A LAPACK implementation of the Dynamic Mode Decomposition II. The symmetric/Hermitian DMD (xSYDMD/xHEDMD)*

Zlatko Drmač†

December 1, 2022

Abstract

This note describes an LAPACK based implementation of the Dynamic Mode Decomposition (DMD) for data snapshots that are generated by real symmetric or complex Hermitian matrices. It is a second part of the recently published LAPACK Working Note 298. The new software implementation is suitable for applications where e.g. the physics of the underlying problem implies hermiticity and the computed Ritz values should be real with the mutually orthogonal corresponding Ritz vectors. Such a solution is not guaranteed by the standard DMD, which may result in non-physical solutions. We analyze the problem and show that adaptation to the Hermitian case has several fine and instructive numerical details. Our analysis also explains the numerics and reveals interesting details behind the physic-informed DMD (piDMD), that is recently introduced by P. J. Baddoo, B. Herrmann, B.J. McKeon, J. N. Kutz and S. L. Brunton.

1 Introduction

The Dynamic Mode Decomposition (DMD) is a computational tool for analysis of the structure of nonlinear dynamical systems, both in data driven scenarios and in computer simulations. Since its introduction by Schmid [20] in the context of computational fluid dynamics (CFD), the DMD has been attracting researchers from various fields and, as a result, it has become a versatile computational tool with many applications in robotics, aeroacustic, epidemiology, algorithmic trading on financial markets, video processing, neural networks, physics informed machine learning and many others; for an overview see [1], [4].

In a nutshell, the DMD works as follows. Suppose we are given a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m+1}$ in $\mathbb{R}^n$ (or in $\mathbb{C}^n$) such that $\mathbf{x}_{i+1} \approx \mathbf{A}\mathbf{x}_i$, with some inaccessible $n \times n$ matrix $\mathbf{A}$; that is, given is

*LAPACK Working Note 300. AIMdyn technical report AIMdyn-DARPA-SBIR-2022, November 2022. Distribution Statement A: Approved for Public Release, Distribution Unlimited.

†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.

a Krylov sequence of $\mathbf{A}$ with an initial vector $\mathbf{x}_1$, but $\mathbf{A}$ is unknown or difficult to apply. The $\mathbf{x}_i$'s may originate in a computer simulation or from measurements of some physical process. In general, $\mathbf{x}_i$ is the numerical value of a vector valued observable $\mathbf{f}$ evaluated at the state $\mathbf{s}_i$ of an underlying discrete dynamical system $\mathbf{s}_{i+1} = \mathbf{T}(\mathbf{s}_i)$, i.e. $\mathbf{x}_i = \mathbf{f}(\mathbf{s}_i)$, $\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{T}(\mathbf{s}_i)) \approx \mathbf{A}\mathbf{f}(\mathbf{s}_i)$. The task of the DMD is to identify k ($k \leq m$) approximate eigenpairs $(\lambda_j, \mathbf{z}_j)$ such that

$$\mathbf{A}\mathbf{z}_j \approx \lambda_j \mathbf{z}_j, \quad j = 1, \ldots, k, \tag{1}$$

and to analyze (decompose) the given sequence (dynamics) using the approximate eigenvectors (modes) $\mathbf{z}_j$ and the eigenvalues λ_j. In the seminal work [20], Schmid proposed a numerical scheme to compute the eigenpairs (1); this triggered an intensive research resulting in various modifications of the algorithm and its adaptation to a host of applications. The resulting decomposition can be used for an analysis of the structure of the dynamics (coherent structures of the flow [21], [22], [23], [17], [24], [14], [15], [19], [12]), or e.g. for forecasting and control, or for a model order reduction.

This seemingly simple matrix computation is rooted in a particular linearization of the underlying dynamics. Namely, to the dynamical system $\mathbf{s}_{i+1} = \mathbf{T}(\mathbf{s}_i)$, there is an associated linear operator $\mathcal{U}$ that, in an infinite dimensional function space of scalar observables, acts as the composition with $\mathbf{T}$, $\mathcal{U}f = f \circ \mathbf{T}$. The operator $\mathcal{U}$, called the Koopman operator, provides a particular global infinite dimensional linearization; see [5], [28], [26], [4]. If $\mathbf{f} = (f_1, \ldots, f_d)^T$ is a vector valued observable, then the composition operator is defined component-wise as $\mathcal{U}_d\mathbf{f} = (\mathcal{U}f_1, \ldots, \mathcal{U}f_d)^T$. The relation $\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{T}(\mathbf{s}_i)) \approx \mathbf{A}\mathbf{f}(\mathbf{s}_i)$ can be written as $\mathbf{x}_{i+1} = (\mathcal{U}_d\mathbf{f})(\mathbf{s}_i) = \mathbf{A}\mathbf{f}(\mathbf{s}_i) + \boldsymbol{\rho}(\mathbf{s}_i)$, i.e.

$$\mathbf{x}_{i+1} = (\mathcal{U}_d\mathbf{f})(\mathbf{s}_i) = \begin{pmatrix} \mathcal{U}f_1(\mathbf{s}_i) \\ \vdots \\ \mathcal{U}f_d(\mathbf{s}_i) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{T}(\mathbf{s}_i)) \\ \vdots \\ f_d(\mathbf{T}(\mathbf{s}_i)) \end{pmatrix} = \mathbf{A} \begin{pmatrix} f_1(\mathbf{s}_i) \\ \vdots \\ f_d(\mathbf{s}_i) \end{pmatrix} + \begin{pmatrix} \rho_1(\mathbf{s}_i) \\ \vdots \\ \rho_d(\mathbf{s}_i) \end{pmatrix} = \mathbf{A}\mathbf{f}(\mathbf{s}_i) + \boldsymbol{\rho}(\mathbf{s}_i), \tag{2}$$

where $\boldsymbol{\rho}(\cdot)$ is a residual. Now, the sequence $\mathbf{x}_1,\ \mathbf{x}_2 = \mathbf{A}\mathbf{x}_1,\ \mathbf{x}_3 = \mathbf{A}\mathbf{x}_2 = \mathbf{A}^2\mathbf{x}_1,\ \mathbf{x}_4 = \mathbf{A}^3\mathbf{x}_1,\ \ldots$ represents evaluation of an observable $\mathbf{f}$ along a trajectory initialized at $\mathbf{s}_1$ and it can be interpreted as a Krylov sequence of $(\mathcal{U}_d, \mathbf{f})$, evaluated at $\mathbf{s}_1$:

$$\mathbf{f}(\mathbf{s}_1),\ (\mathcal{U}_d\mathbf{f})(\mathbf{s}_1),\ (\mathcal{U}_d^2\mathbf{f})(\mathbf{s}_1),\ (\mathcal{U}_d^3\mathbf{f})(\mathbf{s}_1), \ldots$$

More generally, the data may be gathered from several trajectories with different initial conditions, so that we have a sequence of snapshot pairs $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^m$, $m \ll n$, such that $\mathbf{x}_i = \mathbf{f}(\mathbf{s}_i)$, $\mathbf{y}_i = \mathbf{f}(\mathbf{T}(\mathbf{s}_i))$. Then at $s = \mathbf{s}_i$ the relation (2) reads $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \boldsymbol{\rho}(\mathbf{s}_i)$. In matrix notation, this is compactly written as $\mathbf{Y} = \mathbf{A}\mathbf{X} + R$, where $R(:, i) = \boldsymbol{\rho}(\mathbf{s}_i)$. The DMD computes the eigenpairs (1) using the Rayleigh-Ritz extraction procedure with a suitable subspace of the range of $\mathbf{X}$.

Recently, we published a LAPACK implementation of the DMD [7], that is based on our earlier work [9], [6]. During the work on the code, in a discussion on the implementation, Julien Langou (University of Colorado at Denver) raised the question of the symmetric case, i.e. what is the best software solution if it is known that the matrix A that generated the data (2) is real symmetric or complex Hermitian. Using a general DMD code such as [7] is clearly not optimal because the Rayleigh-Ritz procedure does not use hermiticity – the eigenvalues and eigenvectors of the Rayleigh quotient are computed using general eigensolver (e.g. `xGEEV` from LAPACK), which does not guarantee the fundamental properties of a Hermitian eigenvalues problem, namely the real spectrum and an orthonormal system of eigenvectors.

This paper addresses that problem and proposes a LAPACK [2] based software solution to the symmetric/Hermitian DMD. It is derived from the recent implementation [7] of the

DMD. In the process of adaptation of the subroutines `xGEDMD` and `xGEDMDQ` (see [7]) to the Hermitian/symmetric case, we realized that the task was more than a mere *mutatis mutandis*. We had to deal with some subtle details that we describe in this note.

The material is organized as follows. For the reader's convenience, in §2 we review the details of the Schmid's DMD algorithm (§2.1) and its variation [9], [7] (§2.2). The Hermitian DMD is analyzed in detail in §3. We start in §3.1 with a theoretical remark on the residual bound that exploits the hermiticity and guarantees approximation of certain number of eigenpairs of the Hermitian matrix that generated the data. An analysis of the loss of hermiticity and a possible method of restoring it are given in §3.2. In §3.2.3 we briefly discuss how the proposed scheme extends to the skew-Hermitian case. A possible adaptation of the Exact DMD to the Hermitian case is outlined in §3.3. In §4 we analyze another approach to the symmetric/Hermitian DMD, the physics-informed DMD [3] based on a solution of the Hermitian Procrustes problem. Our analysis reveals interesting details about the structure of the computational errors and we also show that the hermiticity/symmetry constraint does not require explicit use of the concept of the Hermitian/symmetric Procrustes problem. In fact, if the data is generated by a Hermitian/symmetric operator then the DMD preserves hermiticity/symmetry without any additional modification and the only problem is how to handle the non-Hermitian perturbations incurred by the finite precision arithmetic. We show that the actual benefit of the formula for the solution of the Hermitian Procrustes problem is that it helps damping the perturbations. In §5, we show that the QR compressed DMD can be formulated for the Hermitian/symmetric case as well. In §6 we discuss our work in progress on a DMD for highly non-normal cases. The software implementation is briefly described in §8.

2 The DMD

With only $\mathbf{X}$ and $\mathbf{Y}$ at hand, $\mathbf{A}$ can be guessed by minimizing the residual $A\mathbf{X}-\mathbf{Y}$ in some convenient norm. The well developed least squares theory motivates taking an $\mathbb{A} \in \arg\min_A \|A\mathbf{X}-\mathbf{Y}\|_F$, and commonly used choice is the unique minimal norm solution[1] $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$, where $\|\cdot\|_F$ is the Frobenius matrix norm and $\mathbf{X}^\dagger$ is the Moore-Penrose generalized inverse of $\mathbf{X}$. Clearly, if B such that $B\mathbf{X} = \mathbf{0}$, then $(\mathbb{A}+B)\mathbf{X} = \mathbb{A}\mathbf{X}$, so that $\mathbb{A}+B$ is a solution as well. From $\mathbf{X}^*B^* = \mathbf{0}$, we see that the columns of B^* can be selected from the at least $(n-m)$-dimensional null space of $\mathbf{X}^*$. Following [7], set

$$[\mathbb{A}] = [\mathbb{A}; \mathbf{X}, \mathbf{Y}] = \arg\min_A \|A\mathbf{X} - \mathbf{Y}\|_F = \{\mathbb{A} + B \ : \ B\mathbf{X} = \mathbf{0}\}. \tag{3}$$

It may seem, at first, that we seek certain number of eigenpairs of a matrix that is not uniquely determined by the data $\mathbf{X}$, $\mathbf{Y}$. Independent of a particular least squares solution $A \in [\mathbb{A}] = [\mathbf{A}]$, we have $A\mathbf{X} = \mathbf{Y}\mathbf{X}^\dagger\mathbf{X} = \mathbf{Y}P_{\mathbf{X}^*}$, where $P_{\mathbf{X}^*}$ is the orthogonal projector onto the range of $\mathbf{X}^*$. The relation $A\mathbf{X} = \mathbf{Y}P_{\mathbf{X}^*}$ should be at the center of any further considerations because it contains information on what A does in the range of $\mathbf{X}$, independent of what particular $A \in [\mathbb{A}]$ is selected. With a slight abuse of notation, we can write $[\mathbb{A}]\mathbf{X} = \mathbf{Y}P_{\mathbf{X}^*}$; if $\mathbf{X}$ is of full rank, then $[\mathbb{A}]\mathbf{X} = \mathbf{Y}$.

Remark 2.1 Focusing to $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$ has potential for a fallacy which lies behind the adjective *exact*. Suppose our data is generated by an inaccessible matrix $\mathbf{A}$ so that $\mathbf{Y} = \mathbf{A}\mathbf{X}$. Since $\mathbb{A}\mathbf{Y} = \mathbf{Y}(\mathbf{X}^\dagger\mathbf{Y})$, the range of $\mathbf{Y}$ is $\mathbb{A}$-invariant and all corresponding eigenvalues and eigenvectors (from

[1] Another choice is selecting a sparse solution as implemented e.g. in the Matlab *backslash* operator for least squares solutions; for a discussion how different solution methods affect the DMD see [10].

the range of $\mathbf{Y}$) can be computed without error (barring finite precision arithmetic errors) as it is done in the Exact DMD, but this exactness does not hold for $\mathbf{A}$. See §3.3 and [7].

2.1 Schmid's DMD

Computing approximate eigenpairs $(\lambda_j, \mathbf{z}_j)$ of $\mathbf{A}$ is thus based only on the information on its action in the range of $\mathbf{X}$, and this calls for the Rayleigh-Ritz extraction procedure. Computationally, working with an orthonormal subspace basis is preferable, and this is the key in the DMD algorithm, proposed by Schmid [20]. In the sequel, we review the DMD algorithm, and in addition, we formulate the relevant formulas in terms of the linear manifold $[\mathbb{A}]$, which will be important for the symmetric/Hermitian case. We use the results and ideas from [9] and [7].

Let r be the rank of $\mathbf{X}$ and let $\mathbf{X} = U\Sigma V^*$ be the economy-size SVD of $\mathbf{X}$; $\Sigma = \mathrm{diag}(\sigma_j)_{j=1}^{r}$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$, $U^*U = V^*V = \mathbb{I}_r$. Since $\mathbf{X}^\dagger = V\Sigma^{-1}U^*$, $\mathbf{X}^\dagger\mathbf{X} = P_{\mathbf{X}^*} = VV^*$, we can write $\mathbb{A}\mathbf{X} = \mathbf{Y}\mathbf{P}_{\mathbf{X}^*}$ as $\mathbb{A}U\Sigma V^* = \mathbf{Y}VV^*$, and then $\mathbb{A}U\Sigma = \mathbf{Y}V$. As a Krylov matrix, $\mathbf{X}$ is expected to be ill-conditioned and even if of full rank $r = m$ it may have tiny singular values $\sigma_i \ll \sigma_1 = \|\mathbf{X}\|_2$, that are difficult to compute numerically. The problem is mitigated using a truncated SVD. The truncation is defined with a user supplied tolerance $\boldsymbol{\tau}$ and a numerical rank of $\mathbf{X}$ is set as

$$k = \max\{i \ : \ \sigma_i > \sigma_1\boldsymbol{\tau}\}. \tag{4}$$

The tolerance level can be set as e.g. $\boldsymbol{\tau} = n\boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is the round-off unit of the working machine precision, or it can be determined based on the noise level in the data. The choice $\boldsymbol{\tau} = 0$ means that all nonzero singular values will be counted, no matter how tiny they are. In addition, a user may set the parameter k directly, based on some prior information.

Let $U_k = U(:, 1:k)$, $V_k = V(:, 1:k)$, $\Sigma_k = \Sigma(1:k, 1:k)$. Then $\mathbb{A}U_k = \mathbf{Y}V_k\Sigma_k^{-1}$, and since $\mathrm{range}(U_k) \subseteq \mathrm{range}(\mathbf{X})$, we can write[2]

$$[\mathbb{A}]U_k = \mathbf{Y}V_k\Sigma_k^{-1}. \tag{5}$$

Hence, the Rayleigh quotient $S_k = U_k^*\mathbb{A}U_k$ with respect to the range of U_k can be expressed as

$$S_k = U_k^*\mathbb{A}U_k = U_k^*\mathbf{Y}_m V_k\Sigma_k^{-1} \ \ (U_k^*[\mathbb{A}]U_k = \{S_k\}). \tag{6}$$

Now we have all ingredients for Rayleigh-Ritz approximations. Let $S_k w_j = \lambda_j w_j$ with $\|w_j\|_2 = 1$, $j = 1, \ldots, k$. Then $\mathbb{A}(U_k w_j) \approx \lambda_j(U_k w_j)$, and we can also write

$$[\mathbb{A}](U_k w_j) = \lambda_j(U_k w_j) + (\mathbb{I}_n - U_k U_k^*)[\mathbb{A}]U_k w_j, \tag{7}$$

i.e. despite our particular choice of $\mathbb{A}$ that connects $\mathbf{X}$ and $\mathbf{Y}$, the computed eigenpairs are well determined for $[\mathbb{A}]$, which includes $\mathbf{A}$.

2.2 xGEDMD and xGEDMDQ

In [9] we introduced some enhancements of the DMD, and implemented them in the subroutines `xGEDMD` and `xGEDMDQ` in [7]. These subroutines are a basis for the symmetric DMD proposed in this paper, and for the reader's convenience we briefly outline the features that are incorporated in the new software.

[2] In our notation, this means that any matrix from $[\mathbb{A}]$ can be used instead of $\mathbb{A}$, including the inaccessible $\mathbf{A}$.

Computable residuals. The residual $R_k = \mathbb{A}U_k - U_k S_k$ is available and $(\mathbb{A} - R_k U_k^*)U_k = U_k S_k$, i.e. the range of U_k is invariant for a perturbed $\mathbb{A} + \delta\mathbb{A}$, $\|\delta\mathbb{A}\|_2 = \|R_k\|_2$, but there is no guarantee that all computed Ritz pairs are good approximations. One way to select good ones is to use computable residuals $r_j = \mathbb{A}z_j - \lambda_j z_j$,

$$r_j = (\mathbb{I}_n - U_k U_k^*)\mathbb{A}(U_k w_j) = \mathbf{Y}V_k\Sigma_k^{-1}w_j - \lambda_j(U_k w_j), \tag{8}$$

where we can set $[\mathbb{A}]$ instead of $\mathbb{A}$. This formula is first used in [9] and proved to be useful in many applications. For each λ_j we can use the fact that

$$(\mathbb{A} - r_j z_j^*)z_j = \lambda_j z_j,$$

and then, by the Bauer-Fike theorem, assuming that the data generator matrix A is diagonalizable with the eigenvectors matrix S and eigenvalues $\alpha_1, \ldots, \alpha_n$,

$$\min_{\alpha_i} |\lambda_j - \alpha_i| \leq \kappa_2(S)\|r_j\|_2.$$

That is, each λ_j approximates some α_{i_j}, but in general there is no guarantee that k eigenvalues of $\mathbf{A}$ are approximated. In the Hermitian case, we can say much more, as we outline in §3.1.

Scaling the data matrices. Optional scaling of the data matrices replaces $\mathbf{X}$, $\mathbf{Y}$ with $\mathbf{X}_c = \mathbf{X}D_{\mathbf{X}}^{-1}$, $\mathbf{Y}_c = \mathbf{Y}D_{\mathbf{X}}^{-1}$, respectively, where $D_{\mathbf{X}} = \mathrm{diag}(\|\mathbf{X}(:,i)\|_2)_{i=1}^m$. If data scaling is allowed, this improves the numerical robustness of the method. It is motivated by the following theorem:

Theorem 2.2 *(Van der Sluis [27]) 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 $\kappa_2(\mathbf{X}_c) \leq \sqrt{m}\min_{D=\mathrm{diag}}\kappa_2(\mathbf{X}D)$.*

As discussed in [7], the effect of scaling is twofold: it allows for a more accurate SVD and a narrower range of the singular values in many cases allows for a larger k.

The structure of the method is given in Algorithm 1. The remaining two technical details in the implementation [7] are the Exact DMD vectors and the QR compressed implementation of the DMD. These themes will be presented in detail in §3.3 and §5 , respectively.

Algorithm 1: $(Z_k, \Lambda_k, r_k, [B_k], [Z_k^{(ex)}]) = \mathrm{xGEDMD}(\mathbf{X}, \mathbf{Y}; \boldsymbol{\tau})$

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 (4).

1: $D_{\mathbf{X}} = \mathrm{diag}(\|\mathbf{X}(:,i)\|_2)_{i=1}^m$; $\mathbf{X}_c = \mathbf{X}D_{\mathbf{X}}^{\dagger}$; $\mathbf{Y}_c = \mathbf{Y}D_{\mathbf{X}}^{\dagger}$.
2: $[U, \Sigma, V] = svd(\mathbf{X}_c)$; {*The thin SVD:* $\mathbf{X}_c = U\Sigma V^*$, $U \in \mathbb{C}^{n\times m}$, $\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^m$}
3: Determine numerical rank k, using (4) 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: $B_k = \mathbf{Y}_c(V_k\Sigma_k^{-1})$; {*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: $[W_k, \Lambda_k] = \mathrm{eig}(S_k)$ {$\Lambda_k = \mathrm{diag}(\lambda_i)_{i=1}^k$; $S_k W_k(:,i) = \lambda_i W_k(:,i)$; $\|W_k(:,i)\|_2 = 1$}
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 (8).*}

Output: Z_k, Λ_k, r_k, $[B_k]$, $[Z_k^{(ex)}]$.

---

3 Hermitian/symmetric case

Suppose we know a priori that there is a Hermitian matrix $\mathbf{H} = \mathbf{H}^*$ such that $\mathbf{H}\mathbf{X} = \mathbf{Y}P_{\mathbf{X}^*}$, i.e. $\mathbf{H} \in \arg\min_A \|A\mathbf{X} - \mathbf{Y}\|_F$. Then $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$ is in general not Hermitian[3] but, as we showed in §2.1, since $\mathbf{H} \in [\mathbb{A}]$, the Rayleigh quotient satisfies $S_k = U_k^*\mathbb{A}U_k = U_k^*\mathbf{H}U_k = S_k^*$. Note that in terms of the linear least squares solution manifold, $[\mathbb{A}] = [\mathbf{H}]$.

3.1 Residual bounds

Let $S_k = W\Lambda W^*$ be the spectral decomposition of S_k, with unitary W ($W^*W = \mathbb{I}_k$) and $\Lambda = \mathrm{diag}(\lambda_j)_{j=1}^k$, $\lambda_1 \geq \cdots \geq \lambda_k$. Then the Ritz vectors $Z_k = (z_1, \ldots, z_k) = U_kW$ are orthonormal ($Z_k^*Z_k = \mathbb{I}_k$) and for each pair (λ_j, z_j) we can compute its corresponding residual (8) and decide whether to accept or reject it, depending on a given tolerance threshold.

Here too, we can compute the residual $R_k = \mathbf{H}U_k - U_kS_k$ and check that $U_k^*R_k = \mathbf{0}$ and

$$(\mathbf{H} - (R_kU_k^* + U_kR_k^*))Z_k = Z_kS_k$$

This means that with $\delta\mathbf{H} = -(R_kU_k^* + U_kR_k^*)$, the Rayleigh-Ritz procedure ensures precisely k eigenpairs of $\mathbf{H} + \delta\mathbf{H}$ (counted with multiplicities) and $\|\delta\mathbf{H}\|_2 = \|R_k\|_2$. More precisely, there are k eigenvalues $\alpha_{i_1}, \ldots, \alpha_{i_k}$ of H (counted with multiplicities) such that

$$\max_{j=1:k} |\lambda_j - \alpha_{i_j}| \leq \|\mathbf{Y}_c(V_k\Sigma_k^{-1}) - U_kS_k\|_2.$$

Here we used, in terms of Algorithm 1, that $\mathbf{H}U_k = \mathbf{Y}_c(V_k\Sigma_k^{-1})$. For details of the Rayleigh-Ritz theory see e.g. [18, §11.5], [25, §3.4].

This means that (barring finite precision computation errors) the DMD algorithm will automatically exploit the underlying symmetry, and the computed Ritz pairs will have the proper structure – real Ritz values and orthonormal Ritz vectors. There is no need to determine a Hermitian $H \in \arg\min_A \|A\mathbf{X} - \mathbf{Y}\|_F$; the prior assumption/knowledge that the data is generated by a symmetric/Hermitian matrix $\mathbf{H}$ suffices. Note also that the Rayleigh quotient also inherits the positive (semi)definiteness of $\mathbf{H}$.

3.2 Correcting the loss of symmetry in finite precision arithmetic

Unfortunately, in a software implementation the main ingredients in the formula $S_k = U^*\mathbf{Y}V_k\Sigma_k^{-1}$ are the elements of the SVD of a potentially highly ill-conditioned data matrix $\mathbf{X}$. As a result, the computed matrix $\widetilde{S}_k$ will not be symmetric/Hermitian and, if this issue is ignored, its eigenvalues may be computed as complex and the orthogonality of the eigenvectors may be lost. This may be the case if we e.g. in Matlab use the function `eig` that will detect the non-symmetry and use a non-symmetric/non-Hermitian eigensolver. In another situation, when using LAPACK, we may ignore the possible loss of symmetry and simply call the symmetric/Hermitian subroutines `xSYEV`/`xHEEV` with an option `'U'` (or `'L'`) to indicate that the matrix is implicitly given by its upper (or lower) triangle and assumed symmetric/Hermitian. This ignores a possibility that the content of the lower (upper) triangle may indicate substantial loss of symmetry. The question is what Hermitian matrix we should use instead of $\widetilde{S}_k$, and how to justify a particular choice.

[3] This is another argument that $\mathbb{A}$ should be considered as just an element of $[\mathbf{H}] = [\mathbb{A}]$.

A natural way to correct $\widetilde{S}_k$ and restore hermiticity is to replace it with a close Hermitian matrix. By a classical result of Fan and Hoffman [11], the matrix

$$\widehat{S}_k = \frac{1}{2}(\widetilde{S}_k + \widetilde{S}_k^*) \tag{9}$$

satisfies, for any unitarily invariant norm $\|\cdot\|$,

$$\|\widetilde{S}_k - \widehat{S}_k\| = \min_{H=H^*} \|\widetilde{S}_k - H\|. \tag{10}$$

This optimality of $\widehat{S}_k$ in a large class of norms, as well as its simple computation (9), makes it a good candidate to replace $\widetilde{S}_k$. Is this the best we can do? Is it superfluous to ask whether is it best to chose the optimal approximation $\widehat{S}_k$, or, what could go wrong if we replaced $\widetilde{S}_k$ with its closest Hermitian matrix?

3.2.1 On errors in the computed Rayleigh quotient $\widetilde{S}_k$

Although the choice of $\widehat{S}_k$ is optimal, there is a small snag: it is the closest Hermitian matrix to a non-Hermitian approximation $\widetilde{S}_k$ of S_k, and if the errors in $\widetilde{S}_k$ have a particular structure being close to $\widetilde{S}_k$ does not guarantee being close to S_k. If the error is big e.g. in some entries in the upper triangle, and small in the symmetric positions in the lower triangle, then $\widehat{S}_k$ will in some sense symmetrize and average the error at both sets of entries and using its spectral decomposition in the Rayleigh-Ritz procedure may introduce large errors.

The main source of errors is the SVD of $\mathbf{X}$, in particular because the columns of $\mathbf{X}$ are taken from one of more Krylov sequences, possibly with initial vectors with norms that span several orders of magnitude. To gain some insights in the errors in the computed $\widetilde{S}_k$, one can use perturbation theory to estimate the errors in the computed SVD $\mathbf{X} \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$, i.e. to bound the errors in the singular vectors (the columns of $\widetilde{U}$ and $\widetilde{V}$) and in the singular values (diagonal entries of $\widetilde{\Sigma}$). Then, the rounding errors in the computation of $\widetilde{S}_k = \mathit{computed}(\widetilde{U}_k^*\mathbf{Y})\widetilde{V}_k\widetilde{\Sigma}_k^{-1})$ are estimated and, altogether, one can bound the error $S_k - \widetilde{S}_k$. This is a tedious work that we omit; in fact we will show that such an analysis is needed. Instead, we follow [7], that is based on the backward error analysis, and use it to our purpose of assessing the loss of symmetry in $\widetilde{S}_k$.

The numerically computed SVD $\mathbf{X} \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$ can be interpreted as

$$(\mathbf{X} + \delta\mathbf{X})\widetilde{V} = \widetilde{U}\widetilde{\Sigma}, \quad \|\delta\mathbf{X}\|_2 \leq \epsilon_x\|\mathbf{X}\|_2, \tag{11}$$

where $\widetilde{U}$ and $\widetilde{V}$ are numerically unitary matrices, $\|\widetilde{U}^*\widetilde{U} - \mathbb{I}_n\|_2 \leq \epsilon_u$, $\|\widetilde{V}^*\widetilde{V} - \mathbb{I}_n\|_2 \leq \epsilon_v$, and $\widetilde{\Sigma} = \mathrm{diag}(\widetilde{\sigma}_j)_{j=1}^n$. The error bounds $\epsilon_u, \epsilon_v, \epsilon_x$ depend on the details of a particular algorithm and its software implementation, and can be estimated by expressions of the form $f(m,n)\boldsymbol{\varepsilon}$, where $f(m,n)$ is a modestly growing function of the dimensions and $\boldsymbol{\varepsilon}$ is the round-off unit of the working precision. In [7], we showed that the backward error (11) can be estimated with a sharper snapshot-wise manner if a particular SVD algorithm is used, but for the sake of brevity we use (11), which holds e.g. for the Matlab function `svd()`.

Since $\widetilde{U}_k$ is not orthonormal, the Rayleigh-Ritz procedure requires working with the pencil $\widetilde{U}_k^*\mathbf{H}\widetilde{U}_k - \lambda\widetilde{U}_k^*\widetilde{U}_k$, but since $\widetilde{U}_k^*\widetilde{U}_k$ is almost identity, the perturbation theory allows to proceed with $\widetilde{U}_k^*\mathbf{H}\widetilde{U}_k - \lambda\mathbb{I}_k$ – the errors caused by this simplification are much smaller than the ones already introduced up to this point. Note that for the purpose of finding approximate eigenpairs of $\mathbf{H}$ we do not need to assess the accuracy of $\widetilde{U}_k$; we only need an accurate (data driven) application of the Rayleigh-Ritz extraction procedure using the range of $\widetilde{U}_k$.

Assume for simplicity that $\mathbf{X}$ and $\mathbf{X}+\delta\mathbf{X}$ are of full column rank; although $\mathbf{X}$ may be ill-conditioned this assumption is reasonable.[4] Then $P_{\mathbf{X}^*} = \mathbb{I}_n$ and

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

Hence, even if we could compute without roundoff errors $\widetilde{S}_k = \widetilde{U}_k^*\mathbf{Y}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$, it would differ from the Hermitian Rayleigh quotient $\widetilde{U}_k^*\mathbf{H}\widetilde{U}_k$, with an error $\delta\widetilde{S}_k = \widetilde{U}_k^*E_k$, where $E_k = \mathbf{H}\delta\mathbf{X}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$ is the error in the approximation of $\mathbf{H}\widetilde{U}_k$. We can estimate E_k as follows:

$$\frac{\|E_k(:,j)\|_2}{\|\mathbf{H}\|_2} \leq \|\delta\mathbf{X}\|_2\|\widetilde{V}_k(:,j)\|_2/\widetilde{\sigma}_j \leq \epsilon_x\sigma_1\|\widetilde{V}_k(:,j)\|_2/\widetilde{\sigma}_j \leq \epsilon_x\frac{\sqrt{1+\epsilon_v}}{1-\epsilon_x}\frac{\widetilde{\sigma}_1}{\widetilde{\sigma}_j} \tag{12}$$

$$\frac{\|E_k(:,j)\|_2}{\|\mathbf{H}\widetilde{U}_k(:,j)\|_2} \leq \frac{\|\mathbf{H}\|_2}{1/\|(\mathbf{H}_{|\mathrm{range}(U_k)})^\dagger\|_2}\epsilon_x\frac{\sqrt{1+\epsilon_v}}{\sqrt{1-\epsilon_u}(1-\epsilon_x)}\frac{\widetilde{\sigma}_1}{\widetilde{\sigma}_j}, \tag{13}$$

and then the column-wise errors in $\widetilde{S}_k$ as

$$\frac{\|\delta S_k(:,j)\|_2}{\|\mathbf{H}\|_2} \leq \frac{\|\widetilde{U}_k^*\mathbf{H}\|_2}{\|\mathbf{H}\|_2}\epsilon_x\frac{\sqrt{1+\epsilon_v}}{1-\epsilon_x}\frac{\widetilde{\sigma}_1}{\widetilde{\sigma}_j}.$$

These bounds indicate that the accuracy in $\widetilde{S}_k$ may be deteriorating with the increased column index, which means that the upper triangle of $\widetilde{S}_k$ may be more exposed to the effects of $\delta\mathbf{X}$ (i.e. the errors in the computation of the SVD of $\mathbf{X}$) and the column scaling by $\widetilde{\Sigma}_k^{-1}$. Clearly, the accuracy of the computed residual will be also affected, and the above analysis gives an estimate. This simple model can also be used to assess the effects of the noise $\Delta\mathbf{X}$, $\Delta\mathbf{Y}$ in the initial data.

Note that the above analysis does not include rounding errors in the computation $\widetilde{U}_k^*\mathbf{Y}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$, because they are not the main source of the loss of symmetry.

Let us try to test this analysis and its prediction using a numerical example.

Example 3.1 In this synthetic example we first generate a random real symmetric matrix $\mathbf{H}$ and use it to generate real data $\mathbf{X}$, $\mathbf{Y}$, with $n = 100$, $m = 44$. The truncation index $k = 42$ is determined using $\boldsymbol{\tau} = \boldsymbol{\varepsilon}$. We will use $\mathbf{H}$ explicitly (only for the purpose of the test) to compute the errors E_k and δS_k and to compare them with the predicted estimates. The algorithm and the error estimates do not have access to $\mathbf{H}$. We do not use concrete upper bounds for $\epsilon_x, \epsilon_u, \epsilon_v$ because they depend on a particular algorithm implementation. Instead, to estimate the trend of the errors we set $\epsilon_x = \boldsymbol{\varepsilon}$ and ignore all other errors.

We first consider E_k. The results shown in Figure 1 are instructive.

[4] The rank deficient case is only technically more involved.

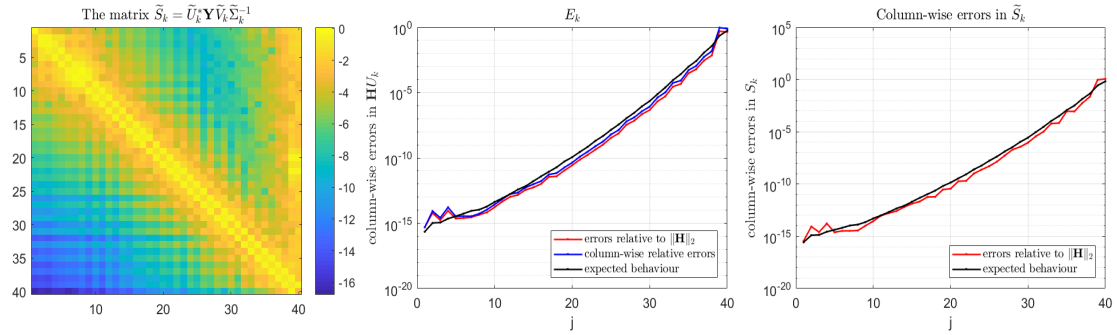

Figure 1: (Example 3.1) *First panel:* the structure of the computed $\widetilde{S}_k = \widetilde{U}^* \mathbf{Y} \widetilde{V}_k \widetilde{\Sigma}_k^{-1}$, visualized using `imagesc(log10(abs(`$\widetilde{S}_k$`)))`. The loss of symmetry is apparent. *Middle panel:* the column norms of E_k and their predicted trend. *Third panel:* the columns of $\delta\widetilde{S}_k$ and their predicted trend. Except for the error at the noise level $m\boldsymbol{\varepsilon}$, the analysis correctly reveals/predicts the behavior of the error.

For the sake of the experiment and better understanding of the structure of the errors in $\widetilde{S}_k = \widetilde{U}_k^* \mathbf{Y} \widetilde{V}_k \widetilde{\Sigma_k}^{-1}$, we compare $\widetilde{S}_k$ with the explicitly computed $S_k = \widetilde{U}_k^* \mathbf{H} \widetilde{U}_k$ entry-wise, i.e. we compute the relative error in each matrix entry. Further, motivated by the above analysis and the results of numerical experiment, we define

$$\widetilde{H}_k = \mathrm{diag}((\widetilde{S}_k)_{ii})_{i=1}^{k} + \widetilde{L}_k + \widetilde{L}_k^*, \tag{14}$$

where $\widetilde{L}_k$ is the strict lower triangle of $\widetilde{S}_k$, and considering as a candidate to replace $\widetilde{S}_k$. We note here that requiring entry-wise small relative errors is indeed too much to ask, but nevertheless we check them to test whether the above analysis correctly identifies the problem. The result shown in Figure 2 are precisely as predicted.

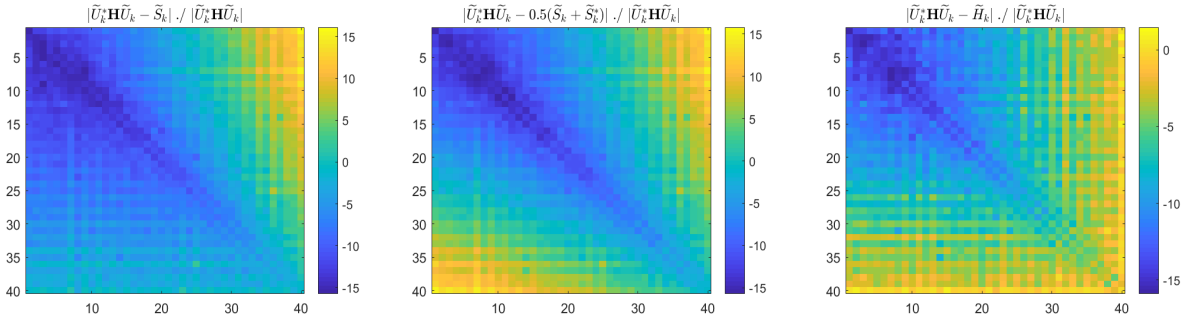

Figure 2: (Example 3.1) The entry-wise relative errors $\log_{10}(|(S_k)_{ij} - (\widetilde{S}_k)_{ij}|/|(S_k)_{ij}|)$ (*first panel*) and $\log_{10}(|(S_k)_{ij} - (\widehat{S}_k)_{ij}|/|(S_k)_{ij}|)$ (*second panel*). Note that the upper triangle of $\widetilde{S}_k$ has large error that is symmetrized in $\widehat{S}_k = 0.5(\widetilde{S}_k + \widetilde{S}_k^*)$ and transplanted into the lower triangle. The *third panel* shows the entry-wise errors in $\widetilde{H}_k$, which indicates that using $\widetilde{H}_k$ may be better than $\widehat{S}_k$. (Note that the scale in the color bar of this panel is different from the first two.)

Using smaller number k of the leading singular values and vectors produces more accurate $\mathbf{Y}\widetilde{V}_k\widetilde{\Sigma_k}^{-1}$, but such an aggressive truncation causes loss of spectral information as $\mathbf{H}$ is compressed onto a much lower dimensional subspace.

Remark 3.2 If $\mathbf{H}$ is positive (semi)definite, then S_k inherits the definiteness, but in ill-conditioned cases a symmetrizer of $\widetilde{S}_k$ is not guaranteed to be positive (semi)definite. Under this implicit assumption on (semi)definiteness, if we compute the spectral decomposition of

$\widetilde{H}_k$ (or any other symmetrizer) and if some of the eigenvalues are negative, we can replace them with zeros, thus implicitly replacing $\widetilde{H}_k$ with the closest positive semidefinite matrix.[5] Depending on the user's preferences, all or only positive Ritz values can be returned.

3.2.2 Software implementation details

In LAPACK, using the symmetrizer (14) is easy to implement: it suffices to set the `UPLO` parameter to `'L'` in the selected eigenvalue solver `xSYEV`/`xHEEV` or `xSYEVD`/`xHEEVD`. Another practical advantage of this is that it does not combine a matrix with its transposed.[6]

This shows the importance of a seemingly trivial detail of choosing whether to use the upper or the lower triangle of a computed apparently Hermitian/symmetric matrix.

The new LAPACK subroutines for the Hermitian/symmetric DMD are designated as `xSYDMD` (`x` $\in$ {`S`, `D`}) and `xHEDMD` (`x` $\in$ {`C`, `Z`}), following the LAPACK subroutine naming convention.

3.2.3 A remark on the skew-Hermitian case

In the case of a skew-Hermitian $\mathbf{H} = -\mathbf{H}^*$, we have $S_k = -S_k^*$ and the Ritz values will be purely imaginary and with orthonormal vectors. LAPACK does not have a special subroutine for the spectral decomposition of skew-Hermitian matrices. A quick fix, at least in the complex skew-Hermitian case is to rotate $\mathbf{H}$ and apply a Hermitian solver (e.g. `xHEEV` or `xHEEVD`) to $\mathbf{i}S_k$ (a Rayleigh quotient of $\mathbf{i}\mathbf{H}$). The computed real eigenvalues are then rotated back to the imaginary axis, and the eigenvectors are used unchanged. Equivalently, the data $\mathbf{Y}$ can be changed to $\mathbf{i}\mathbf{Y}$, making the problem Hermitian, and the computed real eigenvalues should be rotated back to the imaginary axis..

3.3 On a Hermitian Exact DMD

The Exact DMD (EDMD) [26, §2.2, §2.3] is a formulation of the DMD often used in the literature. In [7], we showed that the Exact DMD eigenvectors can be interpreted as a result of one step of the power iterations with (the unknown) matrix $\mathbf{H}$ applied to the eigenvectors (i.e. Ritz vectors) computed by the original DMD algorithm. This is evident from the lines 5, 8, 9 of Algorithm 1.

In the Hermitian case, the Ritz vectors $Z_k = U_k W_k = (\mathbf{z}_1, \ldots, \mathbf{z}_k)$ (line 8 in Algorithm 1) will be numerically orthonormal (see §3.2), but the EDMD vectors $Z_k^{(ex)} = (\mathbf{z}_1^{(ex)}, \ldots, \mathbf{z}_k^{(ex)}) = \mathbf{H}Z_k$ are not guaranteed to be mutually numerically orthogonal. Hence, if we want to use the EDMD vectors in a Hermitian DMD, we need an additional step of orthogonalization. Such an orthogonalization, using the Gram-Schmidt algorithm, replaces the vectors $\mathbf{z}_j^{(ex)}$ with a numerically orthonormal set $\widehat{\mathbf{z}}_1^{(ex)}, \ldots, \widehat{\mathbf{z}}_k^{(ex)}$, but this change of the vectors will change the residuals. Note that we cannot adjust the Ritz values and use the optimality of the Rayleigh quotient $(\widehat{\mathbf{z}}_j^{(ex)})^* \mathbf{H} \widehat{\mathbf{z}}_j^{(ex)} / ((\widehat{\mathbf{z}}_j^{(ex)})^* \widehat{\mathbf{z}}_j^{(ex)})$ because the data does not contain information to compute $\mathbf{H}\widehat{\mathbf{z}}_j^{(ex)}$. Another possibility is to replace $Z_k^{(ex)}$ with the closest unitary matrix. Following Kahan [16], the nearest orthonormal matrix is obtained from the economy-size SVD $Z_k^{(ex)} = \Phi \Sigma_Z \Psi^*$ as $\Omega_k^{(ex)} = \Phi\Psi^*$, that is the unique closest orthonormal matrix to $Z_k^{(ex)}$ in both the spectral

[5] Recall that we cannot talk about the closest positive definite matrix because the set of positive definite matrices is open.

[6] It would be convenient to have a BLAS 3 subroutine that returns only the lower or the upper triangle of a product of two matrices.

and the Frobenius norm. Note that changing the order of columns of $Z_k^{(ex)}$ changes the closest orthonormal matrix with the same column permutation.

Example 3.3 We compare the residuals of the Ritz pairs as returned by DSYDMD and explicitly computed using **H**. The residuals of the EDMD eigenpairs $(\lambda_j, \mathbf{z}_j^{(ex)})$ are computed explicitly using **H**. Then we perform the following experiment: We first orthogonalize (using the Gram-Schmidt algorithm) the EDMD vectors in order in which they are received and then in order of non-decreasing residuals. As the third option, we replace $Z_k^{(ex)}$ with $\Omega_k^{(ex)}$. In all three cases the vector keep their originally assigned Ritz values, and we compute the new residuals of $(\lambda_j, \widehat{\mathbf{z}}_j^{(ex)})$ explicitly, using **H**. The results are shown in Figure 3.

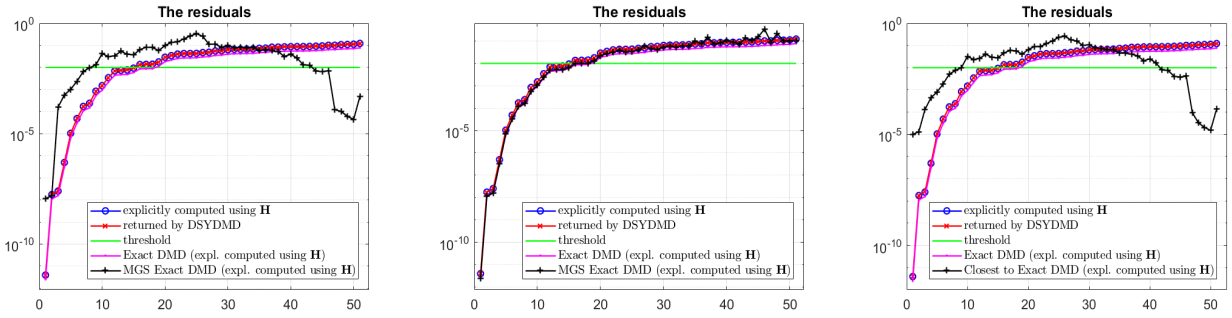

Figure 3: (Example 3.3) *First panel:* The residuals of $(\lambda_j, \widehat{\mathbf{z}}_j^{(ex)})$ with the orthogonalization of the EDMD vectors in order computed in DSYDMD. *Second panel:* The residuals of $(\lambda_j, \widehat{\mathbf{z}}_j^{(ex)})$ when the EDMD vectors are ordered with non-decreasing residuals. *Third panel:* The residuals when the EDMD vectors are replaced with the closest unitary matrix $\Omega_k^{(ex)}$. The horizontal green line shows a user specified threshold of 10^{-2}.

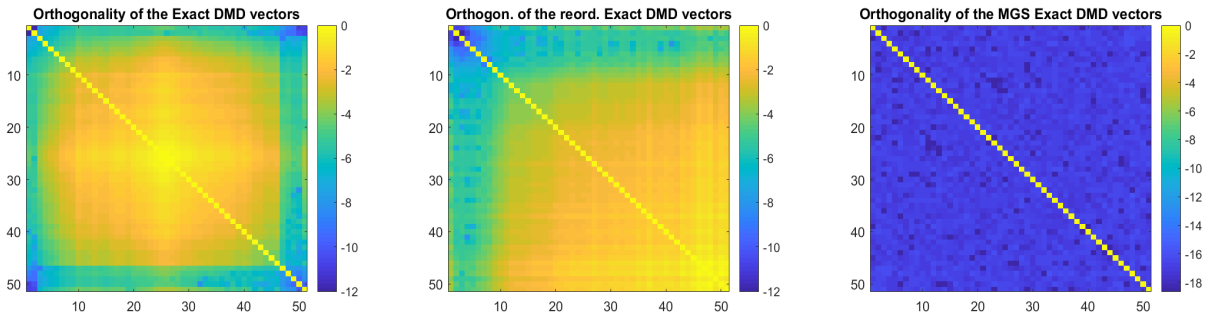

Figure 4: (Example 3.3) *First panel:* Orthogonality of the normalized EDMD vectors in order of computation in DSYDMD. The data shown is $\log_{10}[|(\mathbf{z}_i^{(ex)})^* \mathbf{z}_j^{(ex)}|/(\|\mathbf{z}_i^{(ex)}\|_2 \|\mathbf{z}_j^{(ex)}\|_2)]$. *Second panel:* The data form the first panel with the EDMD vectors reordered to get non-decreasing residuals. Compare these two panels with the first two panels in Figure 3. *Third panel:* The orthogonality of the new vectors $\widehat{\mathbf{z}}_j^{(ex)}$, the data shown is $\log_{10}[|(\widehat{\mathbf{z}}_i^{(ex)})^* \widehat{\mathbf{z}}_j^{(ex)}|/(\|\widehat{\mathbf{z}}_i^{(ex)}\|_2 \|\widehat{\mathbf{z}}_j^{(ex)}\|_2)]$.

Remark 3.4 Correcting a loss of orthogonality of a computed set of eigenvectors is a well studied theme, but in the context of a data driven DMD application, our maneuvering space is limited because we are only given how **H** transforms the range of **X**.

3.3.1 A discussion

One can argue that using the Gram-Schmidt orthogonalization on the EDMD vectors ordered to have non-decreasing residuals of the Ritz pairs is, given the data driven setting, in a sense

optimal. First note that this ordering is not based on the residuals corresponding to the EDMD vectors, because computing those residuals is not feasible. However, based on [7], we know that the EDMD residuals behave similarly, so that this ordering lists the EDMD vectors in order of the quality measured by the residual. We ease the notation and assume that $\mathbf{z}_1^{(ex)}, \ldots, \mathbf{z}_k^{(ex)}$ are already listed in that manner. Further, we assume that they are all normalized to unit Euclidean length. Hence, $\mathbf{z}_1^{(ex)}$ is the most valuable EDMD vector and we want to keep its direction (span), and the first step of the Gram-Schmidt orthogonalization is $\widehat{\mathbf{z}}_1^{(ex)} = \mathbf{z}_1^{(ex)}$. Because the eigenvectors should be orthonormal and $\widehat{\mathbf{z}}_1^{(ex)}$ is accepted as a good approximate eigenvector, no other eigenvector should have any component in its direction, i.e. we update them as

$$\mathbf{z}_2^{(ex)} \leftarrow \mathbf{z}_2^{(ex)} - \widehat{\mathbf{z}}_1^{(ex)} (\widehat{\mathbf{z}}_1^{(ex)})^* \mathbf{z}_2^{(ex)},\ \mathbf{z}_3^{(ex)} \leftarrow \mathbf{z}_3^{(ex)} - \widehat{\mathbf{z}}_1^{(ex)} (\widehat{\mathbf{z}}_1^{(ex)})^* \mathbf{z}_3^{(ex)},\ \ldots,\ \mathbf{z}_k^{(ex)} \leftarrow \mathbf{z}_k^{(ex)} - \widehat{\mathbf{z}}_1^{(ex)} (\widehat{\mathbf{z}}_1^{(ex)})^* \mathbf{z}_k^{(ex)} \tag{15}$$

Now, $\widehat{\mathbf{z}}_1^{(ex)} = \mathbf{z}_1^{(ex)}$ and $\mathbf{z}_2^{(ex)}$ are the two best EDMD vectors, so their span is a good approximate invariant subspace and it makes sense to require that $\widehat{\mathbf{z}}_1^{(ex)}$ and $\widehat{\mathbf{z}}_2^{(ex)}$ span the same subspace, and, since $\mathbf{z}_2^{(ex)}$ is considered a good (i.e. second best) EDMD vector it should be changed as little as possible when defining $\widehat{\mathbf{z}}_2^{(ex)}$. This is precisely what the Gram-Schmidt orthogonalisation does, and after the update (15) we have $\widehat{\mathbf{z}}_2^{(ex)} = \mathbf{z}_2^{(ex)} / \|\mathbf{z}_2^{(ex)}\|_2$. This way of reasoning continues analogously to (15) – $\mathbf{z}_3^{(ex)}, \ldots, \mathbf{z}_k^{(ex)}$ are purged from components in the direction of $\widehat{\mathbf{z}}_2^{(ex)}$ etc.

We have included this orthonormalization of the Exact DMD eigenvestors as an option in the new subroutines, but it is not clear whether in an application the benefits (only slightly improved residual) of using EDMD justify the computational effort.

4 A Physics Informed (piDMD) solution

The symmetric DMD has been already analyzed in[7] [3] in a framework of physics informed DMD (piDMD), where a prior knowledge of the underlying dynamics determines that $\mathbf{A} \in \mathcal{M}$, where a matrix manifold $\mathcal{M}$ is defined by the additional (physics informed) constraints such that e.g. $\mathbf{A}$ must be Hermitian, or skew-Hermitian, unitary, Toeplitz etc. In our case, $\mathcal{M}$ stands for Hermitian matrices. Let us briefly review the piDMD approach proposed in [3]. It is nicely motivated by numerical examples e.g. with learning the energy states of a quantum Hamiltonian [3, §4.3.1] where it is shown that the loss of hermiticity/symmetry may result in a non-physical and thus inaccurate/useless results. To mitigate the problem, [3] proposes selecting a DMD matrix that solves

$$A \in \arg\min_{A \in \mathcal{M}} \|A\mathbf{X} - \mathbf{Y}\|_F. \tag{16}$$

This is a well studied structured (symmetric/Hermitian) Procrustes problem with an explicitly known solution [13]. Although the solution presented in [13] is derived for $n \leq m$, the same procedure applies for $n \gg m$ and we present it here because of the details needed to analyze the piDMD solution and to compare it to the material in §3.2.

[7]The author is indebted to Allan Avila, AIMdyn Inc. Santa Barbara CA, for calling his attention to this paper.

4.1 Hermitian Procrustes problem

Let $\mathbf{X} = U_x \left(\begin{smallmatrix}\Sigma\\ \mathbf{0}\end{smallmatrix}\right) V^* = U\Sigma V^*$ be a full SVD of $\mathbf{X}$ with $n \times n$ unitary U_x. Let r be the rank of $\mathbf{X}$, $\Sigma_r = \mathrm{diag}(\sigma_i)_{i=1}^r$, $\sigma_1 \geq \cdots \geq \sigma_r > 0$. Then

$$\begin{aligned}
\|A\mathbf{X} - \mathbf{Y}\|_F^2 &= \|AU_x \begin{pmatrix}\Sigma\\ \mathbf{0}\end{pmatrix} V^* - \mathbf{Y}\|_F^2 = \| \underbrace{(U_x^* A U_x)}_{M} \begin{pmatrix}\Sigma\\ \mathbf{0}\end{pmatrix} - \underbrace{U_x^* \mathbf{Y} V}_{C} \|_F^2 = \|M \begin{pmatrix}\Sigma\\ \mathbf{0}\end{pmatrix} - C\|_F^2 \\
&= \| \begin{pmatrix} G & L^* \\ L & K \end{pmatrix} \begin{pmatrix}\Sigma\\ \mathbf{0}\end{pmatrix} - \begin{pmatrix} C_{[1]} \\ C_{[2]} \end{pmatrix} \|_F^2, \;\; G = G^* \in \mathbb{C}^{m\times m}, \; C_{[1]} \in \mathbb{C}^{m\times m}, \\
&= \|G\Sigma - C_{[1]}\|_F^2 + \|L\Sigma - C_{[2]}\|_F^2. \qquad (17)
\end{aligned}$$

Clearly, $K = K^* \in \mathbb{C}^{(n-m)\times(n-m)}$ can be taken arbitrary Hermitian, and the optimal choice of L in the second term in (17) is

$$L = \begin{pmatrix} C_{[2]}(:, 1:r)\Sigma_r^{-1} & L_{[2]} \end{pmatrix}, \;\; L_{[2]} \in \mathbb{C}^{(n-m)\times(m-r)} \text{ arbitrary.}$$

Further, taking the hermiticity into account, the first term in (17) reads

$$\|G\Sigma - C_{[1]}\|_F^2 = \sum_{j=1}^{m} |g_{jj}\sigma_j - c_{jj}|^2 + \sum_{i=2}^{m}\sum_{j=1}^{i-1} (|g_{ij}\sigma_j - c_{ij}|^2 + |g_{ij}\sigma_i - \overline{c_{ji}}|^2),$$

which is minimized for

$$g_{ii} = \begin{cases} \dfrac{\Re(c_{jj})}{\sigma_j}, \;\; j = 1, \ldots, r \\ \text{arbitrary real}, \;\; j = r+1, \ldots, m \end{cases}, \;\; g_{ij} = \overline{g_{ji}} = \begin{cases} \dfrac{\sigma_j c_{ij} + \sigma_i \overline{c_{ji}}}{\sigma_i^2 + \sigma_j^2}, \;\; \sigma_i + \sigma_j \neq 0 \\ \text{arbitrary whenever } \sigma_i + \sigma_j = 0 \end{cases}. \qquad (18)$$

The structure of M can be illustrated as follows:

$$M = \left(\begin{array}{cc|cc||ccc} \star & \star & \times & \times & + & + & + \\ \star & \star & \times & \times & + & + & + \\ \hline \times & \times & \otimes & \otimes & \oplus & \oplus & \oplus \\ \times & \times & \otimes & \otimes & \oplus & \oplus & \oplus \\ \hline\hline + & + & \oplus & \oplus & \blacksquare & \blacksquare & \blacksquare \\ + & + & \oplus & \oplus & \blacksquare & \blacksquare & \blacksquare \\ + & + & \oplus & \oplus & \blacksquare & \blacksquare & \blacksquare \end{array}\right) \qquad (19)$$

Note the two levels of the non-uniqueness in M. First, the matrix K (elements denoted by $\blacksquare$) is arbitrary Hermitian and this freedom comes from $m < n$. If $r < m$, the elements $\otimes$, $\oplus$ can be selected freely under the constraint that the matrix remains Hermitian (or real symmetric). Setting all free entries to zero yields the solution of minimal Frobenius norm.

Any matrix $A = A^*$ that solves (16) is then of the form $A = U_x M U_x^*$.

Remark 4.1 Note that a skew-Hermitian Procrustes problem is reduced to the Hermitian one by a multiplication with the imaginary unit $\mathbf{i}$, which rotates a skew-Hermitian $B = -B^*$ into the Hermitian $H = \mathbf{i}B$, so that $\|B\mathbf{X} - \mathbf{Y}\|_F = \|H\mathbf{X} - \mathbf{i}\mathbf{Y}\|_F$.

4.2 An analysis and a twist

Note that the solution $A = U_x M U_x^*$ is $n \times n$ and forming it explicitly is not needed. A low rank approximation of A, proposed in [3] is $\mathbb{A}_\pi = U_k G_k U_k^*$, where $G_k = G(1:k, 1:k)$ and U_k is as in §2.1. (Note that there is no guarantee that $\mathbb{A}_\pi$ is in the solution set of (16).) Then, using the spectral decomposition $G_k = W\Lambda W^*$, the Ritz vectors are computed as the columns of $U_k W$ and the Ritz values are $\lambda_i = \Lambda_{ii}$.

A closer look at these formulas reveals that the elements c_{ij} used in (18) to compute G_k are the entries of the matrix $C_k = U_k^* \mathbf{Y} V_k$, which is actually $C_k = S_k \Sigma_k$, where $S_k = U_k^* \mathbf{H} U_k$ is Hermitian. This implies in (18) that, for $1 \leq i, j \leq k \leq r$,

$$g_{ij} = \overline{g_{ji}} = \frac{\sigma_j c_{ij} + \sigma_i \overline{c_{ji}}}{\sigma_i^2 + \sigma_j^2} = \frac{\sigma_j^2 s_{ij} + \sigma_i^2 \overline{s_{ji}}}{\sigma_i^2 + \sigma_j^2} = \frac{\sigma_j^2 s_{ij} + \sigma_i^2 s_{ij}}{\sigma_i^2 + \sigma_j^2} = s_{ij} = \overline{s_{ji}}. \tag{20}$$

In other words, using a low rank approximation of a solution of the structured Procrustes problem (16) did not produce anything new if the data is indeed generated by a Hermitian matrix.

What about the computed matrix $\widetilde{G}_k$? In a software implementation [3] computes the diagonal and the strictly upper triangle using the computed $\widetilde{C}_k$, and the lower triangle is set to enforce $\widetilde{G}_k = \widetilde{G}_k^*$, so that the rounding errors do not destroy the hermiticity. But, we saw in §3.2.1 that the upper triangle of $\widetilde{S}_k$ is a bad choice, and, since $C_k = S_k \Sigma_k$, we may expect that the upper triangle of $\widetilde{C}_k$ may have large errors in its entries. Now, if the upper triangle of $\widetilde{G}_k$ is computed using the upper triangle of $\widetilde{C}_k$, the question is how accurate is $\widetilde{G}_k$.

To get a clue, we use the data from Example 3.1 and test the accuracy of $\widetilde{C}_k$ and $\widetilde{G}_k$. The results are shown in Figure 5.

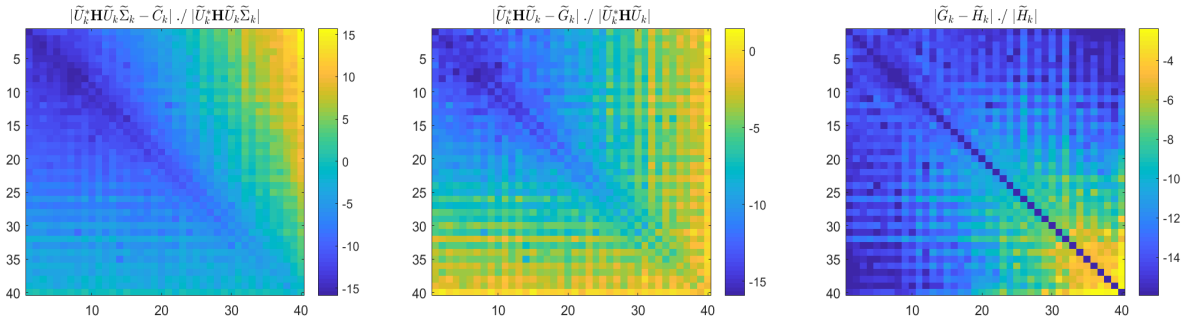

Figure 5: (continuation of Example 3.1) *First panel:* The entry-wise relative errors $\log_{10}(|(C_k)_{ij} - (\widetilde{C}_k)_{ij}|/|(C_k)_{ij}|)$ where $C_k = \widetilde{U}_k^* \mathbf{H} \widetilde{U}_k \widetilde{\Sigma}_k$ is computed explicitly using $\mathbf{H}$. *Second panel:* $\log_{10}(|(S_k)_{ij} - (\widetilde{G}_k)_{ij}|/|(S_k)_{ij}|)$, where $S_k = \widetilde{U}_k^* \mathbf{H} \widetilde{U}_k$. Note that the large errors in the upper triangle of $\widetilde{C}_k$ did not pollute the symmetrizing matrix $\widetilde{G}_k$. The *third panel* shows the entry-wise difference between $\widetilde{H}_k$ and $\widetilde{G}_k$. Recall that in exact computation $H_k = G_k$.

There an interesting subtlety here. Although the upper triangle of $\widetilde{C}_k$ has large relative errors (as compared to $\widetilde{U}_k^* \mathbf{H} \widetilde{U}_k \widetilde{\Sigma}_k$) in the northeastern corner, the matrix $\widetilde{G}_k$ approximates the Rayleigh quotient $S_k = \widetilde{U}_k^* \mathbf{H} \widetilde{U}_k$ to the same level of accuracy as $\widetilde{H}_k$ does, as shown in Figure 5.

Consider now computation of the elements $\widetilde{g}_{ij}$ in the upper triangle ($i \leq j \leq k$) of $\widetilde{G}_k$. We continue using the simplified model of error analysis where the only error is the one from the computed SVD of $\mathbf{X}$. The rounding errors in computing e.g. $\widetilde{C}_k = \widetilde{U}_k^* \mathbf{Y} \widetilde{V}_k$ are neglected. We have, using $S_k = \widetilde{S}_k + \delta \widetilde{S}_k$ and $\widetilde{C}_k = \widetilde{S}_k \widetilde{\Sigma}_k$,

$$\begin{aligned} \widetilde{g}_{ij} &= \frac{\widetilde{\sigma}_j \widetilde{c}_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i \overline{\widetilde{c}_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} = \frac{\widetilde{\sigma}_j^2 \widetilde{s}_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i^2 \overline{\widetilde{s}_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} = \frac{\widetilde{\sigma}_j^2 (s_{ij} - \delta \widetilde{s}_{ij})}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i^2 (\overline{s_{ji}} - \overline{\delta \widetilde{s}_{ji}})}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \\ &= \frac{\widetilde{\sigma}_j^2 s_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i^2 \overline{s_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} - \frac{\widetilde{\sigma}_j^2 \delta \widetilde{s}_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} - \frac{\widetilde{\sigma}_i^2 \overline{\delta \widetilde{s}_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} = s_{ij} - \left(\frac{\widetilde{\sigma}_j^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \delta \widetilde{s}_{ij} + \frac{\widetilde{\sigma}_i^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \overline{\delta \widetilde{s}_{ji}} \right) \end{aligned}$$

i.e.

$$s_{ij} = \widetilde{g}_{ij} + \delta \widetilde{g}_{ij}, \quad \delta \widetilde{g}_{ij} = \left(\frac{\widetilde{\sigma}_j^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \delta \widetilde{s}_{ij} + \frac{\widetilde{\sigma}_i^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \overline{\delta \widetilde{s}_{ji}} \right).$$

Note that the entries of $\widetilde{G}_k$ are computed from the entries of $\widetilde{S}_k$ as convex combinations that put more weight of the more accurate lower triangle. Indeed, for $\widetilde{\sigma}_j \ll \widetilde{\sigma}_i$, δs_{ij} is scaled with $\widetilde{\sigma}_j^2/(\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2) \ll 1$. This is illustrated in Figure 6.

Note the difference from the computation of $\widehat{S}_k$ in (9) where the upper and the lower triangle are averaged, in a convex combination with the coefficient 1/2, i.e. $\widehat{S}_k = S_k - (0.5\delta\widetilde{S}_k + 0.5\delta\widetilde{S}_k^*)$. Further, $\widetilde{H}_k$ computed as in (14), is contaminated by only the lower triangle of the error in $\widetilde{S}_k$.

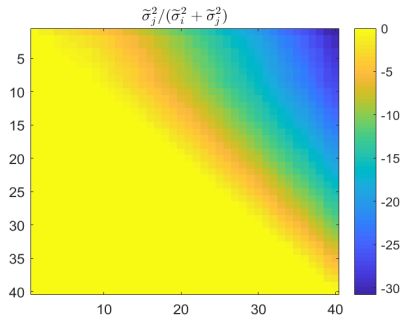

Figure 6: The distribution of the values $\widetilde{\sigma}_j^2/(\widetilde{\sigma}_i^2+\widetilde{\sigma}_j^2)$ illustrate why the large errors in the northeastern corner of $\widetilde{C}_k$ (the first panel in Figure 5) did not pollute the entries of $\widetilde{G}_k$ (the second panel in Figure 5). See how the triangles are differently weighted in the convex combination in relation (20).

Finally, note the entry-wise matching of $\widetilde{H}_k$ and $\widetilde{G}_k$ in the third panel in Figure 5. Just comparing their diagonals seems to indicate how well their leading submatrices approximate S_k. In our software implementation, by setting a suitable work parameter, a user can choose between $\widetilde{H}_k$ and $\widetilde{G}_k$. When computing $\widetilde{G}_k$ we evaluate its lower triangle – same analysis holds as for the upper triangle and this compatibility with the computation of $\widetilde{H}_k$ makes the code simpler. Using $\widetilde{G}_k$ and the framework of the Hermitian Procrustes problem may be advantageous if the data is corrupted by noise so that the hermiticity/symmetry is simply imposed by working with nearest Hermitian/symmetric matrix. However, it should be noted that the truncation at $k < r$ (see the definition of $\mathbb{A}_\pi$) does not inherit the optimality.

5 QR compressed symmetric/Hermitian DMD

In [9], [6], [7] we used the QR factorization to compute a basis of a subspace in the state space $\mathbb{C}^n$ (or $\mathbb{R}^n$) that contains all snapshots. In the case of interest where n is much larger than the number of available snapshots, the snapshots are represented in the lower dimensional subspace which allows for a more efficient computation. This QR compressed DMD can be formulated for $\mathbf{X}, \mathbf{Y}$ that contain data from an arbitrary number of trajectories. Here we briefly review the single trajectory case, using the material from [9], [7] and show that this step preserves the hermiticity/symmetry.

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 auxiliary subspace is of dimension $m+1$. If we compute the QR factorization

$$(\mathbf{z}_1, \ldots, \mathbf{z}_m, \mathbf{z}_{m+1}) = Q \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix} = \widehat{Q} R, \quad \text{where} \quad Q^*Q = \mathbb{I}_n, \quad \widehat{Q} = Q(:, 1:m+1), \tag{21}$$

then

$$\mathbf{X} = Q \begin{pmatrix} R_x \\ \mathbf{0} \end{pmatrix} = \widehat{Q} R_x, \quad \mathbf{Y} = Q \begin{pmatrix} R_y \\ \mathbf{0} \end{pmatrix} = \widehat{Q} R_y,$$

where

$$R = \begin{pmatrix} \times & * & * & * & \div \\ & * & * & * & \div \\ & & * & * & \div \\ & & & * & \div \\ & & & & \div \end{pmatrix}, \quad R_x = R(:, 1:m) = \begin{pmatrix} \times & * & * & * \\ & * & * & * \\ & & * & * \\ & & & * \\ & & & 0 \end{pmatrix}, \quad R_y = R(:, 2:m+1) = \begin{pmatrix} * & * & * & \div \\ * & * & * & \div \\ & * & * & \div \\ & & * & \div \\ & & & \div \end{pmatrix}. \tag{22}$$

$$\|A\mathbf{X} - \mathbf{Y}\|_F^2 \;=\; \|AQ \begin{pmatrix} R_x \\ \mathbf{0} \end{pmatrix} - Q \begin{pmatrix} R_y \\ \mathbf{0} \end{pmatrix} \|_F^2 = \| \underbrace{Q^*AQ}_{M} \mathbf{X}' - \mathbf{Y}' \|_F^2, \;\; \mathbf{X}' = \begin{pmatrix} R_x \\ \mathbf{0} \end{pmatrix}, \; \mathbf{Y}' = \begin{pmatrix} R_y \\ \mathbf{0} \end{pmatrix}.$$

In the new coordinates, the matrix representation of the linear operator changes by similarity, and in the new representation we have $M = Q^*AQ$ and the data snapshots are $\left(\begin{smallmatrix} R_x \\ \mathbf{0} \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} R_y \\ \mathbf{0} \end{smallmatrix}\right) = M \left(\begin{smallmatrix} R_x \\ \mathbf{0} \end{smallmatrix}\right)$. Clearly, if $\mathbf{H} = \mathbf{H}^* \in \arg\min_A \|A\mathbf{X} - \mathbf{Y}\|_F$, then $\mathbf{M} = Q^*\mathbf{H}Q = \mathbf{M}^* \in \arg\min_M \|M\mathbf{X}' - \mathbf{Y}'\|_F$. Hence, we have arrived at an equivalent formulation of the original Hermitian DMD problem. According to the discussions in the previous sections, if we set $\mathbb{M} = \mathbf{Y}'(\mathbf{X}')^\dagger$, then

$$\mathbb{M} = \mathbf{Y}'(\mathbf{X}')^\dagger = Q^*\mathbf{Y}\mathbf{X}^\dagger Q = Q^*\mathbb{A}Q, \;\; \mathbf{Y}'(\mathbf{X}')^\dagger = \begin{pmatrix} R_y \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} R_x^\dagger & \mathbf{0} \end{pmatrix} = \begin{pmatrix} R_y R_x^\dagger & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Further, using the notation of (3), we have $[\mathbb{A}; \mathbf{X}, \mathbf{Y}] = Q[\mathbb{M}; \mathbf{X}', \mathbf{Y}']Q^*$. This is the situation in the n-dimensional state space. On the practical side, everything we need takes place in the $(m+1)$-dimensional range of $\widehat{Q}$ and can be described as follows. Let $R_x = U_x\Sigma V^*$ be the economy-size SVD of R_x; Σ is $r \times r$, and V is $m \times r$, where $r = \mathrm{rank}(\mathbf{X}) = \mathrm{rank}(R_x)$. Note that then $\mathbf{X} = (\widehat{Q}U_x)\Sigma V^*$ is the SVD of $\mathbf{X}$, and that $\mathbf{H}\widehat{Q}R_x = \widehat{Q}R_yVV^*$. Hence

$$\mathbf{H}\widehat{Q}U_x\Sigma V^* = \widehat{Q}R_yVV^* \;\text{ and }\; \mathbf{H}\widehat{Q}U_x\Sigma = \widehat{Q}R_yV. \tag{23}$$

We can truncate (23) at an index k determined as in (4) to obtain

$$\mathbf{H}\widehat{Q}U_x(:, 1:k)\Sigma_k = \widehat{Q}R_yV_k \;\text{ and }\; U_x(:, 1:k)^*\widehat{Q}^*\mathbf{H}\widehat{Q}U_x(:, 1:k) = U_x(:, 1:k)^*R_yV_k\Sigma_k^{-1}.$$

Since $\mathbf{H} = \mathbf{H}^*$, the matrix $S_k = U_x(:, 1:k)^*\widehat{Q}^*\mathbf{H}\widehat{Q}U_x(:, 1:k)$ is also Hermitian and the analysis of §3 applies.[8] S_k is the Rayleigh quotient of $\mathbf{H}$ with respect to the range of $\widehat{Q}U_x(:, 1:k)$. If $S_kw_j = \lambda w_j$, $\|w_j\|_2 = 1$, then

$$(\lambda_j, \mathbf{z}_j), \;\; \text{where } \mathbf{z}_j = \widehat{Q}U_x(:, 1:k)w_j = Q \begin{pmatrix} U_x(:, 1:k)w_j \\ \mathbf{0} \end{pmatrix} \tag{24}$$

is the corresponding Ritz pair of $\mathbf{H}$. Note that $(\lambda_j, U_x(:, 1:k)w_j)$ is a Ritz pair for $\widehat{Q}^*\mathbf{H}\widehat{Q}$ from the range of $U_x(:, 1:k)$. Hence, the QR compressed DMD first compresses the underlying $\mathbf{H}$ onto an $(m+1)$-dimensional subspace, computes the $(m+1)$-dimensional DMD using the projected data and then lifts the Ritz pairs back to the original n-dimensional state space (24). This lifting preserves the orthogonality of the Ritz vectors, as well as the orthonormality of the Exact DMD vectors.

The LAPACK implementation of the QR compressed Hermitian/symmetric DMD contains subroutines `xSYDMDQ` (`x` ∈ {`S`, `D`}) and `xHEDMDQ` (`x` ∈ {`C`, `Z`}).

[8] Here one can formulate the Hermitian Procrustes problem and proceed as in §4.

Remark 5.1 Note that, based on §3.2.3, the same procedure applies to the skew-Hermitian case, because the unitary congruence $\cdot \mapsto Q^* \cdot Q$ preserves both hermiticity and skew-hermiticity. Since this congruence also preserves unitarity (if A is unitary, so is Q^*AQ), the QR compression can be also used as a dimension reduction preprocessing for the unitary DMD described in [3]. Working on the numerical details of the unitary case (the data are generated by a unitary operator) in a way analogous to this note on the Hermitian case is an interesting challenging problem, in particular because the Rayleigh quotient does not inherit unitarity/orthogonality, and because the data must satisfy $\mathbf{X}^*\mathbf{X} = \mathbf{Y}^*\mathbf{Y}$. These issues are not addressed in [3].

6 Concluding remarks and outlook

The orthogonality of the Ritz vectors (Koopman modes) in the Hermitian (and also skew-Hermitian) cases makes the applications of the DMD (such as e.g. spatio-temporal analysis of the snapshots) more efficient and numerically robust. In a general case, the Ritz vectors can be highly ill-conditioned which renders modal analysis numerically difficult.

Following [9], in [8], we have developed a new approach for working with orthonormal vectors, based on partial Schur decomposition of numerical compressions of the Koopman operator to a suitable finite dimensional function subspaces. This has both theoretical and computational advantages in the case of highly non-normal operators. A software implementation of the new proposed KSDMD (Koopman-Schur DMD), that contains the Hermitian and skew-Hermitian DMD presented in this report as special cases, is currently under testing.

7 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] 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.

[2] 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.

[3] Peter J. Baddoo, Benjamin Herrmann, Beverley J. McKeon, J. Nathan Kutz, and Steven L. Brunton. Physics-informed dynamic mode decomposition (piDMD). *arXiv e-prints*, page arXiv:2112.04307, December 2021.

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

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

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

[7] Z. Drmač. A LAPACK implementation of the Dynamic Mode Decomposition I. Technical report, Department of Mathematics, University of Zagreb, Croatia, and AIMdyn Inc. Santa Barbara, CA, October 2022. LAPACK Working Note 298.

[8] Z. Drmač and I. Mezić. A data driven Koopman-Schur decomposition for computational analysis of nonlinear dynamics. Technical report, Univerity of Zagreb and AIMdyn Inc. Santa Barbara, CA, December 2021.

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

[10] 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.

[11] Ky Fan and A. J. Hoffman. Some metric inequalities in the space of matrices. *Proceedings of the American Mathematical Society*, 6(1):111–116, 1955.

[12] 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.

[13] Nicholas J. Higham. The symmetric Procrustes problem. *BIT*, 28:133–143, 1988.

[14] 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.

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

[16] W. Kahan. The nearest orthogonal or unitary matrix. Lecture notes, University of California, Berkeley, August 2011.

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

[18] B. N. Parlett. *The Symmetric Eigenvalue Problem, Classics In Applied Mathematics 20.* SIAM, Philadelphia, PA, 1998.

[19] 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.

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

[21] 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.

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

[23] 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.

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

[25] G. W. Stewart and Ji-G. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[26] 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.

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

[28] 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.

8 Appendix: Software – implementation details

The proposed library for Hermitian DMD consists of eight subroutines as follows:

- For real data:
  - `SSYDMD` (single precision symmetric DMD), `DSYDMD` (double precision symmetric DMD)
  - `SSYDMDQ` (single precision QR compressed symmetric DMD), `DSYDMDQ` (double precision QR compressed symmetric DMD)
- For complex data:
  - `CHEDMD` (single precision Hermitian DMD), `ZHEDMD` (double precision Hermitian DMD)
  - `CHEDMDQ` (single precision QR compressed Hermitian DMD), `ZHEDMDQ` (double precision QR compressed Hermitian DMD)

  In the case of complex data, the subroutines can be used to compute a skew-Hermitian DMD, by changing the input data $(\mathbf{X}, \mathbf{Y})$ to $(\mathbf{X}, \mathbf{i}\mathbf{Y})$.

The structure of the proposed subroutines is analogous to `xGEDMD` and `xGEDMDQ` in [7], with three main differences that are induced by the Hermitian/symmetric structure of the problem:
(i) We added a correction of the computed Rayleigh quotient to restore hermiticity/symmetry, based on §3.2 and §4.
(ii) Hermitian/symmetric eigensolvers (`xSYEV`, `xHEEV` or `xSYEVD`, `xHEEVD`) are used instead of the general solver `xGEEV`.
(iii) If the Exact DMD eigenvectors are requested, we added a reorthogonalization procedure to obtain a numerically orthonormal system. For this correction, as discussed in §3.3, a simple Gram-Schmidt orthogonalization suffices; in the code we used (because LAPACK does not have a Gram-Schmidt orthogonalization) `xGEQRF` and `xORGQR`, `xUNGQR`.
Next, we briefly describe the interface to the real double precision subroutine `DSYDMD`.

8.1 DSYDMD

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

8.1.1 Brief description of the arguments of DSYDMD

JOBS, JOBZ, JOBR, JOBF

`JOBS` determines whether the initial data snapshots should be scaled with a diagonal matrix that normalizes the columns of $\mathbf{X}$ or $\mathbf{Y}$. 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.

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.

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.

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.
'X' The Exact DMD vectors are orthogonalized and returned in the array B. To preserve the residuals of the orthogonalized EDMD vectors they are reordered and the reordering permutation is stored and returned in the array IWORK. See the descriptions of B and IWORK.
'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 $\in \{1,2,3,4\}$). 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 $\mathbf{Y}$.

WHTSYM Specifies the method for restoring the symmetry of the Rayleigh quotient.

1 The lower triangle of the computed Rayleigh quotient is used to symmetrize the matrix,
2 The formulas for the lower triangle of a truncated solution of the symmetric Procrustes problem are used to symmetrize the computed Rayleigh quotient.

WHTEIG Specifies the symmetric eigensolver to compute the eigenvalues and eigenvectors of the symmetric Rayleigh quotient.

1 DSYEV (the QR algorithm)
2 DSYEVD (the divide and conquer algorithm)

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, EIGS, 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 DSYEV or DSYEVD, as specified in WHTEIG. The eigenvectors are returned in the first K columns of Z.

If JOBZ=='F', then the above descriptions hold for the columns of X(:,1:K)$\star$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, RES(j) = $\|\mathbf{H}\star$Z(:,j) − EIGS(j)$\star$Z(:,j)$\|_2$.

B, LDB, W, LDW, S, LDS S is the K-by-K Rayleigh quotient, i.e. the matrix $S_k = U_k^* \mathbf{H} U_k$ described in §3. The array W is used to temporarily hold the right singular values of $\mathbf{X}$, and on return it contains the eigenvectors of S, as computed by the eigencolver selected by WHTEIG. The array B is used only if the data for computing the refined Ritz vectors (JOBF='R') or the Exact DMD eigenvectors (JOBF='E' or JOBF='X') 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. On exit, WORK(1:N) contain the computed singular values of $\mathbf{X}$. If JOBF=='X', on return IWORK(1:N) contains a reordering permutation used to compute orthonormalized Exact DMD eigenvectors, so that EIGS(IWORK(i)) is the eigenvalue that corresponds to the i-th EDMD vector. See the descriptions of JOBF and B.

INFO On exit, INFO contains status information on the DSYDMD 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.

8.2 DSYDMDQ

```
SUBROUTINE DSYDMDQ(JOBS, JOBZ, JOBR, JOBQ, JOBT, JOBF, WHTSVD, WHTSYM, WHTEIG,&
                   M, N, F, LDF, X, LDX, Y, LDY, NRNK, TOL, K, EIGS, Z, LDZ,  &
                   RES, B, LDB, V, LDV, S, LDS, WORK, LWORK, IWORK, LIWORK, INFO)
```

8.2.1 Brief description of the arguments of DSYDMDQ

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

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, WHTSYM, WHTEIG These arguments are defined as in DSYDMD.

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, EIGS, Z, LDZ, RES, W, LDW, S, LDS These parameters are defined as in DSYDMD.

B, LDB are defined as in DSYDMD, but the content of B is in the lower 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 DSYDMDQ run. It is defined as in DSYDMD.

8.3 Numerical tests

The proposed routines passed the tests described in [7], with random Hermitian/real symmetric matrices, so that the graphical presentation of the results is as in [7] and it is omitted for the sake of brevity. The computed eigenvectors are numerically orthonormal and the eigenvalues are real by design, so those structural properties are guaranteed.

As an illustration, we show a result of test with the discretized two-dimensional Laplace operator over a 30×30 discrete grid. In Figure 7, we show the dominant (normlaized) eigenvector computed by DSYDMD with 400 samples from two Krylov sequences. The reference vector is computed using the function eig() from Matlab.

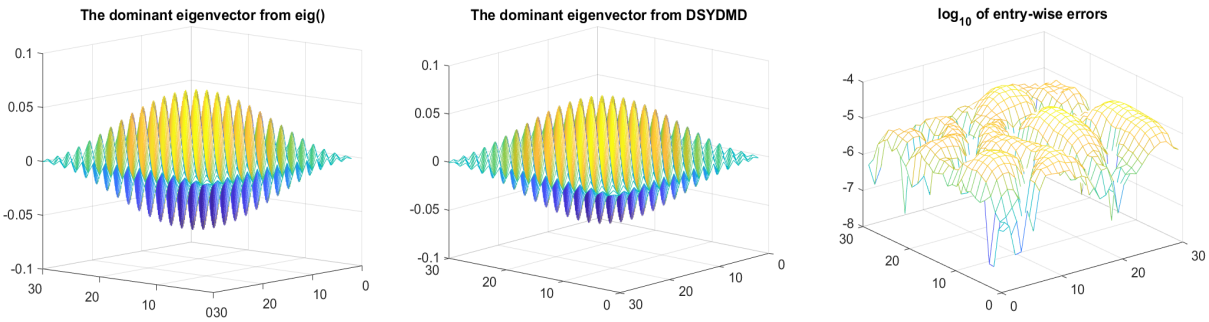

Figure 7: