Dynamic Mode Decomposition for Wave Propagation

Fabio Matti

April 30, 2024

Abstract

Dynamic mode decomposition (DMD) is a computational method for extracting the relevant components in the evolution of a non-linear dynamical system. In this project we apply DMD to wave-like phenomena varying from the propagation of electromagnetic waves in non-linear media to traction tests for the study of material property. We propose multiple implementation techniques for these systems which help the DMD algorithms DMD perform better and at scale. Among other things, we exploit the Khatri-Rao product structure of the observations to develop a tensor-based DMD and propose a more stable and efficient approach to updating DMDs from a data stream. Further, we investigate the suitability of DMD for observations stemming from problems governed by the wave equation, and make a conclusive recommendation for the suitability of DMD in this scenario.

1 Introduction

The Koopman formalism introduces an infinite dimensional operator which linearlizes the time evolution of a non-linear dynamical system [8]. Making use of an invariant subspace of this linear operator alows us to express the dynamics in a compact formula whose constituents can be further analyzed to identify dominant structures and reduce the order of the model. Even when the state of a system evolves non-linearly, we can construct observations in such a way that they – at least locally – approximately exhibit a linear evolution. To extract the dominant components making up these observations, we can use a wide variety of tools, which were originally applied to fluid flows, but can be applied in more general settings. These methods are often termed dynamic mode decomposition (DMD) [14, 2, 3, 15].

The goal of this project is to adapt the DMD to the numerical solution of certain time-dependent partial differential equations governing wave type phenomena and to investigate their suitability for this particular family of problems. The prototypical example of such problems is the wave equation. Let $\Omega \subset \mathbb{R}^d$

be an open connected domain with Lipschitz boundary and let $I = [0, T]$ be the time domain with $T > 0$ denoting the final time. We look for $u : \Omega \times [0, T] \to \mathbb{R}$ such that

$$
\begin{cases}
\ddot{u} - \nabla \cdot (c^2 \nabla u) = f & \text{in } \Omega \times (0, T], \\
u = g_D & \text{on } \partial\Omega_D \times (0, T], \\
c^2 \nabla u \cdot n = g_N & \text{on } \partial\Omega_N \times (0, T], \\
u = u_0 & \text{in } \Omega, \\
\dot{u} = v_0 & \text{in } \Omega,
\end{cases}
\tag{1.1}
$$

where u_0 and v_0 are two initial conditions, g_D and g_N are Dirichlet and Neumann boundary conditions, respectively, and $\partial\Omega_D$ and $\partial\Omega_N$ form a disjoint partition of the boundary (i.e. $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$). The positive valued function c represents the wave velocity. Similar looking PDEs arise in structural dynamics, where the unknown is a vector valued displacement field and the differential operator is the divergence of a stress tensor. A Galerkin discretization of the spatial variables leads to solving the semi-discrete problem

$$
\begin{cases}
\underline{\mathbf{M}}\ddot{\mathbf{u}}(t) + \underline{\mathbf{S}}\mathbf{u}(t) = \mathbf{f}(t) \quad \text{for } t \in [0, T] \\
\mathbf{u}(0) = \mathbf{u}_0 \\
\dot{\mathbf{u}}(0) = \boldsymbol{v}_0
\end{cases}
\tag{1.2}
$$

where the stiffness matrix $\underline{\mathbf{S}} \in \mathbb{R}^{n \times n}$ is symmetric and the mass matrix $\underline{\mathbf{M}} \in \mathbb{R}^{n \times n}$ is symmetric positive definite and $\mathbf{u}(t) \in \mathbb{R}^n$ is the solution vector which contains the approximation of the solution in a finite dimensional subspace [11]. Our task is to determine a low-dimensional representation of the solutions $\mathbf{u}_i = \mathbf{u}(t_i)$ which result from solving (1.2) for discrete and uniformly spaced time steps $t_1, t_2, \ldots, t_m$.

1.1 Related work

The DMD for dimensionality reduction of observations derived from a non-linear fluid flows has already been popularized by [14]. Since then, a number of improvements to the standard DMD procedure have been proposed On one hand, refining the DMD eigenvalues and modes to even better reflect the spectral properties of the linear relation between the observations was proposed [2]. On the other, an alternative and more robust algorithm for reconstructing the snapshots from a set of DMD eigenvalues and modes was developed [3].

The DMD has been successfully applied to a wide set of problems outside the domain of fluid dynamics. Partial differential equations of a similar nature to our problem have already been studied in multiple cases [12]. Also, related problems in structural mechanics have already been discussed in [16]. To our knowledge, we are the first to exploit the approximate Khatri-Rao structure for these types of problems.

1.2 Notation

We try to keep the notation as consistent throughout the report as possible. In particular,

- we exclusively work with objects over the real numbers $\mathbb{R}$ or the complex numbers $\mathbb{C}$ and non-zero integers $\mathbb{N}$;
- scalar are represented by lower case Greek or Latin letters (s, ε, ...), vectors are additionally printed in bold ($\boldsymbol{\nu}$, $\boldsymbol{\mu}$, ...), matrices are additionally underlined ($\underline{\mathbf{A}}$, $\underline{\boldsymbol{\Omega}}$, ...), and tensors are printed caligraphically ($\mathcal{A}$, $\mathcal{Y}$, ...);
- the components of a vector $\boldsymbol{\nu} \in \mathbb{C}^n$ are $\nu_i \in \mathbb{C}, i = 1, \ldots, n$, and the elements of a matrix $\underline{\mathbf{A}} \in \mathbb{C}^{n \times n}$ are $a_{ij} \in \mathbb{C}, i, j = 1, \ldots, n$;
- the diagonal of a matrix $\underline{\mathbf{A}} \in \mathbb{C}^{n \times n}$ is the vector $\operatorname{diag}(\underline{\mathbf{A}}) = [a_{11}, a_{22}, \ldots, a_{nn}]^\top$ while for a vector $\boldsymbol{\nu} \in \mathbb{C}^n$ the matrix $\operatorname{diag}(\boldsymbol{\nu})$ is a diagonal matrix with the components of $\boldsymbol{\nu}$ on its diagonal. Similarly, the diagonal of a d-th order tensor $\mathcal{A} \in \mathbb{C}^{n \times n \times \cdots \times n}$ is $\operatorname{diag}(\mathcal{A}) = [a_{11\ldots1}, a_{22\ldots2}, \ldots, a_{nn\ldots n}]^\top$;
- the identity matrix $\underline{\mathbf{I}}_n = \operatorname{diag}(1, \ldots, 1) \in \mathbb{C}^{n \times n}$ carries ones on its diagonal and zeros everywhere else. The zero matrix $\underline{\mathbf{0}}_{n \times m} \in \mathbb{C}^{n \times m}$ consists of only zero entries;
- the eigenvalues of a square matrix $\underline{\mathbf{A}} \in \mathbb{C}^{n \times n}$ are all scalars λ which, together with some non-zero vectors $\boldsymbol{\nu}$, satisfy the condition $\underline{\mathbf{A}}\boldsymbol{\nu} = \lambda\boldsymbol{\nu}$. We denote them as $\lambda_1 \geqslant \cdots \geqslant \lambda_n$;
- the transpose of a matrix $\underline{\mathbf{A}}$ is denoted with $\underline{\mathbf{A}}^\top$, while the Hermitian conjugate is defined as $\underline{\mathbf{A}}^* = \overline{\underline{\mathbf{A}}}^\top$;
- all matrices $\underline{\mathbf{A}} \in \mathbb{C}^{n \times m}$ with $n \geqslant m$ allow a spectral decomposition $\underline{\mathbf{A}} = \underline{\mathbf{U}}\underline{\boldsymbol{\Sigma}}\underline{\mathbf{V}}^*$ with $\underline{\boldsymbol{\Sigma}} = [\operatorname{diag}(\sigma_1, \ldots, \sigma_n), \underline{\mathbf{0}}_{n \times (n-m)}] \in \mathbb{C}^{n \times m}$, $\underline{\mathbf{U}} \in \mathbb{C}^{n \times n}$ such that $\underline{\mathbf{U}}^*\underline{\mathbf{U}} = \underline{\mathbf{I}}_n$, and $\underline{\mathbf{V}} \in \mathbb{C}^{m \times m}$ such that $\underline{\mathbf{V}}^*\underline{\mathbf{V}} = \underline{\mathbf{I}}_m$. The computation of this decomposition is denoted with $[\underline{\mathbf{U}}, \underline{\boldsymbol{\Sigma}}, \underline{\mathbf{V}}] = \texttt{svd}(\underline{\mathbf{A}})$;
- all matrices $\underline{\mathbf{A}} \in \mathbb{C}^{n \times m}$ with $n \geqslant m$ allow a factorization $\underline{\mathbf{A}} = \underline{\mathbf{Q}}\underline{\mathbf{R}}$ $\underline{\mathbf{Q}} \in \mathbb{C}^{n \times m}$ such that $\underline{\mathbf{Q}}^*\underline{\mathbf{Q}} = \underline{\mathbf{I}}_n$, and upper triangular matrix $\underline{\mathbf{R}} \in \mathbb{C}^{m \times m}$. The computation of this decomposition is denoted with $[\underline{\mathbf{Q}}, \underline{\mathbf{R}}] = \texttt{qr}(\underline{\mathbf{A}})$;
- most rectangular matrices we work with allow a spectral decomposition $\underline{\mathbf{A}} = \underline{\mathbf{W}}\underline{\boldsymbol{\Lambda}}\underline{\mathbf{W}}^*$ with $\underline{\boldsymbol{\Lambda}} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n \times n}$ and $\underline{\mathbf{W}} \in \mathbb{C}^{n \times n}$ such that $\underline{\mathbf{W}}^*\underline{\mathbf{W}} = \underline{\mathbf{I}}_n$. The computation of this decomposition is denoted with $[\underline{\mathbf{W}}, \underline{\boldsymbol{\Lambda}}] = \texttt{eig}(\underline{\mathbf{A}})$;
- norms are denoted with $\|\cdot\|$. In particular, the 2-norm of a vector $\mathbf{x} \in \mathbb{C}^n$ is defined as $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$;

- the pseudoinverse of a matrix is denoted with $\underline{\mathbf{A}}^\dagger$. It satisfies $\underline{\mathbf{A}}^\dagger = (\underline{\mathbf{A}}^\top \underline{\mathbf{A}})^\dagger \underline{\mathbf{A}}^\top$, and if $\underline{\mathbf{A}}$ has linearly independent columns, it acts as a left inverse in the sense that $\underline{\mathbf{A}}^\dagger \underline{\mathbf{A}} = \underline{\mathbf{I}}_n$;
- the Kronecker product of two matrices $\underline{\mathbf{A}} \in \mathbb{C}^{n\times m}$ and $\underline{\mathbf{B}} \in \mathbb{C}^{p\times q}$ is defined as

$$\underline{\mathbf{A}} \otimes \underline{\mathbf{B}} = \begin{bmatrix} a_{11}\underline{\mathbf{B}} & \dots & a_{1m}\underline{\mathbf{B}} \\ \vdots & & \vdots \\ a_{n1}\underline{\mathbf{B}} & \dots & a_{nm}\underline{\mathbf{B}}, \end{bmatrix}; \tag{1.3}$$

- the Khatri-Rao product of $\underline{\mathbf{A}} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m] \in \mathbb{C}^{n\times m}$ with $\underline{\mathbf{B}} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m] \in \mathbb{C}^{p\times m}$ is defined as $\underline{\mathbf{A}} \odot \underline{\mathbf{B}} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \dots, \mathbf{a}_m \otimes \mathbf{b}_m]$
- the Hadamard product of of $\underline{\mathbf{A}}, \underline{\mathbf{B}} \in \mathbb{C}^{n\times m}$ is $(\underline{\mathbf{A}} * \underline{\mathbf{B}})_{ij} = a_{ij} b_{ij}$;
- the outer product of d vectors $\mathbf{a}_i \in \mathbb{C}^{n_i}$ is the $n_1 \times n_2 \times \cdots \times n_d$ tensor $\mathcal{A}$ with elements $a_{i_1 i_2 \dots i_d} = (\mathbf{a}_1)_{i_1} (\mathbf{a}_2)_{i_2} \cdots (\mathbf{a}_d)_{i_d}$;
- the inner product of two d-th order tensors the $\mathcal{X}, \mathcal{Y} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$ is

$$\langle \mathcal{X}, \mathcal{Y} \rangle \sum_{i_1}^{n_1} \sum_{i_2}^{n_2} \cdots \sum_{i_d}^{n_d} x_{i_1 i_2 \dots i_d} y_{i_1 i_2 \dots i_d}; \tag{1.4}$$

- the canonical polyadic (CP) format of a tensor $\mathcal{X} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$ is denoted with

$$\mathcal{X} = [\![\lambda; \underline{\mathbf{A}}_1, \dots, \underline{\mathbf{A}}_d]\!] = \sum_{j=1}^{r} \lambda_j (\mathbf{a}_j^{(1)} \circ \mathbf{a}_j^{(2)} \circ \cdots \circ \mathbf{a}_j^{(d)}) \tag{1.5}$$

  where $\underline{\mathbf{A}}_i = [\mathbf{a}_1^{(i)}, \mathbf{a}_2^{(i)} \ \dots \ \mathbf{a}_r^{(i)}] \in \mathbb{C}^{n_i \times r}$ and $\lambda \in \mathbb{C}^r$. Numerically approximating a d-th order tensor with the CP-rank r is denoted with $[\underline{\mathbf{A}}_1, \underline{\mathbf{A}}_2, \dots, \underline{\mathbf{A}}_d] = \mathtt{cp}(\mathcal{X}, r)$;
- The μ-mode multiplication of a tensor $\mathcal{X} \in \mathbb{C}^{n_1 \times n_2 \times dots \times n_d}$ with a matrix $\underline{\mathbf{A}} \in \mathbb{R}^{n_\mu \times m}$ is a $n_1 \times n_2 \times \cdots \times n_{\mu-1} \times m \times n_{\mu+1} \times \cdots \times n_d$ tensor defined as

$$(\mathcal{X} \circ_\mu \underline{\mathbf{A}})_{i_1 i_2 \dots i_d} = \sum_{k=1}^{n_\mu} x_{i_1 i_2 \dots i_{\mu-1} k i_{\mu+1} \dots i_d} a_{k i_\mu}; \tag{1.6}$$

- the vectorization of a matrix $\underline{\mathbf{A}} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m] \in \mathbb{C}^{n\times m}$ is the vector of columns

$$\mathrm{vec}(\underline{\mathbf{A}}) = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_m \end{bmatrix} \in \mathbb{C}^{nm}. \tag{1.7}$$

Similarly, the vetorization of a tensor is defined in reverse lexicographical ordering;

- a matrix $\underline{\mathbf{A}} \in \mathbb{C}^{n \times n}$ is said to be positive semi-definite if its eigenvalues are all non-negative;
- closed interval are written as $[a, b] \subset \mathbb{R}$ and open intervals as $(a, b) \subset \mathbb{R}$;
- and finally, we use $\mathcal{O}$ to describe the asymptotic lower and upper bounds on the complexity of a computation.

1.3 Organization of the report

This paragraph concludes the introduction. We start the next section by giving a brief summary of the Koopman formalism and how the DMD comes to use in it. We explain the intuition behind multiple DMD algorithms. One of which we extend to the streaming framework with a more robust method than what is conventionally used. Section 2 is then concluded with a summary of some ways the observations can be reconstructed from a DMD. In section 3 we fine tune the DMD algorithms to the specific problem we study. In particular, we exploit the approximate Khatri-Rao structure of the observations to achieve a more favorable scaling in higher spatial dimensions and overall improved algorithmic complexities. Section 4 contains the numerical experiments which we have run. The report is then concluded in a short discussion.

2 From the Koopman operator to dynamic mode decomposition

We consider dynamical systems which undergo a temporal evolution of the form

$$x_{i+1} = \mathcal{F}\{x_i\} \tag{2.1}$$

for a nonlinear operator $\mathcal{F} : M \to M$ which transforms the system from a state x_i into its next state x_{i+1}. In most cases, directly studying properties of a non-linear operator is a difficult task. Instead, we can work in an embedding of observables $\phi : M \to \mathbb{C}$ of the system's state. Then the mapping $\mathcal{K}\{\phi\} = \phi \circ \mathcal{F}$ which maps an observable ϕ to its composition with the operator $\mathcal{F}$ describes the evolution of these observables

$$\phi(x_{i+1}) = \mathcal{K}\{\phi(x_i)\} \tag{2.2}$$

The linear, infinite dimensional operator $\mathcal{K}$ and is refered to as the *Koopman operator*. Due to the linearity of $\mathcal{K}$, it has a set of eigenvalues and eigenfunctions

$$\mathcal{K}\varphi_j(x) = \lambda_j \varphi_j(x),\ j = 1, 2, \ldots. \tag{2.3}$$

In the basis of eigenfunctions we can subsequently expand any observable

$$\phi(x) = \sum_{j=1}^{\infty} \varphi_j(x) z_j \tag{2.4}$$

or in the case of vector-valued observables $\boldsymbol{\phi} : M \to \mathbb{C}^n$, each component of the observable as

$$\boldsymbol{\phi}(x) = \sum_{j=1}^{\infty} \varphi_j(x) \boldsymbol{z}_j \tag{2.5}$$

for some coefficient vectors $\boldsymbol{z}_j \in \mathbb{C}^n, j = 1, 2, \ldots$. Note that in this case, we can express the iterates of the observables as

$$\boldsymbol{\phi}(x_{m+1}) = \sum_{j=1}^{\infty} \mathcal{K}^m \varphi_j(x_1) \boldsymbol{z}_j = \sum_{j=1}^{\infty} \lambda_j^m \varphi_j(x_1) \boldsymbol{z}_j = \sum_{j=1}^{\infty} \lambda_j^m \alpha_j \boldsymbol{z}_j \tag{2.6}$$

where $\{\lambda_j, \varphi_j\}_{j=1}^{\infty}$ are the eigenvalues and eigenfunctions of the Koopman operator and $\{\boldsymbol{z}_j\}_{j=1}^{\infty}$ the coefficient vectors of the decomposition of the observables in the eigenbasis of $\mathcal{K}$, often called the *Koopman modes*.

It turns out to be more convenient to interpret the expansion (2.5) as an expansion of the observables $\boldsymbol{\phi}$ terms of the Koopman modes $\boldsymbol{z}_j$. Our task is now to find a finite set of Koopman eigenvalues and Koopman modes $\{\widetilde{\lambda}_j, \widetilde{\boldsymbol{z}}_j\}_{j=1}^{k}$ in which we can expand $\boldsymbol{\phi}(x)$ as exactly as possible.

However, achieving an exact expansion is usually only possible in the case where the observations $\boldsymbol{\phi}_i = \boldsymbol{\phi}(x_i)$, $i = 1, 2, \ldots$ evolve linearly

$$\boldsymbol{\phi}_{i+1} = \underline{\mathbf{K}} \boldsymbol{\phi}_i, \; i = 1, 2, \ldots \tag{2.7}$$

for the (unknown) matrix $\underline{\mathbf{K}} \in \mathbb{C}^{n \times n}$ representing the linear operator $\mathcal{K}$.

Thus, the two basic steps of any DMD procedure should be to

1. determine observables $\boldsymbol{\phi}(x_i)$ of the system states x_i which approximately evolve as $\boldsymbol{\phi}_{i+1} = \underline{\mathbf{K}} \boldsymbol{\phi}_i$ for a constant matrix $\underline{\mathbf{K}} \in \mathbb{C}^{n \times n}$;

2. and approximate some eigenvalues and eigenvectors $\{\widetilde{\lambda}_j, \widetilde{\boldsymbol{z}}_j\}_{j=1}^{k}$ of the matrix $\underline{\mathbf{K}}$ based on a sequence of observables $\boldsymbol{\phi}(x_1), \boldsymbol{\phi}(x_2), \ldots, \boldsymbol{\phi}(x_m)$.

2.1 Approximating DMD modes and eigenvalues

We will now present a variety of techniques which approximate eigenvalues and eigenvectors of the matrix $\underline{\mathbf{K}}$ without explicitly forming it.

Krylov DMD As we increase the number of observables, the Krylov matrix

$$\underline{\mathbf{\Phi}}_{1:m} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_m] = [\boldsymbol{\phi}_1, \underline{\mathbf{K}}\boldsymbol{\phi}_1, \ldots, \underline{\mathbf{K}}^{m-1}\boldsymbol{\phi}_1] \tag{2.8}$$

will capture the dominant features of the underlying physical process, which eventually allows us to represent the observable $\boldsymbol{\phi}_m$ as a linear combination of the previous observables

$$\boldsymbol{\phi}_{m+1} = \underline{\mathbf{K}}\boldsymbol{\phi}_m = c_1\boldsymbol{\phi}_1 + c_2\boldsymbol{\phi}_2 + \cdots + c_m\boldsymbol{\phi}_m + \mathbf{r} = \underline{\mathbf{\Phi}}_{1:m}\mathbf{c} + \mathbf{r} \tag{2.9}$$

for a residual vector $\mathbf{r} \in \mathbb{C}^n$ and the coefficient vector $\mathbf{c} = [c_1, c_2, \ldots, c_m]^\top$. Employing the matrix notation we see by combining (2.7) and (2.9) that

$$\underline{\mathbf{K}}\underline{\mathbf{\Phi}}_{1:m} = \underline{\mathbf{\Phi}}_{2:m+1} = \underline{\mathbf{\Phi}}_{1:m}\underline{\mathbf{H}} + \mathbf{r}\boldsymbol{e}_m^\top \tag{2.10}$$

for the upper Hessenberg matrix

$$\underline{\mathbf{H}} = \begin{bmatrix} 0 & & & & c_1 \\ 1 & 0 & & & c_2 \\ & \ddots & \ddots & & \vdots \\ & & 1 & 0 & c_{m-1} \\ & & & 1 & c_m \end{bmatrix}. \tag{2.11}$$

From the analysis of the Arnoldi method, we know that for small residuals $\mathbf{r}$, i.e. in the case where the observable $\boldsymbol{\phi}_{m+1}$ can be represented well in terms of previous observables $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_m$, the eigenvalues of the upper Hessenberg matrix $\underline{\mathbf{H}}$ approximate some of the eigenvalues of $\underline{\mathbf{K}}$ well, and approximate eigenvectors of $\underline{\mathbf{K}}$ can be derived from those of $\underline{\mathbf{H}}$ [13]. For instance, we can find coefficients $c_1, c_2, \ldots, c_m$ which minimize the residual $\mathbf{r}$ by solving the least squares problem

$$\min_{\mathbf{c}} \|\boldsymbol{\phi}_{m+1} - \underline{\mathbf{\Phi}}_{1:m}\mathbf{c}\|_2 \tag{2.12}$$

In practice, this method is often unstable. The main issue is that for achieving a small residual $\mathbf{r}$ it is often necessray to increase m to such an extent that $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots \boldsymbol{\phi}_m$ become linearly dependent, and consequently the matrix $\underline{\mathbf{\Phi}}_{1:m}$ highly ill-conditioned.

Algorithm 2.1: Krylov DMD

Input: Observations $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{m+1}$
Output: DMD modes, eigenvalues, and residuals $\{z_i, \lambda_i, r_i\}_{i=1}^m$

1: Solve $\min_{\mathbf{c}\in\mathbb{C}^m} \|\boldsymbol{\phi}_{m+1} - \underline{\mathbf{\Phi}}_{1:m}\mathbf{c}\|_2$
2: Form the matrix $\underline{\mathbf{H}}$ from (2.11)
3: $[\underline{\boldsymbol{W}}, \underline{\boldsymbol{\Lambda}}] = \texttt{eig}(\underline{\mathbf{H}})$
4: Let $z_i = \underline{\mathbf{\Phi}}_{1:m}\boldsymbol{w}_i$ and $\lambda_i = (\underline{\boldsymbol{\Lambda}})_{ii}$ for $i = 1, 2, \ldots, m$

5: Let $r_i = (\underline{W})_{m,i}$

Schmid DMD In order to counteract the shortcomings of the above introduced Krylov method, we introduce a preprocessing step by first computing the singular value decomposition (SVD) of the matrix $\underline{\Phi}_{1:m} = \underline{U}\underline{\Sigma}\underline{V}^*$. Potential rank-deficiency of the matrix $\underline{\Phi}_{1:m}$ can then be taken account of by removing the singular values which are zero, and removing the corresponding columns from $\underline{U}$ and $\underline{V}$ to form the truncated SVD $\underline{U}_k\underline{\Sigma}_k\underline{V}_k^*$. Inserting this expression into 2.10 and rearranging the terms we get

$$\underline{\widetilde{H}} = \underline{U}_k^*\underline{\Phi}_{2:m+1}\underline{V}_k\underline{\Sigma}_k^{-1} \tag{2.13}$$

Again, the eigenvalues of $\underline{\widetilde{H}}$ approximate the eigenvalues of $\underline{K}$, and the corresponding eigenvectors are computed by left-multiplication with $\underline{U}_k$.

Algorithm 2.2: Schmid DMD

Input: Observations $\phi_1, \phi_2, \ldots, \phi_{m+1}$, tolerance ε
Output: DMD modes, eigenvalues, and residuals $\{z_i, \lambda_i, r_i\}_{i=1}^m$

1: $[\underline{U}, \underline{\Sigma}, \underline{V}] = \texttt{svd}(\underline{\Phi}_{1:m})$
2: Let $k = \max\{k \in \{1, 2, \ldots, m\} | \sigma_k \geqslant \varepsilon\}$
3: Truncate $\underline{U}_k$, $\underline{\Sigma}_k$, and $\underline{V}_k$
4: Compute $\underline{\widetilde{H}} = \underline{U}_k^*\underline{\Phi}_{2:m+1}\underline{V}_k\underline{\Sigma}_k^{-1}$
5: $[\underline{W}, \underline{\Lambda}] = \texttt{eig}(\underline{\widetilde{H}})$
6: Let $z_i = \underline{U}_k w_i$ and $\lambda_i = (\underline{\Lambda})_{ii}$ for $i = 1, 2, \ldots, m$
7: Let $r_i = \|\underline{\Phi}_{2:m+1}\underline{\Sigma}^{-1}\underline{W} - \underline{\Lambda}\underline{Z}\|_2$

QR-compressed DMD When working with high-dimensional observations, it can be useful to find an orthogonal basis in which the observations are then represented more compactly which makes it faster and easier to compute a DMD. In short, we first compute a decomposition $\underline{\Phi}_{1:m} = \underline{Q}\underline{R}_{1:m}$ for an orthogonal matrix $\underline{Q} \in \mathbb{C}^{n\times m}$ and a small matrix $\underline{R}_{1:m} \in \mathbb{C}^{m\times m}$. The DMD can then be computed on $\underline{R}_{1:m}$ and the resulting eigenvectors recovered by left-multiplying them with $\underline{Q}$.

Algorithm 2.3: QR-compressed DMD

Input: Observations $\phi_1, \phi_2, \ldots, \phi_{m+1}$, tolerance ε
Output: DMD modes, eigenvalues, and residuals $\{z_i, \lambda_i, r_i\}_{i=1}^m$

1: $[\underline{Q}, \underline{R}_{1:m}] = \underline{\Phi}_{1:m}$
2: Run algorithm 2.2 on $\underline{R}_{1:m}$ to obtain DMD modes, eigenvalues, and residuals $\{\widetilde{z}_i, \lambda_i, r_i\}_{i=1}^m$
3: Let $z_i = \underline{Q}\widetilde{z}_i$ for $i = 1, 2, \ldots, m$

This compressed representation also has the advantage that we can find efficient algorithms for updating it with new observations or "forgetting" past observations from the representation. The idea behind these *streaming techniques* can be seen in figure 2.1.

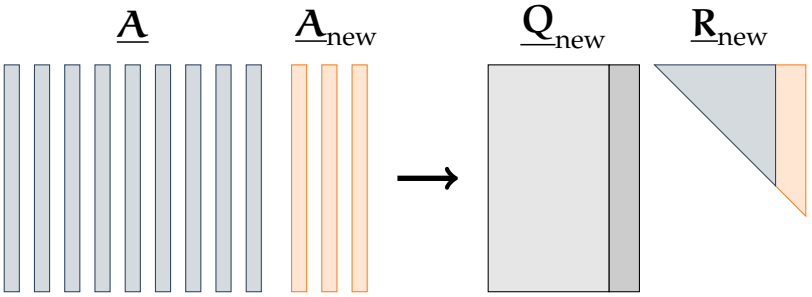

FIGURE 2.1 – The structure of a QR-factorization. The first part of the upper triangular matrix $\underline{\mathbf{R}}$ remains invariant under extension of the snapshot matrix with new observations.

As opposed to the Gram-Schmidt orthogonalization procedure used in [1] we decide to use the Householder QR factorization procedure [6]. In doing so, the orthogonal matrix $\underline{\mathbf{Q}}$ will only need to be formed when when we want to extract DMD eigenvalues and modes, and the formation of $\underline{\mathbf{Q}}$ will not be subject to accumulated round-off errors. However, the way in which the Householder QR algorithm is often presented is not suitable for the streaming setting [17]. Therefore, we need to rewrite the algorithm in such a way that the outermost loop we iterate over the newly added observations. This can be realized by keeping track of previously computed Householder reflectors, which can then effectively be applied to each incoming observation. Compared to re-computing the QR factorization each time we add m_{new} new columns, we can reduce the complexity from $\mathcal{O}(n^2(m + m_{\text{new}}))$ to $\mathcal{O}(nm_{\text{new}}(m + m_{\text{new}}))$.

Algorithm 2.4: Sequential Householder QR

Input: Matrix $\underline{\mathbf{A}}_{\text{new}} \in \mathbb{C}^{n \times m_{\text{new}}}$ where $n > m_{\text{new}}$
Optional input: Matrices $\underline{\mathbf{E}} \in \mathbb{C}^{n \times m}$, $\underline{\mathbf{V}} \in \mathbb{C}^{n \times m}$, $\underline{\mathbf{Q}} \in \mathbb{C}^{n \times m}$, and $\underline{\mathbf{R}} \in \mathbb{C}^{m \times m}$ from previous executions of the algorithm on parts of the matrix $\underline{\mathbf{A}}$
Output: $\underline{\mathbf{Q}}_{\text{new}} \in \mathbb{C}^{n \times m + m_{\text{new}}}$ and $\underline{\mathbf{R}}_{\text{new}} \in \mathbb{C}^{m + m_{\text{new}} \times m + m_{\text{new}}}$ such that $\underline{\mathbf{Q}}_{\text{new}}\underline{\mathbf{R}}_{\text{new}} = [\underline{\mathbf{A}}, \underline{\mathbf{A}}_{\text{new}}]$

Add m_{new} columns to $\underline{\mathbf{Q}}$ and $\underline{\mathbf{V}}$, and m_{new} columns and rows to $\underline{\mathbf{R}}$
Extend $\underline{\mathbf{E}}$ with m_{new} orthogonal columns
for $k = m + 1, m + 2, \ldots, m + m_{\text{new}}$ **do**
    $\mathbf{a} = \underline{\mathbf{A}}(:, k - m)$
    for $j = 1, 2, \ldots, k - 1$ **do** ▷ Apply reflectors from previous iterations
        $\mathbf{a} \leftarrow \mathbf{a} - 2\underline{\mathbf{V}}(:, j)(\underline{\mathbf{V}}(:, j)^* \mathbf{a})$
        $\underline{\mathbf{R}}(j, k) = \underline{\mathbf{E}}(:, j)^* \mathbf{a}$

8: $\quad\quad \mathbf{a} = \mathbf{a} - \underline{\mathbf{R}}(j,k)\underline{\mathbf{E}}(:,j)$
9: $\quad \underline{\mathbf{R}}(k,k) \leftarrow \|\mathbf{a}\|_2$
10: $\quad \alpha \leftarrow \underline{\mathbf{E}}(:,k)^*\mathbf{a}$
11: $\quad$ **if** $\alpha \neq 0$ **then**
12: $\quad\quad \underline{\mathbf{E}}(:,k) \leftarrow \underline{\mathbf{E}}(:,k)(-\alpha/|\alpha|)$
13: $\quad \underline{\mathbf{V}}(:,k) \leftarrow \underline{\mathbf{R}}(k,k)\underline{\mathbf{E}}(:,k) - \mathbf{a}$
14: $\quad \underline{\mathbf{V}}(:,k) \leftarrow \underline{\mathbf{V}}(:,k) - \underline{\mathbf{E}}(:,1:k-1)(\underline{\mathbf{E}}(:,1:k-1)^*\underline{\mathbf{V}}(:,k))$ ▷ Execute twice to reorthogonalize
15: $\quad \sigma \leftarrow \|\underline{\mathbf{V}}(:,k)\|_2$
16: $\quad$ **if** $\sigma \neq 0$ **then**
17: $\quad\quad \underline{\mathbf{V}}(:,k) \leftarrow \underline{\mathbf{V}}(:,k)/\sigma$
18: $\quad$ **else**
19: $\quad\quad \underline{\mathbf{V}}(:,k) \leftarrow \underline{\mathbf{E}}(:,k)$
20: $\quad$ **for** $j = k, k-1, \ldots, 1$ **do** ▷ Apply reflections to $\underline{\mathbf{Q}}$
21: $\quad\quad \underline{\mathbf{Q}}(:,k) = \underline{\mathbf{Q}}(:,k) - 2\underline{\mathbf{V}}(:,j)(\underline{\mathbf{V}}(:,j)^*\underline{\mathbf{Q}}(:,k))$

In a similar way, Householder reflectors can be employed to efficiently update an existing QR-factorization when removing certain columns from the original matrix $\underline{\mathbf{A}}$. Since this procedure is quite standard, we do not go into further detail.

Refined Rayleigh-Ritz DMD The eigenvector approximations we get from computing the Schmid DMD are not optimal in the span $\mathcal{Z}_k = \operatorname{span}\{z_1, z_2, \ldots, z_k\}$ of all eigenvectors [2]. We can improve them by determining the minimizer

$$\min_{z \in \mathcal{Z}_k, \|z\|_2 = 1} \|\underline{\mathbf{K}} z - \lambda z\|_2 = \min_{w \in \mathbb{C}^n, \|w\|_2 = 1} \|\underline{\mathbf{K}}\underline{\mathbf{Z}}_k w - \lambda \underline{\mathbf{Z}}_k w\|_2 = \sigma_{\min}(\underline{\mathbf{K}}\underline{\mathbf{Z}}_k - \lambda \underline{\mathbf{Z}}_k) \tag{2.14}$$

for each eigenvalue λ. What results are "refined" DMD modes and eigenvalues.

Algorithm 2.5: Refined Rayleigh-Ritz DMD

Input: Observations $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{m+1}$, tolerance ε
Output: DMD modes, eigenvalues, and residuals $\{z_i, \lambda_i, r_i\}_{i=1}^m$
1: $[\underline{\mathbf{U}}, \underline{\mathbf{\Sigma}}, \underline{\mathbf{V}}] = \texttt{svd}(\underline{\mathbf{\Phi}}_{1:m})$
2: Let $k = \max\{k \in \{1, 2, \ldots, m\} | \sigma_k \geqslant \varepsilon\}$
3: Truncate $\underline{\mathbf{U}}_k$, $\underline{\mathbf{\Sigma}}_k$, and $\underline{\mathbf{V}}_k$
4: $\underline{\mathbf{B}}_k = \underline{\mathbf{\Phi}}_{2:m+1}(\underline{\mathbf{V}}_k \underline{\mathbf{\Sigma}}_k^{-1})$
5: $[\underline{\mathbf{Q}}, \underline{\mathbf{R}}] = \texttt{qr}([\underline{\mathbf{U}}_k, \underline{\mathbf{B}}_k])$
6: $\underline{\mathbf{H}} = \operatorname{diag}(\operatorname{diag}(\underline{\mathbf{R}}))\underline{\mathbf{R}}(1:k, k+1:2k)$
7: $\widetilde{\underline{\mathbf{\Lambda}}} = \texttt{eig}(\underline{\mathbf{H}})$
8: **for** $i = 1, 2, \ldots, k$ **do**

9: Compute smallest singular value σ_i and right singular vector $\boldsymbol{w}_i$ of

$$\begin{bmatrix} \underline{\mathbf{R}}(1:k, k+1:2k) - \widetilde{\lambda}_i \underline{\mathbf{R}}(1:k, 1:k) \\ \underline{\mathbf{R}}(k+1:2k, k+1:2k) \end{bmatrix}$$

10: Compute $\lambda_i = \boldsymbol{w}_i^* \underline{\mathbf{H}} \boldsymbol{w}_i$
11: Let $\boldsymbol{z}_i = \underline{\mathbf{U}}_k \boldsymbol{w}_i$

Randomized SVD The most costly operation in Schmid's DMD algorithm is the computation of the SVD of the $n \times m$ snapshot-matrix $\underline{\mathbf{\Phi}}_{1:m}$. In most applications of the DMD, we know that the rank of this matrix is known to be significantly lower than the number of observations m. We can make use of the randomized SVD (RSVD) to significantly speed up this procedure at almost no loss of accuracy [5].

Algorithm 2.6: Randomized singular value decomposition

Input: Matrix $\underline{\mathbf{A}} \in \mathbb{C}^{n \times m}$ where $n > m$ and target rank r
Output: $\underline{\mathbf{U}} \in \mathbb{C}^{n \times r}, \underline{\mathbf{\Sigma}} \in \mathbb{C}^{r \times r}, \underline{\mathbf{V}} \in \mathbb{C}^{r \times m}$ such that $\underline{\mathbf{A}} \approx \underline{\mathbf{U}}\underline{\mathbf{\Sigma}}\underline{\mathbf{V}}^*$

1: Generate Gaussian random matrix $\Omega \in \mathbb{R}^{m \times r}$
2: Create a range sketch $\underline{\mathbf{Y}} = \underline{\mathbf{A}}\underline{\mathbf{\Omega}}$
3: Orthogonalize the range sketch $\underline{\mathbf{Q}} = \mathtt{qr}(\underline{\mathbf{Y}})$
4: Compute the economic SVD $[\widetilde{\mathbf{U}}, \underline{\mathbf{\Sigma}}, \underline{\mathbf{V}}] = \mathtt{svd}(\underline{\mathbf{Q}}^* \underline{\mathbf{Y}})$
5: Form $\underline{\mathbf{U}} = \underline{\mathbf{Q}}\widetilde{\mathbf{U}}$

Using this approach we reduce the complexity of this step in Schmid's DMD algorithm from $\mathcal{O}(nm^2)$ to $\mathcal{O}(nmr)$. In table 2.1 we compare runtimes and reconstruction errors for the RSVD with the classical SVD approach on a standard problem for various choices of the target rank.

TABLE 2.1 – Median runtimes and mean absolute reconstruction errors for the DMD algorithm with SVD and RSVD for different target ranks r.

method	r	runtime (s)	reconstruction error
SVD	-	6.414	1.327×10^{-5}
RSVD	50	0.468	1.556×10^{-2}
RSVD	100	1.654	1.346×10^{-4}
RSVD	200	2.557	1.333×10^{-5}

2.2 Reconstruction

Based on the observation in (2.6), we can try to approximate the observations with the help of the approximate DMD modes and eigenvalues, determined with one of the procedures mentioned in the previous section, with the formula

$$\boldsymbol{\phi}_{m+1} \approx \sum_{j=1}^{k} \lambda_j^m \alpha_j z_j. \tag{2.15}$$

Based on (2.6) we know that the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$ represent $\boldsymbol{\phi}_1$ in the span of $\{z_1, z_2, \ldots, z_k\}$. We can simply determine these coefficients by solving the linear system $\boldsymbol{x}_1 = [z_1, z_2, \ldots, z_k]\boldsymbol{\alpha}$.

However, in general, particularly when the linear recurrence (2.7) does not hold exactly, the coefficients $\alpha_j \in \mathbb{C}$, $j = 1, 2, \ldots$ – determined solely based on the first observation $\boldsymbol{\phi}_1$ – will not be acceptable for the remaining trajectory (see figure 2.2). One approach to finding coefficients which better match the whole trajectory is by determining α_j, $j = 1, 2, \ldots$ such that the *reconstruction error* along the whole trajectory is at a minimum, i.e.

$$\min_{\boldsymbol{\alpha} \in \mathbb{C}^k} \sum_{i=1}^{m} \| \boldsymbol{\phi}_i - \sum_{j=1}^{k} \lambda_j^{i-1} \alpha_j z_j \|_2^2 \tag{2.16}$$

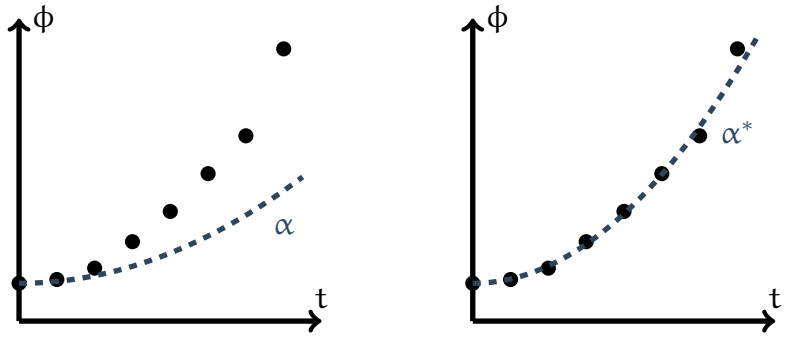

FIGURE 2.2 – For observations which do not evolve exactly linearly, the standard coefficients α in the reconstruction formula often lead to a drift away from the observed observations. Finding coefficients α^* such that the reconstruction error is at its minimum along the whole trajectory can help circumvent this issue.

In [3] it is shown that this problem can be solved with the following algorithm:

Algorithm 2.7: Reconstruction from normal equations

Input: Observables $\underline{\boldsymbol{\Phi}}_{1:m}$, DMD eigenvalues and modes $\{\lambda_j, z_j\}_{j=1}^k$
Output: Coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$

1: $[\underline{\mathbf{Q}}, \underline{\mathbf{R}}] = \texttt{qr}([z_1, z_2, \ldots, z_k])$

2: $\underline{\mathbf{G}} = \underline{\mathbf{Q}}^* \underline{\mathbf{\Phi}}_{1:m}$
3: Assemble the matrix $\underline{\mathbf{V}}$ with $v_{ij} = \lambda_i^j,\ i = 1, 2, \ldots, k,\ j = 0, 1, \ldots, m$
4: $\underline{\mathbf{C}} = (\underline{\mathbf{R}}^* \underline{\mathbf{R}}) * \overline{\underline{\mathbf{V}}\underline{\mathbf{V}}^*}$
5: $\mathbf{b} = \overline{\underline{\mathbf{V}}} * (\underline{\mathbf{R}}^* \underline{\mathbf{G}})\mathbf{1}$
6: Solve least squares problem $\min_{\boldsymbol{\alpha} \in \mathbb{C}^k} \|\underline{\mathbf{C}}\boldsymbol{\alpha} - \mathbf{b}\|_2$

Alternatively, a stability analysis of the normal equations shows that an even better procedure is given by the following algorithm:

Algorithm 2.8: Reconstruction from seminormal equations

Input: Observables $\underline{\mathbf{\Phi}}_{1:m}$, DMD eigenvalues and modes $\{\lambda_j, \mathbf{z}_j\}_{j=1}^k$
Output: Coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$
1: $[\underline{\mathbf{Q}}, \underline{\mathbf{R}}] = \mathtt{qr}([\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k])$
2: Assemble the matrix $\underline{\mathbf{V}}$ with $v_{ij} = \lambda_i^j,\ i = 1, 2, \ldots, k,\ j = 0, 1, \ldots, m$
3: Compute triangular factor $\underline{\mathbf{R}}_S$ of the QR-factorization of $\underline{\mathbf{S}} = \underline{\mathbf{V}}^\top \odot \underline{\mathbf{R}}$
4: $\mathbf{g}_S = (\overline{\underline{\mathbf{V}}} * (\underline{\mathbf{R}}^* \underline{\mathbf{G}}))\mathbf{1}$
5: $\boldsymbol{\alpha} = \underline{\mathbf{R}}_S^{-1}(\underline{\mathbf{R}}_S^{-*}\mathbf{g}_S)$
6: $\mathbf{r}_\square = \underline{\mathbf{G}} - \underline{\mathbf{R}}\,\mathrm{diag}(\boldsymbol{\alpha})\underline{\mathbf{V}}$
7: $\mathbf{r}_S = (\overline{\underline{\mathbf{V}}} * (\underline{\mathbf{R}}\mathbf{r}_\square))\mathbf{1}$
8: $\alpha = \alpha + \underline{\mathbf{R}}_S^{-1}(\underline{\mathbf{R}}^{-*}\mathbf{r}_S)$

3 Wave propagation

Applying a time-stepping scheme to (1.2), we can generate a sequence approximations of the solution of (1.1) in a finite-dimensional subspace, which can be represented by the vectors

$$\mathbf{u}_{i+1} = \mathcal{F}(\mathbf{u}_i) \tag{3.1}$$

where $\mathbf{u}_i \in \mathbb{R}^n,\ i = 1, 2, \ldots$ for a fixed $n \in \mathbb{N}$ and an operator $\mathcal{F}$ which depend on the way the original problem was discretized.

3.1 Linearization of observables

Clearly, the evolution of the system in time described by (1.2) is in general not linear; there is little hope to get high quality approximations from the DMD of the solution vectors. In these cases, it is often advised to augment the feature space with polynomial extensions or kernel methods [9]. But because these methods would ruin the structure which the observations already exhibit, we avoid using these techniques.

Fortunately, in some cases we can construct observables which evolve linearly in time. For example, it can be checked that for constant $\mathbf{f}(t) = \mathbf{f}$ that

$$\boldsymbol{\phi}(t) = \begin{bmatrix} \mathbf{u}(t) \\ \dot{\mathbf{u}}(t) \end{bmatrix} - \begin{bmatrix} & -\underline{\mathbf{M}} \\ \underline{\mathbf{S}} & \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{3.2}$$

evolves linearly with (1.2), i.e. it holds

$$\dot{\boldsymbol{\phi}}(t) = \underline{\mathbf{K}}\boldsymbol{\phi}(t) \tag{3.3}$$

for some matrix $\underline{\mathbf{K}}$. However, such a representation cannot be found for a general $\mathbf{f}(t)$. As a compromise, we "partially" linearize the system by forming the observables

$$\boldsymbol{\phi}(t) = \begin{bmatrix} \mathbf{u}(t) \\ \dot{\mathbf{u}}(t) \end{bmatrix}. \tag{3.4}$$

It turns out that this choice of observables already significantly helps improve the quality of the DMD over merely using $\mathbf{u}(t)$ for this purpose.

3.2 Khatri-Rao structure of the physical problem

In $d \geqslant 2$ dimensions, the factorized nature of the finite element basis functions allows us to approximately express solutions in terms of Kronecker products [7]. Therefore, the vector of observations can roughly be viewed as a Khatri-Rao product $\underline{\mathbf{A}}_1 \odot \underline{\mathbf{A}}_2 \odot \cdots \odot \underline{\mathbf{A}}_d$ or as a sum of q Khatri-Rao products.

To exploit this structure, we now describe a procedure for approximating a given matrix $\underline{\mathbf{A}} \in \mathbb{C}^{n_1 n_2 \ldots n_d \times m}$ by a sum of q Khatri-Rao products; i.e. $\underline{\mathbf{A}} \approx \sum_{k=1}^{q} \underline{\mathbf{A}}_1^{(k)} \odot \underline{\mathbf{A}}_2^{(k)} \odot \cdots \odot \underline{\mathbf{A}}_d^{(k)}$. We consider the minimization problem in the Frobenius norm

$$\min \left\| \underline{\mathbf{A}} - \sum_{k=1}^{q} \underline{\mathbf{A}}_1^{(k)} \odot \underline{\mathbf{A}}_2^{(k)} \odot \cdots \odot \underline{\mathbf{A}}_d^{(k)} \right\|_F^2 = \min \sum_{j=1}^{m} \left\| \mathbf{a}_j - \sum_{k=1}^{q} \mathbf{a}_{1j}^{(k)} \otimes \mathbf{a}_{2j}^{(k)} \otimes \cdots \otimes \mathbf{a}_{dj}^{(k)} \right\|_2^2 \tag{3.5}$$

where $\mathbf{a}_j$ and $\mathbf{a}_{ij}^{(k)}$ denote the jth column of A and $A_i^{(k)}$, respectively. Thus, the columns of $A_i^{(k)}$ are computed by solving m independent minimization problems for each column of A. Moreover,

$$\left\| \mathbf{a}_j - \sum_{k=1}^{q} \mathbf{a}_{1j}^{(k)} \otimes \mathbf{a}_{2j}^{(k)} \otimes \cdots \otimes \mathbf{a}_{dj}^{(k)} \right\|_2^2 = \left\| \mathcal{A}_j - \sum_{k=1}^{q} \mathbf{a}_{1j}^{(k)} \circ \mathbf{a}_{2j}^{(k)} \circ \cdots \circ \mathbf{a}_{dj}^{(k)} \right\|_F^2 \tag{3.6}$$

where $\circ$ denotes the outer product and $\mathcal{A}_j \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$ is obtained by reshaping (and permuting) $\mathbf{a}_j$ into a d-th order tensor. Finally, the vectors $\mathbf{a}_{ij}^{(k)}$ for $i = 1, 2, \ldots, d$ are obtained by computing a CP rank q approximation of $\mathcal{A}_j$. This last problem is rather standard in tensor calculus and several software packages are available for this purpose; e.g. the Tensor Toolbox[1].

[1] https://www.tensortoolbox.org/

Algorithm 3.1: Khatri-Rao rank q approximation

Input: Matrix $\underline{\mathbf{M}} \in \mathbb{R}^{n_1 n_2 \dots n_d \times m}$, partitioned as $\underline{\mathbf{M}} = [\mathbf{m}_1, \dots, \mathbf{m}_m]$, rank q.
Output: Tensors $\mathcal{A}_i \in \mathbb{R}^{n_i \times m \times q}$ for $i = 1, \dots, d$.

1: Initialize $\mathcal{A}_i$ for $i = 1, 2, \dots, d$.
2: **for** $j = 1, 2, \dots, m$ **do**
3: $\quad \mathcal{M}_j \leftarrow \texttt{reshape}(\mathbf{m}_j, (n_d, n_{d-1}, \dots, n_1))$
4: $\quad \mathcal{M}_j \leftarrow \texttt{permute}(\mathcal{M}_j, (d, d-1, \dots, 1))$
5: $\quad [\mathcal{A}_1(:, j, :), \dots, \mathcal{A}_d(:, j :)] = \texttt{cp}(\mathcal{M}_j, \text{rank} = q)$

This structure may be exploited within the DMD framework. For large scale problems, the matrix of observations cannot be formed explicitly due to the prohibitive storage requirements, as shown in figure 3.1.

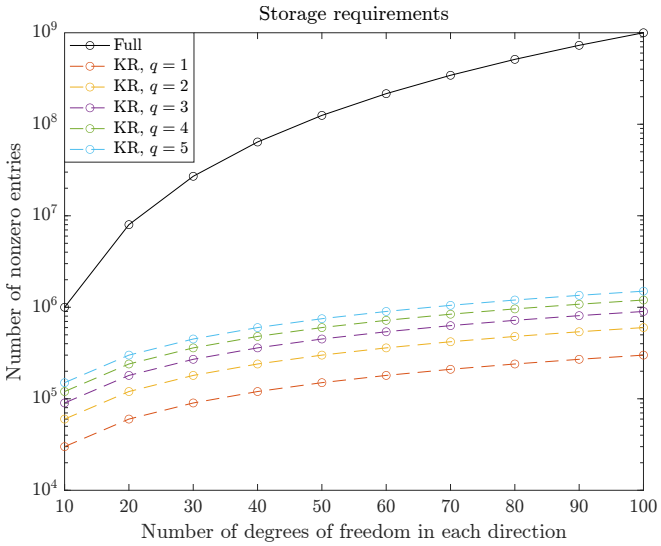

FIGURE 3.1 – Comparison of the full storage of the snapshot matrix and its structured Khatri-Rao (KR) approximation for $d = 3$, $m = 1000$ and various values of q.

Therefore, we generally cannot afford computing its full SVD and we must seek alternatives. The Golub-Kahan Lanczos bidiagonalization method is a well established method for computing a few dominant singular values and vectors. It is summarized below.

Algorithm 3.2: Golub-Kahan Lanczos bidiagonalization

Input: Matrix $\underline{\mathbf{A}} \in \mathbb{R}^{n \times m}$, unit norm starting vector $\mathbf{p}_1 \in \mathbb{R}^m$.
Output: Approximate truncated SVD $\underline{\mathbf{A}} \approx \underline{\mathbf{U}}_s \underline{\mathbf{\Sigma}}_s \underline{\mathbf{V}}_s^\top$.

1: Set $\underline{\mathbf{P}}_1 = \mathbf{p}_1$, $\mathbf{q}_1 = \underline{\mathbf{A}}\mathbf{p}_1$, $\alpha_1 = \|\mathbf{q}_1\|$, $\mathbf{q}_1 = \mathbf{q}_1/\alpha_1$, $\underline{\mathbf{Q}}_1 = \mathbf{q}_1$.
2: **for** $j = 1, \dots, s$ **do**

3: $\mathbf{r}_j = \underline{\mathbf{A}}^\top \mathbf{q}_j - \alpha_j \mathbf{p}_j$
4: $\mathbf{r}_j = \mathbf{r}_j - \underline{\mathbf{P}}_j \underline{\mathbf{P}}_j^\top \mathbf{r}_j$ ▷ Full reorthogonalization
5: $\beta_j = \|\mathbf{r}_j\|_2$
6: $\mathbf{p}_{j+1} = \mathbf{r}_j / \beta_j$
7: $\underline{\mathbf{P}}_{j+1} = [\underline{\mathbf{P}}_j \ \mathbf{p}_{j+1}]$
8: $\mathbf{q}_{j+1} = \underline{\mathbf{A}}\mathbf{p}_{j+1} - \beta_j \mathbf{q}_j$
9: $\mathbf{q}_{j+1} = \mathbf{q}_{j+1} - \underline{\mathbf{Q}}_j \underline{\mathbf{Q}}_j^\top \mathbf{q}_{j+1}$ ▷ Full reorthogonalization
10: $\alpha_{j+1} = \|\mathbf{q}_{j+1}\|_2$
11: $\mathbf{q}_{j+1} = \mathbf{q}_{j+1} / \alpha_{j+1}$
12: $\underline{\mathbf{Q}}_{j+1} = [\underline{\mathbf{Q}}_j \ \mathbf{q}_{j+1}]$
13: Set $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_s)$
14: Set $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{s-1})$
15: Set $\underline{\mathbf{B}}_s = \operatorname{diag}(\boldsymbol{\alpha}, 0) + \operatorname{diag}(\boldsymbol{\beta}, 1)$ ▷ Upper bidiagonal matrix
16: $\underline{\mathbf{B}}_s = \underline{\mathbf{F}}_s \underline{\mathbf{\Sigma}}_s \underline{\mathbf{G}}_s$ ▷ SVD of $\underline{\mathbf{B}}_s$
17: Set $\underline{\mathbf{U}}_s = \underline{\mathbf{Q}}_s \underline{\mathbf{F}}_s$
18: Set $\underline{\mathbf{V}}_s = \underline{\mathbf{P}}_s \underline{\mathbf{G}}_s$

The method requires computing matrix-vector products with $\underline{\mathbf{A}}$ and $\underline{\mathbf{A}}^\top$. However, as we have seen in figure 3.1, forming these matrices explicitly is often not possible due to memory restrictions. The following proposition gives us a formula for computing matrix-vector products of matrices in the Khatri-Rao format to circumvent this issue.

Proposition 3.3: Matrix-vector multiplication in Khatri-Rao format

Let $\underline{\mathbf{A}}_i \in \mathbb{R}^{n_i \times m}$ for $i = 1, \ldots, d$, $\mathbf{x} \in \mathbb{R}^m$, $\mathcal{Y} \in \mathbb{R}^{n_d \times \cdots \times n_1}$ and $\mathbf{y} = \operatorname{vec}(\mathcal{Y})$. Then,

1. $(\underline{\mathbf{A}}_1 \odot \cdots \odot \underline{\mathbf{A}}_d)\mathbf{x} = \operatorname{vec}(\llbracket \mathbf{x}; \underline{\mathbf{A}}_d, \ldots, \underline{\mathbf{A}}_1 \rrbracket)$,
2. $(\underline{\mathbf{A}}_1 \odot \cdots \odot \underline{\mathbf{A}}_d)^\top \mathbf{y} = \operatorname{diag}(\mathcal{Y} \circ_1 \underline{\mathbf{A}}_d \cdots \circ_d \underline{\mathbf{A}}_1)$.

Proof. We prove each property below

1. Using the definition of the CP-format and standard manipulations on tensor products we obtain

$$\begin{aligned} \operatorname{vec}(\llbracket \mathbf{x}; \underline{\mathbf{A}}_d, \ldots, \underline{\mathbf{A}}_1 \rrbracket) &= \operatorname{vec}\Big(\sum_{j=1}^{m} x_j (\mathbf{a}_j^{(d)} \circ \cdots \circ \mathbf{a}_j^{(1)})\Big) \\ &= \sum_{j=1}^{m} x_j (\mathbf{a}_j^{(1)} \otimes \cdots \otimes \mathbf{a}_j^{(d)}) \\ &= (\underline{\mathbf{A}}_1 \odot \cdots \odot \underline{\mathbf{A}}_d)\mathbf{x}. \end{aligned} \tag{3.7}$$

2. Using index notation we obtain

$$
\begin{aligned}
(\mathcal{Y} \circ_1 \underline{\mathbf{A}}_d \cdots \circ_d \underline{\mathbf{A}}_1)_{jj\ldots j} &= \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} y_{i_1 \ldots i_n} a^{(d)}_{i_1 j} \ldots a^{(1)}_{i_d j} \\
&= \langle \mathcal{Y}, \mathbf{a}^{(d)}_j \circ \cdots \circ \mathbf{a}^{(1)}_j \rangle \\
&= \langle \operatorname{vec}(\mathcal{Y}), \operatorname{vec}(\mathbf{a}^{(d)}_j \circ \cdots \circ \mathbf{a}^{(1)}_j) \rangle \\
&= \langle \mathbf{y}, \mathbf{a}^{(1)}_j \otimes \cdots \otimes \mathbf{a}^{(d)}_j \rangle \\
&= ((\underline{\mathbf{A}}_1 \odot \cdots \odot \underline{\mathbf{A}}_d)^\top \mathbf{y})_j. \qquad (3.8)
\end{aligned}
$$

This proves the second assertion component-wise.

□

3.3 Downsampling

It can happen that observations $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2 \ldots, \boldsymbol{\phi}_m$ are available at a significantly higher sample rate than the rate of change of the relevant dynamics of a system. For example, when using an explicit method to solve (1.2), the size of the steps in time may be limited by stability considerations. In this case, the DMD algorithms may be able to model the dynamics of a system from just a subset of these observations $\boldsymbol{\phi}_s, \boldsymbol{\phi}_{2s} \ldots, \boldsymbol{\phi}_{ks}$ where $s \in \mathbb{N}$ represents the sampling step and k is such that $ks \leqslant m$. This downsampling reduces the number of observations the algorithm has to treat by a factor of s, which is usually equivalent to a speed-up of the algorithms by a factor of s. The downsampling also helps draw the attention of the algorithm on the global evolution rather than local oscillations which usually result from noise.

In the setting of wave propagation, we can observe that the algorithms behave differently when we downsample the observations (see figure 3.2). The DMD algorithms which are based on Schmid's approximation of the Rayleigh quotient work just as well when only every 7-th observation is considered in the computation, compared to when every observation is used. On the other hand, both the companion matrix based algorithm and the DMD on the QR factorization do not react as favorably to a reduce sampling rate.

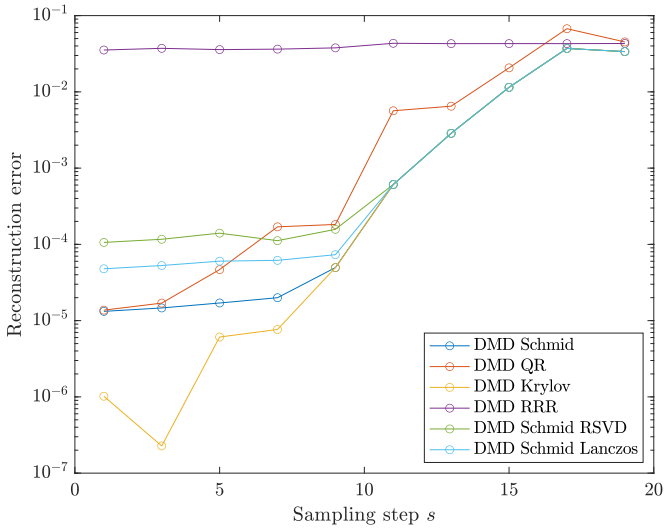

FIGURE 3.2 – Effect of downsampling by a factor s on the reconstruction error of the observations from the problem described in section 4.1.3.

4 Numerical results

4.1 Single-patch geometries

4.1.1 2D standing wave

The first example is the academic example of a standing wave on a unit square, described by $u(x, y, t) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi t)$ for $t \in [0, 1]$. The initial and boundary conditions of the PDE are computed accordingly. Two observations of the discretized solution are shown in figure 4.1. The snapshot matrix in this example is 1089×201 but we will consider throughout this section different levels of refinement in space and time to illustrate different properties of the method.

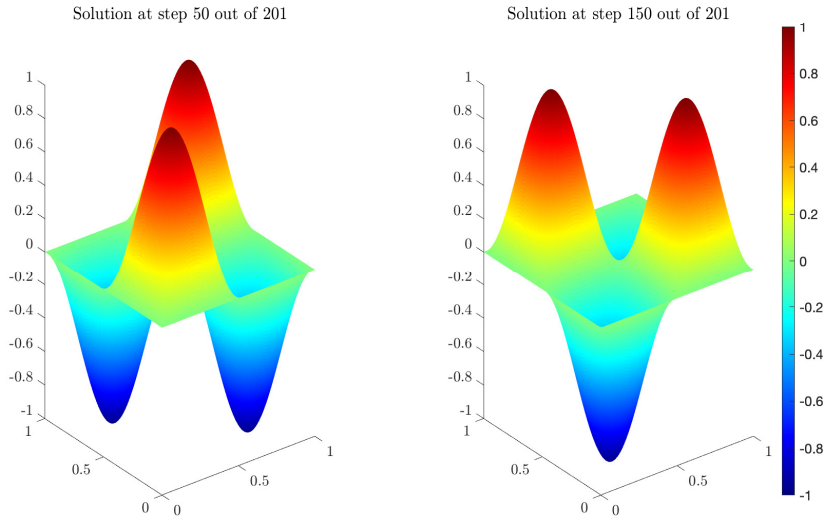

FIGURE 4.1 – Standing wave of the unit square, discretized with cubic B-splines and 30 subdivisions in each direction (section 4.1.1).

4.1.2 3D standing wave

The second example is simply the 3D counterpart of the first one and provides a simple and yet conclusive example of the "curse of dimensionality"; i.e. the prohibitive (exponential) growth of memory and operations with the dimension. The exact solution is $u(x, y, z, t) = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)\sin(2\pi t)$ for $t \in [0, 1]$. Two observations of the discretized solution, shown along slices of the unit cube, are displayed in figure 4.2. The snapshot matrix for this problem is 35937×201; i.e. 33 times larger than its 2D counterpart.

4.1.3 2D Laplace Fraschini

Our third example, inspired from [4], is more realistic and models the acoustic wave equation (1.1) in a non-homogeneous medium. The wave speed of the medium is $c(x, y) = 1 + y$. We consider homogeneous Neumann boundary

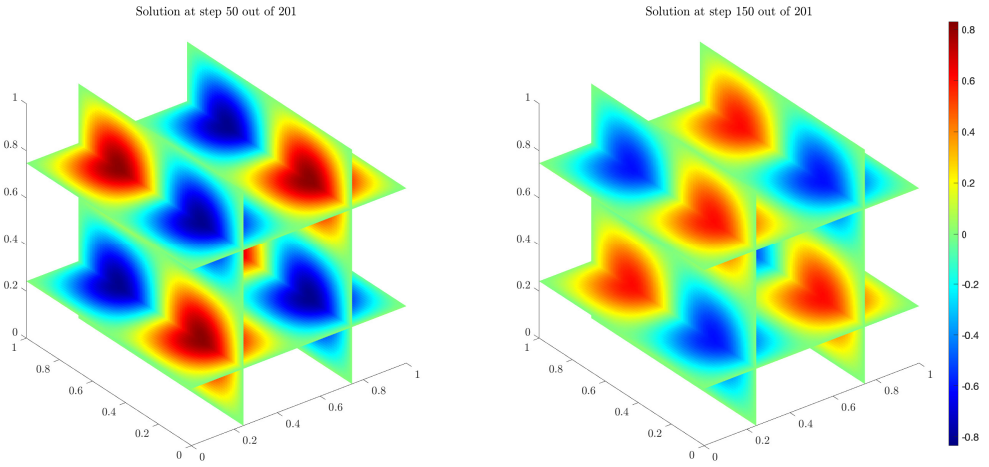

FIGURE 4.2 – Standing wave of the unit cube, discretized with cubic B-splines and 30 subdivisions in each direction (section 4.1.2).

conditions over the entire boundary and initial conditions $u_0(x, y) = e^{-124x^2}$ and $v_0(x, y) = 0$. The Gaussian pulse travels from left to right in the medium before hitting the right boundary and rebounding. figure 4.3 shows a few observations of the solution. The snapshot matrix has size 62025×1001. The geometry is discretized with quintic spline of C^4 smoothness.

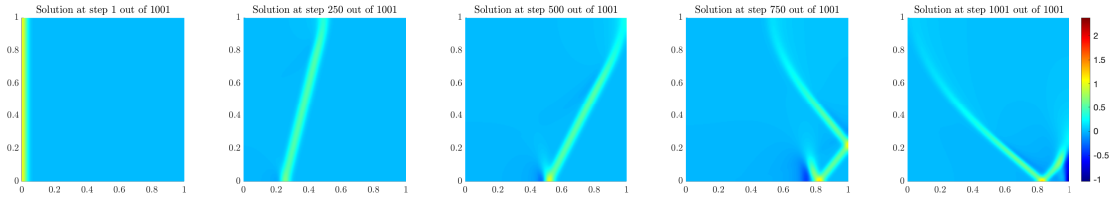

FIGURE 4.3 – Observations for the wave propagation in a medium with a non-homogeneous index of refraction (section 4.1.3).

	n	m	q
4.1.1 I	1089	201	1
4.1.1 II	41209	1501	1
4.1.2 I	79507	201	1
4.1.2 II	79507	1001	1
4.1.3	65025	1001	40

TABLE 4.1 – Summary of the single-patch examples with the corresponding problem sizes. n is the size of each observation, while m is the number of observations. q represents the number of summands in the approximation.

	SVD	Lanczos SVD		Randomized SVD	
		Explicit	Implicit	Explicit	Implicit
4.1.1 I	1.27×10^{-2}	6.81×10^{-4}	9.36×10^{-4}	1.02×10^{-3}	2.93×10^{-3}
4.1.1 II	5.71×10^{0}	3.85×10^{-2}	2.11×10^{-2}	5.13×10^{-2}	2.71×10^{-2}
4.1.2 I	4.01×10^{-1}	1.70×10^{-2}	1.73×10^{-1}	1.98×10^{-2}	1.52×10^{-1}
4.1.2 II	5.37×10^{0}	4.61×10^{-2}	8.22×10^{-1}	6.40×10^{-2}	7.29×10^{-1}
4.1.3	4.80×10^{0}	1.77×10^{0}	1.44×10^{1}	4.98×10^{-1}	1.44×10^{1}

TABLE 4.2 – Timings (in seconds) for Schmid's DMD for the observations $\mathbf{u}$. Green and red cells identify the smallest and largest timings, respectively, for each example.

	SVD	Lanczos SVD		Randomized SVD	
		Explicit	Implicit	Explicit	Implicit
4.1.1 I	2.12×10^{-2}	2.10×10^{-3}	1.50×10^{-1}	2.13×10^{-3}	3.58×10^{-2}
4.1.1 II	1.09×10^{1}	9.27×10^{-2}	2.48×10^{0}	9.65×10^{-2}	1.80×10^{0}
4.1.2 I	7.01×10^{-1}	8.50×10^{-2}	1.10×10^{0}	3.85×10^{-2}	2.97×10^{-1}
4.1.2 II	9.83×10^{0}	2.99×10^{-1}	5.47×10^{0}	1.18×10^{-1}	1.55×10^{0}
4.1.3	9.03×10^{0}	5.51×10^{0}	3.80×10^{2}	1.09×10^{0}	1.32×10^{1}

TABLE 4.3 – Timings (in seconds) for Schmid's DMD for the augmented observations $[\mathbf{u}, \dot{\mathbf{u}}]$. Green and red cells identify the smallest and largest timings, respectively, for each example.

	Schmid	QR	Krylov	RRR	RSVD	Lanczos
4.1.1 I	6.46×10^{-5}	6.46×10^{-5}	8.54×10^{-5}	2.18×10^{-1}	6.46×10^{-5}	6.46×10^{-5}
4.1.1 II	1.32×10^{-6}	1.32×10^{-6}	3.15×10^{-6}	2.50×10^{-1}	1.32×10^{-6}	1.32×10^{-6}
4.1.2 I	9.23×10^{-2}	9.02×10^{-2}	1.47×10^{-1}	2.05×10^{-1}	9.23×10^{-2}	9.23×10^{-2}
4.1.2 II	9.34×10^{-2}	9.29×10^{-2}	1.85×10^{-1}	2.06×10^{-1}	9.34×10^{-2}	9.34×10^{-2}
4.1.3	1.20×10^{-2}	6.36×10^{-3}	6.88×10^{-11}	2.53×10^{0}	9.58×10^{-2}	1.38×10^{-1}

TABLE 4.4 – Reconstruction errors for each example. Red cells identify the largest reconstruction errors.

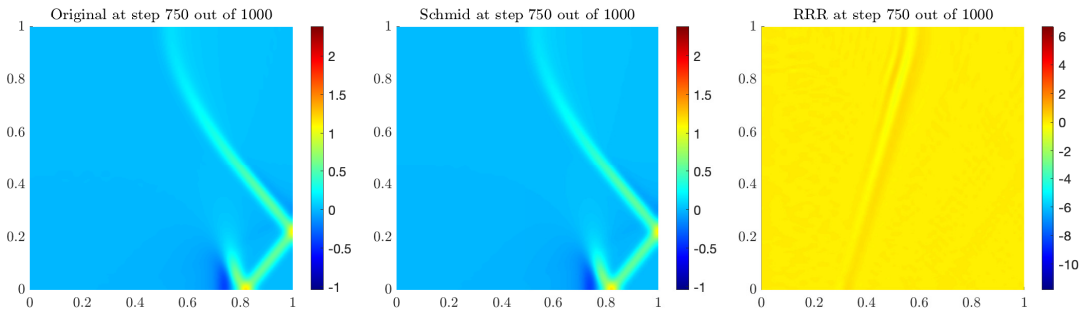

FIGURE 4.4 – Reconstruction comparison for section 4.1.3.

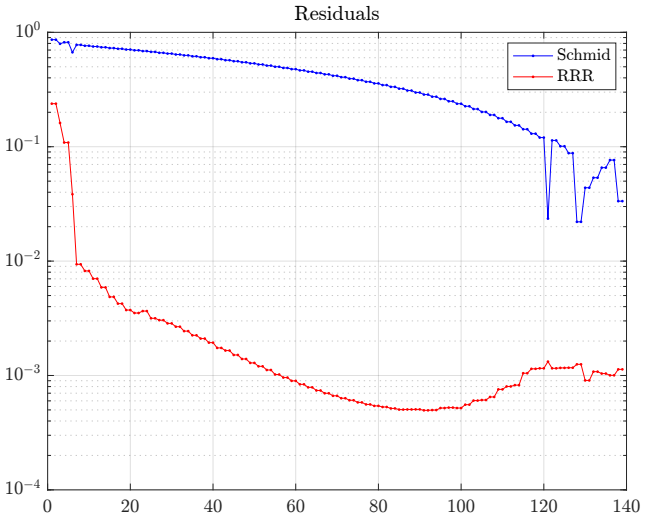

FIGURE 4.5 – Residuals comparison for section 4.1.3.

4.2 Multi-patch geometries

4.2.1 Square with hole

Our first example is a slight variation of section 4.1.3, where a small cavity is added in the middle of the square. The homogeneous Dirichlet boundary conditions prescribed over its boundary create multiple reflections, which further complicate the dynamics. The initial conditions are $u_0(x, y) = e^{-30^2(x+0.6)^2}$ and $v_0(x, y) = 0$. The wave velocity is constant ($c = 1$) and all other data are set to zero. observations of the solution are shown in 4.6. The snapshot matrix for this problem has size 62000×1001.

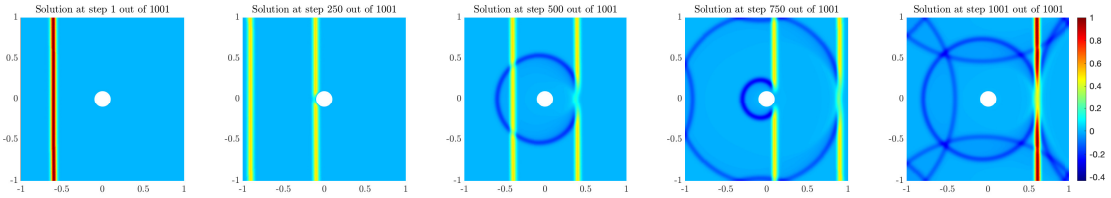

FIGURE 4.6 – Solution observations.

4.2.2 Rod

Our second example is simpler and models a traction test, commonly employed in material science for measuring the resistance of rods. Non-homogeneous Dirichlet boundary conditions are imposed at both ends of the rod and in the initial phase of the test, the slowly varying displacement field follows a linear elastic model. Figure 4.7 shows the horizontal displacement field at different times. The snapshot matrix for this problem has size 3096×1501.

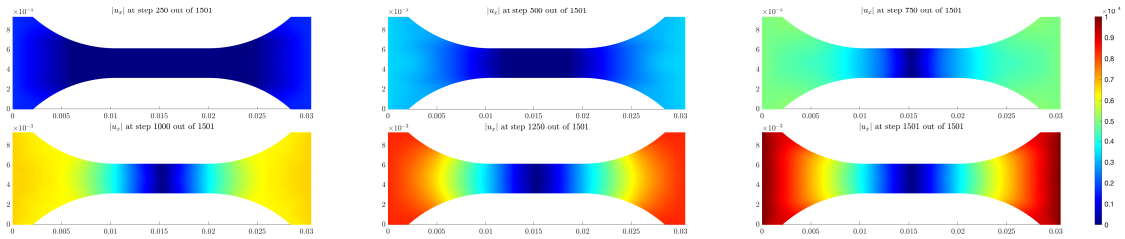

FIGURE 4.7 – Observations of the displacement field in the traction test on a rod (section 4.2.2).

The size of the snapshot matrix for both examples is recalled in table 4.5.

	n	m
4.2.1	62000	1001
4.2.2	3096	1501

TABLE 4.5 – Summary of the multi-patch examples with the corresponding problem sizes. n is the size of each observation, while m is the number of observations.

	SVD	Lanczos SVD	Randomized SVD
4.2.1	8.63×10^{0}	3.25×10^{0}	1.01×10^{0}
4.2.2	2.55×10^{0}	3.05×10^{-1}	1.13×10^{-1}

TABLE 4.6 – Timings (in seconds) for Schmid's DMD for the augmented observations $[\mathbf{u}, \dot{\mathbf{u}}]$. Green and red cells identify the smallest and largest timings, respectively, for each example.

	Schmid	QR	Krylov	RRR	RSVD	Lanczos
4.2.1	4.27×10^{-4}	4.28×10^{-4}	1.16×10^{0}	9.20×10^{0}	2.53×10^{-1}	1.87×10^{-1}
4.2.2	1.71×10^{-3}	1.69×10^{-3}	5.66×10^{-3}	3.55×10^{-1}	1.74×10^{-2}	1.55×10^{-2}

TABLE 4.7 – Reconstruction errors for each example. Red cells identify the largest reconstruction errors.

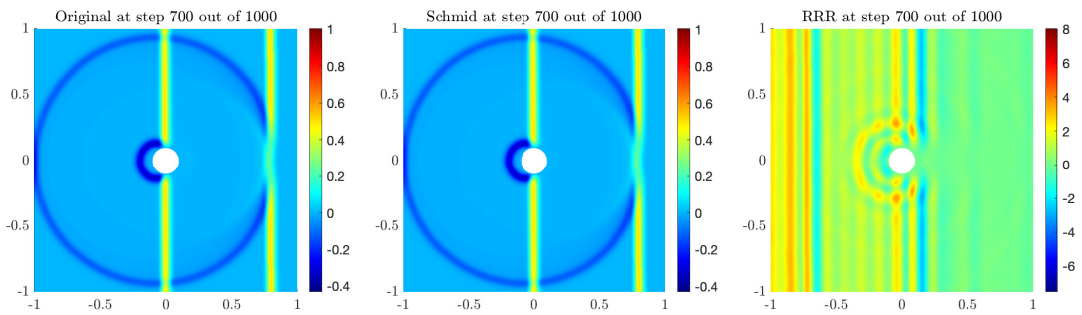

FIGURE 4.8 – Reconstruction comparison for section 4.2.1.

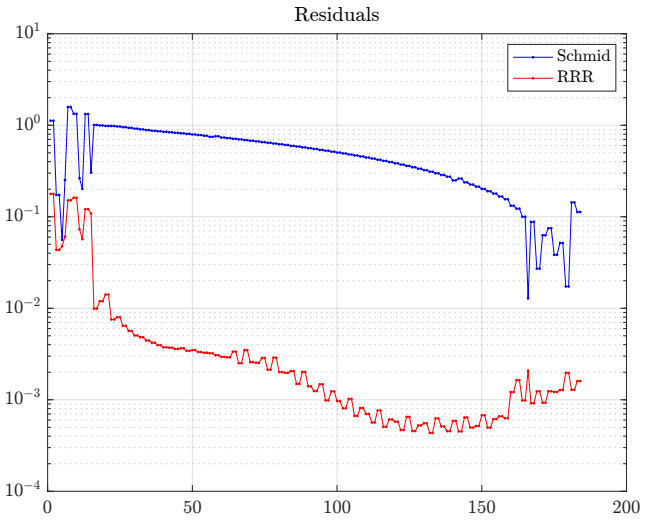

FIGURE 4.9 – Residuals comparison for section 4.2.1.

4.3 Streaming

In our last numerical experiment we employ the QR-compressed DMD algorithm which is based on the sequential Householder QR factorization which we have developed in section 2.1. We use it to see how the reconstruction error changes as we increase the number of observations used as inputs to the algorithm. We do this for the example from section 4.1.3. This allows us to judge for how long the extracted DMD modes can represent the system well until the first and last observation are too different from each other.

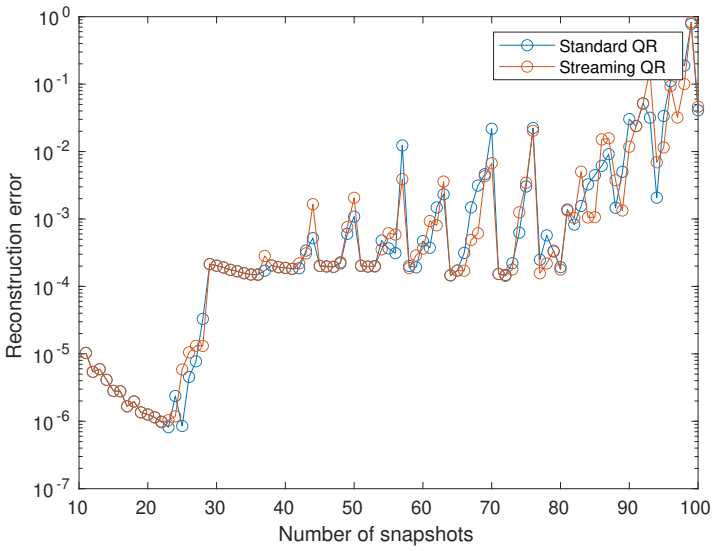

FIGURE 4.10 – Evolution of reconstruction when sequentially adding new snapshots to the QR-compressed DMD algorithm. We compare our streaming QR compression based on Householder reflectors to the full QR factorization every time an observation is added.

5 Discussion

In our project we have extended the DMD algorithms for problems governed by the wave equation. In particular, we have proposed multiple ways which allow faster and more scalable computations of the DMD eigenvalues and modes of certain observations. We have verified our developments on various examples.

Overall, we see that our adaptions of the DMD algorithms to wave-like problems are capable of modeling our test cases reasonably well. We are able to show that by using alternative methods for the computation of the SVD which exploit the structure of the observations, we can speed up the procedure considerably. Further, exploiting the approximate Khatri-Rao structure of the observations yields a scalable and efficient algorithm for computing DMD eigenvalues and modes in this setting.

During our project we have observed that treating the DMD algorithms as "black-box" tools can often lead to unexpected results, especially in the case where the observations evolve highly non-linearly. We have come to the conclusion that the most important part of any DMD is the construction of observations which evolve linearly; either through hand-picked augmentations or informed by the underlying physical system. However, despite having constructed observables which evolve almost linearly in time, we see for example in figure 4.10 that the reconstruction error can only be kept acceptably for observations which span only a small time window. That is, we are only able to model the evolution of the system locally, but not globally. Furthermore, there is an inherent inability of the SVD to capture translational invariances and correlations between observations, which are crucial in many of our examples [10].

We observe that the refinement of the DMD modes on one hand improves the residual significantly, but on the other gives us significantly worse reconstructions of the observations based on the DMD eigenvalues and modes. This could be due to many causes, but we think that the non-linear nature of the evolution of the observations causes the refined DMD modes to span a space which – despite very small residuals – is no longer representative of the dynamics.

In the future it might be interesting to also try to convert the augmented observations to a format which can be approximated by Khatri-Rao products. The tensor product structure might also be exploited with a structured sketch in the RSVD computation.

References

[1] Zlatko Drmač. A LAPACK Implementation of the Dynamic Mode Decomposition. *ACM Transactions on Mathematical Software*, 50(1):1:1–1:32, 2024. DOI: 10.1145/3640012.

[2] Zlatko Drmač, Igor Mezić, and Ryan Mohr. Data Driven Modal Decompositions: Analysis and Enhancements. *SIAM Journal on Scientific Computing*, 40(4):A2253–A2285, 2018. DOI: 10.1137/17M1144155.

[3] Zlatko Drmač, Igor Mezić, and Ryan Mohr. On Least Squares Problems with Certain Vandermonde–Khatri–Rao Structure with Applications to DMD. *SIAM Journal on Scientific Computing*, 42(5):A3250–A3284, 2020. DOI: 10.1137/19M1288474.

[4] Sara Fraschini, Gabriele Loli, Andrea Moiola, and Giancarlo Sangalli. An unconditionally stable space-time isogeometric method for the acoustic wave equation, 2024.

[5] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, 2011. DOI: 10.1137/090771806.

[6] Nicholas J. Higham. 19. QR Factorization. In *Accuracy and Stability of Numerical Algorithms*, Other Titles in Applied Mathematics, pages 353–380. Society for Industrial and Applied Mathematics, 2002. ISBN 978-0-89871-521-7. DOI: 10.1137/1.9780898718027.ch19.

[7] Boris N. Khoromskij and Christoph Schwab. Tensor-Structured Galerkin Approximation of Parametric and Stochastic Elliptic PDEs. *SIAM Journal on Scientific Computing*, 33(1):364–385, 2011. DOI: 10.1137/100785715.

[8] B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931. DOI: 10.1073/pnas.17.5.315.

[9] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. Chapter 10: DMD on Nonlinear Observables. In *Dynamic Mode Decomposition*, Other Titles in Applied Mathematics, pages 159–176. Society for Industrial and Applied Mathematics, 2016. ISBN 978-1-61197-449-2. DOI: 10.1137/1.9781611974508.ch10.

[10] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. Chapter 1: Dynamic Mode Decomposition: An Introduction. In *Dynamic Mode Decomposition*, Other Titles in Applied Mathematics, pages 1–24. Society for Industrial and Applied Mathematics, 2016. ISBN 978-1-61197-449-2. DOI: 10.1137/1.9781611974508.ch1.

[11] Alfio Quarteroni. *Numerical Models for Differential Problems*, volume 16 of *MS&A*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-49315-2 978-3-319-49316-9. DOI: 10.1007/978-3-319-49316-9.

[12] Miha Rot, Martin Horvat, and Gregor Kosec. Dynamic mode decomposition as an analysis tool for time-dependent partial differential equations. In *2022 7th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6, 2022. DOI: 10.23919/SpliTech55088.2022.9854243.

[13] Yousef Saad. 6. Krylov Subspace Methods. In *Numerical Methods for Large Eigenvalue Problems*, Classics in Applied Mathematics, pages 125–162. Society for Industrial and Applied Mathematics, 2011. ISBN 978-1-61197-072-2. DOI: 10.1137/1.9781611970739.ch6.

[14] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. DOI: 10.1017/S0022112010001217.

[15] Peter J. Schmid. Dynamic Mode Decomposition and Its Variants. *Annual Review of Fluid Mechanics*, 54(1):225–254, 2022. DOI: 10.1146/annurev-fluid-030121-015835.

[16] C. Hari Manoj Simha and Mohammad Biglarbegian. A note on the use of Dynamic Mode Decomposition in mechanics. *Mechanics Research Communications*, 120:103848, 2022. DOI: 10.1016/j.mechrescom.2022.103848.

[17] Lloyd N. Trefethen. Householder triangularization of a quasimatrix. *IMA Journal of Numerical Analysis*, 30(4):887–897, 2010. DOI: 10.1093/imanum/drp018.