Numerical aspects of the Koopman and the dynamic mode decompositions for model reduction

Zlatko Drmač
Department of Mathematics, University of Zagreb, Croatia

Research supported by: DARPA Contract HR0011-16-C-0116 *"On a Data-Driven, Operator-Theoretic Framework for Space-Time Analysis of Process Dynamics"* and the DARPA Contract HR0011-18-9-0033 *"The Physics of Artificial Intelligence"*

joint work with

Igor Mezić and Ryan Mohr, University of California, Santa Barbara

8th European Congress of Mathematics, Portorož, Slovenia
June 20–26 2021

Overview

Setting the scene: DS and Koopman operator

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

with state space $\mathcal{X}$ (smooth N-dimensional compact manifold, with Borel σ algebra $\mathcal{B}$; $\mathcal{X} \subset \mathbb{R}^N$) and vector-valued nonlinear function $\mathbf{F} : \mathcal{X} \to \mathbb{R}^N$. The corresponding flow map is

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

The state may not be accessible. Instead, have observable $f : \mathcal{X} \to \mathbb{C}$, $f \in \mathcal{F}$; e.g. $\mathcal{F} = L^p(\mathcal{X}, \mu)$ $(1 \le p \le \infty)$.

Koopman operator semigroup $(\mathcal{U}_{\boldsymbol{\varphi}^t})_{t \ge 0}$

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

$\mathcal{U}_{\boldsymbol{\varphi}^t}$ is linear operator. It is an infinite dimensional linearization of (1) that takes the action into the space $\mathcal{F}$ of observables.

Setting the scene: DS and Koopman operator

Consider a discrete dynamical system $\mathbf{s}_{i+1} = \mathbf{T}(\mathbf{s}_i)$, where $\mathbf{T} : \mathcal{X} \longrightarrow \mathcal{X}$ is a measurable nonlinear map on a state space $\mathcal{X}$ and $i \in \mathbb{Z}$. The Koopman operator $\mathcal{U} \equiv \mathcal{U}_{\mathbf{T}}$ for the discrete system is defined analogously by

$$\mathcal{U}f = f \circ \mathbf{T}, \quad f \in \mathcal{F}. \tag{4}$$

If we run a numerical simulation of the ODE's (1) in a time interval $[t_0, t_*]$, the numerical solution is obtained on a discrete equidistant grid with fixed time lag Δt: $t_0,\ t_1 = t_0 + \Delta t,\ \ldots,\ t_{i-1} = t_{i-2} + \Delta t,\ t_i = t_{i-1} + \Delta t,\ \ldots$ In this case, a black-box software toolbox acts as a discrete dynamical system $\mathbf{s}_i = \mathbf{T}(\mathbf{s}_{i-1})$ that produces the discrete sequence of $\mathbf{s}_i \approx \mathbf{x}(t_i)$; this is sampling with noise.
For $t_i = t_0 + i\Delta t$ we have (using $\boldsymbol{\varphi}^{\Delta t}$, $\mathcal{U}_{\boldsymbol{\varphi}^{\Delta t}}$ and the group property)

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

where $\mathcal{U}^i_{\boldsymbol{\varphi}^{\Delta t}} = \mathcal{U}_{\boldsymbol{\varphi}^{\Delta t}} \circ \ldots \circ \mathcal{U}_{\boldsymbol{\varphi}^{\Delta t}}$.

Setting the scene: DS and Koopman operator

On the other hand, using $\mathcal{U}f = f \circ \mathbf{T}$, $\mathbf{s}_i \approx \mathbf{x}(t_i)$,

$$f(\mathbf{s}_i) = f(\mathbf{T}(\mathbf{s}_{i-1})) = \ldots = f(\mathbf{T}^i(\mathbf{s}_0)) = (\mathcal{U}^i f)(\mathbf{s}_0), \tag{5}$$

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

$$(\mathcal{U}^i f)(\mathbf{s}_0) \approx (\mathcal{U}^i_{\boldsymbol{\varphi}^{\Delta t}} f)(\mathbf{s}_0), \;\; f \in \mathcal{F},\; \mathbf{s}_0 \in \mathcal{X}, \;\; i = 0, 1, 2, \ldots \tag{6}$$

This can be obviously extended to vector valued observables:

$$\text{for } \mathbf{g} = (g_1, \ldots, g_d) : \mathcal{X} \longrightarrow \mathbb{C}^d \text{ define } \mathcal{U}_d\mathbf{g} = \begin{pmatrix} g_1 \circ \mathbf{T} \\ \vdots \\ g_d \circ \mathbf{T} \end{pmatrix} = \begin{pmatrix} \mathcal{U}g_1 \\ \vdots \\ \mathcal{U}g_d \end{pmatrix}. \tag{7}$$

The observables can be physical quantities (e.g. temperature, pressure, energy) and mathematical constructs using suitable classes of functions (e.g. multivariate Hermite polynomials, radial basis functions). In particular, if we set $d = N$, $g_i(\mathbf{s}) = e_i^T\mathbf{s}$, where $\mathbf{s} \in \mathbb{C}^N$, $e_i = (\boldsymbol{\delta}_{ji})_{j=1}^N$, $i = 1, \ldots, N$, then $\mathbf{g}(\mathbf{s}) = \mathbf{s}$ is full state observable and $(\mathcal{U}_d\mathbf{g})(\mathbf{s}_i) = \mathbf{s}_{i+1}$.

Data driven framework

Snapshot – a numerical value of a scalar or vector valued observable at a specific instance in time. Explicit knowledge of the mappings $\mathbf{F}$ or $\mathbf{T}$ may not be available.

For example, snapshots may be obtained as/by

- high speed camera recording of a combustion process in a turbine
- new cases of covid 19 infections, reported daily
- wind tunnel measurements
- numerical simulation of (1) represented by (4), (5), (6), where we can feed an initial s_0 to a software tool (representing $\mathbf{T}$, or its linearization through a numerical scheme encoded in a software toolbox) to obtain a Krylov sequence: $\mathbf{f}(\mathbf{s}_0) = (\mathcal{U}_d^0 \mathbf{f})(\mathbf{s}_0)$,

  $$\mathbf{f}(\mathbf{s}_1) = (\mathcal{U}_d \mathbf{f})(\mathbf{s}_0),\ \mathbf{f}(\mathbf{s}_2) = (\mathcal{U}_d^2 \mathbf{f})(\mathbf{s}_0), \ldots, \mathbf{f}(\mathbf{s}_{M+1}) = (\mathcal{U}_d^{M+1} \mathbf{f})(\mathbf{s}_0),$$

  where $\mathbf{f} = (f_1, \ldots, f_d)^T$ is a vector valued $(d > 1)$ observable with the action of $\mathcal{U}_d$ defined component-wise.

Data driven framework – snapshots. Examples.

Snapshot matrix $\mathbf{S}$ with columns $\mathbf{f}(\mathbf{s}_0), \mathbf{f}(\mathbf{s}_{k+1}) = (\mathcal{U}_d\mathbf{f})(\mathbf{s}_k)$, $\mathbf{s}_{k+1} = \mathbf{T}(\mathbf{s}_k)$:

$$\mathbf{S} = \begin{pmatrix} \mathbf{f}(\mathbf{s}_0) & \mathbf{f}(\mathbf{s}_1) & \ldots & \mathbf{f}(\mathbf{s}_M) & \mathbf{f}(\mathbf{s}_{M+1}) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{s}_0) & f_1(\mathbf{s}_1) & \ldots & f_1(\mathbf{s}_M) & f_1(\mathbf{s}_{M+1}) \\ f_2(\mathbf{s}_0) & f_2(\mathbf{s}_1) & \ldots & f_2(\mathbf{s}_M) & f_2(\mathbf{s}_{M+1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_d(\mathbf{s}_0) & f_d(\mathbf{s}_1) & \ldots & f_d(\mathbf{s}_M) & f_d(\mathbf{s}_{M+1}) \end{pmatrix} \in \mathbb{C}^{d\times(M+2)},$$

(i) The snapshots are generated by a nonlinear system.
(ii) The snapshots are a Krylov sequence $\mathbf{f}, \mathcal{U}_d\mathbf{f}, \mathcal{U}_d^2\mathbf{f}, \ldots$, driven by the linear operator $\mathcal{U}_d$ and evaluated along a trajectory initialized at $\mathbf{s}_0$.

It makes sense to find a matrix $\mathbb{A} \in \mathbb{C}^{d\times d}$ such that

$$\mathbb{A}\mathbf{f}(\mathbf{s}_k) = (\mathcal{U}_d\mathbf{f})(\mathbf{s}_k) = \begin{pmatrix} (\mathcal{U}f_1)(\mathbf{s}_k) \\ \vdots \\ (\mathcal{U}f_d)(\mathbf{s}_k) \end{pmatrix} = \mathbf{f}(\mathbf{T}(\mathbf{s}_k)), \quad k = 0, \ldots, M. \tag{8}$$

Thus, if we set $\mathbf{X} = \mathbf{S}(1:d, 1:M+1)$, $\mathbf{Y} = \mathbf{S}(1:d, 2:M+2)$, then such an $\mathbb{A}$ would satisfy $\mathbf{Y} = \mathbb{A}\mathbf{X}$, and this could be extended linearly to the span of the columns of $\mathbf{X}$ by $\mathbb{A}(\mathbf{X}v) = \mathbf{Y}v$, $v \in \mathbb{C}^{M+1}$. The action of $\mathbb{A}$ outside the column space of $\mathbf{X}$ is not specified by the available data.

The DMD matrix $\mathbb{A}$

Flexible: can use many trajectories

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

Existence and uniqueness of $\mathbb{A}$: $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$

Depending on the parameters d and M, the matrices $\mathbf{X}$, $\mathbf{Y}$ can be square, tall, or wide. Then, we can search for a linear transformation $\mathbb{A}$ such that $\mathbf{Y} = \mathbb{A}\mathbf{X}$. Such an $\mathbb{A}$ may not exist.
However, we can always define a particular matrix $\mathbb{A}$ which minimizes $\|\mathbf{Y} - \mathbb{A}\mathbf{X}\|_F$. Clearly, if $\mathbf{X}^T$ has a nontrivial null-space, $\mathbb{A}$ is not unique; we can choose B so that $B\mathbf{X} = \mathbf{0}$ and thus $(\mathbb{A} + B)\mathbf{X} = \mathbb{A}\mathbf{X}$. An additional condition of minimality of $\|\mathbb{A}\|_F$ yields the well known solution $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$, expressed using the Moore-Penrose pseudoinverse $\mathbf{X}^{\dagger}$ of $\mathbf{X}$.

Finite dimensional compression of $\mathcal{U}$

Read the information in the snapshots matrix row-wise:

$$\widehat{\mathbf{S}}(1:M+1,1:d) = \begin{pmatrix} f_1(\mathbf{s}_0) & f_2(\mathbf{s}_0) & f_3(\mathbf{s}_0) & \ldots & f_d(\mathbf{s}_0) \\ f_1(\mathbf{s}_1) & f_2(\mathbf{s}_1) & f_3(\mathbf{s}_1) & \ldots & f_d(\mathbf{s}_1) \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ f_1(\mathbf{s}_M) & f_2(\mathbf{s}_M) & f_3(\mathbf{s}_M) & \ldots & f_d(\mathbf{s}_M) \end{pmatrix} = \mathbf{X}^T,$$

$$\widehat{\mathbf{S}}(2:M+2,1:d) = \begin{pmatrix} f_1(\mathbf{T}(\mathbf{s}_0)) & f_2(\mathbf{T}(\mathbf{s}_0)) & f_3(\mathbf{T}(\mathbf{s}_0)) & \ldots & f_d(\mathbf{T}(\mathbf{s}_0)) \\ f_1(\mathbf{T}(\mathbf{s}_1)) & f_2(\mathbf{T}(\mathbf{s}_1)) & f_3(\mathbf{T}(\mathbf{s}_1)) & \ldots & f_d(\mathbf{T}(\mathbf{s}_1)) \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ f_1(\mathbf{T}(\mathbf{s}_M)) & f_2(\mathbf{T}(\mathbf{s}_M)) & f_3(\mathbf{T}(\mathbf{s}_M)) & \ldots & f_d(\mathbf{T}(\mathbf{s}_M)) \end{pmatrix} = \mathbf{Y}^T,$$

Consider the action of $\mathcal{U}$ on the space $\mathcal{F}_{\mathcal{D}}$ spanned by the dictionary of scalar functions $\mathcal{D} = \{f_1, \ldots, f_d\}$. That is, we seek a matrix representation $\mathbb{U}$ of the compression $\mathbf{\Psi}_{\mathcal{F}_{\mathcal{D}}}\mathcal{U}_{|\mathcal{F}_{\mathcal{D}}} : \mathcal{F}_{\mathcal{D}} \longrightarrow \mathcal{F}_{\mathcal{D}}$, where $\mathbf{\Psi}_{\mathcal{F}_{\mathcal{D}}}$ is a suitable projection with the range $\mathcal{F}_{\mathcal{D}}$. This is the standard construction: we need a representation of $\mathcal{U}f_i$ of the form

$$(\mathcal{U}f_i)(s) = f_i(\mathbf{T}(s)) = \sum_{j=1}^{d} \mathbf{u}_{ji} f_j(s) + \rho_i(s), \quad i = 1, \ldots, d, \quad s \in \mathcal{X}. \qquad (9)$$

Finite dimensional compression of $\mathcal{U}$

Given limited information, the projection is feasible only in the discrete (algebraic) sense: we can define the matrix $\mathbb{U} = (\mathbf{u}_{ji}) \in \mathbb{C}^{d\times d}$ column-wise by minimizing the residual $\rho_i(s)$ in

$$(\mathcal{U} f_i)(s) = f_i(\mathbf{T}(s)) = \sum_{j=1}^{d} \mathbf{u}_{ji} f_j(s) + \rho_i(s), \quad i = 1, \ldots, d, \quad s \in \mathcal{X}$$

over the states $s = \mathbf{s}_k$, using the values

$$(\mathcal{U} f_i)(\mathbf{s}_k) = f_i(\mathbf{T}(\mathbf{s}_k)), \quad i = 1, \ldots, d; \quad k = 0, \ldots, M. \tag{10}$$

To that end, write the least squares residual

$$\frac{1}{M+1} \sum_{k=0}^{M} |\rho_i(\mathbf{s}_k)|^2 = \frac{1}{M+1} \sum_{k=0}^{M} |\sum_{j=1}^{d} \mathbf{u}_{ji} f_j(\mathbf{s}_k) - f_i(\mathbf{T}(\mathbf{s}_k))|^2, \tag{11}$$

which is the L^2 residual with respect to the empirical measure defined as the sum of the Dirac measures concentrated at the $\mathbf{s}_k$'s, $\boldsymbol{\delta}_{M+1} = (1/(M+1)) \sum_{k=0}^{M} \boldsymbol{\delta}_{\mathbf{s}_k}$.

Compression of $\mathcal{U}$ – matrix representation

Hence, the columns of the matrix representation $\mathbb{U}$ are defined as the solutions of the least squares problems

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

for $i = 1, \dots, d$; $\gamma_M = 1/(M+1)$. The solutions of the above algebraic least squares problems for all $i = 1, \dots, d$ are compactly written as the matrix $\mathbb{U} \in \mathbb{C}^{d \times d}$ that minimizes $\|\mathbf{X}^T \mathbb{U} - \mathbf{Y}^T\|_F$. Recall that we seek an $\mathbb{A}$ such that $\mathbb{A}\mathbf{X} = \mathbf{Y}$, i.e. $\|\mathbb{A}\mathbf{X} - \mathbf{Y}\|_F = \|\mathbf{X}^T \mathbb{A}^T - \mathbf{Y}^T\|_F \to \min$. Hence,

$$\mathbb{U} = (\mathbf{X}^T)^\dagger \mathbf{Y}^T \equiv (\mathbf{Y}\mathbf{X}^\dagger)^T = \mathbb{A}^T, \qquad (12)$$

and the action of $\mathcal{U}$ can be represented, using (9), as

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

Assume $\mathbb{U}$ is diagonalizable: $\mathbb{U} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$, with $\Lambda = \mathrm{diag}(\lambda_i)_{i=1}^d$, $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_d)$, $\mathbb{U}\mathbf{q}_i = \lambda_i \mathbf{q}_i$.

Compression of $\mathcal{U}$ – approximate eigenfunctions

For $s \in \mathcal{X}$,

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

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

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

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

$$\begin{pmatrix} \phi_1(\mathbf{s}_0) & \phi_2(\mathbf{s}_0) & \ldots & \phi_d(\mathbf{s}_0) \\ \phi_1(\mathbf{s}_1) & \phi_2(\mathbf{s}_1) & \ldots & \phi_d(\mathbf{s}_1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(\mathbf{s}_{M+1}) & \phi_2(\mathbf{s}_{M+1}) & \ldots & \phi_d(\mathbf{s}_{M+1}) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{s}_0) & f_2(\mathbf{s}_0) & \ldots & f_d(\mathbf{s}_0) \\ f_1(\mathbf{s}_1) & f_2(\mathbf{s}_1) & \ldots & f_d(\mathbf{s}_1) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(\mathbf{s}_{M+1}) & f_2(\mathbf{s}_{M+1}) & \ldots & f_d(\mathbf{s}_{M+1}) \end{pmatrix}\mathbf{Q} = \mathbf{S}^T\mathbf{Q}.$$

Koopman Mode Decomposition (KMD)

Let now $\mathbf{g}(s)^T = (g_1(s), \ldots, g_d(s))$ be a vector valued observable and let $\mathbf{g}(s)^T = (f_1(s), \ldots, f_d(s))\Gamma$, $\Gamma = (\gamma_{ji}) \in \mathbb{C}^{d\times d}$. (If $g_i = f_i$, then $\Gamma = \mathbb{I}_d$)

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

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

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

The Koopman mode decomposition (KMD) for $k = 0, 1, \ldots$ reads

$$(\mathcal{U}_d^k \mathbf{g})(s) = \begin{pmatrix} (\mathcal{U}^k g_1)(s) \\ \vdots \\ (\mathcal{U}^k g_d)(s) \end{pmatrix} \approx \sum_{i=1}^{d} \mathbf{z}_i \phi_i(s) \lambda_i^k. \tag{13}$$

$\mathbb{A}^T = \mathbb{U} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$ implies $\mathbb{A}\mathbf{Q}^{-T} = \mathbf{Q}^{-T}\Lambda$, i.e. the columns of $\mathbf{Q}^{-T}$ are the (right) eigenvectors of $\mathbb{A}$. Hence, to compute the KMD, we start with computing the eigenvalues and eigenvectors of $\mathbb{A}$.

Data driven spectral analysis

Suppose we are given a sequence of snapshots $\mathbf{f}_i \in \mathbb{C}^n$ of an underlying dynamics, that are driven by an unaccessible *black box* linear operator $\mathbb{A}$;

$$\mathbf{f}_{i+1} = \mathbb{A}\mathbf{f}_i, \;\; i = 1, \ldots, m, \;\; m < n, \tag{14}$$

with some initial $\mathbf{f}_1$ and a time lag δt. No other information is available.

The two basic tasks of the Dynamic Mode Decomposition (DMD) are

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

$$\mathbb{A}z_j \approx \lambda_j z_j, \;\; \lambda_j = |\lambda_j| e^{\mathrm{i}\omega_j \delta t}, \;\; j = 1, \ldots, k; \;\; k \leq m, \tag{15}$$

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

$$\mathbf{f}_i \approx \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j \lambda_{\varsigma_j}^{i-1} \equiv \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j |\lambda_{\varsigma_j}|^{i-1} e^{\mathrm{i}\omega_{\varsigma_j}(i-1)\delta t}, \;\; i = 1, \ldots, m. \tag{16}$$

Data driven spectral analysis – deep connections to Koopman operator theory/applications

The decomposition of the snapshots (16) reveals dynamically relevant spatial structures, the z_{ς_j}'s, that evolve with amplitudes and frequencies encoded in the corresponding λ_{ς_j}'s. It is desirable to have small number ℓ of the most important modes $z_{\varsigma_1}, \ldots, z_{\varsigma_\ell}$, $\varsigma_j \in \{1, \ldots, k\}$.

Such a sequence of snapshot (vectors of observables) can be obtained e.g. using black–boxed high performance ODE/PDE software, or e.g. by hi speed camera in an analysis of combustion instabilities in flame dynamics.

In a carefully designed framework with reach enough set of properly selected observables, the DMD can be considered as a finite dimensional spectral approximation of the Koopman operator associated with the dynamics under study [Arbabi+Mezić]. This deep theoretical connection gives the DMD a pivotal role in computational study of complex phenomena in fluid dynamics, see e.g. [Rowley], [Williams].

Data driven spectral analysis - applications and software implementations

Other successful applications of DMD include e.g. aeroacoustics [Lele+Nichols], affective computing (analysis of videos for human emotion recognition [Cat Le Ngo+et al.]), robotics (filtering external perturbation using DMD based prediction [Berger+et al.]), algorithmic trading on financial markets [Mann+Kutz], analysis of infectious disease spread [Proctor+Eckhoff], neuroscience [Brunon+et al.] – just to name a few.

Computational aspects (for software implementation):

- matrix multiplication and other simple matrix/vector operations
- SVD decomposition, Moore-Penrose pseudo-inverse, least squares
- eigenvalues and eigenvectors of matrices of moderate dimensions

All necessary software implementations available in state of the art packages such as Matlab, Python (NumPy, SciPy) – LAPACK based.

So is there anything left to do for a numerical analyst?

Tool: Krylov subspaces

- For $i = 1, 2, \ldots, m$, define the Krylov matrices

$$\mathbf{X}_i = \begin{pmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \ldots & \mathbf{f}_{i-1} & \mathbf{f}_i \end{pmatrix}, \;\; \mathbf{Y}_i = \begin{pmatrix} \mathbf{f}_2 & \mathbf{f}_3 & \ldots & \mathbf{f}_i & \mathbf{f}_{i+1} \end{pmatrix} \equiv \mathbb{A}\mathbf{X}_i,$$

  and the corresponding Krylov subspaces $\mathcal{X}_i = \mathrm{range}(\mathbf{X}_i) \subset \mathbb{C}^n$.
- Assume that at the index m, $\mathbf{X}_m$ is of full column rank. This implies

$$\mathcal{X}_1 \subsetneq \mathcal{X}_2 \subsetneq \cdots \subsetneq \mathcal{X}_i \subsetneq \mathcal{X}_{i+1} \subsetneq \cdots \subsetneq \mathcal{X}_m \subsetneq \cdots \subsetneq \mathcal{X}_\ell = \mathcal{X}_{\ell+1}, \;\; ,$$

  i.e. $\dim(\mathcal{X}_i) = i$ for $i = 1, \ldots, m$, and there must be the smallest saturation index ℓ at which $\mathcal{X}_\ell = \mathcal{X}_{\ell+1}$.
- $\mathbb{A}\mathcal{X}_\ell \subseteq \mathcal{X}_\ell$, It is well known that then $\mathcal{X}_\ell$ is the smallest $\mathbb{A}$-invariant subspace that contains $\mathbf{f}_1$.
- The action of $\mathbb{A}$ on $\mathcal{X}_m$ is known, $\mathbb{A}(\mathbf{X}_m v) = \mathbf{Y}_m v$ for any $v \in \mathbb{C}^m$. Hence, useful spectral information can be obtained using the computable restriction $\mathbf{P}_{\mathcal{X}_m}\mathbb{A}\big|_{\mathcal{X}_m}$, that is, the Ritz values and vectors extracted using the Rayleigh quotient of $\mathbb{A}$ with respect to $\mathcal{X}_m$.

Tool: Krylov decomposition and Rayleigh-Ritz extraction

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

$$\|\mathbf{f}_{m+1} - \mathbf{X}_m c\|_2 \longrightarrow \min_c \tag{1}$$

and let $r_{m+1} = \mathbf{f}_{m+1} - \mathbf{X}_m c$ be the corresponding residual. Recall that, by virtue of the theorem of projection, $\mathbf{X}_m c = \mathbf{P}_{\mathcal{X}_m}\mathbf{f}_{m+1}$ and that r_{m+1} is orthogonal to the range of $\mathbf{X}_m$, $\mathbf{X}_m^* r_{m+1} = 0$.

- Let $E_{m+1} = r_{m+1} e_m^T$, $e_m = \begin{pmatrix}0, \ldots, 0, 1\end{pmatrix}^T$. The Krylov decomposition reads:

$$\mathbb{A}\mathbf{X}_m = \mathbf{X}_m C_m + E_{m+1}, \quad C_m = \begin{pmatrix} 0 & 0 & \ldots & 0 & c_1 \\ 1 & 0 & \ldots & 0 & c_2 \\ 0 & 1 & \ldots & 0 & c_3 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & c_m \end{pmatrix}.$$

Rayleigh–Ritz extraction – basic properties

1. $C_m = (\mathbf{X}_m^* \mathbf{X}_m)^{-1} (\mathbf{X}_m^* \mathbb{A} \mathbf{X}_m) \equiv \mathbf{X}_m^{\dagger} \mathbb{A} \mathbf{X}_m = (\mathbf{X}_m^* \mathbf{X}_m)^{-1} (\mathbf{X}_m^* \mathbf{Y}_m)$ is the Rayleigh quotient, i.e. the matrix representation of $\mathbf{P}_{\mathcal{X}_m} \mathbb{A}\big|_{\mathcal{X}_m}$
2. If $r_{m+1} = 0$ (and thus $E_{m+1} = 0$ and $m = \ell$) then $\mathbb{A}\mathbf{X}_m = \mathbf{X}_m C_m$ and each eigenpair $C_m w = \lambda w$ of C_m yields an eigenpair of $\mathbb{A}$, $\mathbb{A}(\mathbf{X}_m w) = \lambda(\mathbf{X}_m w)$.
3. If $r_{m+1} \neq 0$, then $(\lambda, z \equiv \mathbf{X}_m w)$ is an approximate eigenpair, $\mathbb{A}(\mathbf{X}_m w) = \lambda(\mathbf{X}_m w) + r_{m+1}(e_m^T w)$, i.e. $\mathbb{A}z = \lambda z + r_{m+1}(e_m^T w)$. The Ritz pair (λ, z) is acceptable if the residual

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

is sufficiently small. It holds that $z^* r_{m+1} = 0$, and

$$\lambda = \frac{z^* \mathbb{A} z}{z^* z} = \operatorname{argmin}_{\zeta \in \mathbb{C}} \|\mathbb{A}z - \zeta z\|_2$$

(λz is the orthogonal projection of $\mathbb{A}z$ onto the span of z).

Beautiful structure and bad news

The spectral decomposition of C_m has beautiful structure. Assume for simplicity that the eigenvalues λ_i, $i = 1, \ldots, m$, are algebraically simple. It is easily checked that the spectral decomposition of C_m reads

$$C_m = \mathbb{V}_m^{-1} \Lambda_m \mathbb{V}_m, \;\text{ where }\; \Lambda_m = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix}, \;\; \mathbb{V}_m = \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \ldots & \lambda_2^{m-1} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & \lambda_m & \ldots & \lambda_m^{m-1} \end{pmatrix}.$$

The Ritz vectors are the columns of $Z_m = \mathbf{X}_m \mathbb{V}_m^{-1}$.

Bad news: The Vandermonde matrix $\mathbb{V}_m$ is ill-conditioned!

The condition number $\kappa_2(\mathbb{V}_m) \equiv \|\mathbb{V}_m\|_2 \|\mathbb{V}_m^{-1}\|_2$ of any 100×100 real Vandermonde matrix is larger than $3 \cdot 10^{28}$,
$(\kappa_2(\mathbb{V}_m) \geq 2^{m-2}/\sqrt{m}, m = 100$, [Gautschi]).

Better: Schmid's DMD – compute the Rayleigh quotient using a POD (truncated SVD) basis of $\mathbf{X}_m$.

Schmid's DMD

To avoid the ill-conditioning, Schmid used the thin truncated SVD $\mathbf{X}_m = U\Sigma V^* \approx U_k\Sigma_k V_k^*$, 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 $\Sigma_k = \mathrm{diag}(\sigma_i)_{i=1}^k$ contains the largest k singular values of $\mathbf{X}_m$. In brief, U_k is the POD basis for the snapshots $\mathbf{f}_1,\ldots,\mathbf{f}_m$. Since

$$\mathbf{Y}_m = \mathbb{A}\mathbf{X}_m \approx \mathbb{A}U_k\Sigma_k V_k^*, \text{ and } \mathbb{A}U_k = \mathbf{Y}_m V_k \Sigma_k^{-1}, \tag{2}$$

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

$$S_k = U_k^*\mathbf{Y}_m V_k \Sigma_k^{-1}, \tag{3}$$

which is suitable for data driven setting because it does not use $\mathbb{A}$ explicitly. Clearly, (2, 3) only require that $\mathbf{Y}_m = \mathbb{A}\mathbf{X}_m$; it is not necessary that $\mathbf{Y}_m$ is shifted $\mathbf{X}_m$. Each eigenpair (λ, w) of S_k generates the corresponding Ritz pair $(\lambda, U_k w)$ for $\mathbb{A}$.

Schmid's DMD

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

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

Data driven (computable) residual

Not all computed Ritz pairs will provide good approximations of eigenpairs of the underlying $\mathbb{A}$, 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, the DMD Algorithm can be enhanced with residual computation, without using $\mathbb{A}$ explicitly.

Proposition

For the Ritz pairs $(\lambda_i, Z_k(:,i) \equiv U_k W_k(:,i))$, $i = 1, \ldots, k$, computed in the DMD Algorithm, the residual norms can be computed as follows:

$$r_k(i) = \|\mathbb{A} Z_k(:,i) - \lambda_i Z_k(:,i)\|_2 = \|(\mathbf{Y}_m V_k \Sigma_k^{-1}) W_k(:,i) - \lambda_i Z_k(:,i)\|_2. \tag{4}$$

Further, if $\mathbb{A} = S \mathrm{diag}(\alpha_i)_{i=1}^n S^{-1}$, then $\min_{\alpha_j} |\lambda_i - \alpha_j| \le \kappa_2(S) r_k(i)$ (by the Bauer–Fike Theorem).

Data driven (computable) residual

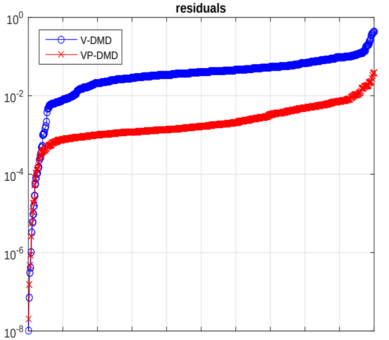
Example

test The well studied and understood model of laminar flow around a cylinder is based on the two-dimensional incompressible Navier-Stokes equations

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v}\cdot\nabla)\mathbf{v} + \nu\Delta\mathbf{v} - \frac{1}{\rho}\nabla p, \quad \nabla\cdot\mathbf{v} = 0, \tag{5}$$

where $\mathbf{v} = (v_x, v_y)$ is velocity field, p is pressure, ρ is fluid density and ν is kinematic viscosity. The flow is characterized by the Reynolds number $\mathfrak{Re} = v^* D/\nu$ 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 [Bagheri], [Glaz+et al.]; for a more in depth description of the Koopman analysis of fluid flow we refer to [Mezić], [Rowley].

Data driven (computable) residual



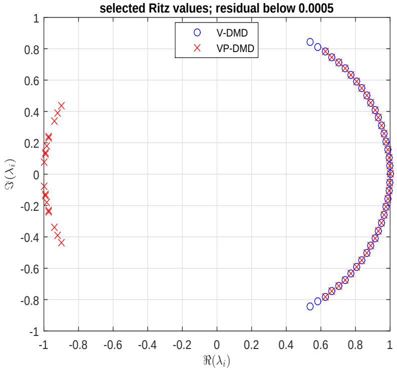


Figure: Left: Comparison of the residuals of the Ritz pairs computed by the DMD_RRR Algorithm with velocities as observables (V-DMD, circles ∘) and with both velocities and pressures (VP-DMD, crosses, ×). Right: Selected Ritz values with velocities as observables (∘) and with both velocities and pressures (×).

Refined Ritz vectors

The Ritz vectors are not optimal eigenvectors approximations from a given subspace $\mathcal{U}_k = \mathrm{range}(U_k)$. Hence, for a computed Ritz value λ, instead of the associated Ritz vector, we can choose a vector $z \in \mathcal{U}_k$ that minimizes the residual. From the variational characterization of the singular values, it follows that

$$\begin{aligned}\min_{z\in\mathcal{U}_k\setminus\{0\}} \frac{\|\mathbb{A}z - \lambda z\|_2}{\|z\|_2} &= \min_{w\neq 0} \frac{\|\mathbb{A}U_k w - \lambda U_k w\|_2}{\|U_k v\|_2} \\ &= \min_{\|w\|_2=1} \|(\mathbb{A}U_k - \lambda U_k)w\|_2 = \sigma_{\min}(\mathbb{A}U_k - \lambda U_k),\end{aligned}$$

where $\sigma_{\min}(\cdot)$ denotes the smallest singular value of a matrix, and the minimum is attained at the right singular vector w_λ corresponding to $\sigma_\lambda \equiv \sigma_{\min}(\mathbb{A}U_k - \lambda U_k)$. As a result, the refined Ritz vector corresponding to λ is $U_k w_\lambda$ and the optimal residual is σ_λ. Detailed analysis by Jia.

Data driven refinement of Ritz vectors

The minimization of the residual can be replaced with computing the smallest singular value with the corresponding right singular vector of $B_k - \lambda U_k$, where $B_k \equiv \mathbb{A} U_k = \mathbf{Y}_m V_k \Sigma_k^{-1}$. Compute the QRF

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

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

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

Theorem

Let for the Ritz value $\lambda = \lambda_i$, w_{λ_i} denote the right singular vector of the smallest singular value σ_{λ_i} of the matrix R_{λ_i}. Then $z = z_{\lambda_i} \equiv U_k w_{\lambda_i}$ minimizes the residual, whose minimal value equals $\sigma_{\lambda_i} = \|R_{\lambda_i} w_{\lambda_i}\|_2$.

Enhanced DMD Algorithm

```
$[Z_k, \Lambda_k, r_k, \rho_k] = \mathrm{DMD\_RRRR}(\mathbf{X}_m, \mathbf{Y}_m; \epsilon)$ {*Refined Rayleigh-Ritz DMD*}
 1: $D_x = \mathrm{diag}(\|\mathbf{X}_m(:,i)\|_2)_{i=1}^m$; $\mathbf{X}_m^{(1)} = \mathbf{X}_m D_x^\dagger$; $\mathbf{Y}_m^{(1)} = \mathbf{Y}_m D_x^\dagger$
 2: $[U, \Sigma, V] = svd(\mathbf{X}_m^{(1)})$ ; numerical rank: $k = \max\{i \ : \ \sigma_i \geq \sigma_1 \epsilon\}$.
 3: Set $U_k = U(:, 1:k)$, $V_k = V(:, 1:k)$, $\Sigma_k = \Sigma(1:k, 1:k)$
 4: $B_k = \mathbf{Y}_m^{(1)}(V_k \Sigma_k^{-1})$; {*Schmid's formula for* $\mathbb{A}U_k$}
 5: $[Q, R] = qr(\begin{pmatrix} U_k, & B_k \end{pmatrix})$; {*The thin QR factorization*}
 6: $S_k = \mathrm{diag}(\overline{R_{ii}})_{i=1}^k R(1:k, k+1:2k)$ {$S_k = U_k^* \mathbb{A} U_k$}
 7: $\Lambda_k = \mathrm{eig}(S_k)$ {$\Lambda_k = \mathrm{diag}(\lambda_i)_{i=1}^k$; *Ritz values, i.e. eigenvalues of* $S_k$}
 8: **for** $i = 1, \ldots, k$ **do**
 9:     $[\sigma_{\lambda_i}, w_{\lambda_i}] = svd_{\min}(\begin{pmatrix} R(1:k,k+1:2k) - \lambda_i R(1:k,1:k) \\ R(k+1:2k,k+1:2k) \end{pmatrix})$;
10:     $W_k(:, i) = w_{\lambda_i}$; $r_k(i) = \sigma_{\lambda_i}$ {*Optimal residual,* $\sigma_{\lambda_i} = \|R_{\lambda_i} w_{\lambda_i}\|_2$}
11:     $\rho_k(i) = w_{\lambda_i}^* S_k w_{\lambda_i}$ {*Rayleigh quotient,* $\rho_k(i) = (U_k w_{\lambda_i})^* \mathbb{A}(U_k w_{\lambda_i})$}
12: **end for**
13: $Z_k = U_k W_k$ {*Refined Ritz vectors*}
```

Residuals of refined selected pairs

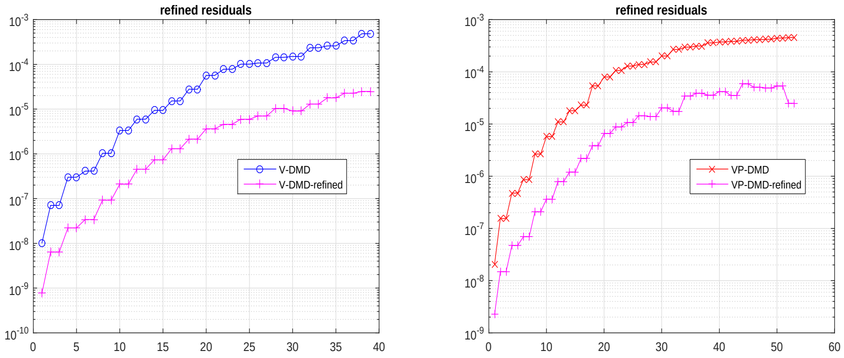


Figure: Comparison of the refined residuals of the Ritz pairs computed by the DMD_RRR Algorithm with velocities as observables (top 39 pairs in V-DMD, circles ∘) and with both velocities and pressures (top 53 pairs in VP-DMD, crosses, ×). The noticeable staircase structure on the graphs corresponds to complex conjugate Ritz pairs.

A synthetic example

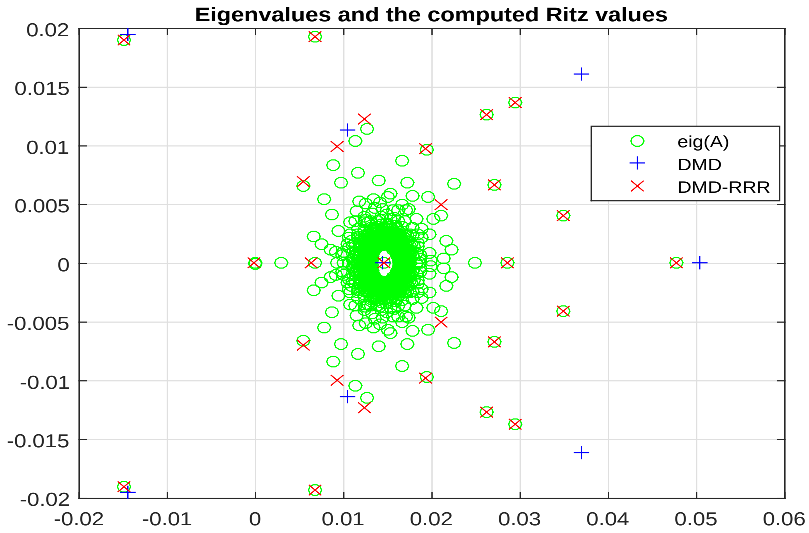
Goal: DMD black-box software

The main goal of the modifications of the DMD algorithm 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.

Example (A case study)

The test matrix is generated as $A = \mathbf{e}^{-B^{-1}}$ where B is pseudo-random with entries uniformly distributed in $[0, 1]$, and then $\mathbb{A} = A/\|A\|_2$. 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, see e.g. Mohr & Mezić 2014.

Accuracy of the computed Ritz values



Comparing residuals

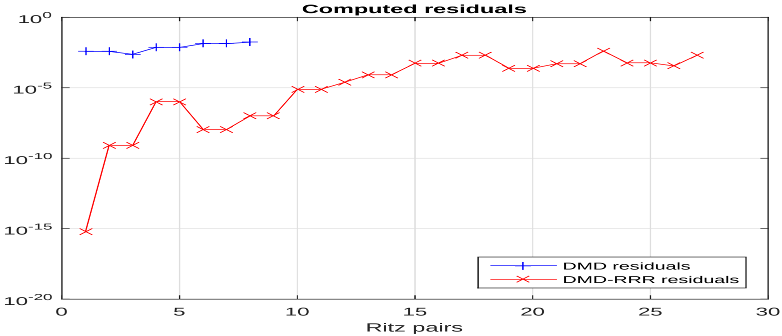
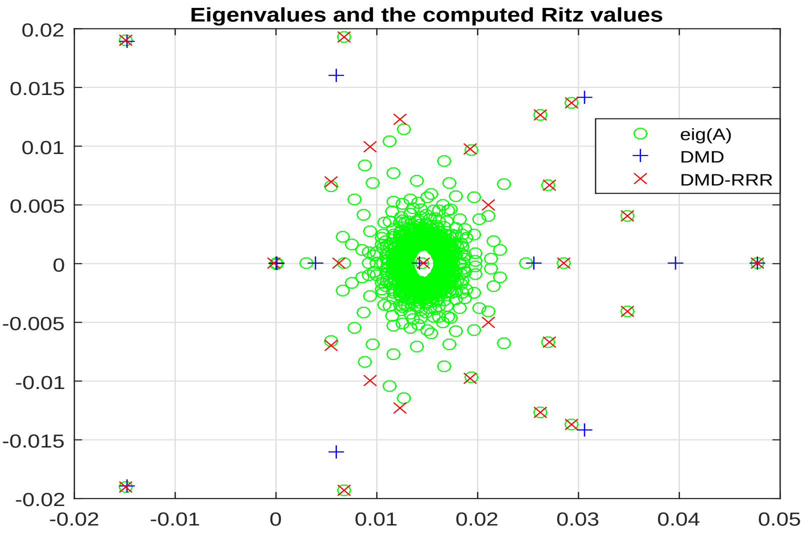
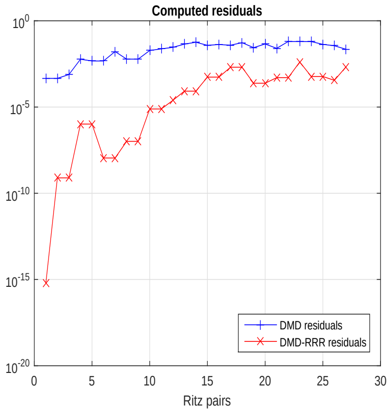


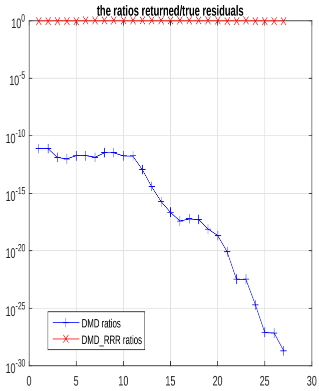
Figure: Comparison of the residuals of the Ritz pairs computed by the DMD Algorithm (pluses $+$) and the DMD_RRR Algorithm (crosses, $\times$), with the same threshold in the truncation criterion for determining the numerical rank.

Ritz values wit $k = 27$ (hard coded)



Residuals wit $k = 27$ (hard coded)





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

Singular values of $\mathbf{X}_m$ computed three times

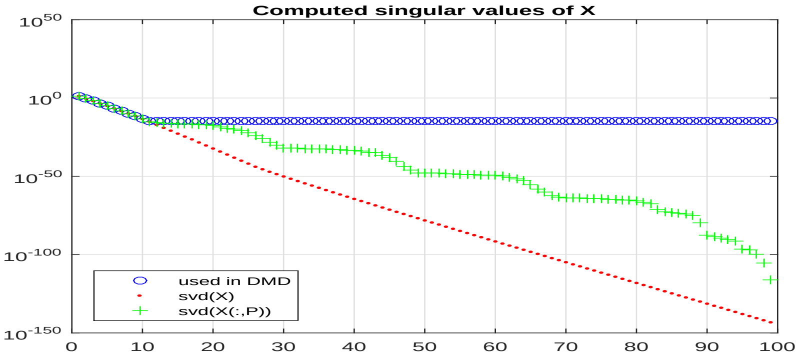


Figure: The blue circles ($\circ$) are the values used in the DMD Algorithm and are computed as $[U, \Sigma, V] = \texttt{svd}(\mathbf{X}_m, 'econ')$. The red dots ($\cdot$) are computed as $\Sigma = \texttt{svd}(\mathbf{X}_m)$, and the pluses ($+$) are the results of $\Sigma = \texttt{svd}(\mathbf{X}_m(:, P))$, where P is randomly generated permutation.

Dicussion on the SVD

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 $[U, S, V] = \mathtt{svd}(\mathbf{X}_m, 'econ')$. It is very likely that the full SVD, including the singular vectors, is computed using the divide and conquer algorithm, `xGESDD()` in LAPACK.
- For computing only the singular values $S = \mathtt{svd}(X)$ calls the QR SVD, `xGESVD()` in LAPACK.

Note that the same fast `xGESDD()` subroutine is (to our best knowledge) under the hood of the Python function `numpy.linalg.svd`.
Numerical robustness of both `xGESVD()`, `xGESDD()` depends on $\kappa_2(\mathbf{X}_m)$, and if one does not take advantage of the fact that scaling is allowed, the problems illustrated in this example are likely to happen.
Better: Jacobi SVD (`xGEJSV()`, `xGESVJ()` in LAPACK, **Z.D.** 2009.) and preconditioned QR (`xGESVDQ()`, LAPACK, **Z.D.** 2018.).

... rewind ...◀◀... Beautiful structure and bad news

The spectral decomposition of C_m has beautiful structure. Assume for simplicity that the eigenvalues λ_i, $i = 1, \ldots, m$, are algebraically simple. It is easily checked that the spectral decomposition of C_m reads

$$C_m = \mathbb{V}_m^{-1} \Lambda_m \mathbb{V}_m, \ \text{ where } \ \Lambda_m = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix}, \ \mathbb{V}_m = \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \ldots & \lambda_2^{m-1} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & \lambda_m & \ldots & \lambda_m^{m-1} \end{pmatrix}.$$

The Ritz vectors are the columns of $Z_m = \mathbf{X}_m \mathbb{V}_m^{-1}$.

Bad news: The Vandermonde matrix $\mathbb{V}_m$ is ill-conditioned!

The condition number $\kappa_2(\mathbb{V}_m) \equiv \|\mathbb{V}_m\|_2 \|\mathbb{V}_m^{-1}\|_2$ of any 100×100 real Vandermonde matrix is larger than $3 \cdot 10^{28}$,
$(\kappa_2(\mathbb{V}_m) \geq 2^{m-2}/\sqrt{m}, m = 100$, [Gautschi]).

Better: Schmid's DMD – compute the Rayleigh quotient using a POD (truncated SVD) basis of $\mathbf{X}_m$.

Vandermonde x DFT = Cauchy; DFT = Vandermonde

Let $\mathbb{F}$ denote the DFT matrix, $\mathbb{F}_{ij} = \omega^{(i-1)(j-1)}/\sqrt{m}$, where $\omega = \mathbf{e}^{2\pi\mathfrak{i}/m}$, $\mathfrak{i} = \sqrt{-1}$. Now, recall that DFT transforms Vandermonde into Cauchy matrices as follows: If $\lambda_i^m \neq 1$, then

$$(\mathbb{V}_m\mathbb{F})_{ij} = \left[\frac{\lambda_i^m - 1}{\sqrt{m}}\right]\left[\frac{1}{\lambda_i - \omega^{1-j}}\right]\left[\omega^{1-j}\right] \equiv (\mathcal{D}_1)_{ii}\, \mathcal{C}_{ij}\, (\mathcal{D}_2)_{jj},\ 1 \leq j \leq m. \tag{1}$$

If $\lambda_i = \omega^{1-j}$ for some index j, write $\lambda_i^m - 1 = \prod_{k=1}^{m}(\lambda_i - \omega^{1-k})$ and replace (1) with the equivalent formula for the i-th row

$$(\mathbb{V}_m\mathbb{F})_{ij} = \underbrace{\frac{1}{\sqrt{m}}}_{(\mathcal{D}_1)_{ii}} \prod_{\substack{k=1 \\ k\neq j}}^{m}(\lambda_i - \omega^{1-k})\, \underbrace{\omega^{1-j}}_{(\mathcal{D}_2)_{jj}},\ \ (\mathbb{V}_m\mathbb{F})_{ik} = 0 \text{ for } k \neq j. \tag{2}$$

This is the starting point for accurate computation of the SVD of $\mathbb{V}_m$ [Demmel]. We use it for $Z_m = \mathbf{X}_m\mathbb{V}_m^{-1} \equiv (\mathbf{X}_m\mathbb{F})(\mathbb{V}_m\mathbb{F})^{-1}$.

How this transforms $\mathbb{A}\mathbf{X}_m = \mathbf{X}_m C_m + r_{m+1}e_m^T$?

It is interesting to see how this transformation $\mathbb{V}_m \mapsto \mathbb{V}_m\mathbb{F}$ fits the framework of the Krylov decomposition $\mathbb{A}\mathbf{X}_m = \mathbf{X}_m C_m + r_{m+1}e_m^T$, where $C_m = \mathbb{V}_m^{-1}\Lambda_m\mathbb{V}_m$. Post-multiply this with $\mathbb{F}$ to obtain

$$\mathbb{A}(\mathbf{X}_m\mathbb{F}) = (\mathbf{X}_m\mathbb{F})\mathbb{F}^*(\mathbb{V}_m^{-1}\Lambda_m\mathbb{V}_m)\mathbb{F} + r_{m+1}e_m^T\mathbb{F} \qquad (3)$$

and then, using $\mathbb{V}_m\mathbb{F} = \mathcal{D}_1\mathcal{C}\mathcal{D}_2$,
$\mathbb{F}^*C_m\mathbb{F} = \mathbb{F}^*(\mathbb{V}_m^{-1}\Lambda_m\mathbb{V}_m)\mathbb{F} = \mathcal{D}_2^*\mathcal{C}^{-1}\mathcal{D}_1^{-1}\Lambda_m\mathcal{D}_1\mathcal{C}\mathcal{D}_2 = \mathcal{D}_2^*\mathcal{C}^{-1}\Lambda_m\mathcal{C}\mathcal{D}_2$ and

$$\begin{aligned} \mathbb{A}(\mathbf{X}_m\mathbb{F}) &= (\mathbf{X}_m\mathbb{F})((\mathcal{C}\mathcal{D}_2)^{-1}\Lambda_m(\mathcal{C}\mathcal{D}_2)) + r_{m+1}e_m^T\mathbb{F} \iff & (4)\\ \mathbb{A}(\mathbf{X}_m\mathbb{F}\mathcal{D}_2^*) &= (\mathbf{X}_m\mathbb{F}\mathcal{D}_2^*)(\mathcal{C}^{-1}\Lambda_m\mathcal{C}) + r_{m+1}e_m^T\mathbb{F}\mathcal{D}_2^*. & (5)\end{aligned}$$

If we think of each row $\mathbf{X}_m(i,:)$ as a time trajectory of the corresponding observable, then $\mathbf{X}_m(i,:)\mathbb{F}$ represents its image in the frequency domain, and (4, 5) is the corresponding Krylov decomposition. Possible insightful connections (?) to the Laskar algorithm [Laskar, Arbabi+Mezić] and (data centered) DMD and temporal DFT see [Chen+Tu+Rowley].

But κ_2(Vandermonde x DFT) = κ_2(Vandermonde) ?!

Note that the matrices $\mathcal{D}_1$, $\mathcal{C}$, $\mathcal{D}_2$ are given implicitly by the parameters λ_i (eigenvalues, available on input) and the m-th roots of unity $\zeta_j = \omega^{1-j}$, $j = 1, \ldots, m$ (easily precomputed to any desired precision and tabulated), so that the DFT $\mathbb{V}_m\mathbb{F}$ is not done by actually running an FFT.
It suffices to make a note that the λ_i's and the roots of unity are the parameters (original data) that define $\mathbb{V}_m\mathbb{F}$ as in (1), (2).

Besides nice matrix–theoretical connection, what is the gain?

Applying the DFT to $\mathbb{V}_m$ in order to avoid the ill–conditioning of $\mathbb{V}_m$ may seem a futile effort – since $\mathbb{F}$ is unitary, $\kappa_2(\mathcal{D}_1\mathcal{C}\mathcal{D}_2) = \kappa_2(\mathbb{V}_m\mathbb{F}) = \kappa_2(\mathbb{V}_m)$. Further, Cauchy matrices are also notoriously ill-conditioned, so, in essence, we have traded one badly conditioned structure to another one.

How bad it can be? What is the meaning of bad, ill–conditioned anyway? Is Cauchy matrix really badly conditioned?

Caveat ill-conditioning: Hilbert matrix example

Ill-conditioning is not always obvious in the sizes of its entries – the entries of the innocuous-looking 100×100 Hilbert matrix H_{100} range from $1/199 \approx 5.025 \cdot 10^{-3}$ to 1, and $\kappa_2(H_{100}) > 10^{\mathbf{150}}$.

condition number(condition number)=condition number [Higham]

```
>> cond(hilb(100))
ans = 4.6226e+19
```

If the computed/estimated condition number is above $1/\mathrm{eps}$ (in Matlab, `1/eps=4.503599627370496e+15`), it might be a severe underestimate. This may lead to an underestimate of extra precision needed to handle the ill–conditioning.

High acuracy numerical linear algebra :)

Accurate LU, QR, SVD of any Cauchy or Vandermonde matrix is feasible without higher precision arithmetic. Good algorithms are available!

SVD($D_1 \times \text{Cauchy} \times D_2$)

Given Cauchy matrix $\mathcal{C}$ and any two diagonal matrices D_1, D_2, the SVD of $G = D_1 \mathcal{C} D_2$ can be computed to nearly full precision as follows:

1. Compute the LDU, $P_1 G P_2 = LDU$ using explicit determinant based formulas to update the Schur complement [Demmel]. This is entry wise forward stable computation of L, D, U. Moreover, $\kappa(L)$, $\kappa(U)$ are moderate. (Small $\|\delta L\|/\|L\|$, $\|\delta U\|/\|U\|$, $|\delta D_{ii}|/|D_{ii}|$ is also OK) ($G = hilb(100)$, $\kappa_2(G) > \mathbf{10}^{150}$, $\kappa_2(L) = \kappa_2(U) \approx$ 72.24 (see Viswanath and Trefethen), $\kappa_2(D) \approx 2.32 \cdot \mathbf{10}^{149}$)
2. Compute the SVD of the product LDU using a Jacobi type SVD algorithm (product SVD [**Z.D.**]). The forward error is determined by $\max(\kappa(L), \kappa(U))$, independent of $\kappa_2(D)$. The backward errors $\|\Delta L\|/\|L\|$, $\|\Delta U\|/\|U\|$, $\Delta D_{ii}/D_{ii}$ are small.

The key is in forward stable reparametrization, so that the new representation is well-conditioned (<u>for particular algorithm</u>).

$$\begin{tikzcd}
 & & L \arrow[r, "\clubsuit"] & \widetilde{L} = L + \delta L \arrow[dr, "\blacktriangledown"] & \\
\boxed{\mathbb{V}_m} \arrow[r, Rightarrow, "\spadesuit"'] \arrow[d, "\blacklozenge"'] & \boxed{\boxed{\mathbb{V}_m \mathbb{F}}} \arrow[ur, "\blacktriangle"] \arrow[r, "\blacktriangle"] \arrow[dr, "\blacktriangle"] & D \arrow[r, "\clubsuit"] & \widetilde{D} = D + \delta D \arrow[r, "\blacktriangledown"] & \boxed{\Pi_1^T \widetilde{L}\widetilde{D}\widetilde{U}\Pi_2^T} \\
\mathbb{V}_m + E & & U \arrow[r, "\clubsuit"] & \widetilde{U} = U + \delta U \arrow[ur, "\blacktriangledown"] &
\end{tikzcd}$$

Figure: Legend: $\spadesuit$ = the DFT of $\mathbb{V}_m$ using the explicit formulas (1); $\blacktriangle$ = the forward stable pivoted LDU [Demmel]; $\clubsuit$ = forward errors in the computed factors $\widetilde{L}$, $\widetilde{D}$, $\widetilde{U}$; $\blacktriangledown$ = implicit representation of $\mathbb{V}_m\mathbb{F}$ as the product $\Pi_1^T \widetilde{L}\widetilde{D}\widetilde{U}\Pi_2^T$; $\blacklozenge$ = direct computation with $\mathbb{V}_m$, using standard algorithms, produces backward error E that is small in matrix norm, and the condition number is $\kappa_2(\mathbb{V}_m)$.

$$|\widetilde{L}_{ij} - L_{ij}| \leq \epsilon |L_{ij}|, \quad |\widetilde{D}_{ii} - D_{ii}| \leq \epsilon |D_{ii}|, \quad |\widetilde{U}_{ij} - U_{ij}| \leq \epsilon |U_{ij}|, \tag{6}$$

$$\widehat{W}_m = (((((\mathbf{X}_m \mathbb{F})\Pi_2)U^{-1})D^{-1})L^{-1})\Pi_1. \tag{7}$$

See [Demmel], [Demmel+Koev], [Dopico+Molera] for error analysis.

Matlab code for $Z_m = \mathbf{X}_m \mathbb{V}_m^{-1}$

```
function Z = X_inv_Vandermonde( lambda, X  )
% X_inv_Vandermonde computes Z = X*inv(V(lambda)), where X has m
% columns and V(lambda)=fliplr(vander(lambda)) is the m x m
% Vandermonde matrix defined by the  m x 1 vector lambda;
% V(lambda)_{ij} = lambda(i)^(j-1), i,j=1,...,m.
%.................................................................
% Coded by Zlatko Drmac, drmac@math.hr.
%.................................................................
%
m = length(lambda) ;
% .. pivoted LDU of V(lambda)*DFT ; p1, p2 permutations
[ L, D, U, p1, p2 ] = Vand_DFT_LDU( lambda, m, 'LDU' ) ;
Z = ifft(X,[],2)  ;
Z = ( ( Y(:,p2) / U ) * diag(sqrt(m)./D) ) / L ;
p1i(p1) = 1:m ; Z = Z(:,p1i) ; % p1i is the inverse of p1
end
```

- Not as simple as `Zm = Xm/Vm`
- More accurate than `Zm = Xm/Vm`; independent of the distribution of the λ_i's

Numerical stress test drive: Q2D Kolmogorov flow

Example

We use the simulation data of a 2D model obtained by depth averaging the Navier–Stokes equations for a shear flow in a thin layer of electrolyte suspended on a thin lubricating layer of a dielectric fluid; see [Tithof+et al], [Suri+et al] for more detailed description of the experimental setup.[a] The (scalar) vorticity field data consists of n_t snapshots of dimensions $n_x \times n_y$; in this particular example $n_t = 1201 \equiv m+1$, $n_x = n_y = 128$. The $n_x \times n_y \times n_t$ tensor is matricized into $n_x \cdot n_y \times n_t$ matrix of snapshots $(\mathbf{f}_1, \ldots, \mathbf{f}_{n_t})$, and $\mathbf{X}_m$ is of dimensions 16384×1200.

[a] We thank Michael Schatz, Balachandra Suri, Roman Grigoriev and Logan Kageorge from the Georgia Institute of Technology for providing us with the data.

This is a good stress test because $\kappa_2(\mathbb{V}_m) > 10^{76}$

Want to show that we can use $C_m, \mathbb{V}_m$ in working precision (IEEE 64 bit) despite the fact that $\kappa_2(\mathbb{V}_m) > 10^{76} \gg 1/\mathtt{roundoff}_{64} \approx 4.5 \cdot 10^{15}$

Test the reconstruction potential

Reconstructing snapshots using selected modes

$$\mathbf{f}_i \approx \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j \lambda_{\varsigma_j}^{i-1} \equiv \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j |\lambda_{\varsigma_j}|^{i-1} e^{\mathrm{i}\omega_{\varsigma_j}(i-1)\delta t}, \quad i = 1, \ldots, m.$$

where, for simplicity, the modes are selected by taking given number of modes with absolutely largest amplitudes $|\alpha_j|$ (*dominant modes*).

1. inversion of the Vandermonde matrix by the backslash operator in Matlab ; $\kappa_2(\mathbb{V}_m) > 10^{76} \gg 1/\mathtt{roundoff}_{64} \approx 4.5 \cdot 10^{15}$
2. inversion of the row scaled Vandermonde matrix by the backslash operator in Matlab: $\mathbb{V}_m = D_r \mathbb{V}_m^{(r)}$, $\widehat{W}_m = (\mathbf{X}_m (\mathbb{V}_m^{(r)})^{-1}) D_r^{-1}$, where $D_r = \mathrm{diag}(\|\mathbb{V}_m(i,:)\|)_{i=1}^m$; $\kappa_2(\mathbb{V}_m^{(r)}) \approx 3.1 \cdot 10^7 \approx 0.45 \cdot 1/\mathtt{roundoff}_{32}$
3. inversion of the column scaled Vandermonde matrix by the backslash operator in Matlab: $\mathbb{V}_m = \mathbb{V}_m^{(c)} D_c$, $\widehat{W}_m = (\mathbf{X}_m D_c^{-1})(\mathbb{V}_m^{(c)})^{-1}$, where $D_c = \mathrm{diag}(\|\mathbb{V}_m(:,i)\|)_{i=1}^m$. $\kappa_2(\mathbb{V}_m^{(c)}) \approx 3.0 \cdot 10^{21}$

Björck-Pereyera, DMD, DFT+Cauchy

We now test the following three methods:

1. Companion matrix formulation with the Björck-Pereyera method for Vandermonde systems. Although forward stable in the special case of real and ordered λ_i's, this method may be very sensitive in the case of general complex λ_i's and relatively large dimension m.
2. Companion matrix formulation with the DFT and inversion of the Cauchy matrix. Since $\mathbb{F}$ and $\mathcal{D}_2$ in (1) are unitary, the algorithm solves linear system with the matrix $\mathcal{D}_1\mathcal{C} = \mathbb{V}_m\mathbb{F}\mathcal{D}_2^*$ of condition number bigger than 10^{76}. No additional scaling is used; we want to illustrate the claim that such high condition number cannot spoil the result.
3. Schmid's DMD method. Here we expect good reconstruction results, provided it is feasible for given data and the parameters. The SVD is not truncated because $\kappa_2(\mathbf{X}_m) \approx 5.5 \cdot 10^{10}$ ($\sigma_{\max}(\mathbf{X}_m) \approx 4.2 \cdot 10^3$, $\sigma_{\min}(\mathbf{X}_m) \approx 7.5 \cdot 10^{-8}$).

f_{321} with 300 modes; similar results for other f_i's

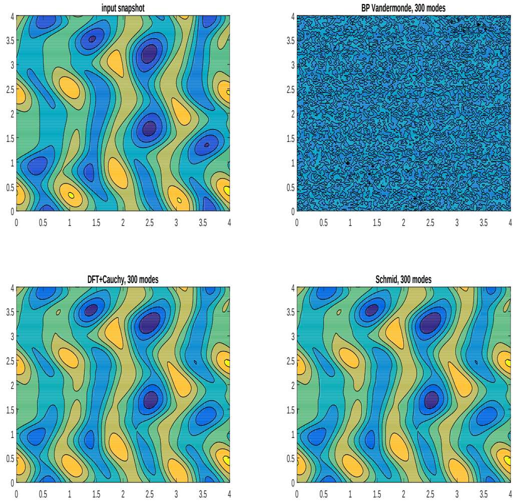


Figure: Reconstruction of $\mathbf{f}_{321}$ using 300 dominant modes. Björck-Pereyera method (second plot in the first row) failed to produce any useful data. The DFT+Cauchy inversion and the Schmid's DMD reconstruction (second row) succeeded in reconstructing $\mathbf{f}_{321}$ pretty much using 300 modes with dominant amplitudes.

On the numerical aspects of snapshot reconstruction

For given (λ_j, z_j)'s and nonnegative weights $\mathfrak{w}_i$, find the α_j's to achieve

$$\sum_{i=1}^{m} \mathfrak{w}_i^2 \|\mathbf{f}_i - \sum_{j=1}^{\ell} z_j \alpha_j \lambda_j^{i-1}\|_2^2 \longrightarrow \min . \qquad (1)$$

Set $\mathbf{W} = \mathrm{diag}(\mathfrak{w}_i)_{i=1}^m$. The weights $\mathfrak{w}_i > 0$ are used to emphasize snapshots whose reconstruction is more important. Let $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_j)_{j=1}^{\ell}$,

$$\Delta_{\boldsymbol{\alpha}} = \begin{pmatrix} \alpha_1 & 0 & \cdot & 0 \\ 0 & \alpha_2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & 0 & \alpha_\ell \end{pmatrix}, \; \Lambda_i = \begin{pmatrix} \lambda_1^{i-1} \\ \lambda_2^{i-1} \\ \cdot \\ \lambda_\ell^{i-1} \end{pmatrix}, \;\; \Delta_{\Lambda_i} = \begin{pmatrix} \lambda_1^{i-1} & 0 & \cdot & 0 \\ 0 & \lambda_2^{i-1} & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & 0 & \lambda_\ell^{i-1} \end{pmatrix} \equiv \boldsymbol{\Lambda}^{i-1},$$

and write the objective (1) as the function of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\ell)^T$,

$$\Omega^2(\boldsymbol{\alpha}) \equiv \| \left[\mathbf{X}_m - Z_\ell \Delta_{\boldsymbol{\alpha}} \begin{pmatrix} \Lambda_1 & \Lambda_2 & \ldots & \Lambda_m \end{pmatrix} \right] \mathbf{W} \|_F^2 \longrightarrow \min, \qquad (2)$$

$$\begin{pmatrix} \Lambda_1 & \Lambda_2 & \ldots & \Lambda_m \end{pmatrix} = \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{m-1} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & \lambda_\ell & \ldots & \lambda_\ell^{m-1} \end{pmatrix} \equiv \mathbb{V}_{\ell,m} \in \mathbb{C}^{\ell \times m}. \qquad (3)$$

Explicit normal equations solution: weighted case

QRF $Z_\ell = QR$; $\mathbf{g}_i = Q^*\mathbf{f}_i$, $\vec{\mathbf{g}}^T = (\mathbf{g}_1, \ldots, \mathbf{g}_m)$. Solve equivalently

$$\|(\mathbf{W} \otimes \mathbb{I}_\ell)\,[\vec{\mathbf{g}} - S\boldsymbol{\alpha}]\,\|_2 \rightarrow \min, \text{ where } S = (\mathbb{I}_m \otimes R)\begin{pmatrix} \Delta_{\Lambda_1} \\ \vdots \\ \Delta_{\Lambda_m} \end{pmatrix} \equiv \begin{pmatrix} R\Delta_{\Lambda_1} \\ \vdots \\ R\Delta_{\Lambda_m} \end{pmatrix}.$$

Observation: $S = \mathbb{V}_{\ell,m}^T \odot R$ (Khatri-Rao product)

Theorem

With the notation as above, the unique solution $\boldsymbol{\alpha}$ of the LSP (1) is

$$\boldsymbol{\alpha} = [(R^*R) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})]^{-1}[(\overline{\mathbb{V}_{\ell,m}\mathbf{W}} \circ (R^*G\mathbf{W}))\mathbf{e}], \tag{4}$$

where $G = \begin{pmatrix}\mathbf{g}_1 & \ldots & \mathbf{g}_m\end{pmatrix}$, $\mathbf{e} = \begin{pmatrix}1 & \ldots & 1\end{pmatrix}^T$. In terms of $\mathbf{X}_m$, Z_ℓ,

$$\boldsymbol{\alpha} = [(Z_\ell^* Z_\ell) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})]^{-1}[(\overline{\mathbb{V}_{\ell,m}\mathbf{W}} \circ (Z_\ell^*\mathbf{X}_m\mathbf{W}))\mathbf{e}]. \tag{5}$$

This includes the DMDSP of [Jovanović+et al] and solution for scattering coefficients in multistatic antenna array processing [Lev-Ari] as unweighted cases. Are normal equations safe to use? What is the impact of the Hadamard matrix product $\circ$? Let us experiment with a small dimension

Squaring the conditon number – losing definiteness

Let $\mathbf{W} = I$. Let $\ell = 3$, $m = 4$, $\xi = \sqrt{\varepsilon}$, $\lambda_1 = \xi$, $\lambda_2 = 2\xi$, $\lambda_3 = 0.2$, so that the Vandermonde section $\mathbb{V}_{\ell,m}$ equals

$$\mathbb{V}_{\ell,m} = \begin{pmatrix} 1 & 1.490116119384766e{-08} & 2.220446049250313e{-16} & 3.308722450212111e{-24} \\ 1 & 2.980232238769531e{-08} & 8.881784197001252e{-16} & 2.646977960169689e{-23} \\ 1 & 2.000000000000000e{-01} & 4.000000000000001e{-02} & 8.000000000000002e{-03} \end{pmatrix},$$

$$R = \begin{pmatrix} 1 & 1 & 1 \\ 0 & \xi/2 & \xi \\ 0 & 0 & \xi \end{pmatrix} = \begin{pmatrix} 1 & 1.000000000000000e{+00} & 1.000000000000000e{+00} \\ 0 & 7.450580596923828e{-09} & 1.490116119384766e{-08} \\ 0 & 0 & 1.490116119384766e{-08} \end{pmatrix}.$$

Here $\kappa_2(\mathbb{V}_{\ell,m}) \approx 10^9$, $\kappa_2(R) \approx 10^9 \ll 1/\mathtt{roundoff}_{64} \approx 4.5 \cdot 10^{15}$.

```
>> chol(Vlm*Vlm')
Error using chol
Matrix must be ....
```

```
>> chol((R'*R).*(Vlm*Vlm'))
Error using chol
Matrix must be positive definite.
```

```
>> chol(R'*R)
Error using chol
Matrix must be positive definite.
```

Normal equations matrix is not definite!

Indefinite ∘ Indefinite = Positive Definite ?!

Use the same $\mathbb{V}_{\ell,m}$ but change the definition of R to

$$R = \begin{pmatrix} 1 & 1 & 1 \\ 0 & \xi & \xi \\ 0 & 0 & \xi/2 \end{pmatrix} = \begin{pmatrix} 1 & 1.000000000000000e{+}00 & 1.000000000000000e{+}00 \\ 0 & 1.490116119384766e{-}08 & 1.490116119384766e{-}08 \\ 0 & 0 & 7.450580596923828e{-}09 \end{pmatrix}.$$

If we repeat the experiment with the Cholesky factorizations, we obtain

```
>> chol(Vlm*Vlm')
Error using chol
Matrix must be positive definite.
>> chol(R'*R)
Error using chol
Matrix must be positive definite.
>> TC = chol((R'*R).*(Vlm*Vlm'))
TC =
1     1.000000000000000e+00     1.000000002980232e+00
0     1.490116119384765e-08     1.999999880790710e-01
0                         0     4.079214149695062e-02
```

Theorem

Let A ad B be Hermitian positive semidefinite matrices with positive diagonal entries, and let $C = A \circ B$. If $A_s = (a_{ij}/\sqrt{a_{ii}a_{jj}})$, $B_s = (b_{ij}/\sqrt{b_{ii}b_{jj}})$, $C_s = (c_{ij}/\sqrt{c_{ii}c_{jj}})$, then

$$\max(\lambda_{\min}(A_s), \lambda_{\min}(B_s)) \leq \lambda_i(C_s) \leq \min(\lambda_{\max}(A_s), \lambda_{\max}(B_s)). \quad (6)$$

In particular, $\|C_s^{-1}\|_2 \leq \min(\|A_s^{-1}\|_2, \|B_s^{-1}\|_2)$ and $\kappa_2(C_s) \leq \min(\kappa_2(A_s), \kappa_2(B_s))$. If A or B is diagonal, all inequalities in this theorem become equalities.

Corollary

Let $C \equiv (R^*R) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})$, $C_s = (c_{ij}/\sqrt{c_{ii}c_{jj}})$. Further, let $R = R_c\Delta_r$ and $\mathbb{V}_{\ell m}\mathbf{W} = \Delta_v(\mathbb{V}_{\ell m}\mathbf{W})_r$ with diagonal scaling matrices Δ_r and Δ_v such that R_c has unit columns and $(\mathbb{V}_{\ell m}\mathbf{W})_r$ has unit rows (in Euclidean norm). Then $\kappa_2(C_s) \leq \min(\kappa_2(R_c)^2, \kappa_2((\mathbb{V}_{\ell,m}\mathbf{W})_r)^2)$.

In the Q2D Kolmogorov flow example, $\kappa_2(C) > 10^{87} \gg \kappa_2(C_s) \approx$ 8.5+01.

Numerical example: effect of weighted reconstruction

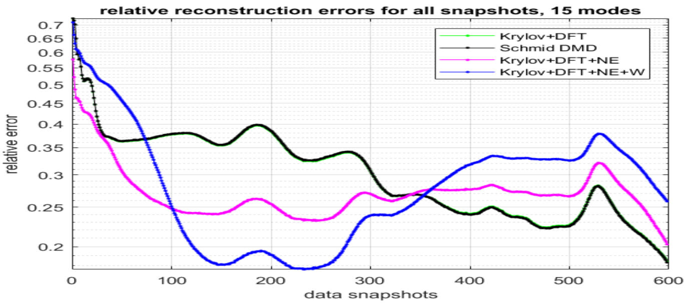


Figure: Example with 600 snapshots. Blue curve shows the effects of weighting.

Data: 2D model obtained by depth averaging the Navier–Stokes equations for a shear flow in a thin layer of electrolyte suspended on a thin lubricating layer of a dielectric fluid.[1]

[1]We thank M.Schatz, B. Suri, R. Grigoriev and L. Kageorge from the Georgia Institute of Technology for providing us with the data.

$\|\vec{\mathbf{g}} - S\boldsymbol{\alpha}\|_2 \longrightarrow \min$; $S = Q_S R_S$, $\boldsymbol{\alpha} = R_S^{-1}(Q_S^* \vec{\mathbf{g}})$

$$\boldsymbol{\alpha} = R_S^{-1}(R_S^{-*}(S^*\vec{\mathbf{g}})),\ r = \vec{\mathbf{g}} - S\boldsymbol{\alpha}$$
$$\delta\boldsymbol{\alpha} = R_S^{-1}(R_S^{-*}(S^* r)),\ \ \boldsymbol{\alpha}_* = \boldsymbol{\alpha} + \delta\boldsymbol{\alpha} \tag{7}$$

Algorithm: Corrected semi-normal solution

Input: R, Λ, G, S

Output: Corrected solution $\boldsymbol{\alpha}_*$

```
1: Compute the triangular factor $R_S$ in the QR factorization of $S$.
2: $g_S = [(\overline{\mathbb{V}_{\ell,m}} \circ (R^* G))\mathbf{e}]$ {Note, $g_S = S^*\vec{\mathbf{g}}$. Use xTRMM from BLAS 3.}
3: $\boldsymbol{\alpha} = R_S^{-1}(R_S^{-*} g_S)$ {Use xTRSM or xTRTRS or xTRSV from LAPACK.}
4: $r_\square = G - R\begin{pmatrix}\boldsymbol{\alpha} & \Lambda\boldsymbol{\alpha} & \Lambda^2\boldsymbol{\alpha} & \ldots & \Lambda^{m-1}\boldsymbol{\alpha}\end{pmatrix} \equiv G - R\mathrm{diag}(\boldsymbol{\alpha})\mathbb{V}_{\ell,m}$
5: $r_S = [(\overline{\mathbb{V}_{\ell,m}} \circ (R^* r_\square))\mathbf{e}]$ {Note, $r_S = S^* r$. Use xTRMM from BLAS 3.}
6: $\delta\boldsymbol{\alpha} = R_S^{-1}(R_S^{-*} r_S)$ {Use xTRSM or xTRTRS or xTRSV from LAPACK.}
7: $\boldsymbol{\alpha}_* = \boldsymbol{\alpha} + \delta\boldsymbol{\alpha}$
```

Considerably improves over normal equations, but needs QR factorization of $S = \mathbb{V}_{\ell,m}^T \odot R$. How to compute it efficiently, using the structure of S?

Algorithm: Recursive QR factorization of S for $m = 2^p$

Input: Upper triangular $R \in \mathbf{c}^{\ell\times\ell}$; diagonal $\mathbf{\Lambda} \in \mathbf{c}^{\ell\times\ell}$; number of snapshots $m = 2^p$
Output: Upper triangular QR factor $R_S = T_p$ of $S \in \mathbf{c}^{2^p\ell\times\ell}$

$$\begin{array}{|c|r|r|r||r|}
\boxed{\boxed{T_4}} & \longleftarrow \boxed{T_3} & \longleftarrow \boxed{T_2} & \longleftarrow \boxed{T_1} & \longleftarrow R\mathbf{\Lambda}^0 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \longleftarrow R\mathbf{\Lambda}^1 \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \longleftarrow T_1\mathbf{\Lambda}^2 & R\mathbf{\Lambda}^2 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^3 \\ \hline\hline
\mathbf{0} & \mathbf{0} & \longleftarrow T_2\mathbf{\Lambda}^4 & T_1\mathbf{\Lambda}^4 & R\mathbf{\Lambda}^4 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^5 \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & T_1\mathbf{\Lambda}^6 & R\mathbf{\Lambda}^6 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^7 \\ \hline\hline\hline
\mathbf{0} & \longleftarrow T_3\mathbf{\Lambda}^8 & T_2\mathbf{\Lambda}^8 & T_1\mathbf{\Lambda}^8 & R\mathbf{\Lambda}^8 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^9 \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & T_1\mathbf{\Lambda}^{10} & R\mathbf{\Lambda}^{10} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^{11} \\ \hline\hline
\mathbf{0} & \mathbf{0} & T_2\mathbf{\Lambda}^{12} & T_1\mathbf{\Lambda}^{12} & R\mathbf{\Lambda}^{12} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^{13} \\ \hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & T_1\mathbf{\Lambda}^{14} & R\mathbf{\Lambda}^{14} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & R\mathbf{\Lambda}^{15}
\end{array},$$

```
1 :  $T_0 = R$
2 :  for $i = 1 : p$ do
3 :      $\begin{pmatrix} \boxed{T_i} \\ \mathbf{0} \end{pmatrix} = \texttt{qr}(\begin{pmatrix} \boxed{T_{i-1}} \\ T_{i-1}\mathbf{\Lambda}^{2^{i-1}} \end{pmatrix})$
4 :  end for
```

Matlab QRF code for $S = \mathbb{V}_{\ell,m}^T \odot R$, $m = 2^p$

```
function T = QR_Khatri_Rao_VTR_2p( R, Lambda, p )
% QR_Khatri_Rao_VTR_2p computes the upper triangular factor
% in the QR factorization of the Khatri-Rao product
% S=Khatri_Rao(Vlm.',R), where R is an <ell x ell> upper
% triangular matrix, and Vlm is an  <ell x m> Vandermonde
% matrix V, whose columns are V(:,i) = Lambda.^(i-1),
% i = 1,...,m, and m=2^p.
% Input:
% R      upper triangular matrix
% Lambda vector, defines Vlm = Vandermonde matrix
% p      integer >=0    defines m = 2^p
% Output:
% T      triangular QR fator of Khatri_Rao(Vlm.',R)
T = R ; D = Lambda ;
%
for i = 1 : p
[~, T] = qr( [ T ; T*diag(D)], 0 ) ;
D = D.^2 ;
end
end
```

Input: Upper triangular $R \in \mathbb{C}^{\ell\times\ell}$; diagonal $\mathbf{\Lambda} \in \mathbf{c}^{\ell\times\ell}$; m
Output: Upper triangular QR factor $R_S = \mathbb{T}_{j-1}$ of S.

```
 1: Compute the binary representation of $m$: $m \equiv \mathfrak{b} = (\mathfrak{b}_{\lfloor \log_2 m \rfloor}, \ldots, \mathfrak{b}_1, \mathfrak{b}_0)_2$, $m \equiv \sum_{j=1}^{j^*} 2^{i_j}$
 2: Let $\lfloor \log_2 m \rfloor = i_{j^*} > i_{j^*-1} > \cdots > i_2 > i_1 \geq 0$
 3: $T_0 = R$
 4: if $i_1 = 0$ then
 5:     $\mathbb{T}_1 = T_0$; $j = 2$; $\wp = 1$
 6: else
 7:     $\mathbb{T}_0 = []$; $j = 1$; $\wp = 0$
 8: end if
 9: for $k = 1 : i_{j^*}$ do
10:     $\begin{pmatrix} T_k \\ \mathbf{0} \end{pmatrix} = \mathtt{qr}(\begin{pmatrix} T_{k-1} \\ T_{k-1}\mathbf{\Lambda}^{2^{k-1}} \end{pmatrix})$ {Local factor.}
11:     if $k = i_j$ then
12:         if $\mathbb{T}_{j-1} \neq []$ then
13:             $\begin{pmatrix} \mathbb{T}_j \\ \mathbf{0} \end{pmatrix} = \mathtt{qr}(\begin{pmatrix} \mathbb{T}_{j-1} \\ T_k\mathbf{\Lambda}^{\wp} \end{pmatrix})$ {Global factor.}
14:         else
15:             $\mathbb{T}_j = T_k$
16:         end if
17:         $j := j + 1$; $\wp := \wp + 2^k$
18:     end if
19: end for
```

Comments and concluding remarks

- Provably small backward error

$$\|\delta S(:,j)\|_2 \leq \eta \|S(:,j)\|_2, \;\; j = 1, \ldots, \ell; \;\; \eta \leq f(\ell, m)\boldsymbol{\varepsilon},$$

- The relevant condition number is of the column scaled S:

Corollary

$$\begin{aligned}\kappa_2(S_c) = \sqrt{\kappa_2(C_s)} &\leq \min(\kappa_2(R_c), \kappa_2((\mathbb{V}_{\ell,m})_r)) \\ &\leq \sqrt{\ell}\min(\min_{D=diag}\kappa_2(RD), \min_{D=diag}\kappa_2(D\mathbb{V}_{\ell,m})).\end{aligned}$$

- If the data is real, can work in real arithmetic even if the eigenvalues are complex (conjugate pairs)
- Improves the result in difficult (severely ill-conditioned) cases.

Let us, at the end, comment two examples where new highly accurate NLA algorithms open new possibilities in some computational tasks in rational approximations.

Example: VF algorithm

1. Given: The sampling data $\mathbf{H}(\xi_i)$ for $i = 1, \ldots, \ell$; maximal number of iterations $k_{\max}$.
2. Set $k \leftarrow 0$ and make an initial pole selection $\boldsymbol{\lambda}^{(k+1)} \in \mathbb{C}^r$.
3. WHILE $\{$ stopping criterion not satisfied and $k \leq k_{\max}$ $\}$
    - Form $\mathbf{A}^{(k+1)}$ and $\mathbf{b}$.
    - Compute $\mathbf{B}^{(k+1)} = \Pi(\mathcal{Q}^{(k+1)})^* \mathbf{A}^{(k+1)}$ and $\mathbf{s}^{(k+1)} = \Pi(\mathcal{Q}^{(k+1)})^* \mathbf{b}$ and partition as required.
    - Solve $\|\mathbf{B}_{[22]}^{(k+1)} \boldsymbol{\varphi}^{(k+1)} - \mathbf{s}_2^{(k+1)}\|_2 \longrightarrow \min$ for $\boldsymbol{\varphi}^{(k+1)}$.
    - Set $k \leftarrow k+1$ and compute $\boldsymbol{\lambda}^{(k+1)} = zeros(1 + \sum_{j=1}^{r} \varphi_j^{(k)}/(s - \lambda_j^{(k)}))$.
4. END WHILE
5. $\boldsymbol{\Phi} = (\mathbf{B}_{[11]}^{(k)})^{-1} \mathbf{s}_1^{(k)}$.

Computing $\boldsymbol{\Phi}$ requires solving a sequence of weighed LS problems with scaled Cauchy coefficient matrices.

Ill-conditioning

Extracting residues from weighted Cauchy LS problems

$$\|\mathcal{D}_\rho \left(\mathcal{C}^{(k+1)}\Phi^{(k+1)}(u,v,:) - \mathbb{S}(u,v,:)\right)\|_2 \longrightarrow \min, \;\; u = 1:p, \;\; v = 1:m.$$

To simplify the notation, write $\|\mathcal{D}_\rho \mathcal{C} x - h\|_2 \longrightarrow \min$, where $\mathcal{C} = \mathcal{C}_{\boldsymbol{\xi},\boldsymbol{\lambda}}$ is a Cauchy matrix, h is the corresponding scaled right-hand side, $\boldsymbol{\lambda}$ is closed under conjugation and and the solution vector should also be closed under conjugation. Consider equivalent augmented unconstrained LS

$$\left\| \begin{pmatrix} \mathcal{D}_\rho \mathcal{C}_{\boldsymbol{\xi},\boldsymbol{\lambda}} \\ \mathcal{D}_\rho \mathcal{C}_{\overline{\boldsymbol{\xi}},\boldsymbol{\lambda}} \end{pmatrix} x - \begin{pmatrix} h \\ \overline{h} \end{pmatrix} \right\|_2 \equiv \|\widehat{\mathcal{C}} x - \widehat{h}\|_2 \longrightarrow \min$$

with the coefficient matrix again of the diagonally scaled Cauchy structure, $\widehat{\mathcal{C}} = (\mathcal{D}_\rho \oplus \mathcal{D}_\rho)\mathcal{C}_{(\boldsymbol{\xi},\overline{\boldsymbol{\xi}}),\lambda}$.

Accurate regularized LS solution

Let $\widehat{\mathcal{C}} = W\Sigma V^*$ be the SVD and let the unique[2] LS solution be $x = V\Sigma^\dagger W^* = \sum_{i=1}^{r} v_i(w_i^*\widehat{h})/\sigma_i$. Unfortunately, an accurate SVD is not enough to have the LS solution computed to high relative accuracy, and additional regularization techniques must be deployed. This is in particular important if the right-hand side is contaminated by noise. In the Tichonov regularization, we choose $\mu \geq 0$ and use the solution of $\|\widehat{\mathcal{C}}x - \widehat{h}\|_2^2 + \mu^2\|x\|_2^2 \to \min$, explicitly computable as

$$x_\mu = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \mu^2}(w_i^*\widehat{h})v_i. \tag{1}$$

The parameter μ can be further adjusted using the Morozov discrepancy principle, i.e., to achieve $\|\widehat{\mathcal{C}}x_\mu - \widehat{h}\|_2 \approx \nu$, where ν is the estimated level of noise $\delta\widehat{h}$ in the right-hand side, $\nu \approx \|\delta\widehat{h}\|_2$.

[2]Since all nodes are distinct and the poles are assumed simple, the matrix is theoretically of full column rank, but numerically potentially severely ill-conditioned.

Example: Rational approximation – AAK theory and con-eigenvalues

Haut and Beylkin (2011) used Adamyan–Arov–Krein theory to show that nearly L^∞–optimal rational approximation on $|z| = 1$ of

$$f(z) = \sum_{i=1}^{n} \frac{\alpha_i}{z - \gamma_i} + \sum_{i=1}^{n} \frac{\overline{\alpha_i} z}{1 - \overline{\gamma_i} z} + \alpha_0$$

with $\max_{|z|=1} |f(z) - r(z)| \rightsquigarrow \min$,

$$r(z) = \sum_{i=1}^{m} \frac{\beta_i}{z - \eta_i} + \sum_{i=1}^{m} \frac{\overline{\beta_i} z}{1 - \overline{\eta_i} z} + \alpha_0$$

is numerically feasible if one can accurately compute the con–eigenvalues and con–eigenvectors

$$Cu = \lambda \overline{u}, \quad C_{ij} = \frac{\sqrt{\alpha_i}\sqrt{\overline{\alpha_j}}}{\gamma_i^{-1} - \overline{\gamma_j}}$$

Con–eigenvalues

Here $C = (\frac{\sqrt{\alpha_i}\sqrt{\overline{\alpha_j}}}{\gamma_i^{-1} - \overline{\gamma_j}})$ is positive definite Cauchy matrix C.
The con–eigenvalue problem $Cu = \lambda\overline{u}$ is equivalent to solving

$$\overline{C}Cu = |\lambda|^2 u,$$

where C is factored as $C = XD^2X^*$. The problem reduces to computing the SVD of the product $G = DX^T XD$. Accurate SVD via the PSVD based on the Jacobi SVD (**Z.D.**). Haut and Beylkin tested the accuracy with $\kappa_2(C) > 10^{200}$ and using Mathematica with 300 hundred digits for reference values. Over 500 test examples of size 120, the maximal error in IEEE 16 digit arithmetic ($\varepsilon \approx 2.2 \cdot 10^{-16}$) was

$$\frac{|\tilde{\lambda}_i - \lambda_i|}{|\lambda_i|} < 5.2 \cdot 10^{-12}, \quad \frac{\|\tilde{u}_i - u_i\|_2}{\|u_i\|_2} < 5.4 \cdot 10^{-12}.$$

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.
- Using high accuracy numerical linear algebra techniques we were able to curb the ill-conditioning of the companion matrix's associated Vandermonde matrix allowing for an accurate inversion and computing the DMD modes accurately.
- Ill–conditioning can be artificial, an artifact of a particular formulation and/or algorithm, and not the underlying problem. In many cases accurate computation is possible, despite high classical condition numbers. We have illustrated with two examples from rational approximation practice.

For more on this and complete references list see

- Z. Drmač, I. Mezić, and R. Mohr, On least squares problem with certain Vandermonde–Khatri–Rao structure with applications to DMD. *SIAM Journal on Scientific Computing, 42(5), A3250–A3284., 2020*.

- Z. Drmač, I. Mezić, and R. Mohr, Data driven Koopman spectral analysis in Vandermonde-Cauchy form via the DFT: numerical method and theoretical insights. *SIAM Journal on Scientific Computing*, 41(5): A3118–A3151, 2019.

- Z. Drmač, I. Mezić, and R. Mohr, Data driven modal decompositions: analysis and enhancements. *SIAM Journal on Scientific Computing*, 40(4):A2253–A2285, 2018.

- Z. Drmač, S. Gugercin, and C. Beattie, *Vector Fitting for matrix-valued rational approximation*, SIAM Journal on Scientific Computing 37 (5), A2346–A2379, 2015.

- Z. Drmač, S. Gugercin, and C. Beattie, *Quadrature-based vector fitting for discretized $\mathcal{H}_2$ approximation*, SIAM Journal on Scientific Computing 37 (2), A625–A652, 2015.