 14.

Example 5.3 In the last test with the data set of this subsection, we follow Algorithm 2 and use the column-scaled data with the truncation device (23). We show in Figure 15 the results for DGEDMD using DGEJSV. The results in all other case are nearly the same – 287 Ritz pairs are computed and 214 selected using the residual threshold.

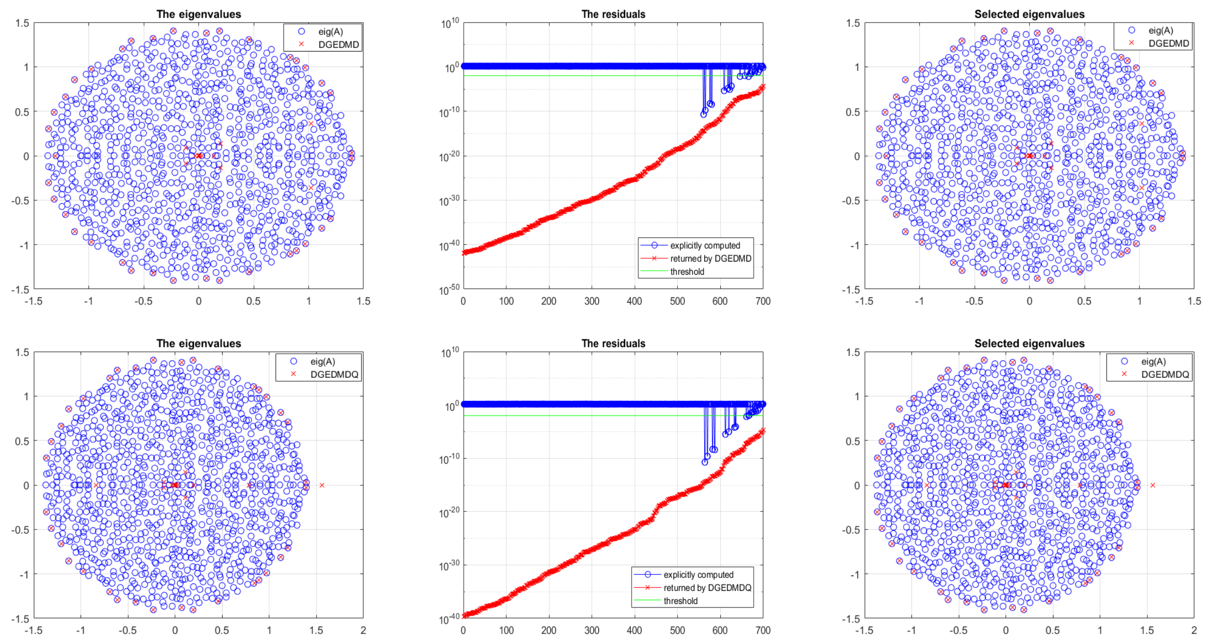

Figure 11: **Example 5.1** (`DGEDMD` (first row) and `DGEDMDQ` (second row) using `DGESDD` of $\mathbf{X}$) The number of Ritz pairs $k = 700$ is set as input parameter.

5.1.2 A comparison of single precision and double precision

The next example is chosen to further emphasize the relevance of the analysis in §3.3, and it illustrates how, with a particular choice of the SVD subroutine, single precision computation can outperform double precision in numerical accuracy.

Example 5.4 We generate $m = 200$ snapshot pairs in $\mathbb{R}^{1000}$ in the same way as in §3.5. Then we run `DGEDMD` using `DGESVD` and `SGEDMD` using `SGEJSV`. The data matrices are strongly graded and $\kappa_2(\mathbf{X}) > 10^{30}$, but $\kappa_2(\mathbf{X}_c) < 10^7$. A stress test condition is imposed: k is explicitly set to m and the original (unscaled) data is used. The results are shown in Figure 16. If we scale the data then, as guaranteed by the theory and already seen in the previous examples, `DGEDMD` using `DGESVD` performs well.

5.1.3 An example with the Exact DMD

In this example, we illustrate the discussion on the Exact DMD from §3.1.1.

Example 5.5 Test data are generated as in the previous synthetic examples. We set $n = 2000$, $m = 300$. With the JOB parameter `JOBF`, `DGEDMD` returns both the DMD Ritz vectors and the Exact DMD eigenvectors. Now the quality of the computed eigenpairs can be tested as follows: we can use $\mathbf{A}$ (our pseudo-random matrix that generates the data and is known for the purpose of the test, but not accessible to the tested algorithms) to compute the residuals for both the DMD vectors and the Exact DMD vectors. Also, for the Exact DMD vectors (normalized in the same way as the Ritz vectors in the DMD), we can use $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$ to compute the residuals. (The computed eigenvalues are the same in both cases.) The condition number of $\mathbf{X}$ is estimated as $\kappa_2(\mathbf{X}) > 10^{45}$, and $\kappa_2(\mathbf{X}_c) < 5 \cdot 10^8$. We first use unscaled data. The results shown in Figure 17 are instructive.

Now we repeat the test, but with the scaling option turned on. Since $\kappa_2(\mathbf{X}_c) < 5 \cdot 10^8$, the full set of 300 eigenpairs is computed (instead of 59 in the unscaled case). The residuals, computed in a similar fashion as in Figure 17 are shown in Figure 18.

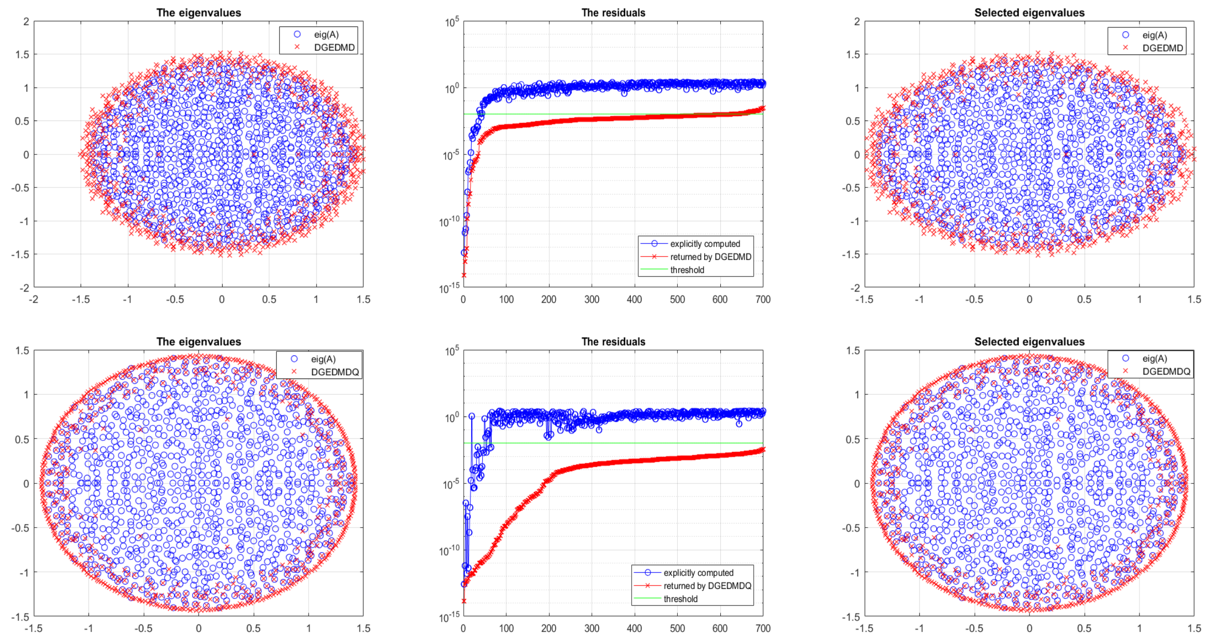

Figure 12: **Example 5.1** (`DGEDMD` (first row) and `DGEDMDQ` (second row) using `DGESVDQ` of $\mathbf{X}$) The number of Ritz pairs $k = 700$ is set as input parameter. Compare with Figure 13.

Similar results are obtained if the Exact DMD is used in the QR compressed form (using `JOBF='E'` in `DGEDMDQ`). Since the graphs show similar behavior, they are omitted for the sake of the length of the paper.

6 Acknowledgments

This material is based upon work supported by the DARPA Small Business Innovation Research Program (SBIR) Program Office under Contract No. W31P4Q-21-C-0007 to AIMdyn, Inc. Part of this work was supported by the DARPA PAI project "Physics-Informed Machine Learning Methodologies" Contract No: HR0011-18-9-0033 and the DARPA MoDyL project "A Data-Driven, Operator-Theoretic Framework for Space-Time Analysis of Process Dynamics" Contract No: HR0011-16-C-0116.

References

[1] Ahmad Abdelfattah, Hartwig Anzt, Aurelien Bouteiller, Anthony Danalis, Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, Stephen Wood, Panruo Wu, Ichitaro Yamazaki, and Asim YarKhan. Roadmap for the development of a linear algebra library for exascale computing: SLATE: Software for Linear Algebra Targeting Exascale. SLATE Working Notes 01, ICL-UT-17-02, 2017-06 2017.

[2] S. Akshay, K. P. Soman, Neethu Mohan, and S. Sachin Kumar. *Dynamic Mode Decomposition and Its Application in Various Domains: An Overview*, pages 121–132. Springer International Publishing, Cham, 2021.

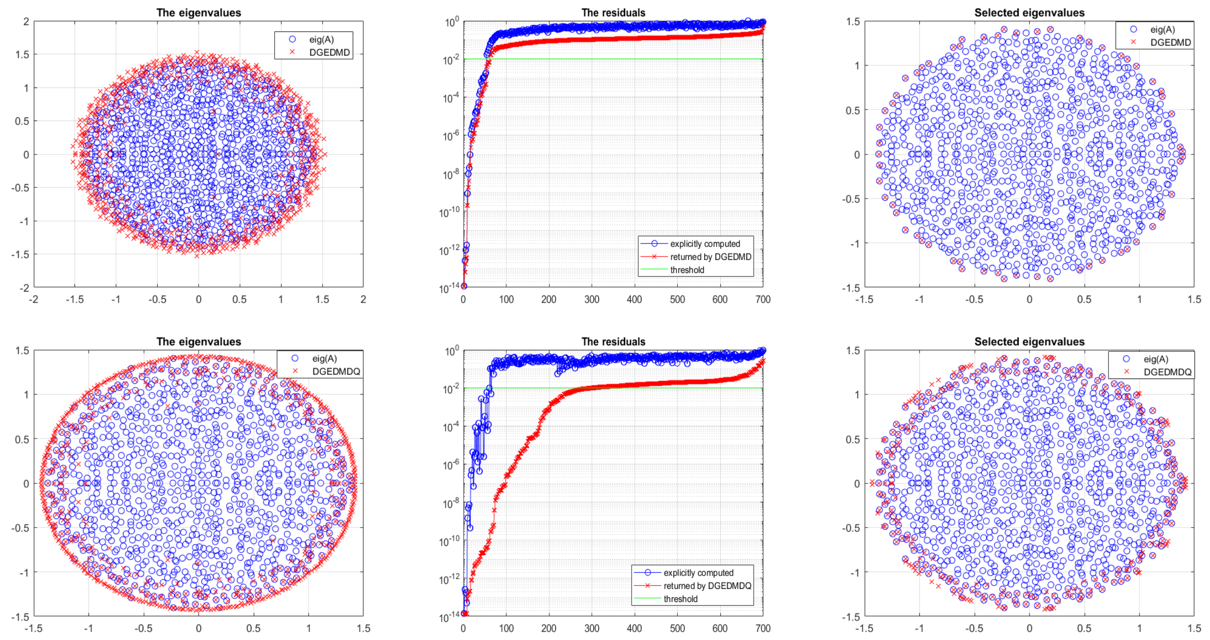

Figure 13: **Example 5.1** (`DGEDMD` (first row) and `DGEDMDQ` (second row) using `DGEJSV` of $\mathbf{X}$) The number of Ritz pairs $k = 700$ is set as input parameter. Compare with Figure 12.

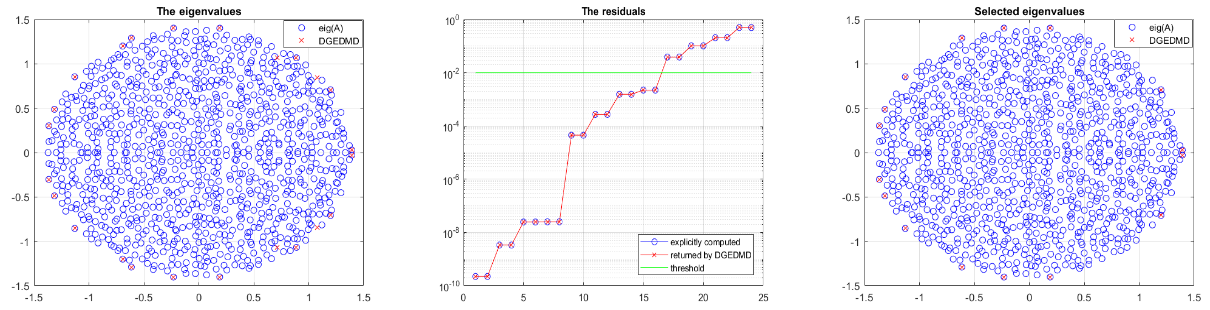

Figure 14: **Example 5.2** (`DGEDMD` using `DGESVD` of $\mathbf{X}$) The number of Ritz pairs $k = 24$ is determined using (23), and 16 are selected with the residual threshold set to 10^{-2}.

[3] D. J. Alford-Lago, C. W. Curtis, A. T. Ihler, and O. Issan. Deep learning enhanced dynamic mode decomposition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3):033116, 2022.

[4] Alessandro Alla and J. Nathan Kutz. Nonlinear model order reduction via dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 39(5):B778–B796, 2017.

[5] Sreevatsa Anantharamu and Krishnan Mahesh. A parallel dynamic mode decomposition algorithm using modified full orthogonalization arnoldi for large sequential snapshots, 05 2018.

[6] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen. *LAPACK Users' Guide (Third Ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

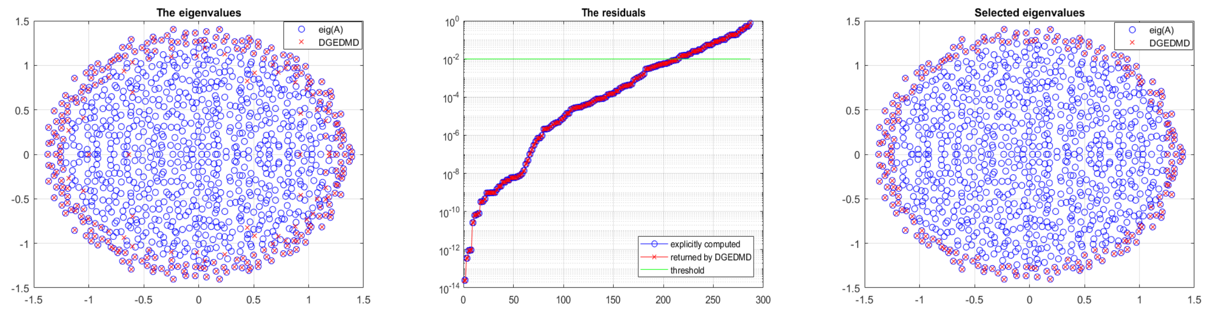

Figure 15: **Example 5.3** (`DGEDMD` using `DGEJSV` of $\mathbf{X}_c$) The number of Ritz pairs $k = 287$ is determined using (23), and 214 are selected with the residual threshold set to 10^{-2}.

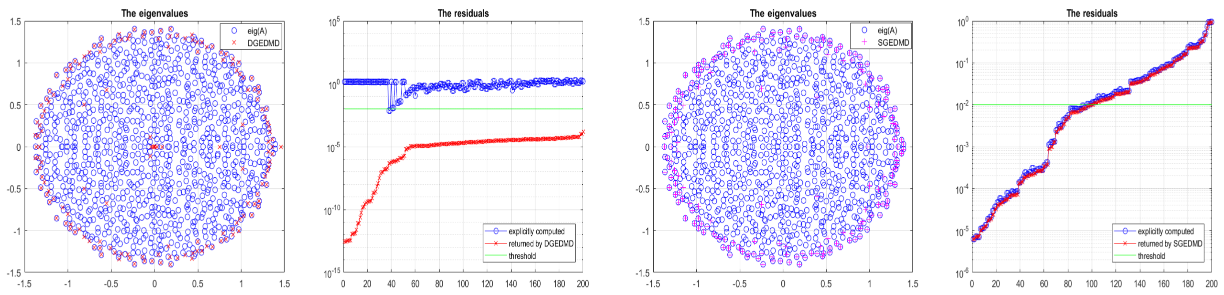

Figure 16: **Example 5.4** (`DGEDMD` using `DGESVD` and `SGEDMD` using `SGEJSV`) The number of Ritz pairs $k = 200$ is set as input parameter. $\mathbf{X}$ is not scaled and $\kappa_2(\mathbf{X}) > 10^{30}$, $\kappa_2(\mathbf{X}_c) < 10^7$. A single precision code (with proper components) can outperform (in numerical accuracy) a double precision code.

[7] L. S. Blackford, J. Choi, A. Cleary, E. D'Azeuedo, J. Demmel, I. Dhillon, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK User's Guide*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[8] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022.

[9] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

[10] A. J. Cox and N. J. Higham. Stability of Householder QR factorization for weighted least squares problems. In D. F. Griffiths, D. J. Higham, and G. A. Watson, editors, *Numerical Analysis 1997, Proceedings of the 17th Dundee Biennial Conference*, volume 380 of *Pitman Research Notes in Mathematics*, pages 57–73. ,, 1998.

[11] James W Daniel, Walter Bill Gragg, Linda Kaufman, and Gilbert W Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation*, 30(136):772–795, 1976.

[12] S. T. M. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):42, 2016.

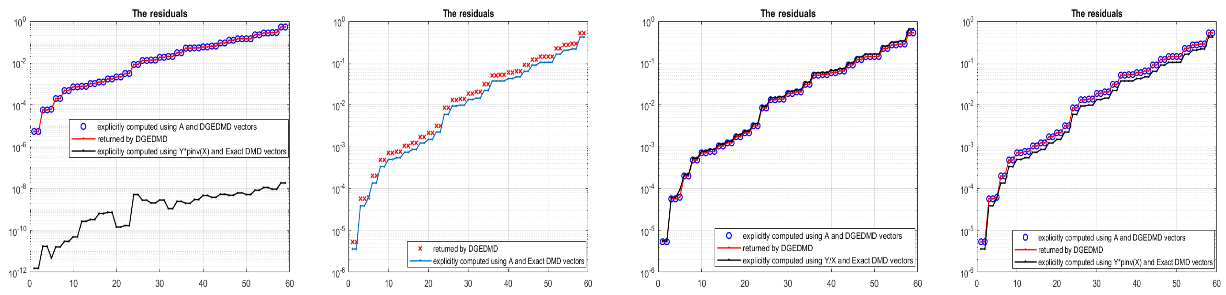

Figure 17: **Example 5.5** (`DGEDMD` using `DGEJSV` with the Exact DMD option) The 2000×300 data matrices $\mathbf{X}$, $\mathbf{Y}$ are not scaled; $\kappa_2(\mathbf{X}) > 10^{45}$. The total of 59 eigenpairs are computed. *First panel:* The residuals computed by the DMD, the explicitly computed residuals of the DMD eigenvectors using $\mathbf{A}$, and the explicitly computed residuals of the Exact DMD eigenvectors using explicitly computed $\mathbb{A} = \mathbf{YX}^{\dagger}$ in Matlab as `Y*pinv(X)`. *Second panel:* The residuals returned by the DMD and the explicitly computed residuals of the Exact DMD eigenvectors using $\mathbf{A}$. *Third panel:* Similar as the first, but $\mathbf{YX}^{\dagger}$ is computed in Matlab as `Y/X`. *Fourth panel:* Similar as the first, but $\mathbf{X}$ and $\mathbf{Y}$ are scaled before computing `Y*pinv(X)`.

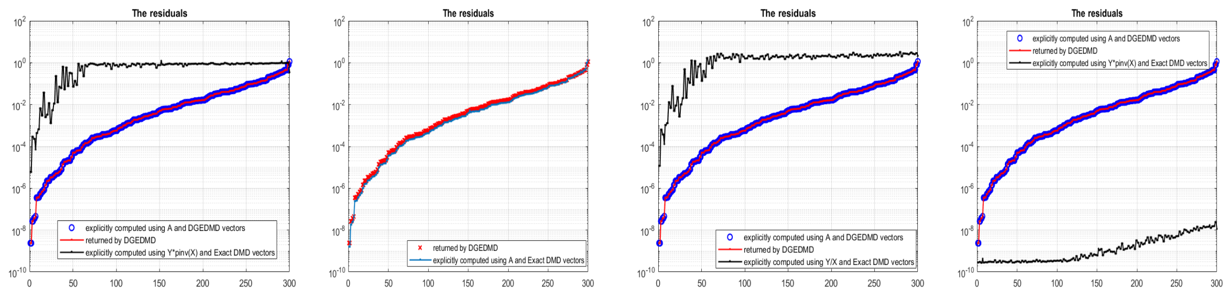

Figure 18: **Example 5.5** (`DGEDMD` using `DGEJSV` with the Exact DMD option) The panels show the results obtained in similarly as in Figure 17, with the difference that the DMD and the Exact DMD use scaled data (See Algorithm 2). In this example the numerical rank of $\mathbf{X}$ is determined as 59 and the numerical rank of $\mathbf{X}_c$ is 300.

[13] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM Journal on Scientific Computing*, 34(1):A206–A239, 2012.

[14] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. *Lin. Alg. Appl.*, 299:21–80, 1999.

[15] James Demmel and Krešimir Veselić. Jacobi's method is more accurate than QR. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1204–1245, 1992.

[16] Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and Ichitaro Yamazaki. *Accelerating Numerical Dense Linear Algebra Calculations with GPUs*, pages 3–28. Springer International Publishing, Cham, 2014.

[17] Zlatko Drmač. *Dynamic Mode Decomposition-A Numerical Linear Algebra Perspective*, pages 161–194. Springer International Publishing, Cham, 2020.

[18] Z. Drmač. A posteriori computation of the singular vectors in a preconditioned Jacobi SVD algorithm. *IMA J. Numer. Anal.*, 19:191–213, 1999.

[19] Z. Drmač and K. Veselić. New fast and accurate Jacobi SVD algorithm: I. *SIAM J. Matrix Anal. Appl.*, 29(4):1322–1342, 2008.

[20] Z. Drmač and K. Veselić. New fast and accurate Jacobi SVD algorithm: II. *SIAM J. Matrix Anal. Appl.*, 29(4):1343–1362, 2008.

[21] Zlatko Drmač. Algorithm 977: A QR–preconditioned QR SVD method for computing the SVD with high accuracy. *ACM Trans. Math. Softw.*, 44(1), July 2017.

[22] Zlatko Drmač, Igor Mezić, and Ryan Mohr. Data driven modal decompositions: Analysis and enhancements. *SIAM Journal on Scientific Computing*, 40(4):A2253–A2285, 2018.

[23] Zlatko Drmač, Igor Mezić, and Ryan Mohr. Identification of nonlinear systems using the infinitesimal generator of the koopman semigroup-a numerical implementation of the mauroy-goncalves method. *Mathematics*, 9(17), 2021.

[24] Stanley C. Eisenstat and Ilse C. F. Ipsen. Relative perturbation techniques for singular value problems. *SIAM Journal on Numerical Analysis*, 32(6):1972–1988, 1995.

[25] T. Eisner, B. Farkas, M. Haase, and R. Nagel. *Operator Theoretic Aspects of Ergodic Theory*. Springer, New York, 2014.

[26] Dimitrios Giannakis and Suddhasattwa Das. Extraction and prediction of coherent patterns in incompressible flows through space-time Koopman analysis. *Physica D: Nonlinear Phenomena*, 402:132211, 2020.

[27] Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11):111701, 2014.

[28] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. Low-rank and sparse dynamic mode decomposition. *Center for Turbulence Research, Annual Research Briefs*, pages 139–152, 2012.

[29] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, February 2014.

[30] M. Korda and I. Mezić. On convergence of extended dynamic mode decomposition to the Koopman operator. *arXiv preprint arXiv:1703.04680*, 2017.

[31] Milan Korda, Mihai Putinar, and Igor Mezić. Data-driven spectral analysis of the Koopman operator. *Applied and Computational Harmonic Analysis*, 48(2):599–629, 2020.

[32] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM-Society for Industrial and Applied Mathematics, USA, 2016.

[33] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

[34] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[35] Jaime Liew, Tuhfe Göçmen, Wai Hou Lio, and Gunner Chr. Larsen. Streaming dynamic mode decomposition for short-term forecasting in wind farms. *Wind Energy*, 25(4):719–734, 2022.

[36] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

[37] I. Mezić. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Reviews of Fluid Mechanics*, 45:357–378, 2013.

[38] H. D. Nguyen and J. Demmel. Reproducible tall-skinny QR. In *2015 IEEE 22nd Symposium on Computer Arithmetic*, pages 152–159, June 2015.

[39] S. Yu. Pilyugin. Theory of pseudo-orbit shadowing in dynamical systems. *Differential Equations*, 47(13):1929–1938, Dec 2011.

[40] S.Yu Pilyugin. *Shadowing in Dynamical Systems*, volume 1706 of *Lecture Notes in Mathematics*. Springer, 1999.

[41] M. J. D. Powell and J. K. Reid. On applying Householder transformations to linear least squares problems. In *Information Processing 68, Proc. International Federation of Information Processing Congress, Edinburgh, 1968*, pages 122–126. North Holland, Amsterdam, 1969.

[42] Lothar Reichel and William B Gragg. Algorithm 686: FORTRAN subroutines for updating the QR decomposition. *ACM Transactions on Mathematical Software (TOMS)*, 16(4):369–377, 1990.

[43] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[44] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, August 2010.

[45] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25(1):249–259, 2011.

[46] Peter J. Schmid. Application of the dynamic mode decomposition to experimental data. *Experiments in Fluids*, 50(4):1123–1130, 2011.

[47] Peter J. Schmid. Data-driven and operator-based tools for the analysis of turbulent flows. In Paul Durbin, editor, *Advanced Approaches in Turbulence*, pages 243–305. Elsevier, 2021.

[48] Peter J. Schmid. Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, 54(1):225–254, 2022.

[49] RK Singh and JS Manhas. *Composition operators on function spaces.* Number 179. North Holland, 1993.

[50] Stanimire Tomov, Jack Dongarra, and Marc Baboulin. Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Computing*, 36(5-6):232–240, June 2010.

[51] Stanimire Tomov, Rajib Nath, Hatem Ltaief, and Jack Dongarra. Dense linear algebra solvers for multicore with GPU accelerators. In *Proc. of the IEEE IPDPS'10*, pages 1–8, Atlanta, GA, April 19-23 2010. IEEE Computer Society. DOI: 10.1109/IPDPSW.2010.5470941.

[52] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[53] A. van der Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14:14–23, 1969.

[54] M. O Williams, I. G Kevrekidis, and C. W Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, June 2015.

[55] Hao Zhang, Clarence W. Rowley, Eric A. Deem, and Louis N. Cattafesta. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems*, 18(3):1586–1609, 2019.