Data driven modal decompositions: analysis and enhancements

Zlatko Drmač1,∗, Igor Mezić2, Ryan Mohr3
1 Faculty of Science, Department of Mathematics, University of Zagreb, Croatia.
2 Department of Mechanical Engineering and Mathematics, University of California, Santa Barbara, CA, 93106, USA; AIMdyn, Inc., Santa Barbara, CA 93101, USA.
3 AIMdyn, Inc., Santa Barbara, CA 93101, USA; Department of Mechanical Engineering, University of California, Santa Barbara, CA, 93106, USA.

Abstract. The Dynamic Mode Decomposition (DMD) is a tool of trade in computational data driven analysis of fluid flows. More generally, it is a computational device for Koopman spectral analysis of nonlinear dynamical systems, with a plethora of applications in applied sciences and engineering. Its exceptional performance triggered developments of several modifications that make the DMD an attractive method in data driven framework. This work offers further improvements of the DMD to make it more reliable, and to enhance its functionality. In particular, data driven formula for the residuals allows selection of the Ritz pairs, thus providing more precise spectral information of the underlying Koopman operator, and the well-known technique of refining the Ritz vectors is adapted to data driven scenarios. Further, the DMD is formulated in a more general setting of weighted inner product spaces, and the consequences for numerical computation are discussed in detail. Numerical experiments are used to illustrate the advantages of the proposed method, designated as DDMD_RRR (Refined Rayleigh Ritz Data Driven Modal Decomposition).

AMS subject classifications: 15A12, 15A23, 65F35, 65L05, 65M20, 65M22, 93A15, 93A30, 93B18, 93B40, 93B60, 93C05, 93C10, 93C15, 93C20, 93C57

Key words: Dynamic Mode Decomposition, Koopman operator, Krylov subspaces, Proper Orthogonal Decomposition, Rayleigh-Ritz approximation, weighted inner product

1. Introduction

Dynamic Mode Decomposition (DMD) has become a major tool in the data-driven analysis of complex dynamical systems. DMD was first introduced in 2008 by P. Schmid [1] for the study of fluid flows where it was conceptualized as an algorithm to decompose the flow field into component fluid structures, called “dynamic modes” or “DMD modes”, that described the evolution of the flow. The method asserts the existence of a linear operator that maps a collection of snapshots of the fluid flow forward one step in time [1]. For a nonlinear evolution operator, such as the one generated by the Navier-Stokes equations, the proposed linear operator is equivalent to a linear tangent space approximation [2]. The DMD modes and their temporal behavior are given by the spectral analysis of the linear operator, which is constructed from data since it is assumed that direct access to it is not available. Rowley et al. [3] gave the method theoretical underpinnings by connecting it to the spectral analysis of the Koopman operator — a linear operator that can be associated with any nonlinear dynamical system — which evolves observables of that system forward in time.

∗Corresponding author. Email addresses: drmac@math.hr (Z. Drmač), mezic@engineering.ucsb.edu (I. Mezić), mohrr@aimdyn.com (R. Mohr)
The algorithm was cast as a Krylov subspace method in which the operator was represented as a companion matrix in the Krylov basis formed from the data snapshots.

Many variants of the basic algorithm have been introduced since then (see, for example, [4, 5, 6, 7, 8, 9]) all purporting to more accurately, robustly, or efficiently compute the eigenvalues and modes under various assumptions on the data. However, deep numerical analyses giving some certificate of accuracy for these algorithms have been absent. This is especially troubling as the DMD method, in all of its various guises, has enjoyed large scale deployment in fields such as fluid dynamics (see [10] and the references therein) where it is often taken as an “off-the-shelf” algorithm whose results are implicitly trusted. This is contrasted with the subset of practitioners who recognize that the method often produces spurious or inaccurate eigenvalues that are not associated with spectrum of the operator generating the data. This can even be true in the simplest case where the data snapshots are produced by powers of a matrix applied to an initial vector – the standard Schmid-type DMD method can fail to accurately capture the spectrum of the matrix, even if the supplied data is rich in spectral information.

The detection of these spurious or inaccurate eigenvalues has been approached in an ad hoc manner. Eigenvalues are often ranked in decreasing importance by the $L^2$-norm (energy) of their associated mode and are deemed non-essential if the norm is sufficiently small. Recently, D. Giannakis has proposed using a different measure that imposes a penalty for the eigenvalues based on their mode’s “roughness” via their Dirichlet energy [11]. This modification captures the physical reasoning that real systems are more likely to produce smooth modes, which is a conjecture that itself must be justified.

Focusing on the magnitude of the energy, however, can lead to discarding physically relevant dynamics, especially if the high energy of the mode is an artifact of the units the data is reported in. For example, snapshots can be formed from data acquired via several different sensors, with each sensor reporting information in different units. From a scientist's perspective, there is no difference in reporting, say the power consumption of a system, in watts or milliwatts; both numbers represent the same physical quantity. Numerically, however, there can be a large difference. The situation is exacerbated further by data which contains measurements of quantities with fundamentally different physical nature.

Despite these concerns, DMD methods have demonstrated exceptional performance in many applications. However, this often requires deep, domain-specific knowledge to determine the reliability of the algorithm’s output. The question of when it fails and how badly is still open. If the outputs of the algorithm are not known to be spurious or real, the inferences based on these outputs cannot be known to be reliable. Therefore, DMD should be analyzed in depth so that there are guarantees on the accuracy and reliability of the algorithm. Furthermore, this analysis should be divorced from domain-specific knowledge of the current application. This will not only reassure the algorithm’s fitness for further nontrivial applications, moving it toward a true “off-the-shelf” method that a non-expert can apply to their particular problem, but also allow modifications that will improve its numerical reliability and robustness.

1.1. Contributions and overview

In this work, we excogitate ways to address the aforementioned issues with several modifications and enhancements of the DMD. In §2, we set the stage and briefly review Krylov subspaces with the corresponding decomposition, and the Rayleigh-Ritz procedure for extracting spectral information from a given Krylov subspace. We briefly discuss how the Krylov subspaces naturally appear in spectral approximations of the Koopman operator, and we review the DMD algorithm. In §3, we first show how the DMD algorithm can be equipped with residual estimate that can be used
to assess the quality of each particular Ritz pair. Further, we show how to apply the well known Ritz vector refinement technique to the DMD data driven setting, and we discuss the importance of data scaling. All these modifications are integrated in §3.5 where we propose a new version of the DMD, designated as DDMD_RRR (Refined Rauleigh-Ritz Data Driven Modal Decomposition). In §4 we provide numerical examples that show the benefits of our modifications, and we discuss the fine details of software implementation. In §5 we use the Exact DMD [12] to show that our modifications apply to other versions of DMD. In §6, we provide a compressed form of the new DDMD_RRR, designed to improve the computational efficiency in case of extremely large dimensions. A matrix-root-free modification of the Forward-Backward DMD [13] is presented in §7. The column scaling used in the new DDMD implementation in §3.5 is just a particular case of a more general weighting scheme that we address in §8. Using the concept of the generalized SVD introduced by Van Loan [14], we define weighted DDMD with the Hilbert space structures in the spaces of the snapshots and the observables (spatial and temporal) given by two positive definite matrices.

2. DMD as data driven Krylov+Rayleigh-Ritz procedure

In the framework of dynamic mode decomposition and analysis, we are given e.g. the flow field data\(^1\) \(f_1, \ldots, f_{m+1} \in \mathbb{C}^n\), under the assumption that it has been generated by an unknown linear operator \(A\) such that \(f_{i+1} = Af_i, i = 1, 2, \ldots\). We can think of \(A\) as a discretization of the underlying physics that drives the measured \(f_i\)'s. In a pure data driven setting we have no access to \(A\). Instead, the \(f_i\)'s are the results of measurements, e.g. computed from pixel values from a high speed camera recorded video, see e.g. [15, §3.1]. No other information on the action of \(A\) is available.

In another scenario of data acquisition, \(A\) represents PDE/ODE solver (software toolbox) that generates solution in high resolution, with given initial condition \(f_1\). In such a framework, the discrete time evolution \(f_{i+1} = Af_i\) can be stopped at some (time) index \(i\) and then restarted with new initial condition. In both scenarios, \(n\) is expected to be large, say \(n > 10^4\), and the number \(m\) of snapshots is typically much smaller. The goal is to extract useful spectral information on \(A\), based solely on these measurements and/or numerical simulation data.

2.1. Connection with the Koopman operator

The seemingly simple sequence of the \(f_i\)'s, the result of the power method applied to \(A\), can be interpreted as a discretization of power iterations applied to a linearization of complex nonlinear dynamics. In analyzing a nonlinear dynamical system \(T : M \to M\) there is an associated infinite-dimensional linear operator \(U : \mathcal{H} \to \mathcal{H}\) defined by the composition operation \(U\psi := \psi \circ T\), where \(\mathcal{H}\) is a Hilbert space of functions on \(M\) closed under composition with \(T\). The spectral properties of this so-called Koopman operator are useful in the analysis, prediction, and control of the underlying nonlinear dynamical system [16, 17].

There are two essentially different types of approximations of the Koopman operator that DMD techniques provide [18]. The first one is related to the methodology introduced in [3], and is interpreted in [18] as follows. Let \(\mathcal{S} = \{x_1, \ldots, x_m\}\) be an invariant set for \(T\). Consider the space \(\mathcal{C}_{|\mathcal{S}}\) of continuous functions in \(\mathcal{H}\) restricted to \(\mathcal{S}\). This is an \(m\)-dimensional vector space. The restriction of the Koopman operator to \(\mathcal{C}_{|\mathcal{S}}, U|_{\mathcal{S}}\) is then a finite-dimensional linear operator that can be represented in a basis by an \(m \times m\) matrix. An explicit example is given when \(x_j, j = 1, \ldots, m\) represent successive points on a periodic trajectory, and the resulting matrix representation in the

\(^1\)In some applications the data can be complex.
standard basis is the \( m \times m \) cyclic permutation matrix

\[
P = \begin{pmatrix}
0 & 0 & \cdots & 0 & 1 \\
1 & 0 & \cdots & 0 & 0 \\
0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \cdots & \cdots & \cdots & \vdots \\
0 & \cdots & \cdots & \cdots & 1 \\
\end{pmatrix}.
\]  

(2.1)

If \( S \) is not an invariant set, an \( m \times m \) approximation of the reduced Koopman operator can still be provided. Namely, if we know \( m \) independent functions' restrictions (\( f_j \))|\( S \), \( j = 1, \ldots, m \) in \( \mathcal{C}|_S \), and we also know \( f_j(Tx_k) \), \( j, k \in \{1, \ldots, m\} \), we can provide a matrix representation of \( \mathcal{U}|_S \). However, while in the case where \( S \) is an invariant set, the iterate of any function in \( \mathcal{C}|_S \) can be obtained in terms of the iterate of \( m \) independent functions, for the case when \( S \) is not invariant this is not necessarily so. Namely, the fact that \( T \) is not invariant means that functions in \( \mathcal{C}|_S \) do not necessarily experience linear dynamics under \( \mathcal{U}|_S \). However, one can take \( m \) observables \( f_j \), \( j = 1, \ldots, n \), where \( n > m \), and approximate the nonlinear dynamics using linear regression on \( f(x) \equiv (f(x_1), \ldots, f(x_m)) \), where \( f(\cdot) = (f_1(\cdot), \ldots, f_n(\cdot))^T \) – i.e by finding an \( m \times m \) matrix \( C \) that gives the best approximation of the data in the Frobenius norm,

\[
C = \arg\min_{B \in \mathbb{C}^{m \times m}} ||f(Tx) - f(x)B||_F \equiv \arg\min_{B \in \mathbb{C}^{m \times m}} ||(f_j(Tx_k))_{j,k=1,1}^{n,m} - (f_j(x_k))_{j,k=1,1}^{n,m}B||_F.
\]  

(2.2)

Under certain conditions this approximation converges weakly to the Koopman operator on an invariant set that the \( x_j \)'s are densely distributed on, see [18].

DMD algorithms and the spectral analysis of the Koopman operator can also be connected by considering finite sections of the matrix associated with the operator [18]. Let \((\phi_1, \phi_2, \ldots) \subset \mathcal{H}\) be a (not necessarily orthogonal) basis for \( \mathcal{H} \) and \((\hat{\phi}_1, \hat{\phi}_2, \ldots) \subset \mathcal{H}\) the dual basis satisfying \((\phi_i, \hat{\phi}_j) = \delta_{ij}.2\) Let \( \mathcal{H}_n = \text{span}(\phi_1, \ldots, \phi_n) \) and \( P_n : \mathcal{H} \rightarrow \mathcal{H}_n \) be the orthogonal projection onto \( \mathcal{H}_n \). We consider a compression of the operator \( \mathcal{U}_n := P_n\mathcal{U}|_{\mathcal{H}_n} : \mathcal{H}_n \rightarrow \mathcal{H}_n \) and find its matrix representation \( \Lambda \in \mathbb{C}^{n \times n} \) in the basis \((\phi_1, \ldots, \phi_n)\). We first note that \( P_n \equiv \Phi_n \hat{\Phi}_n \) where \( \Phi_n : \mathbb{C}^n \rightarrow \mathcal{H}_n \) and \( \hat{\Phi}_n : \mathcal{H} \rightarrow \mathbb{C}^n \) are given, respectively, by

\[
\Phi_n((c_1, \ldots, c_n)^T) = \sum_{k=1}^n c_k \phi_k, \quad \hat{\Phi}_n \psi = ((\psi, \hat{\phi}_1), \ldots, (\psi, \hat{\phi}_n))^T.
\]  

(2.3)

The matrix \( \Lambda \) has elements defined as \( \Lambda_{ij} = (\mathcal{U}\phi_j, \hat{\phi}_i) \) for \( 1 \leq i, j \leq n \) and it can be checked that \( \mathcal{U}_n = \Phi_n \Lambda \hat{\Phi}_n \). Since \( \hat{\Phi}_n \Phi_n = I_{\mathbb{C}^n} \), we have the identity \( \mathcal{U}_n^i = \Phi_n \Lambda^i \hat{\Phi}_n \) for all \( i \geq 0 \). Now, fixing a function \( \psi \in \mathcal{H}_n \) and evolving it with the compression \( \mathcal{U}_n \) gives \( \mathcal{U}_n^i \psi = \Phi_n \Lambda^i \hat{\Phi}_n \psi \) for \( i \geq 0 \). If we define \( f_1 := \hat{\Phi}_n \psi \) and \( f_{i+1} := \hat{\Phi}_n \mathcal{U}_n^i \psi \) for \( i \geq 0 \), then we have that \( f_{i+1} = \Lambda^i f_i \) from the identity \( \hat{\Phi}_n \mathcal{U}_n^i = \Lambda^i \hat{\Phi}_n \). The data sequence \((f_1, f_2, \ldots)\) represents the evolution of the function \( \psi \in \mathcal{H} \) due to the nonlinear dynamics \( T \) in the coordinates given by the basis \((\phi_1, \ldots, \phi_n)\). This representation is amenable to the DMD algorithms we discuss in this paper.

The computed eigenvalues and eigenvectors (eigenmodes) of \( \Lambda \) are the key ingredients of the Dynamic Mode Decomposition (DMD), introduced by Schmid [19]. Schmid's algorithm is widely used and it has become one of the tools of trade in analysis of fluid flows. One of its features, stressed both in applications and the development of Schmid type DMD methods is the low dimensional approximation of the data using the Singular Value Decomposition (SVD).

---

\(^2\)Following standard math notation (as opposed to physics notation), our inner product is linear in the first variable and conjugate linear in the second.
Remark 2.1. Note that DMD produces approximate eigenpairs of $\Lambda$ with an error (that depends on the details of a particular implementation), and that the overall error with respect to some eigenvalues of $U$ (part of its point spectrum) depends on the discretization, i.e. on the choice of the finite dimensional subspace $X_m$. In this work, we do not consider the discretization error.

2.2. Preliminaries

To set the stage, introduce notation and for the reader’s convenience, we briefly review some basic facts about Krylov subspaces in eigenvalue computations, and on SVD based low rank approximation. For more details and deeper insights we refer to [20], [21], [22].

2.2.1. Krylov decomposition. For $i = 1, 2, \ldots, m$, define the Krylov matrices

$$X_i = (f_1 \ f_2 \ \cdots \ f_{i-1} \ f_i), \quad Y_i = (f_2 \ f_3 \ \cdots \ f_i \ f_{i+1}) \equiv A X_i,$$

and the corresponding Krylov subspaces $X_i = \text{range}(X_i) \subset \mathbb{C}^n$. From the assumption $n \gg m$, $X_i$ and $Y_i$ are tall and skinny matrices. The space $\mathbb{C}^n$ is endowed with the complex Euclidean structure; the inner product is $(x, y) = y^* x$, the corresponding norm is $\|x\|_2 = \sqrt{(x, x)}$, and the induced matrix (operator) norm is $\|A\|_2 = \max_{\|x\|_2 =1} \|Ax\|_2$. The orthogonal projection onto the subspace $X_i$ is denoted by $P_{X_i}$.

Assume that at the index $m$, $X_m$ is of full column rank. This implies that

$$X_1 \subsetneq X_2 \subsetneq \cdots \subsetneq X_i \subsetneq X_{i+1} \subsetneq \cdots \subsetneq X_m \subsetneq \cdots \subsetneq X_\ell = X_{\ell+1}, \quad A X_\ell \subseteq X_\ell,$$

i.e. $\dim(X_i) = i$ for $i = 1, \ldots, m$, and there must be the smallest saturation index $\ell$ at which $X_\ell = X_{\ell+1}$. It is well known that then $X_\ell$ is the smallest $A$-invariant subspace that contains $f_1$.

Obviously, with given $f_1, \ldots, f_{m+1}$, the action of $A$ on the range $X_m$ of $X_m$ is known, as $A(X_m v) = X_m v$ for any $v \in \mathbb{C}^m$. Hence, useful spectral information can be obtained using the computable restriction $P_{X_m} A |_{X_m}$, that is, the Ritz values and vectors extracted using the Rayleigh quotient of $A$ with respect to $X_m$.

To that end, let the vector $c = (c_i)_{i=1}^m$ be computed from the least squares approximation,

$$c = \arg \min_{v \in \mathbb{C}^n} \|f_{m+1} - X_m v\|_2,$$

and let $r_{m+1} = f_{m+1} - X_m c$ be the corresponding residual. Recall that, by virtue of the projection theorem, $X_m c = P_{X_m} f_{m+1}$ and that $r_{m+1}$ is orthogonal to the range of $X_m$, $X_m^* r_{m+1} = 0$. Then, since $f_{i+1} = A f_i$, $i = 1, \ldots, m$, and $f_{m+1} = X_m c + r_{m+1}$, we have the Krylov decomposition

$$A X_m = X_m C_m + E_{m+1}, \quad C_m = \begin{pmatrix} 0 & 0 & \cdots & 0 & c_1 \\ 1 & 0 & \cdots & 0 & c_2 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & c_m \end{pmatrix}, \quad E_{m+1} = r_{m+1} e_m^T, \quad e_m = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

Clearly, $C_m = \arg \min_{B \in \mathbb{C}^{m \times m}} \|A X_m - X_m B\|_F$; cf. §2.1. The following theorem summarizes well known facts on spectral approximation from $X_m$:

THEOREM 2.2. Assume (2.4), (2.5), (2.6), (2.7), and let $X_m$ be of full column rank. Then

1. The companion matrix $C_m = (X_m^* X_m)^{-1} (X_m^* A X_m) \equiv X_m^* A X_m$ is the Rayleigh quotient, i.e. the matrix representation of $P_{X_m} A |_{X_m}$ in the Krylov basis $X_m$ of $X_m$.

Here $X_m^*$ denotes the Moore-Penrose generalized inverse of $X_m$.  

aimdyn:201708.004v1

AIMdyn INC.
2. If \( r_{m+1} = 0 \) (and thus \( E_{m+1} = 0 \) and \( m = \ell \) in (2.5)) then \( A X_m = X_m C_m \) and each eigenpair \( C_m w = \lambda w \) of \( C_m \) yields an eigenpair of \( A \), \( \lambda (X_m w) = \lambda (X_m w) \).

3. If \( r_{m+1} \neq 0 \), then \( (\lambda, z \equiv X_m w) \) is an approximate eigenpair; \( \lambda (X_m w) = \lambda (X_m w) + r_{m+1} (e_m^T w) \), i.e. \( \lambda z + r_{m+1} (e_m^T w) \). The Ritz pair \( (\lambda, z) \) is acceptable if the residual

\[
\frac{\| A z - \lambda z \|_2}{\| z \|_2} = \frac{\| r_{m+1} \|_2}{\| z \|_2} |e_m^T w|
\]

is sufficiently small. It holds that \( z^* r_{m+1} = 0 \), and \( \lambda \) is the optimal choice that minimizes the residual (2.8), i.e.

\[
\lambda = \frac{z^* A z}{z^* z} = \arg \min_{\zeta \in \mathbb{C}} \| A z - \zeta z \|_2
\]

(\( \lambda z \) is the orthogonal projection of \( A z \) onto the span of \( z \)).

4. The residual can be pushed in the backward error, thus making the current Krylov subspace an exactly invariant subspace of a perturbed initial matrix \( A \):

\[
(\lambda + \Delta A) X_m = X_m C_m, \quad \text{where} \quad \Delta A = -r_{m+1} e_m^T (X_m^T X_m)^{-1} X_m^T.
\]

Hence, if \( C_m w = \lambda w \), then \( (\lambda, X_m w) \) is an exact eigenpair of \( A + \Delta A \).

5. If \( A \) is diagonalizable\(^3\) with the eigenvalues \( \alpha_1, \ldots, \alpha_n \) and the eigenvector matrix \( S \), then for each eigenvalue \( \lambda \) of \( C_m \), \( \min_{\alpha_i} | \lambda - \alpha_i | \leq \kappa_2(S) \| \Delta A \|_2 \) (Bauer–Fike theorem [24]) and \( \min_{\alpha_i} | \lambda - \alpha_i | / | \lambda | \leq \kappa_2(S) \| A^{-1} \Delta A \|_2 \) (Eisenstat–Ipsen [25]). Here \( \kappa_2(S) = \| S \|_2 \| S^{-1} \|_2 \), and \( \kappa_2(S) = 1 \) if \( A \) is normal.

**Remark 2.3.** The matrix \( X_m \) can be nearly rank deficient. To illustrate, assume that \( A \) is diagonalizable with eigenpairs \( \lambda a_i = \alpha_i a_i \), and that its eigenvalues \( \alpha_i \) are enumerated so that \( 0 \neq | \alpha_1 | \geq | \alpha_2 | \geq \cdots \geq | \alpha_n | \). Let \( f_1 \) be expressed in the eigenvector basis as \( f_1 = \phi_1 a_1 + \cdots + \phi_n a_n \). Then \( f_{i+1} = \lambda^i f_1 = \alpha_1^i \left( \phi_1 a_1 + (\alpha_2/\alpha_1)^i \phi_2 a_2 + (\alpha_3/\alpha_1)^i \phi_3 a_3 + \cdots + (\alpha_n/\alpha_1)^i \phi_n a_n \right) \). Hence, if e.g. \( | \alpha_2 | > | \alpha_3 | \), then for \( j \geq 3 \), \( \lim_{i \rightarrow \infty} (\alpha_j/\alpha_1)^i = 0 \), and thus, with big enough \( i \) the \( f_i \)'s will stay close to the span of \( a_1 \) and \( a_2 \) provided that \( \phi_1 \neq 0, \phi_2 \neq 0 \). This means that relatively small changes of \( X_m \) can make it rank deficient; its range may change considerably under tiny perturbations. In the context of spectral approximations, this is desirable and we hope that the \( f_i \)'s will become numerically linearly dependent as soon as possible; on the other hand we must stay vigilant in computing with \( X_m \) and \( Y_m \) as numerical detection of rank deficiency in the presence of noise is a delicate issue. Further, from Theorem 2.2 one can clearly see the advantage of replacing \( X_m \) with an orthonormal matrix, i.e. executing the Rayleigh–Ritz procedure in orthonormal basis.

**Remark 2.4.** Clearly, if the subspace \( X_m \) determined as the span of the given dataset does not contain information on a desired part of the spectrum, then we cannot expect any method to provide detailed insight into the spectral properties of \( A \). On the other hand, if it does, then we must deploy many different techniques to extract relevant spectral information. Any so devised method, in order to be used with confidence, must be accompanied with an error estimate.

---

\(^3\)In the case of nontrivial Jordan structure of (non-diagonalizable) \( A \), one can use the theory from [23].
2.2.2. Condition number, SVD and low rank approximations. The ill-conditioning of $X_m$ will pose difficulties. Recall, the spectral condition number of $X_m$ is defined as

$$\kappa_2(X_m) = \|X_m\|_2 \|X_m^+\|_2 = \frac{\sigma_1}{\sigma_m}, \quad (2.10)$$

where $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$ are the singular values of $X_m$. High condition number implies closeness to rank deficiency, which is nicely expressed in the following classical theorem.

**Theorem 2.5** (Eckart-Young [26], Mirsky [27]). Let the SVD of $M \in \mathbb{C}^{n \times m}$ be

$$M = U \Sigma V^*, \quad \Sigma = \text{diag}(\sigma_i)_{i=1}^{\min(m,n)}, \quad \sigma_1 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0.$$  

For $k \in \{1, \ldots, \min(m,n)\}$, define $U_k = U(:,1:k)$, $\Sigma_k = \Sigma(1:k,1:k)$, $V_k = V(:,1:k)$, and $M_k = U_k \Sigma_k V_k^*$. The optimal rank $k$ approximations in $\| \cdot \|_2$ and the Frobenius norm $\| \cdot \|_F$ are

$$\min_{\text{rank}(N) \leq k} \| M - N \|_2 = \| M - M_k \|_2 = \sigma_{k+1} \quad (2.11)$$

$$\min_{\text{rank}(N) \leq k} \| M - N \|_F = \| M - M_k \|_F = \sqrt{\sum_{i=k+1}^{\min(m,n)} \sigma_i^2}. \quad (2.12)$$

Hence, if $\sigma_m \ll \sigma_1$, the condition number (2.10) is large, $X_m$ can be made singular with a perturbation $\delta X_m$ such that $\|\delta X_m\|_2 / \|X_m\|_2 = \sigma_m / \sigma_1 = 1/\kappa_2(X_m) \ll 1$.

This very clearly stresses the importance of the numerical issues, from the purely theoretical questions in perturbation theory to the practical software implementations and computations in the machine finite precision arithmetic.

2.3. Schmid’s DMD method

The coefficient matrix $X_m$ in the least squares problem (2.6) may be highly ill-conditioned, and even when the QR factorization $X_m = Q_m R_m$ is available, it is in general ill-advised to compute $c$ as $c = R_m^{-1} Q_m^* f_m$, or $C_m$ using the formula from item 1. in Theorem 2.2 as $C_m = X_m^\dagger Y_m = R_m^{-1} Q_m^* Y_m$ as it has been done e.g. in [15, Algorithm 1].

To avoid the ill-conditioning, Schmid [19] used the thin truncated SVD $X_m = U \Sigma V^* \approx U_k \Sigma_k V_k^*$, where $U_k = U(:,1:k)$ is $n \times k$ orthonormal ($U_k^* U_k = I_k$), $V_k = V(:,1:k)$ is $m \times k$, also orthonormal ($V_k^* V_k = I_k$), and $\Sigma_k = \text{diag}(\sigma_i)_{i=1}^k$ contains the largest $k$ singular values of $X_m$. In brief, $U_k$ is the POD basis for the snapshots $f_1, \ldots, f_m$. Since

$$Y_m = A X_m \approx A U_k \Sigma_k V_k^*,$$

the Rayleigh quotient $S_k = U_k^* A U_k$ with respect to the range of $U_k$ can be computed as

$$S_k = U_k^* Y_m V_k \Sigma_k^{-1}, \quad (2.13)$$

which is suitable for data driven setting because it does not use $A$ explicitly. Clearly, (2.13, 2.14) only require that $Y_m = A X_m$; it is not necessary that $Y_m$ is shifted $X_m$ as in (2.4). Each eigenpair $(\lambda, w)$ of $S_k$ generates the corresponding Ritz pair $(\lambda, U_k w)$ for $A$. This is the essence of the Schmid’s method [19], summarized in Algorithm 2.1 below.

---

\(^{4}\)This is possible even if the underlying $A$ is unitary.
3. New approach to computing the DMD

Our goal is to devise a robust software tool for DMD, that will be capable of producing reliable results even in cases where the data and the output vary over several orders of magnitude. To that end, we first review some details from Algorithm 2.1, and then we propose some improvements. In particular, we enhance the algorithm with a computable residual error bound, as well as with refinement of the Ritz pairs. These techniques are well known in the projection based large scale eigenvalue computation, and we just adapt them to the data driven framework. Finally we propose certain scalings of the data.

3.1. Preliminaries

For the sake of completeness and for the reader’s convenience, we recall some well-known facts and provide few technical details on the structure of the DMD algorithm.

3.1.1. Choosing the dimension of the POD basis. The range of the POD basis \( U_k \), among all \( k \)-dimensional spaces, best captures the snapshots in the least squares sense. Namely, if \( W \) is any \( n \times k \) matrix with orthonormal columns (\( W^*W = I_k \)) then, based on Theorem 2.5,

\[
\sum_{i=1}^{m} \| f_i - WW^*f_i \|_2^2 = \| X_m - W(W^*X_m) \|_F^2 \geq \text{rank}(WW^*X_m) \leq k
\]

\[
\geq \| X_m - U_k\Sigma_kV_k^* \|_F^2 = \| X_m - U_kU_k^*X_m \|_F^2 = \sum_{i=1}^{m} \| f_i - U_kU_k^*f_i \|_2^2 = \sum_{i=k+1}^{m} \sigma_i^2.
\]

This is, of course, the PCA [28] of the data. It is interesting to note that this optimal subspace may not contain any of the \( f_i \)’s.

The value of \( k \), representing the numerical rank of \( X_m \) is determined by inspecting the singular values \( \sigma_1 \geq \cdots \geq \sigma_m \geq 0 \) of \( X_m \) and determining \( k \) as the largest index such that \( \sigma_k > \sigma_1 \epsilon \), i.e.

\[
k = \max \{ i : \sigma_i > \epsilon \sigma_1 \}, \quad (3.1)
\]

where \( \epsilon \in (0, 1) \) is user supplied tolerance. This is justified by (2.11) in Theorem 2.5. Alternatively, we can use (2.12) and define \( k = \max \{ i : \sum_{j=1}^{m} \sigma_j^2 > \epsilon^2 \sum_{j=1}^{m} \sigma_j^2 \} \). See [29] for an in depth analysis.

Choosing an appropriate threshold \( \epsilon \) is nontrivial in the case of noisy data, and it requires additional case-by-case basis information; see e.g. an analysis in the case of particle image velocimetry...
data [30]. For more details see e.g. [31, §8.2], [13]. In this paper we do not consider those issues and focus to the computational aspects of the DMD (see §8), assuming that the threshold \( \epsilon \) (or some other strategy for choosing \( k \)) is given.

### 3.1.2. Structure of the Rayleigh quotient

The following proposition explains the relation between \( C_m \) and \( S_k \), and reveals the details behind the formula for \( S_k \).

**Proposition 3.1.** Let \( S_k \) be computed as in Algorithm 2.1, with \( X_m \) and \( Y_m \) as in (2.4). If \( k = m \), then \( S_m \) and \( C_m \) are similar matrices, where the similarity is realised by the matrix \( V \Sigma^{-1} \). If \( k < m \), then \( S_k \) is the Rayleigh quotient of \( C_m \), with the matrix \( V_k \Sigma_k^{-1} \).

**Proof.** If \( k = m \), then (2.7) yields

\[
\mathcal{A}U_m \Sigma_m V_m^* = U_m \Sigma_m V_m^* C_m + r_{m+1} e_m^T \implies S_m \equiv U_m^* \mathcal{A}U_m = \Sigma_m V_m^* C_m V_m \Sigma_m^{-1} \equiv \Sigma V_m^* C_m V \Sigma^{-1}.
\]

In other words, the Rayleigh quotient is computed using another basis for \( X_m \), necessarily yielding a similar matrix. A similar observation in [32] is justified only for the full rank case \( k = m \).

On the other hand, if \( k < m \) then \( X_m = U_k \Sigma_k V_k^* + \delta X_m \) where \( U_k \Sigma_k V_k^* \) is the best rank \( k \) approximation (in the sense of Theorem 2.5) of \( X_m \) and \( \delta X_m = \sum_{i=k+1}^m \sigma_i U(:, i) V(:, i)^* \). Hence, \( \| \delta X_m \|_2 = \sigma_{k+1} \) and \( U_k^* \delta X_m = 0 \). (Note that this implies \( \mathcal{A} U_k = Y_m V_k \Sigma_k^{-1} \) as in (2.13). In fact, the formula (2.13) is in [19], but derived only for \( k = m \). Here we see that its validity for \( k < m \) is based on these orthogonality relations \(^5\) between the truncated part \( \delta X_m \) and the leading left and right singular vectors of \( X_m \). Also note that \( U_m^* r_{m+1} = 0 \) and \( (\delta X_m)^* r_{m+1} = 0 \).

In terms of this low rank approximation of \( X_m \), relation (2.7) reads

\[
\mathcal{A}(U_k \Sigma_k V_k^* + \delta X_m) = (U_k \Sigma_k V_k^* + \delta X_m) C_m + r_{m+1} e_m^T,
\]

i.e. \( \mathcal{A}(U_k \Sigma_k V_k^* + \delta X_m) = (U_k \Sigma_k V_k^*) C_m + r_{m+1} e_m^T + \delta X_m C_m - \mathcal{A} \delta X_m \), and thus (since \( \delta X_m V_k = 0 \))

\[
\mathcal{A} U_k = U_k (\Sigma_k V_k^* C_m V_k \Sigma_k^{-1}) + r_{m+1} e_m^T V_k \Sigma_k^{-1} + \delta X_m C_m V_k \Sigma_k^{-1}.
\]

In this case, \( S_k = U_k^* \mathcal{A} U_k = \Sigma_k V_k^* C_m V_k \Sigma_k^{-1} \) is a Rayleigh quotient of \( C_m \).

**Remark 3.2.** Note that in relation (3.2)

\[
\mathcal{A} U_k = U_k S_k + r_{m+1} g_k + G_k, \quad g_k = e_m^T V_k \Sigma_k^{-1}, \quad G_k = \delta X_m C_m V_k \Sigma_k^{-1},
\]

where \( \| G_k \|_2 = \| \delta X_m C_m V_k \Sigma_k^{-1} \|_2 \leq \| C_m \|_2 \sigma_{k+1}/\sigma_k \). This means that neglecting \( G_k \) and using the approximate Krylov decomposition \(^6\) \( \mathcal{A} U_k \approx U_k S_k + r_{m+1} g_k \) is acceptable only if the singular values are distributed so that \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \gg \sigma_{k+1} \geq \cdots \geq \sigma_m \), i.e. only in the case of sharp drop after the index \( k \). This much stronger truncation criterion is not taken into account in (3.1).

### 3.2. Residual estimates – data driven residual computation

Not all computed Ritz pairs will provide good approximations of eigenpairs of the underlying \( \mathcal{A} \), and it is desirable that each pair is accompanied with an error estimate that will determine whether it can be accepted and used in the next steps of a concrete application. The residual is computationally feasible and usually reliable measure of fitness of a Ritz pair. With a simple modification, Algorithm 2.1 can be enhanced with residual computation, without using \( \mathcal{A} \) explicitly.

---

\(^5\)In finite precision computation this orthogonality is only numerical, i.e. up to rounding errors that depend on a particular algorithm.

\(^6\)See [33].
PROPOSITION 3.3. For the Ritz pairs $(\lambda_i, Z_k(\cdot; i)) \equiv U_k W_k(\cdot; i)$, $i = 1, \ldots, k$, computed in Algorithm 2.1, the residual norms can be computed as follows:

$$r_k(i) = \|A(U_k W_k(\cdot; i)) - \lambda_i(U_k W_k(\cdot; i))\|_2 = \|(Y_m V_k \Sigma_k^{-1}) W_k(\cdot; i) - \lambda_i Z_k(\cdot; i)\|_2. \quad (3.4)$$

Further, if $A = S \text{diag}(\alpha_i)_{i=1}^n S^{-1}$, then $\min_{\alpha_j} |\lambda_i - \alpha_j| \leq \kappa_2(S) r_k(i)$ (see Theorem 2.2).

Since the formula (3.4) requires the matrix $A U_k \equiv B_k \equiv Y_m V_k \Sigma_k^{-1}$, the order of computation must be changed to compute (and store for later use) $B_k$ at the cost of $2nmk + mk$ flops; then, computing $S_k = U_k^* B_k$ takes additional $2nk^2$ flops. On the other hand, the flop count of the computation of $S_k$ in line 4 in Algorithm 2.1, as indicated by the parentheses, can be estimated at $2nmk + 2mk^2 + k^2$ operations. Note that computing $Z_k$ in line 6 takes $2nk^2$ flops. Hence, this additional computation of the residuals only mildly increases the overall complexity, but we consider this information an important part of the output and thus the overhead it incurs as justifiable.

In a data driven setting, the right hand side in (3.4) is the best one can do. Unfortunately, in finite precision computation, the formula uses computed quantities and it may fail. We discuss this problem and how to fix it in §3.4, §3.5, §4.1.2, §4.1.3.

3.3. Data driven refinement of Ritz vectors

It is known that the Ritz vectors are not optimal eigenvectors approximations from a given subspace $U_k = \text{range}(U_k)$. Hence, for a computed Ritz value $\lambda$, instead of the associated Ritz vector, we can choose a vector $z$ that minimizes the residual (2.8). From the variational characterization of the singular values [34, Theorem 3.1.2], it follows that

$$\min_{z \in U_k \setminus \{0\}} \frac{\|A z - \lambda z\|_2}{\|z\|_2} = \min_{w \neq 0} \frac{\|A U_k w - \lambda U_k w\|_2}{\|U_k w\|_2} = \min_{\|w\|_2 = 1} \|A U_k - \lambda U_k\| w = \sigma_{\min}(A U_k - \lambda U_k), \quad (3.5)$$

where $\sigma_{\min}(\cdot)$ denotes the smallest singular value of a matrix, and the minimum is attained at the right singular vector $w_\lambda$ corresponding to $\sigma_\lambda \equiv \sigma_{\min}(A U_k - \lambda U_k)$. As a result, the refined Ritz vector corresponding to $\lambda$ is $U_k w_\lambda$ and the optimal residual is $\sigma_\lambda$. For a thorough analysis of the refined Ritz vectors and convergence issues we refer the reader to [35], [36], [37], [38], [39].

Here, we need to adapt the refinement procedure to the data driven framework. Using (2.13), the minimization of the residual (3.5) can be replaced with computing the smallest singular value with the corresponding right singular vector of $B_k - \lambda U_k$, where $B_k = Y_m V_k \Sigma_k^{-1}$.

Since (3.5) requires singular value and vector computation of an $n \times k$ matrix for $\lambda = \lambda_i$, $i = 1, \ldots, k$, refined Ritz vectors come with an additional computational cost. It can be alleviated using the following preprocessing that replaces the dimension $n$ with much smaller value $2k$:7

Compute the QR factorization

$$(U_k \quad B_k) = QR, \quad R = \begin{pmatrix} k & k \\ k & k' \end{pmatrix} \begin{pmatrix} R_{[11]} & R_{[12]} \\ 0 & R_{[22]} \end{pmatrix}, \quad k' = \min(n - k, k), \quad (3.6)$$

and write the pencil $B_k - \lambda U_k$ as

$$B_k - \lambda U_k = Q \left( \begin{pmatrix} R_{[12]} \\ R_{[22]} \end{pmatrix} - \lambda \begin{pmatrix} R_{[11]} \\ 0 \end{pmatrix} \right) \equiv Q R_\lambda, \quad R_\lambda = \begin{pmatrix} R_{[12]} - \lambda R_{[11]} \\ R_{[22]} \end{pmatrix}. \quad (3.7)$$

Obviously, the problem (3.5) is reduced to computing the smallest singular value and the corresponding right singular vector of the $2k \times k$ matrix $R_\lambda$.

7Recall that $k \leq m \ll n$. 
**Proposition 3.4.** For the Ritz value \( \lambda = \lambda_i \), let \( w_{\lambda_i} \) denote the right singular vector of the smallest singular value \( \sigma_{\lambda_i} \) of the matrix \( R_{\lambda_i} \) defined using (3.7) and (3.6). Then the vector \( z = z_{\lambda_i} \equiv U_k w_{\lambda_i} \) minimizes the residual (3.5), whose minimal value equals \( \sigma_{\lambda_i} = \| R_{\lambda_i} w_{\lambda_i} \|_2 \).

Since \( k \) is usually small, the cost of preforming this computation with \( R_{\lambda_i}, \ i = 1, \ldots, k, \) is almost negligible in comparison to the overall cost (see §3.2). In an application, we will typically refine only some of the Ritz pairs, selected by the computed residual and/or applying particular criteria on the Ritz values (e.g. largest real parts, or in absolute value greater than one etc.) and the Ritz vectors (e.g. roughness). The matrix \( Q \) in (3.6) is not needed and the computation of \( R \) takes \( 8nk^2 - 16k^3/3 \) flops if we take no advantage of the fact that the columns of \( U_k \) are already orthonormal.\(^8\)

The overhead incurred by the QR factorization (3.6) can be reduced using an interesting and practically useful relation, revealed in the following proposition.

**Proposition 3.5.** In the QR factorization (3.6), define \( \Phi = \text{diag}( (R_{[1]}; i)|i=1 \). Then

\[
\Phi^* R_{[12]} = U_k^* B_k \equiv S_k. \tag{3.8}
\]

**Proof.** If we partition the unitary matrix \( Q \) in (3.6) as \( Q = (Q_1, Q_2) \), where \( Q_1 \) has \( k \) columns, then \( U_k = Q_1 R_{[1]} - \) this is a QR factorization of \( U_k \). Since \( U_k^* U_k = I_k \), and since the QR factorization is essentially unique, \( \Phi \equiv R_{[1]} \) is a diagonal unitary matrix. The unique QR factorization of \( U_k \), \( U_k = U_k I_k = (Q_1 \Phi)(\Phi^* R_{[11]})) = (Q_1 \Phi)I_k \), yields \( U_k = Q_1 \Phi \). Hence \( U_k^* B_k = \Phi^* Q_1^* B_k = \Phi^* R_{[12]} \).

**Remark 3.6.** The benefit of Proposition 3.5 is that it saves \( 2nk^2 \) flops needed to compute \( S_k \); instead, at most \( k^2 \) complex sign changes in \( R_{[12]} \) are needed. Namely, in the numerical software (e.g. Matlab, LAPACK) the QR factorization is implemented so that the diagonal entries of \( R \) are real even for complex input matrices. Hence, in practical computation the matrix \( \Phi \) is \( \text{diag}(\pm 1)_{i=1}^k \).

### 3.4. Preprocessing raw data: scaling

In Algorithm 2.1, the information carried by the \( f_i \)’s that are of very small norm (i.e. much smaller than \( \max_i \| f_i \|_2 \)) is suppressed by truncating small singular values. Unless the underlying \( A \) is unitary, we may expect that the column norms of \( X_m \) vary over several orders of magnitude, implying large spectral condition number \( \kappa_2(X_m) \equiv \sigma_1/\sigma_m \). If \( \| f_i \|_2, \ i = 1, 2, \ldots, m \), e.g. decay very rapidly, then Algorithm 2.1 will tend to use small \( k \). This may be undesirable in case when small norm of \( f_i \) is in the nature of the information it carries, and perhaps just a consequence of a sudden change/switching in the underlying dynamics, and not because it is just a noise. Also, the data \( X_m, Y_m \) can be aggregated from several experiments under different conditions.

As another example, consider computing empirical Gramians of an LTI dynamical systems. Tu and Rowley [40] use a combination of spectral projection onto the subspace of slow eigenvalues and an analytical treatment of impulse response tails; DMD is used to identify slow eigenvalues and the snapshots naturally appear as scaled by the weights from the quadrature formula, see [40, §3.2].

In other words, the subspace \( X_m \) may contain valuable spectral information that will not be captured by the low rank approximation \( U_k \) as described in §2.2.2 and §3.1.1. In that case no method can extract any useful approximation from \( U_k \) — it is simply gone with the truncation. We stress here an important fact: numerical rank determination is not a simple mechanical procedure of

\(^8\)Here, for the sake of brevity, we skip technical details on efficient software implementation. They will be available in the accompanying blueprints of the upcoming software.
truncating the SVD, and, depending on concrete situation, additional information and appropriate procedure may yield better results.

On the other hand, in the Rayleigh-Ritz procedure, it is the subspace contents that matters and we can additionally process the data to facilitate better numerical robustness as long we do not change the underlying subspaces. One simple operation is scaling. Note that $Y_m = AX_m$ is equivalent to $(Y_mD) = (AX_mD)$ for any nonsingular (diagonal) $m \times m$ matrix $D$. Although all these representations of the input-output relations are mathematically equivalent, representing the same physics, they may have different numerical consequences.

Let $X_m$ be of full column rank, and let $D_x = \text{diag}(\|X_m(:,i)\|_2)_{i=1}^m$, and let $X_m = X_m^{(1)} D_x$. Set $Y_m = Y_m^{(1)} D_x$. Similarly define $D_y = \text{diag}(\|Y_m(:,i)\|_2)_{i=1}^m$, and set $Y_m = Y_m^{(2)} D_y, X_m = X_m^{(2)} D_y$. Note that $X_m^{(1)}$ and $Y_m^{(2)}$ have all columns of unit norm. An important consequence of this normalization is that the condition number is nearly optimized over all scalings (see [41])

$$\kappa_2(X_m^{(1)}) \leq \sqrt{m} \min_{D=\text{diag}} \kappa_2(X_mD) \leq \sqrt{m}\kappa_2(X_m). \quad (3.9)$$

For the sake of brevity, here we consider only the scaling by $D_x$, i.e. we proceed with $X_m^{(1)}$ and $Y_m^{(1)} = AX_m^{(1)}$ as the new input-output pair.

A caveat is in order here. Namely, scaling experimental data that might have been contaminated by noise is a tricky business. If for some $f_i$ with tiny $\|f_i\|_2$ its SNR is low, then equilibration has the effect of noise laundering, i.e. such a noisy data may "legally" participate in numerical rank determination, and it could happen that the POD basis will capture a lot of noise. In such cases, the scaling should be assisted by statistical reasoning, based on additional input on the statistical properties of the noise; see e.g. [13]. In this paper we do not delve into these issues, but we stress their importance that will be addressed in our future work. A numerical linear algebra framework for the corresponding computational methods is set in §8.

### 3.5. New method: DDMD_RRR

As a summary of the preceding considerations, we propose the following Algorithm 3.1 – Refined Rayleigh Ritz Data Driven Modal Decomposition:

---

9DDMD_RRR, to be pronounced as "R-cubed DDMD".
Algorithm 3.1 \( |Z_k, \Lambda_k, r_k, \rho_k| = \text{DDMD}_\text{RRR}(X_m, Y_m; \epsilon) \) \{Refined Rayleigh-Ritz DDMD\}

Input:
- \( X_m = (x_1, \ldots, x_m), Y_m = (y_1, \ldots, y_m) \in \mathbb{C}^{n \times m} \) that define a sequence of snapshots pairs \((x_i, y_i) \equiv \mathcal{A}(x_i)\). (Tacit assumption is that \( n \) is large and that \( m \ll n \).)

- Tolerance level \( \epsilon \) for numerical rank determination.

1: \( D_x = \text{diag}(\|X_m(:,i)\|_2)_i^{m} : X_m^{(1)} = X_m D_x^{1/2}, Y_m^{(1)} = Y_m D_x^{1/2} \)

2: [\( U, \Sigma, V \)] = \text{svd}(X_m^{(1)}) \{The thin SVD: \( X_m^{(1)} = U \Sigma V^*, U \in \mathbb{C}^{n \times m}, \Sigma = \text{diag}(\sigma_i)_i^{m} \}

3: Determine numerical rank \( k \), with the threshold \( \epsilon : k = \max\{i : \sigma_i \geq \sigma_1 \epsilon\} \).

4: Set \( U_k = U(:,1:k), V_k = V(:,1:k), \Sigma_k = \Sigma(1:k,1:k) \)

5: \( B_k = Y_m^{(1)}(V_k \Sigma_k^{-1}) \{\text{Schmid’s data driven formula for } \mathcal{A}U_k \}

6: [\( Q, R \)] = \text{qr}(([U_k, B_k]) \{The thin QR factorization: \( (U_k, B_k) = QR; Q \) not computed \}

7: \( S_k = \text{diag}(R_{ii})_i^{k} (1:k,k+1:2k) \) \( \{S_k = U_k^* \mathcal{A}U_k \text{ is the Rayleigh quotient}\}

8: \( \Lambda_k = \text{eig}(S_k) \{\Lambda_k = \text{diag}(\lambda_i)_i^{k} ; \text{Ritz values, i.e. eigenvalues of } S_k\}

9: for \( i = 1, \ldots, k \) do

10: \( \{\sigma_{\lambda_i}, w_{\lambda_i}\} = \text{svdmin}(\begin{bmatrix} R(1:k,k+1:2k) - \lambda_i R(1:k,k+1:2k) \\ R(k+1:2k,k+1:2k) \end{bmatrix}) \) \{Minimal singular value of \( R_{\lambda_i} \) and the corresponding right singular vector\}

11: \( W_k(:,i) = w_{\lambda_i} \)

12: \( r_k(i) = \sigma_{\lambda_i} \) \{Optimal residual, \( \sigma_{\lambda_i} = \|R_{\lambda_i} w_{\lambda_i}\|_2 \}

13: \( \rho_k(i) = w_{\lambda_i}^* S_k w_{\lambda_i} \) \{Rayleigh quotient, \( \rho_k(i) = (U_k w_{\lambda_i})^* A(U_k w_{\lambda_i}) \}

14: end for

15: \( Z_k = U_k W_k \) \{Refined Ritz vectors\}

Output: \( Z_k, \Lambda_k, r_k, \rho_k \)

Few comments are in order.

**Remark 3.7.** In Line 8, only the eigenvalues are computed, which saves \( O(k^3) \) flops as compared to Line 5 in Algorithm 2.1. The Ritz vectors are then computed in the refinement procedure. Only this option is shown for the sake of brevity; our software implementation allows first computing the Ritz vectors with the corresponding residuals, and then refining only the selected ones, recall the discussion in §3.3.

**Remark 3.8.** The simplest way to implement Line 10 is to compute the full \( 2k \times k \) SVD, and then to take the smallest singular value \( \sigma_{\lambda_i} \) and the corresponding right singular vector \( w_{\lambda_i} \). It is an interesting research problem to develop a more efficient method that will target only \( \sigma_{\lambda_i}, w_{\lambda_i} \) and thus reduce the complexity, from \( O(k^3) \) to \( O(k^2) \) per refined vector.

**Remark 3.9.** Proposition 3.4 identifies the smallest singular value \( \sigma_{\lambda_i} \) as the value of the minimal residual. We should be aware that the smallest singular value may be computed with relatively large error, so in practice it may not represent the residual norm exactly. Although we are not interested in the number of accurate digits of the residual, but in its order of magnitude, we can also compute it as \( \sigma_{\lambda_i} = \|R_{\lambda_i} w_{\lambda_i}\|_2 \), or using the formula (3.4).

**Remark 3.10.** Line 13 is motivated by the fact that for any candidate \( x \) for an eigenvector of \( \mathcal{A} \), the Rayleigh quotient \( \rho = x^* \mathcal{A} x / (x^* x) \) minimizes the residual \( \|\mathcal{A} x - \rho x\|_2 \); see item 3. in Theorem 2.2 . Hence, for \( i = 1, \ldots, k \), the pair \( (U_k w_{\lambda_i}, \rho_k(i)) \) can be used instead of \( (U_k w_{\lambda_i}, \lambda_i) \).
4. Numerical examples

The main goal of the modifications of the DMD algorithm, proposed in §3, is to provide a reliable black-box, data driven software device that can estimate part of the spectral information of the underlying linear operator, and that also can provide an error bound. In this section we illustrate the performance of the new, modified DMD algorithm, first using a synthetic example (to illustrate the fine details of numerical software development, and possible causes of failure even in seemingly simple cases), and then in a concrete example from computational fluid dynamics (to illustrate the benefits of enhanced functionality of the new DMD algorithm in deeper analysis of the original physical problem and its numerical model).

A series of numerical tests show that Algorithm 3.1 is never much worse, but it can be substantially better than Algorithm 2.1. We also show how the added functionality of the new algorithm works in practice.

4.1. Case study: a synthetic example

The first test drive of the code uses randomly generated test matrix that is difficult enough to illustrate the improvements, and to expose weaknesses from which we can learn how to devise better methods. Also, we use this example to stress the importance of careful software implementation. We use Matlab 8.5.0.197613 (R2015a).

The test matrix is generated as $A = e^{-B^{-1}}$ where $B$ is pseudo-random with entries uniformly distributed in $[0, 1]$, and then $A = A/\|A\|_2$. (This normalization is only for convenience, because all involved subspaces remain unchanged, and the Ritz values and the eigenvalues change with the same scaling factor.) Although this example is purely synthetic, it may represent a situation with the spectrum entirely in the unit disc, such as e.g. in the case of an off-attractor analysis of a dynamical system, after removing the peripheral eigenvalues [42].

Then, the initial state $f_1$ is taken with entries from uniform distribution in $[0, 1]$, and the sequence $f_{i+1} = A f_i$ is generated for $i = 1, \ldots, m$. The dimensions are taken as $n = 1000$, $m = 99$. The reference (true) eigenvalues of $A$ are computed using the Matlab function $\text{eig()}$. The subspace spanned by the snapshots $f_i$ contains valuable information on the spectrum of $A$ and we test the ability of a method to extract that information and to correctly report its fidelity. The residuals are computed exactly (using $A$, for testing purposes) and also returned by the algorithms (which have no access to $A$), using only the snapshots and Proposition 3.3 and Proposition 3.4. The truncation threshold in (3.1) is taken as $\epsilon = n \varepsilon$, where $\varepsilon$ is the round-off unit; in Matlab $\varepsilon = \text{eps} \approx 2.2 \cdot 10^{-16}$.

4.1.1. Sequential shifted data, numerical rank $k$ defined as in (3.1). We take $X_m = (f_1, f_2, \ldots, f_{99})$, $Y_m = (f_2, f_3, \ldots, f_{100})$, and give them as inputs to both Algorithm 2.1 and Algorithm 3.1. The difference in the quality of the computed Ritz pairs, illustrated in Figure 1 and Figure 2, is striking. We first note that DMD has returned only 8 Ritz pairs, while DDMD_RRR returns 27. The reason is in a sharp decay of the norms $\|f_i\|_2$, forcing the truncated SVD into declaring low numerical rank. As a result, small number of eigenvalues is approximated from a subspace of small dimension, and large portion of the supplied information is truncated, i.e. left unexploited.
4.1.2. Sequential shifted data, numerical rank hard-coded as $k = 27$. In this experiment, we set $k = 27$, so both algorithms will take best 27-dimensional subspaces from their SVD decompositions; hence Algorithm 3.1 will return the same output as in §4.1.1, and Algorithm 2.1 is allowed to use the same dimension\textsuperscript{10} 27, instead of 8 as it would have been taken using (3.1). This is an artificial scenario, as the algorithm must determine the most appropriate value of $k$ autonomously for each given input – the ramifications of choosing $k$ inappropriately are illustrated in §4.1.1. However, it is instructive to see and analyze the results returned by Algorithm 2.1.

The computed Ritz values, shown in Figure 3, reveal an interesting fact – we see only 12 of them (pluses, +) because many are computed as tiny numbers clustered around zero, and the overall residuals, displayed on Figure 4, show no substantial improvement.

\textsuperscript{10}Note that this means that both algorithms use subspaces of same dimensions, but not necessarily the same subspace.
Furthermore, the formula (3.4) for the residuals collapsed when used in Algorithm 2.1, returning the values that were underestimated by several orders of magnitude, as shown by the ratios

$$\eta_i = \frac{\| (Y_m V_k \Sigma_k^{-1}) W_k(:,i) - \lambda_i (U_k W_k(:,i)) \|_2}{\| A (U_k W_k(:,i)) - \lambda_i (U_k W_k(:,i)) \|_2} \equiv 1, \quad i = 1, \ldots, k. \tag{4.1}$$

The computed values of $\eta_i$ for Algorithm 2.1 are below $10^{-10}$, i.e. the residuals returned by the algorithm are e.g. $10^{10}$ times smaller than the actual values (see Figure 4), yielding wrong conclusion that the approximations are good.

Figure 3: (Example in §4.1.2, with sequential data) Comparison of the approximations of the eigenvalues of $A$ (circles ◦) by the Ritz values computed by Algorithm 2.1 (pluses +) and Algorithm 3.1 (crosses, ×), with the same numerical rank $k = 27$. The plus in the vicinity of the origin is actually a cluster of 16 pluses (+) clustered around zero with the Ritz values in modulus equal 5.4e-05, 7.1e-07, 2.0e-08, 1.4e-10, 1.1e-12, 3.8e-14, 2.8e-16, 6.5e-19, 7.5e-20, 1.9e-22, 1.0e-24, 6.9e-28, 6.6e-31, 3.4e-31, 3.6e-33.

Figure 4: (Example in §4.1.2, with sequential data) Comparison of the residuals of the Ritz pairs computed by Algorithm 2.1 (pluses +) and Algorithm 3.1 (crosses, ×), with the same numerical rank $k = 27$. The second plot shows the ratios (4.1) of the returned and the true residuals computed using $A$ explicitly, as in the denominator in (4.1).
4.1.3. Discussion. It is important to understand why the DMD did not take advantage of the supplied information on the numerical rank.

What went wrong? The key insight is in Figure 5. Only 10 singular values used in Algorithm 2.1 (computed in Line 2., using the Matlab function svd()) are accurate to a satisfactory level; the remaining 17 are mostly severely overestimated, as the true singular values of $X_m$ decay very rapidly down to the level of $10^{-150}$. In Figure 5, we display those values, together with the singular values of $X_m$, computed by calling the same Matlab function svd(). Only the first 10 singular values agree to some level of accuracy. Due to the large upward error, the formula for $A U_k$ as $Y_m V_k \Sigma_k^{-1}$ fails and the trailing columns of $Y_m V_k \Sigma_k^{-1}$ are severely underestimated because the computed $\Sigma_k^{-1}$ is not big enough, see Figure 6. (We have checked that the minimal singular value of $A$ is larger than $10^{-6}$, and the minimax theorem implies that the norms of the columns of $A U_k$ must be also larger than $10^{-6}$.) As a result, the trailing 17 columns of the matrix $S_k$ are entirely wrong and mostly very tiny in norm.

Further, we should keep in mind that determining numerical rank (and truncating the SVD of the data matrix) is a delicate procedure that must consider also the scaling of the data, statistical information on the error in the data etc. Hard coded universal threshold, without any data preprocessing, is not always the best one can do.

Why it went wrong? The difference between the computed singular values is due to the fact that Matlab uses different algorithms in the svd() function, depending on whether the singular vectors are requested on output. The faster but less accurate method is used in the call $[U, S, V] = \text{svd}(X_m,'econ')$. To our best knowledge, Matlab documentation does not provide sufficient information, except that the SVD is based on the LAPACK subroutines. It is very likely that the full SVD, including the singular vectors, is computed using the divide and conquer algorithm, xGESDD() in LAPACK and, for computing only the singular values, $S = \text{svd}(X)$ calls the QR SVD, xGESVD() in LAPACK. This policy of not specifying the method, and thus hiding the key information on the numerical reliability of the output may cause numerical errors to inconspicuously degrade the overall computation. Without providing information that may be mission critical, software developers often choose to prefer faster method by sacrificing numerical accuracy, without informing the user on possible consequences. Note that the same fast xGESDD() subroutine is (to the best of our knowledge) under the hood of the Python function numpy.linalg.svd. Numerical robustness of both xGESVD() xGESDD() depends on $\kappa_2(X_m)$, and if one does not take advantage of the fact that scaling is allowed, the problems illustrated in this section are likely to happen. For numerically more robust computation of the SVD independent of scaling, see [43], [44], [45].
4.1.4. Odd-even data splitting. We conclude this experiment with another run, but this time we halve the dimensions of the data subspaces by taking

\[
X_{m/2} = (f_1, f_3, \ldots, f_{97}, f_{99}), \quad Y_{m/2} = (f_2, f_4, \ldots, f_{98}, f_{100}),
\]

i.e. split the snapshots by taking every other into \(X_{m/2}\). The results are given in Figure 7. Again, Algorithm 3.1 has identified more Ritz pairs with smaller residuals than Algorithm 2.1, even with the initial space of half of the dimension, as compared with sequential shifted data. It is worth mentioning that the complexity of the first step in all DMD methods is quadratic in the column dimension, thus the reduction here is with the factor of four.
Figure 7: (Example in §4.1, odd-even data splitting) Left: The eigenvalues of $A$ (circles ◦), the Ritz values computed by Algorithm 2.1 (pluses +) and Algorithm 3.1 (crosses, ×), with the same threshold in the truncation criterion for determining the numerical rank as defined in (3.1). Right: Comparison of the residuals.

4.2. Flow around a cylinder

In this example, we illustrate the benefits of the new proposed techniques in a concrete application – numerical Koopman spectral analysis of the wake behind a circular cylinder. The well studied and understood model of laminar flow around a cylinder is based on the two-dimensional incompressible Navier-Stokes equations

$$
\begin{align*}
\frac{\partial \mathbf{v}}{\partial t} &= - (\mathbf{v} \cdot \nabla) \mathbf{v} + \nu \Delta \mathbf{v} - \frac{1}{\rho} \nabla p, \\
\nabla \cdot \mathbf{v} &= 0,
\end{align*}
$$

where $\mathbf{v} = (v_x, v_y)$ is velocity field, $p$ is pressure, $\rho$ is fluid density and $\nu$ is kinematic viscosity. The flow is characterized by the Reynolds number $Re = v^* D / \nu$ where, for flow around circular cylinder, the characteristic quantities are the inlet velocity $v^*$ and the cylinder diameter $D$. For a detailed analytical treatment of the problem see [46], [47]; for a more in depth description of the Koopman analysis of fluid flow we refer to [10], [3].

We used the velocity and pressure data from a numerical simulation described in detail in [48]. We thank Dr. Stefan Ivić and Dr. Nelida Črnjarić-Žic for providing us with the data. For the sake of completeness and for the reader’s convenience, we briefly describe the numerical procedure.

The flow, with $x$ denoting the streamwise direction, is numerically simulated in the rectangular domain $[-12D, 20D] \times [-12D, 12D]$, with appropriate boundary conditions. The inlet velocity is set to $\mathbf{v} = (v^*, 0)$, and the outlet $\partial \mathbf{v} / \partial x = 0$; on the side bounds the slip wall condition $v_y = 0$ is imposed, while the no-slip wall condition $\mathbf{v} = (0, 0)$ is set on the cylinder edge. For the pressure, homogeneous Neumann boundary condition is used for all domain boundaries except for the outlet where $p = 0$.

With this setup, the laminar incompressible two-dimensional flow model is for $Re = 150$ simulated numerically using OpenFOAM’s icoFOAM solver [49] on a structured grid containing 86000 cells, and with time step $\Delta t = 0.01$ s.

The observables for the DMD analysis, two components of the velocity and the pressure, are collected as follows: After the flow was fully developed, the snapshots are extracted from a time interval of 100 seconds with the time step of $\Delta t = 0.1$ s. All three state variables obtained on the structured CFD mesh are interpolated, and then evaluated on a $201 \times 101$ uniform rectangular grid.
in the $[-10, 30] \times [-10, 10]$ sub-domain. The $201 \times 101$ array data structure is then, for each variable, reshaped into an $20301 \times 1$ array.

The thus obtained data matrix of the snapshots has the block row structure with each snapshot $f_i$ containing as observables the components of the velocities $v_{ix}, v_{iy}$, and the pressure $p_i$,

$$f_i = \begin{pmatrix} v_{ix} \\ v_{iy} \\ p_i \end{pmatrix} \equiv \hat{f}_i,$$

$v_{ix}, v_{iy}, p_i \in \mathbb{R}^{20301}$, $i = 1, \ldots, 1001$.

We also perform the test with only the components of the velocities as observables, i.e. with $\hat{f}_i = (v_{ix}, v_{iy})^T$. For an interpretation of the difference between using $f_i$ and $\hat{f}_i$, recall §2.1.

Our aim with this example is to illustrate:

- the advantage of using the data driven residuals to select good Ritz pairs;
- the advantage of combining the computable residual bounds with the spectral perturbation theory to obtain computable a posteriori error bounds in the approximate eigenvalues
- the potential of the Ritz vector refinement procedure (reducing the residual with better approximate eigenvectors, and improving the approximate eigenvalues)
- the effect of choosing and scaling of the observables (in the context of spectral analysis of the underlying Koopman operator)

Let us now discuss the results.

Figure 8 shows in the left panel the residuals computed as explained in §3.2; for both sets of data the computed Ritz pairs are ordered with increasing residuals. This allows automatic selection of those that better approximate the eigenpairs of the underlying operator. For instance, with a threshold for the residual set to $5 \cdot 10^{-4}$, the selected Ritz values are shown on the right panel. We have checked that the selected Ritz values are close to the unit circle, up to an error of order $10^{-5}$, with quite a few that are up to $O(10^{-7})$ on the circle. The numerical data from the cylinder flow is selected so that the dynamics has stabilized to evolution on the limit cycle attractor. On the attractor, the Koopman operator is unitary with respect to an underlying invariant measure [17]. Thus, the eigenvalues that we obtain are expected to reside close to the unit circle.

As expected, the $f_i$'s carry more information than the $\hat{f}_i$'s, and more Ritz values have been selected. However, here we should keep in mind that in a numerical scheme the pressure values might be computed less accurately than the velocities, so that the advantage of extra information is not necessarily fully exploited. The issues of different level of noise in the observables, different physical nature and scalings (units) of the variables, and proper choice of the norm to measure the residual are important; see §8 for an algebraic framework and numerical algorithms.
For the selected pairs, the refinement procedure of §3.3 additionally reduces the residuals, as shown on Figure 9.

In fact, since this reduction of the residuals is achieved by improving the Ritz vectors, we can achieve even smaller residuals by refining the Ritz values as in item 3. in Theorem 2.2 and Remark 3.10. Furthermore, the classical Bauer–Fike type error bound from item 5. in Theorem 2.2 (also used in Proposition 3.3) gives useful relative error bounds for the Ritz values close to the unit circle, in particular because the underlying $A$ is nearly unitary (in an appropriate discretized inner product) for sufficiently large number of data points (see the discussion in [50] on the convergence of finite-dimensional approximations to the Koopman operator, and notice that the underlying Koopman operator for this data is nearly unitary, as discussed above), so the condition number of its eigenvalues is close to one. For instance, we know for sure that our computed values $\lambda_i$ approximate some eigenvalues of $A$ up to $-\lceil \log_{10}(r_k(i)) \rceil$ digits of accuracy; from Figure 9 we see
that this means between four and nine correct digits.\textsuperscript{11}

Without separate analysis we cannot make an analogous statement with respect to the eigenvalues of the underlying Koopman operator; recall Remark 2.1. This important topic of including the operator discretization error in the overall error bound is the subject of our future research.

The approximate Koopman eigenvalues are computed in the vicinity of unit circle as

\[ \mathcal{R}_j = \frac{\log |\lambda_j| + i \arg \lambda_j}{2\pi \Delta t} \approx \frac{1}{2\pi \Delta t} i \arg \lambda_j. \]  

(4.4)

Using the selected Ritz values (and their refinements) the computed \( \mathcal{R}_j \)'s are as in Figure 10.

![Figure 10](image)

Figure 10: (Example in §4.2) Left panel: Approximate Koopman eigenvalues computed by (4.4) and Algorithm 3.1 with velocities as observables (circles \( \circ \)) and with the additional refinement (crosses, \( \times \)). The refined Ritz values are defined as the Rayleigh quotients with the refined refined vectors; see item 3. in Theorem 2.2 and Remark 3.10. Here the movement closer to the imaginary axis by virtue of (4.4) means getting closer to the unit disc in Figure 8. Observe the scale on the real axis. Right panel: The bar graph of the imaginary parts of the Ritz values selected by the residual thresholding. The cyclic group structure of the eigenvalues on the unit circle is nicely visualized in the \( \mathcal{R}_j \)'s; cf. (4.4). These values nicely correspond to the analytically derived formulas for the Koopman eigenvalues of the flow [46].

We conclude this experiment with an observation that only illustrates the issue of choosing properly scaled variables. This, together with the numerical linear algebra framework set in §8, opens further important themes in numerical analysis of spectral properties of Koopman operator and it is the subject of our future research.

\textsuperscript{11}Here we tacitly assume that the residuals are computed sufficiently accurately, and that the error analysis is done using appropriate norms. See Remark 8.5 and Theorem 8.6.
5. Implications to the Exact DMD

In the Exact DMD [12, Algorithm 2], the action of the unaccessible operator $\hat{A}$ is mimicked by $\hat{A} = Y_m X_m^\dagger$. Since the null-space of $X_m$ is a subspace of the null space of $Y_m = AX_m$, then $\hat{A} X_m = Y_m$; for details we refer to [12]. Based on the available information, we cannot distinguish $A|X_m$ from $\hat{A}|X_m$. Further, Exact DMD cleverly uses the fact that $Y_m$ is $\hat{A}$-invariant ($\hat{A} Y_m = Y_m (X_m^\dagger Y_m)$) and, thus, it contains exact eigenvectors of $\hat{A}$. After computing the Rayleigh quotient matrix $S_k$ and its eigenpairs $\Lambda_k, W_k$ ($S_k W_k = W_k \Lambda_k$) in the same way as the DMD does, instead of the Ritz vectors $Z_k = U_k W_k \Lambda_k^{-1}$, the approximate eigenvectors are computed as $Z_k = Y_m (V_k \Sigma_k^{-1} W_k) \Lambda_k^{-1}$. It is easily checked that $Z_k = A U_k W_k \Lambda_k^{-1}$.

As in the case of DMD, the details of the tolerances for the truncated SVD used in Exact DMD are lacking in the literature. Here we suggest, following the discussion in §3.1.1, to apply the same strategy as in Algorithm 3.1. It should be noted here that the definition of $\hat{A}$ is invariant under the scaling discussed in §3.4. Indeed, $(Y_m D) (X_m D)^\dagger = Y_m X_m^\dagger$ for any $m \times m$ nonsingular matrix $D$. However, the numerical rank and the truncated SVD of $X_m$ will heavily depend on the scaling. In fact, since the Exact DMD and DMD compute the same Ritz eigenvalues, repeating the experiment from §4.1 would reveal the same problem. On the other hand, if we apply scaling, the results are as in Figure 12. This clearly demonstrates that Exact DMD as well can benefit from careful scaling of the data.\footnote{Here we reiterate the comment from the end of §3.4.}

Figure 11: (Example in §4.2) Comparison of the residuals of the Ritz pairs computed by Algorithm 3.1 with velocities and pressure as observables, but with differently scaled pressure values (i.e. to obtain the real pressure). The values of the corresponding refined residuals are also shown. On the left, the residuals of the top 67 pairs (that are below the residual threshold set to $5 \cdot 10^{-4}$) are obtained if the scaling factor is $\rho = 1.225$. On the right panel, the data for 215 selected Ritz pairs corresponds to the scaling $\rho = 1000$.}

\footnote{Here we reiterate the comment from the end of §3.4.}
Unfortunately, we cannot compute the residuals of the Ritz pairs with the so called “exact eigenvectors” \( Z_k = Y_m(V_k \Sigma_k^{-1} W_k) \Lambda_k^{-1} \). Namely, if we use all snapshots, we cannot test the quality of the Ritz vectors as we do not know how to apply \( A \) on them – the data does not contain enough information for computing \( A Y_m \), which is needed to compute \( A Z_k(\cdot; i) \). In the case of sequential data (2.4), the convergence mechanism of power iterations (see Remark 2.3) may help.

**Proposition 5.1.** Let \( X_m \) and \( Y_m \) be as in (2.4), and let \( y = \sum_{i=1}^{m} \eta_i f_{i+1} \in Y_m \). Then

\[
\Lambda y = \hat{A} y + \eta_m \Lambda (I - X_m X_m^\dagger) f_{m+1} \equiv \hat{A} y + \eta_m \Lambda r_{m+1},
\]

where \( r_{m+1} = f_{m+1} - X_m (X_m^\dagger f_{m+1}) \) is the optimal residual as in §2.2.1. In particular, if \( y \) is an eigenvector with \( \hat{A} y = \lambda y \), then \( \Lambda y - \lambda y = \eta_m \Lambda r_{m+1} \).

Hence, if \( f_1 \) is well chosen and \( m \) is big enough, then \( r_{m+1} \) will be small and (5.1) becomes useful, provided that \( \Lambda \) does not amplify the residual. For general data, such that we only have \( Y_m = \Lambda X_m \), not much can be said on \( \Lambda Y_m \).

6. Memory efficient DMD

In the case of high resolution numerical simulations the ambient space dimension \( n \) (determined by the fineness of the grid) can be in millions, and mere memory fetching and storing is the bottleneck that precludes efficient computation, let alone flop count. For instance, [51], [52] report computation with the dimension \( n \) of half billion on a massively parallel computer. Tu and Rowley [40] discuss an implementation of DMD that has been designed to reduce computational work and memory requirements; this approach is also adopted in the library modred [53]. Sayadi and Schmid [51] propose a parallelized SVD of \( X_m \), based on the QR factorization, which improves the performance of the whole DMD since the SVD is its most expensive part.

In fact, it is a well known technique to preprocess the SVD with a QR factorization, \( X_m = Q_x \left( \begin{array}{c} R_x \\ 0 \end{array} \right) \), and then to compute the SVD of the \( m \times m \) upper triangular matrix \( R_x \), and finally to assemble the left singular vectors using the unitary factor \( Q_x \); see [54]. The optimal ratio \( n/m \) for this strategy depends on a concrete SVD algorithm and on a computing hardware and software.

13In the case of sequential data, we can decide not to use \( f_{m+1} \) in the DMD construction, so we can use it for computing the residuals. In that case, the Exact DMD works with the subspaces of dimension \( m - 1 \).
For example, in the LAPACK [55] library, the xGESVD and xGESDD subroutines\(^{14}\) for computing the SVD with the bidiagonalization based QR SVD and the divide and conquer methods, the crossover point is obtained by calling e.g. ILAENV( 6, ‘xGESVD’, JOBU // JOBVT, M, N, 0, 0 ). The LAPACK’s Jacobi SVD subroutines xGEJSV start with the QR factorization as well. In the case \(n \gg m\), the advantage is obvious and in particular because the QR factorization of a tall and skinny matrix can be optimized for various computing platforms [56].

Hence, the modification proposed in [51] is already contained in the LAPACK SVD subroutines.

\subsection{QR Compressed DMD}

On the other hand, all action contained in the sequential data\(^{15}\) in (2.4) at step \(i = m\) is confined to at most \(m + 1\) dimensional range of \(F_{m+1} = (f_1, \ldots, f_{m+1})\). Hence, all previously analyzed algorithms can be represented in \(\text{range}(F_{m+1})\), and it only remains to construct a convenient orthonormal basis. To that end, let

\[ F_{m+1} = Q_f \begin{pmatrix} R_f & 0 \\ 0 & 0 \end{pmatrix} = \tilde{Q}_f R_f, \quad Q_f^* Q_f = I_n, \]

be the QR factorization of \(F_{m+1}: \tilde{Q}_f = Q_f(:,1 : m + 1), \text{range}(\tilde{Q}_f) \supseteq \text{range}(F_{m+1}), \) and \(R_f\) is \((m + 1) \times (m + 1)\) upper triangular. (If \(F_{m+1}\) is of full column rank, then \(\text{range}(\tilde{Q}_f) = \text{range}(F_{m+1})\).)

If we set \(R_x = R_f(:,1 : m), R_y = R_f(:,2 : m + 1)\), then

\[ X_m = \tilde{Q}_f R_x, \quad Y_m = \tilde{Q}_f R_y; \quad R_x = \begin{pmatrix} * & * & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \ddots & \ddots \\ \cdots & \cdots & \cdots & \cdots & \ddots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}, \quad R_y = \begin{pmatrix} * & * & \cdots & \cdots & \cdots \\ \cdots & \cdots & \ddots & \ddots & \ddots \\ \cdots & \cdots & \ddots & \cdots & \ddots \\ \cdots & \cdots & \cdots & \cdots & \ddots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}, \]

and, in the basis of the columns of \(\tilde{Q}_f\), we can identify \(X_m \equiv R_x, Y_m \equiv R_y\), i.e., we can think of \(X_m\) and \(Y_m\) as \(m\) snapshots in an \((m + 1)\) dimensional space. Recall that \(m \ll n\).

If we apply the DMD algorithm to this data, it will return the matrix of approximate eigenvalues \(\Lambda_k\) with the corresponding eigenvectors as the columns of \(\tilde{Z}_k \in \mathbb{C}^{(m + 1) \times k}\). To transform this output in terms of the original data, it suffices to lift the eigenvectors as \(Z_k = \tilde{Q}_f \tilde{Z}_k\). Also note that this change of basis is actually an implicit unitary similarity applied to \(A\), since

\[ X_m = Q_f \begin{pmatrix} R_x & 0 \\ 0 & 0 \end{pmatrix}, \quad Y_m = \Lambda Q_f \begin{pmatrix} R_x & 0 \\ 0 & 0 \end{pmatrix} = Q_f \begin{pmatrix} R_y & 0 \\ 0 & 0 \end{pmatrix} \implies \begin{pmatrix} R_y \\ 0 \end{pmatrix} = (Q_f \Lambda Q_f) \begin{pmatrix} R_x \\ 0 \end{pmatrix}. \]

(6.3)

Note that \(R_y = (\tilde{Q}_f \Lambda \tilde{Q}_f^*) R_x\), and, analogously, in the invertible case, \(R_x = (\tilde{Q}_f^* \Lambda^{-1} \tilde{Q}_f) R_y\). In the case of the exact DMD with \(\tilde{A} = Y_m X_m^\dagger\) we have

\[ X_m^\dagger = \begin{pmatrix} R_x^\dagger \\ 0 \end{pmatrix} Q_f^* = R_x^\dagger \tilde{Q}_f^*, \quad \tilde{A} = Y_m X_m^\dagger = Q_f \begin{pmatrix} R_y R_x^\dagger & 0 \\ 0 & 0 \end{pmatrix} Q_f^* = \tilde{Q}_f R_y R_x^\dagger \tilde{Q}_f. \]

The benefit of this compressed DMD is that the dimension \(n\) is reduced to much smaller dimension \(m + 1\) using the QR factorization that can be optimized for tall and skinny matrices [56]. The cost of this reduction is comparable to the initial QR factorization of \(X_m\) proposed in [51]. However, unlike [51], where this is considered only a preprocessing step toward more efficient SVD of \(X_m\) in DMD running in the original \(n\)-dimensional space, our compressed DMD runs entirely in the \((m + 1)\) dimensional subspace of \(\mathbb{C}^n\), with the nice interpretation (6.3).

\(^{14}\)Following the LAPACK naming convention, in the subroutine name ‘xGESVD’, \(x\) is one of \(S, D, C, Z\) for the four data types.

\(^{15}\)We first consider sequential data and later generalize to general \(X_m\) and \(Y_m = \Lambda X_m\).
This compression trick applies to Algorithm 2.1, Algorithm 3.1 as well as to the Exact DMD. It especially greatly improves the efficiency of Algorithm 3.1 because it reduces the overhead of computing the refined Ritz vectors. For the readers convenience, in Algorithm 6.1, we show the compressed version of Algorithm 3.1; for the other two algorithms, *mutatis mutandis*, the corresponding compressed versions are straightforward.

**Algorithm 6.1** \([Z_k, \Lambda_k, r_k, \rho_k] = \text{DDMD} \_\text{RRR} \_\text{C} (F_m; \epsilon) \) \{QR Compressed Refined DMD\}

**Input:**
- \(F_{m+1} = (f_1, \ldots, f_m, f_{m+1})\) that defines a sequence of snapshots pairs \(f_{i+1} = Af_i\). (Tacit assumption is that \(n\) is large and that \(m \ll n\).)
- Tolerance level \(\epsilon\) for numerical rank determination.

1. \([\hat{Q}_f, R_f] = qr(F_{m+1}, 0)\); \{thin QR factorization\}
2. \(R_x = R_f(1 : m + 1, 1 : m), R_y = R_f(1 : m + 1, 2 : m + 1)\); \{New representaitons of \(X_m, Y_m\)\}
3. \([\hat{Z}_k, \Lambda_k, r_k, \rho_k] = \text{DDMD} \_\text{RRR} (R_x, R_y; \epsilon)\); \{Algorithm 3.1 in \((m+1)\)-dimensional ambient space\}
4. \(Z_k = Q_f \hat{Z}_k\)

**Output:** \(Z_k, \Lambda_k, r_k, \rho_k\)

**Remark 6.1.** In an efficient software implementation, the matrix \(\hat{Q}_f\) is not formed explicitly. Instead, the information on the \(m + 1\) Householder reflectors used in the factorization is stored in the positions of the annihilated entries (see e.g. zGEQRF in LAPACK) and then one can apply such implicitly stored \(Q_f\) and compute \(Z_k = Q_f \left( \begin{smallmatrix} \hat{Z}_k \\ 0 \end{smallmatrix} \right) \equiv \hat{Q}_f \hat{Z}_k\) (see e.g. zORMQ in LAPACK).

**Remark 6.2.** In the case of general data \(X_m, Y_m = AX_m\), instead of the QR factorization (6.1), we can compress the data onto the \(2m\) dimensional subspace using the QR factorization

\[
(X_m \hspace{0.5cm} Y_m) = Q_{xy} \begin{pmatrix} R_{xy} \\ 0 \end{pmatrix} = \hat{Q}_{xy} R_{xy},
\]

and analogously to (6.2), the low dimensional representations \(X_m = \hat{Q}_{xy} R_x, Y_m = \hat{Q}_{xy} R_y\), where \(\hat{Q}_{xy} = Q_{xy}(:, 1 : 2m), R_x = R_{xy}(1 : 2m, 1 : m), R_y = R_{xy}(1 : 2m, m + 1 : 2m)\). As in the case of sequential data (see Algorithm 6.1), the DMD will compute with \(R_x\) and \(R_y\), and the Ritz vectors will be transformed back to \(C^n\) using \(\hat{Q}_{xy}\).

**Remark 6.3.** Using (6.1), (6.2) as in Algorithm 6.1 facilitates efficient updating/down-dating if we keep adding new snapshots and/or dropping the ones at the beginning, e.g. if the snapshots are taken from a sliding (in discrete time steps) window that may even be of variable width. Also, rows (observables) may be added/removed and the decomposition recomputed from the previous one. The key is that only the QR factorization (6.1) needs to be updated, using the well established algorithms (see e.g. [57]) that are also available for parallel computing (see e.g. [58]); the rest of the computation takes place in \((m+1)\)-dimensional space. We omit the details for the sake of brevity.

### 7. Forward–Backward DMD (F-B DMD)

Dawson et al. [13] (see also [31, §8.3]) proposed a Forward–Backward version of DMD designed to reduce the bias caused by sensor noise. The idea is to run (the exact) DMD also backward by swapping \(X_m\) and \(Y_m\), thus working implicitly with \(A^{-1}\). If the corresponding Rayleigh
quotients are $S_k$ and $S_{k,\text{back}}$ (for backward DMD) then, assuming nonsingularity of both Rayleigh quotients, the projected operator is approximated by

$$\hat{S}_k = \sqrt{S_k S_{k,\text{back}}^{-1}}.$$ 

The intuition is that the bias in the eigenvalues of $S_k$ and $S_{k,\text{back}}$ will be subject to cancellation in the product $S_k S_{k,\text{back}}^{-1}$. Now, if $\hat{S}_k w_i = \mu_i w_i$, then for every nonzero $\mu_i$, the corresponding approximate eigenvector is $U_r w_i$ (standard DMD) or $\mu_i^{-1} Y_m V_i \Sigma^{-1} w_i$ (Exact DMD, preferred in [13, Algorithm 1, Algorithm 3]). Numerical examples provided in [13] and [31, §8.3] demonstrate the efficacy of the F-B DMD. However, numerical details of the effective implementation of the method remain open for discussion with a potential for improvement, in particular with respect to the usage of the matrix square root.

In [13], the problem of computing the matrix square root is discussed in detail, in particular with respect to the non-uniqueness and it is proposed to resolve the ambiguity by choosing the root closest to $S_k$ (or $S_{k,\text{back}}^{-1}$). This may be difficult to implement in practice, as computing the matrix square root is a nontrivial task, and this additional constraint would make it even more difficult. Furthermore, in a software implementation of the DMD, one usually resorts to state of the art software packages, such as e.g. Matlab, and uses black-box routine. For instance, the function $\text{sqrtm}()$ (Matlab) computes the branch of the square root such that all eigenvalues have nonnegative real parts; each real negative eigenvalue of $S_k S_{k,\text{back}}^{-1}$ will generally produce in the spectrum of $\sqrt{S_k S_{k,\text{back}}^{-1}}$ a purely imaginary eigenvalue in the upper half plane. Moreover, the computed square root is in general complex, even when $S_k S_{k,\text{back}}^{-1}$ is real. (It is known that real matrix possesses a real square root if and only if each of its elementary divisors with real negative eigenvalues occurs an even number of times [59, Theorem 5].)

In [31, Algorithm 8.6], the square root is computed in Matlab as the matrix fractional power, $(S_k S_{k,\text{back}}^{-1})^{0.5}$. This function is also numerically difficult way to compute the square root, see e.g. [60]. For more systematic treatment of matrix square root function we refer to [61, Chapter 6].

### 7.1. Matrix root free Forward–Backward DMD

Closer look at the Forward–Backward DMD reveals that, intrinsically, we do not need $\hat{S}_k = \sqrt{S_k S_{k,\text{back}}^{-1}}$, but (only) its spectral decomposition. Hence, the FB-DMD can be implemented without invoking the matrix square root; instead we deploy the spectral mapping theorem. The result is the a Matrix Root–Free Forward–Backward DMD, outlined in Algorithm 7.1:

**Remark 7.1.** In Line 5., the complex square root function computes the principal value, $\sqrt{\omega_i} = \sqrt{|\omega_i| + \Re(\omega_i)}/2 \pm i\sqrt{|\omega_i| - \Re(\omega_i)}/2$. Hence $\Re(\sqrt{\omega_i}) \geq 0$, $i = 1, \ldots, k$. If $M_k$ is real, its complex (non-real) eigenvalues occur in complex conjugate pairs and their square roots will inherit closedness under conjugation. (In Matlab, the sign of the imaginary part of $\sqrt{\omega_i}$ matches the sign of the imaginary part of $\omega_i$. For more details on machine implementation of complex functions see [62].) Note however that any real negative $\omega_i$ yields purely imaginary $\lambda_i = \pm \sqrt{-\omega_i}$, and the conjugacy symmetry will be lost. One way to choose the sign in the definition $\lambda_i = \pm \sqrt{\omega_i}$ is e.g. by comparison with $W_k(:,i)^* S_k W_k(:,i)$, or with the Ritz values of $S_k$.

**Remark 7.2.** For more efficient computation, [13] proposes optional projection of both $X_m$ and $Y_m$ onto the POD basis for $X_m$, so that the F-B DMD runs in the POD basis. An alternative for both

A nonsingular matrix $M$ with $d$ distinct eigenvalues has $2^d$ square roots which are matrix functions of $M$. 

\[\text{aimdyn:201708.004v1}\]
Algorithm 7.1 \([Z_k, \Lambda_k, r_k] = \text{DMD\_FB\_MRF}(F_m; \epsilon)\) \{Matrix–Root–Free Forward–Backward DMD\}

**Input:**
- \(X_m = (x_1, \ldots, x_m), Y_m = (y_1, \ldots, y_m) \in \mathbb{C}^{n \times m}\) that define a sequence of snapshots pairs \((x_i, y_i) \equiv A x_i\). (Tacit assumption is that \(n\) is large and that \(m \ll n\).
- Tolerance level \(\epsilon\) for numerical rank determination.

1. Apply a DMD algorithm to \((X_m, Y_m \equiv AX_m)\) and compute the POD basis \(U_k\) and the Rayleigh quotient \(S_k\). Use a threshold strategy to determine the reduced dimension \(k\).
2. Apply the same DMD scheme to \((Y_m, X_m \equiv A^{-1}Y_m)\), but with fixed dimension \(k\) of the POD basis and compute the Rayleigh quotient \(S_{k, \text{back}}\).
3. \(M_k = S_k S_{k, \text{back}}^{-1}\); \{Without explicit inverse, e.g. \(M_k = S_k/S_{k, \text{back}}\) (Matlab).\}
4. \([W_k, \Omega_k] = \text{eig}(M_k)\) \{\(\Omega_k = \text{diag}(\omega_i)\), \(M_k W_k(:,i) = \omega_i W_k(:,i); \|W_k(:,i)\|_2 = 1\}\)
5. \(\lambda_i = \pm \sqrt{\omega_i}, \Lambda_k = \text{diag}(\lambda_i)\); \{Choose the signs carefully; see Remark 7.1.\}
6. \(Z_k = U_k W_k\) \{Ritz vectors. Alternatively, use the vectors from the Exact DMD as in [13].\}
7. \(r_k(i) = \| (Y_m V_k \Sigma_k^{-1}) W_k(:,i) - \lambda_i Z_k(:,i) \|_2, i = 1, \ldots, k\).

**Output:** \(Z_k, \Lambda_k, r_k\)

the original FB-DMD and Algorithm 7.1 is to use the QR Compressed version of the DMD (§6.1), and thus to work with \(R_x\) and \(R_y\) instead of \(X_m\) and \(Y_m\), respectively. This is an attractive option because of availability of efficient software implementations of the QR factorization of tall and skinny matrices, see e.g. [56]. Since this corresponds to an orthogonal (or unitary) transformation of the data, the model for the noise in the derivation of the FB-DMD remains valid.

8. General framework: weighted DMD

The truncated SVD provides optimal low rank approximation to \(X_m\) by favoring most energetic modes, all in the Euclidean inner product structure that is usually taken as default. This may not be optimal and restricting the Rayleigh-Ritz procedure to only a subspace of \(X_m\) may prevent it from finding low energy but relevant modes. Further, truncating the SVD of \(X_m\) in the canonical Euclidean structure (and deeming the vectors large or small in the Euclidean norm) ignores that the numerical values in a snapshot \(f_i\) represent physical variables such as pressure and velocity components, each in its own units.\(^{17}\) Clearly, care must be exercised in computing and interpreting norm of such a vector. Also, if the components of \(f_i\) are measured values contaminated with noise, then statistical information (covariance) may be available, and must be taken into account.

Moreover, dimension reduction by a Galerkin/POD projection in the framework of an inappropriate Hilbert space structure may cause loss of important physical properties (e.g. stability) and some other important principles (e.g. conservation) may get lost in the projection. In such cases, it may be crucial to use an appropriate notion of energy i.e. the inner product that is in the discretized space defined by a positive definite matrix \(M\) \((\langle x, y \rangle_M = y^* M x)\); recall also that discretization of a continuous inner product defined by an integral yields a weighted inner product defined through the appropriate quadrature weights. For an example of construction of \(M\) see e.g. [64]. In some cases, the natural inner product structure is defined by the positive definite solution \(M\) of a Lyapunov matrix equation, see e.g. [65, §6.1], [66], [67]. As an illustration of the importance of the first step we refer to [68], [69], [70] for examples how different inner products yield completely different results – certainly not desirable situation for real world applications.

\(^{17}\)See e.g. [63], where separate DMD is computed for each variable.
If a weighted Galerkin/POD projection is combined with other approximation techniques such as the DEIM [71], [72], [73], [74] or the QDEIM [75], then it is important that the overall computation and the error bounds are done in the same (weighted) inner product; for a recent development of this idea in the context of DEIM see [76], [77].

Hence, it seems natural and important to formulate the first step in the Schmid’s DMD algorithm (as well as in other versions of the DMD) in a general weighted inner product space.

To set the scene, we recall the weighted POD [78], summarized in Algorithm 8.1.

**Algorithm 8.1** $[\hat{U}_k, L, [\hat{U}_k]] = \text{Weighted}_{\text{POD}}(X_m; M; \epsilon)$ (See e.g. [78].)

**Input:**
- The matrix of snapshots $X_m = (f_1, f_2, \ldots, f_m) \in \mathbb{C}^{n \times m}$. (Assumed $m < n$.)
- Hermitian $n \times n$ positive definite $M$ that defines the inner product.
- Tolerance level $\epsilon$ for numerical rank detection.

1: Compute the Cholesky decomposition $M = LL^*$. (Use the definition of $M$ and exploit structure. $M$ can also be diagonal. $M$ can also be given implicitly by providing $L$, which is not necessarily triangular. One can take $L = \sqrt{M}$.)
2: Compute the thin SVD $L^*X_m = \hat{U}\Sigma V^*$. \{ $\Sigma = \text{diag}(\sigma_i)_{i=1}^m \equiv \sigma_1 \geq \cdots \geq \sigma_m$ \}
3: Determine the numerical rank $k$ using the threshold $\epsilon$. (See §3.1.1.)

**Output:** $\hat{U}_k \equiv L^{−*}\hat{U}$, where $\hat{U}_k = \hat{U}(.; 1 : k)$. $\hat{U}_k$ can be returned implicitly by $L$ and $\hat{U}_k$.

The computed weighted POD basis $\hat{U}_k$ is $M$–unitary: $\hat{U}_k^*M\hat{U}_k = I_k$, i.e. its columns are orthonormal set in the inner product $(x, y)_M = y^*Mx$. Any computation built upon the result of Algorithm 8.1 (including the Rayleigh quotient, Ritz pairs and error estimates) should be done in the space $(\mathbb{C}^n, (\cdot, \cdot)_M)$, and the errors should be measured in the corresponding norm $\| \cdot \|_M = \sqrt{(\cdot, \cdot)_M} \equiv \|L^* \cdot\|_2$, where $L$ is any square matrix such that $LL^* = M$; for an example see e.g. [77].

Recall that the operator norm induced by the norm $\| \cdot \|_M$ can be computed as $\|A\|_M = \|L^*AL^{−*}\|_2$.

**Remark 8.1.** The optimality property of the low rank approximation $X_m \approx \hat{U}_k\Sigma_k V_k^*$ (where $\Sigma_k = \Sigma(1 : k, 1 : k)$, $V_k = V(1 : 1 : k)$) still holds true, but measured in the induced matrix norm $\|X_m\|_{2,M} = \max_{z \neq 0} \|X_mz\|_M / \|z\|_2 \equiv \|L^*X_m\|_2$. It is easily checked that $\|X_m\|_{2,M} = \sigma_1$, and that the approximation error $\Delta X_{m,k} \equiv X_m - \hat{U}_k\Sigma_k V_k^*$ is equal to $\|X_m - \hat{U}_k\Sigma_k V_k^*\|_{2,M} = \sigma_{k+1}$.

**8.1. New algorithm**

We now work out the details of the data driven Rayleigh–Ritz approximation in the weighted inner product.

**Proposition 8.2.** Assume that we are given the data $X_m$ and $Y_m = AX_m$, and that Algorithm 8.1 is applied to $X_m$. The $(\cdot, \cdot)_M$–weighted Rayleigh quotient matrix of $A$ with respect to the POD basis $\hat{U}_k$ can be computed from the available data as

$$\hat{S}_k = \hat{U}_k^*L^*Y_mV_k\Sigma_k^{-1}. \quad (8.1)$$

If $(\lambda, \omega)$ is an eigenpair of $\hat{S}_k$ $(\hat{S}_k\omega = \lambda\omega, \omega \neq 0)$ then the corresponding Ritz pair $(\lambda, \hat{U}_k\omega)$ of $A$ has the residual norm computable as

$$\|A(\hat{U}_k\omega) - \lambda(\hat{U}_k\omega)\|_M = \|L^*Y_mV_k\Sigma_k^{-1}\omega - \lambda\hat{U}_k\omega\|_2. \quad (8.2)$$
Proof. Recall that the adjoint of $\hat{U}_k$ (considered as mapping between the inner product spaces $(\mathbb{C}^k, (\cdot, \cdot))$ and $(\mathbb{C}^n, (\cdot, \cdot)_M)$) is $\hat{U}_k^* = \hat{U}_k M$, where $\hat{U}_k$ is the usual conjugate transpose. Since $\hat{U}_k^* \hat{U}_k = I_k$, the Rayleigh quotient is computed as

$$\hat{S}_k = \hat{U}_k^* A \hat{U}_k = \hat{U}_k^* L^* A L^{-1} \hat{U}_k. \quad (8.3)$$

Finally, one can easily check that

$$\tilde{S}_k = \hat{U}_k^* L^* Y_m V_k \Sigma_k^{-1} = \hat{U}_k^* L^* A (\hat{U}_k \Sigma_k V_k^* + (X_m - \hat{U}_k \Sigma_k V_k^*) \Sigma_k^{-1} = \hat{U}_k^* L^* A L^{-1} \hat{U}_k \equiv \hat{S}_k,$$

where we have used $\Delta X_m V_k \Sigma_k^{-1} = 0$, and $A \hat{U}_k = \Lambda (X_m - \Delta X_m, k) V_k \Sigma_k^{-1} = \Lambda X_m V_k \Sigma_k^{-1} = Y_m V_k \Sigma_k^{-1}$. This completes the proof of both (8.1) and (8.2).

The resulting weighted version of the DMD algorithm is as follows:

**Algorithm 8.2** $[Z_k, \lambda_k] = $ Weighted DMD($X_m, Y_m; M; \epsilon$)

**Input:**

- $X_m = (x_1, \ldots, x_m), Y_m = (y_1, \ldots, y_m) \in \mathbb{C}^{n \times m}$ that define a sequence of snapshots pairs $(x_i, y_i \equiv A x_i)$. (Toric assumption is that $n$ is large and that $m \ll n$.)

- Hermitian $n \times n$ positive definite $M$ that defines the inner product.

- Tolerance level $\epsilon$ for numerical rank

1. $[\tilde{U}_k, L, [\hat{U}_k]] = $ Weighted POD($X_m; M; \epsilon$) \{Algorithm 8.1\}
2. $\tilde{S}_k = \hat{U}_k^* L^* Y_m V_k \Sigma_k^{-1}$ \{Rayleigh quotient; see Proposition 8.2.\}
3. $[W_k, \Lambda_k] = $ eig($\tilde{S}_k$) \{$\Lambda_k = \text{diag}(\lambda_i)_{i=1}^k; \tilde{S}_k W_k(:, i) = \lambda_i W_k(:, i); ||W_k(:, i)||_2 = 1$\}
4. $Z_k = \hat{U}_k W_k$ \{Ritz vectors\}

**Output:** $Z_k, \Lambda_k$

**Remark 8.3.** In the case of sequentially shifted data, computing $L^* Y_m$ in Line 2. of Algorithm 8.2 and in (8.2) (if residual norms are wanted as well) can use the trailing $m - 1$ columns of $L^* X_m$ computed in Line 2. of Algorithm 8.1. This saves $O(n^2 m)$ floating point operations.

**Remark 8.4.** If, with given $M$, the natural structure is given in the inner product $(x, y)_M^{-1} \equiv y^* M^{-1} x$, then the above computation is modified as follows: (i) In Line 2. of Algorithm 8.1, compute the SVD $L^{-1} X_m = U \Sigma V^*$; (ii) Define $\hat{U}_k$ as $\tilde{U}_k = L \hat{U}_k$; (iii) In Line 2. of Algorithm 8.2 define the Rayleigh quotient as $\tilde{S}_k = \hat{U}_k^* L^{-1} Y_m V_k \Sigma_k^{-1} (\equiv \tilde{U}_k^* L^{-1} A L \hat{U}_k)$; (iv) for a Ritz pair $(\lambda, \tilde{U}_k w)$ compute the residual as $||A (\tilde{U}_k w) - \lambda (\tilde{U}_k w)||_M^{-1} = ||L^{-1} Y_m V_k \Sigma_k^{-1} w - \lambda \tilde{U}_k w||_2$. These changes simply reflect the factorization $M^{-1} = L^* L^{-1}$, which is equivalent to $M = LL^*$ (for square $L$).

**Remark 8.5.** There is another important aspect of allowing more general inner product in the DMD. Think of $A$ as being a discretized unitary \footnote{Note that this modification trivially applies to the Exact DMD \cite[Algorithm 2]{Drmać2018a}.} operator in a Hilbert space with the inner product $(\cdot, \cdot)_M$, i.e. $A$ is $M$-unitary: $A^* M A = M \equiv LL^*$. Recall that the $M$-adjoint of $A$ reads $A^{[k]} M = M^{-1} A^* M$, and that $L^* A L^{-1}$ (appearing in (8.3)) is unitary in $(\cdot, \cdot)$. Hence, the weighted formulation automatically includes such general form of unitarity. Of course, all aspects of the more general setting must be adjusted for detailed error and perturbation analysis. For the sake of brevity, we only illustrate this point by providing below a weighted version of the Bauer–Fike theorem.
**Theorem 8.6.** Let $A = S \text{diag}(\alpha_i)_{i=1}^n S^{-1}$, and let $\lambda$ be an eigenvalue of $A + \delta A$ with some perturbation $\delta A$. Then

$$
\min_{i=1:n} |\alpha_i - \lambda| \leq \sqrt{\mu_2(M)}(\|S\|_M \|S^{-1}\|_M)\|\delta A\|_M, \quad \text{where} \quad \mu_2(M) = \min_{\Delta \in \text{Diag}_n^+} \kappa_2(\Delta M \Delta),
$$

(8.4)

where $\text{Diag}_n^+$ denotes the set of diagonal, positive definite matrices on $\mathbb{C}^n$. If $A$ is $M$-normal, then $S$ is $M$-unitary and $\kappa_M(S) = \|S\|_M \|S^{-1}\|_M = 1$. If $M$ is diagonal, $\mu_2(M) = 1$. Further, if $A$ is nonsingular then

$$
\min_{i=1:n} |\alpha_i - \lambda| \leq \sqrt{\mu_2(M)}\kappa_M(S)\|A^{-1}\delta A\|_M.
$$

(8.5)

**Proof.** As in the classical proof of the Bauer–Fike theorem (see [79, Theorem 1.6]), we conclude that in the nontrivial case ($\lambda$ not an eigenvalue of $A$) the matrix $I_n + (\text{diag}(\alpha_i)_{i=1}^n - \lambda I_n)^{-1}S^{-1}\delta A S$ must be singular. Hence

$$
1 \leq \|I_n + (\text{diag}(\alpha_i)_{i=1}^n - \lambda I_n)^{-1}S^{-1}\delta A S\|_M \leq \|L^*(I_n + (\text{diag}(\alpha_i)_{i=1}^n - \lambda I_n)^{-1}S^{-1}\delta A S\|_M.
$$

Take any diagonal positive definite $\Delta$ and write $\Delta M \Delta = M_s$, $L_s = \Delta L$ (thus $M_s = L_s L_s^*$) and

$$
1 \leq \|L_s\|_2 \|L_s^{-1}\|_2 \frac{1}{\min_{i=1:n} |\alpha_i - \lambda|} \|S^{-1}\delta A S\|_M \leq \frac{\sqrt{\kappa_2(M_s)}}{\min_{i=1:n} |\alpha_i - \lambda|} \|S^{-1}\|_M \|S\|_M \|\delta A\|_M.
$$

Since this holds for an arbitrary $\Delta \in \text{Diag}_n^+$, the condition number of $M$ enters the error bounds as $\mu_2(M)$, which is potentially much smaller than $\kappa_2(M)$. (For instance, if $M$ is diagonal with arbitrarily high $\kappa_2(M)$, the value of $\mu_2(M)$ is one.) The second assertion follows from the first one by virtue of the nice trick from [25, §2].

In the application to eigenvalue error estimates in DMD, the perturbation $\delta A$ is the residual which should be measured using Proposition 8.2.

Let us also mention that in the case of highly weighted data (e.g. strongly graded row norms in $X_m$) it is advised that the SVD and also the QR factorization (such as e.g. in Algorithm 6.1) are computed more carefully, using row pivoting scheme, see [80], [43], [44].

### 8.2. More general weighting schemes

In Line 2 of Algorithm 8.1, $X_m = \tilde{U} \Sigma \tilde{V}^*$ is an SVD of $X_m$, where $\tilde{U} \equiv L^{-*} \bar{U}$ is $M$-unitary and $\tilde{V}$ is unitary (in the canonical Euclidean inner product). We can think of situations when one would want to weight the snapshots throughout time steps as well. For instance, in a POD analysis of flames in the internal combustion engine [81], the weighting is determined as a function of in-cylinder pressure and used to define weighted average for particular crank angle. As another example, we recall §3, where we used columns scaling of the data (snapshots) as numerical device to curb ill-conditioning.

To define a general setting, in addition to the $M$-induced structure discussed in §8.1, equip $\mathbb{C}^m$ with a weighted inner product $(\cdot, \cdot)_N$ generated by a positive definite matrix $N = KK^*$, or by its inverse $N^{-1}$. (For instance, $N$ could be diagonal with forgetting factors that put less weight on older snapshots, or to represent some other changes in time dynamics.)

With the two norms induced by $M$ and $N$, most appropriate is the weighted SVD, introduced by van Loan [14] and successfully deployed in [82] for solving various weighted least squares problems. In this framework, Line 2 of Algorithm 8.1 should be changed to computing the SVD

$$
L^* X_m K^{-*} = \bar{U} \Sigma \bar{V}^*, \quad \Sigma = \text{diag}(\sigma_i)_{i=1}^m, \quad \sigma_1 \geq \cdots \geq \sigma_m; \quad \bar{U}^* \bar{U} = \bar{V}^* \bar{V} = I_m.
$$

(8.6)
which yields the weighted SVD $X_m = \hat{U}\Sigma\hat{V}^{-1}$ with $M$-unitary $\hat{U} = L^{-*}\tilde{U}$ and $N$-unitary $\hat{V} = K^{-*}\tilde{V}$. Recall that $\|X_m\|_{N,M} = \max_{z \neq 0} \|X_mz\|_M/\|z\|_N \equiv \|L^*X_mK^{-*}\|_2 = \sigma_1$. The $\|\cdot\|_{N,M}$ error of the best rank $k$ approximation $\hat{U}_k\Sigma_k\hat{V}_k^{-1}$ is $\sigma_{k+1}$. We can easily check that the Rayleigh quotient and the residual norm of a Ritz pair $(\lambda, \hat{U}_kw)$ read, respectively,

$$\tilde{S}_k = \hat{U}_k^*L^*Y_mK^{-*}\tilde{V}_k^{-1}, \quad \|A(\hat{U}_kw) - \lambda(\hat{U}_kw)\|_M = \|L^*Y_mK^{-*}\tilde{V}_k^{-1}w - \lambda\tilde{U}_kw\|_2.$$  

(8.7)

It is obvious how to adapt the above formulas if the inner products are generated by $M^{-1}$ and $N^{-1}$. Also, the refinement procedure from §3.3 applies here as well, with the difference that the norm is different and thus the minimal singular value of a product of three matrices is needed. The compressed DMD from §6.1 can be adapted to this general setting with necessary changes; e.g. the QR factorization (6.1) should be computed using $M$-unitary $Q_f$.

In all cases discussed above, we need the SVD of the product of three matrices that can be computed as in [82], [83], and [84].

9. Concluding remarks

We have presented modifications of the DMD algorithm, together with theoretical analysis and justification, discussion of the potential weaknesses of the original method, and examples that illustrate the advantages of the new proposed method. From the point of view of numerical linear algebra, the deployed techniques are not new; however, the novelty is in adapting them to the data driven setting and turning the DMD into a more powerful tool. Also, we provide the fine and sometimes mission critical details of numerical software development.

Moreover, we have defined a more general framework for the DMD, allowing formulations in weighted spaces, i.e. in elliptic norms defined by positive definite matrices. We hope that these modifications and new ideas will trigger further development of data driven methods for spectral analysis.

Given the importance of DMD in computational analysis of various phenomena in applied sciences and engineering, we believe that our work will also facilitate more advanced and robust computational studies of dynamical systems, in particular in computational fluid mechanics. For first encouraging applications of the new method see [85], [86].

Acknowledgments

This research is supported by the DARPA Contract HR0011-16-C-0116 “On a Data-Driven, Operator-Theoretic Framework for Space-Time Analysis of Process Dynamics” and the ARL Contract W911NF-16-2-0160 “Numerical Algorithms for Koopman Mode Decomposition and Application in Channel Flow”.

References

[1] P. J. Schmid and J Sesterhenn. “Dynamic Mode Decomposition of Numerical and Experimental Data”. In: Sixty-First Annual Meeting of the APS Division of Fluid Dynamics. San Antonio, Texas, USA, 2008.

[2] P. J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: Journal of Fluid Mechanics 656 (Aug. 2010), pp. 5–28.
[3] C. W. Rowley et al. “Spectral analysis of nonlinear flows”. In: *Journal of Fluid Mechanics* 641 (2009), pp. 115–127.

[4] K. K. Chen, J. H. Tu, and C. W. Rowley. “Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses”. In: *Journal of Nonlinear Science* 22.6 (Apr. 2012), pp. 29–915.

[5] J. H. Tu et al. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421.

[6] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. “A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition”. In: *Journal of Nonlinear Science* 25.6 (June 2015), pp. 1307–1346.

[7] M. S. Hemati, M. O. Williams, and C. W. Rowley. “Dynamic mode decomposition for large and streaming datasets”. In: *Physics of Fluids* 26.11 (Nov. 2014), p. 111701.

[8] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. “Low-rank and sparse dynamic mode decomposition”. In: *Center for Turbulence Research, Annual Research Briefs* (2012), pp. 139–152.

[9] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. “Sparsity-promoting dynamic mode decomposition”. In: *Physics of Fluids* 26.2 (Feb. 2014), p. 024103.

[10] I. Mezić. “Analysis of fluid flows via spectral properties of the Koopman operator”. In: *Annual Review of Fluid Mechanics* 45 (2013), pp. 357–378.

[11] D Giannakis, J Slawinska, and Z Zhao. “Spatiotemporal Feature Extraction with Data-Driven Koopman Operators”. In: *1st International Workshop Feature extraction Modern Questions and Challenges* (2015).

[12] J. H. Tu et al. “On Dynamic Mode Decomposition: Theory and Applications”. In: *ArXiv e-prints* (Nov. 2013). arXiv: 1312.0041 [math.NA].

[13] S. T. M. Dawson et al. “Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition”. In: *Experiments in Fluids* 57.3 (2016), p. 42. ISSN: 1432-1114. DOI: 10.1007/s00348-016-2127-7.

[14] C. F. V. Loan. “Generalizing the Singular Value Decomposition”. In: *SIAM Journal on Numerical Analysis* 13.1 (1976), pp. 76–83. DOI: 10.1137/0713009. eprint: http://dx.doi.org/10.1137/0713009.

[15] P. J. Schmid et al. “Applications of the dynamic mode decomposition”. In: *Theoretical and Computational Fluid Dynamics* 25.1 (2011), pp. 249–259. ISSN: 1432-2250. DOI: 10.1007/s00162-010-0203-9.

[16] I. Mezić and A. Banaszuk. “Comparison of systems with complex behavior”. In: *Physica D: Nonlinear Phenomena* 197.1 (2004), pp. 101–133.

[17] I. Mezić. “Spectral properties of dynamical systems, model reduction and decompositions”. In: *Nonlinear Dynamics* 41.1 (2005), pp. 309–325.

[18] I. Mezić. “On the Nature of Approximations of the Koopman Operator”. In: *AIMdyn Technical Reports* 201708.003v1 (Aug. 2017). [DARPA HR0011-16-C-0116], pp. 1–26.

[19] P. J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (Aug. 2010), pp. 5–28. DOI: 10.1017/S0022112010001217.

[20] G. W. Stewart. *Matrix Algorithms: Volume II: Eigensystems*. SIAM, 2001.
[21] D. Watkins. The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods. SIAM, 2007.
[22] J. Liesen and Z. Strakos. Krylov Subspace Methods. Oxford Science Publications, 2012.
[23] J. Erxiong. “Bounds for the smallest singular value of a Jordan block with an application to
eigenvalue perturbation”. In: Linear Algebra and its Applications 197 (1994), pp. 691–707. ISSN: 0024-3795. DOI: http://dx.doi.org/10.1016/0024-3795(94)90510-X.
[24] F. L. Bauer and C. T. Fike. “Norms and exclusion theorems”. In: Numerische Mathematik 2.1 (1960), pp. 137–141. ISSN: 0945-3245. DOI: 10.1007/BF01386217.
[25] S. C. Eisenstat and I. C. F. Ipsen. “Three Absolute Perturbation Bounds for Matrix Eigenvalues
Imply Relative Bounds”. In: SIAM Journal on Matrix Analysis and Applications 20.1 (1998), pp. 149–158. DOI: 10.1137/S0895479897323282. eprint: http://dx.doi.org/10.1137/S0895479897323282.
[26] C. Eckart and G. Young. “The approximation of one matrix by another of lower rank”. In: Psychometrika 1.3 (1936), pp. 211–218. ISSN: 1860-0980. DOI: 10.1007/BF02288367.
[27] L. Mirsky. “Symmetric Gauge Functions and Unitarily Invariant Norms”. In: The Quarterly
Journal of Mathematics 11.1 (1960), p. 50. DOI: 10.1093/qmath/11.1.50.
[28] K. Pearson. “On Lines and Planes of Closest Fit to Systems of Points in Space”. In: Philosophical Magazine 2.11 (1901), pp. 559–572.
[29] G. H. Golub, V. C. Klema, and G. W. Stewart. Rank Degeneracy and Least Squares Problems.
Tech. rep. CS-TR-76-559. Stanford, CA, USA, 1976.
[30] B. P. Epps and A. H. Techet. “An error threshold criterion for singular value decomposition
modes extracted from PIV data”. In: Experiments in Fluids 48.2 (2010), pp. 355–367. ISSN: 1432-1114. DOI: 10.1007/s00348-009-0740-4.
[31] J. N. Kutz et al. Dynamic Mode Decomposition. SIAM, 2016.
[32] M. R. Jovanović, P. J. Schmid, and J. W. Nichols. “Low–rank and sparse dynamic mode
decomposition”. In: Annual Research Briefs 2012. Center for Turbulence Research, Stanford
University, 2012, pp. 139–152.
[33] G. W. Stewart. “A Krylov–Schur Algorithm for Large Eigenproblems”. In: SIAM J. Matrix
Anal. Appl. 23.3 (2001), pp. 601–614.
[34] R. A. Horn and C. R. Johnson. Topics in Matrix Analysis. Cambridge University Press, 1991.
[35] Z. Jia. “Refined iterative algorithms based on Arnoldi’s process for large unsymmetric eigen-
problems”. In: Linear Algebra and its Applications 259 (1997), pp. 1–23. ISSN: 0024-3795. DOI: http://dx.doi.org/10.1016/S0024-3795(96)00238-8.
[36] Z. Jia. “Polynomial characterizations of the approximate eigenvectors by the refined Arnoldi
method and an implicitly restarted refined Arnoldi algorithm”. In: Linear Algebra and its
Applications 287.1–3 (1999), pp. 191–214. ISSN: 0024-3795. DOI: http://dx.doi.org/10.1016/S0024-3795(98)10197-0.
[37] Z. Jia. “Residuals of refined projection methods for large matrix eigenproblems”. In: Computers & Mathematics with Applications 41.7 (2001), pp. 813–820. ISSN: 0898-1221. DOI: http://dx.doi.org/10.1016/S0898-1221(00)00321-7.
[38] Z. Jia and G. W. Stewart. “An Analysis of the Rayleigh–Ritz Method for Approximating
Eigenspaces”. In: Math. Comput. 70.234 (Apr. 2001), pp. 637–647. ISSN: 0025-5718. DOI: 10.1090/S0025-5718-00-01208-4.
REFERENCES

[39] Z. Jia. “Some theoretical comparisons of refined Ritz vectors and Ritz vectors”. In: Science in China, Ser. A Mathematics 47 (2004), pp. 222–233.

[40] J. H. Tu and C. W. Rowley. “An improved algorithm for balanced {POD} through an analytic treatment of impulse response tails”. In: Journal of Computational Physics 231.16 (2012), pp. 5317–5333. ISSN: 0021-9991. DOI: http://dx.doi.org/10.1016/j.jcp.2012.04.023.

[41] A. van der Sluis. “Condition Numbers and Equilibration of Matrices”. In: Numerische Mathematik 14 (1969), pp. 14–23.

[42] R. Mohr and I. Mezić. “Construction of eigenfunctions for scalar-type operators via Laplace averages with connections to the Koopman operator”. In: arXiv eprints (Mar. 2014). arXiv: 1403.6559 [math.SP].

[43] Z. Drmač and K. Veselić. “New fast and accurate Jacobi SVD algorithm: I.” In: SIAM J. Matrix Anal. Appl. 29.4 (2008), pp. 1322–1342.

[44] Z. Drmač and K. Veselić. “New fast and accurate Jacobi SVD algorithm: II.” In: SIAM J. Matrix Anal. Appl. 29.4 (2008), pp. 1343–1362.

[45] Z. Drmač. “Algorithm 977: A QR–preconditioned QR SVD method for computing the SVD with high accuracy”. In: ACM Trans. Math. Softw. 44.1 (July 2017), 11:1–11:30. DOI: 10.1145/3061709.

[46] S. Bagheri. “Koopman-mode decomposition of the cylinder wake”. In: Journal of Fluid Mechanics 726 (2013), pp. 596–623. DOI: 10.1017/jfm.2013.249.

[47] B. Glaz et al. “Quasi-Periodic Intermittency in Oscillating Cylinder Flow”. In: arXiv preprint arXiv:1609.06267 (2016).

[48] S. Ivč et al. “Robustness of Koopman modes for cylinder wake”. AIMDyn Tech Report. May 2017.

[49] H. Jasak, A. Jemcov, and v. Tuković. “OpenFOAM: A C++ library for complex physics simulations”. In: International workshop on coupled methods in numerical dynamics. Vol. 1000. IUC Dubrovnik, Croatia. 2007, pp. 1–20.

[50] M. Korda and I. Mezić. On Convergence of Extended Dynamic Mode Decomposition to the Koopman Operator. Apr. 2017. arXiv: 1703.04680 [math.OC].

[51] T. Sayadi and P. J. Schmid. “Parallel QR algorithm for data-driven decompositions”. In: Proceedings of the Summer Program 2014. Center for Turbulence Research, Stanford University. 2014, pp. 335–343.

[52] T. Sayadi and P. J. Schmid. “Parallel data-driven decomposition algorithm for large-scale datasets: with application to transitional boundary layers”. In: Theor. Comput. Fluid Dyn. 30 (2016), pp. 415–428.

[53] B. A. Belson, J. H. Tu, and C. W. Rowley. “Algorithm 945: Modred&mdash;A Parallelized Model Reduction Library”. In: ACM Trans. Math. Softw. 40.4 (July 2014), 30:1–30:23. ISSN: 0098-3500. DOI: 10.1145/2616912.

[54] T. F. Chan. “An improved algorithm for computing the singular value decomposition”. In: ACM Trans. Math. Soft. 8 (1982), pp. 72–83.

[55] E. Anderson et al. LAPACK Users’ Guide (Third Ed.) Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8.
[56] J. Demmel et al. “Communication-optimal Parallel and Sequential QR and LU Factorizations”. In: *SIAM Journal on Scientific Computing* 34.1 (2012), A206–A239. DOI: 10.1137/080731992. eprint: http://dx.doi.org/10.1137/080731992.

[57] S. Hammarling and C. Lucas. *Updating th QR factorization and the least squares problem*. Tech. rep. MIMS EPrint: 2008.111. University of Manchester, 2008.

[58] R. Andrew and N. Dingle. “Implementing QR factorization updating algorithms on GPUs”. In: *Parallel Computing* 40.7 (2014). 7th Workshop on Parallel Matrix Algorithms and Applications, pp. 161 –172. ISSN: 0167-8191. DOI: http://dx.doi.org/10.1016/j.parco.2014.03.003.

[59] N. J. Higham. “Computing real square roots of a real matrix”. In: *Linear Algebra and its Applications* 88 (1987), pp. 405 –430. ISSN: 0024-3795. DOI: http://dx.doi.org/10.1016/0024-3795(87)90118-2.

[60] N. J. Higham and L. Lin. “A Schur–Padé Algorithm for Fractional Powers of a Matrix”. In: *SIAM Journal on Matrix Analysis and Applications* 32.3 (2011), pp. 1056–1078. DOI: 10.1137/10081232X. eprint: http://dx.doi.org/10.1137/10081232X.

[61] N. J. Higham. *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008, pp. xx+425. ISBN: 978-0-898716-46-7.

[62] W. Kahan. “Branch cuts for complex elementary functions; or, Much ado about nothing’s sign bit”. In: *The state of the art in numerical analysis*. Ed. by A. Iserles and M. Powell. Clarendon Press, 1987, pp. 165–211.

[63] F. Richecoeur et al. “DMD algorithms for experimental data processing in combustion”. In: *Proceedings of the Summer Program 2012*. Center for Turbulence Research, Stanford University, 2012, pp. 459–468.

[64] S. C. Reddy, P. J. Schmid, and D. S. Henningson. “Pseudospectra of the Orr–Sommerfeld Operator”. In: *SIAM Journal on Applied Mathematics* 53.1 (1993), pp. 15–47. DOI: 10.1137/0153002. eprint: http://dx.doi.org/10.1137/0153002.

[65] I. Kalashnikova et al. *Reduced order modeling for prediction and control of large–scale systems*. Sandia Report SAND2014–4693. Sandia National Laboratories, 2014.

[66] G. Serre et al. “Reliable reduced-order models for time-dependent linearized Euler equations”. In: *J. Comput. Phys.* 231.15 (2012), pp. 5176–5194.

[67] C. W. Rowley. “Model Reduction for Fluids, Using Balanced Proper Orthogonal Decomposition”. In: *Internat. J. Bifur. Chaos Appl. Sci. Engrg.* 15.3 (2005), pp. 997–1013.

[68] J. B. Freund and T. Colonius. “POD Analysis of Sound Generation by a Turbulent Jet”. In: *40th AIAA Aerospace Sciences Meeting*. 2002.

[69] M. Tabandeh and M. Wei. “On the Symmetrization in POD-Galerkin Model for Linearized Compressible Flows”. In: *54th AIAA Aerospace Sciences Meeting*. 2016.

[70] I. Kalashnikova and S. Arunajatesan. “A stable Galerkin reduced order modeling (ROM) for compressible flow”. In: *10th Wold Congress on Computational Mechanics*. Blucher Mechanical Engineering Proceedings, 2014.

[71] M. Grepl et al. “Efficient reduced–basis treatment of nonaffine and nonlinear partial differential equations”. In: *ESAIM, Math. Model. Numer. Anal.* 41.3 (2007), pp. 575–605.

[72] M. Barrault et al. “An Empirical Interpolation Method: Application to Efficient Reduced-Basis Discretization of Partial Differential Equations”. In: *C. R. Acad. Sci. Paris, Série I*. 339 (2004), pp. 667–672.
REFERENCES

[73] Y. Maday et al. “A general multipurpose interpolation procedure: the magic points”. In: *Communications on Pure and Applied Analysis* 8.1 (2009), pp. 383–404. ISSN: 1534-0392. DOI: 10.3934/cpaa.2009.8.383.

[74] S. Chaturantabut and D. C. Sorensen. “Nonlinear model reduction via discrete empirical interpolation”. In: *SIAM J. Sci. Comput.* 32.5 (2010), pp. 2737–2764.

[75] Z. Drmač and S. Gugercin. “A New Selection Operator for the Discrete Empirical Interpolation Method – improved a priori error bound and extensions”. In: *SIAM J. Sci. Comput.* 38.2 (2016), A631–A648.

[76] Y. Maday et al. “The Generalized Empirical Interpolation Method: Stability theory on Hilbert spaces with an application to the Stokes equation”. In: *Comput. Methods Appl. Mech. and Engrg.* 287 (2015), pp. 310–334.

[77] Z. Drmač and A. K. Saibaba. “The Discrete Empirical Interpolation Method: Canonical Structure and Formulation in Weighted Inner Product Spaces”. In: *ArXiv e-prints* (Apr. 2017). arXiv: 1704.06606 [math.NA].

[78] S. Volkwein. “Model reduction using proper orthogonal decomposition”. In: *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz.* (2011).

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

[80] 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*. North Holland, Amsterdam, 1969, pp. 122–126.

[81] B. K. et al. “Application of proper orthogonal decomposition to the reconstruction of space and time resolved images of flames in internal combustion engines”. In: *XXX Meeting on Combustion of the Italian Section of The Combustion Institute, Ischia (NA).* 2007.

[82] L. M. Ewerbring and F. T. Luk. “Canonical correlations and generalized SVD: applications and new algorithms”. In: *Journal of Computational and Applied Mathematics* 27 (1989), pp. 37–52.

[83] A. W. Bojanczyk et al. “An accurate product SVD algorithm”. In: *Signal Processing* 25 (1991), pp. 189–201.

[84] Z. Drmač. “New Accurate Algorithms for Singular Value Decomposition of Matrix Triplets”. In: *SIAM Journal on Matrix Analysis and Applications* 21.3 (2000), pp. 1026–1050. DOI: 10.1137/S0895479897321209. eprint: http://dx.doi.org/10.1137/S0895479897321209.

[85] N. Črnjarič Žičić, S. Maćešić, and I. Mezić. “Koopman operator spectrum for random dynamical systems”. In: *AIMdyn Technical Reports* 201708.002v1 (Aug. 2017). [DARPA HR0011-16-C-0116], pp. 1–32.

[86] B. Crnković and I. Mezić. “Dynamic Mode Decomposition for Non-autonomous Systems”. In: *AIMdyn Technical Reports* 201708.001v1 (Aug. 2017). [DARPA HR0011-16-C-0116], pp. 1–36.