MATH-656
Numerical linear algebra for Koopman and Dynamic Mode Decomposition (DMD)

Zlatko Drmač
Department of Mathematics, University of Zagreb, Croatia

EPFL, Lausanne March–April 2024

Content I

Content II

Overview of the course

1. Introduction. What is the Koopman operator? What is the DMD and what is the connection? What is the Extended/kernel DMD? How to use kernel trick? What is the Exact DMD?
2. Koopman mode decomposition. Least squares decomposition of the data snapshots. Khatri–Rao structure of the LS problem.
3. Compressed DMD and the streaming DMD.
4. Physics–informed DMD. Hermitian DMD and measure preserving (unitary) DMD. Weighted DMD.
5. Data driven system identification. SINDY and the Mauroy-Goncalves methods. Computations with the infinitesimal generator of the Koopman semigroup.
6. Schur–Koopman modal decomposition for non–normal cases.
7. Software development.
8. Seminar projects.

Introduction

The "Koopmanism" has grown as a vast research subject in several research frameworks:

- Dynamical system theory.
- Operator (semigroup) theory.
- Ergodic theory.
- Application area oriented development that requires corresponding expertise (computational fluid dynamics, machine learning, ...)
- **Computational aspects – numerical methods and development of software tools. Data driven framework.**

Many misconceptions, many open problems, many applications, many publications, ... very active area of research.

Our goal in this course

We want to understand the numerical aspects of the "computational Koopmanism". It is a new research field within the numerical linear algebra. The NLA is the key for computational analysis of nonlinear dynamics. It's a linear world after all :)

Continuous autonomous dynamical systems

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

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 $\mathbf{F} : \mathcal{X} \to \mathbb{R}^N$.

Example

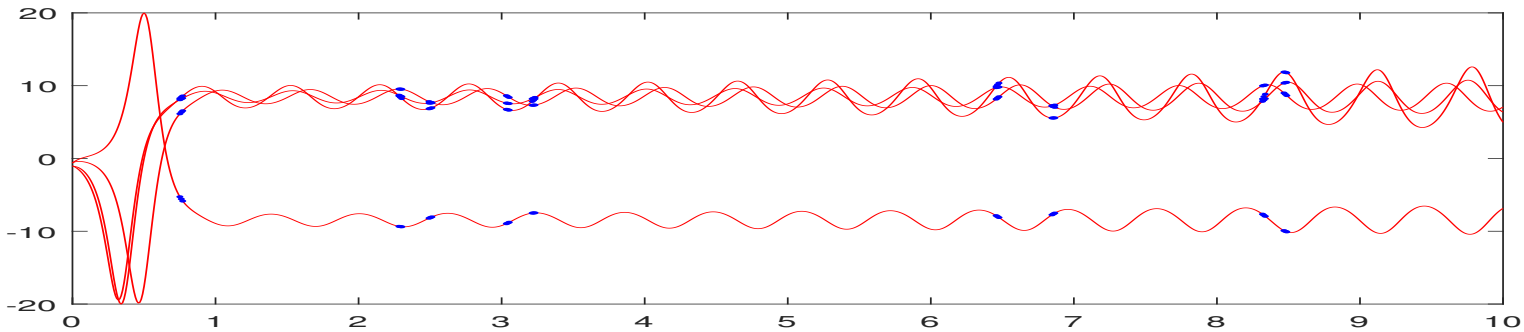
$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & 0 \\ 0 & 0 & -8/3 \end{pmatrix} \underbrace{\begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix}}_{\mathbf{x}(t)} + \begin{pmatrix} 0 \\ -x_1(t)x_3(t) \\ x_1(t)x_2(t) \end{pmatrix}. \tag{2}$$

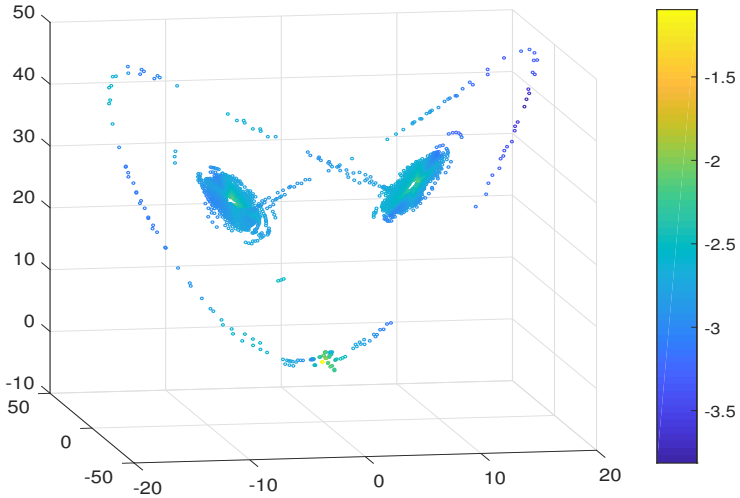
$$\begin{aligned} \mathbf{F}_1(\mathbf{x}(t)) &= -10x_1(t) + 10x_2(t); \\ \mathbf{F}_2(\mathbf{x}(t)) &= 28x_1(t) - x_2(t) - x_1(t)x_3(t); \\ \mathbf{F}_3(\mathbf{x}(t)) &= (-8/3)x_3(t) + x_1(t)x_2(t) \end{aligned}$$

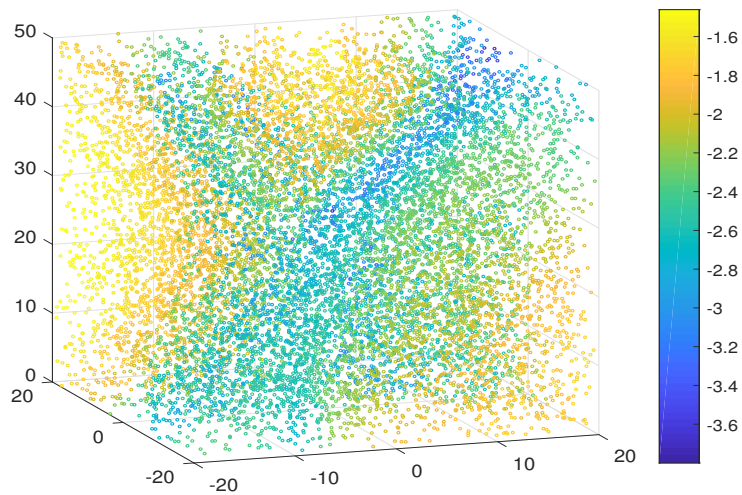
Task: Learn $\dot{x}(t) = \mathbf{F}(x(t))$ from data ($\mathbf{F}$ unknown)

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & 0 \\ 0 & 0 & -8/3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ -x_1 x_3 \\ x_1 x_2 \end{pmatrix}$$

- complex dynamics
- limited understanding







$\log_{10} \epsilon_k$, where $\epsilon_k = \max_{i=1,2,3} \frac{|\widetilde{F_i(\mathbf{x}_k)} - F_i(\mathbf{x}_k)|}{\|\mathbf{F}(\mathbf{x}_k)\|_\infty}$

Task: Reveal latent structure, coherent states

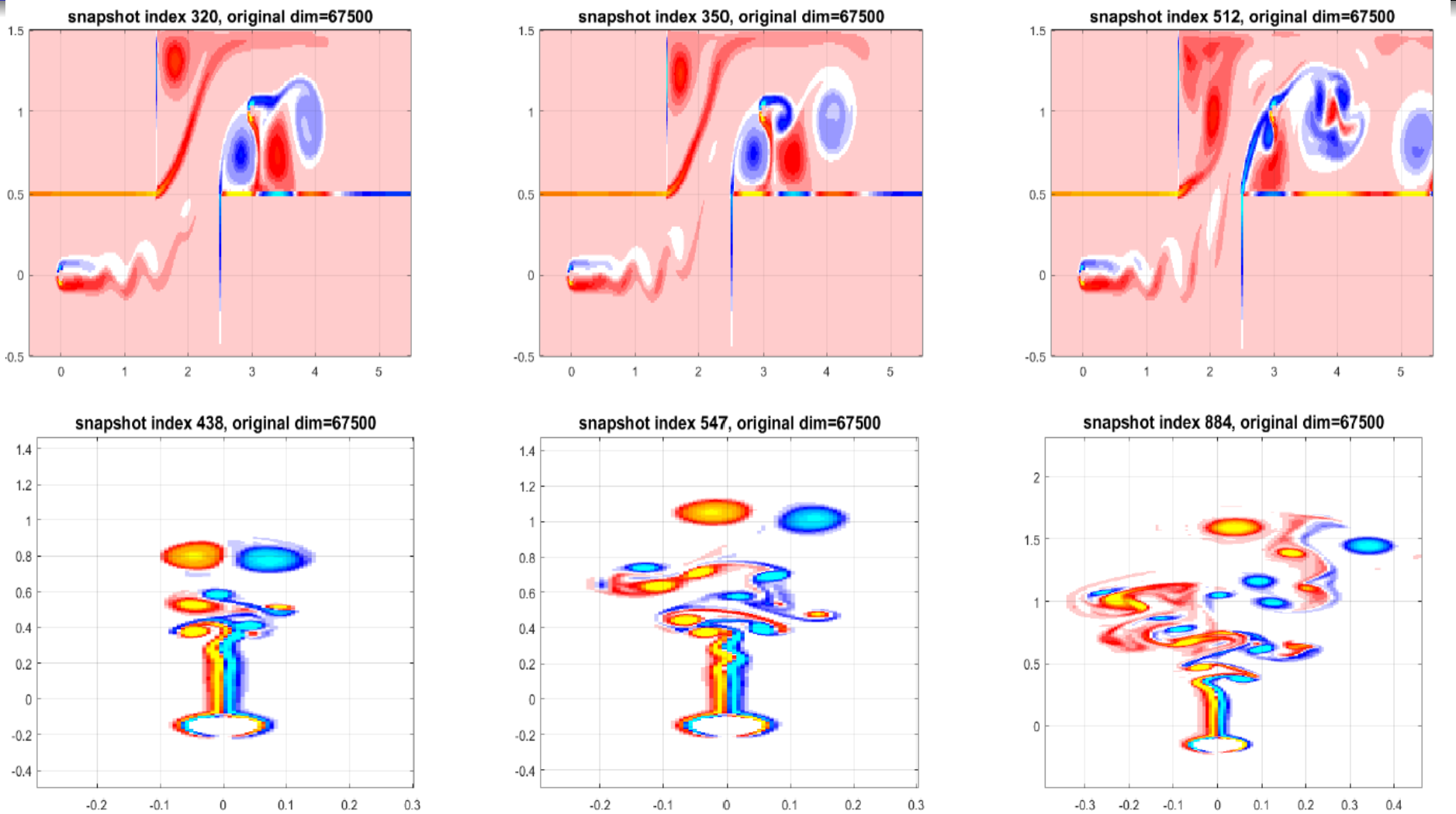


Figure: Use high fidelity numerical simulations to better understand physics. (Data source: Popinet 2004, Baeza Rojo and Günther 2019.)

Special case: Linear time invariant systems

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \equiv A\mathbf{x}(t), \quad A = \begin{pmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \dots & \vdots \\ a_{N1} & \dots & a_{NN} \end{pmatrix}, \quad \mathbf{x}(t_0) = \mathbf{x}_0,$$

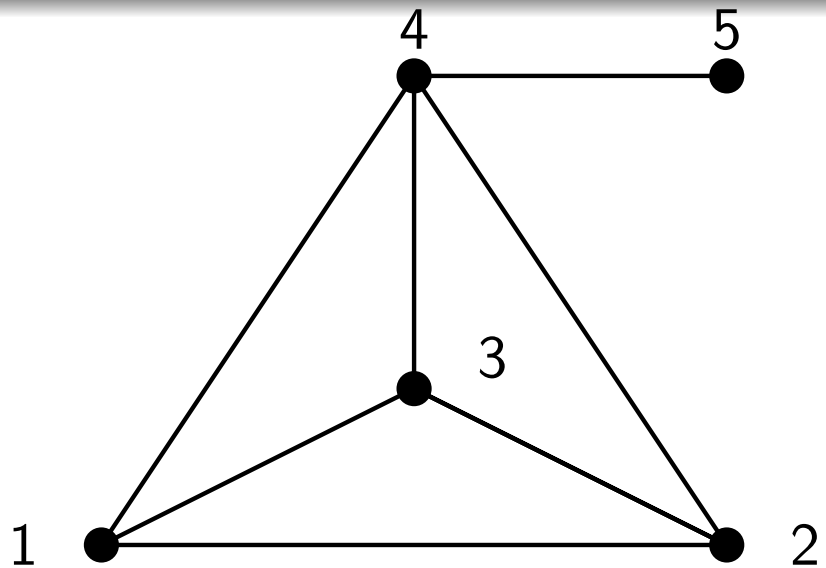
Solution: $\mathbf{x}(t_0 + t) = e^{tA}\mathbf{x}(t_0)$. Suppose A is diagonalizable, $A = S\Lambda S^{-1}$, $\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_N)$, $S = (\mathbf{s}_1, \dots, \mathbf{s}_N)$, $A\mathbf{s}_j = \lambda_j \mathbf{s}_j$; $\lambda_j = \alpha_j + \mathbf{i}\beta_j \in \mathbb{C}$ eigenvalue; $\mathbf{s}_j \in \mathbb{C}^N$ eigenvector. **The structure of the solution $\mathbf{x}(t_0 + t)$ is expressed using the spectral elements of A as follows:**

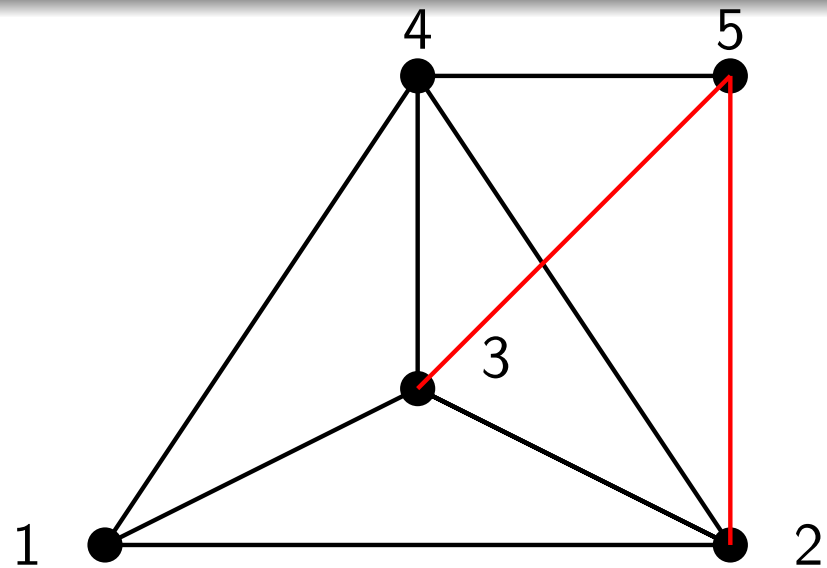
$$\begin{aligned} \mathbf{x}(t_0 + t) &= e^{tA}\mathbf{x}(t_0) = Se^{t\Lambda}S^{-1}\mathbf{x}(t_0) = S \begin{pmatrix} e^{t\lambda_1} & & \\ & \ddots & \\ & & e^{t\lambda_N} \end{pmatrix} \underbrace{S^{-1}\mathbf{x}(t_0)}_{y=(y_1,\dots,y_N)^T} \\ &= \mathbf{s}_1 e^{t\lambda_1} y_1 + \mathbf{s}_2 e^{t\lambda_2} y_2 + \cdots + \mathbf{s}_N e^{t\lambda_N} y_N \end{aligned}$$

$$\begin{pmatrix} (\mathbf{x}(t_0+t))_1 \\ (\mathbf{x}(t_0+t))_2 \\ \vdots \\ (\mathbf{x}(t_0+t))_N \end{pmatrix} = \begin{pmatrix} (\mathbf{s}_1)_1 \\ (\mathbf{s}_1)_2 \\ \vdots \\ (\mathbf{s}_1)_N \end{pmatrix} e^{t\lambda_1} y_1 + \begin{pmatrix} (\mathbf{s}_2)_1 \\ (\mathbf{s}_2)_2 \\ \vdots \\ (\mathbf{s}_2)_N \end{pmatrix} e^{t\lambda_2} y_2 + \cdots + \begin{pmatrix} (\mathbf{s}_N)_1 \\ (\mathbf{s}_N)_2 \\ \vdots \\ (\mathbf{s}_N)_N \end{pmatrix} e^{t\lambda_N} y_N.$$

$$e^{t\lambda_j} = e^{t\alpha_j} e^{\mathbf{i}t\beta_j} = e^{t\alpha_j}(\cos\beta_j t + \mathbf{i}\sin\beta_j t)$$

Discrete dynamical systems: $\mathbf{z}_{i+1} = \mathbf{T}(\mathbf{z}_i)$





$$A_1 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad A_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \ldots$$

Example (Time-evolving graph)

Suppose we are given a time-evolving graph $\mathcal{G} = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_M)$ with adjacency matrices $A_1, A_2, \ldots, A_M$. The goal is to find a low dimensional embedding (a new coordinates system) that reveals the latent structure (e.g. metastable behaviour). Here $\mathcal{G}_{i+1} = \mathbf{T}(\mathcal{G}_i)$, but $\mathbf{T}(.)$ is inaccessible.

Discrete dynamical systems: $\mathbf{z}_{i+1} = \mathbf{T}(\mathbf{z}_i)$

Example (Neural network training as a dynamical system)

Consider a neural network $n : X \times \mathbb{R}^n \longrightarrow \mathbb{R}^d, \ \ (\mathbf{x}, w) \longrightarrow n(\mathbf{x}; w)$, where $\mathbf{x}$ is the input feature vector, w is the vector of network weights (parameters), and $n(\mathbf{x}; w)$ is the output. The weights are determined by minimizing the loss function $L_{tr}(w)$ over the training data. The deployed optimization algorithm (e.g. stochastic gradient descent) can be represented as an iterative process $w_{t+1} = \mathbf{T}(w_t)$. The mapping $\mathbf{T}(.)$ is black–boxed. Goal: learn $\mathbf{T}(.)$ from data and e.g. prune the network (selectively set weights to zero).

Example

Video processing: foreground-background separation Recorded video is a sequence of frames that can be interpreted as data snapshots recorded with fixed time lag. The goal is to separate the static background from the dynamics on the video.

Setting the scene: DS and the Koopman operator

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \equiv \begin{pmatrix} \mathbf{F}_1(\mathbf{x}(t)) \\ \vdots \\ \mathbf{F}_N(\mathbf{x}(t)) \end{pmatrix}, \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{3}$$

The solution formula (the flow map) is

$$\mathbf{x}(t_0 + t) = \boldsymbol{\varphi}^t(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{F}(\mathbf{x}(\tau))d\tau. \tag{4}$$

Observables

The state may not be accessible. Instead, we have observables (indirect measurements of the state) $f : \mathcal{X} \to \mathbb{C}$, $f \in \mathcal{F}$; e.g. $\mathcal{F} \subseteq L^2(\mathcal{X}, \mu)$.

Koopman operator semigroup $(\mathcal{K}_t)_{t\geq 0}$

$$\mathcal{K}_t f = f \circ \boldsymbol{\varphi}^t, \quad f \in \mathcal{F}. \tag{5}$$

Setting the scene: DS and the Koopman operator

$\mathcal{K}_t$ is linear operator

Indeed, let α, β be scalars, f, g observables. Recall,

$$(\alpha f + \beta g)(\boldsymbol{\varphi}^t(x)) = \alpha\, f(\boldsymbol{\varphi}^t(x)) + \beta\, g(\boldsymbol{\varphi}^t(x)).$$

Hence,

$$\begin{aligned} \mathcal{K}_t(\alpha f + \beta g) &= (\alpha f + \beta g) \circ \boldsymbol{\varphi}^t = \alpha\,(f \circ \boldsymbol{\varphi}^t) + \beta\,(g \circ \boldsymbol{\varphi}^t) \\ &= \alpha\, \mathcal{K}_t f + \beta\, \mathcal{K}_t g. \end{aligned}$$

Further, $\boldsymbol{\varphi}^{t_1+t_2} = \boldsymbol{\varphi}^{t_1} \circ \boldsymbol{\varphi}^{t_2} = \boldsymbol{\varphi}^{t_2} \circ \boldsymbol{\varphi}^{t_1}$

$$\boldsymbol{\varphi}^{t_1+t_2}(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t_1+t_2} \mathbf{F}(\mathbf{x}(\tau))d\tau = \boldsymbol{\varphi}^{t_1}(\boldsymbol{\varphi}^{t_2}(\mathbf{x}(t_0)))$$

This implies the (semi)group property:

$$\mathcal{K}_{t_1+t_2} f = f \circ \boldsymbol{\varphi}^{t_1+t_2} = f \circ \boldsymbol{\varphi}^{t_1} \circ \boldsymbol{\varphi}^{t_2} = \mathcal{K}_{t_2}(f \circ \boldsymbol{\varphi}^{t_1}) = \mathcal{K}_{t_2}\mathcal{K}_{t_1} f$$

Example: Koopman eigenpairs for linear systems

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \equiv A\mathbf{x}(t), \quad A = \begin{pmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \dots & \vdots \\ a_{N1} & \dots & a_{NN} \end{pmatrix}, \quad \mathbf{x}(t_0) = \mathbf{x}_0,$$

Solution: $\mathbf{x}(t_0 + t) = e^{tA}\mathbf{x}(t_0)$. Suppose A is diagonalizable, $A = S\Lambda S^{-1}$, $\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_N)$, $S = (\mathbf{s}_1, \dots, \mathbf{s}_N)$, $A\mathbf{s}_j = \lambda_j \mathbf{s}_j$; $\lambda_j = \alpha_j + \mathbf{i}\beta_j \in \mathbb{C}$ eigenvalue; $\mathbf{s}_j \in \mathbb{C}^N$ eigenvector.

$$A = S\Lambda S^{-1} \Longleftrightarrow A^* = S^{-*}\Lambda^* S^* = W\Lambda^* W^{-1}, \; W = S^{-*};$$

$W^*S = I$; $\mathbf{w}_j^* \mathbf{s}_k = \boldsymbol{\delta}_{jk}$ $(\mathbf{w}_j = W(:, j))$; $\Lambda^* = \mathrm{diag}(\overline{\lambda}_1, ..., \overline{\lambda}_N)$.
Then for any $\mathbf{x}$

$$\mathbf{x} = SS^{-1}\mathbf{x} = SW^*\mathbf{x} = \sum_{j=1}^{N} (\mathbf{w}_j^* \mathbf{x})\mathbf{s}_j = \sum_{j=1}^{N} \psi_j(\mathbf{x})\mathbf{s}_j,$$

where $\psi_j(\mathbf{x}) = \mathbf{w}_j^* \mathbf{x} = \langle \mathbf{x}, \mathbf{w}_j \rangle$.

Example: Koopman eigenpairs for linear systems

Consider now $\psi_j(\mathbf{x}) = \mathbf{w}_j^* \mathbf{x} = \langle \mathbf{x}, \mathbf{w}_j \rangle$. Let us apply $\mathcal{K}_t$:

$$(\mathcal{K}_t \psi_j)(\mathbf{x}_0) = \psi_j(\mathbf{x}(t;\mathbf{x}_0)) = \langle \mathbf{x}(t;\mathbf{x}_0), \mathbf{w}_j \rangle$$

What do we know about $\psi_j(\mathbf{x}(t;\mathbf{x}_0))$?

$$\begin{aligned}
\frac{d}{dt}\psi_j(\mathbf{x}(t;\mathbf{x}_0)) &= \frac{d}{dt}\langle \mathbf{x}(t;\mathbf{x}_0), \mathbf{w}_j \rangle = \langle \frac{d}{dt}\mathbf{x}(t;\mathbf{x}_0), \mathbf{w}_j \rangle \\
&= \langle A\mathbf{x}(t;\mathbf{x}_0), \mathbf{w}_j \rangle = \langle \mathbf{x}(t;\mathbf{x}_0), A^*\mathbf{w}_j \rangle \\
&= \langle \mathbf{x}(t;\mathbf{x}_0), \overline{\lambda}_j \mathbf{w}_j \rangle = \lambda_j \langle \mathbf{x}(t;\mathbf{x}_0), \mathbf{w}_j \rangle \\
&= \lambda_j \psi_j(\mathbf{x}(t;\mathbf{x}_0))
\end{aligned}$$

Hence, $\psi_j(\mathbf{x}(t;\mathbf{x}_0)) = e^{\lambda_j t}\psi_j(\mathbf{x}_0)$.

Eigenpairs of $\mathcal{K}_t$

$$(\mathcal{K}_t \psi_j)(\mathbf{x}_0) = e^{\lambda_j t}\psi_j(\mathbf{x}_0)$$

Example: Koopman eigenpairs for linear systems

$$\mathbf{x}_0 = SS^{-1}\mathbf{x}_0 = SW^*\mathbf{x}_0 = \sum_{j=1}^{N}(\mathbf{w}_j^*\mathbf{x}_0)\mathbf{s}_j = \sum_{j=1}^{N}\psi_j(\mathbf{x}_0)\mathbf{s}_j$$

A spectral representation of the action of $\mathcal{K}_t$ is as follows:

$$\begin{aligned}
(\mathcal{K}_t\mathbf{x})(\mathbf{x}_0) &= \mathbf{x}(t;\mathbf{x}_0) = e^{At}\mathbf{x}_0 = \sum_{j=1}^{N}\langle e^{At}\mathbf{x}_0, \mathbf{w}_j\rangle \mathbf{s}_j \\
&= \sum_{j=1}^{N}\langle \mathbf{x}_0, e^{A^*t}\mathbf{w}_j\rangle \mathbf{s}_j = \sum_{j=1}^{N}\langle \mathbf{x}_0, e^{\overline{\lambda}_j t}\mathbf{w}_j\rangle \mathbf{s}_j \\
&= \sum_{j=1}^{N} e^{\lambda_j t}\langle \mathbf{x}_0, \mathbf{w}_j\rangle \mathbf{s}_j = \sum_{j=1}^{N} e^{\lambda_j t}\psi_j(\mathbf{x}_0)\mathbf{s}_j \\
&= \sum_{j=1}^{N}\psi_j(\mathbf{x}(t;\mathbf{x}_0))\mathbf{s}_j;\ (\mathcal{K}_t C\mathbf{x})(\mathbf{x}_0) = \sum_{j=1}^{N} e^{\lambda_j t}\psi_j(\mathbf{x}_0)C\mathbf{s}_j.
\end{aligned}$$

Setting the scene: DS and the Koopman operator

Consider a discrete dynamical system $\mathbf{z}_{i+1} = \mathbf{T}(\mathbf{z}_i)$, 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 $\mathcal{K} \equiv \mathcal{K}_{\mathbf{T}}$ for the discrete system is defined analogously by

$$\mathcal{K}f = f \circ \mathbf{T}, \quad f \in \mathcal{F} \subseteq L^p(\mathcal{X}, \mu). \tag{6}$$

It is tacitly assumed that $\mathbf{T}$ is regular with respect to the measure μ:

$$\mu(S) = 0 \Longrightarrow \mu(\mathbf{T}^{-1}(S)) = 0.$$

This ensures that $\mu(f_1 \neq f_2) = 0 \Longrightarrow \mu(f_1 \circ \mathbf{T} \neq f_2 \circ \mathbf{T}) = 0$.

Operator theoretic issues (such as e.g. choosing the function space of observables, approximations from finite dimensional subspaces) are delicate and are beyond the scope of this course. The important thing is that they are properly treated in the corresponding theoretical frameworks.

Setting the scene: DS and the Koopman operator

If we run a numerical simulation of the ODE's (3) 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 $\mathbf{z}_i = \mathbf{T}(\mathbf{z}_{i-1})$ that produces the discrete sequence of $\mathbf{z}_i \approx \mathbf{x}(t_i)$; this is sampling with noise.
For $t_i = t_0 + i\Delta t$ we have (using $\boldsymbol{\varphi}^{\Delta t}$, $\mathcal{K}_{\Delta t}$ and the group property)

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

where $\mathcal{K}^i_{\Delta t} = \mathcal{K}_{\Delta t} \circ \ldots \circ \mathcal{K}_{\Delta t}$.

Setting the scene: DS and the Koopman operator

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

$$f(\mathbf{z}_i) = f(\mathbf{T}(\mathbf{z}_{i-1})) = \ldots = f(\mathbf{T}^i(\mathbf{z}_0)) = (\mathcal{K}^i f)(\mathbf{z}_0), \tag{7}$$

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 (3) with the initial condition $\mathbf{z}_0 = \mathbf{x}(t_0)$, we have an approximation

$$(\mathcal{K}^i f)(\mathbf{z}_0) \approx (\mathcal{K}^i_{\Delta t} f)(\mathbf{z}_0), \quad f \in \mathcal{F},\ \mathbf{z}_0 \in \mathcal{X}, \quad i = 0, 1, 2, \ldots \tag{8}$$

This can be obviously extended to vector valued observables:

$$\text{for } \mathbf{g} = (g_1, \ldots, g_d) : \mathcal{X} \longrightarrow \mathbb{C}^d \text{ define } \mathcal{K}_d \mathbf{g} = \begin{pmatrix} g_1 \circ \mathbf{T} \\ \vdots \\ g_d \circ \mathbf{T} \end{pmatrix} = \begin{pmatrix} \mathcal{K} g_1 \\ \vdots \\ \mathcal{K} g_d \end{pmatrix}. \tag{9}$$

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 $d = N$, $g_i(\mathbf{z}) = e_i^T \mathbf{z}$, where $\mathbf{z} \in \mathbb{C}^N$, $e_i = (\boldsymbol{\delta}_{ji})_{j=1}^N$, $i = 1, \ldots, N$, then $\mathbf{g}(\mathbf{z}) = \mathbf{z}$ is full state observable and $(\mathcal{K}_d \mathbf{g})(\mathbf{z}_i) = \mathbf{z}_{i+1}$.

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, Particle Image Velocimetry/Thermometry
- numerical simulation of (3) represented by (18), (7), (8), where we can feed an initial $\mathbf{z}_0$ to a software tool (representing $\mathbf{T}$, or its linearization through a numerical scheme encoded in a software toolbox) to obtain the sequence $\mathbf{f}(\mathbf{z}_0) = (\mathcal{K}_d^0\mathbf{f})(\mathbf{z}_0)$,

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

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

Data driven framework: data snapshots

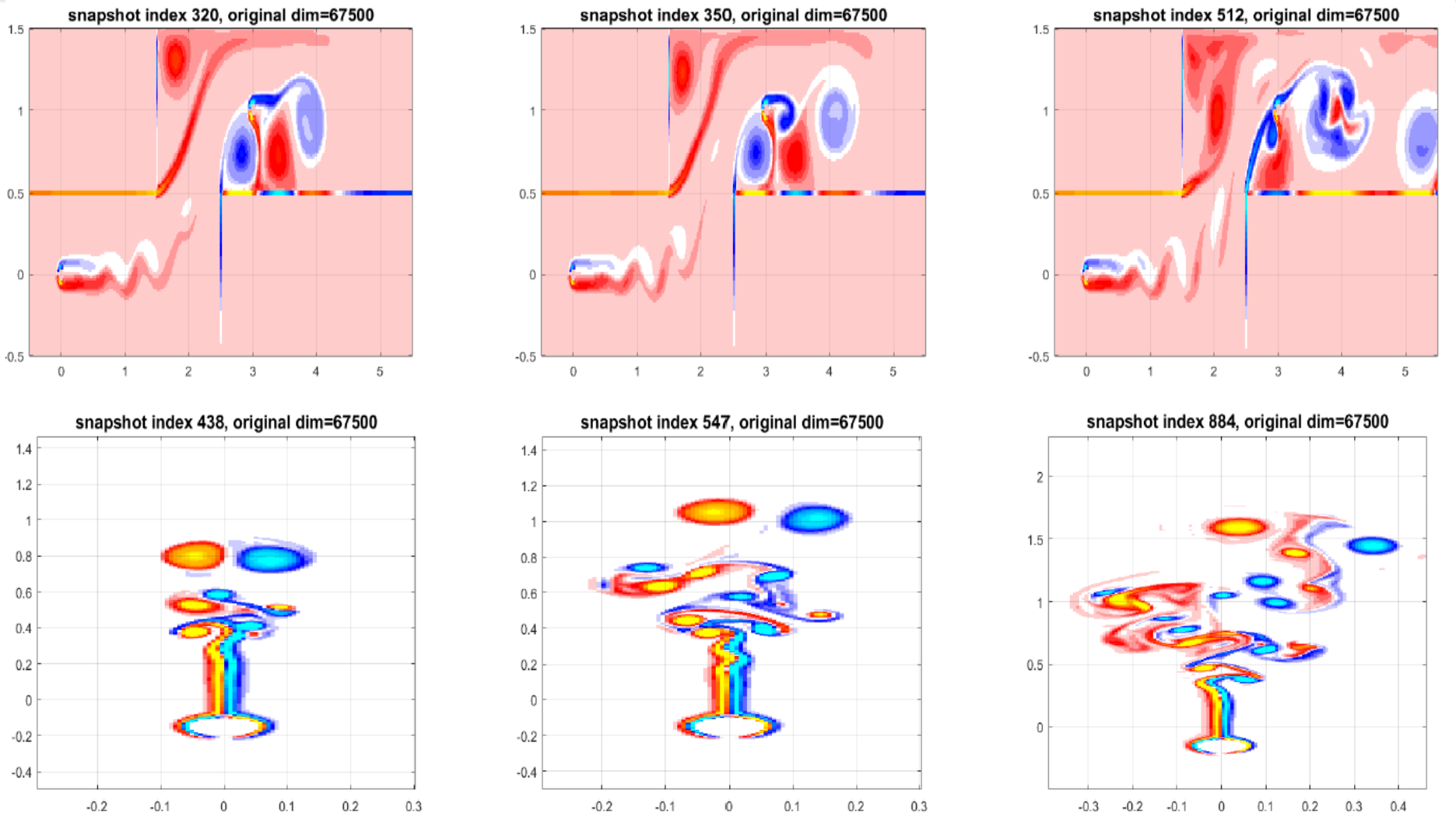
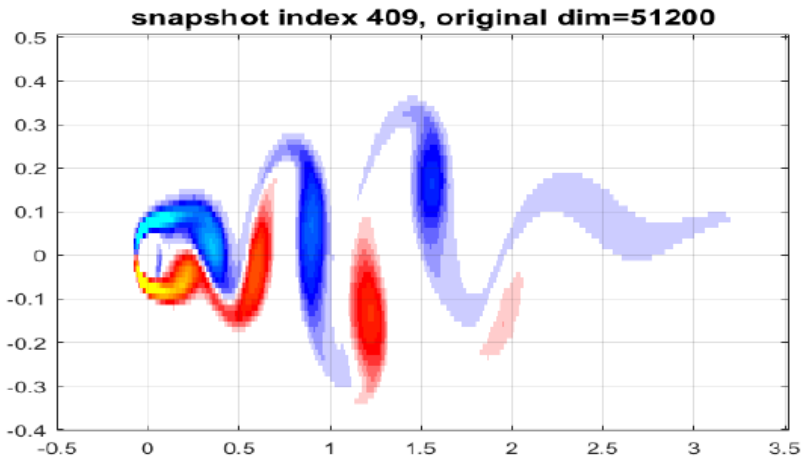


Figure: Vorticity field data snapshots.(Data source: Popinet 2004, Baeza Rojo and Günther 2019.)

Data driven framework: data snapshots



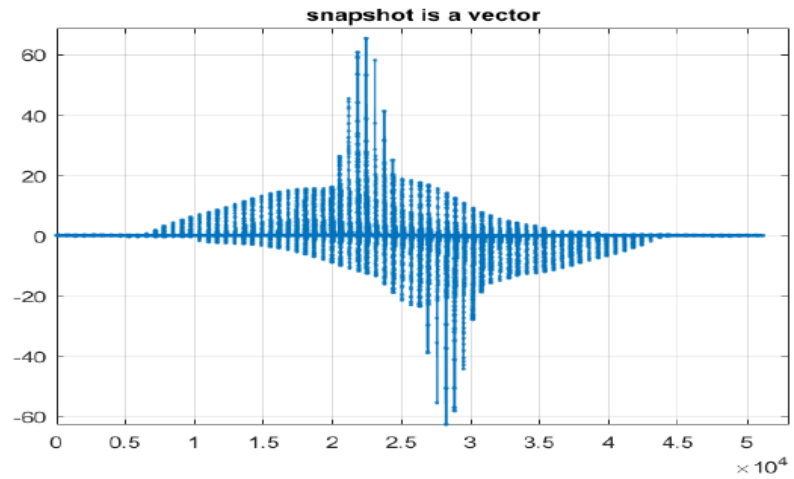


Figure: Snapshots are vectors, that are vector functions $\mathbf{f}(\mathbf{z}_i)$ of the states $\mathbf{z}_i = \mathbf{x}(t_i)$, e.g. $(\mathbf{f}(\mathbf{z}_i))_j = f_j(\mathbf{z}_i) = j$th component of $\mathbf{x}(t_i)$, but can use more general function (embedding) to ensure better mathematical properties.

$$\mathbf{S} = \begin{pmatrix} \mathbf{f}(\mathbf{z}_0) & \mathbf{f}(\mathbf{z}_1) & \dots & \mathbf{f}(\mathbf{z}_M) & \mathbf{f}(\mathbf{z}_{M+1}) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{z}_0) & f_1(\mathbf{z}_1) & \dots & f_1(\mathbf{z}_M) & f_1(\mathbf{z}_{M+1}) \\ f_2(\mathbf{z}_0) & f_2(\mathbf{z}_1) & \dots & f_2(\mathbf{z}_M) & f_2(\mathbf{z}_{M+1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_d(\mathbf{z}_0) & f_d(\mathbf{z}_1) & \dots & f_d(\mathbf{z}_M) & f_d(\mathbf{z}_{M+1}) \end{pmatrix}.$$

Column index :: discrete time steps counter.

Data driven framework

Can run many simulations with different initial conditions, physical parameters and have an abundance of data (large dimensional data matrices). To what end? What are the goals?

1. Understand the data: reveal latent structure.
2. Devise a low-dimensional approximation for faster numerical simulations (e.g. for online applications, or optimization over a parameter domain, digital twin design).
3. Develop forecasting skill.
4. Use for control (Model Predictive Control, MPC).
5. Discover governing equations.
6. Optimal sensor placement in a physical domain.

Data driven framework

Example (Time-evolving graph: Melnyk, Klus, Montavon, Conrad 2020)

Suppose we are given a time-evolving graph $\mathcal{G} = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_M)$ with adjacency matrices $A_1, A_2, \ldots, A_M$. The goal is to find a low dimensional embedding that reveals the latent structure (e.g. metastable behaviour).

$\mathcal{G}_{i+1} = \mathbf{T}(\mathcal{G}_i)$; $\mathcal{K}f = f \circ \mathbf{T}$:: $(\mathcal{K}f)(\mathcal{G}_i) = f(\mathcal{G}_{i+1})$.

Observable $\mathbf{f} = (f_1, \ldots, f_d)^T$ maps the data to higher (including infinitely) dimensional Hilbert space $(\mathbb{H}, \langle \cdot, \cdot \rangle)$ using the kernel trick. Suitable kernel function $k(\mathcal{G}_i, \mathcal{G}_j)$ defines the function $\mathbf{f}$ and the inner product in $\mathbb{H}$ implicitly by

$$\langle \mathbf{f}(\mathcal{G}_i), \mathbf{f}(\mathcal{G}_j) \rangle = k(\mathcal{G}_i, \mathcal{G}_j) = K_{ij}.$$

Method: Approximate a compression of $\mathcal{K}$ onto the subspace spanned by $f_1, \ldots, f_d$ and use its selected eigenfunctions as a basis for new coordinates. An example of the kernel is the Gaussiam kernel

$$k(\mathcal{G}_i, \mathcal{G}_j) = \mathbf{exp}(-\|A_i - A_j\|^2/(2\sigma^2))$$

Data driven framework

Example (Power networks: Susuki, Mezić, Raak, Hikihara)

Model–free precursor diagnostic of instabilities in power networks.
Available data are snapshots $P_1, P_2, \ldots$ of physical power flow variables (e.g. voltage magnitudes and angles) at discrete time steps at m measurement sites (each P_i is $m \times 1$) such as generation plants, substations etc.
The P_i's are determined by internal states x_i of a power system (rotating frequencies and voltages of AC generators, states of controllers in plants and substations etc.) that are assumed to change as $x_{i+1} = \mathbf{T}(x_i)$.
Again, there is associated composition operator $\mathcal{K}f = f \circ \mathbf{T}$.

Base flow patterns as coherent spatial units of power flows identified as spanned by eigenfunctions of $\mathcal{K}$.
The 2006 European interconnected grid disturbance clearly visible in unstable modes of $\mathcal{K}$ (eigenvalues outside unit circle.)

Data driven spectral analysis - applications

Other successful applications of DMD include e.g.

- aeroacoustics
- computational fluid dynamics
- affective computing (analysis of videos for human emotion recognition)
- robotics (filtering external perturbation using DMD based prediction)
- algorithmic trading on financial markets
- analysis of infectious disease spread
- neuroscience
- data driven learning and control of UAV's (drones)

Theoretical contributions and applications by S. Brunton, M. Colbrook, M. Korda, N. Kutz, A. Mauroy, I. Mezić, P. Schmid, A. Townsend, ...

Data driven framework

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

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

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

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

$$\mathbb{A}\mathbf{f}(\mathbf{z}_k) = (\mathcal{K}_d\mathbf{f})(\mathbf{z}_k) = \begin{pmatrix} (\mathcal{K}f_1)(\mathbf{z}_k) \\ \vdots \\ (\mathcal{K}f_d)(\mathbf{z}_k) \end{pmatrix} = \mathbf{f}(\mathbf{T}(\mathbf{z}_k)), \quad k = 0, \dots, M. \qquad (10)$$

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 $\mathbb{A}(\mathbf{X}v) = \mathbf{Y}v$, $v \in \mathbb{C}^{M+1}$. Define $\mathbb{A}$ to solve $\|\mathbf{Y} - \mathbb{A}\mathbf{X}\|_F \to \min$, e.g. $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$. $\mathbb{A}$ may not be unique!

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 $\mathbf{x}_k = \mathbf{f}(\mathbf{z}_k)$, $\mathbf{y}_k = \mathbf{f}(\mathbf{T}(\mathbf{z}_k))$ column-wise so that a kth column in $\mathbf{Y}$ corresponds to the value of the observable in the kth column of $\mathbf{X}$ through the action of $\mathcal{K}_d$.

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 $\mathbf{X}$.

Compression of $\mathcal{K}$

Read the information in the snapshots matrix row-wise:

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

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

Consider the action of $\mathcal{K}$ on the space $\mathcal{F}_\mathcal{D}$ spanned by the dictionary of scalar functions $\mathcal{D} = \{f_1, \dots, f_d\}$. That is, we seek a matrix representation $\mathbb{U}$ of the compression $\mathbf{\Psi}_{\mathcal{F}_\mathcal{D}} \mathcal{K}_{|\mathcal{F}_\mathcal{D}} : \mathcal{F}_\mathcal{D} \longrightarrow \mathcal{F}_\mathcal{D}$, 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 $\mathcal{K} f_i$ of the form

$$(\mathcal{K} f_i)(s) = f_i(\mathbf{T}(s)) = \sum_{j=1}^{d} \mathbf{u}_{ji} f_j(s) + \rho_i(s), \quad i = 1, \dots, d, \quad s \in \mathcal{X}. \quad (11)$$

Compression of $\mathcal{K}$

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

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

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

$$(\mathcal{K}f_i)(\mathbf{z}_k) = f_i(\mathbf{T}(\mathbf{z}_k)), \quad i = 1, \ldots, d; \quad k = 0, \ldots, M. \tag{12}$$

To that end, write the least squares residual

$$\frac{1}{M+1} \sum_{k=0}^{M} |\rho_i(\mathbf{z}_k)|^2 = \frac{1}{M+1} \sum_{k=0}^{M} |\sum_{j=1}^{d} \mathbf{u}_{ji} f_j(\mathbf{z}_k) - f_i(\mathbf{T}(\mathbf{z}_k))|^2, \tag{13}$$

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

Compression of $\mathcal{K}$

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

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

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,

$$\mathbb{U} = (\mathbf{X}^T)^\dagger \mathbf{Y}^T \equiv (\mathbf{Y}\mathbf{X}^\dagger)^T = \mathbb{A}^T, \tag{14}$$

and the action of $\mathcal{K}$ can be represented, using (11), as

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

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

Compression of $\mathcal{K}$

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

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

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

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

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

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

Koopman modes – decomposition

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

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

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

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

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

$$(\mathcal{K}_d^k \mathbf{g})(s) = \begin{pmatrix} (\mathcal{K}^k g_1)(s) \\ \vdots \\ (\mathcal{K}^k g_d)(s) \end{pmatrix} \approx \sum_{i=1}^{d} \mathbf{z}_i \phi_i(s) \lambda_i^k. \tag{15}$$

$\mathbb{A}^T = \mathbb{U} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$ implies $\mathbb{A}\mathbf{Q}^{-T} = \mathbf{Q}^{-T}\Lambda$, i.e. the columns of $\mathbf{Q}^{-T}$ are the (right) eigenvectors of $\mathbb{A}$. Hence, for computing the Koopman modes, we can proceed with computing the eigenvectors of $\mathbb{A}$.

DMD: A data driven spectral analysis

Now we can formulate pure matrix computational problem: 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 $\mathbb{A}$;

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

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

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

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

$$\mathbb{A}z_j \approx \lambda_j z_j, \quad \lambda_j = |\lambda_j| e^{\mathrm{i}\omega_j \delta t}, \quad j = 1, \ldots, k; \quad k \leq m, \tag{2}$$

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

$$\mathbf{f}_i \approx \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j \lambda_{\varsigma_j}^{i-1} \equiv \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j |\lambda_{\varsigma_j}|^{i-1} e^{\mathrm{i}\omega_{\varsigma_j}(i-1)\delta t}, \quad i = 1, \ldots, m. \tag{3}$$

Tool: Krylov subspaces

- For $\mathbf{f}_{i+1} = \mathbb{A}\mathbf{f}_i$, $i = 1, 2, \ldots, m$, define the Krylov matrices

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

  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

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

  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

$$\|\mathbf{f}_{m+1} - \mathbf{X}_m c\|_2 \longrightarrow \min_c \tag{4}$$

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 $E_{m+1} = r_{m+1} e_m^T$, $e_m = \big(0, \ldots, 0, 1\big)^T$. 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

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

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

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]).

Beautiful structure and bad news

The Ritz vectors corresponding to the λ_j's are then the columns of

$$\mathsf{W}_m \equiv \mathbf{X}_m \mathbb{V}_m^{-1} \equiv (w_1 \ \ldots \ w_m). \tag{5}$$

If we define $\alpha_j = \|w_j\|_2$, $z_j = w_j/\alpha_j$, $Z_m = (z_1 \ \ldots \ z_m)$, then the decomposition (6), with $\ell = m$, follows from

$$\begin{aligned}
\mathbf{X}_m &= \mathsf{W}_m \mathbb{V}_m = Z_m \mathrm{diag}(\alpha_j)_{j=1}^m \mathbb{V}_m \\
&= \begin{pmatrix} z_1 & z_2 & \ldots & z_m \end{pmatrix} \begin{pmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_m \end{pmatrix} \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \ldots & \lambda_2^{m-1} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & \lambda_m & \ldots & \lambda_m^{m-1} \end{pmatrix}.
\end{aligned}$$

Ill-conditioning of Vandermonde matrices: examples

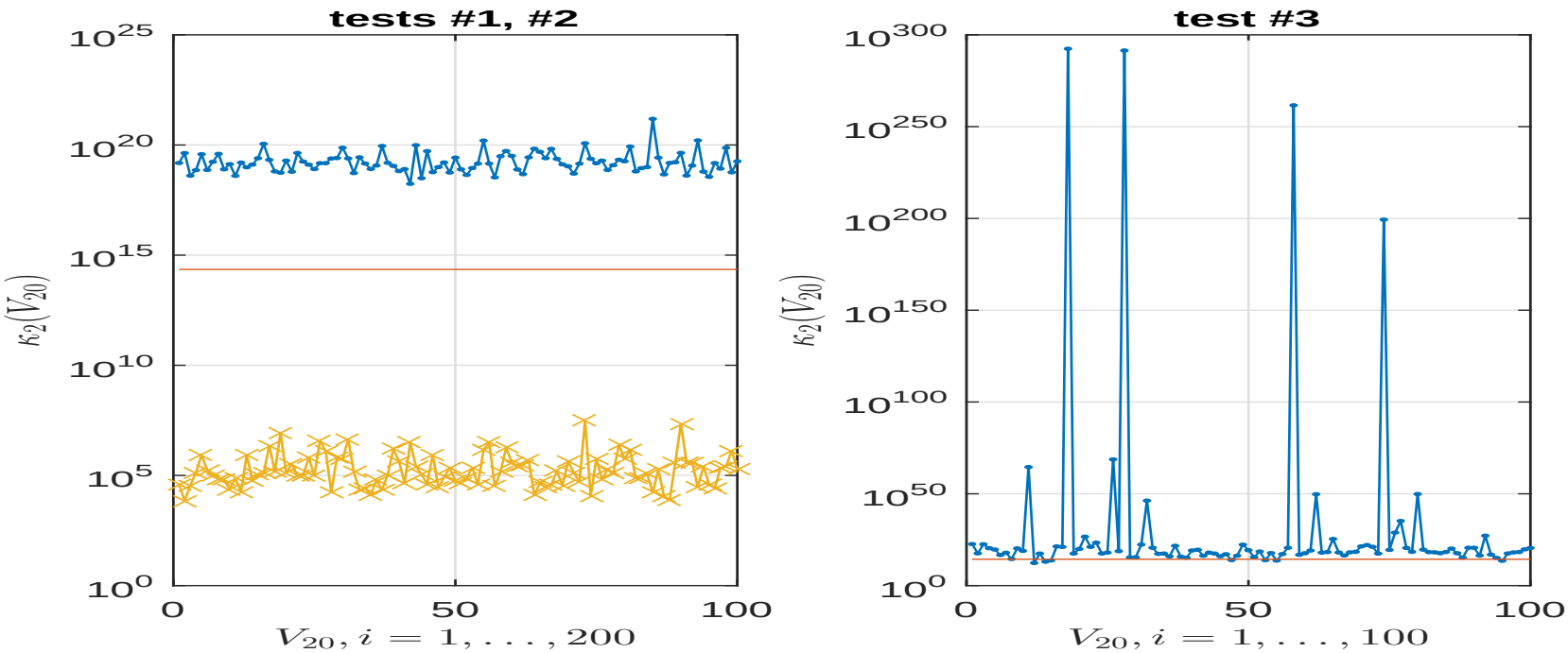


Figure: The spectral condition number over three sets of the total of 300 Vandermonde matrices of dimension $m = 20$, $\mathbb{V}_{20}(\lambda_i)$; $(\lambda_i) = \texttt{eig}(A)$. *Left panel*: First, 100 matrices are generated in Matlab as $A = \texttt{rand(m,m)}$, $A = A/\texttt{max(abs(eig(A)))}$. Then, 100 matrices are generated as $A = \texttt{randn(m,m)}$, $A = A/\texttt{max(abs(eig(A)))}$. *Right panel*: 100 samples of $\mathbb{V}(\lambda_i)$ are generated using the eigenvalues of $A = \texttt{expm(-inv(rand(m,m)))}$, $A = A/\texttt{max(abs(eig(A)))}$. The horizontal line marks $1/(m\varepsilon) \approx \texttt{2.25e+14}$.

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 $H_{ij} = 1/(i+j-1)$ range from $1/199 \approx 5.025 \cdot 10^{-3}$ to 1, and $\kappa_2(H) > 10^{\mathbf{150}}$.

One should keep in mind that the matrix condition number is a matrix function $f(A) = \|A\|\|A^\dagger\|_2$, with its own condition number. By a result of Higham, condition number of the condition number is the condition number itself.

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

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

If the computed/estimated condition number is above $1/\texttt{eps}$ (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.

Exercise

Exercise

We used the fact that for the companion matrix

$$C_m = \begin{pmatrix} 0 & 0 & \dots & 0 & c_1 \\ 1 & 0 & \dots & 0 & c_2 \\ 0 & 1 & \dots & 0 & c_3 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_m \end{pmatrix}$$

with algebraically simple eigenvalues $\lambda_1, \dots, \lambda_m$, the spectral decomposition reads

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

Prove that! Use the connection between the companion matrix and polynomials.

Rayleigh-Ritz extraction is better with ONB

To extract spectral information on $\mathbb{A}$ from the span of $\mathbf{X}_m$, it is preferable to work with an orthonormal basis for the range of $\mathbf{X}_m$.

Orthonormalization via Gram-Schmidt, or the QR factorization

1. $\mathbf{X}_m = QR$. Since $\mathbf{X}_m$ is ill–conditioned Gram–Schmidt must be carefully implemented to make Q numerically orthogonal. Direct QR factorization (e.g. using Householder reflectors) is an alternative. In any case, R is ill–conditioned.
2. From $\mathbb{A}\mathbf{X}_m = \mathbf{Y}_m = \mathbb{A}QR$, $Q^*\mathbb{A}Q = Q^*\mathbf{Y}_m R^{-1}$. Since R is ill–conditioned the data-driven formula for the Rayleigh quotient is badly conditioned.
3. $\mathbf{X}_m$ can be numerically rank-deficient and noisy, so using R^{-1} is ill-advised.

Why is $\mathbf{X}_m$ expected to be ill–conditioned

The matrix $\mathbf{X}_m$ can be nearly rank defficient. To illustrate, assume that $\mathbb{A}$ is diagonalizable with eigenpairs $\mathbb{A}\mathbf{a}_i = \alpha_i \mathbf{a}_i$, and that its eigenvalues α_i are enumerated so that $0 \neq |\alpha_1| \geq |\alpha_2| \geq \cdots \geq |\alpha_n|$. Let $\mathbf{f}_1$ be expressed in the eigenvector basis as $\mathbf{f}_1 = \phi_1 \mathbf{a}_1 + \cdots + \phi_n \mathbf{a}_n$. Then

$$\mathbf{f}_{i+1} = \mathbb{A}^i \mathbf{f}_1 = \alpha_1^i \left(\phi_1 \mathbf{a}_1 + \left(\frac{\alpha_2}{\alpha_1} \right)^i \phi_2 \mathbf{a}_2 + \left(\frac{\alpha_3}{\alpha_1} \right)^i \phi_3 \mathbf{a}_3 + \cdots + \left(\frac{\alpha_n}{\alpha_1} \right)^i \phi_n \mathbf{a}_n \right).$$

Hence, if e.g. $|\alpha_1| \geq |\alpha_2| > |\alpha_3|$, then for $j \geq 3$, $\lim_{i\to\infty} (\alpha_j/\alpha_1)^i = 0$, and thus, with big enough i the $\mathbf{f}_i$'s will stay close to the span of $\mathbf{a}_1$ and $\mathbf{a}_2$, provided that $\phi_1 \neq 0$, $\phi_2 \neq 0$. This means that relatively small changes of $\mathbf{X}_m$ can make it rank deficient; its range may change considerably under tiny perturbations. In the context of spectral approximations, this is desirable and we hope that the $\mathbf{f}_i$'s will become numerically linearly dependent as soon as possible; on the other hand we must stay vigilant in computing with $\mathbf{X}_m$ and $\mathbf{Y}_m$ as numerical detection of rank deficiency in the presence of noise is a delicate issue.

Nearly rank deficient and ill-conditioned – a connection

Suppose $\mathbf{X}_m$ is subject to a perturbation $\delta\mathbf{X}_m$, $\mathbf{X}_m \rightsquigarrow \widetilde{\mathbf{X}_m} = \mathbf{X}_m + \delta\mathbf{X}_m$. If $\widetilde{\mathbf{X}_m}$ is rank deficient, what can be said about the size of $\delta\mathbf{X}_m$? This will be answered in detail using the SVD decomposition, but it is instructive to analyze this directly. Since $\mathbf{X}_m$ is rectangular of full column rank, its generalized inverse is $\mathbf{X}_m^\dagger = (\mathbf{X}_m^*\mathbf{X}_m)^{-1}\mathbf{X}_m^*$, so that $\mathbf{X}_m^\dagger\mathbf{X}_m = \mathbb{I}_m$. Write

$$\widetilde{\mathbf{X}_m} = (\mathbb{I}_n + \delta\mathbf{X}_m\mathbf{X}_m^\dagger)\mathbf{X}_m.$$

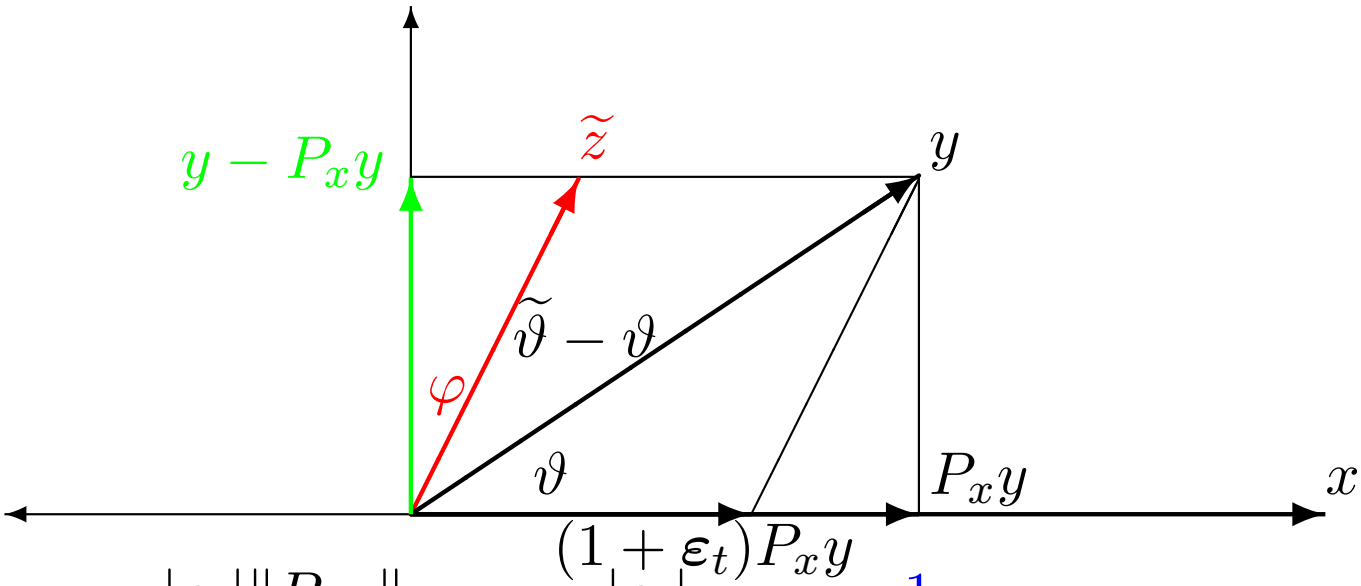
Then $\|\delta\mathbf{X}_m\mathbf{X}_m^\dagger\| \geq 1$ in any matrix norm $\|\cdot\|$. (Otherwise, $\|\delta\mathbf{X}_m\mathbf{X}_m^\dagger\| < 1$, then $\mathrm{spr}(\delta\mathbf{X}_m\mathbf{X}_m^\dagger) < 1$ and $\mathbb{I}_n + \delta\mathbf{X}_m\mathbf{X}_m^\dagger$ would be nonsingular, and $\widetilde{\mathbf{X}_m}$ of full column rank.) Hence, $1 \leq \|\delta\mathbf{X}_m\|\|\mathbf{X}_m^\dagger\|$, i.e.

$$\|\delta\mathbf{X}_m\| \geq \frac{1}{\|\mathbf{X}_m^\dagger\|}, \quad \text{i.e.} \quad \frac{\|\delta\mathbf{X}_m\|}{\|\mathbf{X}_m\|} \geq \frac{1}{\|\mathbf{X}_m\|\|\mathbf{X}_m^\dagger\|} \equiv \frac{1}{\kappa(\mathbf{X}_m)}. \tag{6}$$

$\kappa(\mathbf{X}_m) = \|\mathbf{X}_m\|\|\mathbf{X}_m^\dagger\|$ is the condition number of $\mathbf{X}_m$ in the norm $\|\cdot\|$.

Gram Schmidt: $z \equiv y - P_x y$, where $P_x y = (x^T y / x^T x)x$

$$\widetilde{t} = \frac{x^T y}{x^T x}(1+\varepsilon_t) = t(1+\varepsilon_t);\ z = y - tx;\ \widetilde{z} = y - \widetilde{t}x$$



$$\tan\varphi = \frac{|\varepsilon_t|\|P_x y\|_2}{\|y - P_x y\|_2} = \frac{|\varepsilon_t|}{\frac{\|y-P_x y\|_2}{\|P_x y\|_2}} = \frac{1}{\tan\vartheta}|\varepsilon_t|, \quad \tan\widetilde{\vartheta} = \frac{\tan\vartheta}{|\varepsilon_t|}. \qquad (7)$$

Nonorthogonality caused by ε_t depends on $1/\tan\vartheta$.

Now it should be clear that Gram–Schmidt on the columns of $\mathbf{X}_m$ is numerically not reliable.

SVD and best low rank approximation

Theorem

Eckart-Young-Mirsky Let the SVD of $M \in \mathbb{C}^{n\times m}$ *be*

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

For $k \in \{1, \ldots, \min(m,n)\}$, *define* $U_k = U(:, 1:k)$, $\Sigma_k = \Sigma(1:k, 1:k)$, $V_k = V(:, 1:k)$, *and* $M_k = U_k\Sigma_k V_k^*$. *Then*

$$\min_{\mathrm{rank}(N)\leq k} \|M - N\|_2 \quad = \quad \|M - M_k\|_2 = \sigma_{k+1} \tag{8}$$

$$\min_{\mathrm{rank}(N)\leq k} \|M - N\|_F \quad = \quad \|M - M_k\|_F = \sqrt{\sum_{i=k+1}^{\min(n,m)} \sigma_i^2}. \tag{9}$$

Hence, if $\sigma_m \ll \sigma_1$, the condition number $\kappa_2(\mathbf{X}_m) = \|\mathbf{X}_m\|_2\|\mathbf{X}_m^\dagger\|_2 = \frac{\sigma_1}{\sigma_m}$ is large, $\mathbf{X}_m$ can be made singular with a perturbation $\delta\mathbf{X}_m$ such that $\|\delta\mathbf{X}_m\|_2/\|\mathbf{X}_m\|_2 = \sigma_m/\sigma_1 = 1/\kappa_2(\mathbf{X}_m) \ll 1$.

Least squares problem: a review

The linear least squares problem is generically written as

$$\|Ax - b\|_2 \longrightarrow \min_x, \tag{10}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$.

It is a computational expression of the Gauss-Markov linear model $Ax_0 = b_0$, where $b = b_0 + e$ is assumed noisy measurement of an ideal b_0, and the measurement error vector e is assumed a random vector with expectation zero and variance $\sigma^2\mathbb{I}_m$ (the errors in each measurement (equation) are uncorrelated and with same variance).

By the Gauss–Markov theorem, if A is or rank n, the solution x of (10) is the best linear unbiased estimator of x_0.

In the case of full column rank matrix A, the solution is uniquely determined as the unique solution of a linear system of equation, derived using the projection theorem. If x is optimal, then the residual $Ax - b$ must be orthogonal to the range of A, i.e.

$$A^T(Ax - b) = \mathbf{0},$$

or, equivalently, the solution vector x is the unique solution of the so called normal equations

$$(A^T A)x = A^T b, \quad x = (A^T A)^{-1} A^T b.$$

The same set of equation is (of course) obtained by minimizing

$$F(x) = \|Ax - b\|_2^2$$

using calculus. In the numerical linear algebra, we are very careful and avoid using normal equations.

Läuchli's example

Example (Läuchli: Consider the least squares problem $\|Ax - b\|_2 \longrightarrow \min$)

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ \epsilon & 0 & 0 & 0 & 0 \\ 0 & \epsilon & 0 & 0 & 0 \\ 0 & 0 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & \epsilon & 0 \\ 0 & 0 & 0 & 0 & \epsilon \end{pmatrix}, \quad b = \begin{pmatrix} \epsilon \\ 0 \\ -5 \\ 5 \\ -5 \\ 0 \end{pmatrix}, \quad \text{where } \epsilon \in \mathbb{R} \text{ is such that } |\epsilon| \ll 1.$$

The normal equations matrix

$$H = A^T A = \begin{pmatrix} 1+\epsilon^2 & 1 & 1 & 1 & 1 \\ 1 & 1+\epsilon^2 & 1 & 1 & 1 \\ 1 & 1 & 1+\epsilon^2 & 1 & 1 \\ 1 & 1 & 1 & 1+\epsilon^2 & 1 \\ 1 & 1 & 1 & 1 & 1+\epsilon^2 \end{pmatrix} \tag{11}$$

has eigenvalues $\lambda_1 = 5 + \epsilon^2$, $\lambda_2 = \ldots = \lambda_5 = \epsilon^2$; these are also the singular values of H.

Läuchli's example

Example (... continued)

The singular values of A are thus $\sigma_1 = \sqrt{5+\epsilon^2}$, $\sigma_2 = \ldots = \sigma_5 = |\epsilon|$. According to the Eckart-Young-Mirsky's theorem, the minimal perturbations δA, δH that make, respectively, A and H singular are of sizes

$$\frac{\|\delta A\|_2}{\|A\|_2} = \frac{|\epsilon|}{\sqrt{5+\epsilon^2}}, \quad \frac{\|\delta H\|_2}{\|H\|_2} = \frac{\epsilon^2}{5+\epsilon^2}.$$

This corresponds to a relation between the condition numbers,

$$\kappa_2(A) = \frac{\sqrt{5+\epsilon^2}}{|\epsilon|}, \quad \kappa_2(H) = \frac{5+\epsilon^2}{\epsilon^2} = \kappa_2(A)^2.$$

The extreme case is when $|\epsilon|$ is so small that the floating point value of $1+\epsilon^2$ is 1, so that the computed matrix H stored in the computer memory is the rank–one matrix of all ones. To make the case even more difficult, the vector b is selected so that (for small $|\epsilon|$) is nearly orthogonal to the range of A.

Solving the LS problem using the SVD

Let $A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^*$ be the SVD of A with

$$\Sigma = \begin{pmatrix} \widehat{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \widehat{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{pmatrix}, \quad \sigma_1 \geq \cdots \geq \sigma_r > 0.$$

The rank of A is r and in the partitions $U = (U_r, U_{r\perp})$, $V = (V_r, V_{r\perp})$, U_r spans the range of A, and the columns of $V_{r\perp}$ are an orthonormal basis for the null-space.

This decomposition allows for a convenient change of variable so that the objective function of (10) becomes trivial to optimize:

$$\begin{aligned} \|Ax - b\|_2 &= \|U\Sigma(V^*x) - b\|_2 = \|\Sigma y - c\|_2 \quad (y = V^*x, c = U^*b) \\ &= \left\| \begin{pmatrix} \widehat{\Sigma} & \mathbf{0}_{r,n-r} \\ \mathbf{0}_{m-r,r} & \mathbf{0}_{m-r,n-r} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \right\|_2 \longrightarrow \min . \end{aligned}$$

Clearly, $y_1 = \widehat{\Sigma}^{-1} c_1$ and y_2 can be arbitrary $(n-r) \times 1$. Each y gives an optimal solution $x = Vy = V_r y_1 + V_{r\perp} y_2 = x_1 + x_2$, where $x_2 = V_{r\perp} y_2$ belongs to the null space of A. Note that $\|x\|_2^2 = \|x_1\|_2^2 + \|x_2\|_2^2$.

Solving the LS problem using the SVD

If $r = n$, then $\widehat{\Sigma} = \Sigma$, the null-space basis $V_{r\perp}$ is void and $x = x_1$ is uniquely determined. Otherwise, all possible solutions are a linear manifold $\mathcal{S} = V_r \widehat{\Sigma}^{-1} U_r^* b + \mathcal{N}(A)$. The shortest (in Euclidean norm) vector in $\mathcal{S}$ is

$$x = V_r \widehat{\Sigma}^{-1} U_r^* b, \tag{12}$$

that is obtained with the shortest y, i.e. with $y_2 = \mathbf{0}$ and $x_2 = \mathbf{0}$. Note that the expression (12) for x is linear in b. In the case of square full rank A, the optimal solution $x = A^{-1}b$ is expressed using the SVD as $x = V\Sigma^{-1}U^T b$, which is precisely (12). This motivates a generalization of the matrix inverse to the case of general rectangular matrices, of arbitrary ranks. The solution of (12), that is of minimal norm can be expressed as

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \widehat{\Sigma}^{-1} c_1 \\ \mathbf{0}_{n-r,1} \end{pmatrix} = \begin{pmatrix} \widehat{\Sigma}^{-1} & \mathbf{0}_{r,m-r} \\ \mathbf{0}_{n-r,r} & \mathbf{0}_{n-r,m-r} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \Sigma^\dagger c,$$

$$\Sigma^\dagger = \begin{pmatrix} \widehat{\Sigma}^{-1} & \mathbf{0}_{r,m-r} \\ \mathbf{0}_{n-r,r} & \mathbf{0}_{n-r,m-r} \end{pmatrix}.$$

Solving the LS problem. The Moore-Penrose pseudoinverse

The matrix $\Sigma^\dagger \in \mathbb{R}^{n\times m}$ is in a sense the best that can be done in mimicking the inverse of Σ. It represents a linear operator $\mathbb{R}^m \longrightarrow \mathbb{R}^n$ (hence of dimensions $n \times m$) that satisfies

$$\Sigma^\dagger\Sigma = \begin{pmatrix} \widehat{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \widehat{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbb{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{n\times n},$$

$$\Sigma\Sigma^\dagger = \begin{pmatrix} \widehat{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \widehat{\Sigma}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbb{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{m\times m}.$$

Using $\Sigma^\dagger$, the solution x from (12) can be written as

$$x = Vy = V_r\widehat{\Sigma}^{-1}U_r^T b = V\Sigma^\dagger U^T b = A^\dagger b, \tag{13}$$

where $A^\dagger = V\Sigma^\dagger U^T = V_r\widehat{\Sigma}^{-1}U_r^T$ is the Moore–Penrose generalized inverse of A. (In the case of complex matrix, $A^\dagger = V\Sigma^\dagger U^* = V_r\widehat{\Sigma}^{-1}U_r^*$.)

On using truncated SVD

The solution formula (13) assumes a clean cut in the SVD of A so that the rank of A is indisputable. Further, it is derived purely in the framework of linear algebra, ignoring the fact that in an application $b = b_0 + e$ is contaminated by noise, and that the true vector b_0 is not accessible. Writing (13) in the form ($U = (u_1, ..., u_m)$, $V = (v_1, ..., v_n)$)

$$x = V\Sigma^{\dagger}U^T b = \sum_{i=1}^{r} v_i \frac{u_i^T b}{\sigma_i} = \sum_{i=1}^{r} v_i \left(\frac{u_i^T b_0}{\sigma_i} + \frac{u_i^T e}{\sigma_i}\right)$$

reveals the structure of the solution that can be leveraged to improve the accuracy. Computational difficulties that are immediately observed are

1. The numerically computed SVD is used. If $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_n$ are the computed singular values, the smallest ones might be computed with large errors. (They are exact for some $A + \delta A$.)
2. The noise vector e might have large component $u_i^T e$ so that $u_i^T b_0$ is entirely overshadowed by noise. If the corresponding singular value in the denominator is small and computed with large error, then the solution vector has an inaccurate component in the direction of v_i.

Phillips's example

Example (Phillips's example: Fredholm integral equation of the first kind)

$$\mathsf{y}(\xi) = \int_a^b \mathsf{K}(\xi, \zeta)\mathsf{x}(\zeta)d\zeta.$$

Here y denotes a function that is available as a sequence of measurements at $\xi_1 < \cdots < \xi_m$, $y_i = y(\xi_i) + e_i$, where e_i is a measurement error. The integral with the known kernel $\mathsf{K}(\xi, \zeta)$ models measurement apparatus, and the goal is to find an approximation of the unknown function $\mathsf{x}(\zeta)$, so that

$$\int_a^b \mathsf{K}(\xi_i, \zeta)\mathsf{x}(\zeta)d\zeta \approx y_i = y(\xi_i) + e_i, \quad i = 1, \ldots, m.$$

The integral is computed using quadrature rule with the nodes $\zeta_1 < \cdots < \zeta_n$ and weights $d_1, \ldots, d_n$, and the task is to find $x_j \approx \mathsf{x}(\zeta_j)$

$$\sum_{j=1}^{n} d_j \mathsf{K}(\xi_i, \zeta_j)x_j \approx y_i + \epsilon_i = y(\xi_i) + e_i + \epsilon_i, \quad i = 1, \ldots, m, \tag{14}$$

where ϵ_i denotes the error in computing the integral.

Phillips's example

Example (Phillips's example ... continued)

It is clear that more measurements of y (adding more equations, i.e. larger m) provide more information on the unknown function, and that the equations cannot be satisfied exactly because of errors in the right hand side. Under a realistic assumption that $|e_i| \gg |\epsilon_i|$, the errors $e_i + \epsilon_i$ are dominated by e_i for all i. To write (14) more compactly, set

$$y = (y_i)_{i=1}^m, \quad K = (\mathsf{K}(\xi_i, \zeta_j)) \in \mathbb{R}^{m \times n}, x = (x_j)_{j=1}^n, \; D = \mathrm{diag}(d_j)_{j=1}^n.$$

Assuming sufficiently accurate quadrature formula, the errors ϵ_i are neglected and (14) becomes a linear regression model

$$KDx = y + e.$$

The error vector e is dominated by statistically independent measurement errors from $\mathcal{N}(\mathbf{0}, S^2)$, where the positive definite $S = \mathrm{diag}(s_i)_{i=1}^n$ carries standard deviations of the e_i's. A good estimate of S is usually available.

Phillips's example

Example (Phillips's example ... continued)

To normalize the error variances, the model is scaled with S^{-1} to get

$$b = Ax + e',$$

where

$$b = S^{-1}y, \quad A = S^{-1}KD, \quad e' = S^{-1}e.$$

Note that $e' \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_m)$.
Solving the linear system $Ax = b$ for x is illusory. A reasonable alternative is to make the residual $r = Ax - b$ in a suitable norm as small as possible. In the case of the Euclidean norm, x is defined as the solution of the problem

$$\|Ax - b\|_2 \to \min_x .$$

Phillips's example

Example (Phillips's example ... continued)

D. L. Phillips, A Technique for the Numerical Solution of Certain Integral Equations of the First Kind, J. ACM 9 (1) 1962, pp. 84-97.

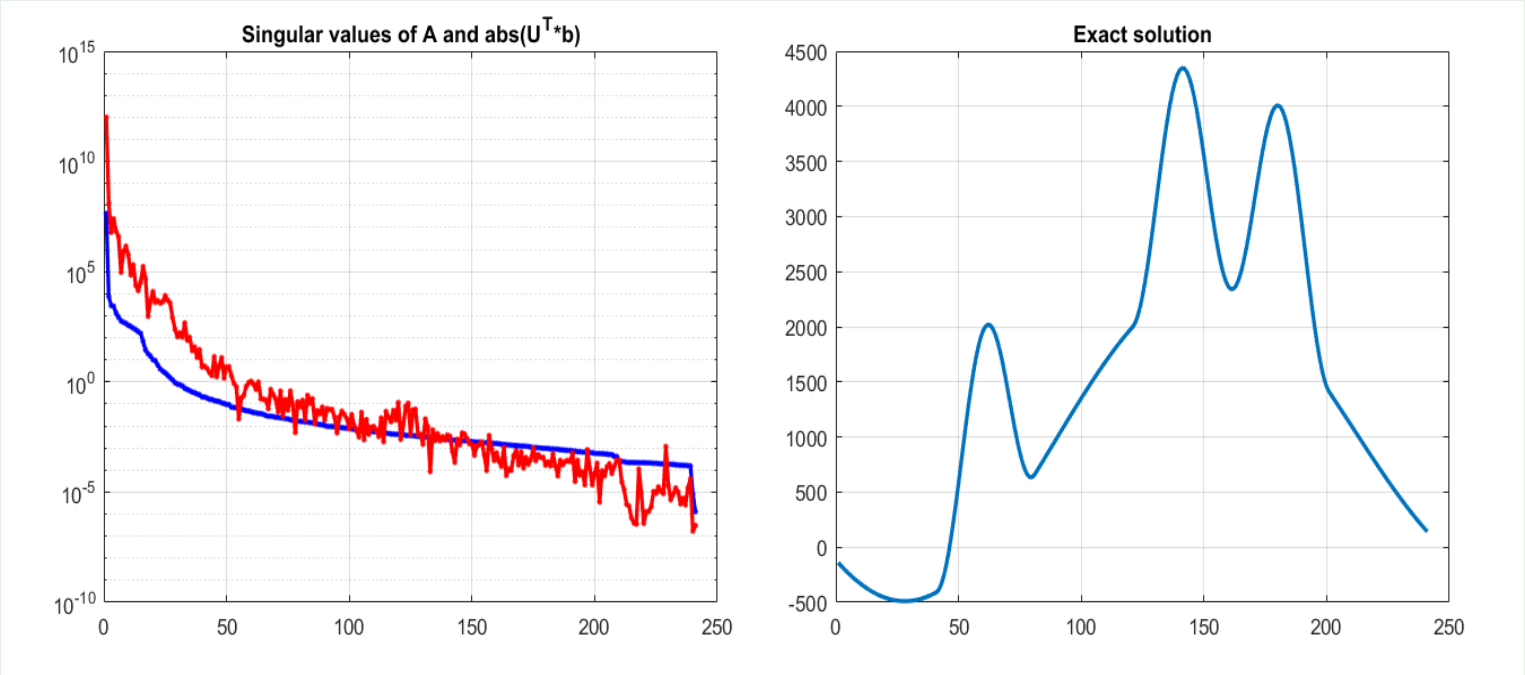


Figure: The data for the Phillips's example. x generated first, then $b = Ax$.

Phillips's example

Example (Phillips's example ... continued)

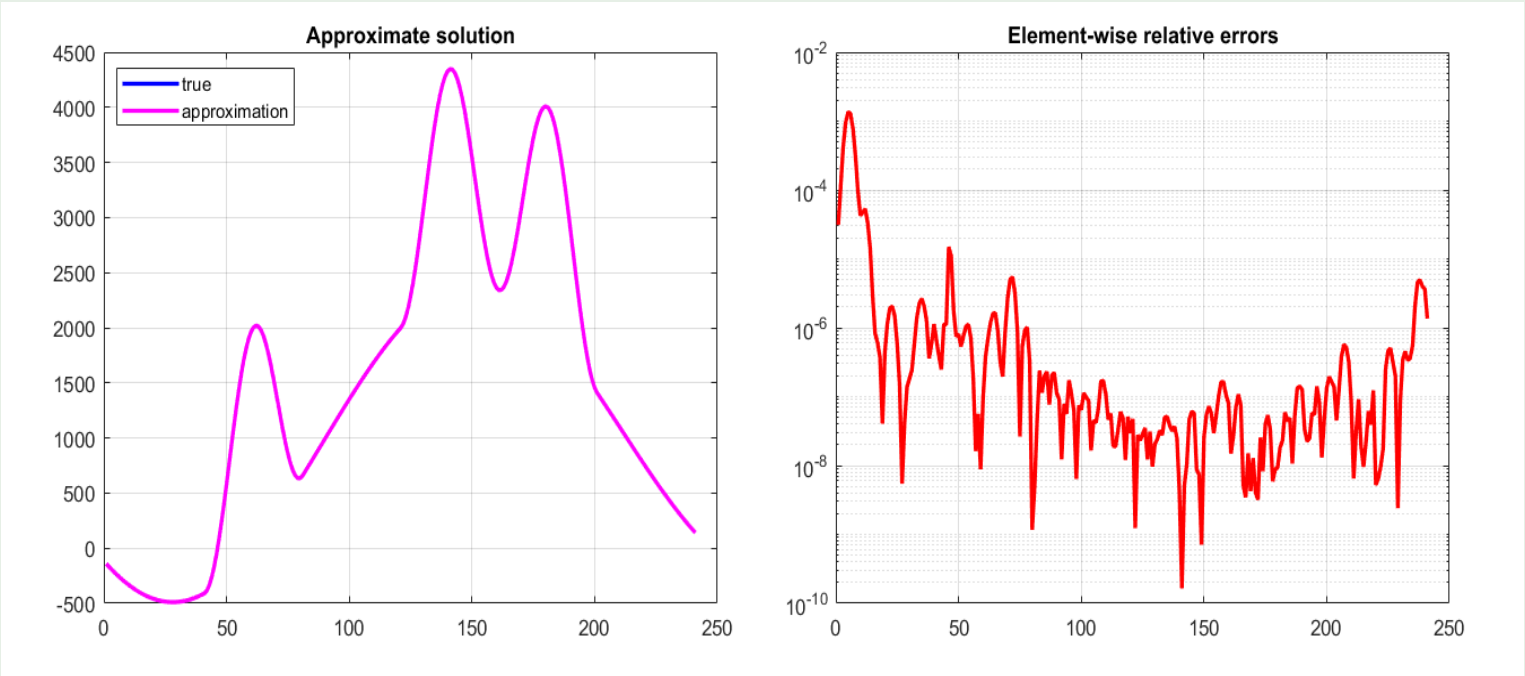


Figure: The SVD solution of the "noise free" least squares problem of a Phillips' example. The condition number of A is $\kappa_2(A) > 10^{13}$.

Now, consider b contaminated by noise and solve as $x = A^{\dagger}b$.

Phillips's example

Example (Phillips's example ... continued)

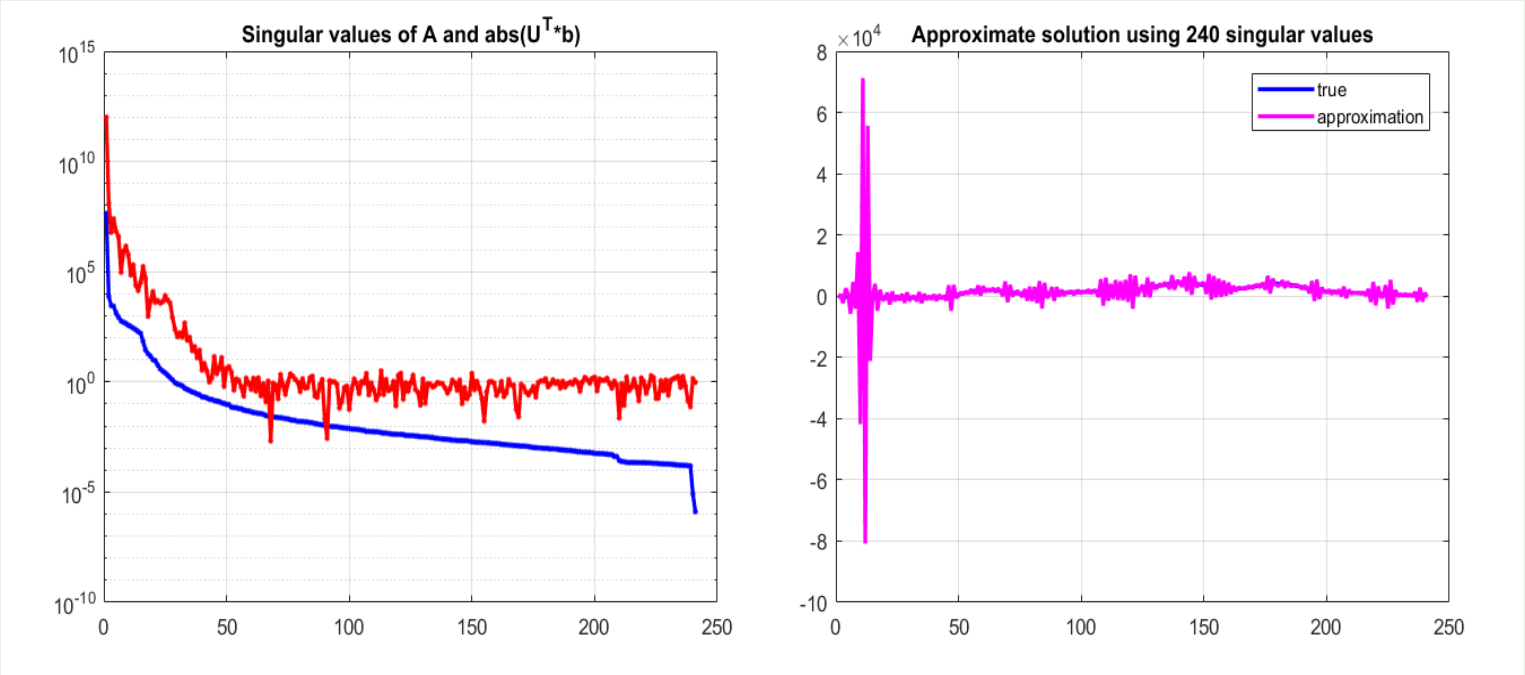


Figure: The SVD solution of a noisy least squares problem of a Phillips' example.

Phillips's example

Example (Phillips's example ... continued)

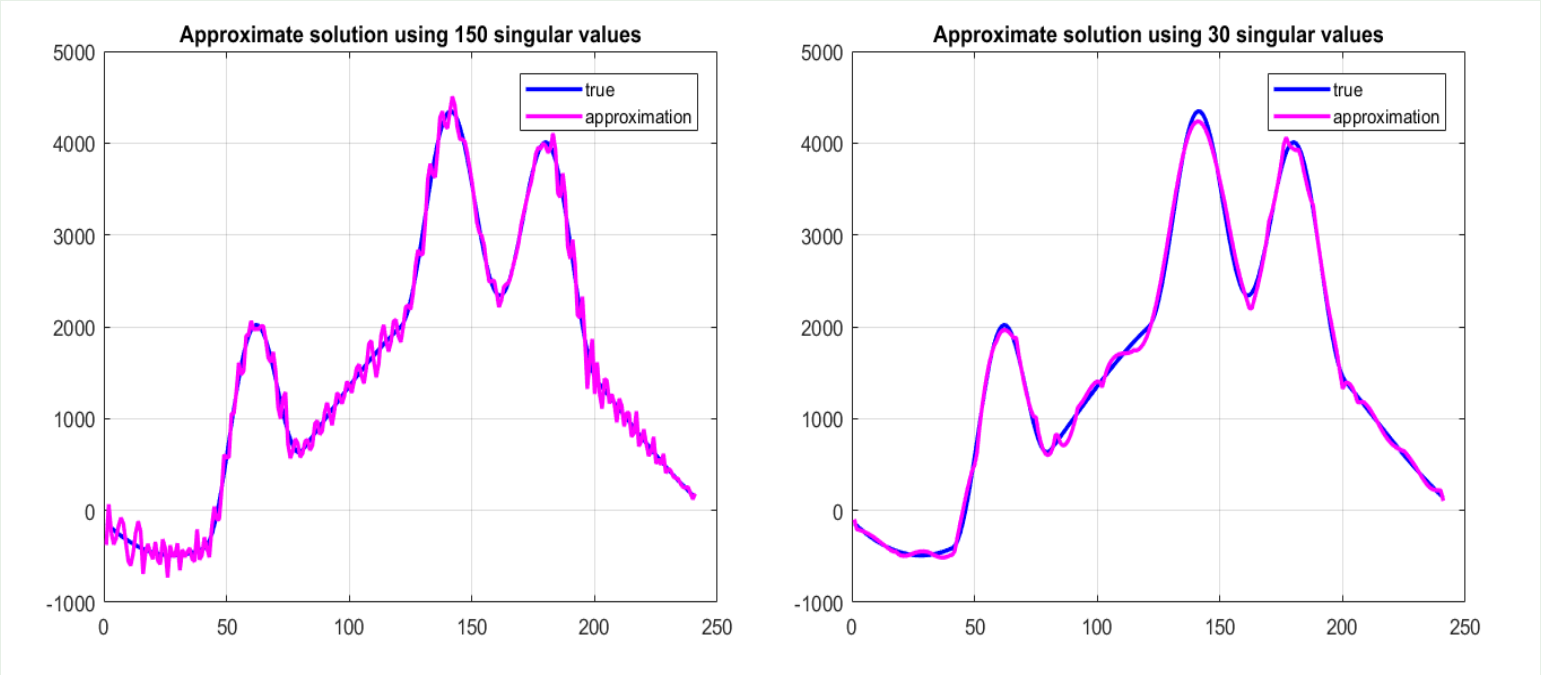


Figure: The SVD solution of a noisy least squares problem of a Phillips' example.

Phillips's example

Example (Phillips's example ... continued)

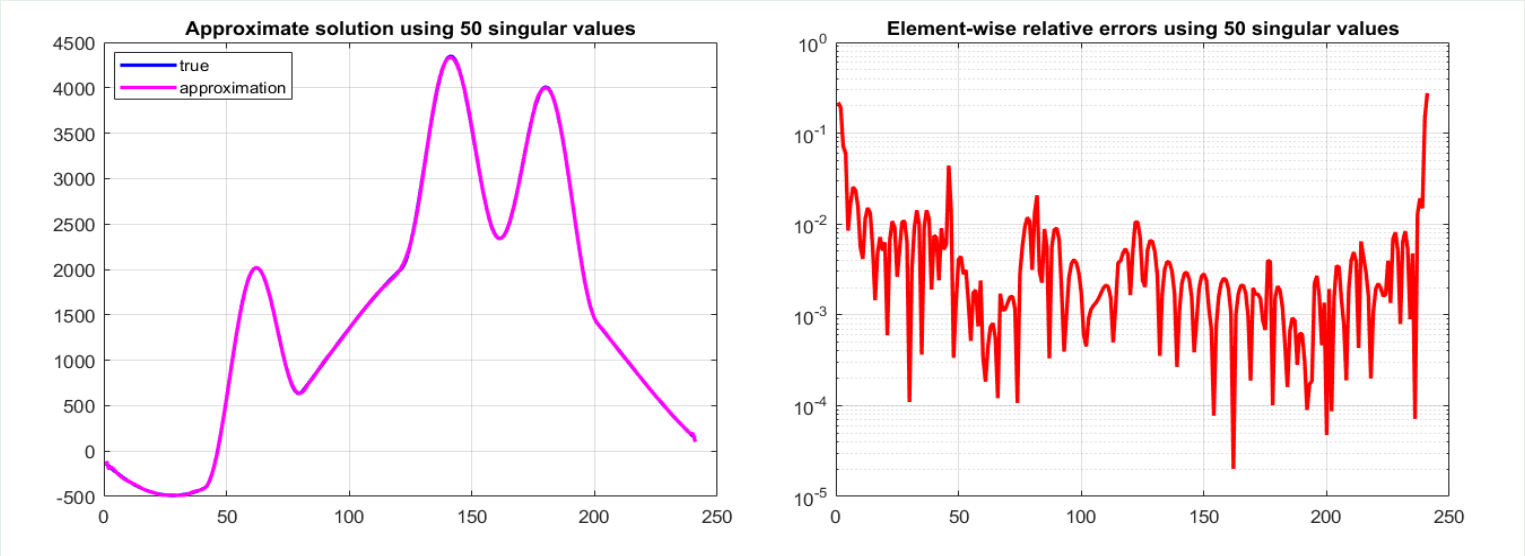


Figure: The SVD solution of a noisy least squares problem of a Phillips' example using 50 dominant singular values, and the element-wise relative errors in the solution vector. Compare with the first plot in Figure 8.

Exercise

Write a Matlab (or Python, R, Octave, ...) code for the Phillips's example. Use D. L. Phillips, A Technique for the Numerical Solution of Certain Integral Equations of the First Kind, J. ACM 9 (1) 1962, pp. 84-97.

Least Squares Solutions: A review

permutation

$$\underbrace{A}_{m\times n}\overbrace{P} = Q\begin{pmatrix} R \\ 0 \end{pmatrix},\quad R = \begin{pmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \hline 0 & 0 & \blacksquare & \bullet & \blacksquare & \blacklozenge \\ 0 & 0 & 0 & \bullet & \blacksquare & \blacklozenge \\ 0 & 0 & 0 & 0 & \blacksquare & \blacklozenge \\ 0 & 0 & 0 & 0 & 0 & \blacklozenge \end{pmatrix}$$

$$Q^*Q = I_m.$$

$$|R_{ii}| \geq \sqrt{\sum_{k=i}^{j} |R_{kj}|^2}, \quad \text{for all } 1 \leq i \leq j \leq n. \tag{15}$$

$$|R_{11}| \geq |R_{22}| \geq \cdots \geq |R_{\rho\rho}| \gg |R_{\rho+1,\rho+1}| \geq \cdots \geq |R_{nn}| \tag{16}$$

The structure (15), (16) may not be rank revealing but it must be guaranteed by the software (e.g. LAPACK, Matlab). Implemented in LINPACK in 1971., adopted by (Sca)LAPACK and used in many packages.

Least Squares Solutions: A review

Let $\mathrm{rank}(A) = r$,

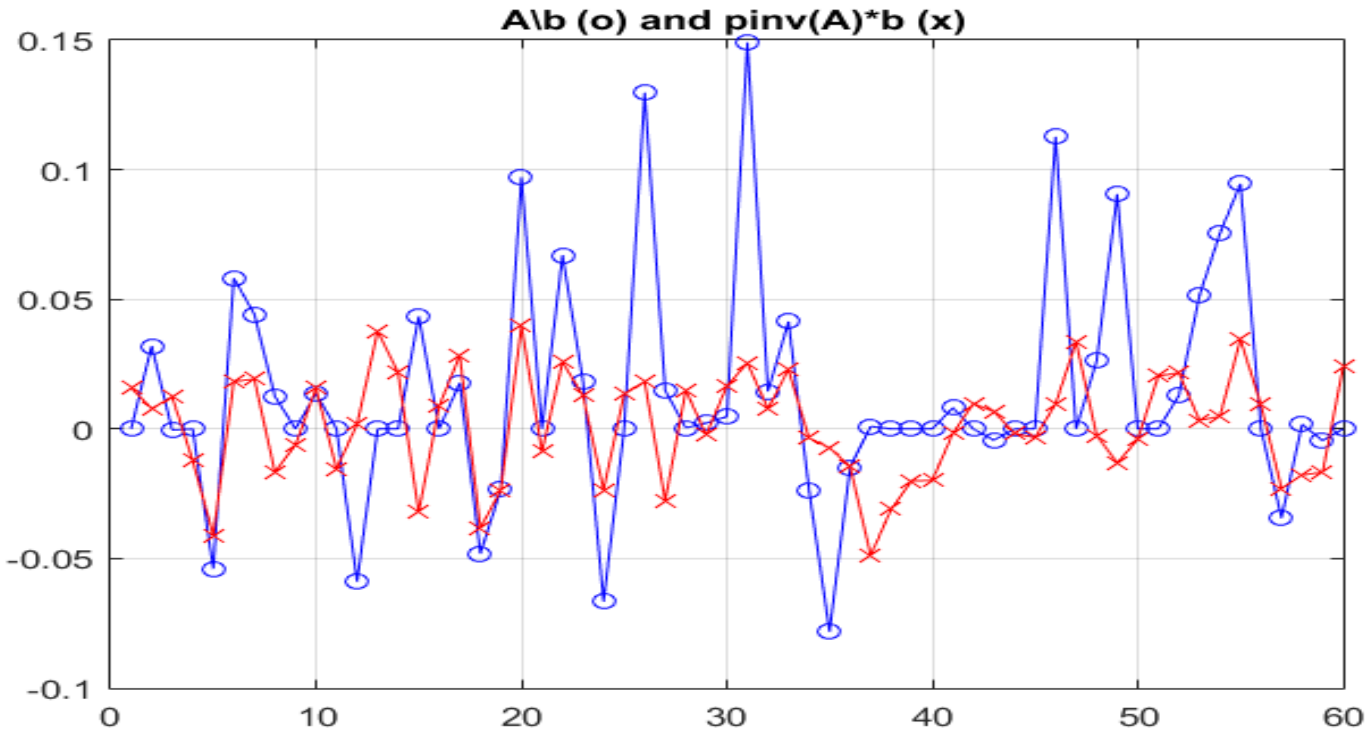
$$AP = QR = \begin{pmatrix} R_{[11]} & R_{[12]} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad Q_r = Q(:, 1:r)$$

$$\begin{aligned} \|Ax - b\|_2 &= \|Q \begin{pmatrix} R_{[11]} & R_{[12]} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} P^T x - b\|_2 \\ &= \| \begin{pmatrix} R_{[11]} & R_{[12]} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \|_2 \to \min, \quad \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} R_{[11]}^{-1} c_1 \\ \mathbf{0} \end{pmatrix} \end{aligned}$$

$x = Py = P \begin{pmatrix} R_{[11]}^{-1} Q_r^* b \\ \mathbf{0} \end{pmatrix}$. x has at least $n - r$ zeros.

If $r = n$ then $x = A^\dagger b$ by virtue of uniqueness. Otherwise, this x is different from $A^\dagger b$.

Least Squares Solutions: Example, 100×60 of rank 40




```
x1=A\b, x2=pinv(A)*b; norm(x1)=0.3539; norm(x2)=0.1599
norm(A*x1-b)=7.623047315933105e+00
norm(A*x2-b)=7.623047315933104e+00
```

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

$$\mathbf{Y}_m = \mathbb{A}\mathbf{X}_m \approx \mathbb{A}U_k\Sigma_k V_k^*, \quad \text{and} \quad \mathbb{A}U_k = \mathbf{Y}_m V_k \Sigma_k^{-1}, \tag{17}$$

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

$$S_k = U_k^* \mathbf{Y}_m V_k \Sigma_k^{-1}, \tag{18}$$

which is suitable for data driven setting because it does not use $\mathbb{A}$ explicitly. Clearly, (17, 18) only require that $\mathbf{Y}_m = \mathbb{A}\mathbf{X}_m$; it is not necessary that $\mathbf{Y}_m$ is shifted $\mathbf{X}_m$ (single trajectory). Each eigenpair (λ, w) of S_k generates a Ritz pair $(\lambda, z) = (\lambda, U_k w)$ for $\mathbb{A}$. If $\mathbb{A}^* = \mathbb{A}$ then (in theory) $S_k^* = S_k$ – important for Hermitian DMD.

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, Λ_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:

$$r_k(i) = \|\mathbb{A} Z_k(:,i) - \lambda_i Z_k(:,i)\|_2 = \|(\mathbf{Y}_m V_k \Sigma_k^{-1}) W_k(:,i) - \lambda_i Z_k(:,i)\|_2. \tag{19}$$

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

Bauer–Fike's theorem

Theorem (Bauer-Fike)

Let A be diagonalizable, $\mathsf{A} = \mathsf{S}\Lambda\mathsf{S}^{-1}$. If ρ is an eigenvalue of $\mathsf{A} + \delta\mathsf{A}$, then for $\|\cdot\| \in \{\|\cdot\|_2, \|\cdot\|_1, \|\cdot\|_\infty\}$

$$\min_{i=1:n} |\lambda_i - \rho| \leq \|\mathsf{S}\|\|\mathsf{S}^{-1}\|\|\delta\mathsf{A}\|.$$

PROOF: Ifρ is an eigenvalue of A a well, then we are done. Otherwise, $\mathsf{A} - \rho I$ an $\Lambda - \rho I$ are nonsingular. Since $\mathsf{S}^{-1}(\mathsf{A} + \delta\mathsf{A} - \rho I)\mathsf{S}$ is singular, $\Lambda + \mathsf{S}^{-1}\delta\mathsf{A}\mathsf{S} - \rho\mathsf{I}$ is singular as well. Then

$$\Lambda + \mathsf{S}^{-1}\delta\mathsf{A}\mathsf{S} - \rho I = (\Lambda - \rho I)(I + (\Lambda - \rho I)^{-1}\mathsf{S}^{-1}\delta\mathsf{A}\mathsf{S})$$

implies $\|(\Lambda - \rho I)^{-1}\mathsf{S}^{-1}\delta\mathsf{A}\mathsf{S}\| \geq 1$ in any matrix norm $\|\cdot\|$. Hence, for $\|\cdot\| \in \{\|\cdot\|_2, \|\cdot\|_1, \|\cdot\|_\infty\}$

$$1 \leq \|\mathsf{S}^{-1}\|\|\mathsf{S}\|\|\delta\mathsf{A}\|\|(\Lambda - \rho I)^{-1}\| = \|\mathsf{S}^{-1}\|\|\mathsf{S}\|\|\delta\mathsf{A}\| \frac{1}{\min_{i=1:n} |\lambda_i - \rho|}.$$

Commutative diagram

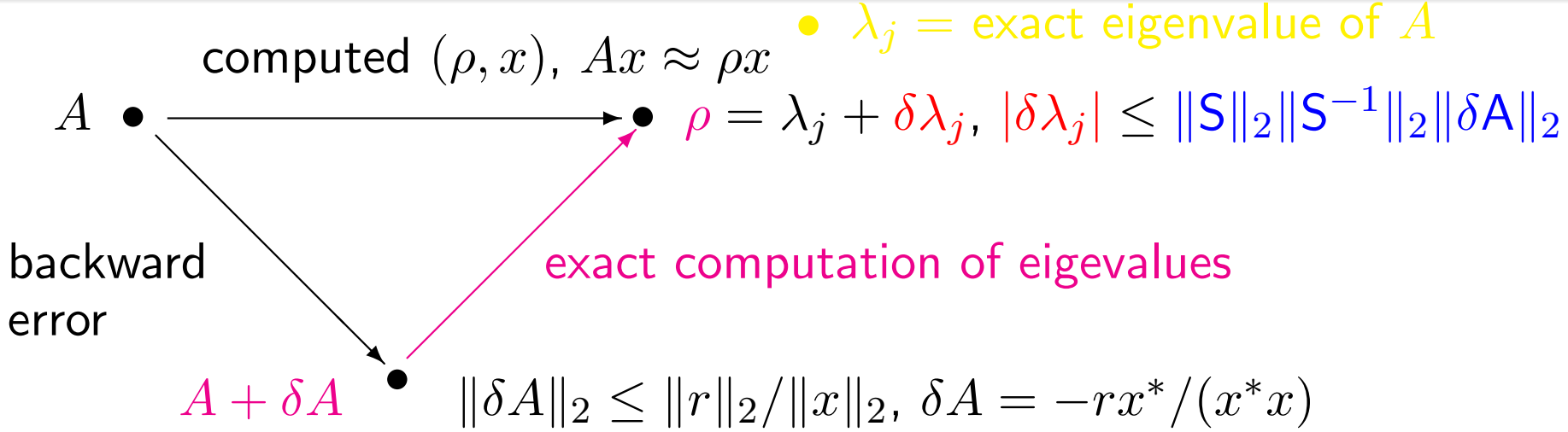


Figure: Commutative diagram for $(A - r/(x^*x))x = \rho x$; $r = Ax - \rho x$.

Error analysis

- Error = distance from an approx. to an unknown object $(\rho - \lambda_j)$.
- Residual = a measure of failure to satisfy defining equation, $r = Ax - \rho x$.
- Backward error = a contrieved change of the input data to justify the computed result. $((A + \delta A)x = \rho x,\ \delta A = -rx^*/(x^*x))$
- Sensitivity analysis (perturbation theory): $A \mapsto A + \delta A$ causes $\lambda \mapsto \lambda + \delta\lambda$; $|\delta\lambda| \leq cond \cdot \|\delta A\|_2$. $cond = \kappa_2(S) = \|S\|_2\|S^{-1}\|_2$.

Data driven (computable) residual

Good Ritz pairs can be selected using a data driven (computable) residuals $\|\mathbb{A}z_i - \lambda_i z_i\|_2$, [Z.D., I. Mezić, R. Mohr, SISC 2018.]

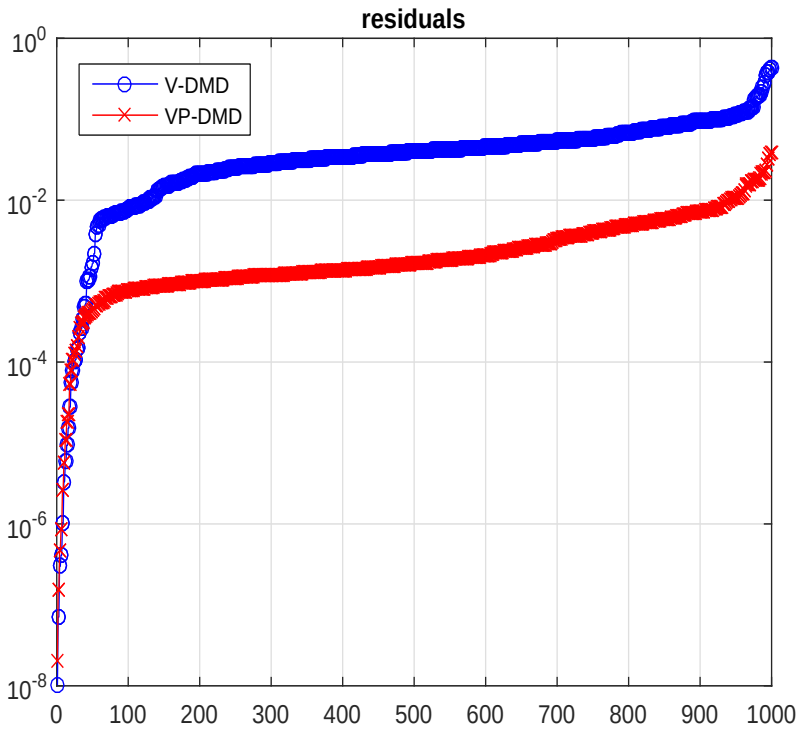
Example

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

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v}\cdot\nabla)\mathbf{v} + \nu\Delta\mathbf{v} - \frac{1}{\rho}\nabla p, \quad \nabla\cdot\mathbf{v} = 0, \tag{20}$$

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



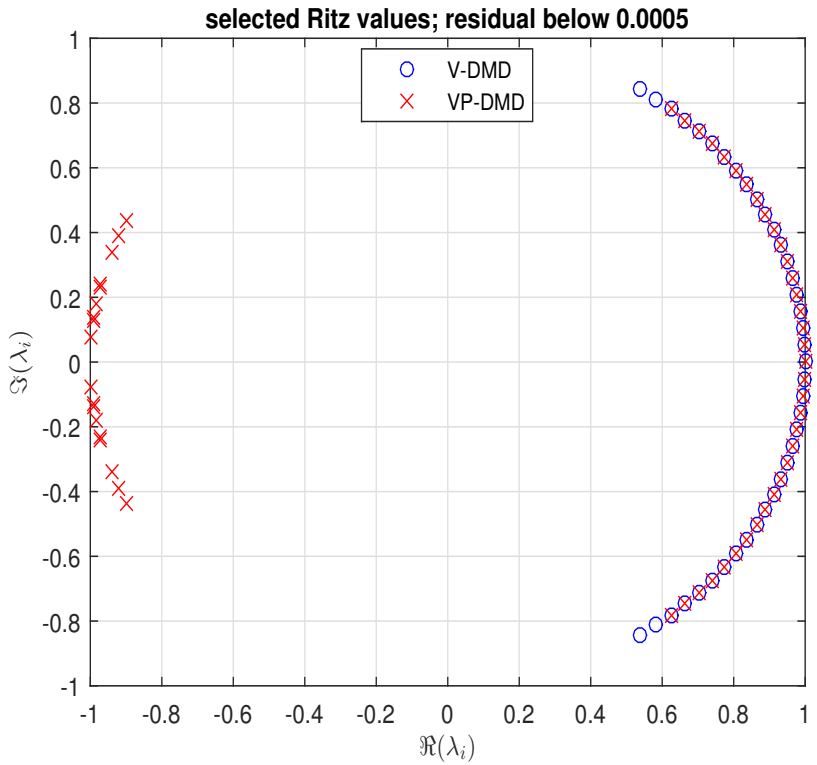


Figure: Left: Comparison of the residuals of the Ritz pairs computed by the DMD_RRR Algorithm with velocities as observables (V-DMD, circles $\circ$) and with both velocities and pressures (VP-DMD, crosses, $\times$). Right: Selected Ritz values with velocities as observables ($\circ$) and with both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

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. Can be applied in data driven setting.

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

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

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: $\quad [\sigma_{\lambda_i}, w_{\lambda_i}] = svd_{\min}(\begin{pmatrix} R(1{:}k, k{+}1{:}2k) - \lambda_i R(1{:}k, 1{:}k) \\ R(k{+}1{:}2k, k{+}1{:}2k) \end{pmatrix})$;

10: $\quad W_k(:,i) = w_{\lambda_i}$; $r_k(i) = \sigma_{\lambda_i}$ {*Optimal residual,* $\sigma_{\lambda_i} = \|R_{\lambda_i} w_{\lambda_i}\|_2$}

11: $\quad \rho_k(i) = w_{\lambda_i}^* S_k w_{\lambda_i}$ {*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

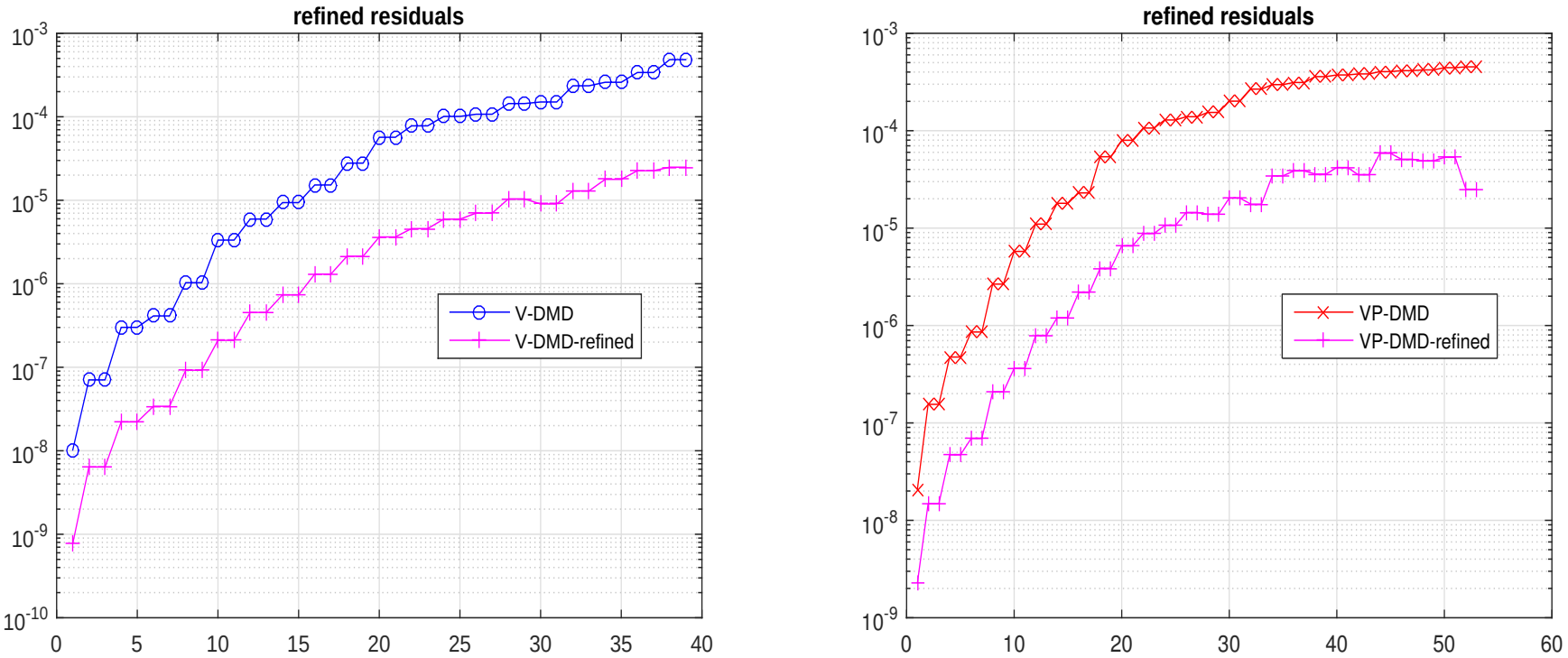


Figure: Comparison of the refined residuals of the Ritz pairs computed by the DMD_RRR Algorithm with velocities as observables (top 39 pairs in V-DMD, circles ○) and with both velocities and pressures (top 53 pairs in VP-DMD, crosses, ×). The noticeable staircase structure on the graphs corresponds to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

Enhanced DMD [Z.D., I. Mezić, R. Mohr, SISC 2018.]

$[Z_k, \Lambda_k, r_k, \rho_k] = \mathrm{DMD\_RRR}(\mathbf{X}_m, \mathbf{Y}_m; \epsilon)$ {*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: $\quad [\sigma_{\lambda_i}, w_{\lambda_i}] = svd_{\min}(\begin{pmatrix} R(1{:}k, k{+}1{:}2k) - \lambda_i R(1{:}k, 1{:}k) \\ R(k{+}1{:}2k, k{+}1{:}2k) \end{pmatrix})$;

10: $\quad W_k(:,i) = w_{\lambda_i}$; $r_k(i) = \sigma_{\lambda_i}$ {*Optimal residual,* $\sigma_{\lambda_i} = \|R_{\lambda_i} w_{\lambda_i}\|_2$}

11: $\quad \rho_k(i) = w_{\lambda_i}^* S_k w_{\lambda_i}$ {*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*}

A synthetic example

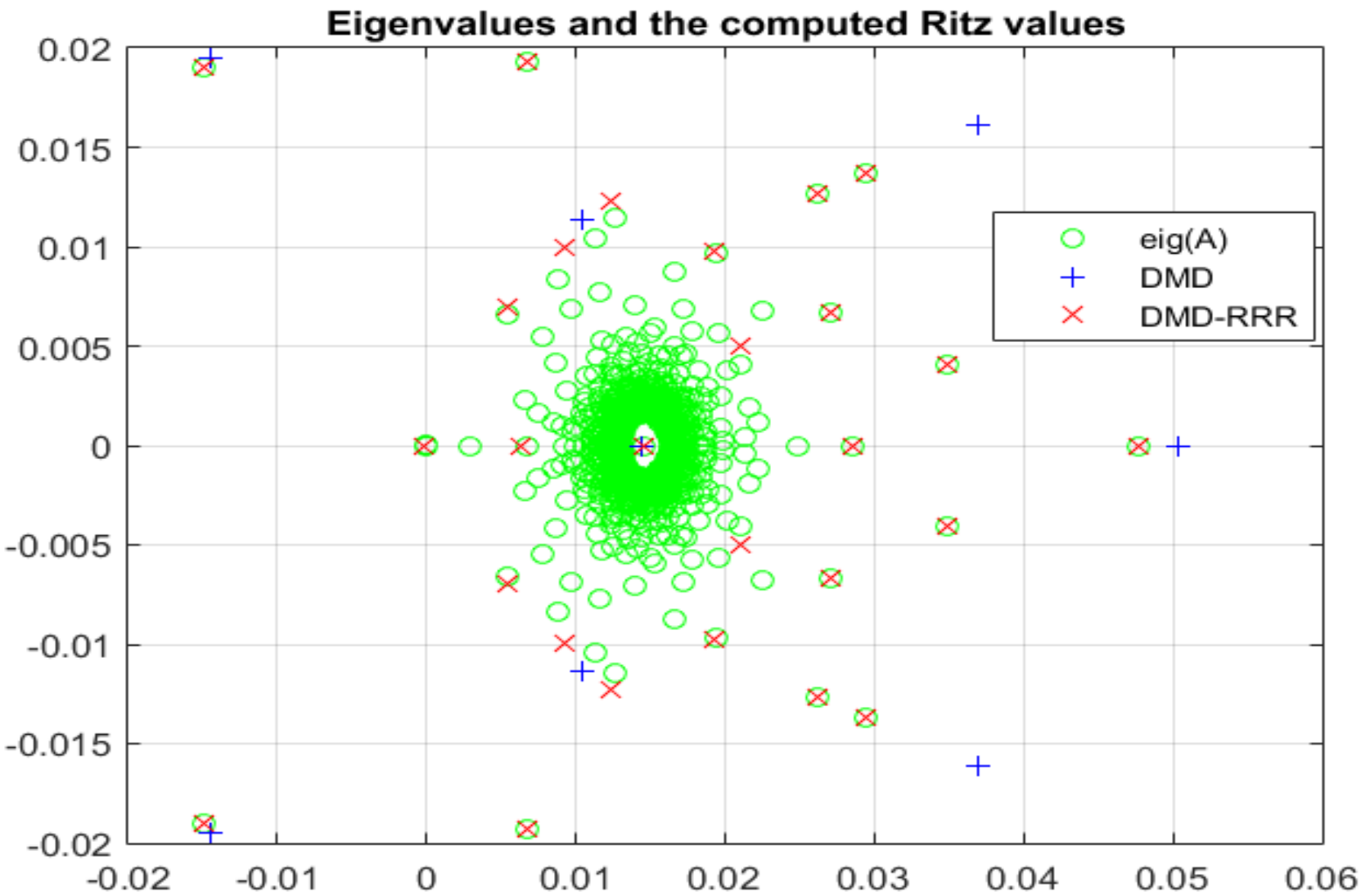
Goal: DMD black-box software

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.

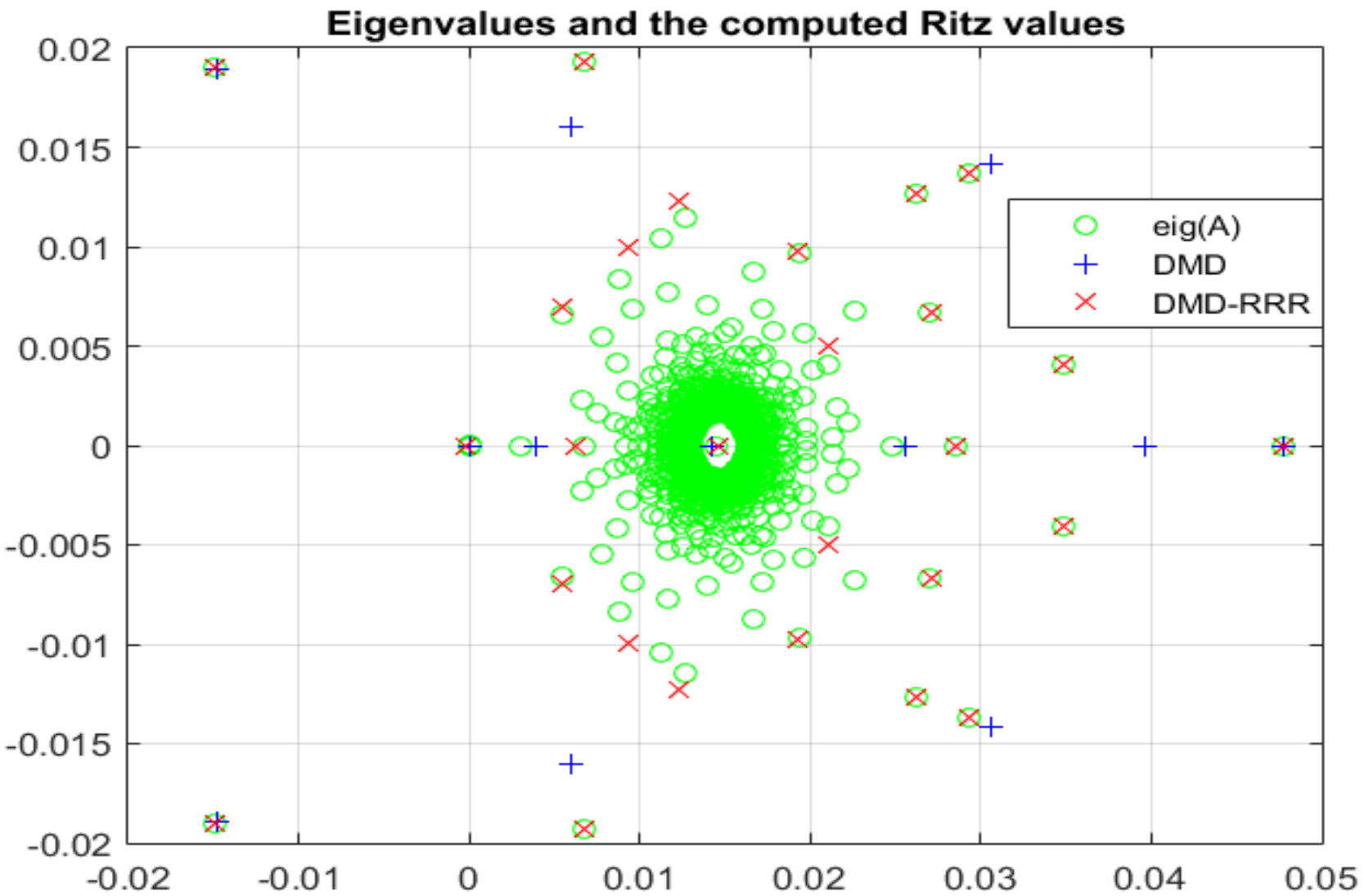
Accuracy of the computed Ritz values



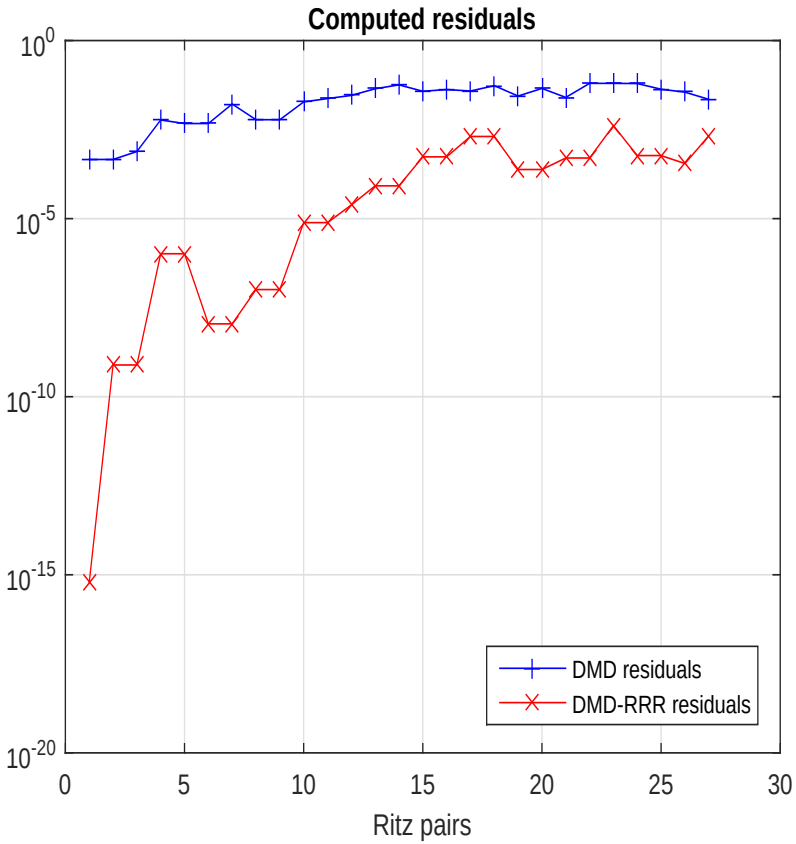
Comparing residuals

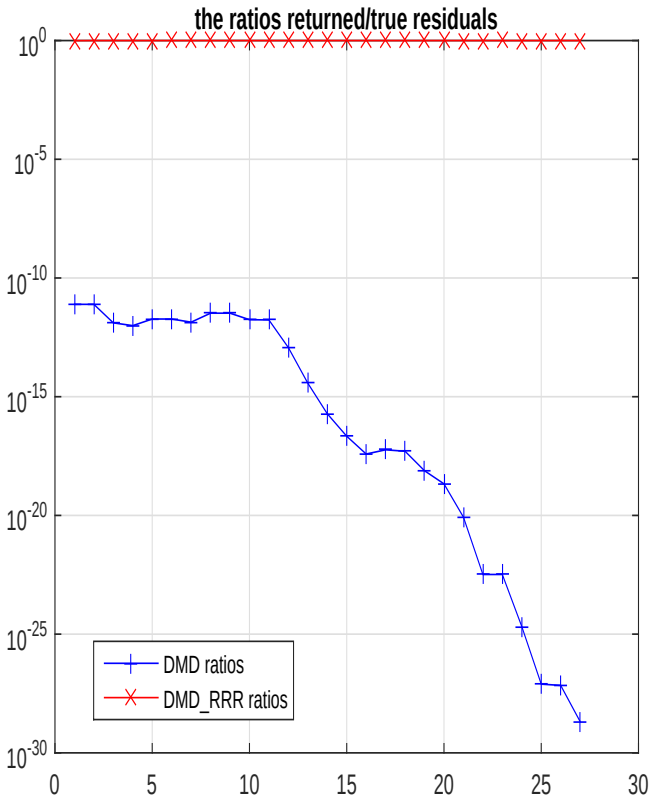
Figure: Comparison of the residuals of the Ritz pairs computed by the DMD Algorithm (pluses $+$) and the DMD_RRR Algorithm (crosses, $\times$), with the same threshold in the truncation criterion for determining the numerical rank.

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

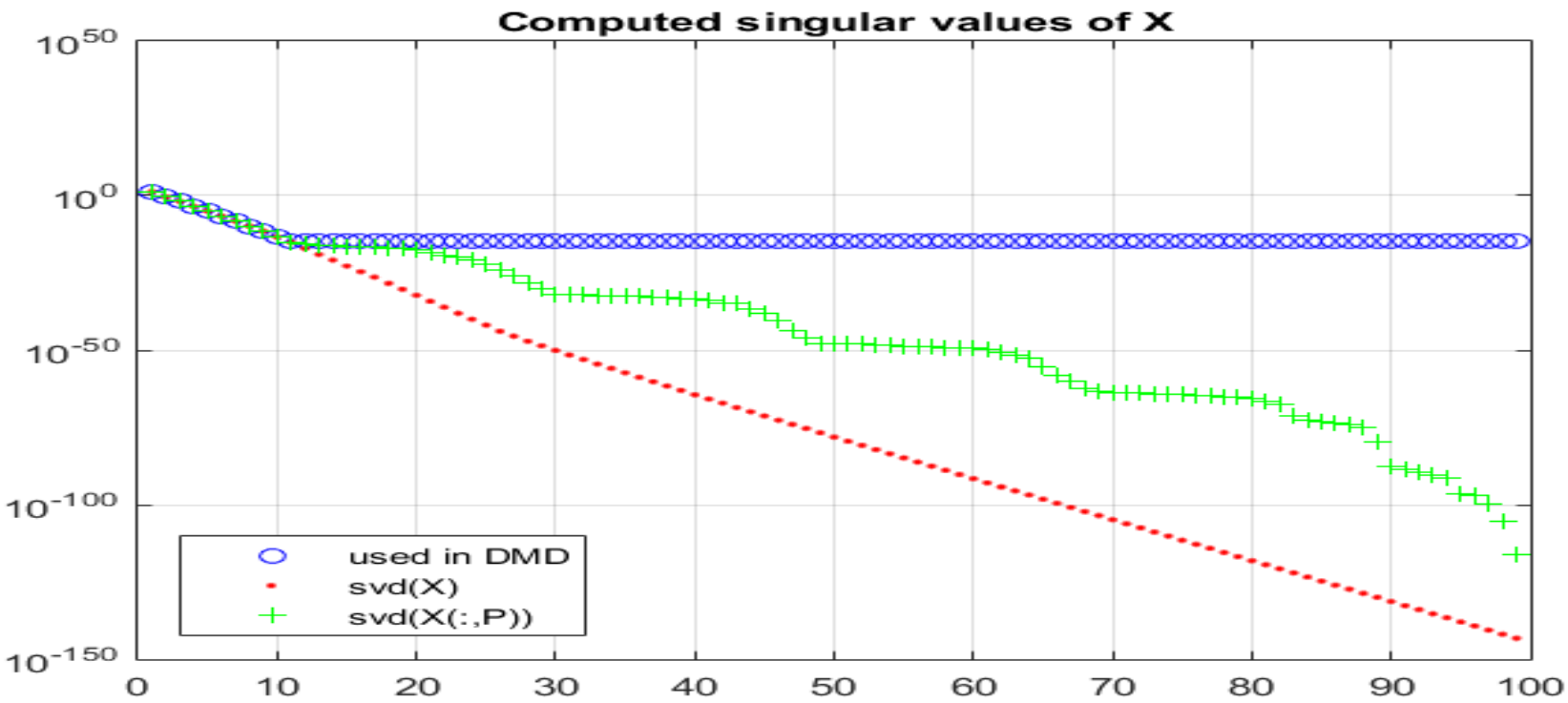


Figure: The blue circles ($\circ$) are the values used in the DMD Algorithm and are computed as $[U, \Sigma, V] = \texttt{svd}(\mathbf{X}_m, 'econ')$. The red dots ($\cdot$) are computed as $\Sigma = \texttt{svd}(\mathbf{X}_m)$, and the pluses ($+$) are the results of $\Sigma = \texttt{svd}(\mathbf{X}_m(:,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

Floating point SVD

If $\mathbf{X}_m \approx \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$ is the computed SVD of $\mathbf{X}_m$, then there exist unitary matrices $\widehat{U}$, $\widehat{V}$, and a perturbation $\delta\mathbf{X}_m$ (*backward error*) such that $\|\widehat{U} - \widetilde{U}\|_2 \leq \epsilon_1$, $\|\widehat{V} - \widetilde{V}\|_2 \leq \epsilon_2$, and

$$\mathbf{X}_m + \delta\mathbf{X}_m = \widehat{U}\widetilde{\Sigma}\widehat{V}^*, \quad \|\delta\mathbf{X}_m\|_2 \leq \epsilon\|\mathbf{X}_m\|_2. \tag{21}$$

Theorem (Weyl and Wieland-Hoffman)

Let the singular values of $\mathbf{X}_m$ and $\mathbf{X}_m + \delta\mathbf{X}_m$ be $\sigma_1 \geq \cdots \geq \sigma_{\min(m,n)}$ and $\widetilde{\sigma}_1 \geq \cdots \geq \widetilde{\sigma}_{\min(m,n)}$, respectively. Then

$$\max_i |\widetilde{\sigma}_i - \sigma_i| \leq \|\delta\mathbf{X}_m\|_2; \quad \sqrt{\sum_{i=1}^{\min(m,n)} |\widetilde{\sigma}_i - \sigma_i|^2} \leq \|\delta\mathbf{X}_m\|_F.$$

Hence, if we combine this Theorem with the backward stability (21), we have that for each computed singular value $\widetilde{\sigma}_i = \sigma_i + \delta\sigma_i$

$$|\delta\sigma_i| \leq \|\delta\mathbf{X}_m\|_2 \leq \epsilon\|\mathbf{X}_m\|_2; \quad |\delta\sigma_i|/\sigma_i \leq \epsilon\|\mathbf{X}_m\|_2/\sigma_i. \tag{22}$$

$$\|\delta \mathbf{f}_i\|_2 \leq \|\delta \mathbf{X}_m\|_2 \leq \epsilon \|\mathbf{X}_m\|_2 \leq \epsilon \sqrt{m}\, \max_i \|\mathbf{f}_i\|_2$$

Bad news for small σ_i's: $\max_i |\delta\sigma_i|/\sigma_i \leq \epsilon \kappa_2(\mathbf{X}_m)$.

Suppose we have backward error $\delta \mathbf{X}_m$ such that

$$\|\delta \mathbf{X}_m(:,i)\|_2 \leq \epsilon \|\mathbf{X}_m(:,i)\|_2, \quad i = 1, \ldots, m. \tag{23}$$

In terms of the snapshots, this reads $\|\delta \mathbf{f}_i\|_2 \leq \epsilon \|\mathbf{f}_i\|_2$, for all snapshots.

Theorem (Eisenstat and Ipsen)

Let $\sigma_1 \geq \cdots \geq \sigma_n$ and $\tilde{\sigma}_1 \geq \cdots \geq \tilde{\sigma}_n$ $A + \delta A = \Xi_1 A \Xi_2$ and let $\xi = \max\{\|\Xi_1 \Xi_1^T - I\|_2, \|\Xi_2^T \Xi_2 - I\|_2\}$. Then

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

Hence, if $\mathbf{X}_m$ is of full column rank, $\mathbf{X}_m + \delta \mathbf{X}_m = (\mathbb{I}_n + \delta \mathbf{X}_m \mathbf{X}_m^\dagger)\mathbf{X}_m$ and n application of this theorem yields

$$\max_i \frac{|\sigma_i - \widetilde{\sigma}_i|}{\sigma_i} \leq 2\|\delta \mathbf{X}_m \mathbf{X}_m^\dagger\|_2 + \|\delta \mathbf{X}_m \mathbf{X}_m^\dagger\|_2^2.$$

$\|\delta \mathbf{X}_m \mathbf{X}_m^\dagger\|_2$ invariant under column scalings!

Discussion 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 $[U, S, V] = \texttt{svd}(\mathbf{X}_m, 'econ')$. 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 $S = \texttt{svd}(X)$ calls the QR SVD, `xGESVD()` in LAPACK.

Note that the same fast `xGESDD()` subroutine is 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, Drmač 2009.) and preconditioned QR (`xGESVDQ()`, LAPACK, Drmač 2018.).

Example: Flow around a cylinder

- J. N. Kutz and S. L. Brunton and B. W. Brunton and J. L. Proctor: Dynamic Mode Decomposition, Society for Industrial and Applied Mathematics 2016.
  Available online at
  `https://epubs.siam.org/doi/abs/10.1137/1.9781611974508`.

- The Matlab codes and the data used in the examples in the book are available at
  `http://dmdbook.com/`

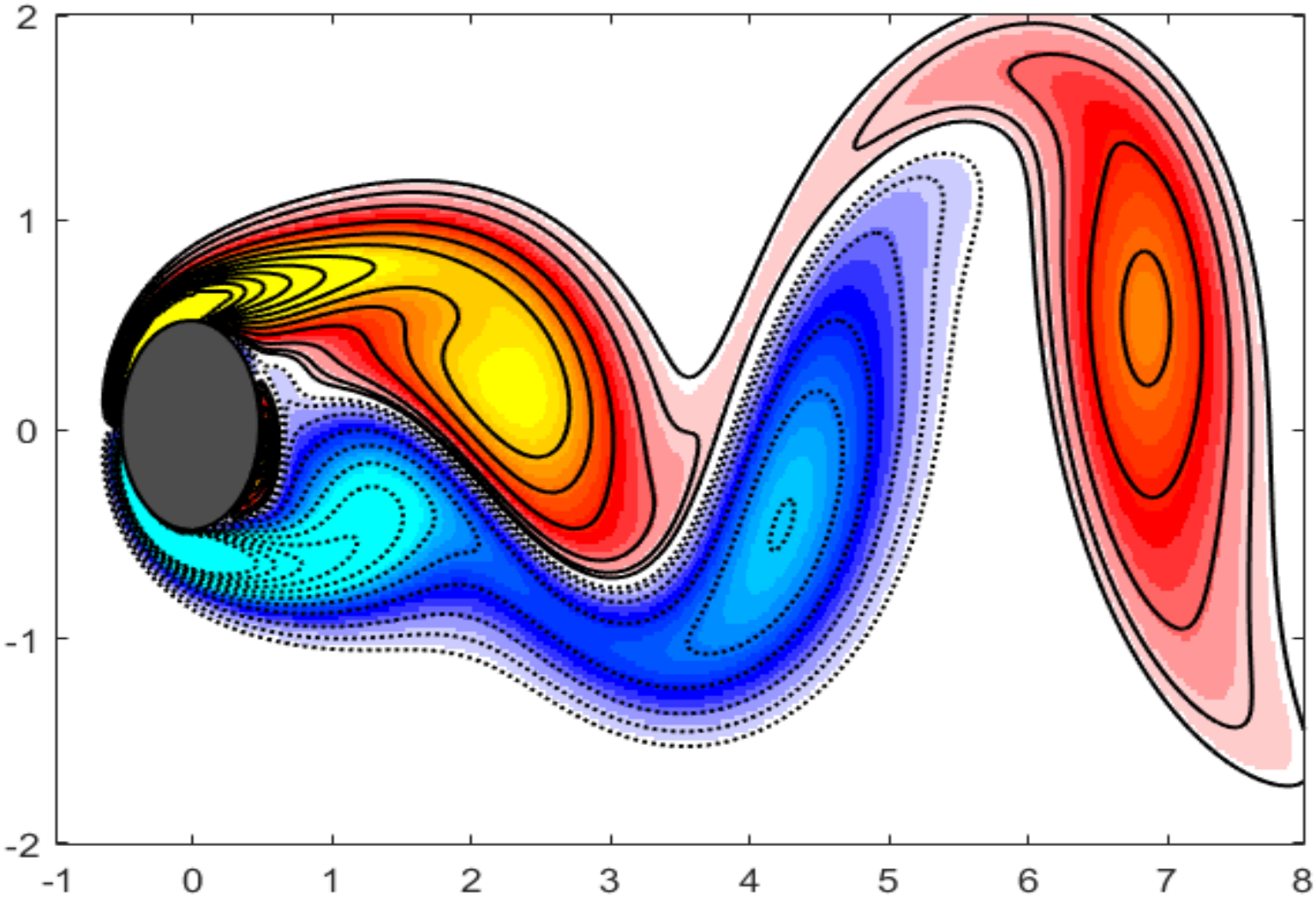
Data snapshots are vorticity data of a flow around cylinder, discretized with dimension 89351. The simulation data with $\delta t = 0.02$ are down–sampled, and the test case contains 151 snapshot. The matrices $\mathbf{X}_m$ and $\mathbf{Y}_m$ are 89351×150, i.e. $m = 50$. For more details on the data set see Chapter 2 of the DMD book.

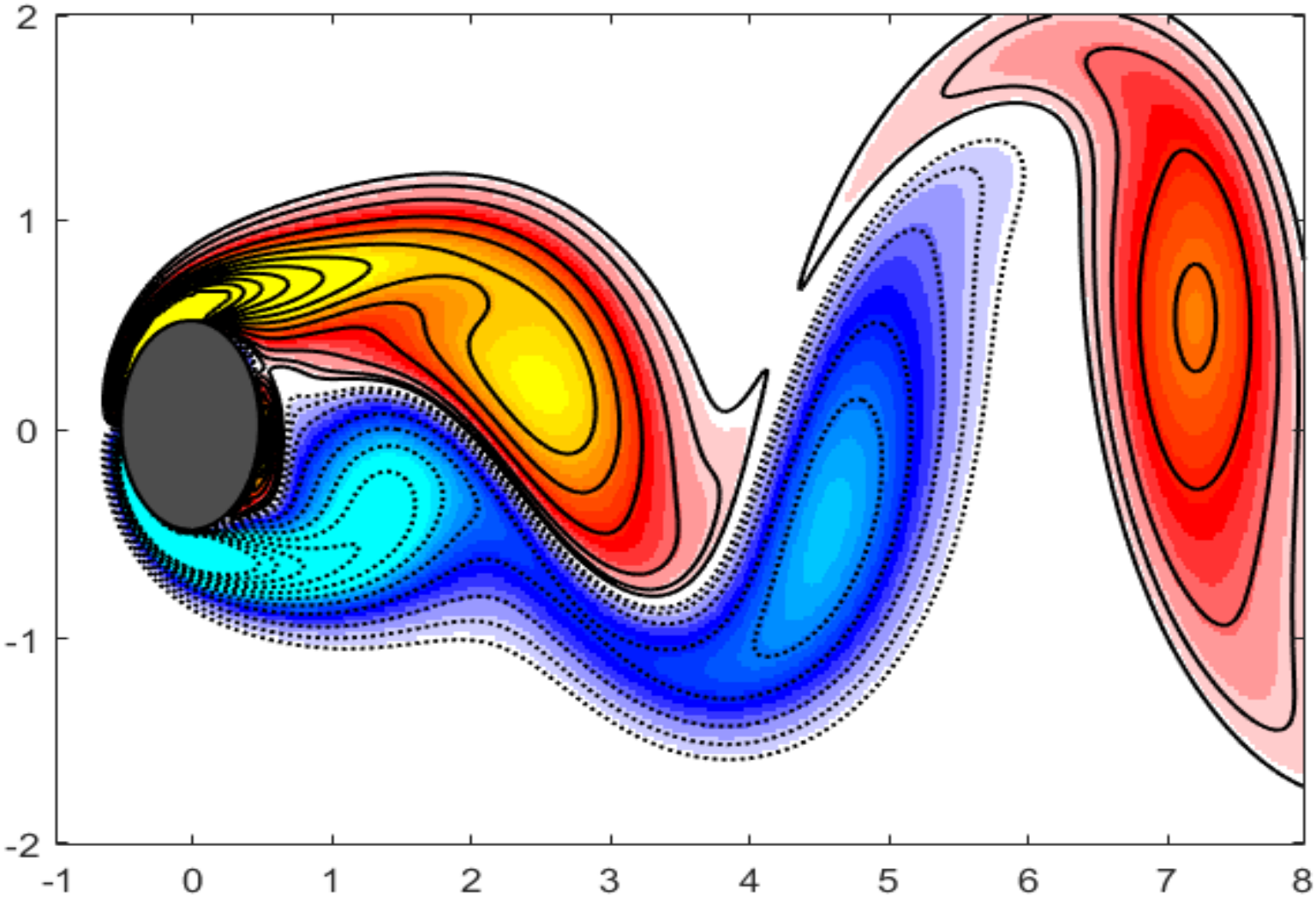
Example: Flow around a cylinder. Goals:

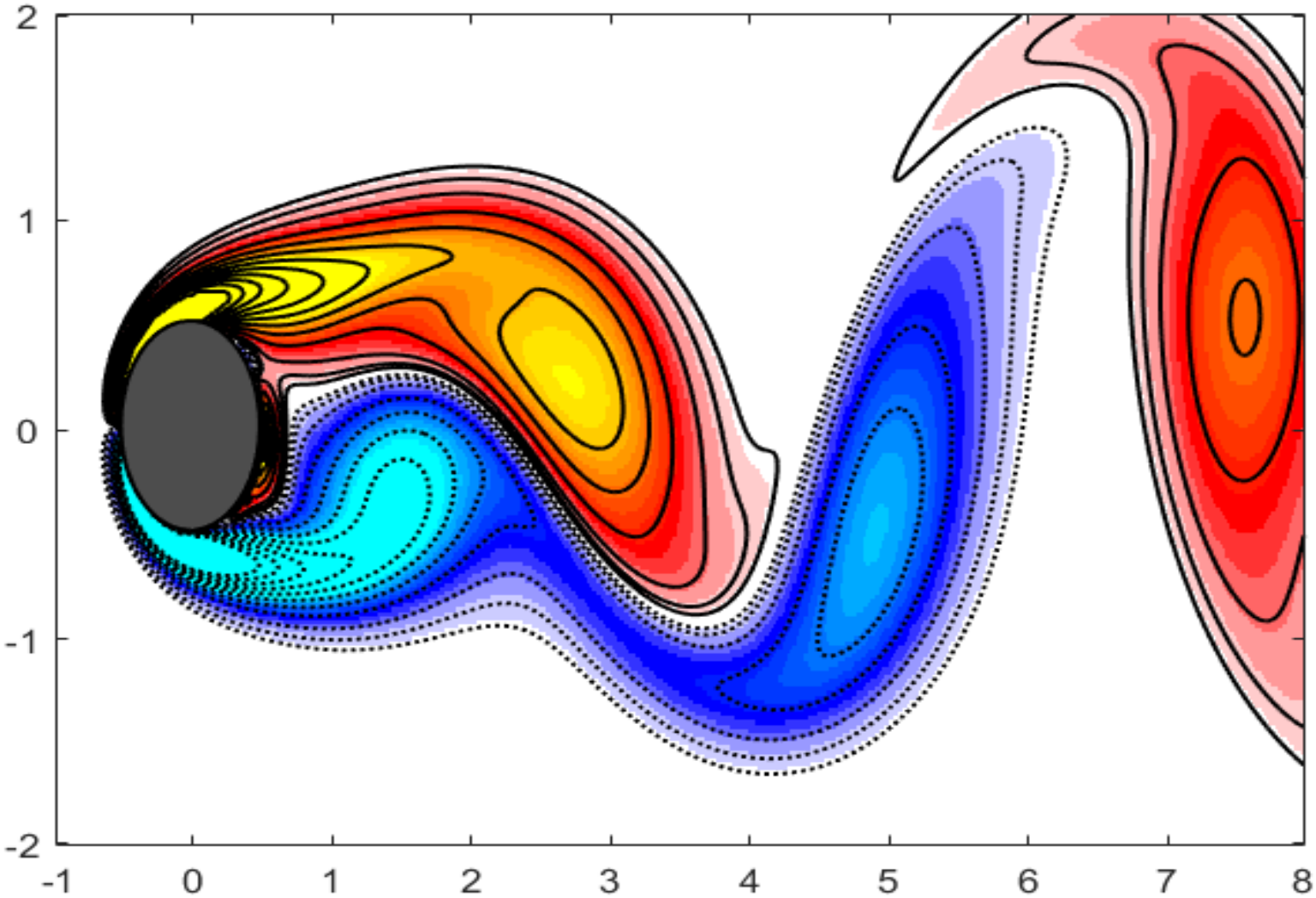
The goals of this example are:

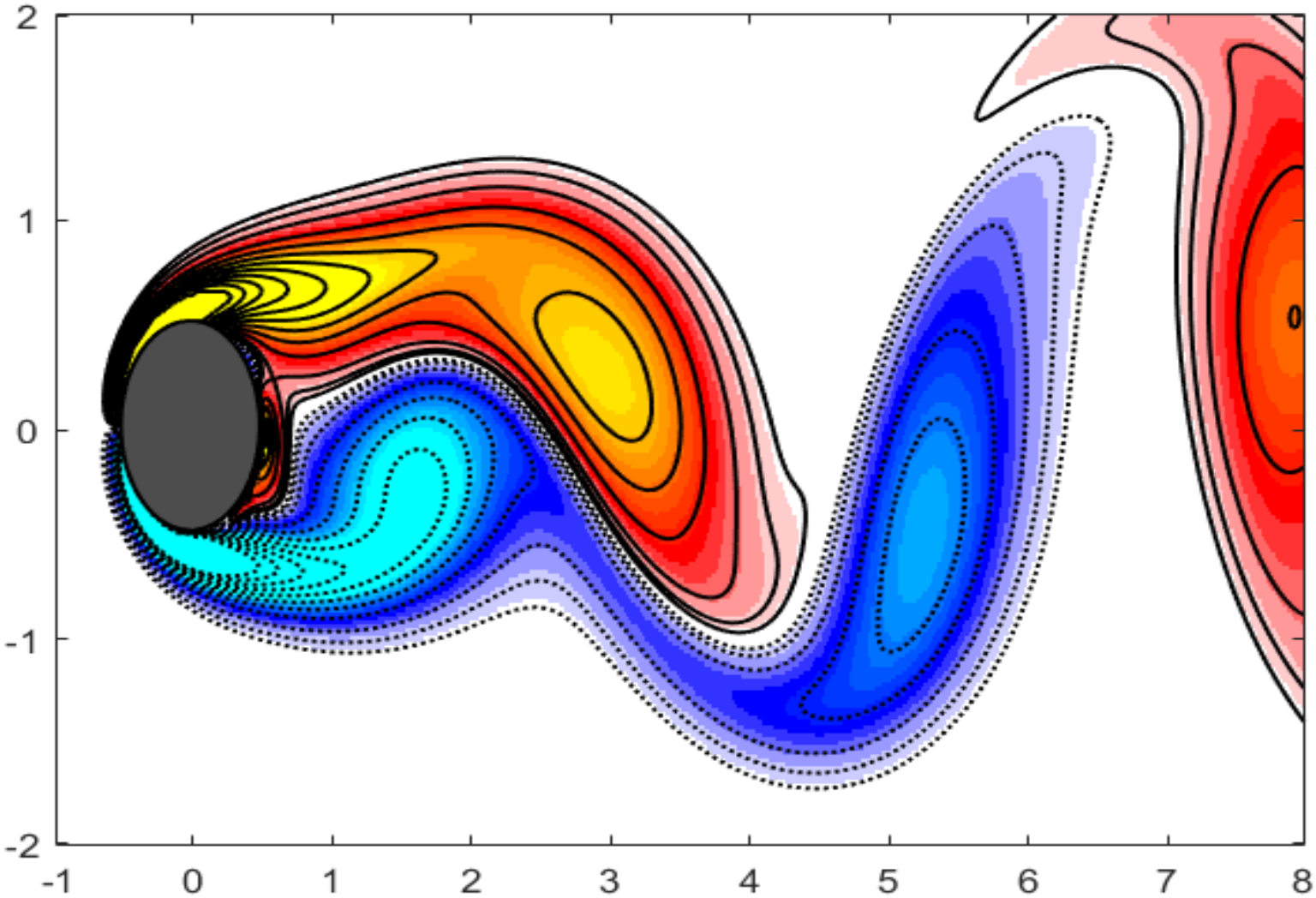
- First contact with the DMD. Intuitive feeling – interpretation of the data snapshots and the modes (eigenvectors) by visualisation.
- Understand and see by example that not all computed eigenpairs (the modes and the eigenvalues) deserve to be accepted.
- Understand and see by example how the residuals make a difference.
- Intuitive feeling – interpretation of the phrase "to reveal latent structure". (We will show later how this gives prediction skills.)

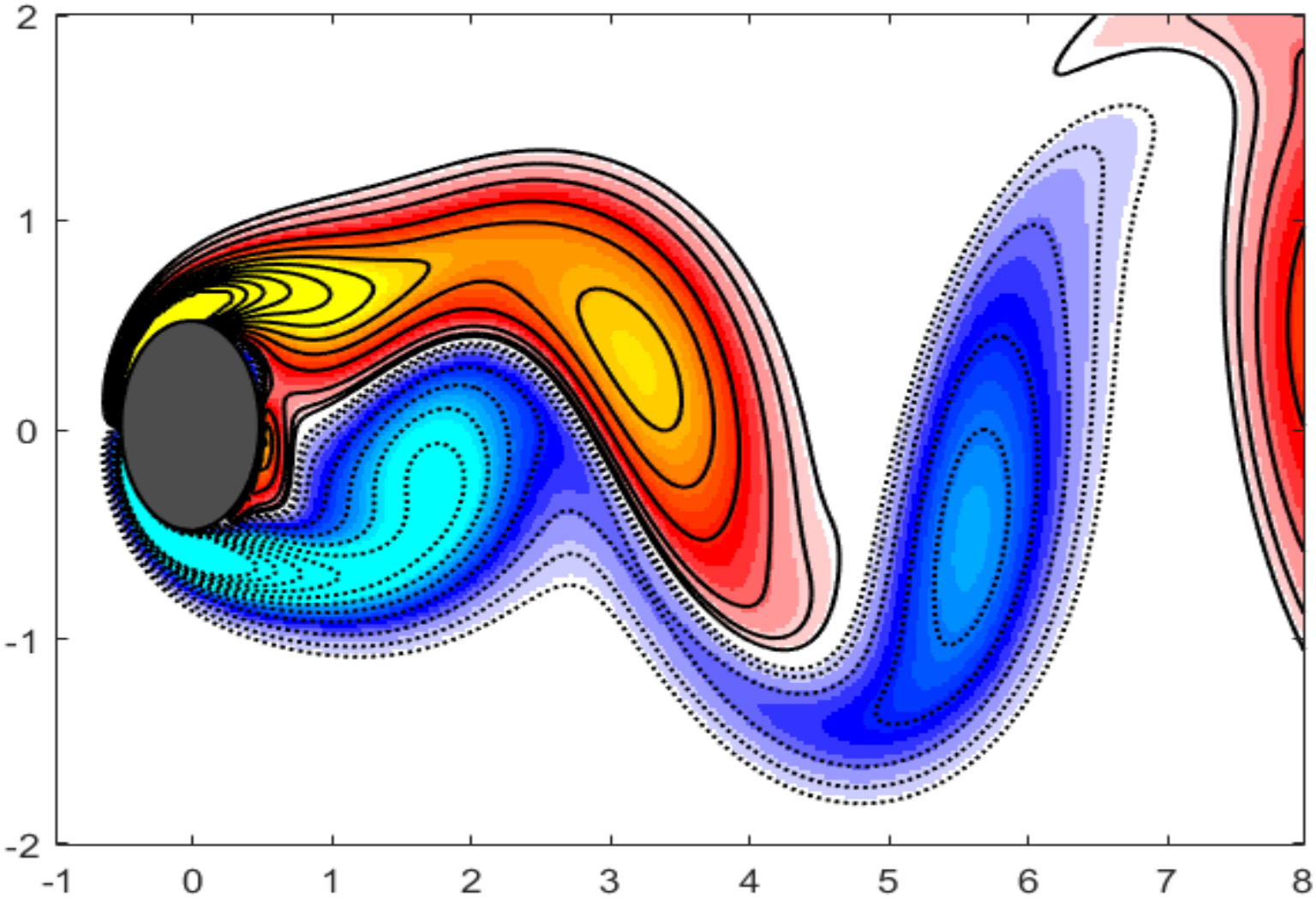
The DMD algorithm is oblivious to the nature of the data – the concept is entirely data driven. It is about finding a structure and being able to predict without knowing what it is about. Of course, expertise in a concrete application field is essential to interpret and to apply the results. The CFD examples are interesting because the key ideas can be nicely visualized. We will not go into the CFD interpretation details.

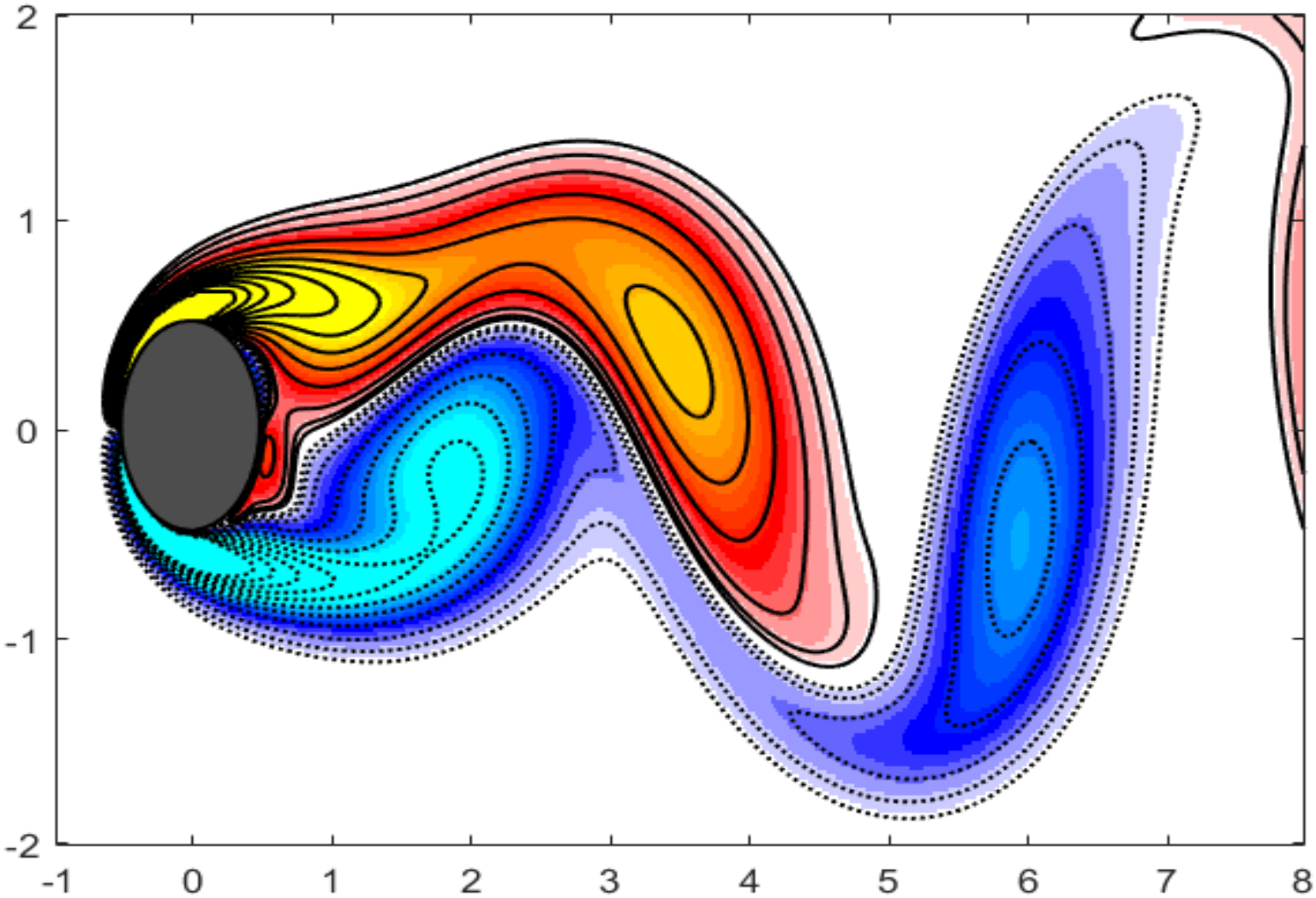

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

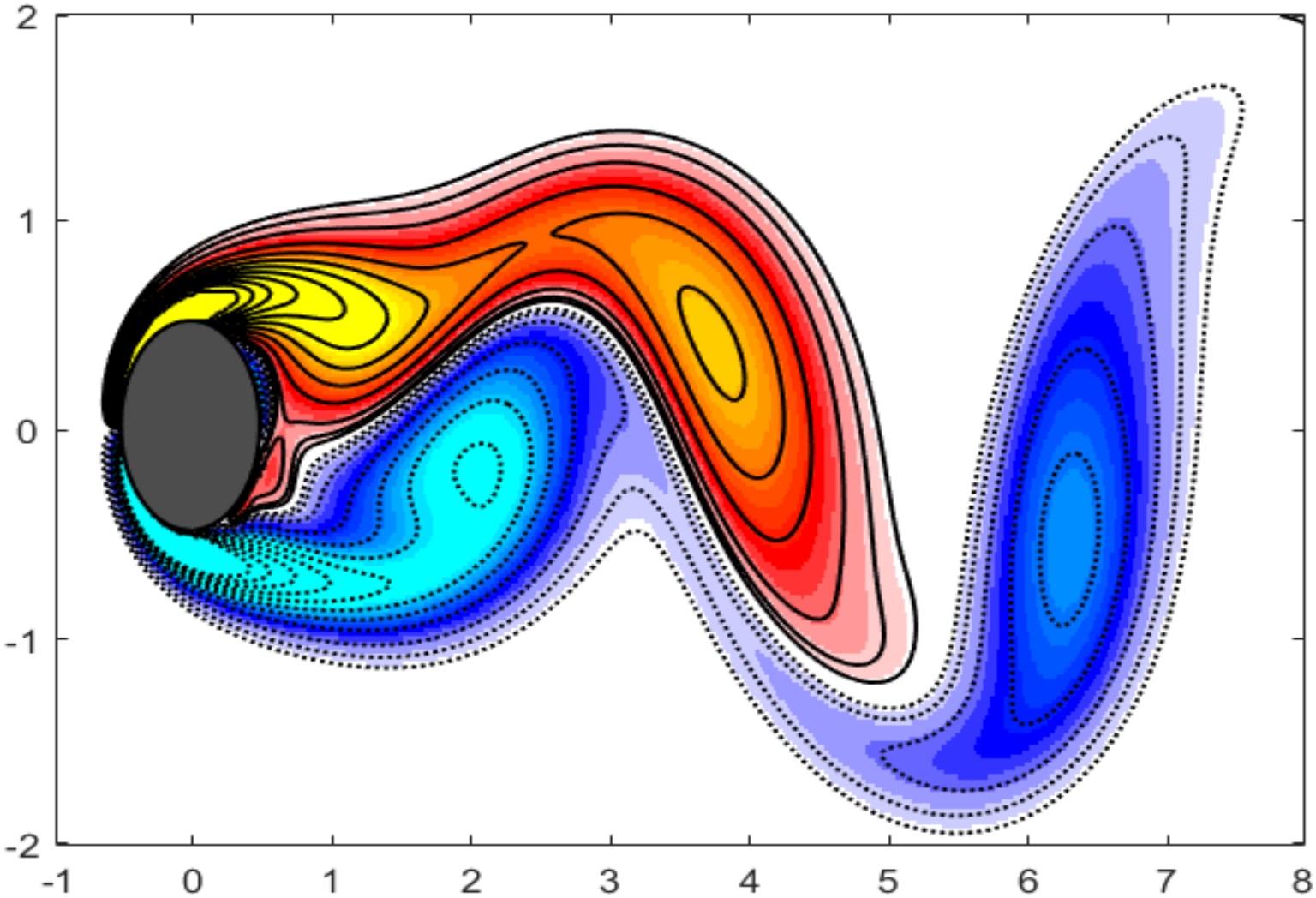

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

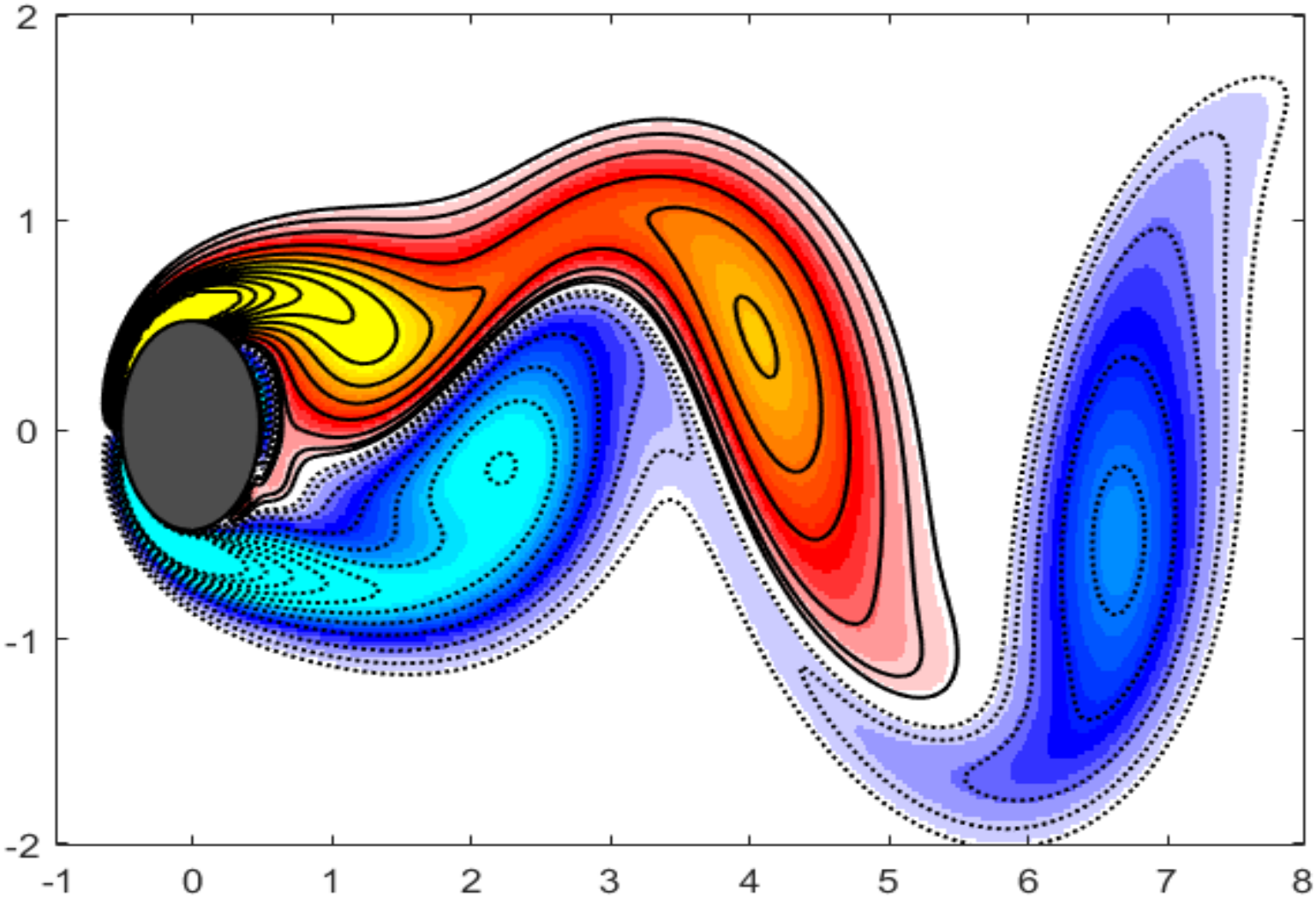

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

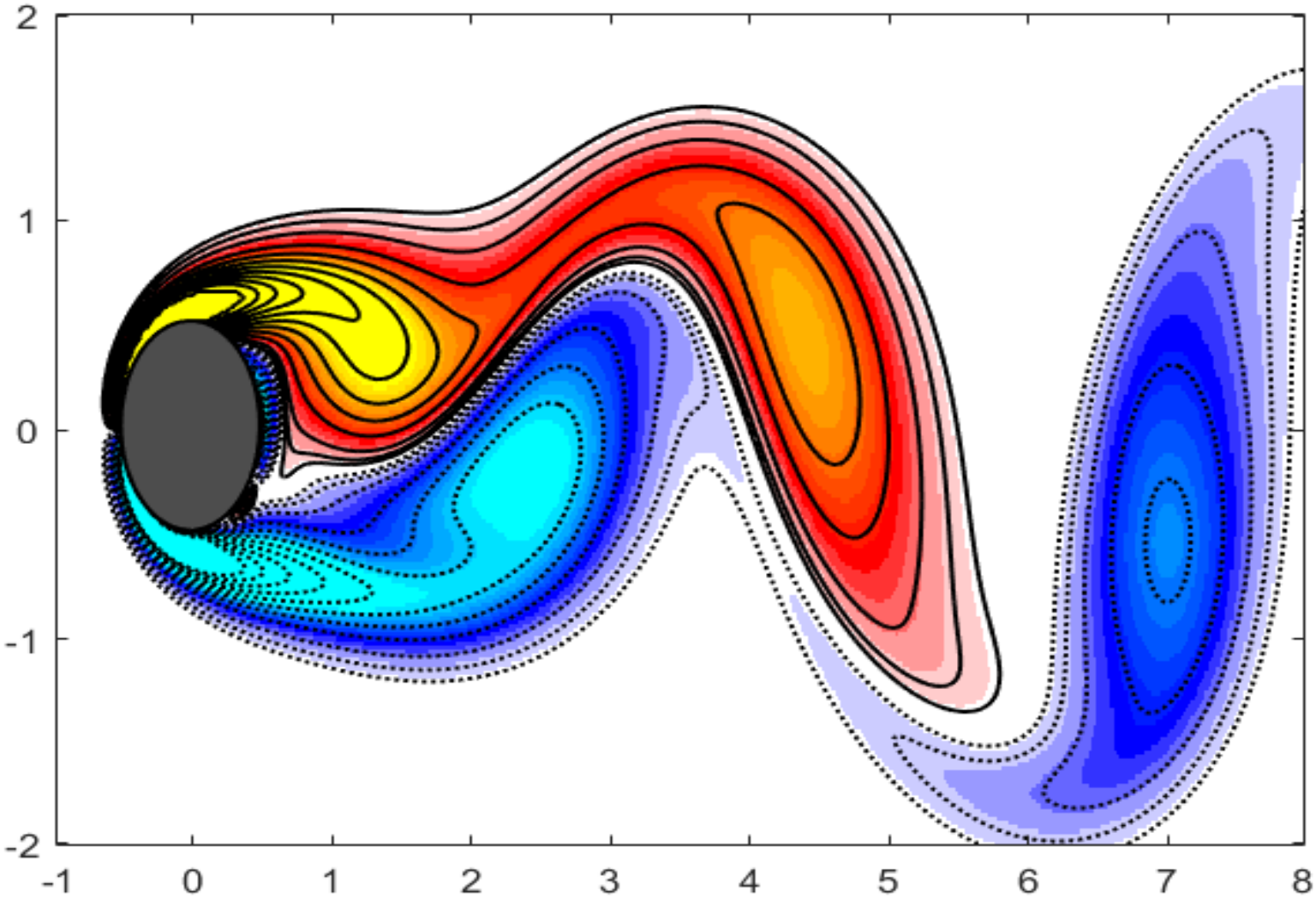

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

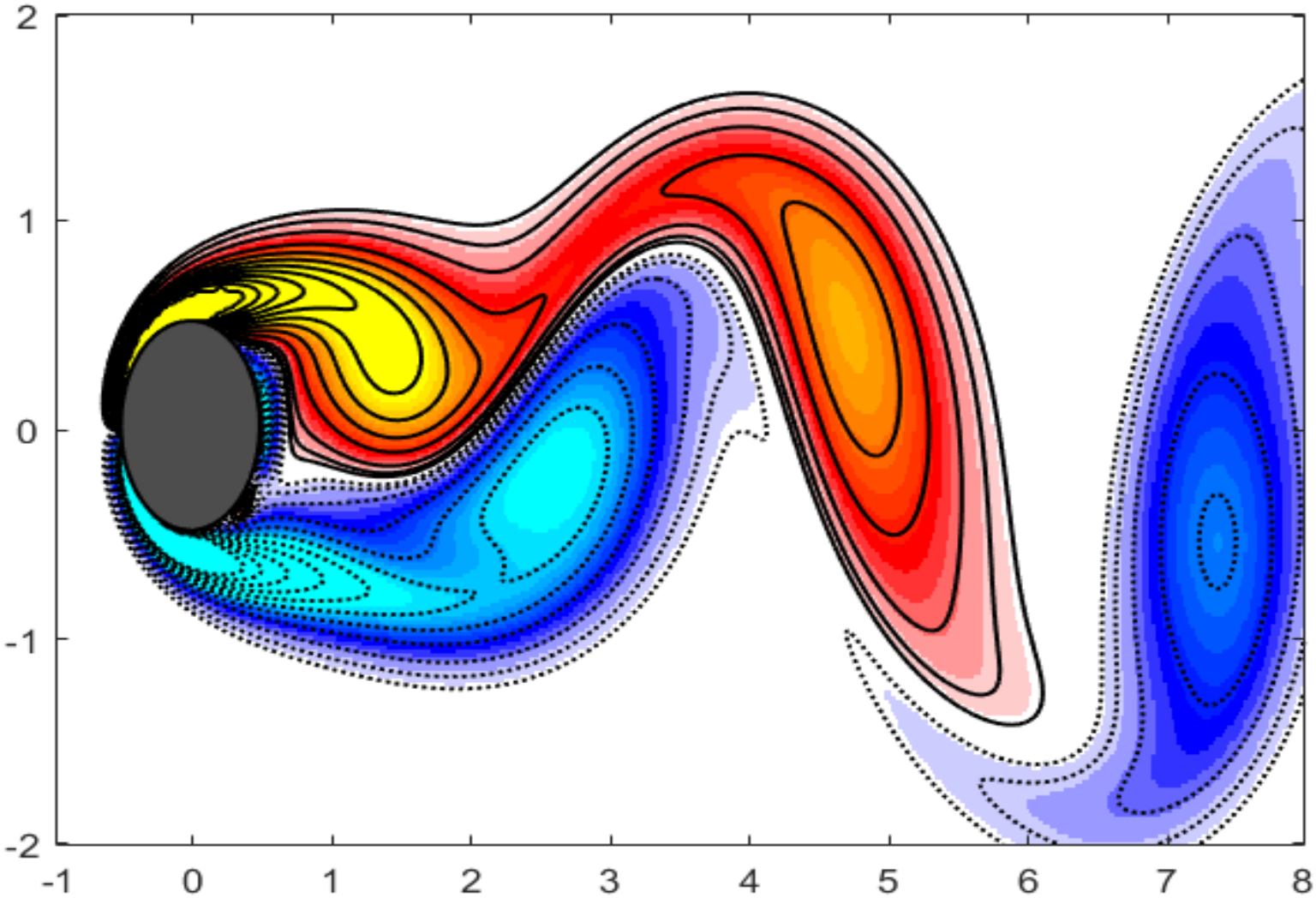

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

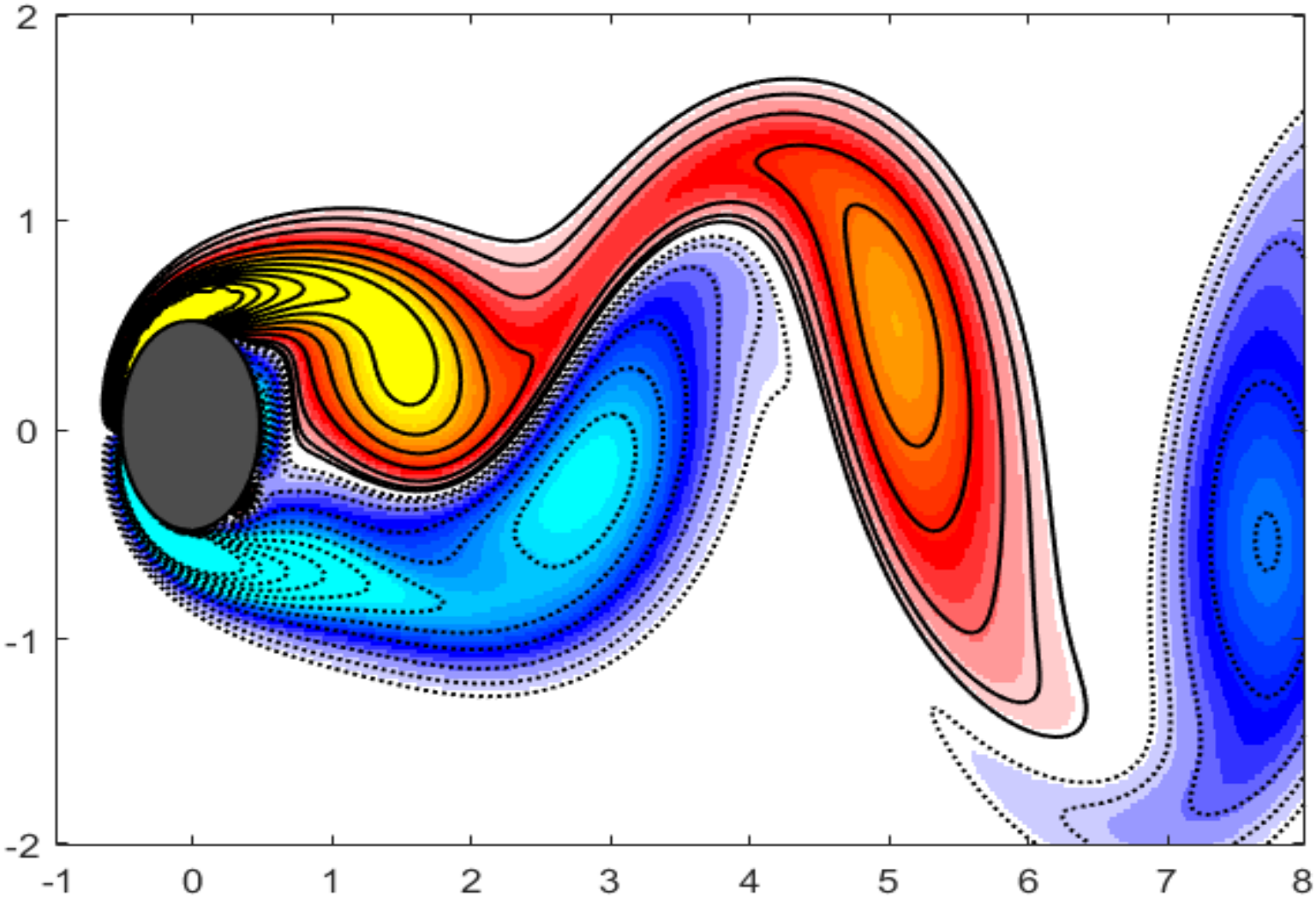

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

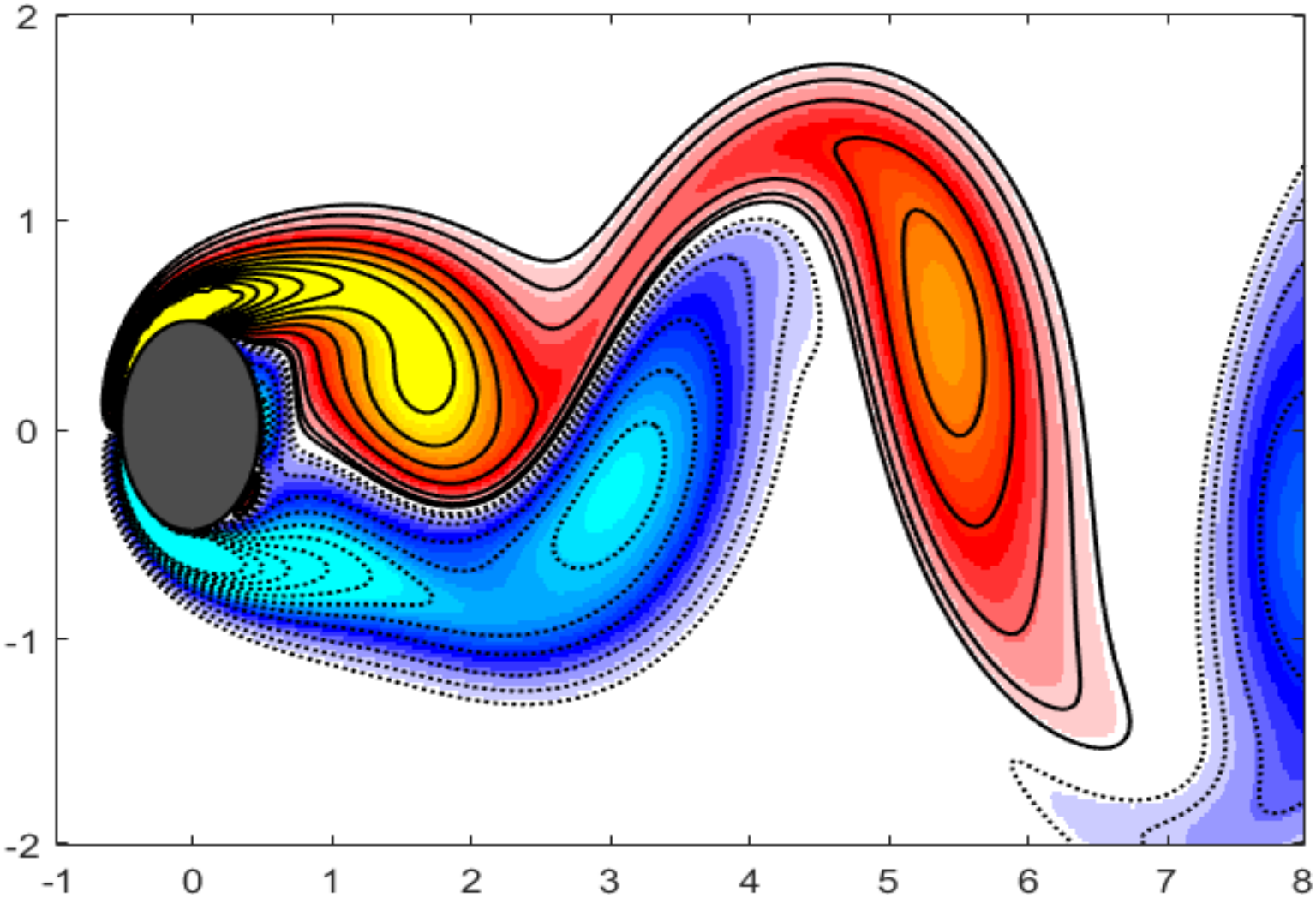

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8


2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

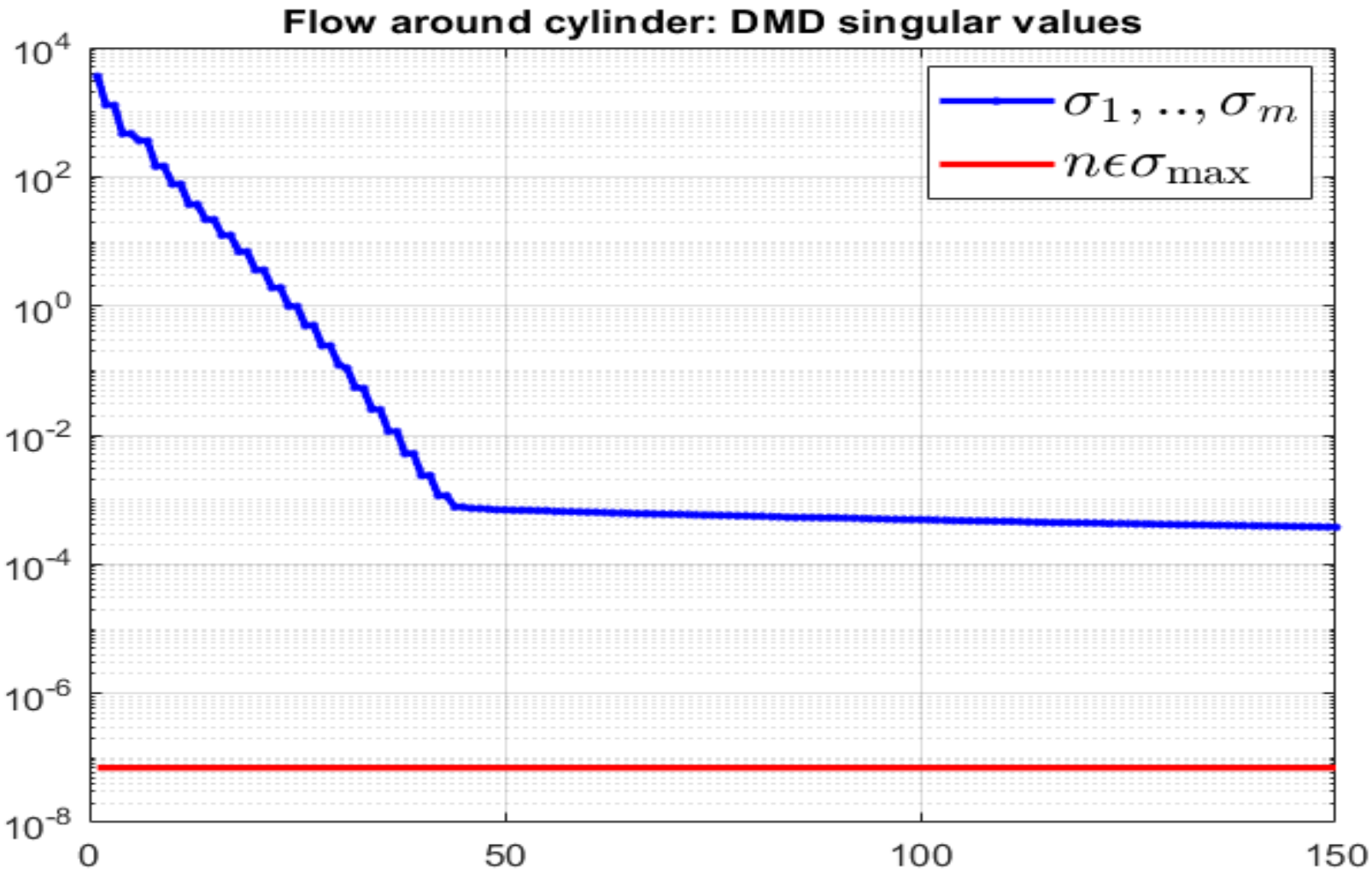

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8


2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8


2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

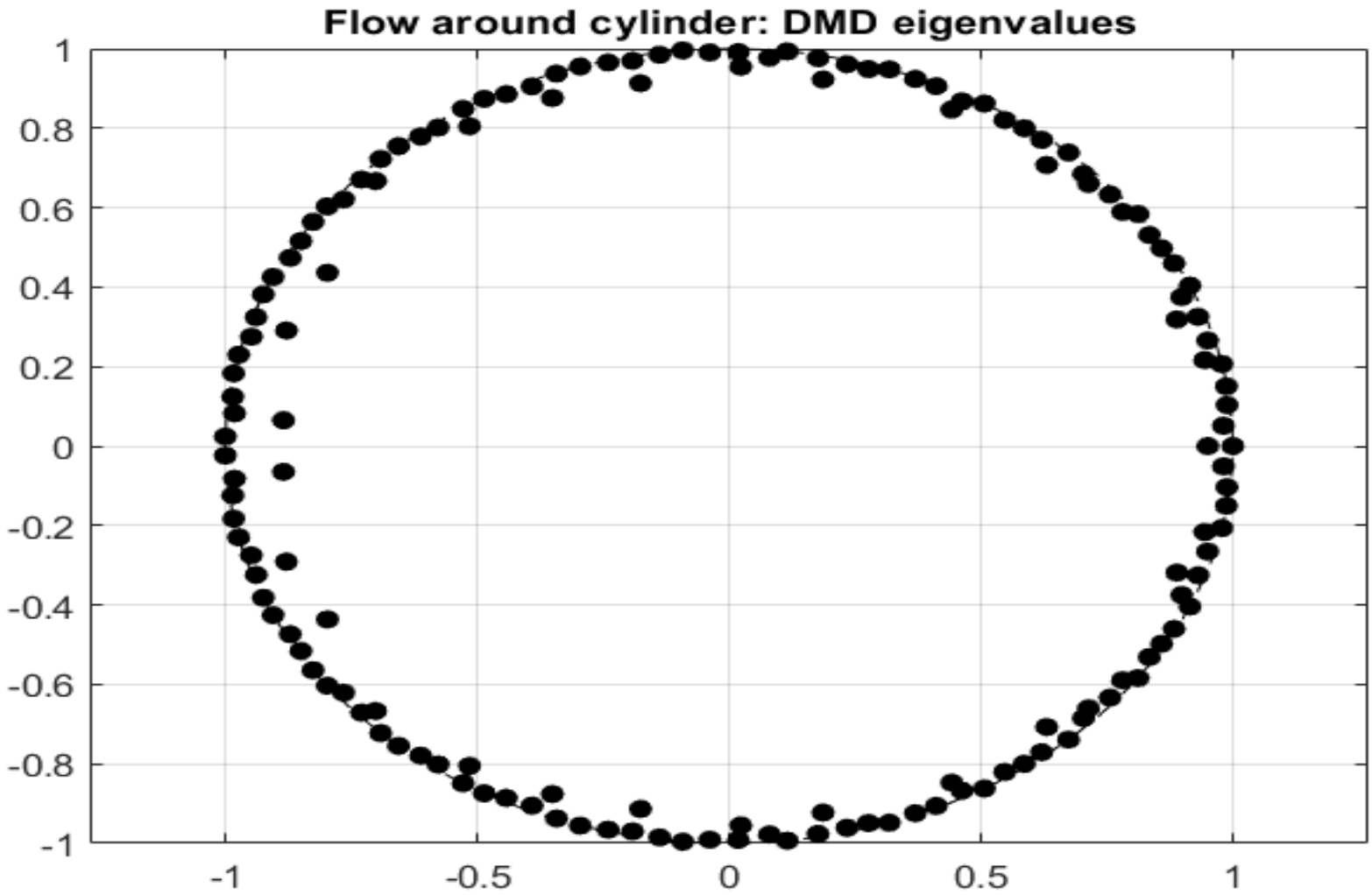

2
1
0
-1
-2
-1
0
1
2
3
4
5
6
7
8

Step 1: The SVD of $\mathbf{X}_m$



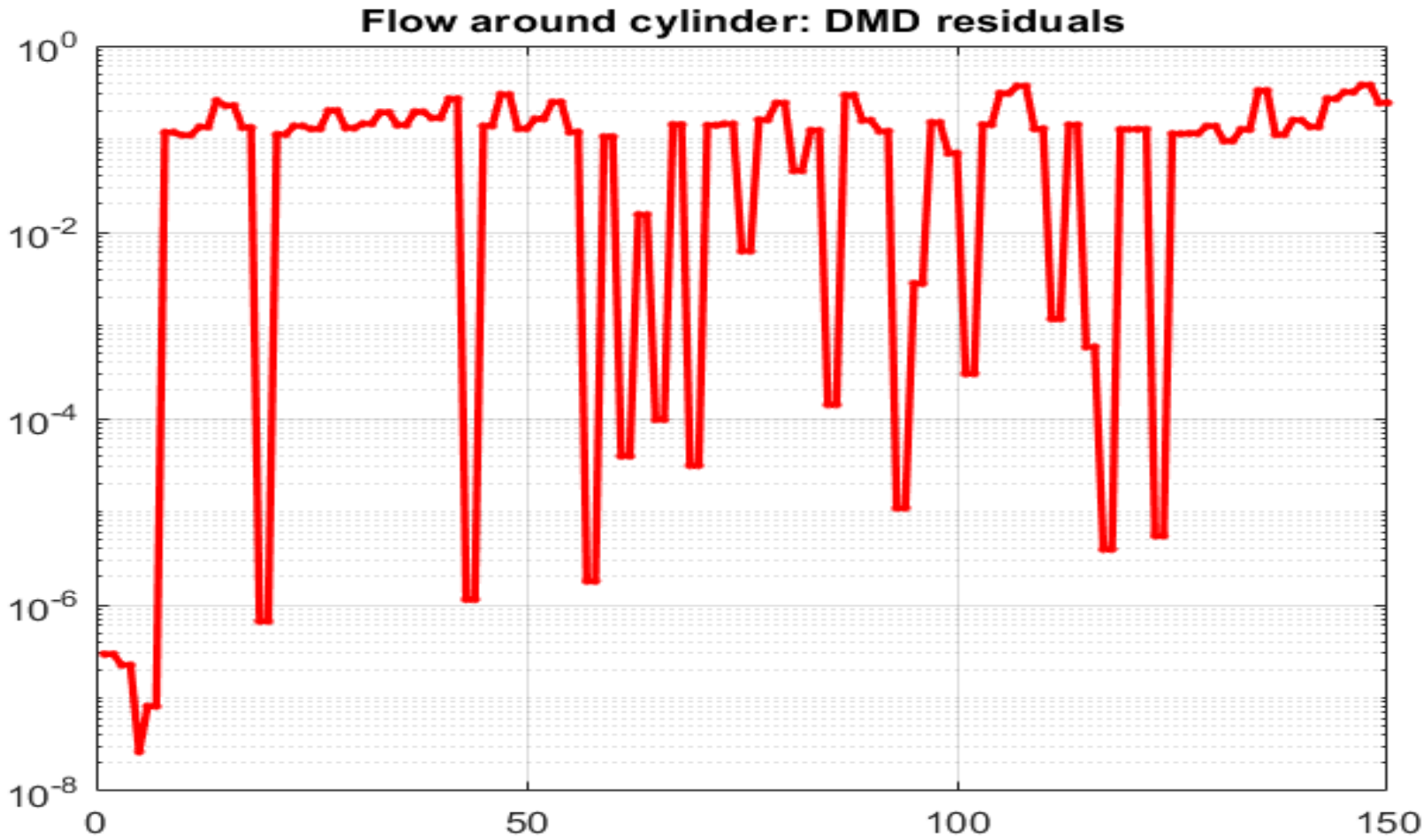
Condition number: $\kappa_2(\mathbf{X}_m) \approx 9.5e+06$.

Step 2: The Ritz values (DMD eigenvalues)



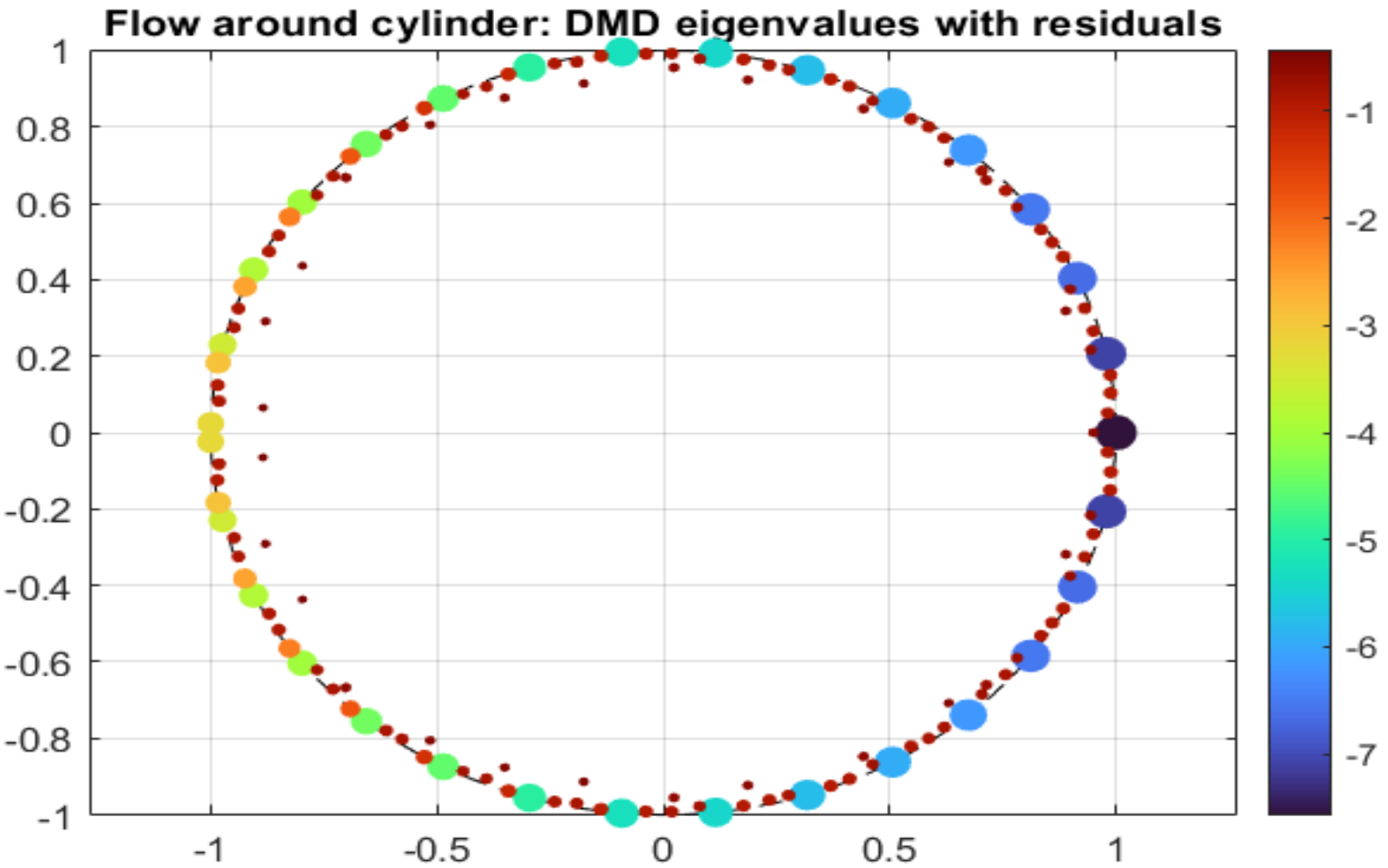
Are all good?

Step 3: The residuals



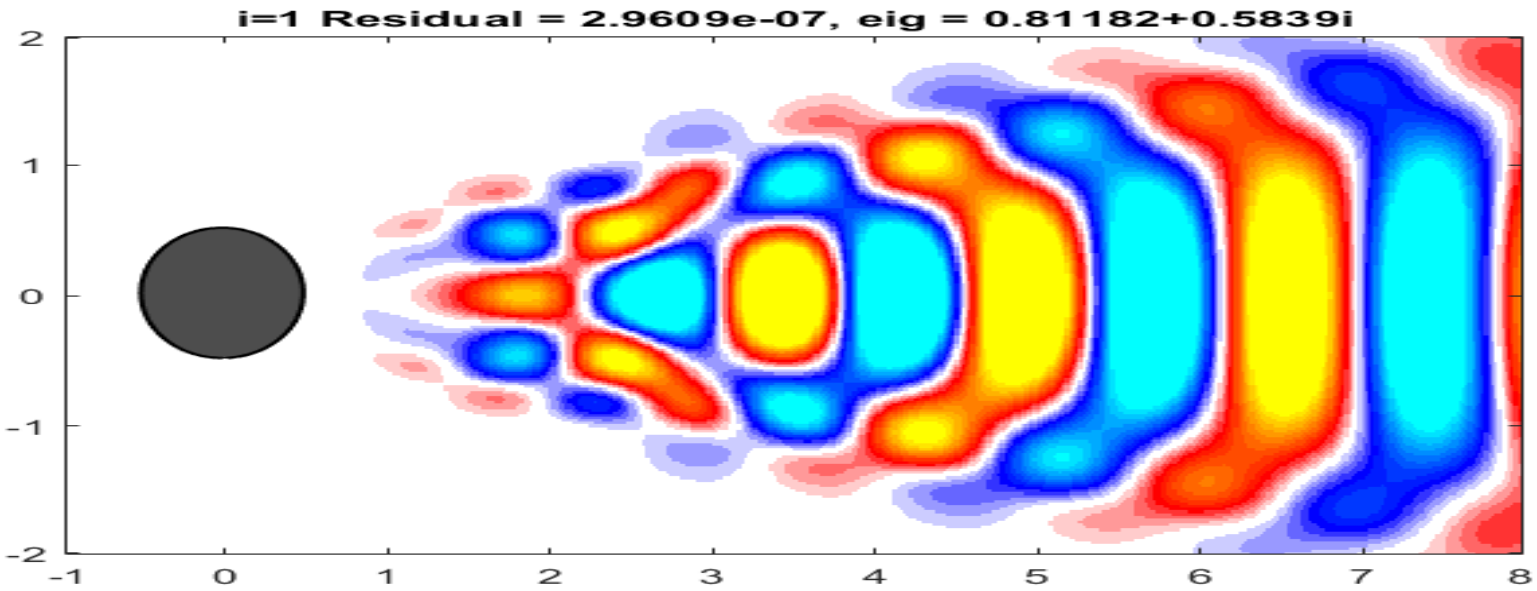
Are all good? Clearly not. Some Ritz pairs are not acceptable as approximate eigenpairs.

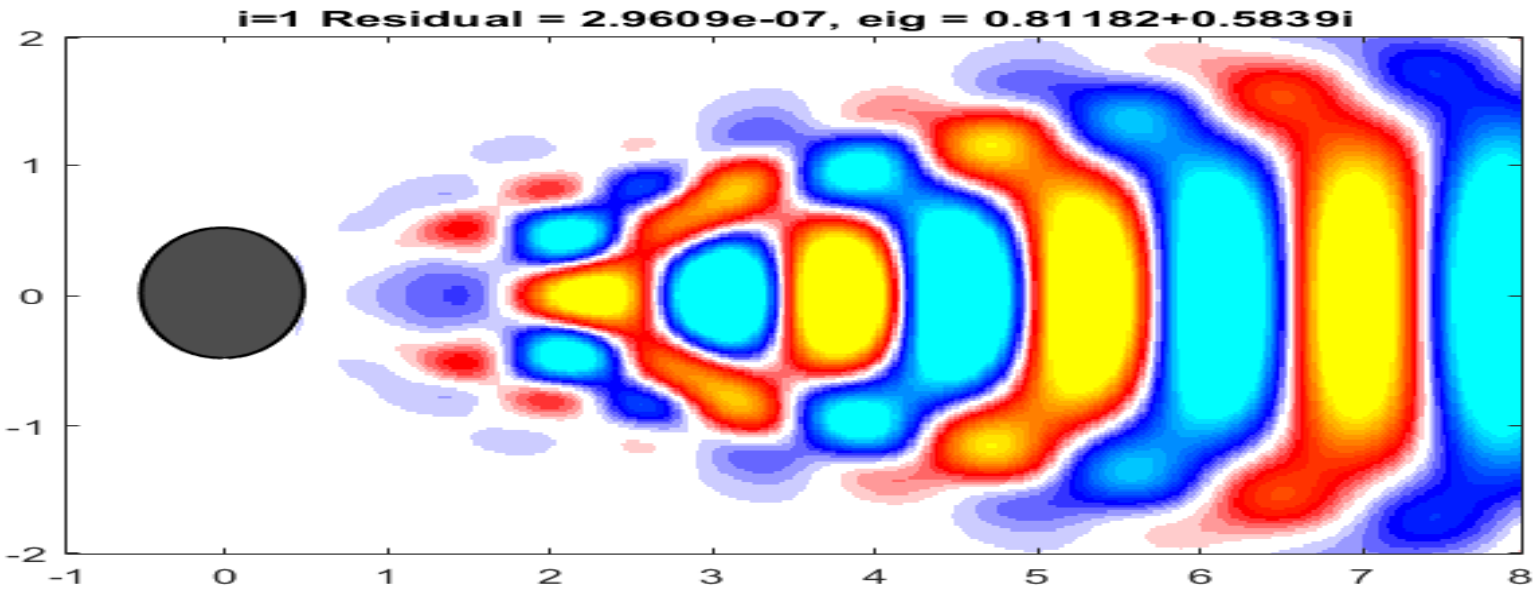
Step 2: The Ritz values with residuals



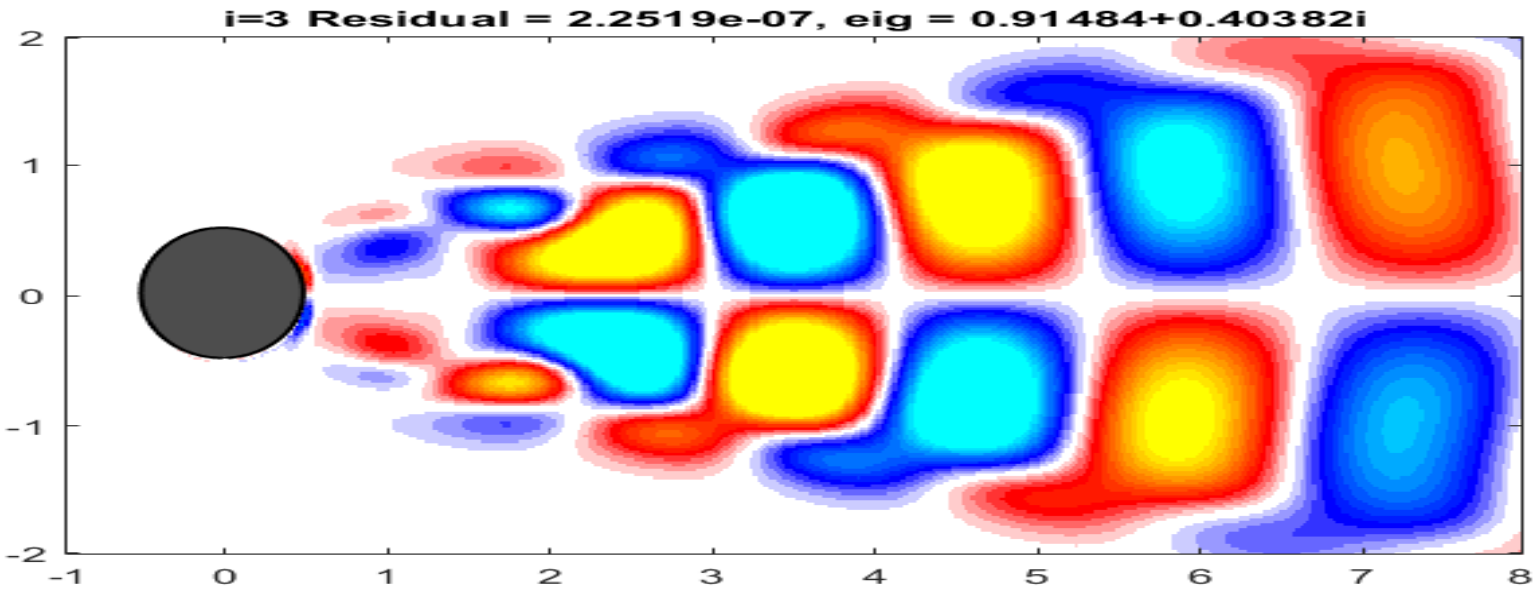
Note complex conjugate pairs. (The snapshots are real.)

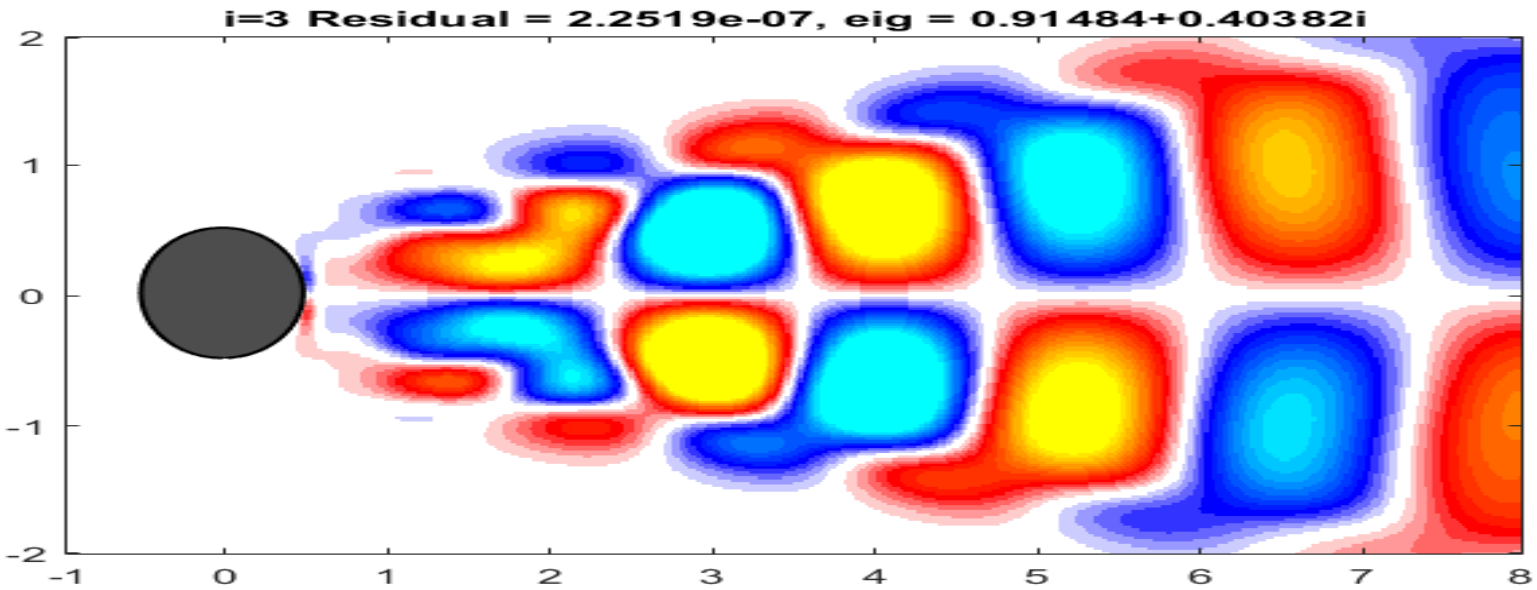
Step 2: The modes with residuals



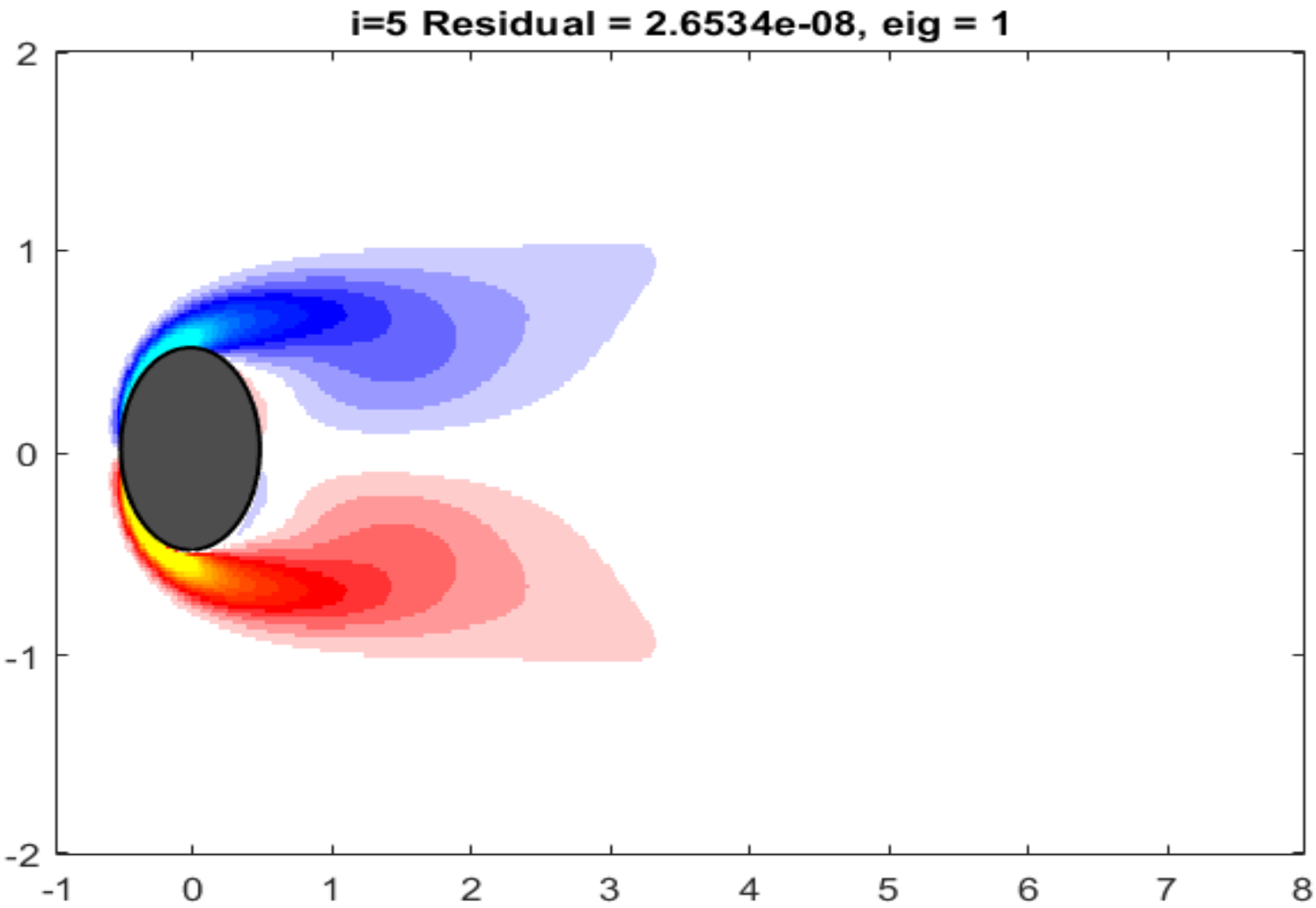


Step 2: The modes with residuals



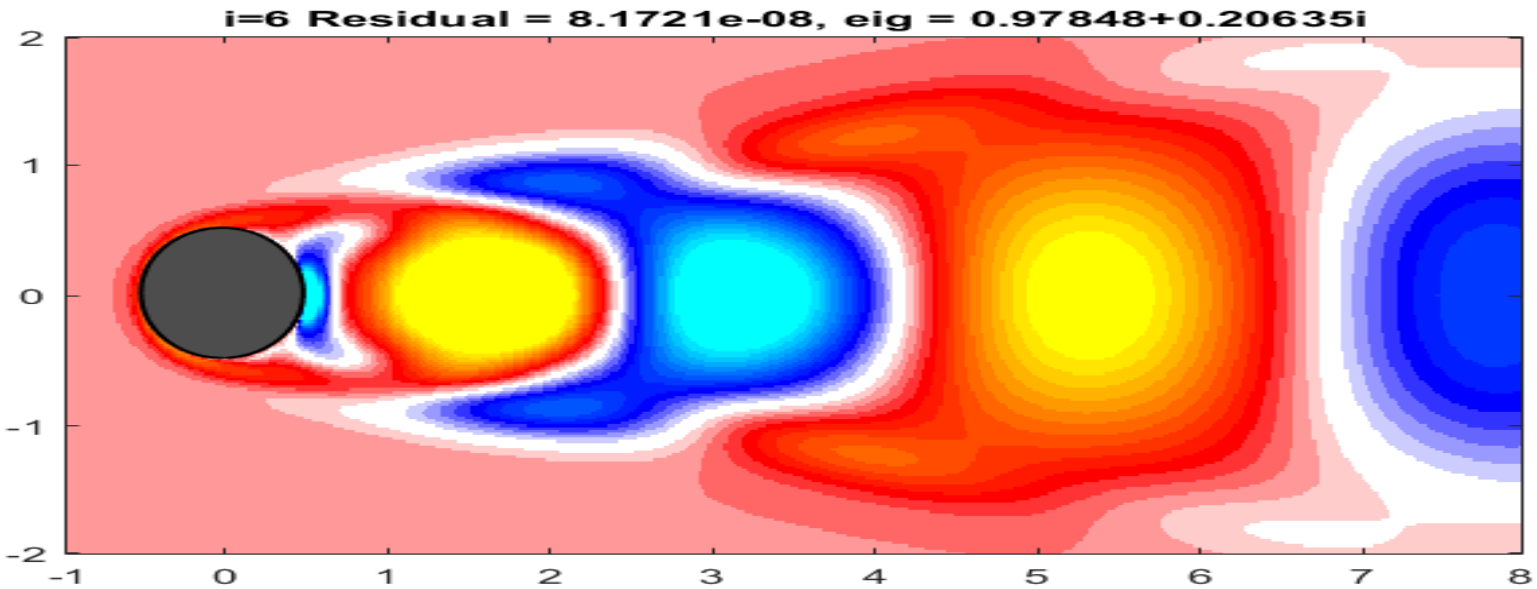


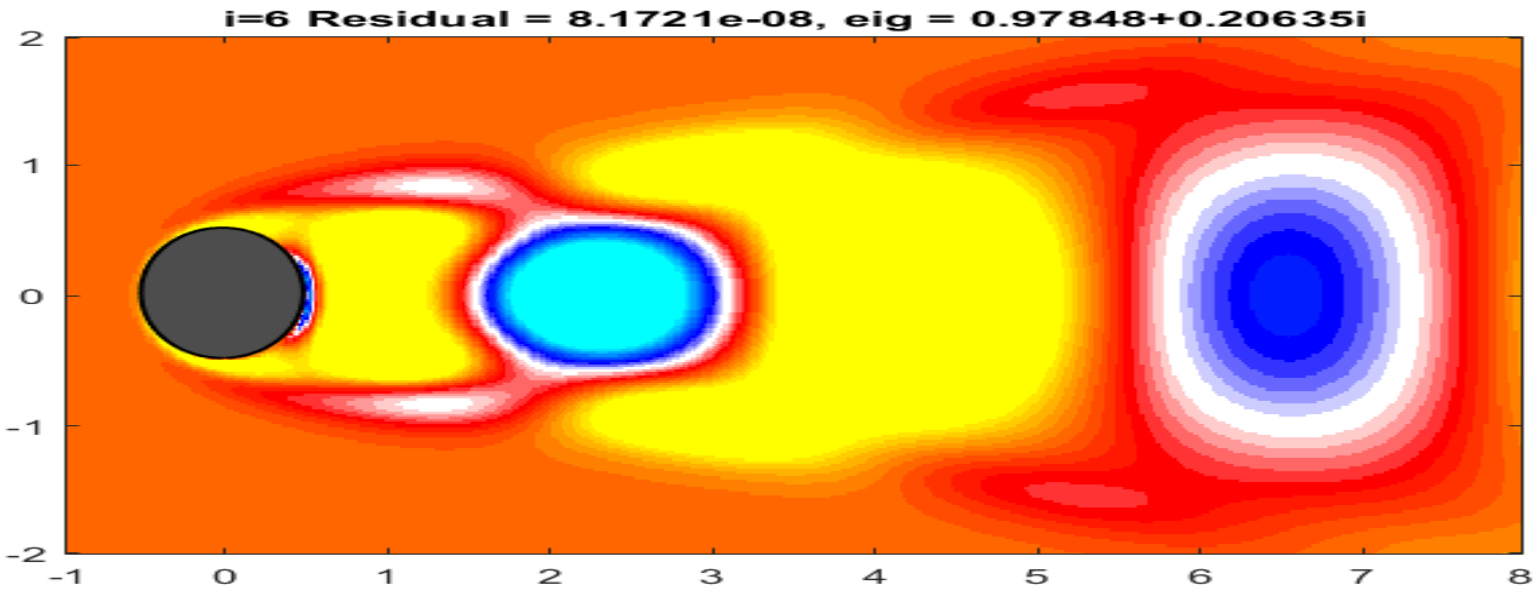
Step 2: The modes with residuals



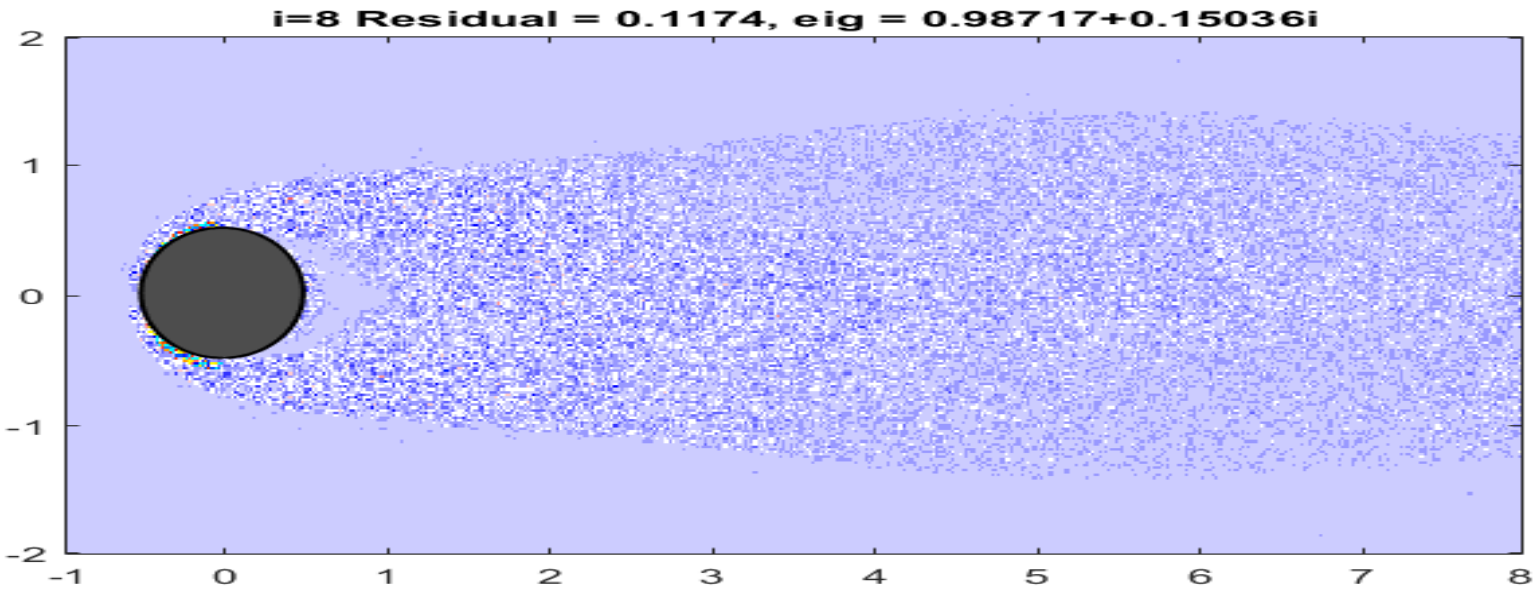
λ_5 is real and the mode is real.

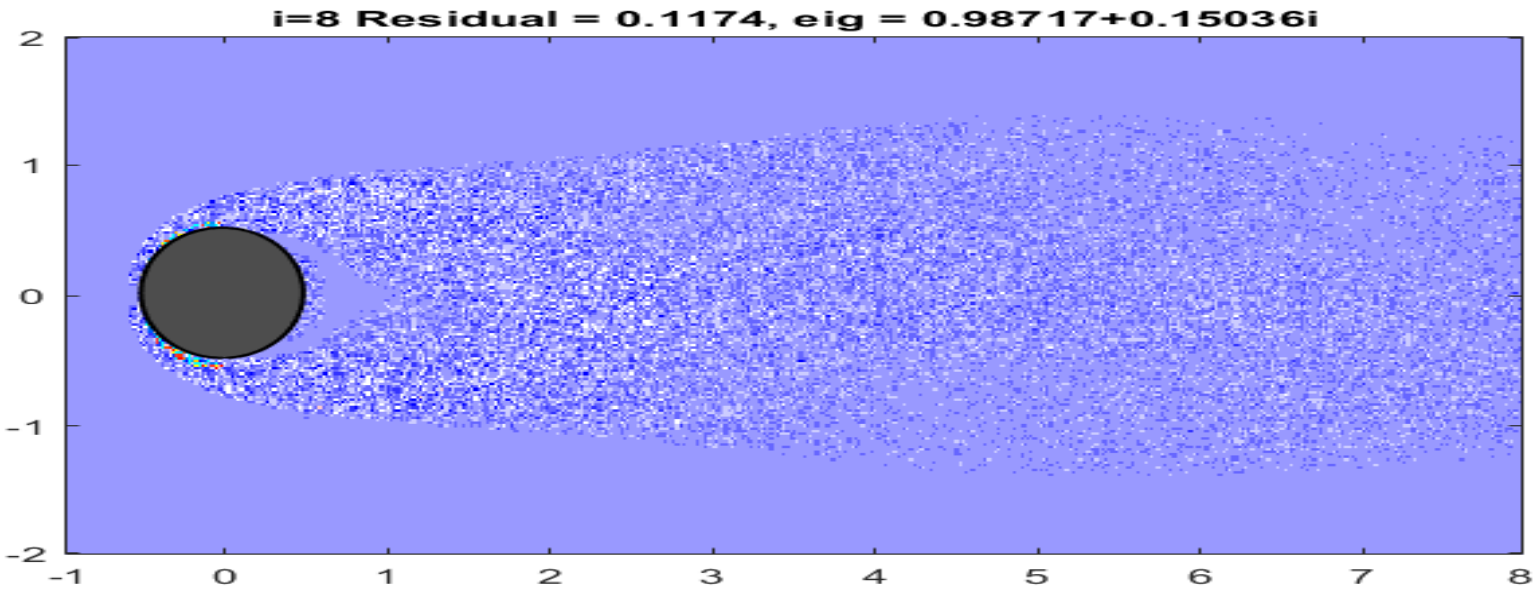
Step 2: The modes with residuals



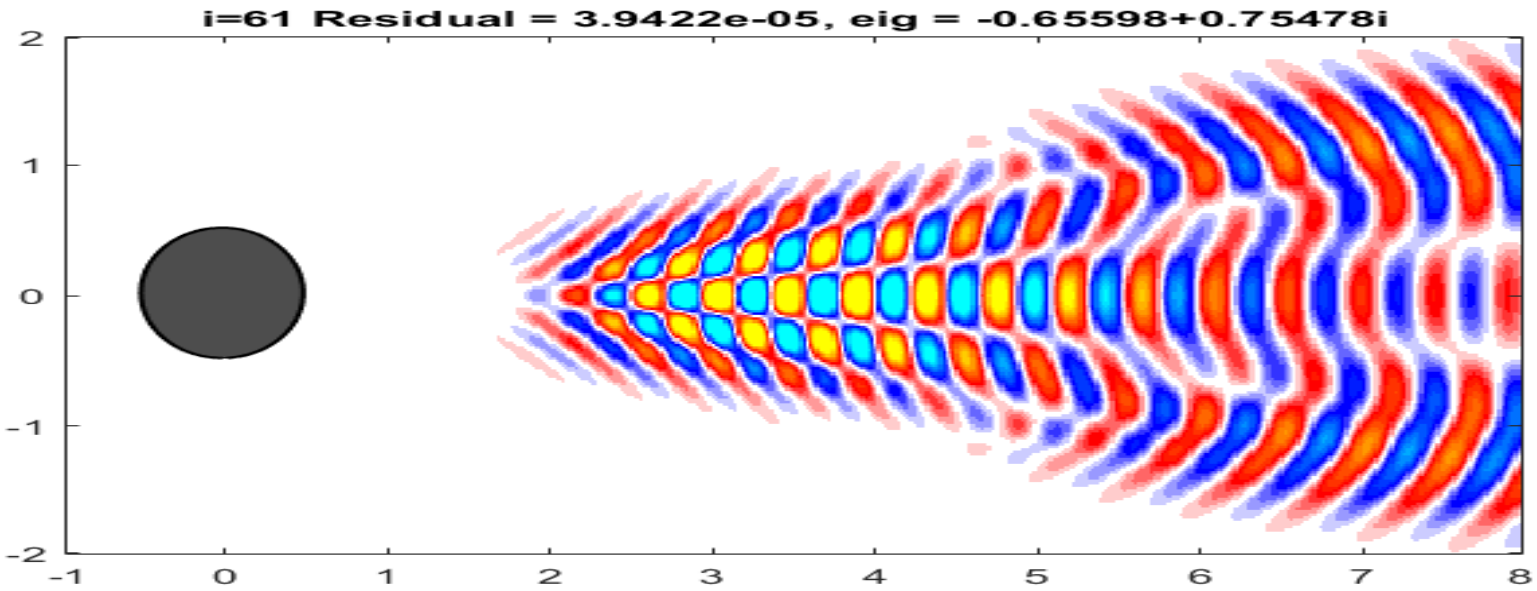


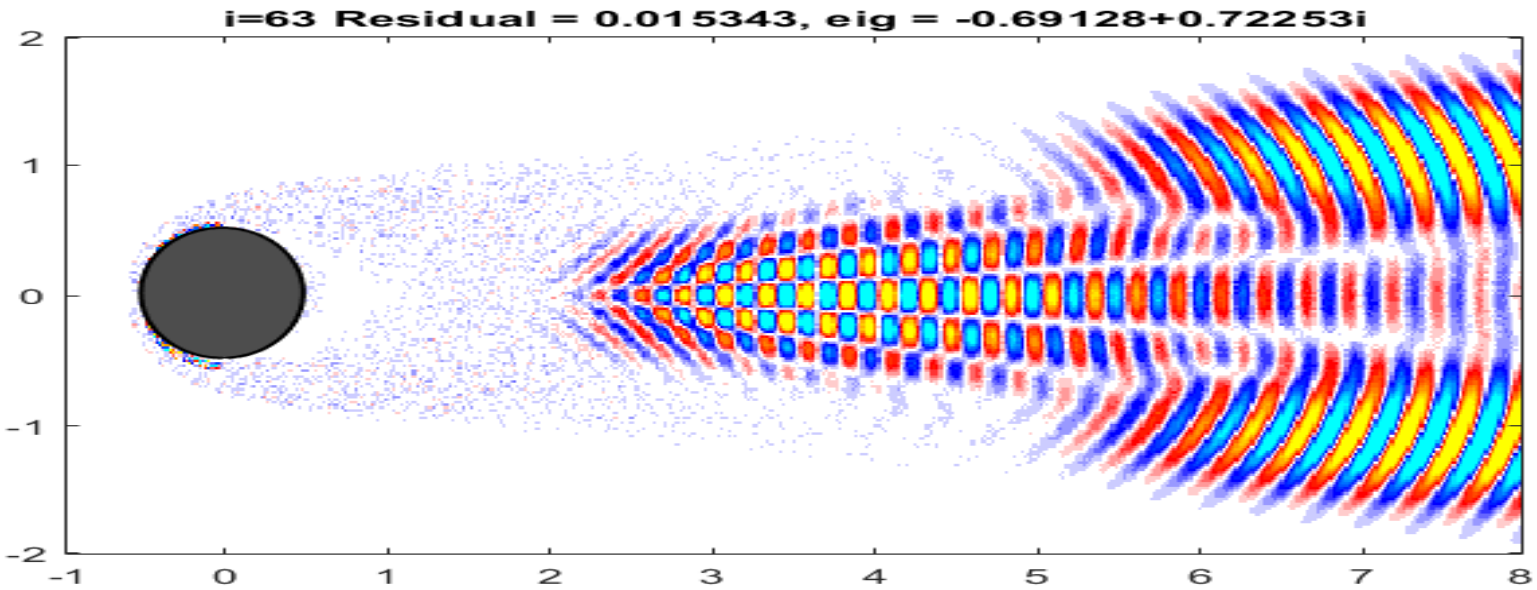
Step 2: The modes with residuals (large residual, bad)





Step 2: The modes with residuals





Exercise

Exercise

- Implement the companion matrix based DMD.
- Implement the Schmid's DMD, with the residuals.

Test both implementations on the following examples:

1. Synthetic examples: first generate A and use it to generate snapshots.
2. Selected examples from `http://dmdbook.com/`, in particular this cylinder flow example (The file DATA.zip).

Exercise

Select a data set from

`https://cgl.ethz.ch/research/visualization/data.php`

This requires some file format manipulations.

Exercise

If you have some data from your own research, try it.

QR compressed DMD

Let $n \gg m$, i.e. the snapshots are high dimensional but span a low dimensional subspace. The QR factorization can be used to generate an orthonormal basis in the subspace that contains all data snapshots (both $\mathbf{X}$ and $\mathbf{Y}$). Then, the DMD is applied to a new lower dimensional representation of the original data. More precisely, the QR factorization

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

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

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

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

$$\mathbf{X} = \widehat{Q}_1 U_x \Sigma_x V_x^* = U \Sigma V^*, \quad U = \widehat{Q}_1 U_x, \quad \Sigma = \Sigma_x, \quad V = V_x, \qquad (2)$$

is the SVD of $\mathbf{X}$.

QR compressed DMD

We select the numerical rank k as before, but using $R_{[11]}$ instead of $\mathbf{X}$, and then $U_k = \widehat{Q}_1 U_{xk}$, where $U_{xk} = U_x(:, 1:k)$ and

$$\begin{aligned} S_k &= U_k^* \mathbf{Y} V_k \Sigma_k^{-1} = U_{xk}^* \widehat{Q}_1^* (\widehat{Q}_1 R_{[12]} + \widehat{Q}_2 R_{[22]}) V_{xk} \Sigma_x(1:k, 1:k)^{-1} \\ &= U_{xk}^* R_{[12]} V_{xk} \Sigma_{xk}^{-1}. \end{aligned} \tag{3}$$

Note that here $\mathbf{X} = \widehat{Q}_1 R_{[11]}$ is the QR factorization of $\mathbf{X}$ and that the same S_k is obtained by computing $S_k = (U_k^* \mathbf{Y}) V_k \Sigma_k^{-1}$. How do we justify the extra effort to compute the QR factorization (1) of $(\mathbf{X}, \mathbf{Y})$?

- It starts paying off already when computing the SVD of $\mathbf{X}$.
- The matrix S_k can be computed with less effort, as we can use $R_{[12]}$.
- The refinement of the Ritz vectors can also be done in the $2m$-dimensional subspace.

Further advantages are ... $\rightsquigarrow$ ...

QR compressed DMD

- The residuals are an important information and the cost of computing the residuals is reduced because

$$r_i = \mathbf{Y} V_k \Sigma_k^{-1} w_i - \lambda_i U_k w_i = \widehat{Q}(R_{[:2]} V_{xk} \Sigma_k^{-1} w_i - \lambda_i U_{xk} w_i)$$

  so that $\|r_i\|_2 = \|R_{[:2]} V_{xk} \Sigma_k^{-1} w_i - \lambda_i U_{xk} w_i\|_2$ is computed more efficiently and the Ritz pairs can be selected using the computation in the $2m$-dimensional subspace. This avoids computation of the $n \times k$ matrix $\mathbf{Y} V_k$ ($(2m-1)nk$ flops) and for each w_i the norm $\|r_i\|_2$ is computed at a cost that does not involve n.
- Another argument is the spatio-temporal representation of the snapshots (6) that is accomplished by solving the structured least squares problem. Due to the unitary invariance of the norm $\|\cdot\|_2$, the optimization can be done (by keeping the modes in factored form) in the $2m$-dimensional (instead of n-dimensional) space.

QR compressed DMD

- The forward-backward DMD [Dawson et al. 2016] applies DMD twice – first with the data $(\mathbf{X}, \mathbf{Y})$ and then, backward in time, with $(\mathbf{Y}, \mathbf{X})$ so that both the SVD of $\mathbf{X}$ and of $\mathbf{Y}$ are computed. With the factorization (1), this means computing the SVD of $R_x = R_{[11]} \in \mathbb{C}^{m\times m}$ and of $R_y = R_{[:2]} \in \mathbb{C}^{2m\times m}$, which is much more efficient if $m \ll n$. Similar improvement of the Optimal DMD.
- In the case of extremely large dimension n, when the memory capacity and the cost of memory traffic become major issues, after computing the out-of-core QR factorization, we can compute the DMD in $2m$-dimensional subspace. On modern multi-core hardware, highly optimized implementations of the QR factorization of tall and skinny matrices is available [Demmel et al. 2012], [Ngyen, Demmel 2015].

Simplifies for one long trajectory $\mathbf{F} = (\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{x}_{m+1})$, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, $\mathbf{Y} = (\mathbf{x}_2, \ldots, \mathbf{x}_{m+1})$.

$[Z_k, \mathbf{\Lambda}_k, r_k, \rho_k] = \mathrm{DMDQR}(\mathbf{X}_{m+1}; \epsilon)$ {*QR Compressed DMD*}

Input:

- $\mathbf{X}_{m+1} = (\mathbf{f}_1, \ldots, \mathbf{f}_m, \mathbf{f}_{m+1})$ that defines a sequence of snapshots pairs $\mathbf{f}_{i+1} = \mathbb{A}\mathbf{f}_i$. (Tacit assumption is that n is large and that $m \ll n$.)
- Tolerance level ϵ for numerical rank determination.

1: $[\widehat{Q}_f, R_f] = qr(\mathbf{X}_{m+1}, 0)$; {*thin QR factorization*}
2: $R_x = R_f(1 : m+1, 1 : m)$, $R_y = R_f(1 : m+1, 2 : m+1)$; {*New representaitons of* $\mathbf{X}_m$, $\mathbf{Y}_m$.}
3: $[\widehat{Z}_k, \mathbf{\Lambda}_k, r_k, \rho_k] = \mathrm{DMD}(R_x, R_y; \epsilon)$; {*DMD in* $(m+1)$*-dimensional ambient space*}
4: $Z_k = \widehat{Q}_f \widehat{Z}_k$

Output: Z_k, $\mathbf{\Lambda}_k$, r_k, ρ_k

Streaming QR compressed DMD

Suppose a block $\mathbf{f}$ of $k \geq 1$ snapshots has been received and the QR compressed representation $\mathbf{F} = QR$ needs to be updated.

$$\mathbf{F}_{new} = (\mathbf{F}, \mathbf{f}) = \begin{pmatrix} Q & (\mathbb{I}_n - QQ^*)\mathbf{f} \end{pmatrix} \begin{pmatrix} R & Q^*\mathbf{f} \\ 0 & \mathbb{I}_k \end{pmatrix}. \tag{4}$$

Note that $\mathbf{f} - QQ^*\mathbf{f}$ is the Gram-Schmidt orthogonalization and that in floating point computation this step should be done with reorthogonalization. If $\mathbf{f} - Q(Q^*\mathbf{f}) = Q_1 R_1$ is the QR factorization, then

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

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

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

Streaming QR compressed DMD

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

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

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

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

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

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

$\mathbf{F}_{new} = Q_{new} R_{new}$, $Q_{new} = Q[(H_1^* H_2^* H_3^*)](:, end - \ell)$. Clearly, the product $H_1^* H_2^* H_3^* \cdots$ is first accumulated and then applied using BLAS 3.

Exercise

Exercise

- Implement the QR compressed DMD and test it on the examples used to test the DMD. (See previous Exercise.)
- Implement updating/downdating. Use a sliding data window (keep adding new and discarding old data) and update/downdate the QR compressed representation of the snapshots.
- A research project: combine this with updating the SVD of the compressed representations. Explore the literature on the online/streaming DMD.

DMD: Data driven spectral analysis of $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{i+1} \approx \mathbb{A}\mathbf{f}_i$

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

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

$$\mathbb{A}z_j \approx \lambda_j z_j, \quad \lambda_j = |\lambda_j| e^{\mathrm{i}\omega_j \delta t}, \quad j = 1, \ldots, k; \quad k \leq m. \tag{5}$$

Completed. (λ_j, z_j), $\|\mathbb{A}z_j - \lambda_j z_j\|_2$ **available for** $j = 1, \ldots, m$.

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

$$\mathbf{f}_i \approx \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j \lambda_{\varsigma_j}^{i-1} \equiv \sum_{j=1}^{\ell} z_{\varsigma_j} \alpha_j |\lambda_{\varsigma_j}|^{i-1} e^{\mathrm{i}\omega_{\varsigma_j}(i-1)\delta t}, \quad i = 1, \ldots, m. \tag{6}$$

The decomposition of the snapshots (6) 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 can also be used for forecasting. It is desirable to have small number ℓ of the most important modes $z_{\varsigma_1}, \ldots, z_{\varsigma_\ell}$, $\varsigma_j \in \{1, \ldots, k\}$. (To ease the notation, $\varsigma_j = j$.)

Snapshot reconstruction – some theory

Wanted are the coefficients $\alpha_1, \ldots, \alpha_\ell$ that minimize

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

If we set $Z_\ell = (z_1, \ldots, z_\ell)$, $\vec{\boldsymbol{\alpha}} = (\alpha_1, \ldots, \alpha_\ell)^T$, $\Delta_{\boldsymbol{\alpha}} = \mathrm{diag}(\vec{\boldsymbol{\alpha}})$, $\Lambda_j = (\lambda_1^{j-1}, \ldots, \lambda_\ell^{j-1})^T$, and $\Delta_{\Lambda_j} = \mathrm{diag}(\Lambda_j)$ then the objective (1) reads

$$\Omega^2(\boldsymbol{\alpha}) \equiv \| \mathbf{X}_m - Z_\ell \Delta_{\boldsymbol{\alpha}} \begin{pmatrix} \Lambda_1 & \Lambda_2 & \ldots & \Lambda_m \end{pmatrix} \|_F^2 \longrightarrow \min . \tag{2}$$

Compute the tall QR factorization $Z_\ell = QR$ and define projected snapshots $\mathbf{g}_i = Q^* \mathbf{f}_i$, then the LS problem can be compactly written as

$$\| \vec{\mathbf{g}} - S \vec{\boldsymbol{\alpha}} \|_2 \longrightarrow \min, \quad \vec{\mathbf{g}} = \begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_m \end{pmatrix}, \quad S = (I_m \otimes R) \begin{pmatrix} \Delta_{\Lambda_1} \\ \vdots \\ \Delta_{\Lambda_m} \end{pmatrix} \equiv \begin{pmatrix} R\Delta_{\Lambda_1} \\ \vdots \\ R\Delta_{\Lambda_m} \end{pmatrix} .$$

Discussion: The structure of Z_ℓ and of the QR factorization $Z_\ell = QR$.

Hadamard, Kronecker and Khatri-Rao products

Hadamard matrix product $C = A * B$ ($C = A \circ B$)

For $A, B \in \mathbb{R}^{m \times n}$, the Hadamard product $C = A * B$ is defined by $c_{ij} = a_{ij} b_{ij}$.

Kronecker matrix product $C = A \otimes B$

For $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$ Kronecker product $A \otimes B$ is defined as

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}$$

It is well defined for A and B of any dimensions, and $A \otimes B$ is $m \cdot p \times n \cdot q$.

$$\begin{pmatrix} \bullet & \bullet & \bullet \end{pmatrix} \otimes \begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

Kronecker product: basic properties

If $A = (a_1, \ldots, a_m)$, $B = (b_1, \ldots, b_q)$ denote column partitions, then

$$\begin{aligned} A \otimes B &= (a_1 \otimes B, a_2 \otimes B, \ldots, a_n \otimes B) \\ &= (a_1 \otimes b_1, a_1 \otimes b_2, \ldots, a_1 \otimes b_q, a_2 \otimes b_1, \ldots, a_n \otimes b_1, \ldots, a_n \otimes b_q) \end{aligned}$$

Basic properties:

- $A \otimes (\alpha B) = \alpha(A \otimes B)$
- $(A + B) \otimes C = A \otimes C + B \otimes C$; $A \otimes (B + C) = A \otimes B + A \otimes C$
- $(A \otimes B) \otimes C = A \otimes (B \otimes C)$
- $(A \otimes B)^T = A^T \otimes B^T$; $(A \otimes B)^* = A^* \otimes B^*$
- $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ (for well defined AC, BD)
- $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ (for regular A, B); $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$
- $\mathrm{vec}(CXD) = (D^T \otimes C)\mathrm{vec}(X)$
- $\mathrm{vec}(xy^T) = y \otimes x$ (x, y column vectors)
- $\mathrm{vec}((x, \widehat{x}) \begin{pmatrix} y^T \\ \widehat{y}^T \end{pmatrix}) = y \otimes x + \widehat{y} \otimes \widehat{x}$

Khatri-Rao product

Khatri-Rao product $C = A \odot B$

Let $A = (a_1, \ldots, a_n) \in \mathbb{R}^{m \times n}$, $B = (b_1, \ldots, b_n) \in \mathbb{R}^{p \times n}$ be column partitions. The Khatri-Rao product of A and B is defined as

$$A \odot B = (a_1 \otimes b_1, a_2 \otimes b_2, \ldots, a_{n-1} \otimes b_{n-1}, a_n \otimes b_n)$$

Basic properties:

- $(A \odot B) \odot C = A \odot (B \odot C)$
- $(A \odot B)^T (A \odot B) = (A^T A) * (B^T B)$ (here $*$ denotes the Hadamard product)
- $(A \odot B)^\dagger = ((A^T A) * (B^T B))^\dagger (A \odot B)^T$

$C \otimes B$ and $C \odot B$

Small dimension example:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = (c_1, c_2), \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = (b_1, b_2)$$

$$C \otimes B = \begin{pmatrix} c_{11}b_{11} & c_{11}b_{12} & c_{12}b_{11} & c_{12}b_{12} \\ c_{11}b_{21} & c_{11}b_{22} & c_{12}b_{21} & c_{12}b_{22} \\ c_{21}b_{11} & c_{21}b_{12} & c_{22}b_{11} & c_{22}b_{12} \\ c_{21}b_{21} & c_{21}b_{22} & c_{22}b_{21} & c_{22}b_{22} \end{pmatrix}$$

$C \odot B$ is a submatrix of $C \otimes B$:

$$C \odot B = \begin{pmatrix} c_{11}b_{11} & c_{12}b_{12} \\ c_{11}b_{21} & c_{12}b_{22} \\ c_{21}b_{11} & c_{22}b_{12} \\ c_{21}b_{21} & c_{22}b_{22} \end{pmatrix} = \begin{pmatrix} c_1 \otimes b_1 & c_2 \otimes b_2 \end{pmatrix}$$

Solution(s) by generalized inverse(s)

The optimal solution is obtained as $\vec{\alpha} = S^{\dagger}\vec{\mathbf{g}}$ using the explicit normal equations formula, DMDSP [Jovanović+Schmid+Nichols]. Here $S^{\dagger}$ is the Moore-Penrose generalized inverse.
On the other hand, deploying the reflexive g–inverse of S,

$$S^{-} = \begin{pmatrix} \Delta_{\Lambda_1} \\ \vdots \\ \Delta_{\Lambda_m} \end{pmatrix}^{\dagger} (I \otimes R^{-1}),$$

we obtain interesting explicit formulas for $\vec{\alpha}_{\star} = S^{-}\vec{\mathbf{g}}$ that reveal a relation to the Generalized Laplace Analysis, GLA, [Mezić+Mohr].

$S^{-} \neq S^{\dagger}$

Recall that, by definition, S^{-} satisfies
$SS^{-}S = S$, $S^{-}SS^{-} = S^{-}$and $S^{-}S = (S^{-}S)^{*}$. In fact, since
$SS^{-} \neq (SS^{-})^{*}$, we have in general that $S^{-} \neq S^{\dagger}$, so $\vec{\alpha}_{\star} \neq S^{\dagger}\vec{\mathbf{g}}$.

$\vec{\alpha}_\star = S^{-}\vec{\mathbf{g}}$ solves a weighted LS problem

$$\vec{\boldsymbol{\alpha}}_\star = \begin{pmatrix} \Delta_{\Lambda_1} \\ \vdots \\ \Delta_{\Lambda_m} \end{pmatrix}^{\dagger} (I \otimes R^{-1}) \begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_m \end{pmatrix} = (\sum_{k=1}^{m} \Delta_{\Lambda_k}^{*} \Delta_{\Lambda_k})^{-1} \sum_{i=1}^{m} \Delta_{\Lambda_i}^{*} (R^{-1}\mathbf{g}_i)$$

$$= \sum_{i=1}^{m} \begin{pmatrix} \frac{\overline{\lambda}_1^{i-1}}{\sum_{k=1}^{m} |\lambda_1|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_1 \\ \frac{\overline{\lambda}_2^{i-1}}{\sum_{k=1}^{m} |\lambda_2|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_2 \\ \vdots \\ \frac{\overline{\lambda}_\ell^{i-1}}{\sum_{k=1}^{m} |\lambda_\ell|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_\ell \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} \frac{\overline{\lambda}_1^{i-1}}{\sum_{k=1}^{m} |\lambda_1|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_1 \\ \sum_{i=1}^{m} \frac{\overline{\lambda}_2^{i-1}}{\sum_{k=1}^{m} |\lambda_2|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_2 \\ \vdots \\ \sum_{i=1}^{m} \frac{\overline{\lambda}_\ell^{i-1}}{\sum_{k=1}^{m} |\lambda_\ell|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_\ell \end{pmatrix}$$

Theorem

Let $M = I \otimes (RR^*)^{-1}$, $(x, y)_M = y^* M x$, and let $\|x\|_M = \sqrt{x^* M x}$. Then $\vec{\boldsymbol{\alpha}}_\star$ is the minimum $\|\cdot\|_2$–norm solution of the weighted least squares problem

$$\|\vec{\mathbf{g}} - S\vec{\boldsymbol{\alpha}}\|_M \longrightarrow \min\,. \tag{3}$$

Comparison with GLA reconstruction [Mezić+Mohr]

$$\vec{\boldsymbol{\alpha}}_\star = \begin{pmatrix} \sum_{i=1}^m \frac{\overline{\lambda}_1^{i-1}}{\sum_{k=1}^m |\lambda_1|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_1 \\ \sum_{i=1}^m \frac{\overline{\lambda}_2^{i-1}}{\sum_{k=1}^m |\lambda_2|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_2 \\ \vdots \\ \sum_{i=1}^m \frac{\overline{\lambda}_\ell^{i-1}}{\sum_{k=1}^m |\lambda_\ell|^{2(k-1)}} (R^{-1}\mathbf{g}_i)_\ell \end{pmatrix}$$

$$\vec{\boldsymbol{\alpha}}_{(GLA)} = \frac{1}{m} \sum_{i=1}^m \Lambda_\ell^{-i+1} R^{-1}\mathbf{g}_i = \begin{pmatrix} \frac{1}{m} \sum_{i=1}^m \lambda_1^{-i+1} (R^{-1}\mathbf{g}_i)_1 \\ \frac{1}{m} \sum_{i=1}^m \lambda_2^{-i+1} (R^{-1}\mathbf{g}_i)_2 \\ \vdots \\ \frac{1}{m} \sum_{i=1}^m \lambda_\ell^{-i+1} (R^{-1}\mathbf{g}_i)_\ell \end{pmatrix}.$$

Proposition

When the spectrum of $\mathbb{A}$ lies on the unit circle, $\vec{\boldsymbol{\alpha}}_\star = \vec{\boldsymbol{\alpha}}_{(GLA)}$.

For more see recent paper [Drmač+Mezić+Mohr].

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

$$\sum_{i=1}^{m} \mathfrak{w}_i^2 \|\mathbf{f}_i - \sum_{j=1}^{\ell} z_j \alpha_j \lambda_j^{i-1}\|_2^2 \longrightarrow \min. \tag{4}$$

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 $\mathbf{\Lambda} = \mathrm{diag}(\lambda_j)_{j=1}^{\ell}$,

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

and write the objective (4) as the function of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\ell)^T$,

$$\Omega^2(\boldsymbol{\alpha}) \equiv \| \left[\mathbf{X}_m - Z_\ell \Delta_{\boldsymbol{\alpha}} \begin{pmatrix} \Lambda_1 & \Lambda_2 & \ldots & \Lambda_m \end{pmatrix} \right] \mathbf{W} \|_F^2 \longrightarrow \min, \tag{5}$$

$$\begin{pmatrix} \Lambda_1 & \Lambda_2 & \ldots & \Lambda_m \end{pmatrix} = \begin{pmatrix} 1 & \lambda_1 & \ldots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \ldots & \lambda_2^{m-1} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & \lambda_\ell & \ldots & \lambda_\ell^{m-1} \end{pmatrix} \equiv \mathbb{V}_{\ell,m} \in \mathbb{C}^{\ell \times m}. \tag{6}$$

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

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

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 (4) is

$$\boldsymbol{\alpha} = [(R^*R) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})]^{-1}[(\overline{\mathbb{V}_{\ell,m}\mathbf{W}} \circ (R^*G\mathbf{W}))\mathbf{e}], \tag{7}$$

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,

$$\boldsymbol{\alpha} = [(Z_\ell^*Z_\ell) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})]^{-1}[(\overline{\mathbb{V}_{\ell,m}\mathbf{W}} \circ (Z_\ell^*\mathbf{X}_m\mathbf{W}))\mathbf{e}]. \tag{8}$$

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? Let us experiment with a small dimension example.

Squaring the condition number – loosing 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 $\kappa_2(\mathbb{V}_{\ell,m}) \approx 10^9$, $\kappa_2(R) \approx 10^9 \ll 1/\texttt{roundoff}_{64} \approx 4.5 \cdot 10^{15}$.

```
>> 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
```

How accurately we can solve with $C =$ (R'*R).*(Vlm*Vlm')?

Based on [Demmel], we know that floating point Cholesky factorization $C = LL^*$ (L lower triangular with positive diagonal) of C is feasible if the matrix $C_s = (c_{ij}/\sqrt{c_{ii}c_{jj}})_{i,j=1}^{\ell}$ is well conditioned. Further, if we solve the linear system $Cx = b \neq 0$ using the Cholesky factor in the forward and backward substitutions, then the computed solution $\widetilde{x}$ satisfies

$$\frac{\|D_C(\widetilde{x} - C^{-1}b)\|_2}{\|D_C\widetilde{x}\|_2} \leq g(\ell)\boldsymbol{\varepsilon}\kappa_2(C_s), \tag{9}$$

where $g(\ell)$ is modest function of the dimension, $D_C = \mathrm{diag}(\sqrt{c_{ii}})_{i=1}^{\ell}$. Note that this implies component-wise error bound for each $\widetilde{x}_i \neq 0$:

$$\frac{|\widetilde{x}_i - (C^{-1}b)_i|}{|\widetilde{x}_i|} \leq \underbrace{\left[\frac{\|D_C\widetilde{x}\|_2}{\sqrt{c_{ii}}|\widetilde{x}_i|}\right]}_{\geq 1} g(\ell)\boldsymbol{\varepsilon}\kappa_2(C_s). \tag{10}$$

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

$$\max(\lambda_{\min}(A_s), \lambda_{\min}(B_s)) \leq \lambda_i(C_s) \leq \min(\lambda_{\max}(A_s), \lambda_{\max}(B_s)). \quad (11)$$

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

$$\kappa_2(C_s) \leq \min(\kappa_2(R_c)^2, \kappa_2((\mathbb{V}_{\ell,m}\mathbf{W})_r)^2).$$

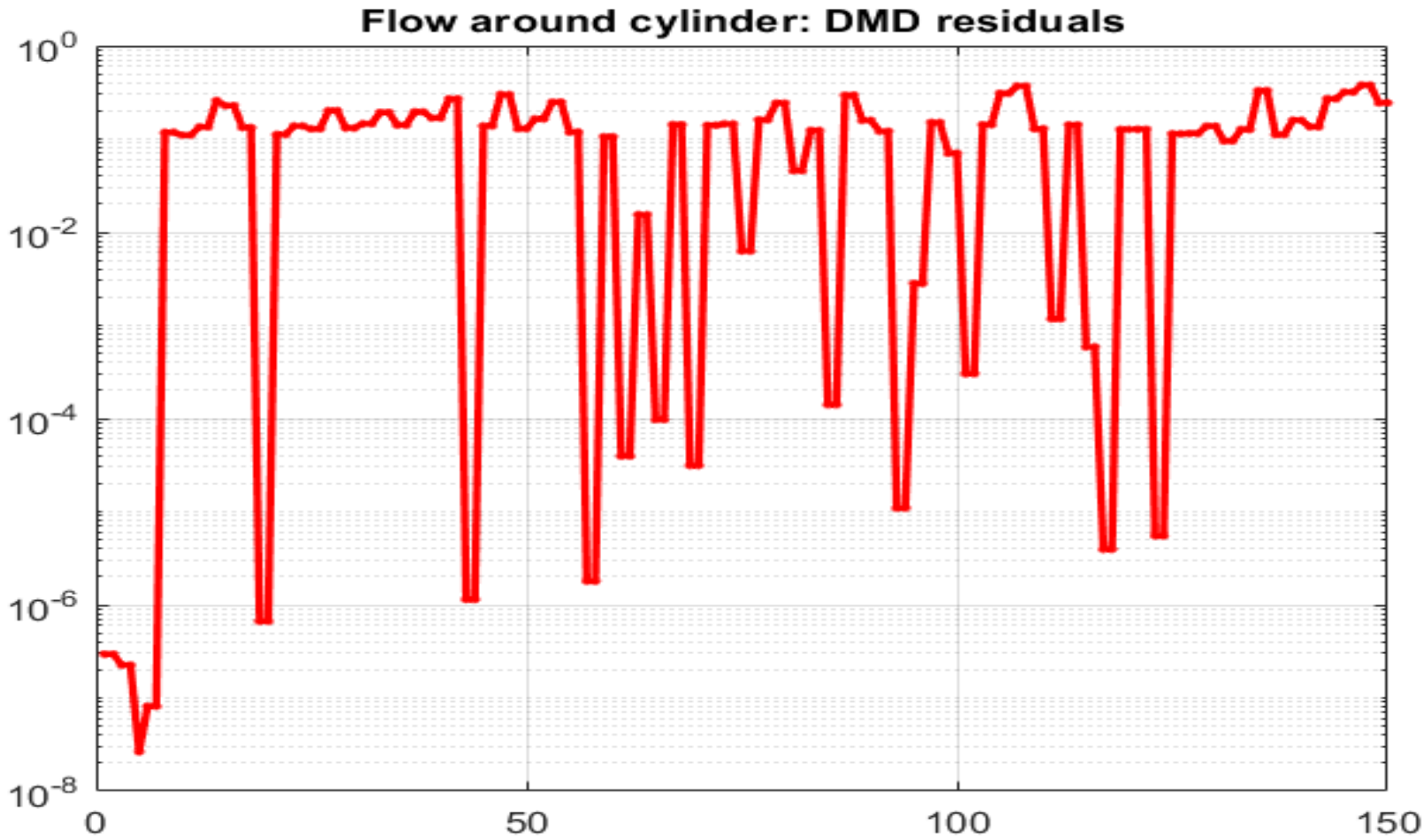
Example: Flow around a cylinder.

We continue with the CFD example

The goals are:

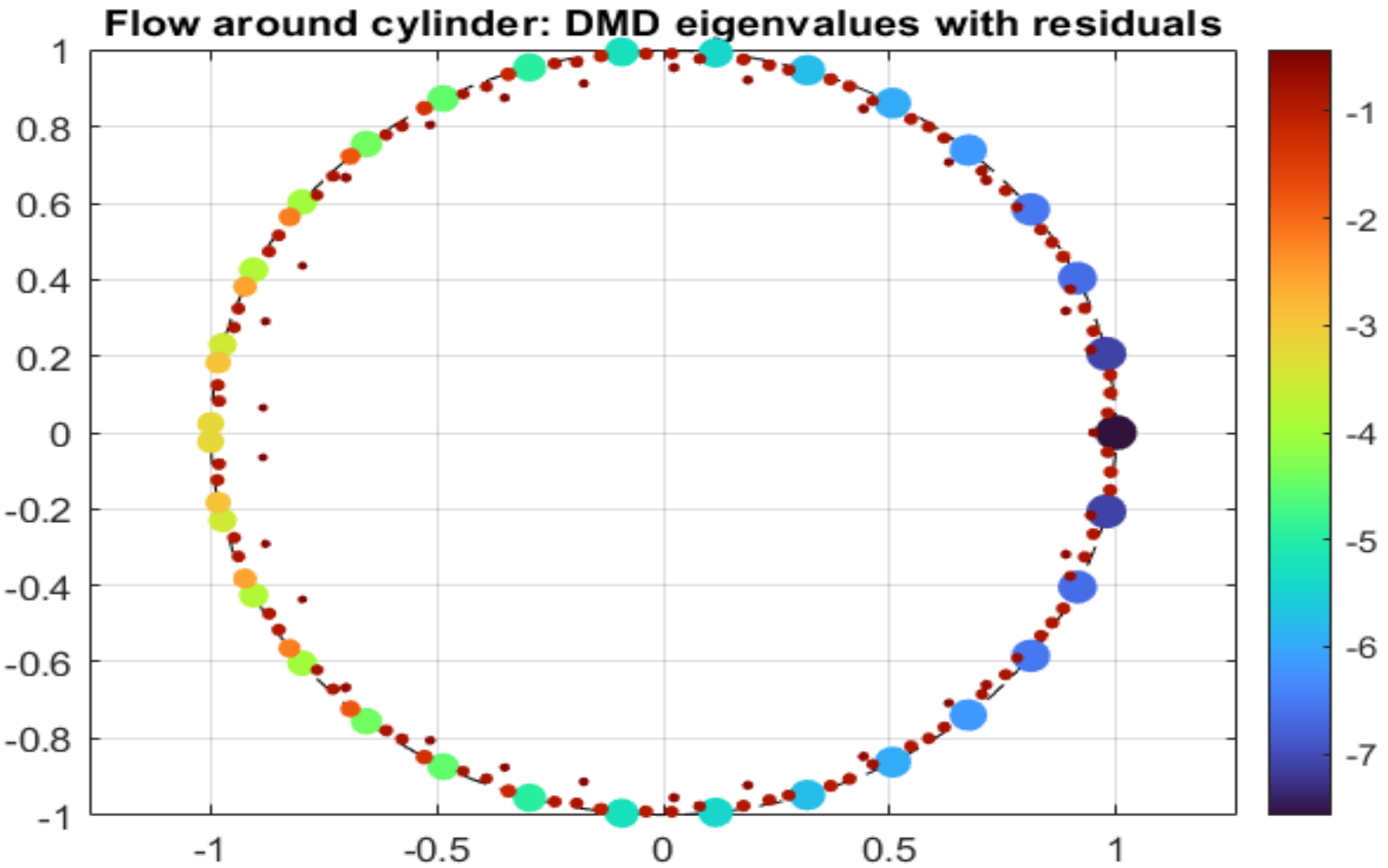
- Test the LS solution procedure and confirm that the data snapshots are reconstructed with small error. Use all computed Ritz pairs (λ_j, z_j), $j = 1, \ldots, m$.
- Examine the structure of the LS solution (the coefficients α_j).
- Use the residuals (of the Ritz pairs) and select a subset of the Ritz pairs for snapshots representations. Check the accuracy of such representations.
- The Ritz pairs are in general complex, and the snapshots are real. Examine how to ensure that the reconstruction remains real?

The residuals



Some Ritz pairs are not acceptable as approximate eigenpairs. How will those affect the spectral representation of the data?

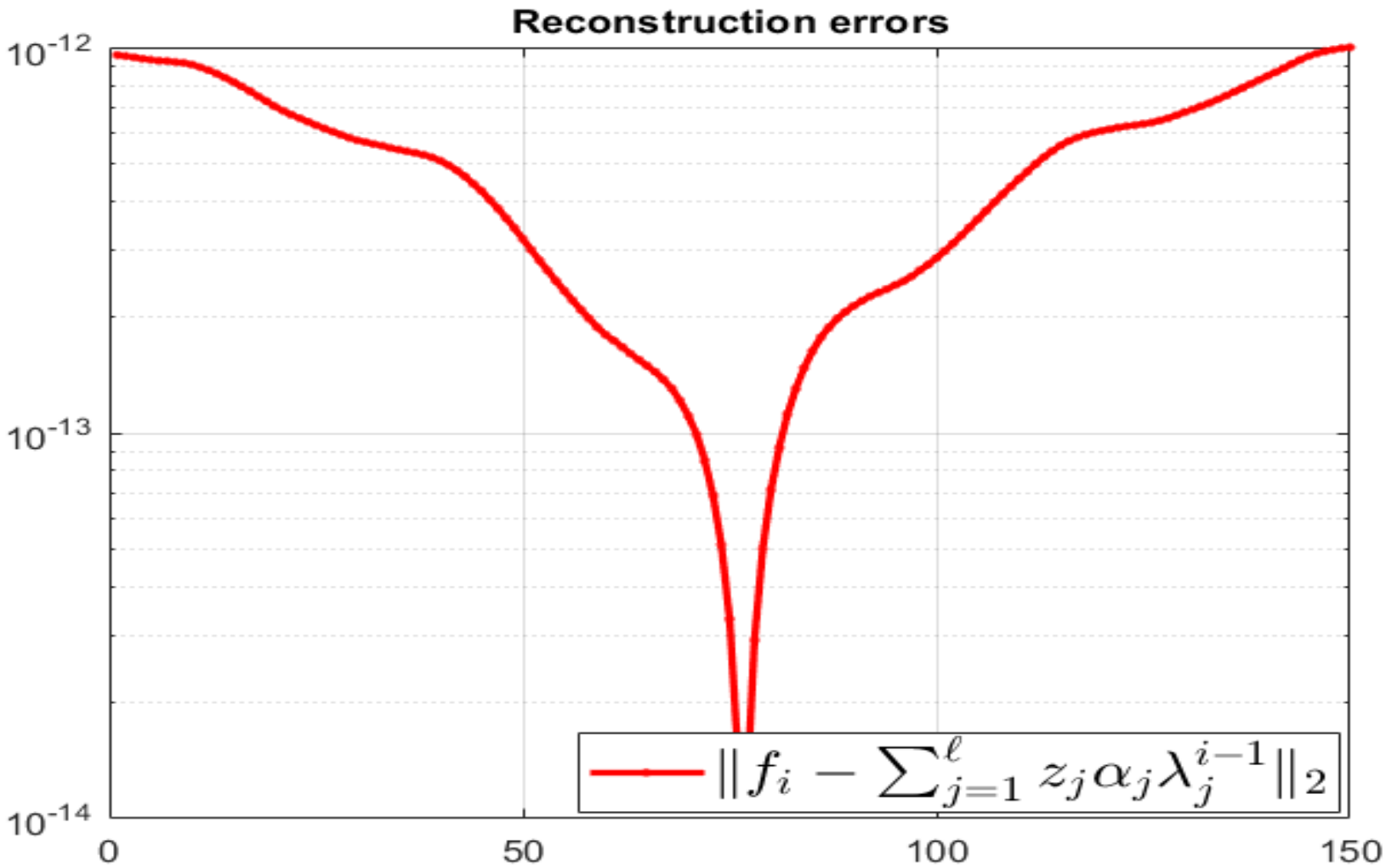
The Ritz values with residuals



(The snapshots are real.)

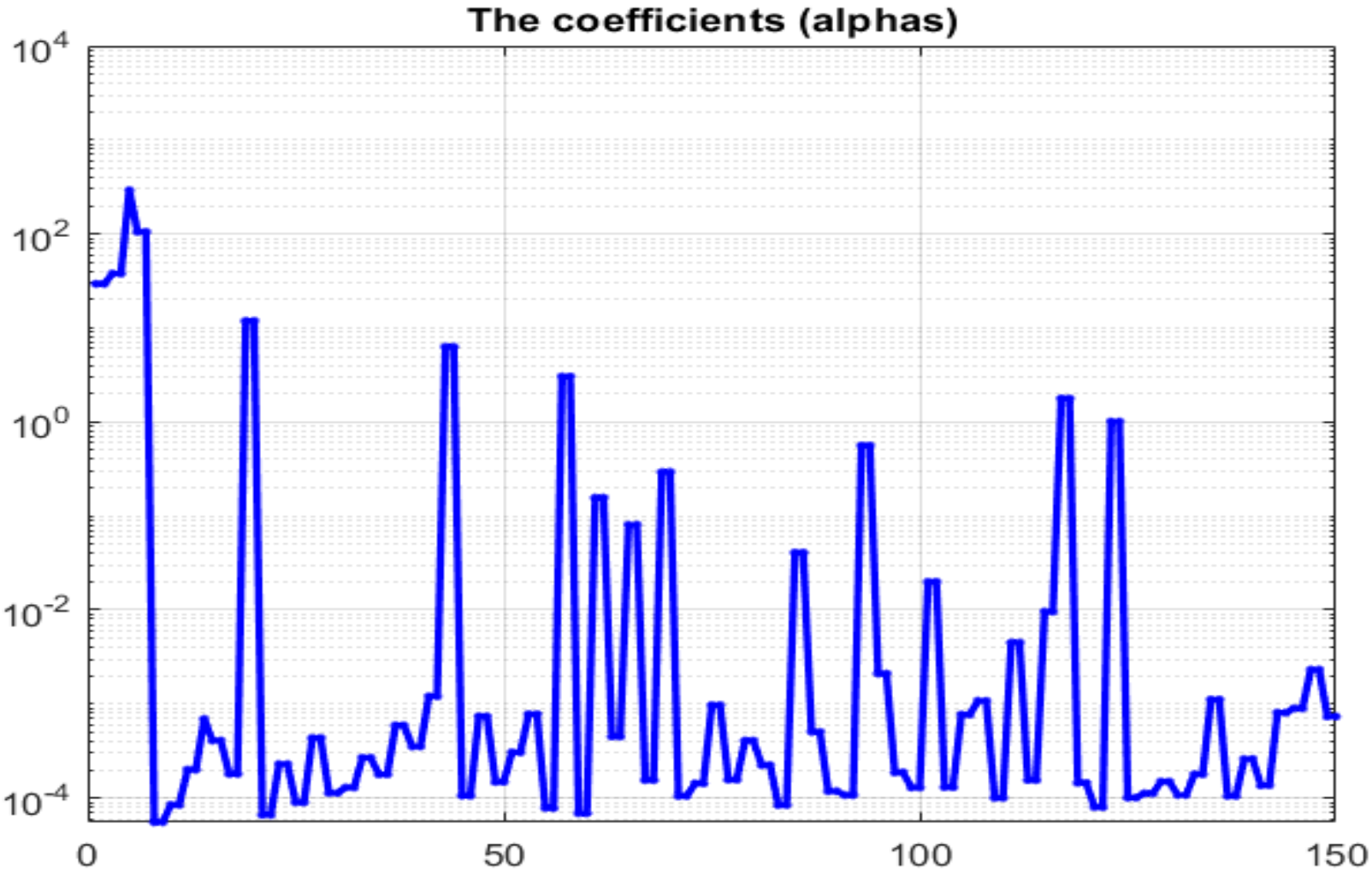
Example: Spectral snapshot reconstruction.

In the first experiment, we take all pairs (λ_j, z_j), $j = 1, \ldots, m = 150$. The reconstruction errors are:



Example: Spectral snapshot reconstruction.

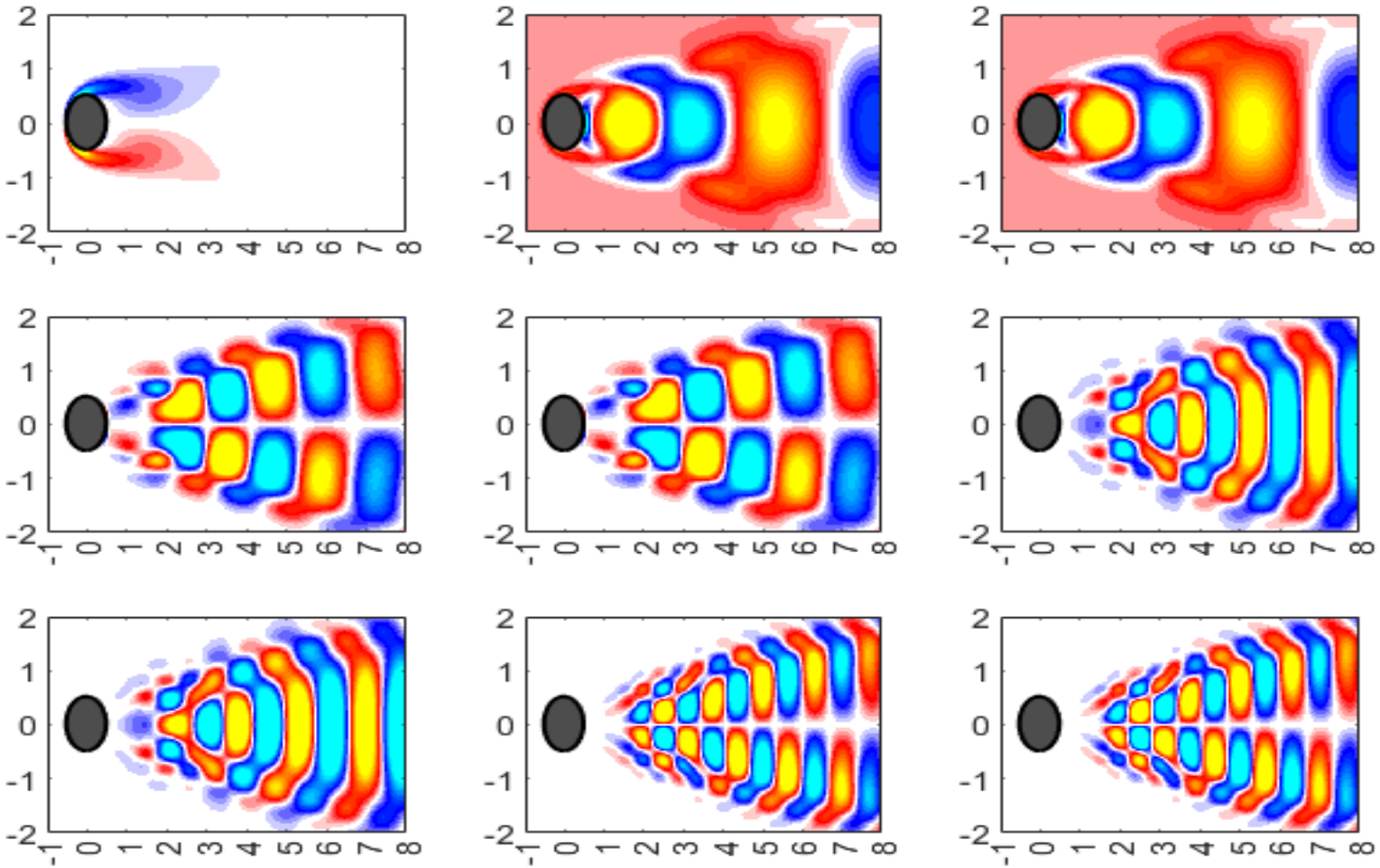
Now, look at the $|\alpha_i|$'s :



Discussion. Closed under complex conjugation (prove it!).

Example: Spectral snapshot reconstruction.

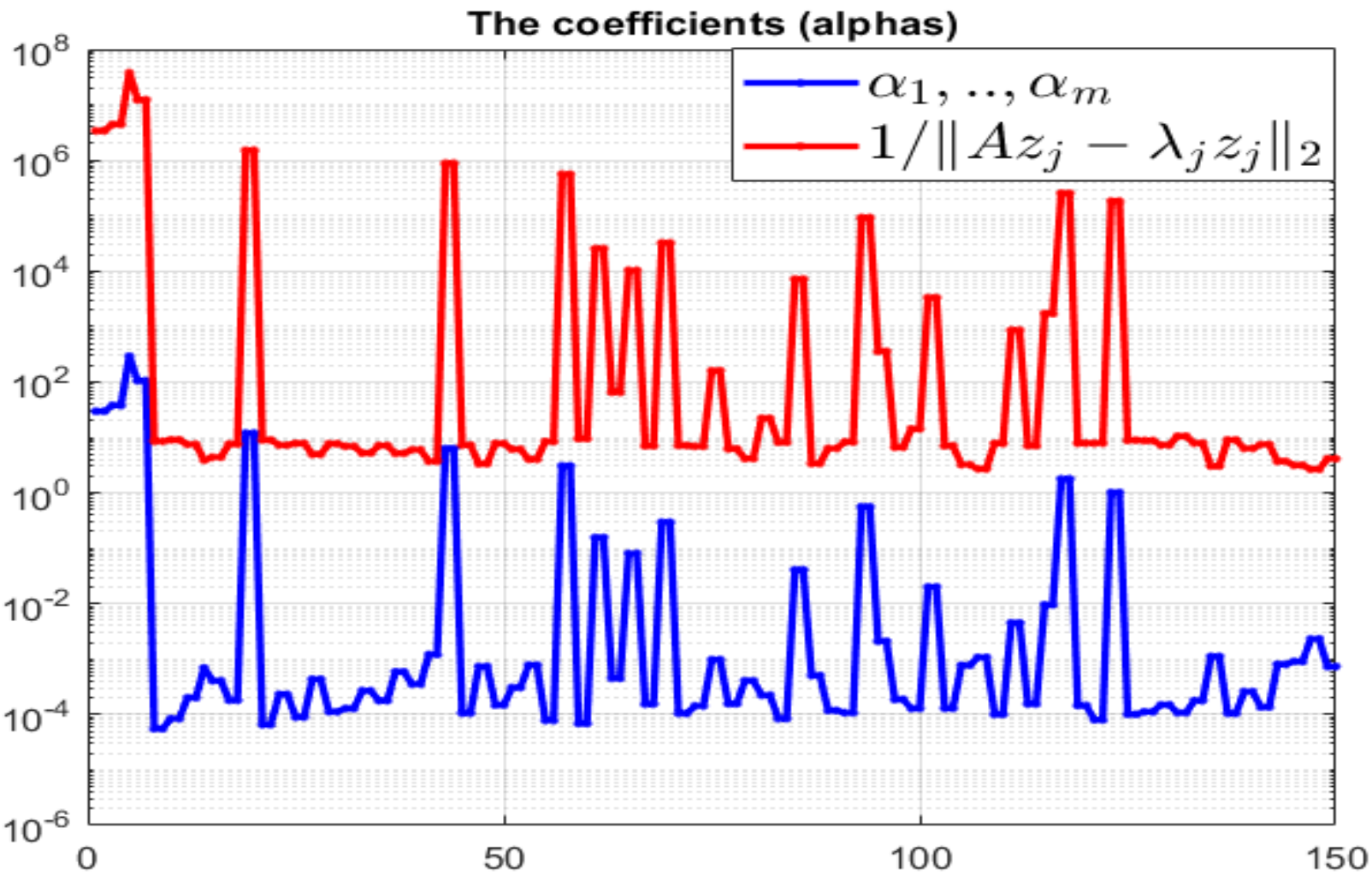
Real parts of modes with largest $|\alpha_j|$'s :



Discussion. Complex conjugate pairs etc.

Example: Spectral snapshot reconstruction.

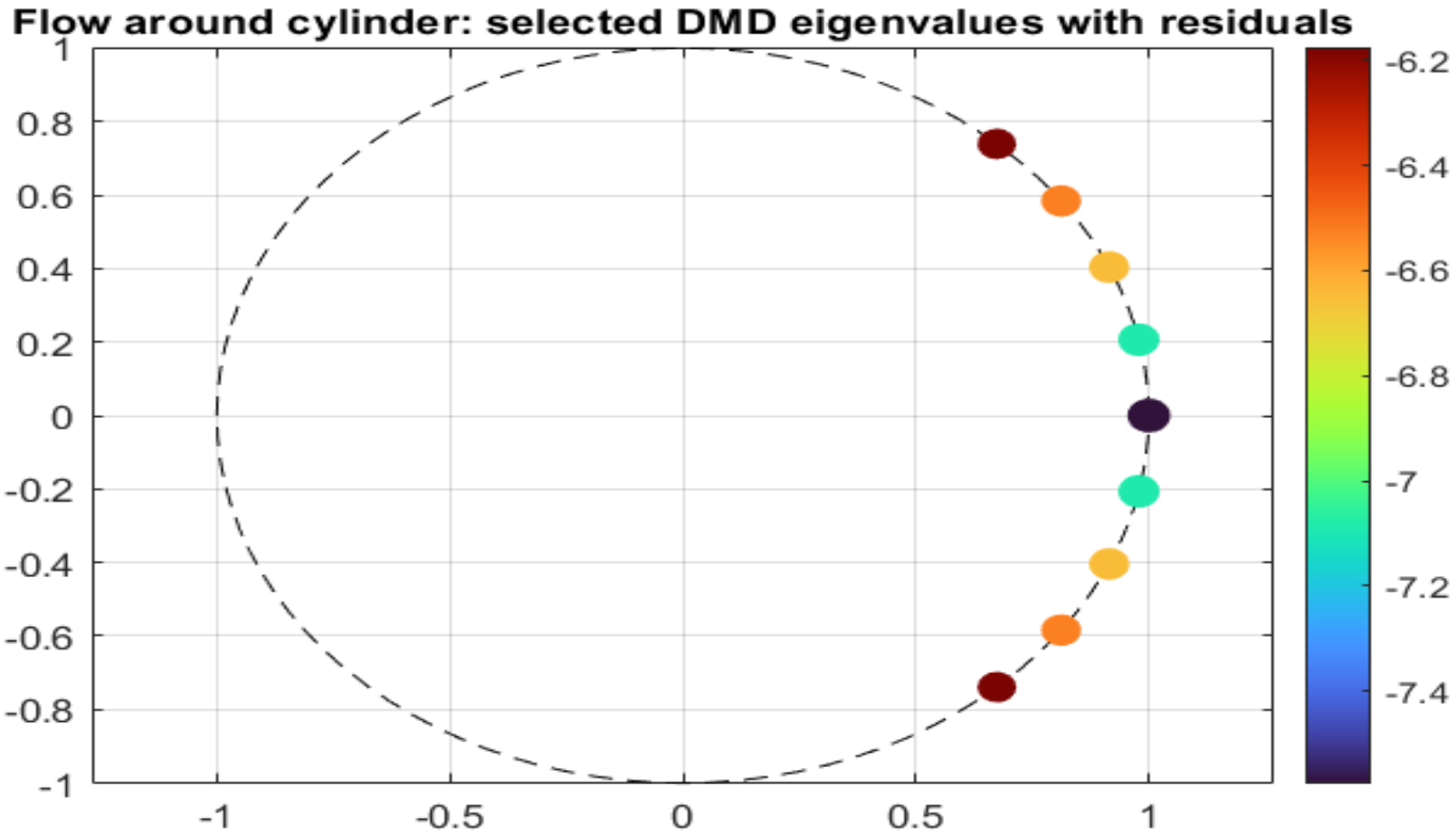
Now, look at the $|\alpha_j|$'s and the residuals $\|\mathbb{A}z_j - \lambda_j z_j\|_2$:



The residuals seem to indicate what modes are relevant.

Example: Spectral snapshot reconstruction.

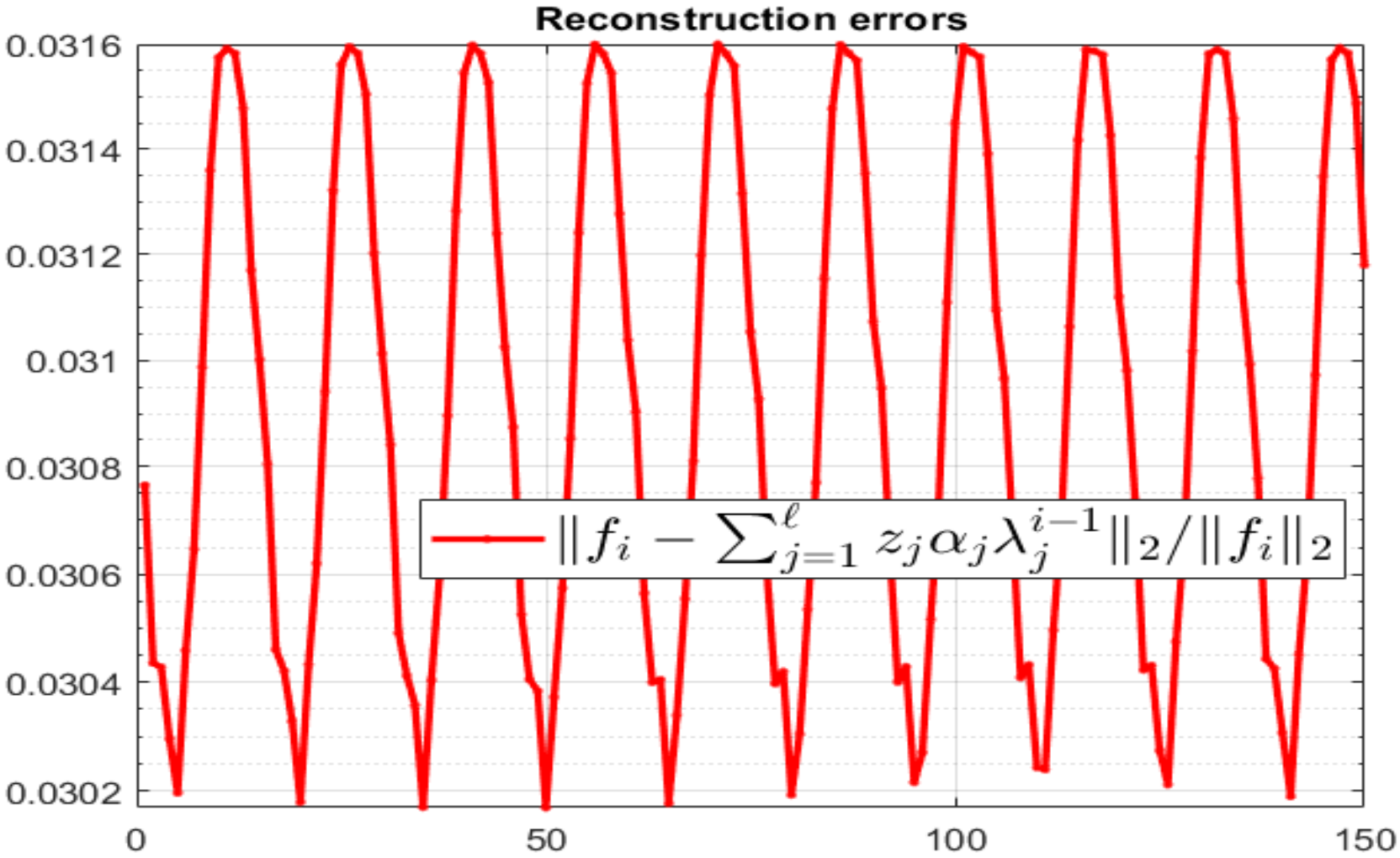
Now, look at the modes with the residuals $\|\mathbb{A}z_i - \lambda_i z_i\|_2 < 10^{-6}$:



Can these 9 modes represent all snapshots using only 9 coefficients $\alpha_1, \ldots, \alpha_9$?

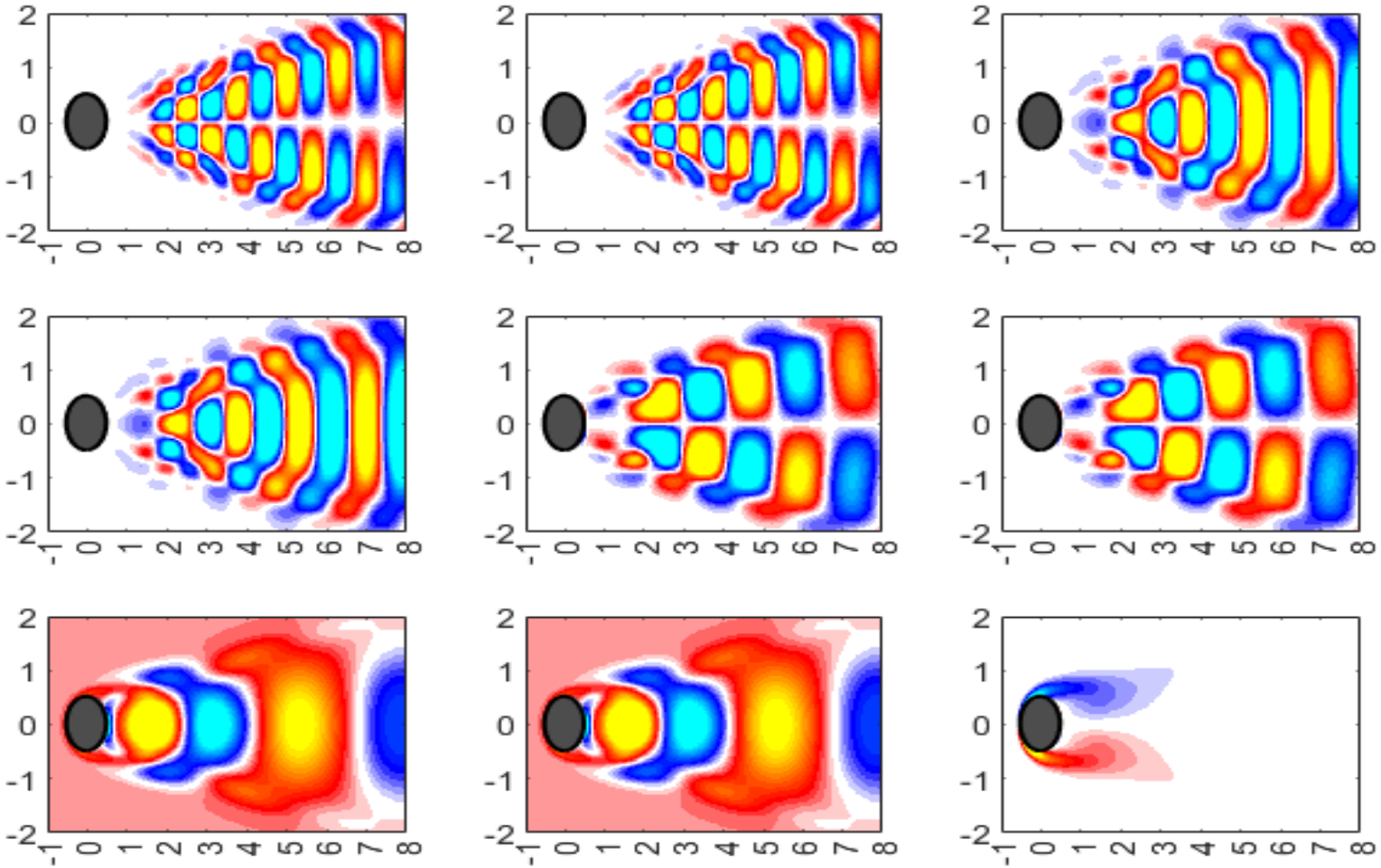
Example: Spectral snapshot reconstruction.

Reconstruction error when using the modes with residuals $\|\mathbb{A}z_j - \lambda_j z_j\|_2 < 10^{-6}$:



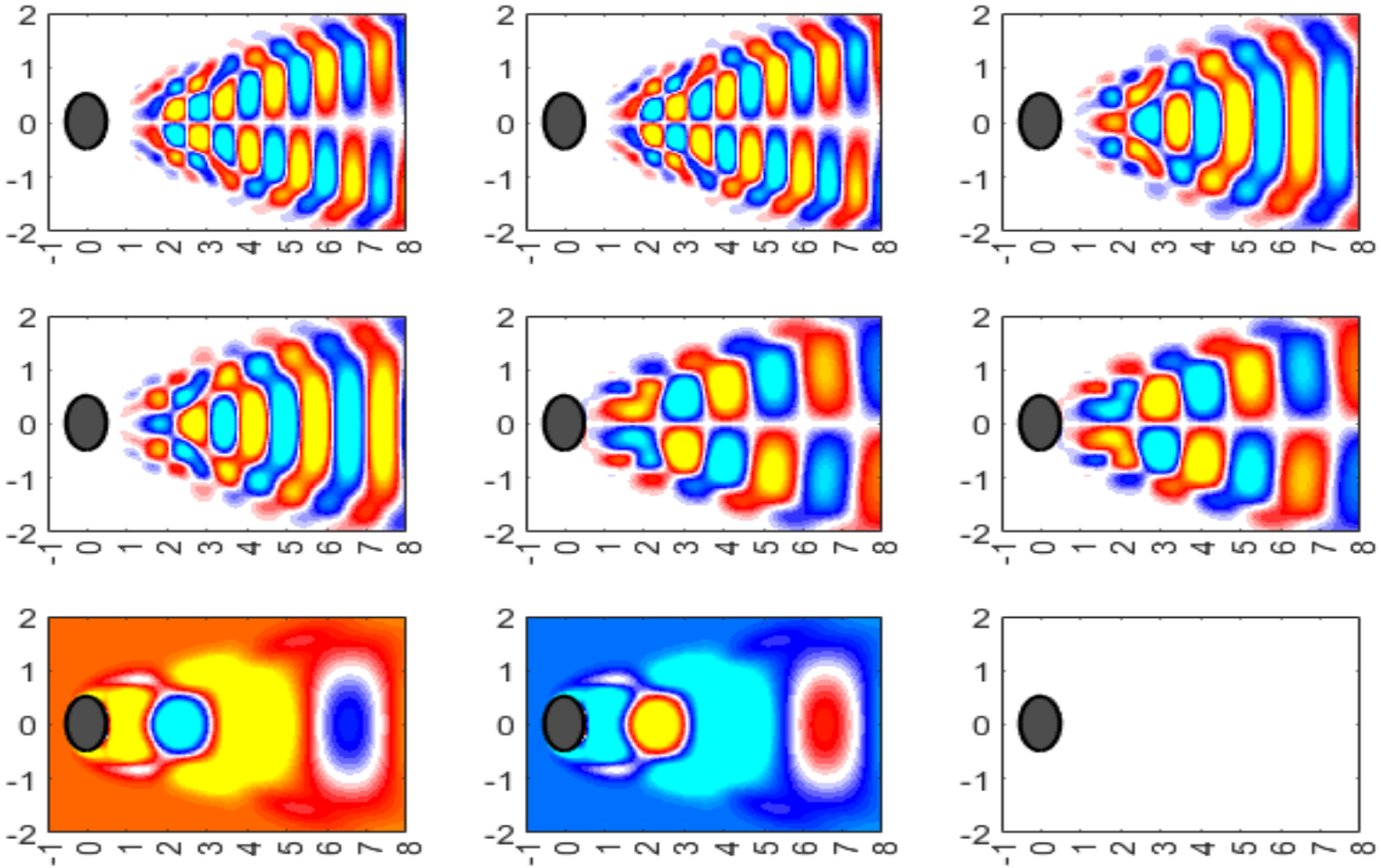
Example: Spectral snapshot reconstruction.

The real parts of the used modes (complex conjugate pairs):

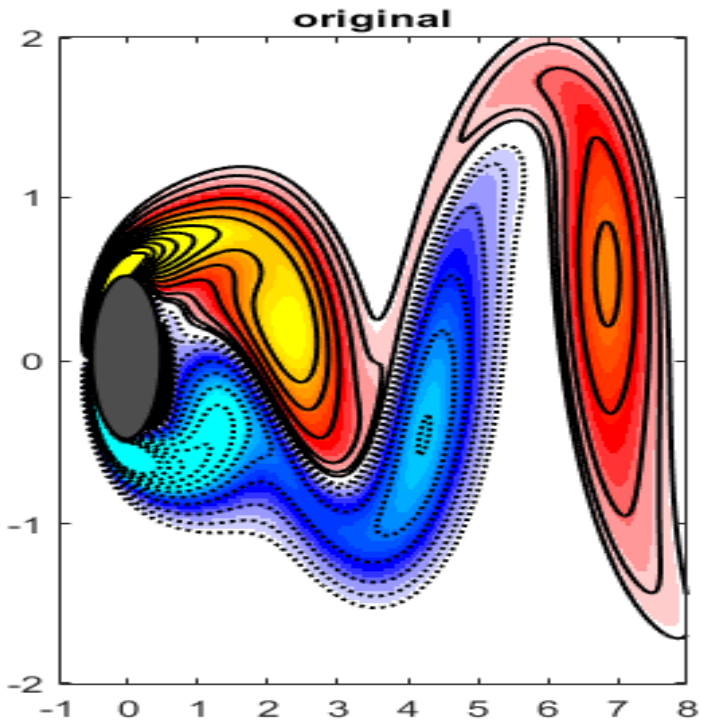


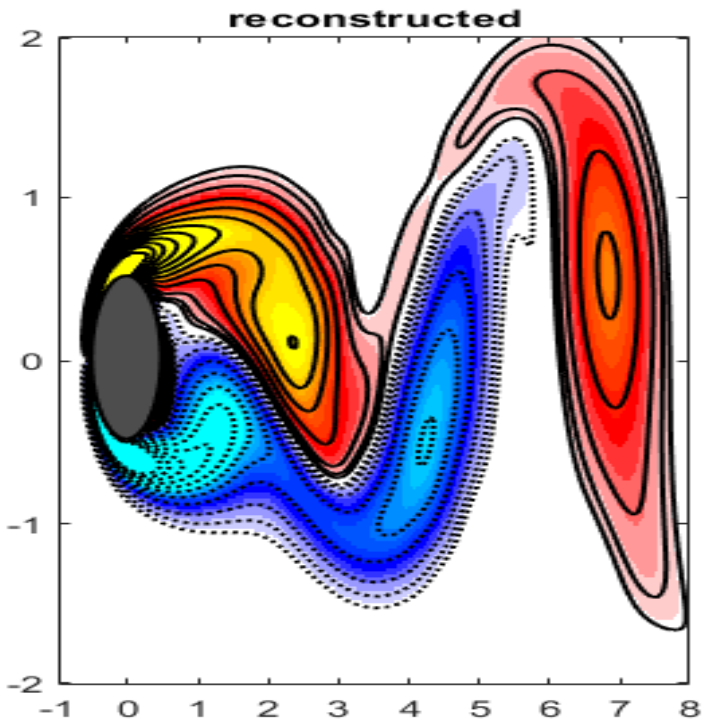
Example: Spectral snapshot reconstruction.

The imaginary parts of the used modes (complex conjugate pairs):

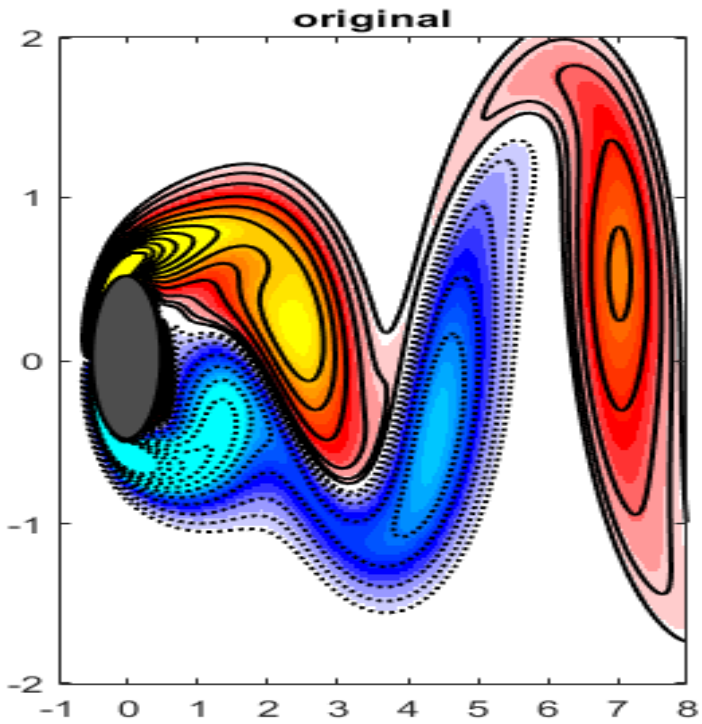


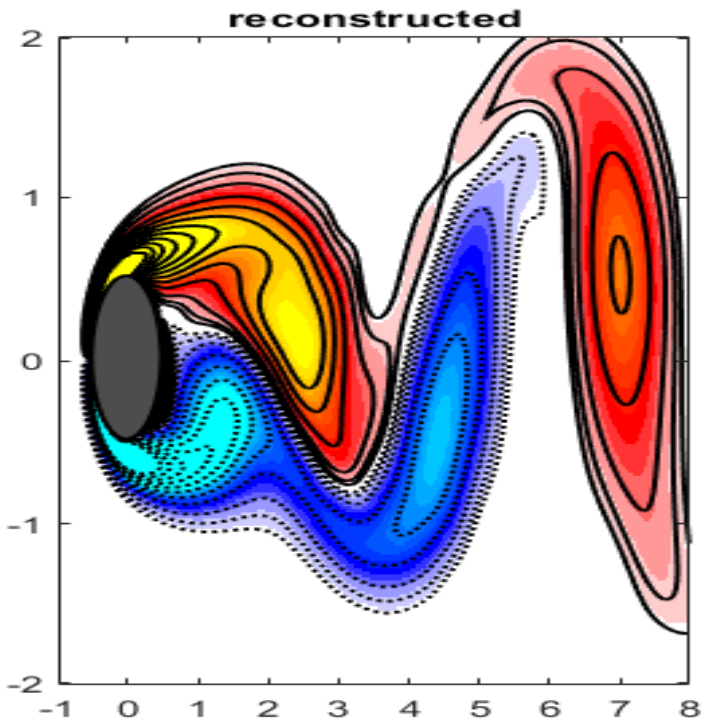
Example: Spectral snapshot reconstruction.



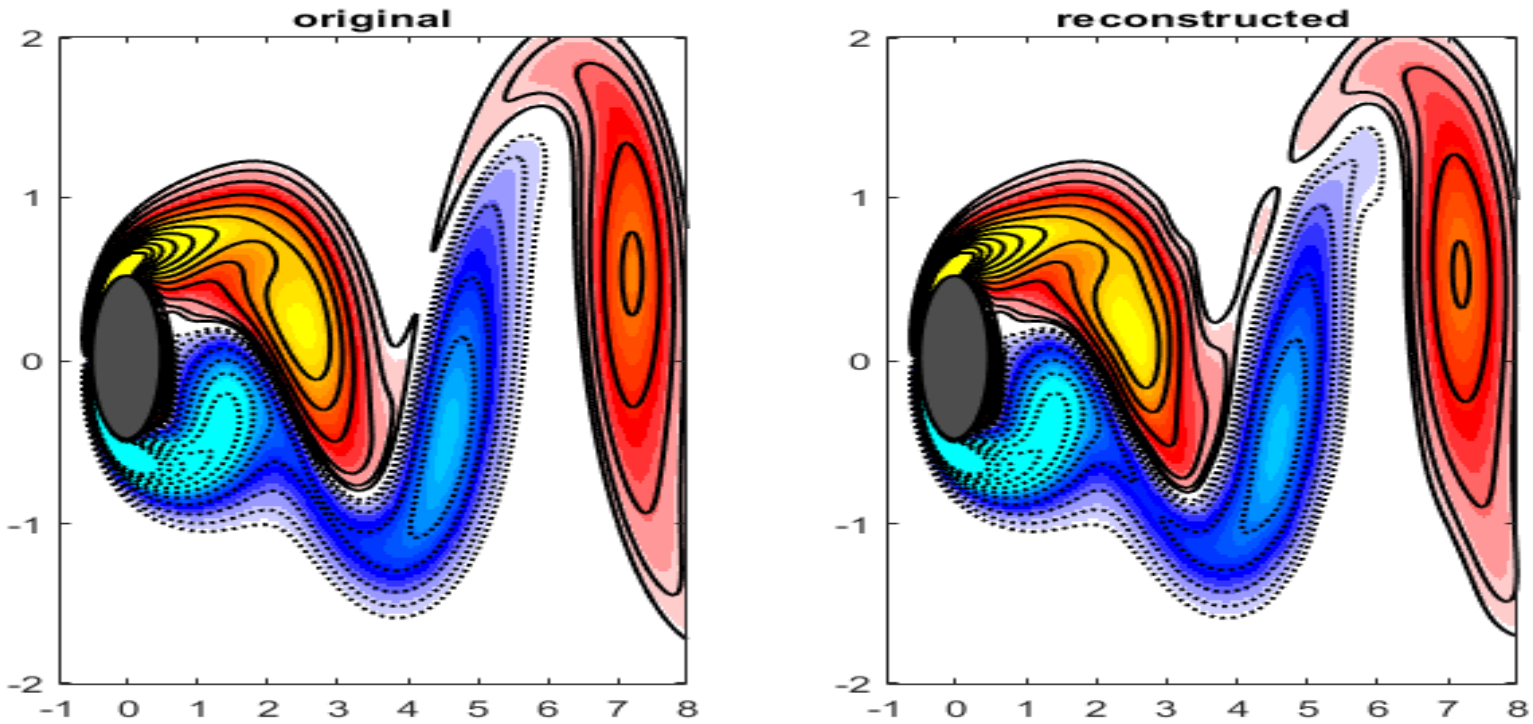


Example: Spectral snapshot reconstruction.

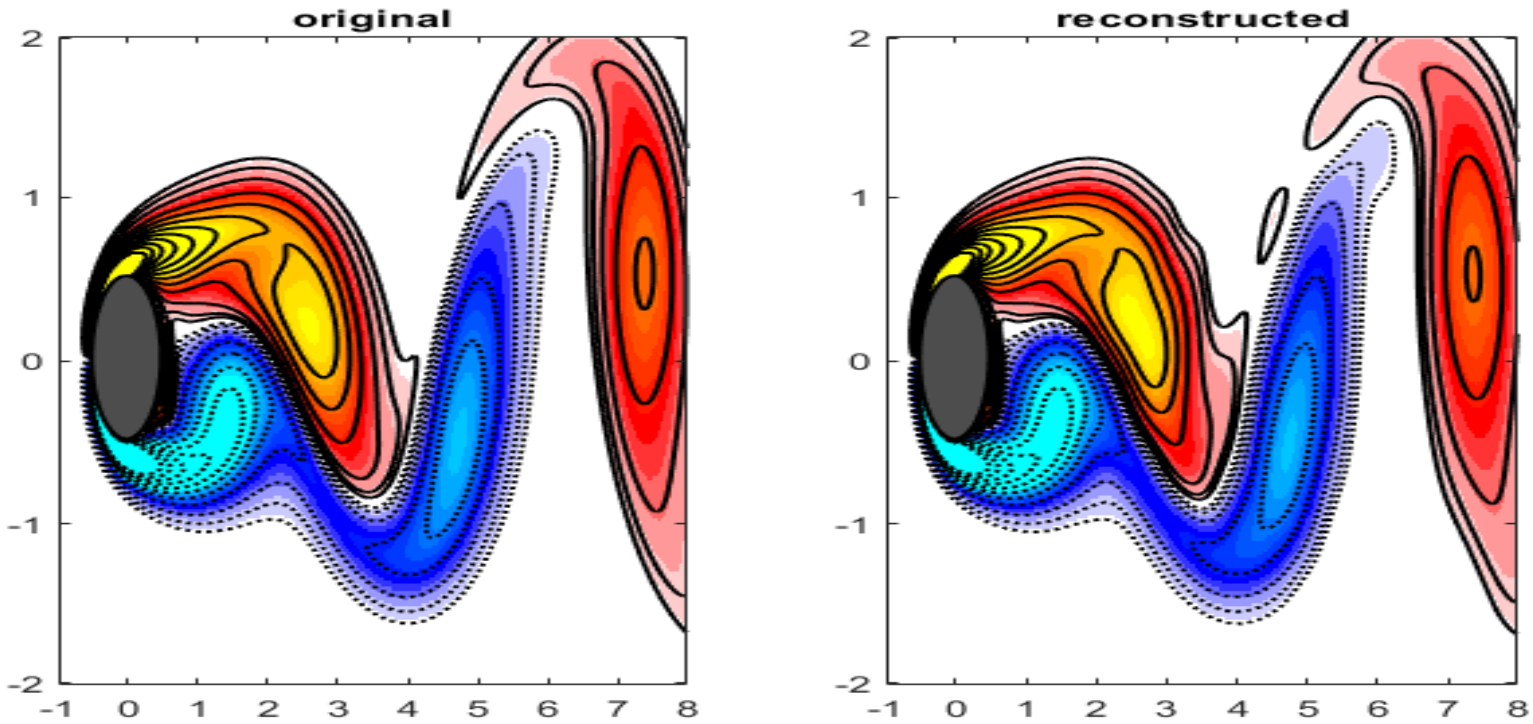




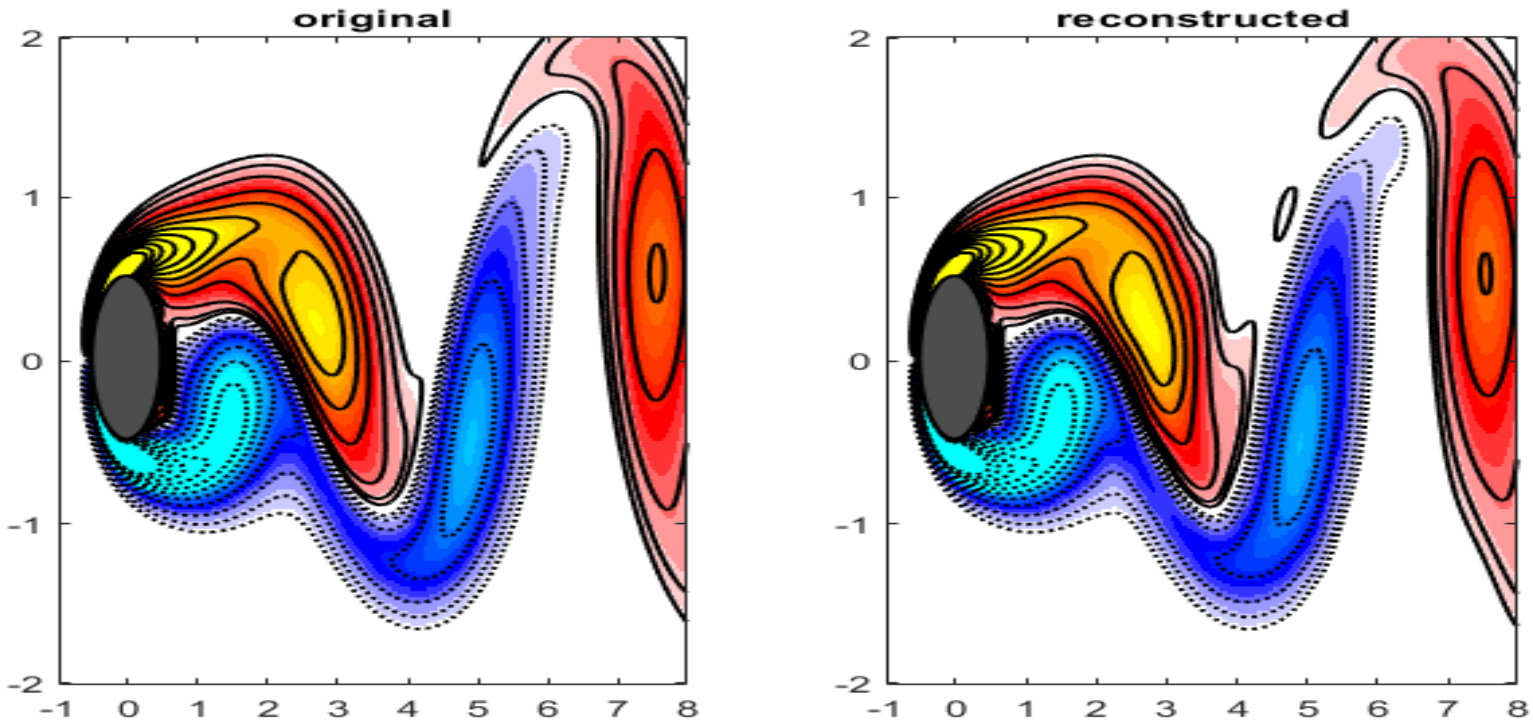
Example: Spectral snapshot reconstruction.



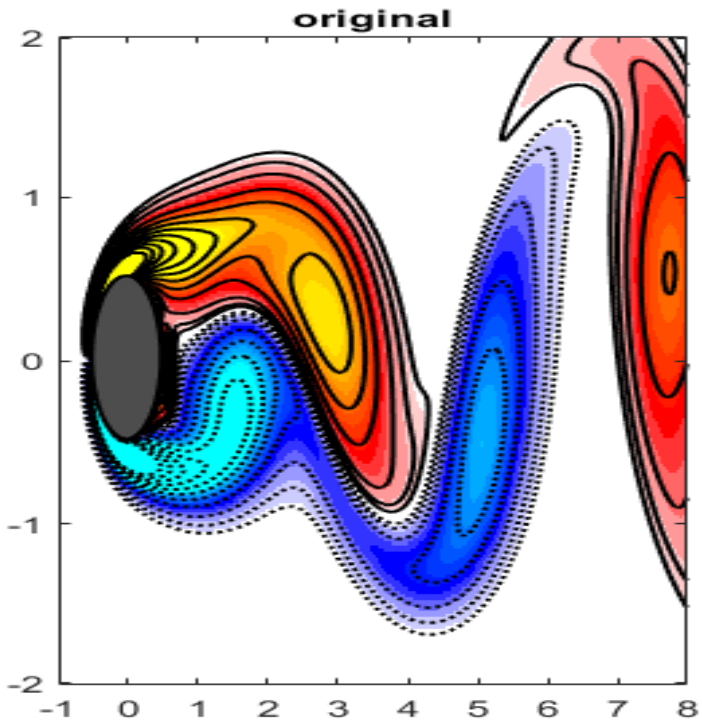
Example: Spectral snapshot reconstruction.

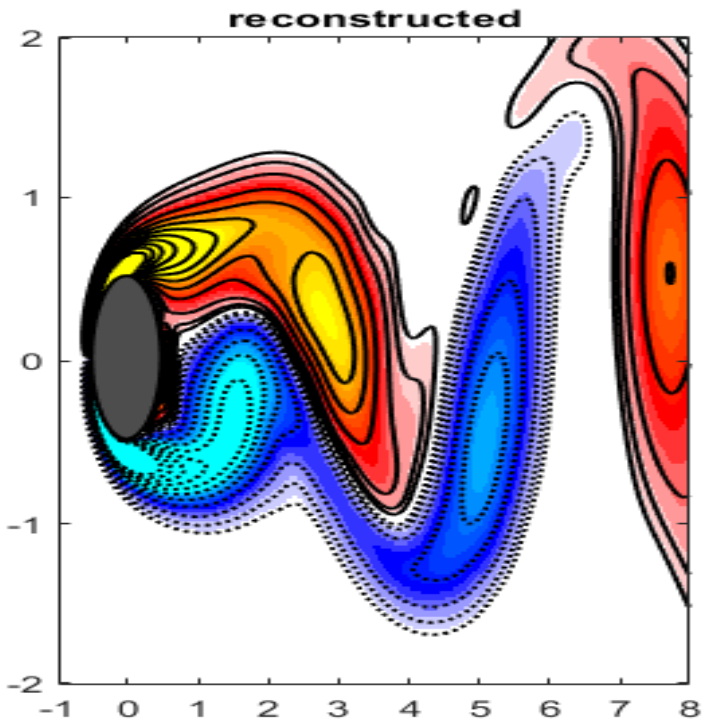


Example: Spectral snapshot reconstruction.

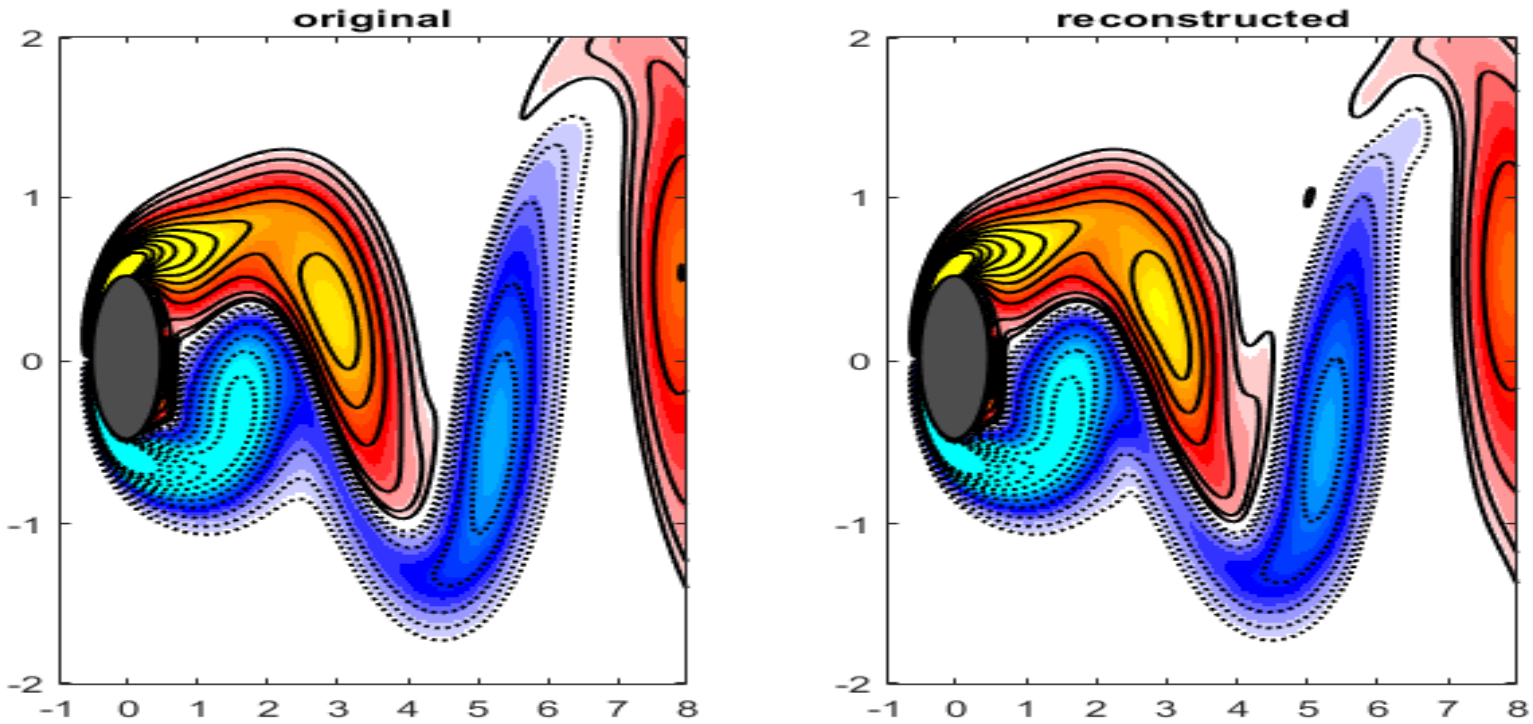


Example: Spectral snapshot reconstruction.

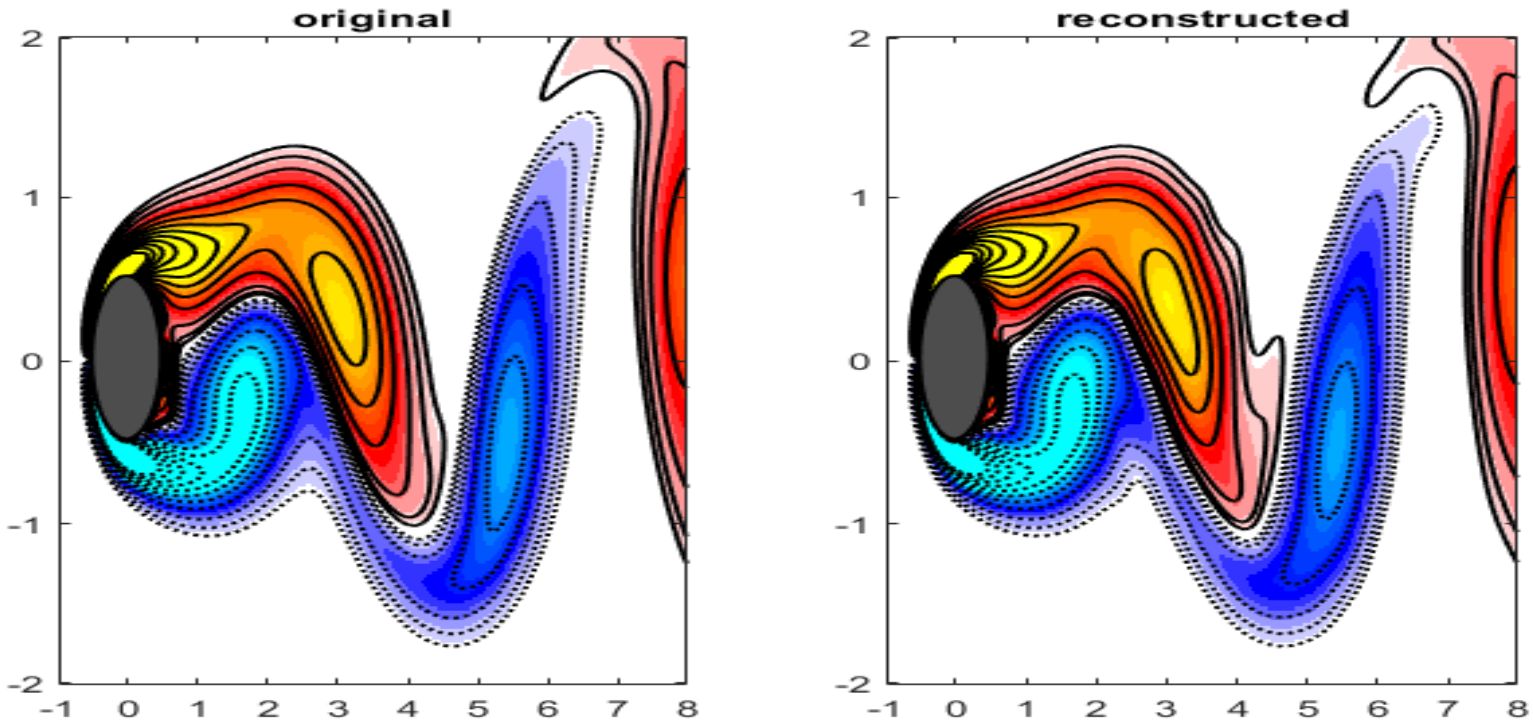




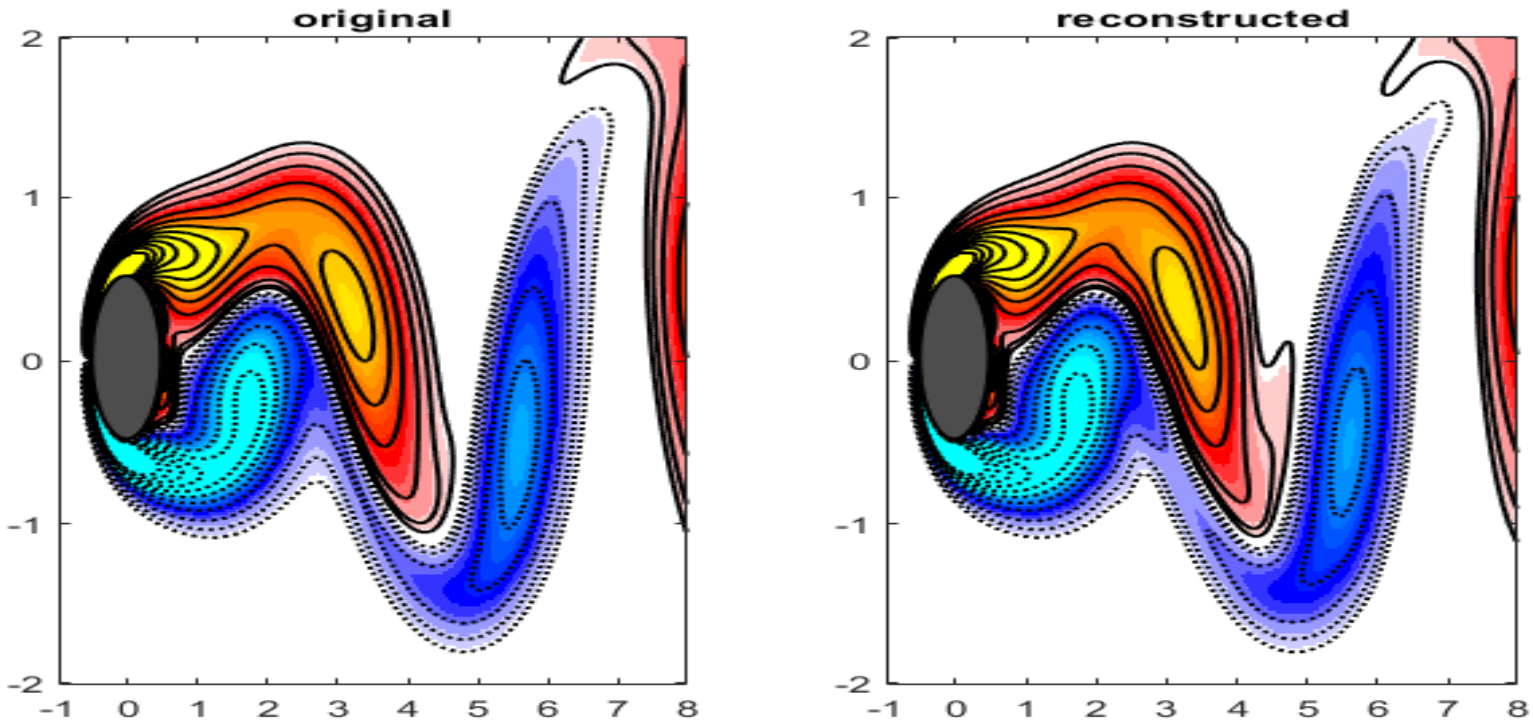
Example: Spectral snapshot reconstruction.



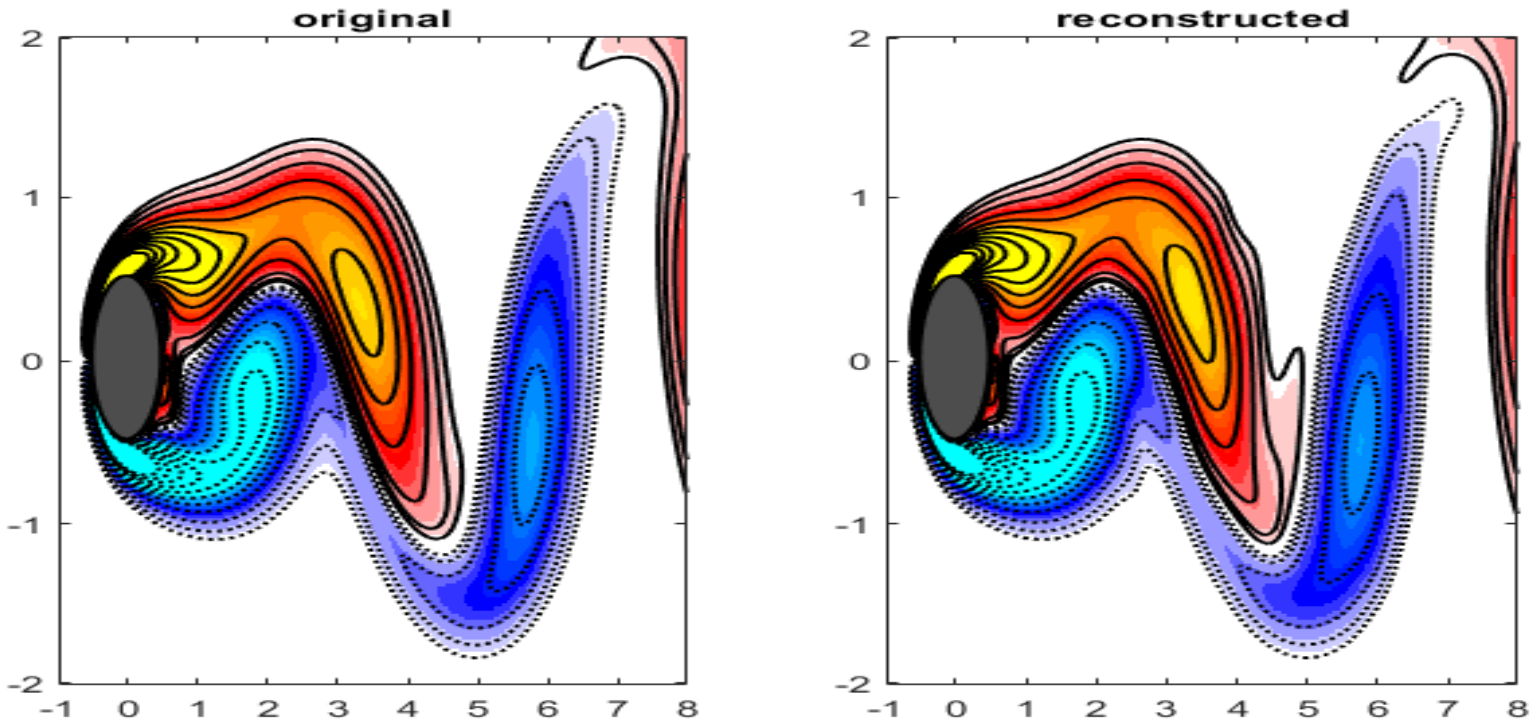
Example: Spectral snapshot reconstruction.



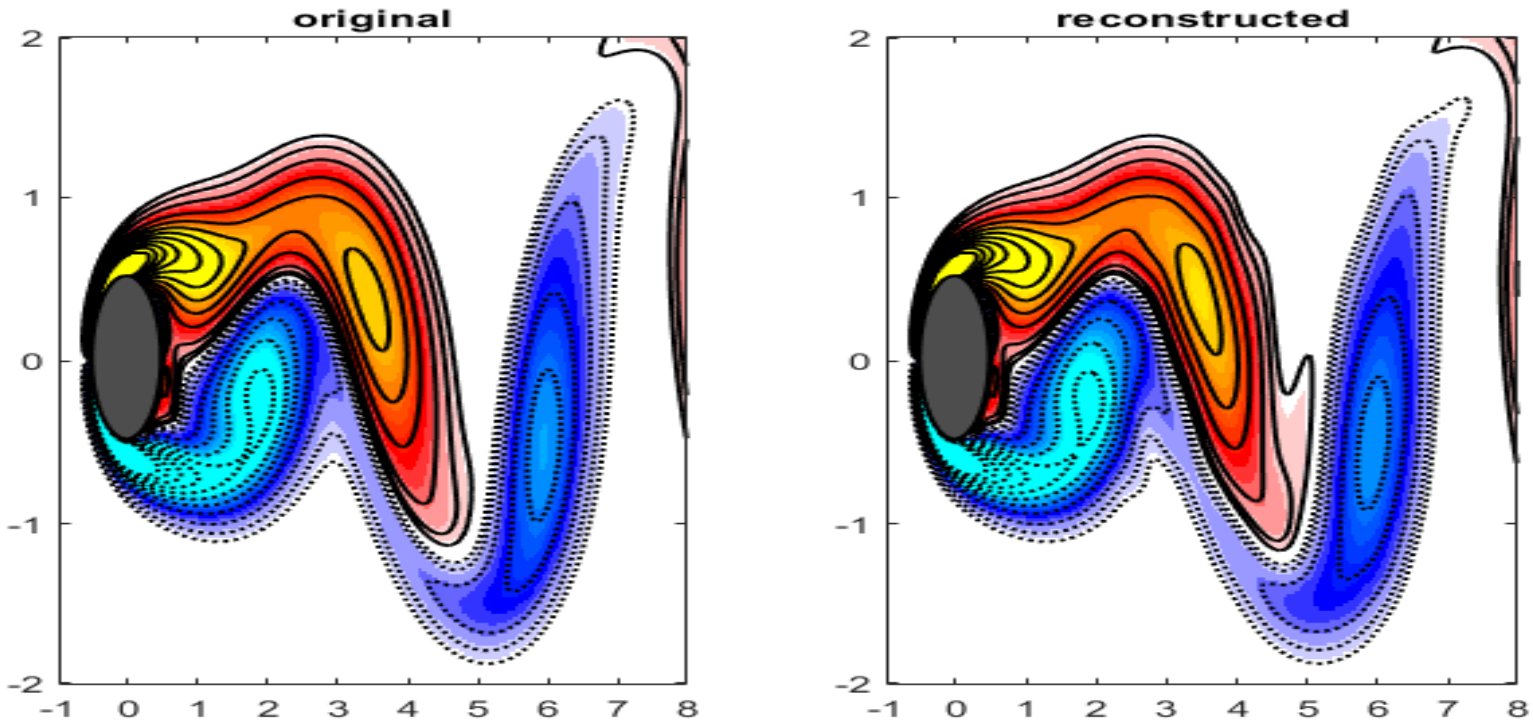
Example: Spectral snapshot reconstruction.



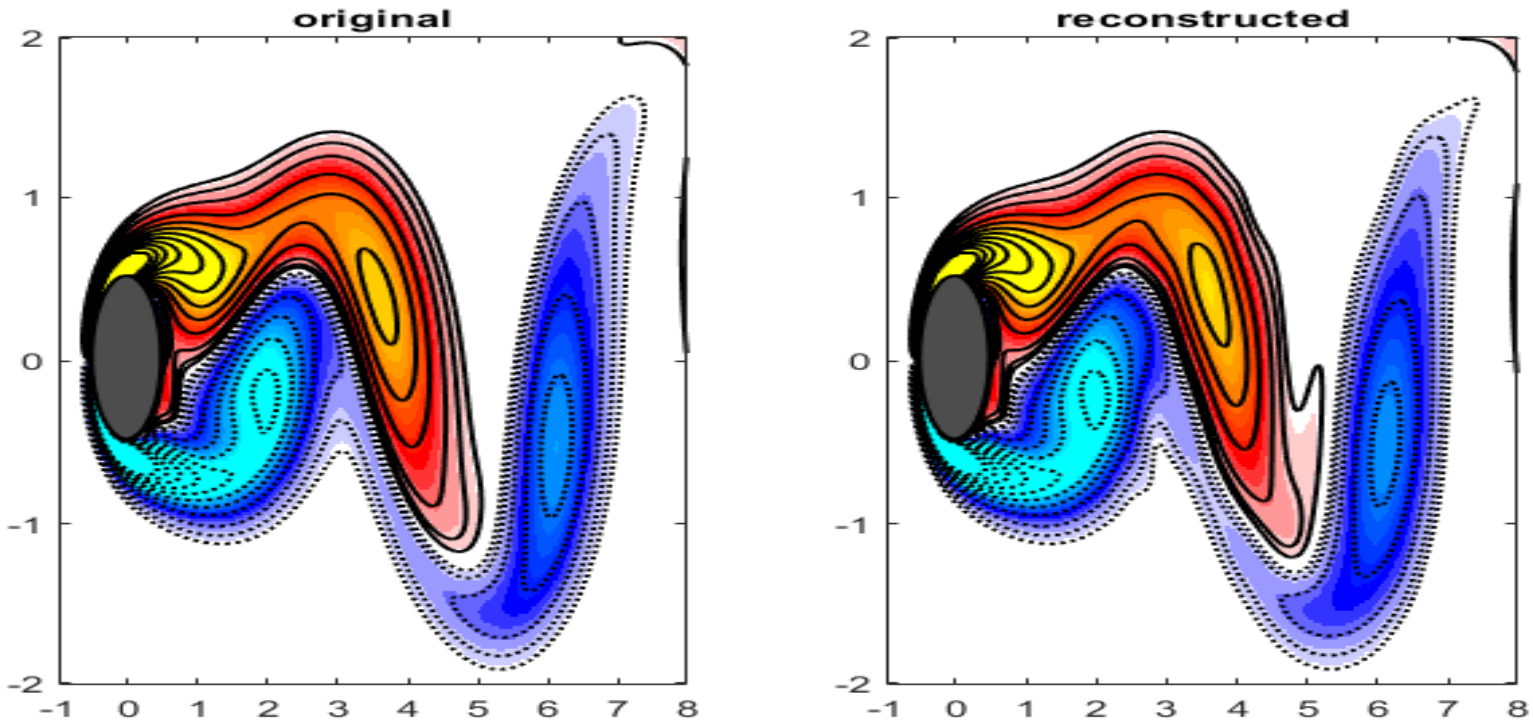
Example: Spectral snapshot reconstruction.



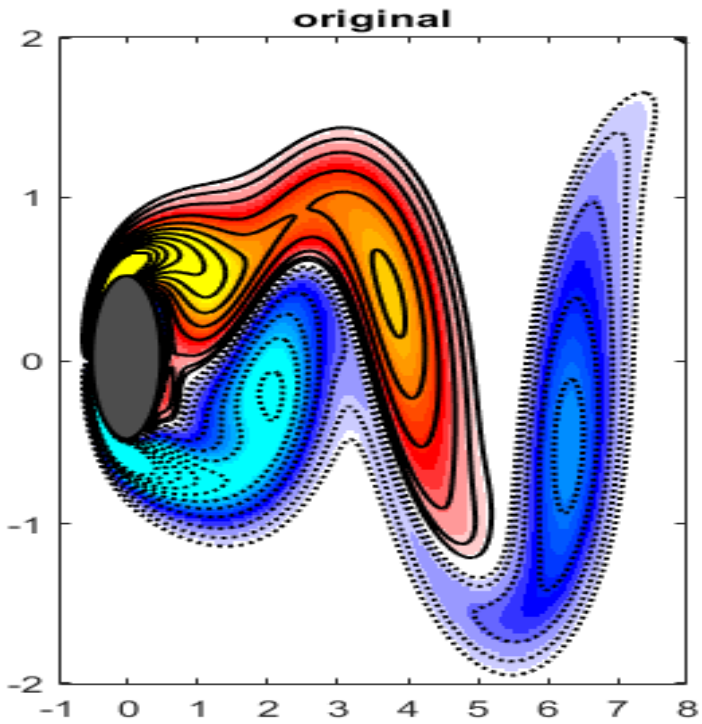
Example: Spectral snapshot reconstruction.

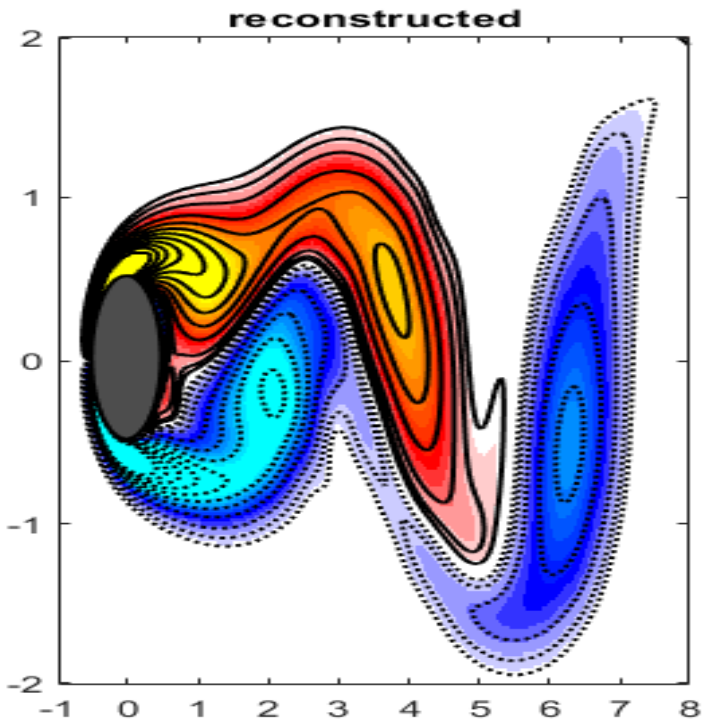


Example: Spectral snapshot reconstruction.

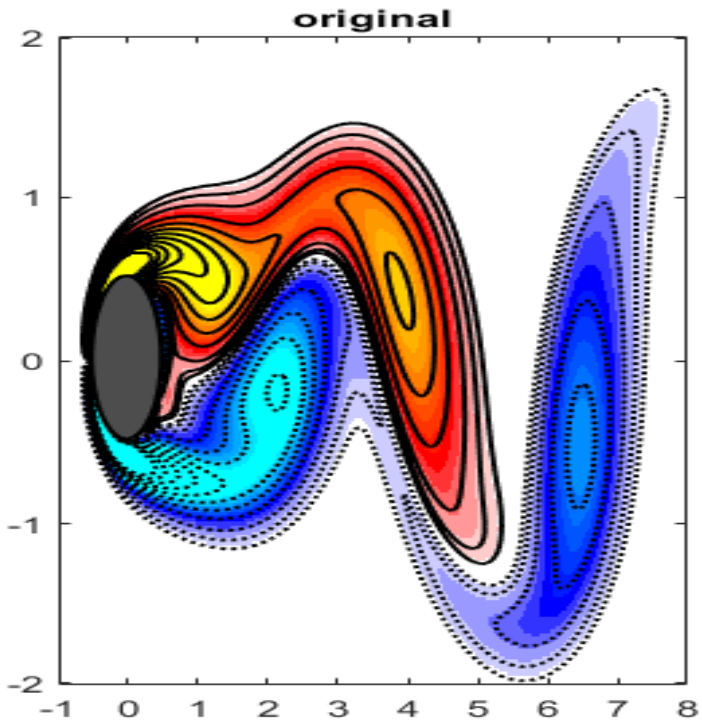


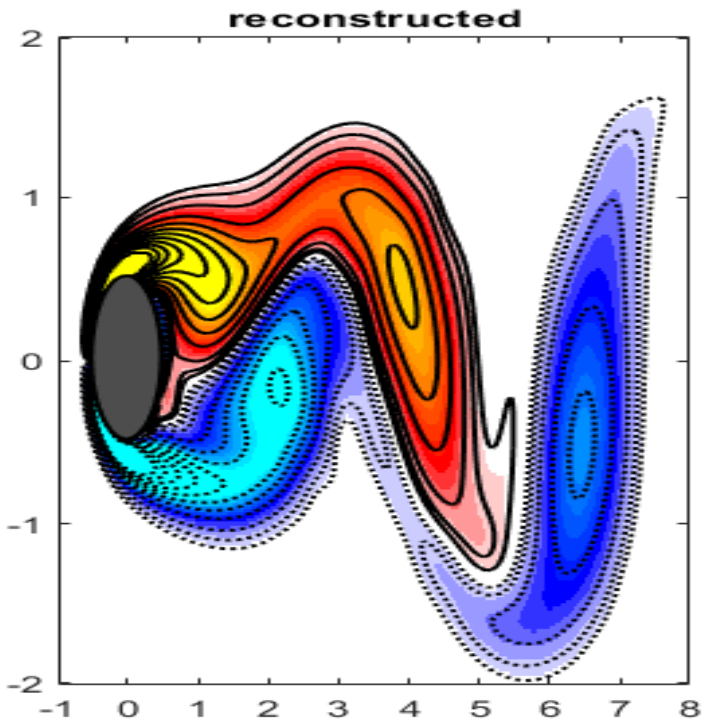
Example: Spectral snapshot reconstruction.



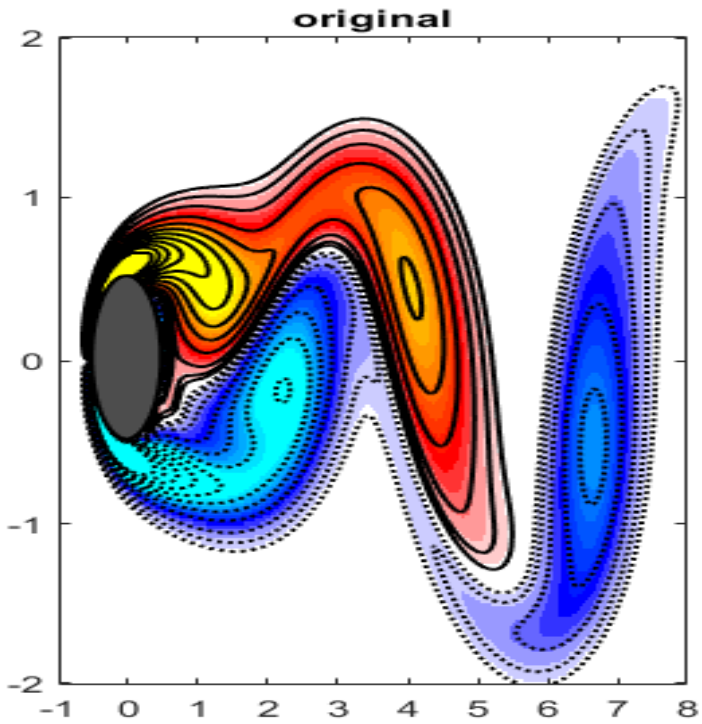


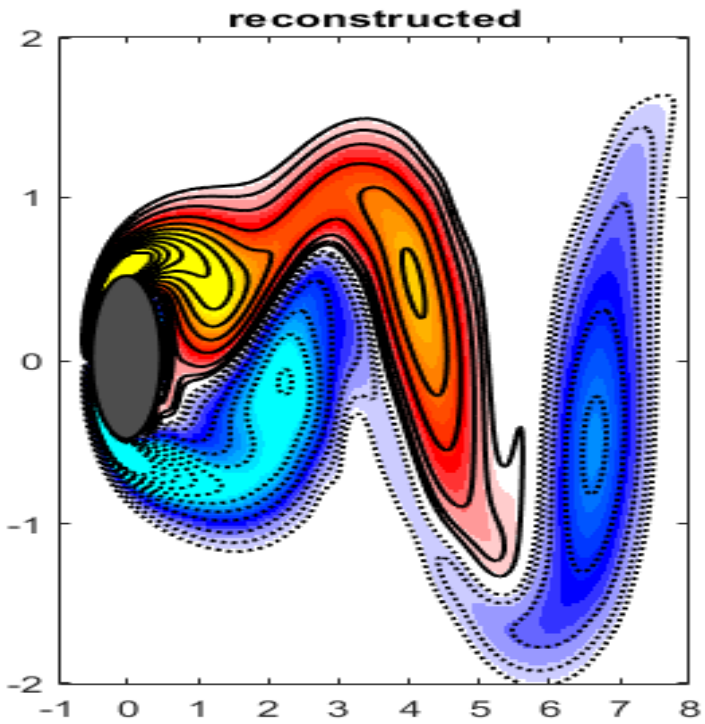
Example: Spectral snapshot reconstruction.



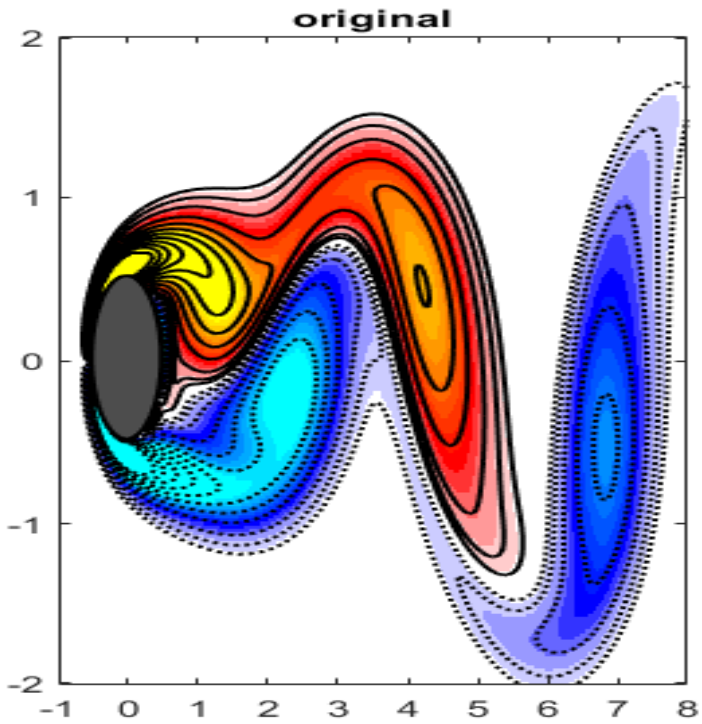


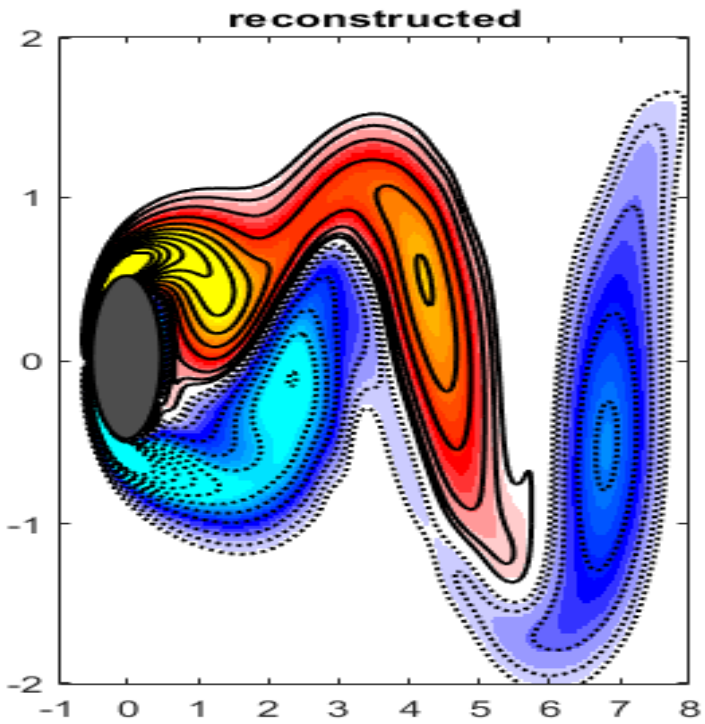
Example: Spectral snapshot reconstruction.



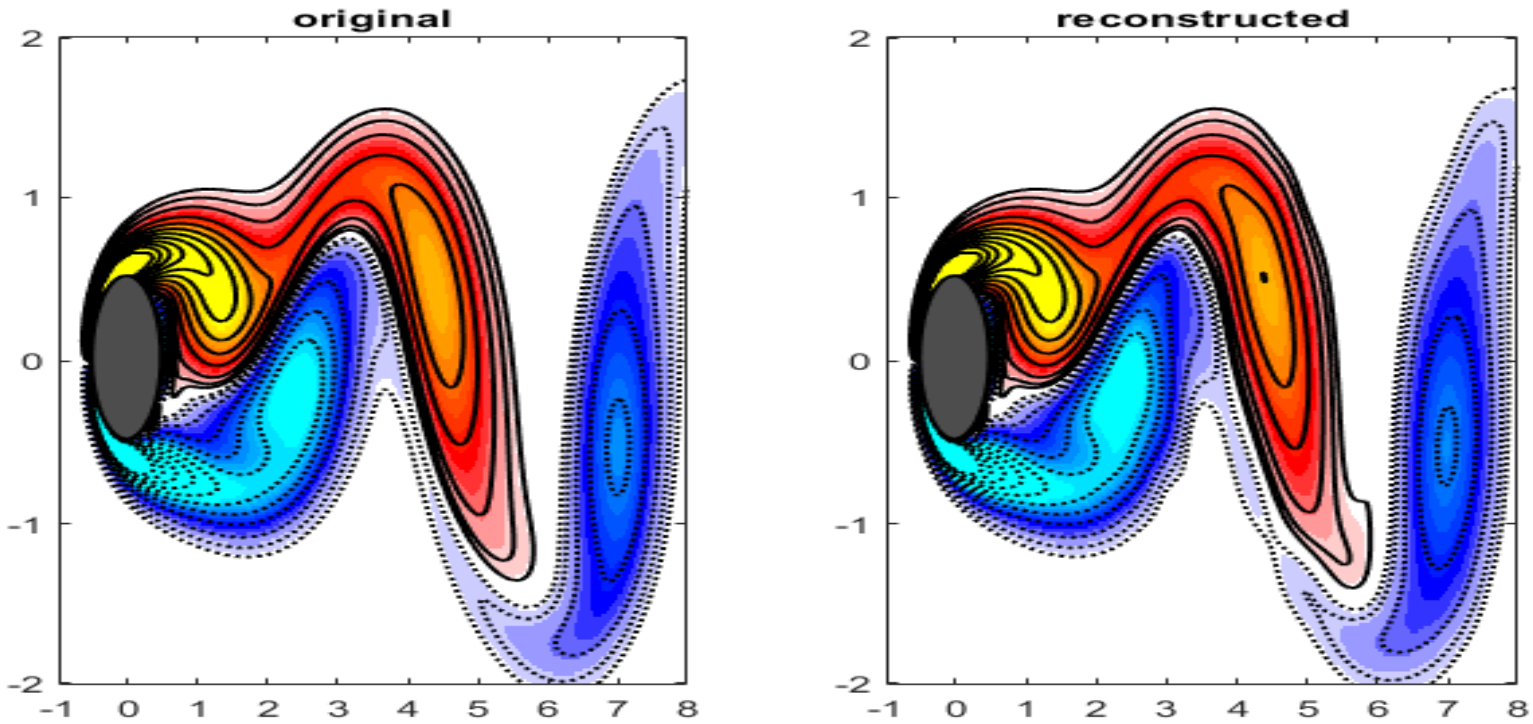


Example: Spectral snapshot reconstruction.

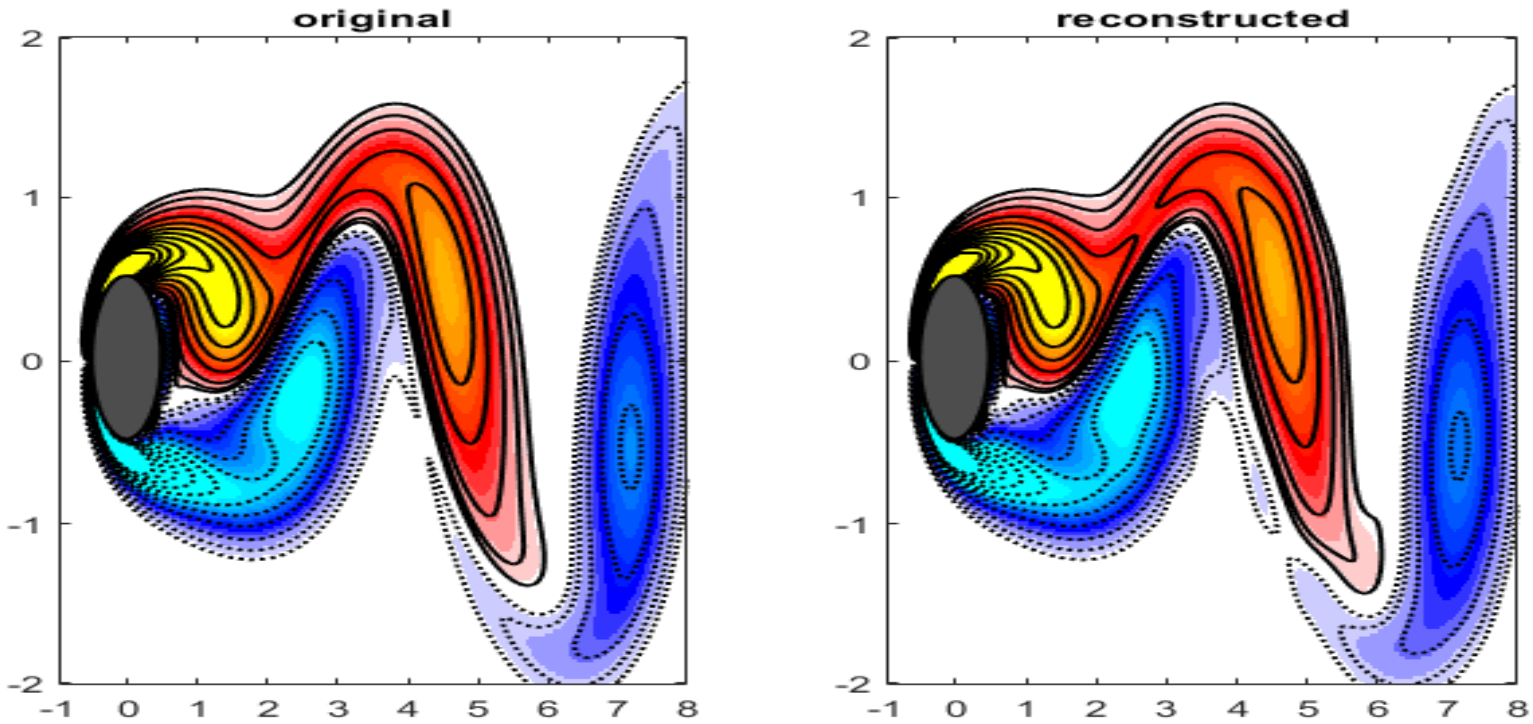




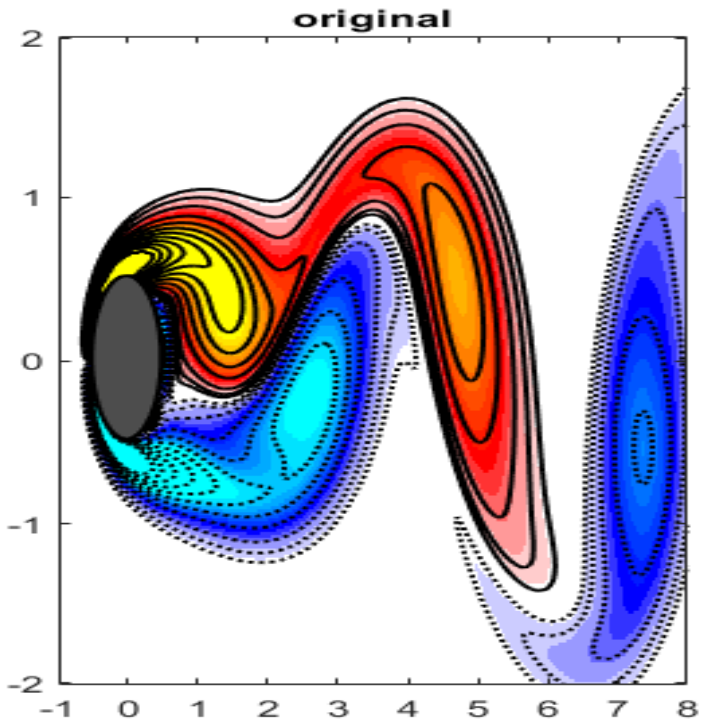
Example: Spectral snapshot reconstruction.

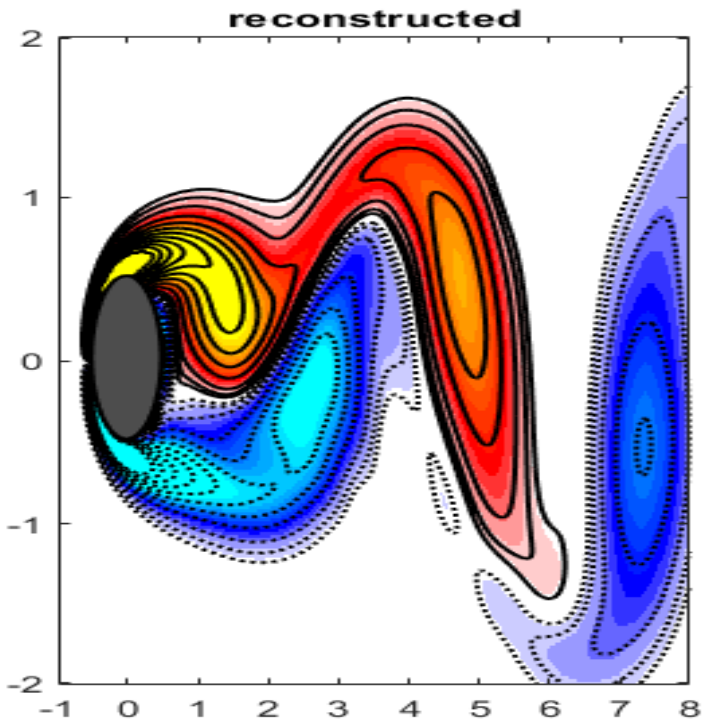


Example: Spectral snapshot reconstruction.

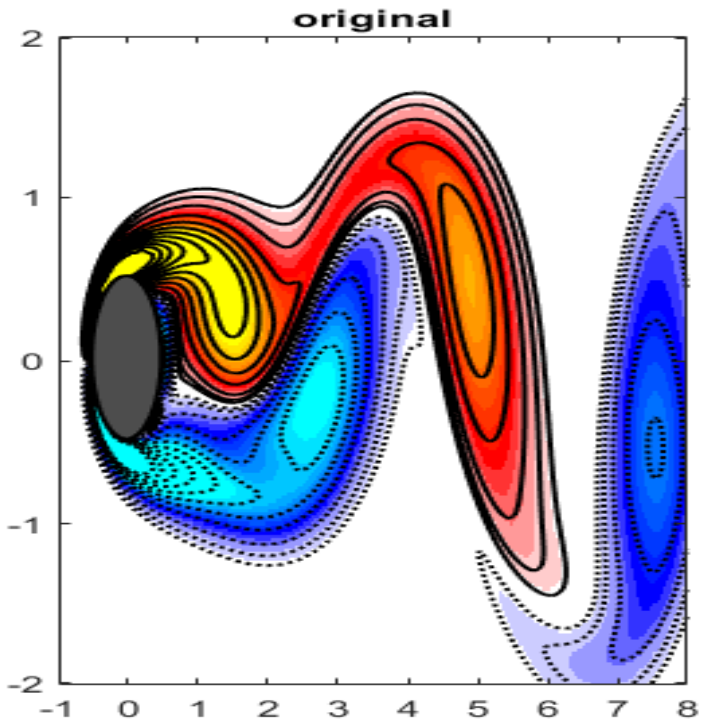


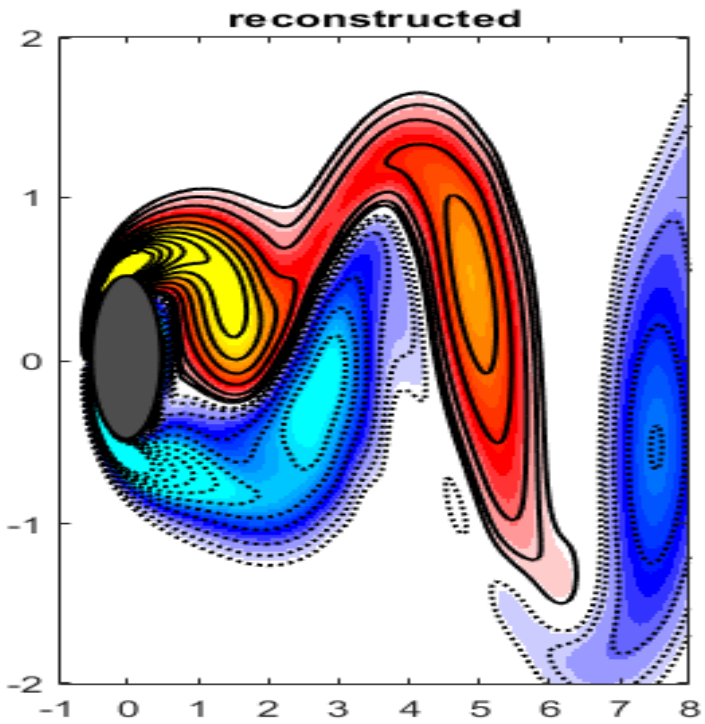
Example: Spectral snapshot reconstruction.





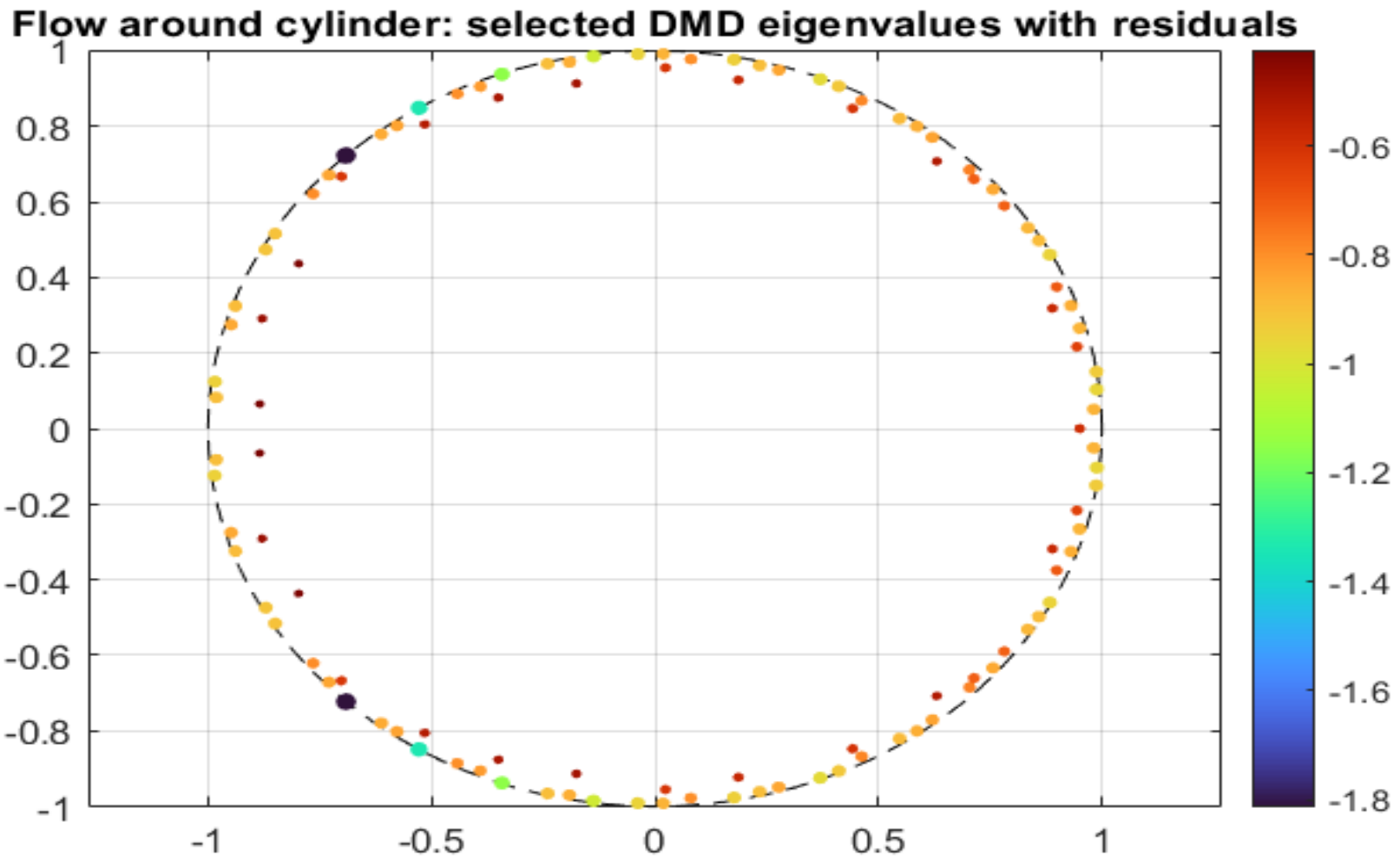
Example: Spectral snapshot reconstruction.





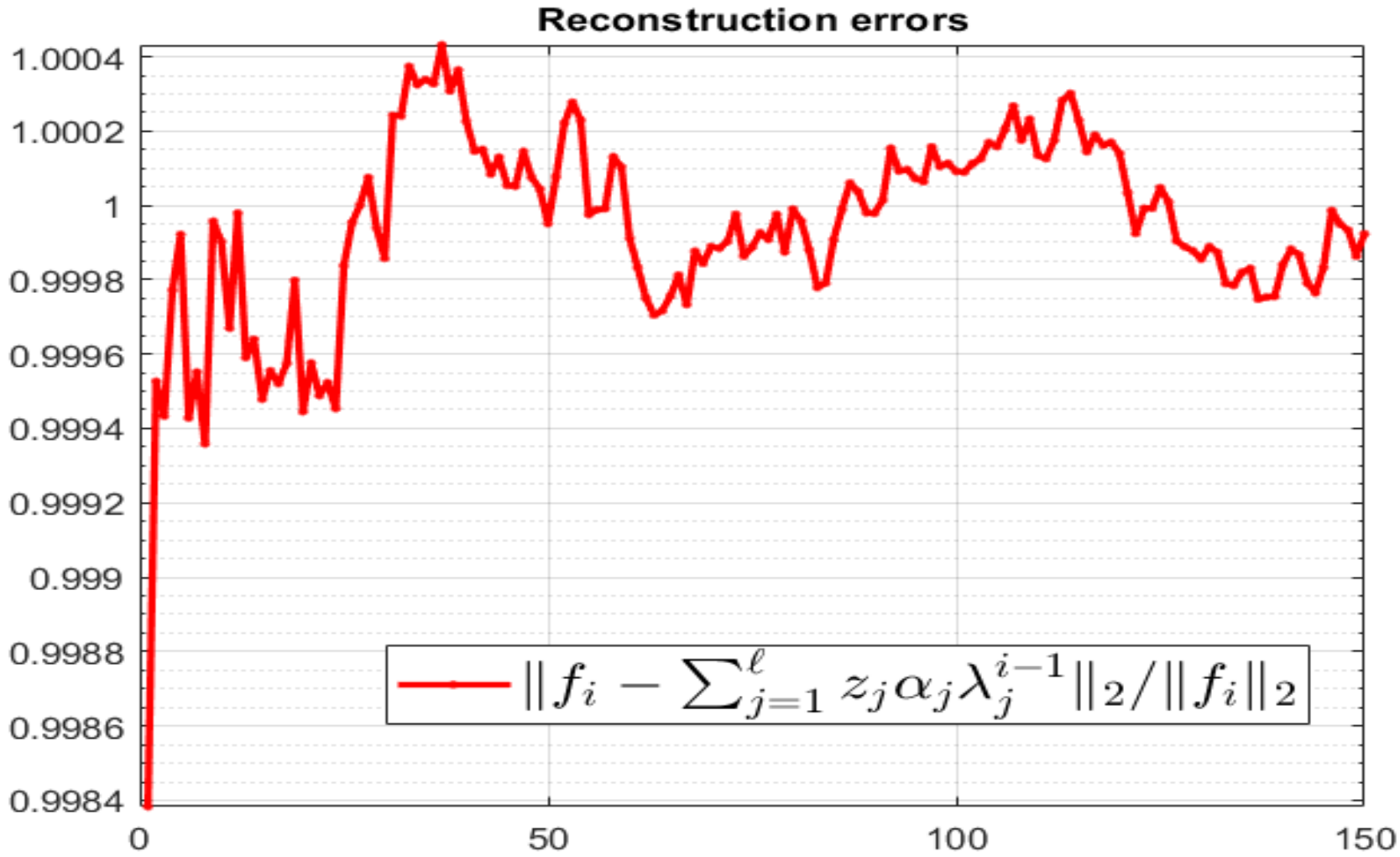
Example: Spectral snapshot reconstruction.

For the sake of an experiment, use the modes with residuals $\|\mathbb{A}z_j - \lambda_j z_j\|_2 > 10^{-2}$. There are 113 of them.



Example: Spectral snapshot reconstruction.

Reconstruction errors when using the modes with residuals $\|\mathbb{A}z_j - \lambda_j z_j\|_2 > 10^{-2}$.



Exercise

Exercise

- Add snapshot reconstruction to your DMD code and repeat this numerical experiment. Also experiment with other data sets.
- Implement a QR compressed version of this reconstruction, integrated with the QR compressed DMD.
- Read on LS solution and normal equations. A good reference is
  - Björck, Åke : Numerical Methods in Matrix Computations, Springer 2015. (See Chapter 2.)
- Experiment with examples of failure of the normal equations.

⇐ 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

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

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 (4) is

$$\boldsymbol{\alpha} = [(R^*R) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})]^{-1}[(\overline{\mathbb{V}_{\ell,m}\mathbf{W}} \circ (R^*G\mathbf{W}))\mathbf{e}], \qquad (12)$$

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,

$$\boldsymbol{\alpha} = [(Z_\ell^*Z_\ell) \circ (\overline{\mathbb{V}_{\ell,m}\mathbf{W}^2\mathbb{V}_{\ell,m}^*})]^{-1}[(\overline{\mathbb{V}_{\ell,m}\mathbf{W}} \circ (Z_\ell^*\mathbf{X}_m\mathbf{W}))\mathbf{e}]. \qquad (13)$$

To ease technical details, $\mathbf{W} = I$, i.e. no weights are used and (12) is $\boldsymbol{\alpha} = (S^*S)^{-1}S^*\vec{\mathbf{g}}$.

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

Projection theorem: The residual $r = \vec{\mathbf{g}} - S\boldsymbol{\alpha}$ must be orthogonal to the range of S, $S^* r = 0$, i.e.

$$S^* S\boldsymbol{\alpha} = S^* \vec{\mathbf{g}}.$$

Let $S^* S = R_S^* R_S$ be the Cholesky factorization; R_S is upper triangular. Then

$$\widetilde{\boldsymbol{\alpha}} = computed(R_S^{-1}(R_S^{-*}(S^* \vec{\mathbf{g}}))),$$

The residual is $\widetilde{r} = \vec{\mathbf{g}} - S\widetilde{\boldsymbol{\alpha}}$. (Note: $S\widetilde{\boldsymbol{\alpha}} = \vec{\mathbf{g}} - \widetilde{r}$.) A corrected solution is obtained as follows:

$$\delta\boldsymbol{\alpha} = R_S^{-1}(R_S^{-*}(S^* \widetilde{r})), \quad \boldsymbol{\alpha}_* = \widetilde{\boldsymbol{\alpha}} + \delta\boldsymbol{\alpha}. \tag{14}$$

See Å. Björck, Stability analysis of the method of seminormal equations for linear least squares problems, Linear Algebra and its Applications Volumes 88-89, April 1987, Pages 31-48.

R_S is the Cholesky factor of $S^* S$, or the triangular factor in the QR factorization of S.

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

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? Sometimes, getting only R_S is acceptable cost, not as bad as the entire QR factorization.

Algorithm: Recursive QR factorization of $S = \mathbb{V}_{\ell,m}^T \odot R$ for $m = 2^p$

Input: Upper triangular $R \in \mathbb{C}^{\ell \times \ell}$; diagonal $\mathbf{\Lambda} \in \mathbb{C}^{\ell \times \ell}$; number of snapshots $m = 2^p$

Output: Upper triangular QR factor $R_S = T_p$ of $S \in \mathbb{C}^{2^p \ell \times \ell}$

$\boxed{T_4}$	$\longleftarrow \boxed{T_3}$	$\longleftarrow \boxed{T_2}$	$\longleftarrow \boxed{T_1}$	$\longleftarrow R\mathbf{\Lambda}^0$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\longleftarrow R\mathbf{\Lambda}^1$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\longleftarrow T_1\mathbf{\Lambda}^2$	$R\mathbf{\Lambda}^2$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^3$
$\mathbf{0}$	$\mathbf{0}$	$\longleftarrow T_2\mathbf{\Lambda}^4$	$T_1\mathbf{\Lambda}^4$	$R\mathbf{\Lambda}^4$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^5$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$T_1\mathbf{\Lambda}^6$	$R\mathbf{\Lambda}^6$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^7$
$\mathbf{0}$	$\longleftarrow T_3\mathbf{\Lambda}^8$	$T_2\mathbf{\Lambda}^8$	$T_1\mathbf{\Lambda}^8$	$R\mathbf{\Lambda}^8$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^9$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$T_1\mathbf{\Lambda}^{10}$	$R\mathbf{\Lambda}^{10}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^{11}$
$\mathbf{0}$	$\mathbf{0}$	$T_2\mathbf{\Lambda}^{12}$	$T_1\mathbf{\Lambda}^{12}$	$R\mathbf{\Lambda}^{12}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^{13}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$T_1\mathbf{\Lambda}^{14}$	$R\mathbf{\Lambda}^{14}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$R\mathbf{\Lambda}^{15}$

,

$1:$ $T_0 = R$

$2:$ **for** $i = 1 : p$ **do**

$3:$ $\begin{pmatrix} \boxed{T_i} \\ \mathbf{0} \end{pmatrix} = \mathtt{qr}\left(\begin{pmatrix} \boxed{T_{i-1}} \\ T_{i-1}\mathbf{\Lambda}^{2^{i-1}} \end{pmatrix} \right)$

$4:$ **end for**

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

```
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 $\mathbf{\Lambda} \in \mathbb{C}^{\ell\times\ell}$; m

Output: Upper triangular QR factor $R_S = \mathbb{T}_{j-1}$ of S.

Compute the binary representation of m:
$m \equiv \mathfrak{b} = (\mathfrak{b}_{\lfloor \log_2 m \rfloor}, \ldots, \mathfrak{b}_1, \mathfrak{b}_0)_2$, $m \equiv \sum_{j=1}^{j^*} 2^{i_j}$
Let $\lfloor \log_2 m \rfloor = i_{j^*} > i_{j^*-1} > \cdots > i_2 > i_1 \geq 0$
$T_0 = R$
if $i_1 = 0$ **then**
  $\mathbb{T}_1 = T_0$; $j = 2$; $\wp = 1$
else
  $\mathbb{T}_0 = []$; $j = 1$; $\wp = 0$
end if
for $k = 1 : i_{j^*}$ **do**
  $\begin{pmatrix} T_k \\ \mathbf{0} \end{pmatrix} = \mathtt{qr}\left(\begin{pmatrix} T_{k-1} \\ T_{k-1}\mathbf{\Lambda}^{2^{k-1}} \end{pmatrix}\right)$ {Local triangular factor.}
  if $k = i_j$ **then**
    if $\mathbb{T}_{j-1} \neq []$ **then**
      $\begin{pmatrix} \mathbb{T}_j \\ \mathbf{0} \end{pmatrix} = \mathtt{qr}\left(\begin{pmatrix} \mathbb{T}_{j-1} \\ T_k\mathbf{\Lambda}^{\wp} \end{pmatrix}\right)$ {Global triang. factor.}
    else
      $\mathbb{T}_j = T_k$
    end if
    $j := j + 1$; $\wp := \wp + 2^k$
  end if
end for

Comments: condition number and implementation details

QR factorization approach to the LS reconstruction:

- Provably small backward error

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

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

Corollary

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

Square root of the condition number is great advantage.

- Other technical details: If the data is real, can work in real arithmetic even if the eigenvalues are complex (conjugate pairs)

Numerical example: limits of normal equations formula

Besides contrived examples where QRF approach outperforms the commonly used method, it is interesting to point out that in many interesting cases the normal equations approach fails dramatically, while the QR based approach provably succeeds.

An example: Ikeda map

For instance, we used the Hankel matrix rearrangement of the snapshots generated by the Ikeda map (evolution of laser light across a nonlinear optical resonator)

$$\begin{aligned} x_{n+1} &= \phi + \psi(x_n \cos(\rho - \tfrac{\omega}{1+x_n^2+y_n^2}) - y_n \sin(\rho - \tfrac{\omega}{1+x_n^2+y_n^2})) \\ y_{n+1} &= \psi(x_n \sin(\rho - \tfrac{\omega}{1+x_n^2+y_n^2}) - y_n \cos(\rho - \tfrac{\omega}{1+x_n^2+y_n^2})) \end{aligned}, n = 0, 1, \ldots$$

with $\phi = 1$, $\psi = 0.6$, $\rho = 0.4$, $\omega = 6$, and initial condition (x_0, y_0). We generated 3500 snapshots and arranged them in the 5802×600 Hankel matrix. The widely used normal equations approach failed in double precision (16 digits arithmetic), while the QR factorization based algorithm can deliver accuracy to eight decimal places.

Numerical example: effect of weighted reconstruction

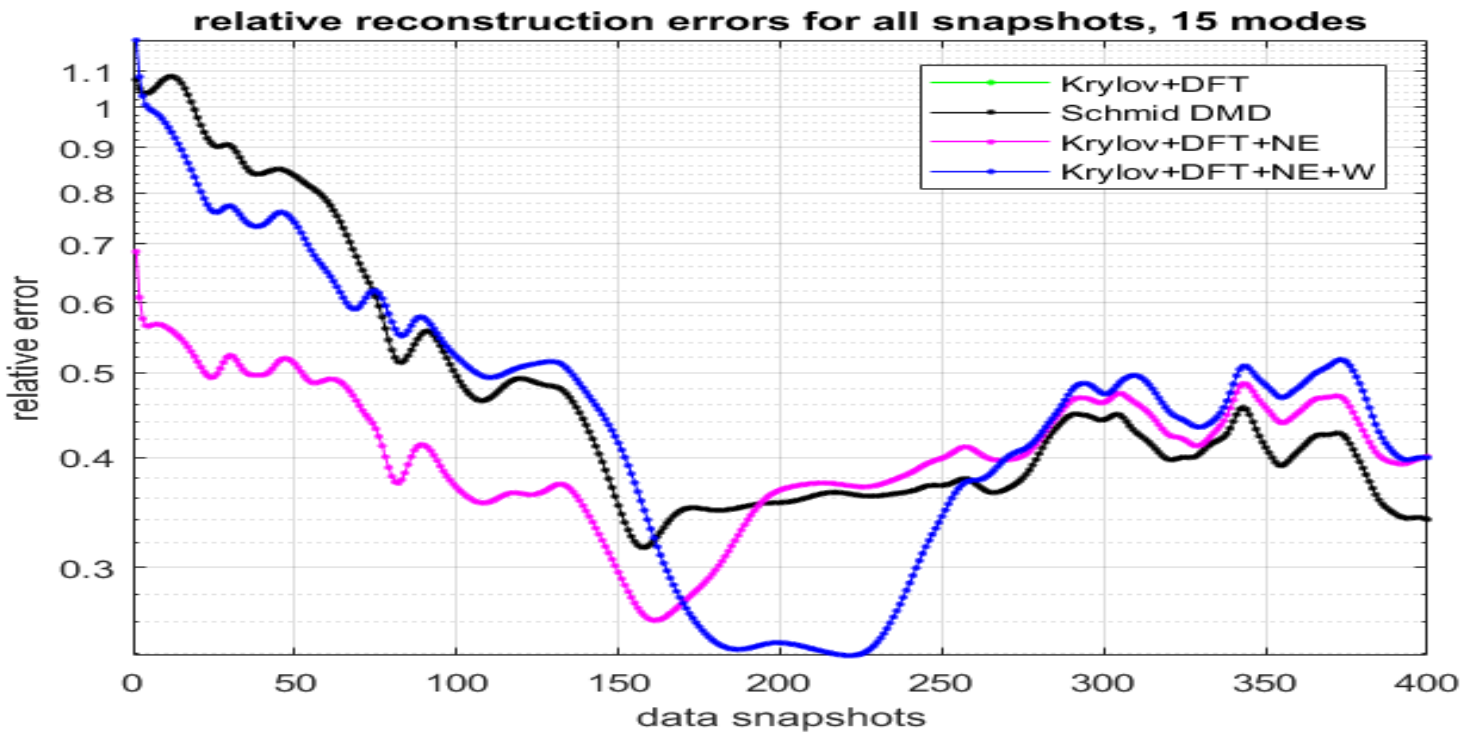


Figure: In all cases, the errors of the Schmid DMD and the Krylov+DFT algorithm (with the coefficients from the full reconstruction) are nearly the same, so the graphs overlap. Recomputing the coefficient using normal equation significantly reduces the error (-., Krylov+DFT+NE). The blue curve shows the effects of weighting. Note how by choosing the weights we can enforce higher reconstruction accuracy for snapshot in a specified (discrete time) subinterval.

Numerical example: effect of weighted reconstruction

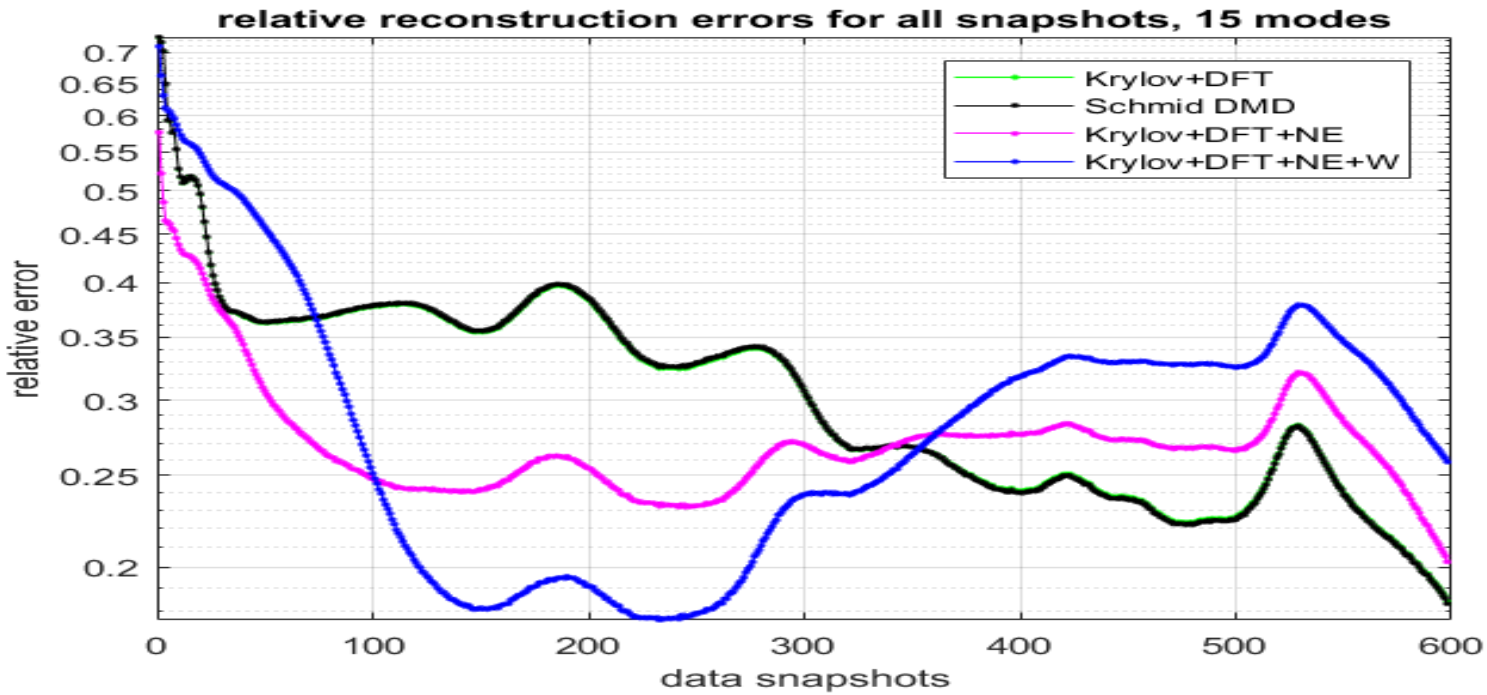


Figure: Example with 600 snapshots. Blue curve shows the effects of weighting.

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] Thanks M.Schatz, B. Suri, R. Grigoriev and L. Kageorge from the Georgia Institute of Technology for providing the data.

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, Λ_k

On the DMD matrix

In the DMD literature, the DMD matrix $\mathbb{A}$ is defined as the solution of the least squares problem

$$\|\mathbf{Y}_m - A\mathbf{X}_m\|_F \to \min_A . \quad (\Longleftrightarrow \|\mathbf{X}_m^T A^T - \mathbf{Y}_m^T\|_F \to \min_A)$$

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

Independent of the choice of

$$\mathbb{A} \in \mathrm{argmin}_A \|\mathbf{Y}_m - A\mathbf{X}_m\|_F,$$

it holds that $\mathbb{A}\mathbf{X}_m = \mathbf{Y}_m\mathbf{P}_{\mathbf{X}_m^T}$, where $\mathbf{P}_{\mathbf{X}_m^T}$ is the orthogonal projector onto the range of $\mathbf{X}_m^T$.

On the DMD matrix

In the DMD theory, the specifications for $\mathbb{A}$ is strenghtened with a constraint of minimality of $\|\mathbb{A}\|_F$, which yields

$$\mathbb{A} = \mathbf{Y}_m \mathbf{X}_m^\dagger,$$

expressed using the Moore-Penrose pseudoinverse $\mathbf{X}_m^\dagger$ of $\mathbf{X}_m$.

The interpretability of such a constraint, besides ensuring unique least squares solution, is rather vague. Keep in mind that the only information contained in the data is that $\mathbb{A}\mathbf{X} = \mathbf{Y}\mathbf{P}_{\mathbf{X}^T}$ so that $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$ is just a particular element in the linear manifold

$$[\mathbb{A}] = \{\mathbf{Y}\mathbf{X}^\dagger + B \ : \ B\mathbf{X} = \mathbf{0}\}. \tag{1}$$

Using the particular choice $\mathbb{A} = \mathbf{Y}\mathbf{X}^\dagger$ can be useful in some estimates if one can exploit the fact that in that case $\|\mathbb{A}\|_F$ is minimal.

Exact DMD

Note that the above computation of the Ritz pairs we never used $\mathbb{A} = \mathbf{Y}_m \mathbf{X}_m^\dagger$; instead we used that

$$\mathbb{A}\mathbf{X}_m = \mathbf{Y}_m V_r V_r^*,$$

which is equivalent to say that $\mathbb{A}$ is a solution to $\|\mathbf{Y} - \mathbb{A}\mathbf{X}\|_F \to \min$.

The RQ S_k is obtained using the SVD for the best rank k approximation $\mathbf{X}_m \approx U_k \Sigma_k V_k^*$, $\mathbf{X}_m^\dagger \approx V_k \Sigma_k^{-1} U_k^*$, and then $\mathbb{A}_k = \mathbf{Y}_m V_k \Sigma_k^{-1} U_k^*$.

Further, nothing is gained if we try to use the non-uniqueness and replace $\mathbb{A}$ with some $\widetilde{\mathbb{A}} = \mathbb{A} + B$ such that $B\mathbf{X}_m = \mathbf{0}$ (i.e. $\widetilde{\mathbb{A}} \in [\mathbb{A}]$). Then $BU_k = \mathbf{0}$, $\widetilde{\mathbb{A}} U_k = \mathbb{A} U_k = \mathbf{Y}_m V_k \Sigma_k^{-1}$ and

$$\widetilde{S}_k = U_k^* \widetilde{\mathbb{A}} U_k = U_k^* \mathbb{A} U_k = S_k.$$

Exact DMD

A variant of the DMD, proposed in [Tu, Rowley, 2014, §2.2, §2.3] and designated as the Exact Dynamic Mode Decomposition (Exact DMD) is entirely built on the computation of exact eigenvalues and eigenvectors of $\mathbb{A} = \mathbf{Y}\mathbf{X}^{\dagger}$. Since $\mathbf{Y}$ is $\mathbb{A}$-invariant this is possible.

The Excact DMD algorithm follows the lines 1.–5. of the DMD Algorithm, and in the last step, instead of $Z_k(:,i) = U_k W_k(:,i)$, for a computed nonzero eigenvalue λ_i, the corresponding eigenvector is returned as

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

Note that $Z_k^{(ex)}(:,i) = (1/\lambda_i)\mathbb{A}U_k W_k(:,i) = (1/\lambda_i)\mathbb{A}Z_k(:,i)$.

This modification can be understood/interpreted as follows: If v is a unit eigenvector belonging to a nonzero eigenvalue μ of a matrix M, then $\widetilde{v} = (1/\mu)Mv = v$. If v is only an approximate eigenvector, then Mv is one step of the power method that may contribute (without guarantee) to improving v in the direction of the dominant eigenvector.

Exact DMD

Proposition

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

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

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

An observation:

The vectors $\lambda_i Z_k^{(ex)}(:,i)$, $i = 1,\ldots,k$ are computed if the residuals $\|\mathbb{A}Z_k(:,i) - \lambda_i Z_k(:,i)\|_2$ for the pairs $(\lambda_i, U_kW_k(:,i))$ are requested.

Exact DMD

Remark

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

$$U_k U_k^* \mathbf{Y} V_k \Sigma_k^{-1} W_k(:,i) = U_k S_k W_k(:,i) = \lambda_i U_k W_k(:,i), \quad \|W_k(:,i)\|_2 = 1,$$

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

Exercise

Test the "exactness" of the Exact DMD by first generating A, and then using A to generate $\mathbf{X}_m$ and $\mathbf{Y}_m$. Compare the eigenvalues and eigenvectors computed by the Exact DMD with the corresponding values of the true matrix A (not accessible to the algorithm).

Symmetric DMD

Physics informed DMD (piDMD)

In a framework of physics informed DMD (piDMD), a prior knowledge of the underlying dynamics determines that $A \in \mathcal{M}$, where a matrix manifold $\mathcal{M}$ is defined by the additional (physics informed) constraints such that e.g. A must be Hermitian, or skew-Hermitian, unitary, Toeplitz etc.

- [piDMD] Baddoo P. J., Herrmann B., McKeon B. J., Kutz J. N., and Brunton S. L.. 2021. Physics-informed dynamic mode decomposition (piDMD). arXiv:2112.04307

Let us consider the case when $\mathcal{M}$ stands for Hermitian matrices. It is nicely motivated by numerical examples e.g. with learning the energy states of a quantum Hamiltonian [piDMD,§4.3.1] where it is shown that the loss of hermiticity/symmetry may result in a non-physical and thus inaccurate/useless results.

Symmetric DMD

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

This means that in an error-free setting the DMD algorithm will automatically exploit symmetry, and the computed Ritz pairs will have the proper structure – real Ritz values and orthonormal Ritz vectors. There is no need to determine a Hermitian $H \in \operatorname{argmin}_A \|A\mathbf{X} - \mathbf{Y}\|_F$. Note also that the Rayleigh quotient inherits the positive definiteness of $\mathbf{H}$.

In real world applications (noisy data, finite precision), the symmetry will be lost. This is easy to illustrate by numerical example.

Symmetric DMD: loss of symmetry

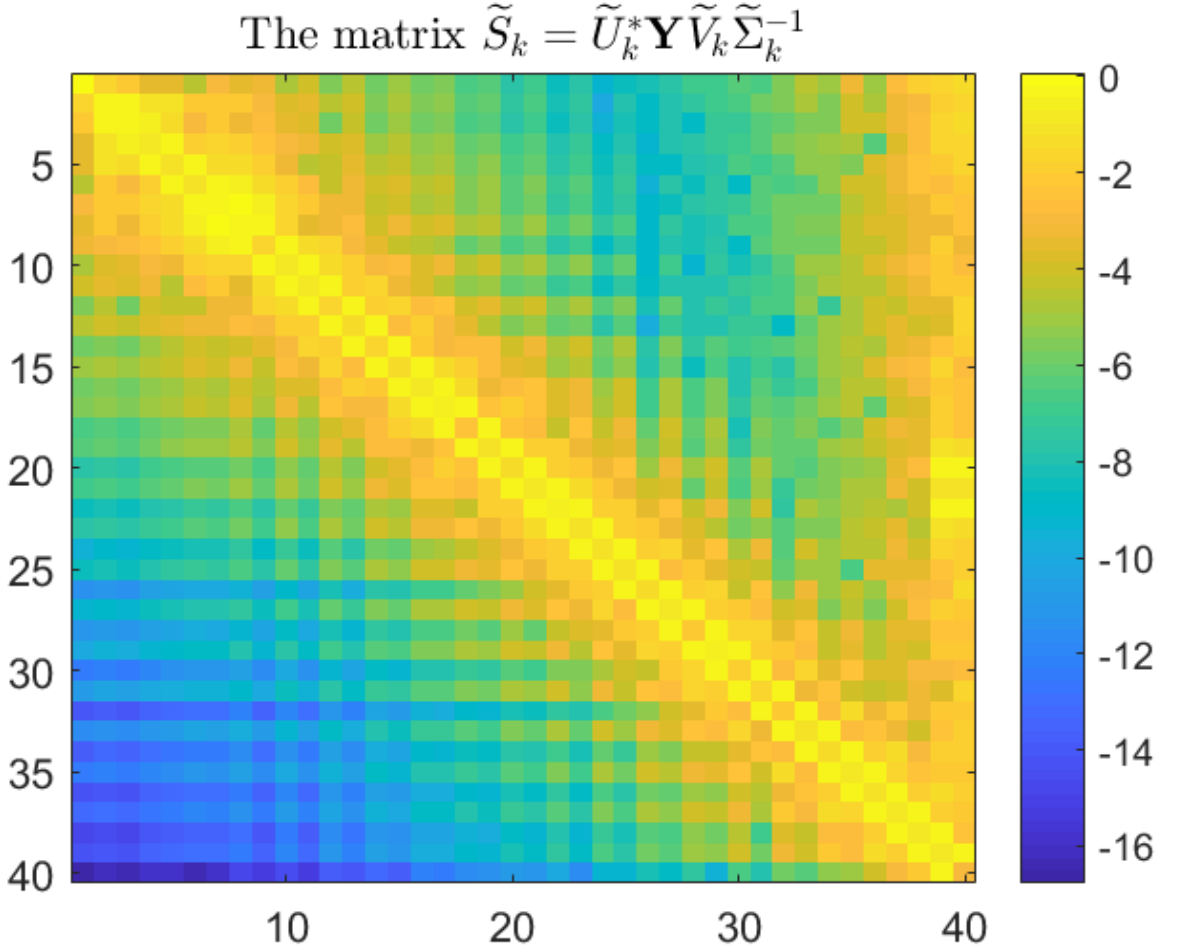


Figure: The structure of the computed $\widetilde{S}_k = \widetilde{U}^* \mathbf{Y} \widetilde{V}_k \widetilde{\Sigma}_k^{-1}$, visualized using `imagesc(log10(abs(`$\widetilde{S}_k$`)))`. The loss of symmetry is apparent.

To fix the problem, we first have to analyze it.

Loss of symmetry: structure of the error

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

$$(\mathbf{X} + \delta\mathbf{X})\widetilde{V} = \widetilde{U}\widetilde{\Sigma}, \quad \|\delta\mathbf{X}\|_2 \leq \epsilon_x \|\mathbf{X}\|_2, \tag{1}$$

where $\widetilde{U}$ and $\widetilde{V}$ are numerically unitary matrices, $\|\widetilde{U}^*\widetilde{U} - \mathbb{I}_n\|_2 \leq \epsilon_u$, $\|\widetilde{V}^*\widetilde{V} - \mathbb{I}_n\|_2 \leq \epsilon_v$, and $\widetilde{\Sigma} = \mathrm{diag}(\widetilde{\sigma}_j)_{j=1}^n$. Here $\epsilon_u, \epsilon_v, \epsilon_x$ depend on the details of a particular algorithm and its software implementation, and can be estimated by $f(m,n)\boldsymbol{\varepsilon}$, where $f(m,n)$ is a modestly growing function and $\boldsymbol{\varepsilon}$ is the round-off unit of the working precision.

Assume that $\mathbf{X}$ and $\mathbf{X} + \delta\mathbf{X}$ are of full column rank. Then $P_{\mathbf{X}^*} = \mathbb{I}_n$ and

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

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

Loss of symmetry: structure of the error

We can estimate $E_k = \mathbf{H}\delta\mathbf{X}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$ as follows:

$$\frac{\|E_k(:,j)\|_2}{\|\mathbf{H}\|_2} \leq \|\delta\mathbf{X}\|_2\|\widetilde{V}_k(:,j)\|_2/\widetilde{\sigma}_j \leq \epsilon_x\sigma_1\|\widetilde{V}_k(:,j)\|_2/\widetilde{\sigma}_j \tag{2}$$

$$\leq \epsilon_x\frac{\sqrt{1+\epsilon_v}}{1-\epsilon_x}\frac{\widetilde{\sigma_1}}{\widetilde{\sigma}_j} \tag{3}$$

$$\frac{\|E_k(:,j)\|_2}{\|\mathbf{H}\widetilde{U}_k(:,j)\|_2} \leq \frac{\|\mathbf{H}\|_2}{1/\|(\mathbf{H}_{|\mathrm{range}(U_k)})^\dagger\|_2}\epsilon_x\frac{\sqrt{1+\epsilon_v}}{\sqrt{1-\epsilon_u}(1-\epsilon_x)}\frac{\widetilde{\sigma_1}}{\widetilde{\sigma}_j}, \tag{4}$$

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

$$\frac{\|\delta S_k(:,j)\|_2}{\|\mathbf{H}\|_2} \leq \frac{\|\widetilde{U}_k^*\mathbf{H}\|_2}{\|\mathbf{H}\|_2}\epsilon_x\frac{\sqrt{1+\epsilon_v}}{1-\epsilon_x}\frac{\widetilde{\sigma_1}}{\widetilde{\sigma}_j}.$$

These bounds indicate that the accuracy in $\widetilde{S}_k$ may be deteriorating with the increased column index, which means that the upper triangle of $\widetilde{S}_k$ may be more exposed to the effects of $\delta\mathbf{X}$ (i.e. the errors in the computation of the SVD of $\mathbf{X}$) and the column scaling by $\widetilde{\Sigma}_k^{-1}$.

Loss of symmetry: structure of the error

Clearly, the accuracy of the computed residual will be also affected, and the above analysis gives an estimate. This simple model can also be used to assess the effects of the noise $\Delta\mathbf{X}$, $\Delta\mathbf{Y}$ in the initial data.
Note that the above analysis does not include rounding errors in the computation $\widetilde{U}_k^*\mathbf{Y}\widetilde{V}_k\widetilde{\Sigma}_k^{-1}$, because they are not the main source of the loss of symmetry.

Motivated by the above analysis, we define a symmetrizer:

Symmetrizer of $\widetilde{S}_k$

$$\widetilde{H}_k = \mathrm{diag}((\widetilde{S}_k)_{ii})_{i=1}^{k} + \widetilde{L}_k + \widetilde{L}_k^*, \tag{5}$$

where $\widetilde{L}_k$ is the strict lower triangle of $\widetilde{S}_k$, and we consider it as a candidate to replace $\widetilde{S}_k$.

Let us compare the estimated and the actual error in a controlled synthetic experiment, using $\widetilde{S}_k$ (from the last Figure).

Loss of symmetry: structure of the error

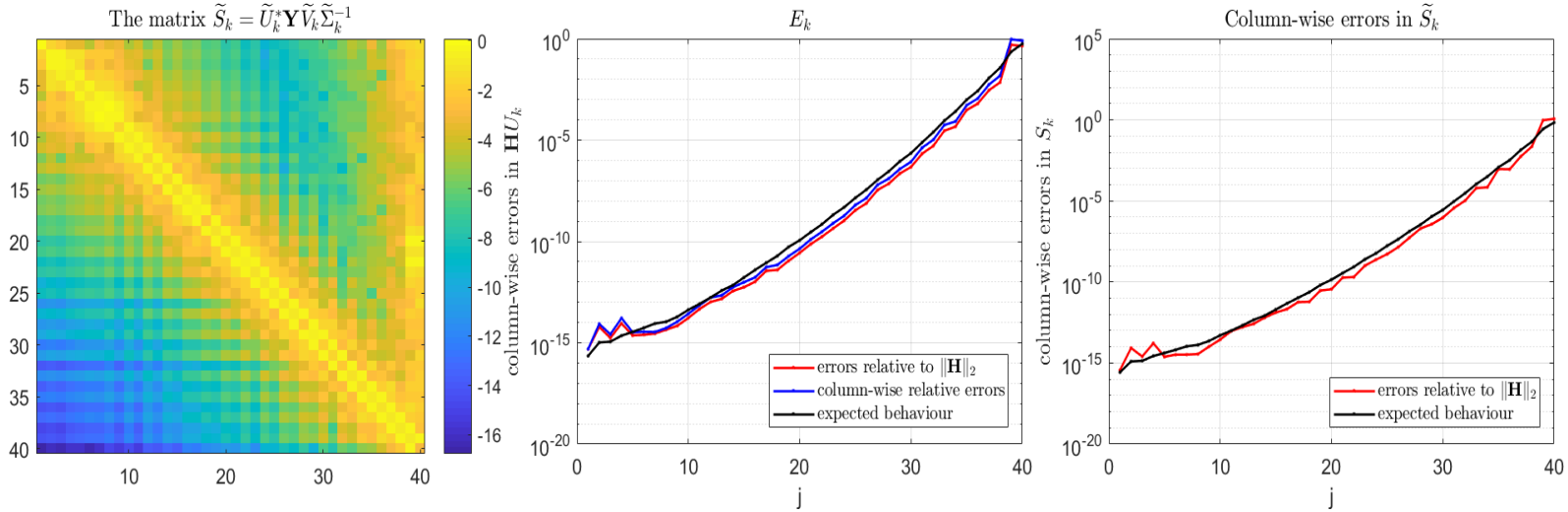


Figure: *First panel:* the structure of the computed $\widetilde{S}_k = \widetilde{U}^* \mathbf{Y} \widetilde{V}_k \widetilde{\Sigma}_k^{-1}$, visualized using `imagesc(log10(abs(`$\widetilde{S}_k$`)))`. The loss of symmetry is apparent. *Middle panel:* the column norms of E_k and their predicted trend. *Third panel:* the columns of $\delta\widetilde{S}_k$ and their predicted trend.

Except for the error at the noise level $m\varepsilon$, the analysis (although simplified) correctly reveals/predicts the behavior of the error. Hence, using the symmetrizer $\widetilde{H}_k$ might work. What else could one try? For instance, matrix theory provides provably optimal symmetric approximation of $\widetilde{S}_k$.

Symmetric DMD

A natural way to correct $\widetilde{S}_k$ and restore hermiticity is to replace it with a close Hermitian matrix.

Theorem (K. Fan, A. J. Hoffman. Some metric inequalities in the space of matrices. Proc. Amer. Math. Soc. 6, 1 (1955), 111-116.)

The matrix

$$\widehat{S}_k = \frac{1}{2}(\widetilde{S}_k + \widetilde{S}_k^*) \tag{6}$$

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

$$\|\widetilde{S}_k - \widehat{S}_k\| = \min_{H=H^*} \|\widetilde{S}_k - H\|. \tag{7}$$

This optimality of $\widehat{S}_k$ in a large class of norms, as well as its simple computation (6), makes it a good candidate to replace $\widetilde{S}_k$. Is this the best we can do? Is it superfluous to ask whether is it best to chose the optimal approximation $\widehat{S}_k$, or, what could go wrong if we replaced $\widetilde{S}_k$ with its closest Hermitian matrix?

Loss of symmetry: structure of the error

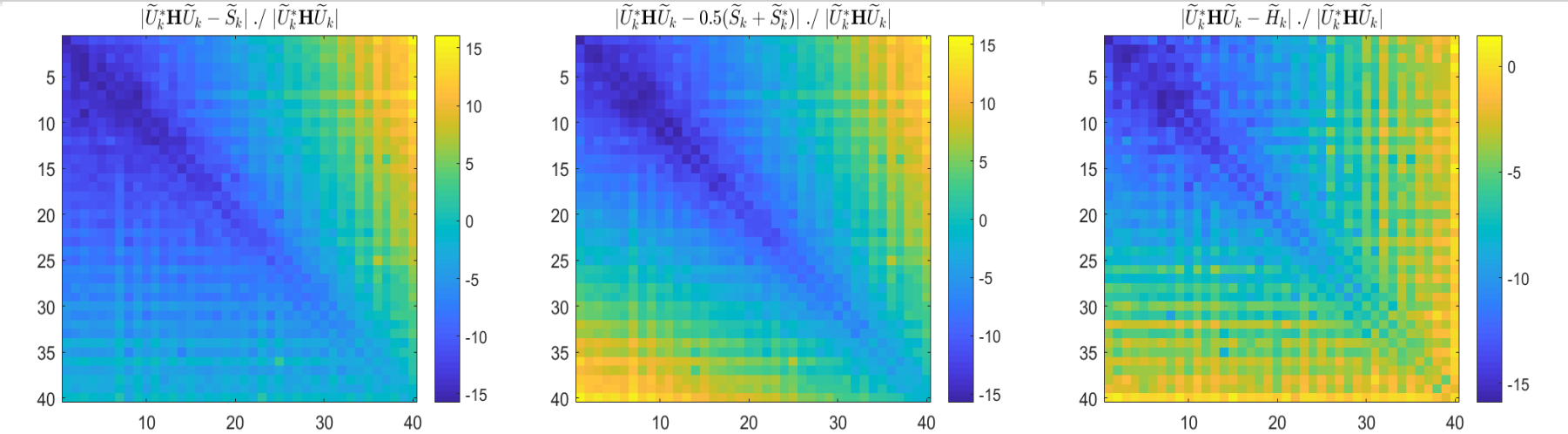


Figure: The entry-wise relative errors $\log_{10}(|(S_k)_{ij} - (\widetilde{S}_k)_{ij}|/|(S_k)_{ij}|)$ (*first panel*) and $\log_{10}(|(S_k)_{ij} - (\widehat{S}_k)_{ij}|/|(S_k)_{ij}|)$ (*second panel*). Note that the upper triangle of $\widetilde{S}_k$ has large error that is symmetrized in $\widehat{S}_k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

We note here that requiring entry-wise small relative errors is indeed too much to ask, but nevertheless we check them to test whether the above analysis correctly identifies the problem. The result shown in the Figure are precisely as predicted.

Loss of symmetry: structure of the error

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

If $\mathbf{H}$ is positive (semi)definite, then S_k inherits the definiteness, but in ill-conditioned cases a symmetrizer of $\widetilde{S}_k$ is not guaranteed to be positive (semi)definite. Under this implicit assumption on (semi)definiteness, if we compute the spectral decomposition of $\widetilde{H}_k$ (or any other symmetrizer) and if some of the eigenvalues are negative, we can replace them with zeros, thus implicitly replacing $\widetilde{H}_k$ with the closest positive semidefinite matrix.[a] Depending on the user's preferences, all or only positive Ritz values can be returned.

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

Symmetric DMD – a Procrustes' approach

Procrustes' approach

To mitigate the problem, [piDMD] proposes selecting a DMD matrix as

$$A \in \operatorname{argmin}_{A=A^*} \|A\mathbf{X} - \mathbf{Y}\|_F. \tag{8}$$

This is a well studied structured (symmetric/Hermitian) Procrustes problem with an explicitly known solution by Nick Higham

- Higham Nicholas J.. 1988. The symmetric Procrustes problem. BIT 28 (1988).

Orthogonal/unitary Procrustes' problem

The orthogonal Procrustes' problem is

$$\|Y - XQ\|_F \longrightarrow \min_{Q^*Q=I}$$

The symmetric problem (8) can be solved using the SVD of $\mathbf{X}$.

Symmetric DMD – a Procrustes' approach

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

$$\begin{aligned} \|A\mathbf{X} - \mathbf{Y}\|_F^2 &= \|AU_x \begin{pmatrix} \Sigma \\ \mathbf{0} \end{pmatrix} V^* - \mathbf{Y}\|_F^2 = \| \underbrace{(U_x^* A U_x)}_{M} \begin{pmatrix} \Sigma \\ \mathbf{0} \end{pmatrix} - \underbrace{U_x^* \mathbf{Y} V}_{C} \|_F^2 \\ &= \|M \begin{pmatrix} \Sigma \\ \mathbf{0} \end{pmatrix} - C\|_F^2 = \| \begin{pmatrix} G & L^* \\ L & K \end{pmatrix} \begin{pmatrix} \Sigma \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} C_{[1]} \\ C_{[2]} \end{pmatrix} \|_F^2, \\ & \quad G = G^* \in \mathbb{C}^{m\times m},\ C_{[1]} \in \mathbb{C}^{m\times m}, \\ &= \|G\Sigma - C_{[1]}\|_F^2 + \|L\Sigma - C_{[2]}\|_F^2. \end{aligned} \tag{9}$$

Clearly, $K = K^* \in \mathbb{C}^{(n-m)\times(n-m)}$ can be taken arbitrary Hermitian, and the optimal choice of L in the second term in (9) is

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

Symmetric DMD – a Procrustes' approach

Further, taking the hermiticity into account, the first term in (9) reads

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

which is minimized for

$$g_{ii} = \begin{cases} \dfrac{\Re(c_{jj})}{\sigma_j}, & j = 1, \ldots, r \\ \text{arbitrary real}, & j = r+1, \ldots, m \end{cases}, \tag{10}$$

$$g_{ij} = \overline{g_{ji}} = \begin{cases} \dfrac{\sigma_j c_{ij} + \sigma_i \overline{c_{ji}}}{\sigma_i^2 + \sigma_j^2}, & \sigma_i + \sigma_j \neq 0 \\ \text{arbitrary whenever } \sigma_i + \sigma_j = 0 \ \ (\sigma_i = \sigma_j = 0) \end{cases}. \tag{11}$$

Symmetric DMD – a Procrustes' approach

The structure of M can be illustrated as follows:

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

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

Any matrix $A = A^*$ that solves the Hermitian Procrustes problem is then of the form $A = U_x M U_x^*$.

Symmetric DMD – a Procrustes' approach

Note that $A = U_x M U_x^*$ is $n \times n$ and forming it explicitly is not needed. A low rank approximation of A, proposed in [piDMD] is $\mathbb{A}_\pi = U_k G_k U_k^*$, where $G_k = G(1:k, 1:k)$ and $U_k = U_x(:, 1:k)$. (Note that there is no guarantee that $\mathbb{A}_\pi = \mathbb{A}_\pi^*$ is in the solution set of (8).) Then, using the spectral decomposition $G_k = W \Lambda W^*$, $W^* W = I_k$, the Ritz vectors are computed as the columns of $U_k W$ and the Ritz values are $\lambda_i = \Lambda_{ii} \in \mathbb{R}$.

A closer look at these formulas reveals that the elements c_{ij} used in (10) to compute G_k are the entries of the matrix $C_k = U_k^* \mathbf{Y} V_k$, which is actually $C_k = S_k \Sigma_k$, where $S_k = U_k^* \mathbf{H} U_k$ is Hermitian. This implies in (10) that, for $1 \leq i, j \leq k \leq r$,

$$g_{ij} = \overline{g_{ji}} = \frac{\sigma_j c_{ij} + \sigma_i \overline{c_{ji}}}{\sigma_i^2 + \sigma_j^2} = \frac{\sigma_j^2 s_{ij} + \sigma_i^2 \overline{s_{ji}}}{\sigma_i^2 + \sigma_j^2} = \frac{\sigma_j^2 s_{ij} + \sigma_i^2 s_{ij}}{\sigma_i^2 + \sigma_j^2} = s_{ij} = \overline{s_{ji}}. \tag{12}$$

In other words, using a low rank approximation of a solution of the structured Procrustes problem (8) did not produce anything new if the data is indeed generated by a Hermitian matrix.

Symmetric DMD – a Procrustes' approach

Check how this works on the previous synthetic example:

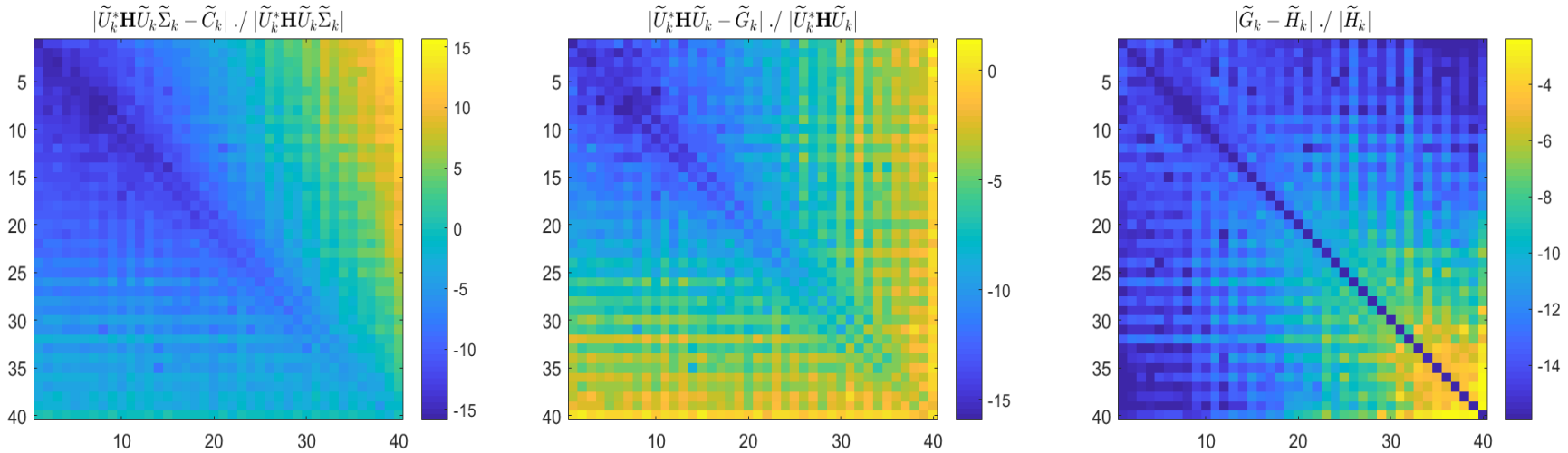


Figure: *First panel:* The entry-wise relative errors $\log_{10}(|(C_k)_{ij} - (\widetilde{C}_k)_{ij}|/|(C_k)_{ij}|)$ where $C_k = \widetilde{U}_k^*\mathbf{H}\widetilde{U}_k\widetilde{\Sigma}_k$ is computed explicitly using $\mathbf{H}$. *Second panel:* $\log_{10}(|(S_k)_{ij} - (\widetilde{G}_k)_{ij}|/|(S_k)_{ij}|)$,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

Symmetric DMD – a Procrustes' approach

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

$$\widetilde{g}_{ij} = \frac{\widetilde{\sigma}_j \widetilde{c}_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i \overline{\widetilde{c}_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} = \frac{\widetilde{\sigma}_j^2 \widetilde{s}_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i^2 \overline{\widetilde{s}_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \tag{13}$$

$$= \frac{\widetilde{\sigma}_j^2 (s_{ij} - \delta\widetilde{s}_{ij})}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i^2 (\overline{s_{ji}} - \overline{\delta\widetilde{s}_{ji}})}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2}$$

$$= \frac{\widetilde{\sigma}_j^2 s_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} + \frac{\widetilde{\sigma}_i^2 \overline{s_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} - \frac{\widetilde{\sigma}_j^2 \delta\widetilde{s}_{ij}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} - \frac{\widetilde{\sigma}_i^2 \overline{\delta\widetilde{s}_{ji}}}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \tag{14}$$

$$= s_{ij} - \left(\frac{\widetilde{\sigma}_j^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \delta\widetilde{s}_{ij} + \frac{\widetilde{\sigma}_i^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2} \overline{\delta\widetilde{s}_{ji}}\right) \tag{15}$$

Symmetric DMD – a Procrustes' approach

Hence

$$s_{ij} = \widetilde{g}_{ij} + \delta\widetilde{g}_{ij}, \quad \delta\widetilde{g}_{ij} = \left(\frac{\widetilde{\sigma}_j^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2}\delta\widetilde{s}_{ij} + \frac{\widetilde{\sigma}_i^2}{\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2}\overline{\delta\widetilde{s}_{ji}}\right). \tag{16}$$

Note that the entries of $\widetilde{G}_k$ are computed from the entries of $\widetilde{S}_k$ as convex combinations that put more weight of the more accurate lower triangle. Indeed, for $\widetilde{\sigma}_j \ll \widetilde{\sigma}_i$, δs_{ij} is scaled with $\widetilde{\sigma}_j^2/(\widetilde{\sigma}_i^2 + \widetilde{\sigma}_j^2) \ll 1$. This is illustrated in Figure 21.

Note the difference from the computation of $\widehat{S}_k$ in (6) where the upper and the lower triangle are averaged, in a convex combination with the coefficient $1/2$, i.e. $\widehat{S}_k = S_k - (0.5\delta\widetilde{S}_k + 0.5\delta\widetilde{S}_k^*)$. Further, $\widetilde{H}_k$ computed as in (5), is contaminated only by the lower triangle of the error in $\widetilde{S}_k$.

Symmetric DMD – a Procrustes' approach

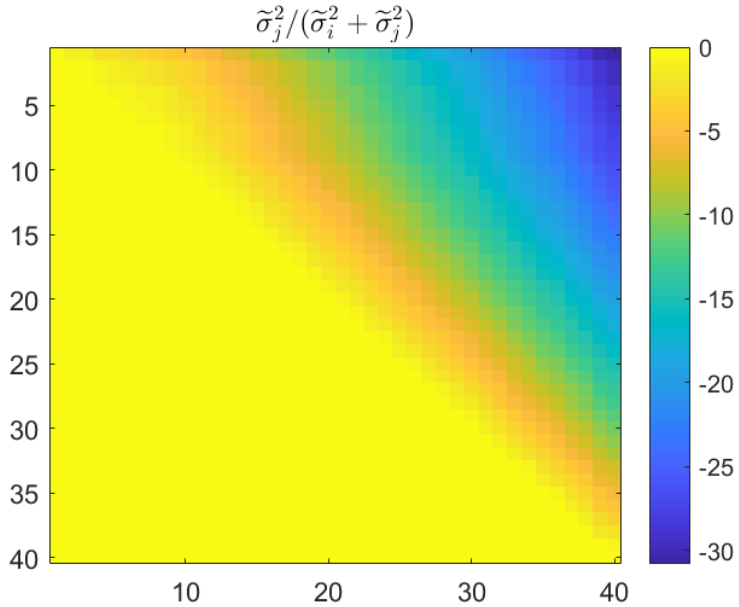


Figure: The distribution of the values $\widetilde{\sigma}_j^2/(\widetilde{\sigma}_i^2+\widetilde{\sigma}_j^2)$ illustrate why the large errors in the northeastern corner of $\widetilde{C}_k$ did not perturb the entries of $\widetilde{G}_k$ too much. The two triangles of G_k are differently weighted in the convex combination in relation (12), and the expressions for the entry-wise errors (15), (16) explain the observed accuracy and distribution of the errors in the matrix entries.

Symmetric DMD - QR compression I

In the case of data from a single trajectory $\mathbf{S} = (\mathbf{z}_1, \ldots, \mathbf{z}_m, \mathbf{z}_{m+1})$, we have $\mathbf{X} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$, $\mathbf{Y} = (\mathbf{z}_2, \ldots, \mathbf{z}_{m+1})$, and the auxiliary subspace is of dimension $m+1$. If we compute the QR factorization

$$(\mathbf{z}_1, \ldots, \mathbf{z}_m, \mathbf{z}_{m+1}) = Q \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix} = \widehat{Q} R, \quad Q^* Q = \mathbb{I}_n, \quad \widehat{Q} = Q(:, 1 : m+1), \tag{17}$$

then

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

where

$$R = \begin{pmatrix} \times & \divideontimes & \divideontimes & \divideontimes & \div \\ & \divideontimes & \divideontimes & \divideontimes & \div \\ & & \divideontimes & \divideontimes & \div \\ & & & \divideontimes & \div \\ & & & & \div \end{pmatrix}, \quad R_x = R(:, 1 : m) = \begin{pmatrix} \times & \divideontimes & \divideontimes & \divideontimes \\ & \divideontimes & \divideontimes & \divideontimes \\ & & \divideontimes & \divideontimes \\ & & & \divideontimes \\ & & & 0 \end{pmatrix}, \quad R_y = R(:, 2 : m+1) \tag{18}$$

Symmetric DMD - QR compression II

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

$$\mathbf{X}' = \begin{pmatrix}R_x\\ \mathbf{0}\end{pmatrix},\ \mathbf{Y}' = \begin{pmatrix}R_y\\ \mathbf{0}\end{pmatrix}.$$

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

$$\mathbb{M} = \mathbf{Y}'(\mathbf{X}')^\dagger = Q^*\mathbf{Y}\mathbf{X}^\dagger Q = Q^*\mathbb{A}Q,$$

Symmetric DMD - QR compression III

$$\mathbf{Y}'(\mathbf{X}')^{\dagger} = \begin{pmatrix} R_y \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} R_x^{\dagger} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} R_y R_x^{\dagger} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Further, we have $[\mathbb{A}; \mathbf{X}, \mathbf{Y}] = Q[\mathbb{M}; \mathbf{X}', \mathbf{Y}']Q^*$. This is the situation in the n-dimensional state space.

On the practical side, everything we need takes place in the $(m+1)$-dimensional range of $\widehat{Q}$ and can be described as follows. Let $R_x = U_x \Sigma V^*$ be the economy-size SVD of R_x; Σ is $r \times r$, and V is $m \times r$, where $r = \mathrm{rank}(\mathbf{X}) = \mathrm{rank}(R_x)$. Note that then $\mathbf{X} = (\widehat{Q}U_x)\Sigma V^*$ is the SVD of $\mathbf{X}$, and that $\mathbf{H}\widehat{Q}R_x = \widehat{Q}R_y V V^*$. Hence

$$\mathbf{H}\widehat{Q}U_x\Sigma V^* = \widehat{Q}R_y V V^* \quad \text{and} \quad \mathbf{H}\widehat{Q}U_x\Sigma = \widehat{Q}R_y V. \tag{19}$$

We can truncate (19) at an index k (chopping of small singular values)

$$\mathbf{H}\widehat{Q}U_x(:, 1:k)\Sigma_k = \widehat{Q}R_y V_k,$$

Symmetric DMD - QR compression IV

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

Since $\mathbf{H} = \mathbf{H}^*$, the matrix $S_k = U_x(:,1:k)^*\widehat{Q}^*\mathbf{H}\widehat{Q}U_x(:,1:k)$ is also Hermitian.[2] S_k is the Rayleigh quotient of $\mathbf{H}$ with respect to the range of $\widehat{Q}U_x(:,1:k)$. If $S_kw_j = \lambda w_j$, $\|w_j\|_2 = 1$, then

$$(\lambda_j, \mathbf{z}_j), \quad \text{where} \quad \mathbf{z}_j = \widehat{Q}U_x(:,1:k)w_j = Q\begin{pmatrix} U_x(:,1:k)w_j \\ \mathbf{0} \end{pmatrix} \tag{20}$$

is the corresponding Ritz pair of $\mathbf{H}$. Note that $(\lambda_j, U_x(:,1:k)w_j)$ is a Ritz pair for $\widehat{Q}^*\mathbf{H}\widehat{Q}$ from the range of $U_x(:,1:k)$.
Hence, the QR compressed DMD first compresses the underlying $\mathbf{H}$ onto an $(m+1)$-dimensional subspace, computes the $(m+1)$-dimensional DMD using the projected data and then lifts the Ritz pairs back to the original n-dimensional state space (20). This lifting preserves the orthogonality of the Ritz vectors.

[2] Here one can formulate the Hermitian Procrustes problem and proceed as before.

The problem of non-normality

Problem of ill-conditioned modes

The DMD assumes that the Rayleigh quotient matrix is diagonalizable. In some cases the matrix is not diagonalizable or highly non-normal so that the Ritz vectors are badly conditioned.

Recently proposed Koopman-Schur Decomposition (KSD) solves the problem and uses orthonormal modes with the same funcitonality (e.g. forecasting) as the DMD. For more details see

- Z. Drmač, I. Mezić: A data driven Koopman-Schur decomposition for computational analysis of nonlinear dynamics. arXiv:2312.15837v1 [math.NA]

Measure preserving transformation = unitary operator

Consider a DDS $x^{(i+1)} = F(x^{(i)})$ where F is measure preserving on a probability space $(\Omega, \mathcal{B}, \omega)$ ($\omega(F^{-1}(\mathcal{S})) = \omega(\mathcal{S})$, $\mathcal{S} \in \mathcal{B}$).

The corresponding Koopman operator $\mathcal{K}f = f \circ F$ is an isometry on $L^2(\Omega, \omega)$; the inner product and the norm are $\langle \cdot, \cdot \rangle_\omega$, $\| \cdot \| = \sqrt{\langle \cdot, \cdot \rangle_\omega}$. ($\mathcal{K}$ has unitary extension, but those details are out of scope of this course.)

Goal: discretization that corresponds to isometry

When we compress $\mathcal{K}$ onto N-dimensional subspace $\mathcal{V}_N \subset L^2(\Omega, \omega)$, the corresponding $N \times N$ matrix K should represents a linear operator that preserves the inner product $\langle \cdot, \cdot \rangle_\omega$ in $\mathcal{V}_N$.

This is more difficult than preserving hermiticity. (E.g. a Rayleigh quotient of a Hermitian matrix is Hermitian, but in the unitary case this is not true.) We go back to square one and first review the process of building the matrix K, keeping in mind the above condition.

Matrix representation of an operator compression - a review

Consider a DDS $x^{(i+1)} = F(x^{(i)})$. Suppose we are given:

- Data $x^{(i)}$, $y^{(i)} = F(x^{(i)})$, $i = 1, \ldots, M$. (Direct numerical simulations and/or measurements.)
- Basis functions (or, a dictionary) of $\psi_1, \ldots, \psi_N$ that span an N-dimensional subspace $\mathcal{V}_N$ in the ambient Hilbert space $L^2(\Omega, \omega)$.

Goal: matrix K of the compression of the Koopman operator $\mathcal{K}$ to $\mathcal{V}_N$.

Take $g \in \mathcal{V}_N$. With fixed basis, g is identified with a vector $\mathbf{g}$ as follows:

$$g(x) = \sum_{j=1}^{N} g_j \psi_j(x) = \Psi(x)\mathbf{g}, \ \Psi(x) = \begin{pmatrix} \psi_1(x) & \ldots & \psi_N(x) \end{pmatrix}, \ \mathbf{g} = \begin{pmatrix} g_1 \\ \vdots \\ g_N \end{pmatrix}$$

$\mathcal{V}_N \equiv \mathbb{C}^N$; $g \equiv \mathbf{g}$. $\mathcal{K}g$ will be represented by $K\mathbf{g}$.

Matrix representation of a compression - a review

$$
\begin{aligned}
(\mathcal{K}g)(x) &= \sum_{j=1}^{N}(\mathcal{K}\psi_j)(x)g_j = \sum_{j=1}^{N}\psi_j(F(x))g_j \\
&= \underbrace{\Psi(x)K\mathbf{g}}_{\text{desired form}} + \underbrace{\left(\sum_{j=1}^{N}\psi_j(F(x))g_j - \Psi(x)K\mathbf{g}\right)}_{\text{residual } R(\mathbf{g};x)}
\end{aligned}
$$

The matrix K should be determined so that the residual is minimized. Given severe restriction of data driven scenario, the minimization will only mimic the proper construction of an operator compression in Hilbert spaces.

Set $\Psi(F(x)) = (\psi_1(F(x)), \ldots, \psi_N(F(x)))$.

Matrix representation of a compression - a review

$$R(\mathbf{g};x) = \sum_{j=1}^{N} \psi_j(F(x))g_j - \Psi(x)K\mathbf{g} = (\Psi(F(x)) - \Psi(x)K)\mathbf{g}$$

When defining K, it suffices to define over the sphere $\|\mathbf{g}\|_2 = 1$. Note that

$$\max_{\|\mathbf{g}\|_2=1} |R(\mathbf{g};x)| = \max_{\|\mathbf{g}\|_2=1} |(\Psi(F(x)) - \Psi(x)K)\mathbf{g}| = \|\Psi(F(x)) - \Psi(x)K\|_2.$$

It is desirable that

$$\int_\Omega |(\mathcal{K}g)(x) - \Psi(x)K\mathbf{g}|^2 d\omega(x) = \int_\Omega |R(\mathbf{g};x)|^2 d\omega(x).$$

is minimal. Note that

$$\int_\Omega |R(\mathbf{g};x)|^2 d\omega(x) \leq \int_\Omega \|\Psi(F(x)) - \Psi(x)K\|_2^2 d\omega(x).$$

Matrix representation of a compression - a review

In a data driven scenario, the integral is only approximated by a quadrature formula that can only use the available data snapshots, possibly with weights that improve the accuracy.

Hence, instead of the integral, we consider a weighted sum

$$\sum_{i=1}^{M} w_i \|\Psi(y^{(i)}) - \Psi(x^{(i)})K\|_2^2.$$

With the notation $W = \mathrm{diag}(w_1, \ldots, w_M)$ and

$$\Psi_X = \begin{pmatrix} \Psi(x^{(1)}) \\ \vdots \\ \Psi(x^{(m)}) \end{pmatrix}, \quad \Psi_Y = \begin{pmatrix} \Psi(y^{(1)}) \\ \vdots \\ \Psi(y^{(m)}) \end{pmatrix}, \quad (y^{(i)} = F(x^{(i)}))$$

the problem can be compactly written as

$$\|W^{1/2}(\Psi_Y - \Psi_X K)\|_F \longrightarrow \min_K$$

Matrix representation of a compression - a review

The solution matrix K is

$$K = (W^{1/2}\Psi_K)^\dagger (W^{1/2}\Psi_Y).$$

This can also be written in a normal equations form

$$K = G^\dagger A, \quad G = \Psi_X^* W \Psi_X, \quad A = \Psi_X^* W \Psi_Y.$$

$$G_{jk} \approx \langle \psi_k, \psi_j \rangle_\omega, \quad A_{jk} \approx \langle \mathcal{K}\psi_k, \psi_j \rangle_\omega.$$

Approaches to establish convergence:

- random sampling
- ergodic sampling (long trajectory, ergodic systems)
- good quadrature formulas

mpEDMD (M. Colbrook)

Now suppose that the compression K is constrained to correspond to a unitary operator - i.e. it has to preserve the inner product and the (induced) norm.

Consider two functions $g, h \in \mathcal{V}_N$, $g = \Psi\mathbf{g}$, $h = \Psi\mathbf{h}$.

$$\langle g, h\rangle_\omega = \langle \Psi\mathbf{g}, \psi\mathbf{h}\rangle_\omega = \langle \sum_{i=1}^{N} g_i\psi_i, \sum_{j=1}^{N} h_j\psi_j\rangle_\omega = \sum_{i=1}^{N}\sum_{j=1}^{n} g_i\overline{h}_j\langle\psi_i,\psi_j\rangle_\omega$$

$$\approx \sum_{i=1}^{N}\sum_{j=1}^{n} g_i\overline{h}_j G_{ji} = \mathbf{h}^* G\mathbf{g}$$

In particular, $\|g\|_\omega^2 = \|\Psi\mathbf{g}\|_\omega^2 \approx \mathbf{g}^* G\mathbf{g} = \|G^{1/2}\mathbf{g}\|_2^2$.

Since $\mathcal{K}g \approx \Psi K\mathbf{g}$ and $\|\Psi K\mathbf{g}\|_\omega^2 \approx \mathbf{g}^* K^* G K\mathbf{g}$, we can mimic $\|g\|_\omega = \|\mathcal{K}g\|_\omega$ by imposing the condition

$$K^* G K = G.$$

mpEDMD (Colbrook)

Consider the residual over the unit sphere $\|G^{1/2}\mathbf{g}\|_2 = 1$:

$$\begin{aligned}\max_{\|G^{1/2}\mathbf{g}\|_2=1} |R(\mathbf{g};x)| &= \max_{\|G^{1/2}\mathbf{g}\|_2=1} |(\Psi(F(x)) - \Psi(x)K)\mathbf{g}| \\ &= \|\Psi(F(x))G^{-1/2} - \Psi(x)KG^{-1/2}\|_2.\end{aligned}$$

As before, consider minimizing the discretized integral of the residual

$$\sum_{i=1}^{M} w_i \|\Psi(y^{(i)})G^{-1/2} - \Psi(x^{(i)})KG^{-1/2}\|_2^2,$$

under the constraint that $K^*GK = G$. Note that this means that $Q = G^{1/2}KG^{-1/2}$ is unitary and the objective is to minimize

$$\sum_{i=1}^{M} w_i \|\Psi(y^{(i)})G^{-1/2} - \Psi(x^{(i)})G^{-1/2}Q\|_2^2 \longrightarrow \min_{Q^*Q=I}.$$

mpEDMD (Colbrook)

Putting all together in a compact form yields a unitary Procrustes problem

$$\|W^{1/2}\Psi_Y G^{-1/2} - W^{1/2}\Psi_X G^{-1/2} Q\|_F \longrightarrow \min_{Q^*Q=I}.$$

This is of the form $\|A - BQ\|_F \to \min_{Q^*Q=I}$. The optimal Q is $Q = UV^*$, where $B^*A = U\Sigma V^*$ is the SVD. (See Appendix 2.) Once we compute Q, the matrix of the compressed unitary Koopman operator is

$$K = G^{-1/2} Q G^{1/2}.$$

For more details, theoretical analysis, examples and software see

- M. Colbrook: The mpEDMD Algorithm for Data-Driven Computations of Measure-Preserving Dynamical Systems. SIAM Journal on Numerical Analysis Vol. 61, Iss. 3 (2023).

Appendix: Review of the symmetric eigenvalue problem

Theorem

Let H be Hermitian with the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and the corresponding orthonormal eigenvectors $u_1, \ldots, u_n$, i.e. $Hu_i = \lambda_i u_i$, $i = 1, \ldots, n$ and $u_i^* u_j = \boldsymbol{\delta}_{ij}$ ($u_i \perp u_j$ for $i \neq j$). Then

$$\begin{aligned} \lambda_1 &= \max_{x \neq \mathbf{0}} \frac{x^* H x}{x^* x} = \max_{\|x\|_2 = 1} x^* H x = u_1^* H u_1 \\ \lambda_i &= \max_{\substack{\|x\|_2 = 1 \\ x \perp u_1, \ldots, u_{i-1}}} x^* H x = u_i^* H u_i, \quad i = 2, \ldots, n. \end{aligned}$$

Analogously, the eigenvalues are the minima of the constrained quadratic form $x^* H x$:

$$\begin{aligned} \lambda_n &= \min_{x \neq \mathbf{0}} \frac{x^* H x}{x^* x} = \min_{\|x\|_2 = 1} x^* H x = u_n^* H u_n \\ \lambda_i &= \min_{\substack{\|x\|_2 = 1 \\ x \perp u_{i+1}, \ldots, u_n}} x^* H x = u_i^* H u_i \quad i = 1, \ldots, n-1. \end{aligned}$$

Appendix: Review of the symmetric eigenvalue problem

Theorem (Poincaré's inequality)

Let $H = H^* \in \mathbb{C}^{n\times n}$ have the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. If $\mathcal{S} \subseteq \mathbb{C}^n$ is an arbitrary i–dimensional subspace, then for some unit vectors $x, y \in \mathcal{S}$

$$x^*Hx \leq \lambda_i, \quad y^*Hy \geq \lambda_{n-i+1}. \tag{1}$$

Theorem

The eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ of H are the optimal values of a sequence of constrained optimization problems:

$$\lambda_i = \max_{\substack{\mathcal{S}\subseteq\mathbb{C}^n \\ \dim(\mathcal{S})=i}} \min_{\substack{x\in\mathcal{S} \\ \|x\|_2=1}} x^*Hx \quad = \min_{\substack{\mathcal{S}\subseteq\mathbb{C}^n \\ \dim(\mathcal{S})=n-i+1}} \max_{\substack{x\in\mathcal{S} \\ \|x\|_2=1}} x^*Hx, \quad i = 1, \ldots, n, \tag{2}$$

where the optima are attained at the corresponding eigenvectors, $\lambda_i = u_i^*Hu_i$ ($Hu_i = \lambda_i u_i$, $u_i^*u_j = \boldsymbol{\delta}_{ij}$).

Appendix: Review of the symmetric eigenvalue problem

Theorem (Ky Fan's theorem)

The eigenvaluesa $\lambda_1 \geq \cdots \geq \lambda_n$ of H solve the following constrained trace optimization problem:

$$\lambda_1 + \cdots + \lambda_k = \max\{\mathrm{Trace}(X^* H X) : X \in \mathbb{C}^{n\times k},\ X^*X = \mathbb{I}_k\}, \qquad (3)$$

where the maximum is attained at the matrices of the form $X = \big(u_1, \ldots, u_k\big)\, Q$, where $u_1, \ldots, u_k$ are orthonormal eigenvectors of $\lambda_1, \ldots, \lambda_k$. and Q is arbitrary $k \times k$ unitary matrix. Analogously, for the k smallest eigenvalues

$$\sum_{i=n-k+1}^{n} \lambda_i = \min\{\mathrm{Trace}(X^* H X) : X \in \mathbb{C}^{n\times k},\ X^*X = \mathbb{I}_k\}. \qquad (4)$$

Appendix: Review of the symmetric eigenvalue problem

Theorem (Weyl's theorem)

Let H be $n \times n$ Hermitian and let δH be a Hermitian perturbation. Then the eigenvalues of H and $H + \delta H$ can be compared as follows:

$$\lambda_j(H) + \lambda_n(\delta H) \leq \lambda_j(H + \delta H) \leq \lambda_j(H) + \lambda_1(\delta H), \quad j = 1, \ldots, n, \tag{5}$$

or, equivalently,

$$\lambda_j(H + \delta H) - \lambda_1(\delta H) \leq \lambda_j(H) \leq \lambda_j(H + \delta H) - \lambda_n(\delta H), \quad j = 1, \ldots, n. \tag{6}$$

In particular,

$$\max_{j=1:n} |\lambda_j(H + \delta H) - \lambda_j(H)| \leq \max\{|\lambda_1(\delta H)|, |\lambda_n(\delta H)|\} = \|\delta H\|_2. \tag{7}$$

Further, if δH is positive semidefinite, then $\lambda_j(H + \delta H) \geq \lambda_j(H)$ for all $j = 1, \ldots, n$.

Appendix: Review of the symmetric eigenvalue problem

Theorem (Hoffman-Wielandt's theorem)

Let A and B be normal $n \times n$ matrices with the eigenvalues, respectively, $\lambda_1(A), \ldots, \lambda_n(A)$, and $\lambda_1(B), \ldots, \lambda_n(B)$. There is permutation p such that

$$\sqrt{\sum_{i=1}^{n} |\lambda_i(A) - \lambda_{p(i)}(B)|^2} \leq \|A - B\|_F. \tag{8}$$

Corollary

Let in the Hoffman-Wielandt' theorem the matrix A be Hermitian and let B be normal. If the eigenvalues are indexed so that $\lambda_1(A) \geq \cdots \geq \lambda_n(A)$, and $\Re(\lambda_1(B)) \geq \cdots \geq \Re(\lambda_n(B))$ then

$$\sqrt{\sum_{i=1}^{n} |\lambda_i(A) - \lambda_i(B)|^2} \leq \|A - B\|_F.$$

Appendix: Residual bounds for the symmetric eigenvalue problem

Theorem (Kahan's theorem)

Let H be $n \times n$ Hermitian matrix with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and let X be an $n \times \ell$ orthonormal matrix. If $\mu_1 \leq \cdots \leq \mu_\ell$ are the eigenvalues of $M = X^* H X$, then there are ℓ eigenvalues $\lambda_{i_1}, \ldots, \lambda_{i_\ell}$ of H such that

$$\max_{j=1:\ell} |\lambda_{i_j} - \mu_j| \leq \|R\|_2, \tag{9}$$

$$\sqrt{\sum_{j=1:\ell} |\lambda_{i_j} - \mu_j|^2} \leq \|R\|_F, \tag{10}$$

where $R = HX - XM$.

Appendix: Residual bounds for the SEVP

Proof

Let $X_\perp$ be an orthonormal matrix that spans $\mathcal{X}^\perp$ and let

$$H' = \begin{pmatrix} X & X_\perp \end{pmatrix}^* H \begin{pmatrix} X & X_\perp \end{pmatrix} = \begin{pmatrix} M & K^* \\ K & W \end{pmatrix}, \tag{11}$$

where $M = X^*HX$, $W = X_\perp^* H X_\perp$, $K = X_\perp^* H X$.
The trick is to us the backward error framework. With $\delta H = RX^* + XR^*$, $\mathcal{X}$ becomes an invariant subspace of $\widetilde{H} = H - \delta H$.

$$\widetilde{H}' = \begin{pmatrix} X & X_\perp \end{pmatrix}^* (H - \delta H) \begin{pmatrix} X & X_\perp \end{pmatrix} = \begin{pmatrix} M & \mathbf{0} \\ \mathbf{0} & W \end{pmatrix}. \tag{12}$$

The eigenvalues of M are some ℓ eigenvalues of $\widetilde{H}'$, and this reduces the problem to spectral perturbation theory, i.e. to comparing the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ of H' (unitarily similar to H) and the eigenvalues $\widetilde{\lambda}_1 \geq \cdots \geq \widetilde{\lambda}_n$ of $\widetilde{H}'$.

Appendix: Residual bounds for the SEVP

... proof ... continued ...

A direct application of the Weyl's and Wieland-Hoffman theorems yields

$$\max_{i=1:n} |\widetilde{\lambda}_i - \lambda_i| \leq \|H' - \widetilde{H}'\|_2, \quad \sqrt{\sum_{i=1}^{n} (\widetilde{\lambda}_i - \lambda_i)^2} \leq \|H' - \widetilde{H}'\|_F.$$

It only remains to compute the norm of the perturbation $H' - \widetilde{H}'$.

$$H' - \widetilde{H}' = \begin{pmatrix} \mathbf{0} & K^* \\ K & \mathbf{0} \end{pmatrix}, \quad \|H' - \widetilde{H}'\|_2 = \|K\|_2, \quad \|H' - \widetilde{H}'\|_F = \sqrt{2}\|K\|_F.$$

Due to unitary invariance, the spectral norm of K equals

$$\|K\|_2 = \|X_\perp X_\perp^* H X\|_2 = \|(\mathbb{I}_n - XX^*)HX\|_2 = \|HX - XM\|_2 = \|R\|_2,$$

and, in the same way, $\|K\|_F = \|R\|_F$. Since $\mu_j = \lambda_{i_j}$ for some indices $i_1, \ldots, i_\ell$, the proof of (9) is completed, and for (10) the extra factor $\sqrt{2}$ must be removed using another approach. Note that analogous error estimates hold for the eigenvalues of W.

Appendix: Orthogonal Procrustes's problem

Orthogonal Procrustes problem

For $A, B \in \mathbb{C}^{m\times n}$ find a unitary matrix Q that minimizes

$$\min_{Q^*Q=\mathbb{I}_n} \|A - BQ\|_F. \tag{1}$$

If A and B are real, the optimal Q should be real orthogonal.

This problem arises in applications e.g. in computer vision, robotics, photogrammetry, psychometrics, radiostereometric and morphometric analysis in biomedical engineering.

Pioneering work on the solution of this problem was done by Green 1952 and Schöneman 1966.

In the applications of DMD/Koopman operator for numerical analysis of nonlinear dynamics, (1) is the key for measure preserving/unitary cases.

Appendix: Orthogonal Procrustes's problem

Theorem

A solution of the problem (1) is

$$UV^* \in \arg \min_{Q^*Q=\mathbb{I}_n} \|A - BQ\|_F, \tag{2}$$

where $B^*A = U\Sigma V^*$ is the SVD of B^*A. This solution is unique if and only if B^*A is nonsingular. If A and B are real, then the optimal Q is real orthogonal.

Proof

Since

$$\|A - BQ\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2\Re\mathrm{Trace}(Q^*B^*A),$$

any optimal Q that minimizes (1) maximizes $\Re\mathrm{Trace}(Q^*B^*A)$.

Appendix: Orthogonal Procrustes's problem

... proof ... continued ...

If $B^*A = U\Sigma V^*$ is the SVD of B^*A, $\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^n$, then

$$\mathrm{Trace}(Q^*B^*A) = \mathrm{Trace}(Q^*U\Sigma V^*) = \mathrm{Trace}(V^*Q^*U\Sigma) = \sum_{i=1}^{n} (V^*Q^*U)_{ii}\sigma_i,$$

and thus (since $|\Re(V^*Q^*U)_{ii}| \leq |(V^*Q^*U)_{ii}| \leq 1$ for all i)

$$\Re\mathrm{Trace}(Q^*B^*A) = \sum_{i=1}^{n} \Re(V^*Q^*U)_{ii}\sigma_i \leq \sum_{i=1}^{n} \sigma_i. \tag{3}$$

The above inequality becomes an equality if $\Re(V^*Q^*U)_{ii} = 1$ for all $i = 1, \ldots, n$. If all σ_i's are positive (i.e. B^*A is nonsingular) then this condition is also necessary ($\sum_{i=1}^{n} \sigma_i(1 - \Re(V^*Q^*U)_{ii}) = 0 \Longleftrightarrow$ $\Re(V^*Q^*U)_{11} = \cdots = \Re(V^*Q^*U)_{nn} = 1$). Since V^*Q^*U is unitary, this is possible only if $V^*Q^*U = \mathbb{I}_n$, i.e. $Q = UV^*$.

References I

- Baddoo P. J., Herrmann B., McKeon B. J., Kutz J. N., and Brunton S. L. Physics-informed dynamic mode decomposition (piDMD). *arXiv:2112.04307*.
- M. Colbrook. The mpEDMD Algorithm for Data-Driven Computations of Measure-Preserving Dynamical Systems. *SIAM Journal on Numerical Analysis Vol. 61, Iss. 3 (2023).*
- M. Colbrook. The Multiverse of Dynamic Mode Decomposition Algorithms. *arXiv:2312.00137v2 [math.DS]. December 2023.*
- Z. Drmač, I. Mezić, and R. Mohr. Data driven modal decompositions: analysis and enhancements. *SIAM Journal on Scientific Computing*, 40(4):A2253–A2285, 2018.

References II


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, 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, A LAPACK implementation of the Dynamic Mode Decomposition I., LAPACK Working Note 298, 2022. (ACM Transactions on Mathematical Software Volume 50 Issue 1 Article No.: 1 pp 1-32)

References III

- Z. Drmač, A LAPACK implementation of the Dynamic Mode Decomposition II., LAPACK Working Note 300, 2022. (Hermitian Dynamic Mode Decomposition – Numerical Analysis and Software Solution, ACM Transactions on Mathematical Software Volume 50 Issue 1 Article No.: 2 pp 1-23)
- Higham N. J. The symmetric procrustes problem. *BIT*, 28 (1988), 133-143.
- Jovanović M. R., Schmid P. J., and Nichols J. W. Sparsity-promoting dynamic mode decomposition. Physics of Fluids 26, 2 (Feb. 2014), 024103
- Schmid P. J. Dynamic mode decomposition of numerical and experimental data. Journal of Fluid Mechanics 656 (10 August 2010), 5-28.

References IV


Schmid P. J. Data-driven and operator-based tools for the analysis of turbulent flows. Advanced Approaches in Turbulence, Durbin Paul (Ed.). Elsevier, 243-305.


Stewart G. W. and Sun Ji-G. Matrix Perturbation Theory Academic Press 1990.


Tu J. H., Rowley C. W., Luchtenburg D. M., Brunton S. L., and Kutz J. N. On dynamic mode decomposition: Theory and applications. Journal of Computational Dynamics 1, 2 (2014), 391-421.


Williams M. O., Kevrekidis I. G., and Rowley C. W A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. Journal of Nonlinear Science 25, 6 (June 2015), 1307-1346.

Problem: Learning equations from data

Now assume that the system $\dot{x}(t) = \mathbf{F}(x(t))$ is accessible only through snapshots from a sequence of trajectories with different (possibly unknown) initial conditions. More precisely, we are given

$$(\mathbf{x}_k, \mathbf{y}_k) \in \mathbb{R}^n \times \mathbb{R}^n, \ k = 1, \ldots, K,$$

where

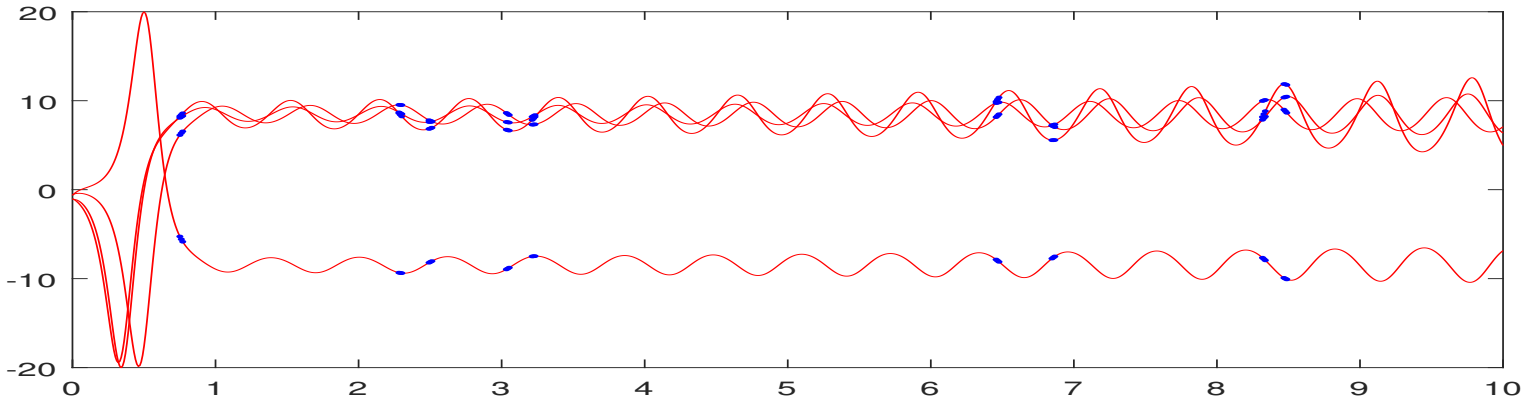
$$\mathbf{y}_k = \varphi^t(\mathbf{x}_k) \tag{1}$$

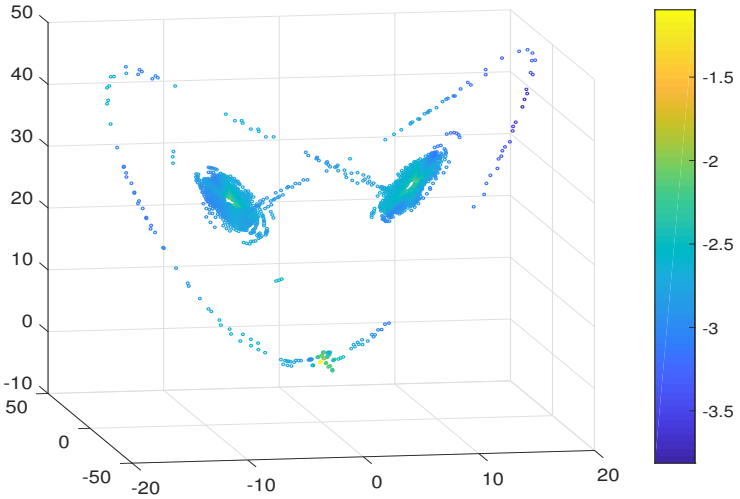
In a real application, t is a fixed time step, and it is possible that the time resolution precludes any approach based on estimating the derivatives by finite differences; the dataset could also be scarce, sparsely collected from several trajectories/short bursts of the dynamics under study.

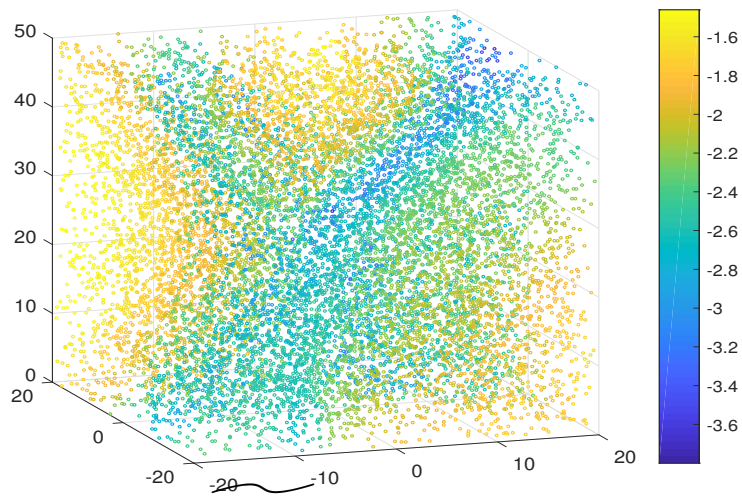
The task is to identify $\mathbf{F}$ and express it analytically, using a suitably chosen class of functions. Our focus is on the computing engine – robust numerical method that translates into reliable numerical software.

Example: Learn $\dot{x}(t) = \mathbf{F}(x(t))$ from data snapshots

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & 0 \\ 0 & 0 & -8/3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ -x_1 x_3 \\ x_1 x_2 \end{pmatrix}. \quad (2)$$







$\log_{10} \epsilon_k$, where $\epsilon_k = \max_{i=1,2,3} \frac{|F_i(\mathbf{x}_k) - F_i(\mathbf{x}_k)|}{\|\mathbf{F}(\mathbf{x}_k)\|_\infty}$

Mauroy and Goncalves: learn the semigroup generator

Mauroy and Goncalves method for learning $\mathbf{F}$ from the data. Step 1:

- Let $\mathbf{X} \subset \mathbb{R}^n$ be compact, forward invariant – *big enough* to contain all data snapshots. (*What happens in* $\mathbf{X}$ *stays in* $\mathbf{X}$.)
- Consider the semigroup $\mathcal{K}^t f = f \circ \varphi^t$ of Koopman operators acting on a space of scalar observables $f \in \mathcal{F}$, where e.g. $\mathcal{F} = L^2(\mathbf{X})$.
- Select a suitable (finite) N-dimensional but rich enough subspace $\mathcal{F}_N \subset \mathcal{F}$, and its basis $\mathcal{B} = \{\wp_1, \ldots, \wp_N\}$. The observables are $(O_X)_{ij} = \wp_j(\mathbf{x}_i) \in \mathbb{C}^{K \times N}$, $(O_Y)_{ij} = \wp_j(\mathbf{y}_i) \in \mathbb{C}^{K \times N}$.
- Compute (a data driven) compression $\Phi_N \mathcal{K}^t_{|\mathcal{F}_N} : \mathcal{F}_N \longrightarrow \mathcal{F}_N$ and its matrix representation U_N (in the basis $\mathcal{B}$)
- Show that $U_N \approx e^{L_N t}$, i.e. $L_N \approx (1/t) \log U_N$, where L_N is the compression of the infinitesimal generator $\mathbb{K}$ defined by

$$\mathbb{K} f = \lim_{t \to 0^+} \frac{\mathcal{K}^t f - f}{t}, \qquad f \in \mathcal{D}(\mathbb{K}).$$

($\mathcal{K}^t$ strongly continuous in $L^2(\mathbf{X})$: $\lim_{t \to 0^+} \|\mathcal{K}^t f - f\|_2 = 0$).

Mauroy and Goncalves: learn the semigroup generator

Mauroy and Goncalves method for learning $\mathbf{F}$ from the data. Step 2:

- Recall the fact that

$$\mathbb{K}f = \mathbf{F} \cdot \nabla f = \sum_{i=1}^{n} F_i \frac{\partial f}{\partial x_i}, \qquad f \in \mathcal{D}(\mathbb{K}). \tag{3}$$

- If we assume $F_i = \sum_k \phi_{ki} \wp_k$, then the action of $\mathbb{K}$ to the basis's vectors $\wp_k$ can be computed, using (3), by straightforward calculus, and its matrix representation will, by comparison with $(1/t) \log U_N$, reveal the coefficients ϕ_{ki}.
- An analysis of convergence (with probability one as $t \to 0$, $N \to \infty$, $K \to \infty$) and numerical experiments provided by Mauroy and Gonsalves show that this approach works well.

Example

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} \sum_{k=1}^{N_F} \phi_{k1} x_1^{s_1^{(k)}} x_2^{s_2^{(k)}} \cdots x_n^{s_n^{(k)}} \\ \vdots \\ \sum_{k=1}^{N_F} \phi_{kn} x_1^{s_1^{(k)}} x_2^{s_2^{(k)}} \cdots x_n^{s_n^{(k)}} \end{pmatrix} = \begin{pmatrix} F_1(\mathbf{x}) \\ \vdots \\ F_n(\mathbf{x}) \end{pmatrix}, \quad F_j(\mathbf{x}) = \sum_{k=1}^{N_F} \phi_{kj} \mathbf{x}^{\mathbf{s}^{(k)}}, \tag{4}$$

where $\mathbf{x}^{\mathbf{s}^{(k)}} = x_1^{s_1^{(k)}} x_2^{s_2^{(k)}} \cdots x_n^{s_n^{(k)}}$ are monomials written in multi-index notation and have total degree of at most m_F.

$$\mathcal{F}_N = \mathrm{span}(\mathcal{B}), \mathcal{B} = \{x_1^{s_1} \cdots x_n^{s_n} \ : \ s_i \in \mathbb{N}_0, \ s_1 + \ldots + s_n \leq m\}, \ N = \binom{n+m}{n} \tag{5}$$

Let ℓ be the index of x_j in the *grlex* ordering, i.e. $\wp_\ell(\mathbf{x}) = x_j$; $\ell = n + 2 - j$. Then the application of $\mathbb{K}$ to $\wp_\ell$ reads

$$(\mathbb{K}\wp_\ell)(\mathbf{x}) = (\mathbf{F} \cdot \nabla \wp_\ell)(\mathbf{x}) = \sum_{i=1}^{n} F_i(\mathbf{x}) \frac{\partial}{\partial x_i} \wp_\ell(\mathbf{x}) = F_j(\mathbf{x}) \equiv F_{n+2-\ell}(\mathbf{x}).$$

Example

Hence $\mathbb{K}\wp_\ell = F_{n+2-\ell}$. If $L_N = \Phi_N \mathbb{K}_{|\mathcal{F}_N}$, then also $L_N \wp_\ell = F_j$ ($F_j(\mathbf{x}) = \sum_{k=1}^{N_F} \phi_{kj} \mathbf{x}^{\mathbf{s}(k)} \in \mathcal{F}_N$). Hence, in the basis $\mathcal{B}$ we have

$$[L_N]_{\mathcal{B}}(:,\ell) = [\Phi_N \mathbb{K} \wp_\ell]_{\mathcal{B}} = [F_j]_{\mathcal{B}} = \begin{pmatrix} \phi_{1j} \\ \phi_{2j} \\ \vdots \\ \phi_{N_F j} \\ \mathbf{0}_{N-N_F} \end{pmatrix}, \ j = n+2-\ell, \ \ell = 2, \ldots, n+1.$$

In other words, the coordinates of F_j are encoded in $[L_N]_{\mathcal{B}}(:, n+2-j)$.

Convergence theory (Mauroy & Goncalves)

$$[L_N]_{\mathcal{B}} = \lim_{t \to 0^+} \frac{1}{t} [\log \Phi_N \mathsf{U}^t_{|\mathcal{F}_N}]_{\mathcal{B}} = \lim_{t \to 0^+} \frac{1}{t} \log [\Phi_N \mathsf{U}^t_{|\mathcal{F}_N}]_{\mathcal{B}}$$

and it follows that, for t small enough,

$$[L_N]_{\mathcal{B}} \approx \frac{1}{t} \log U_N \equiv \frac{1}{t} \log O_X^\dagger O_Y. \tag{6}$$

Koopman semigroup generator - data driven identification

The method (Mauroy and Gonsalves, 2018)

1. Compress $\mathcal{K}^t$ onto a suitable finite dimensional but rich enough subspace $\mathcal{F}_N$ of $\mathcal{F}$ is computed. In a convenient basis $\mathcal{B}$ of $\mathcal{F}_N$, this compression is executed in the discrete least squares setting, yielding the matrix representation $U_N = [\Phi_N \mathcal{K}^t_{|\mathcal{F}_N}]_{\mathcal{B}} \in \mathbb{R}^{n\times n}$. For example, $U_N = O_X^\dagger O_Y$, where O_X, O_Y are the observables.
2. $[L_N]_{\mathcal{B}} \approx \frac{1}{t}[\log \Phi_N \mathcal{K}^t_{|\mathcal{F}_N}]_{\mathcal{B}} = \frac{1}{t}\log[\Phi_N \mathcal{K}^t_{|\mathcal{F}_N}]_{\mathcal{B}} = \frac{1}{t}\log U_N$
3. Recall, $\mathbb{K} = F\cdot\nabla = \sum_{i=1}^n F_i \frac{\partial}{\partial x_i}$.
4. Assume polynomial field, $F_j(x) = \sum_{k=1}^{N_F} \phi_{kj}\mathbf{x}^{\mathbf{s}(k)}$
5. $[L_N]_{\mathcal{B}}(:,\ell) = [L_N \wp_\ell]_{\mathcal{B}} = [F_j]_{\mathcal{B}} = \begin{pmatrix}\phi_{1j} & \phi_{2j} & \dots & \phi_{N_F j} & \mathbf{0}_{N-N_F}\end{pmatrix}^T$, where $j = n+2-\ell,\ \ \ell = 2,\dots,n+1$.

The problem: Numerical stability issue in $\log U_N \equiv \log(O_X^\dagger O_Y)$

Matrix logarithm

For this scheme to work, U_N must be nonsingular, otherwise $\log U_N$ does not exist. Further, to have the primary value of the logarithm (as primary matrix function, i.e. the same branch of the logarithm used in all Jordan blocks), the matrix must not have any real negative eigenvalues. Only under those conditions we can obtain real logarithm as primary function.

Theorem

Let A be real nonsingular matrix. Then A has real logarithm if and only if A has an even number of Jordan blocks of each size for every negative eigenvalue.

Theorem

Suppose that $n \times n$ complex A has no eigenvalue on $(-\infty, 0]$. Then a unique logarithm of A can be defined with eigenvalues in the strip $\{z \in \mathbb{C} : -\pi < \Im(z) < \pi\}$. It is called the principal logarithm and denoted by $\log A$. If A is real, then its principal logarithm is real as well.

An Example

A good way to test robustness of a numerical algorithm is to push it to its limits. In this case, we choose a difficult test case and let the dimensions of the data matrices grow by increasing the total degree m of the polynomial basis (and thus the dimension N) and matching that with increased K so that $K > N$. The main goal is to provide a case study example.

Consider the Lorenz system

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & 0 \\ 0 & 0 & -8/3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ -x_1 x_3 \\ x_1 x_2 \end{pmatrix}. \tag{7}$$

The exact coefficients, ordered to match the *grlex* ordering of the monomial basis are

	1	x_3	x_2	x_1	x_3^2	x_2x_3	x_2^2	x_1x_3	x_1x_2	x_1^2
F_1:	0	0	**1.0000e+1**	**−1.0000e+1**	0	0	0	0	0	0
F_2:	0	0	**−1.0000e+0**	**2.8000e+1**	0	0	0	**−1.0000e+0**	0	0
F_3:	0	**−2.6667e+0**	0	0	0	0	0	0	**1.0000e+0**	0

.

An Example

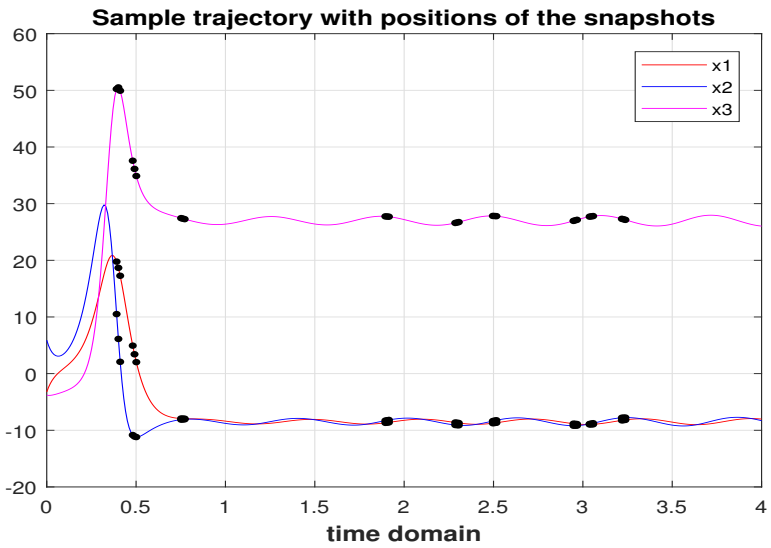
To collect data, we ran simulations with 55 random initial conditions and from each trajectory we randomly (independently) selected 55 points, giving the total of $K = 3025$ pairs $(\mathbf{x}_k, \mathbf{y}_k)$. The simulations were performed in Matlab, using the `ode45()` solver in the time interval $[0, 0.2]$ with the time step $\delta t = 10^{-3}$. We computed the logarithm in Matlab in two ways, as $\texttt{logm}(\texttt{pinv}(O_X) * O_Y)$ and as $\texttt{logm}(O_X \backslash O_Y)$, [3] and obtained nearly the same matrix. The computed approximations of the coefficients of (7), with $m = 3$, $N = 20$ and $m_F = 2$, are

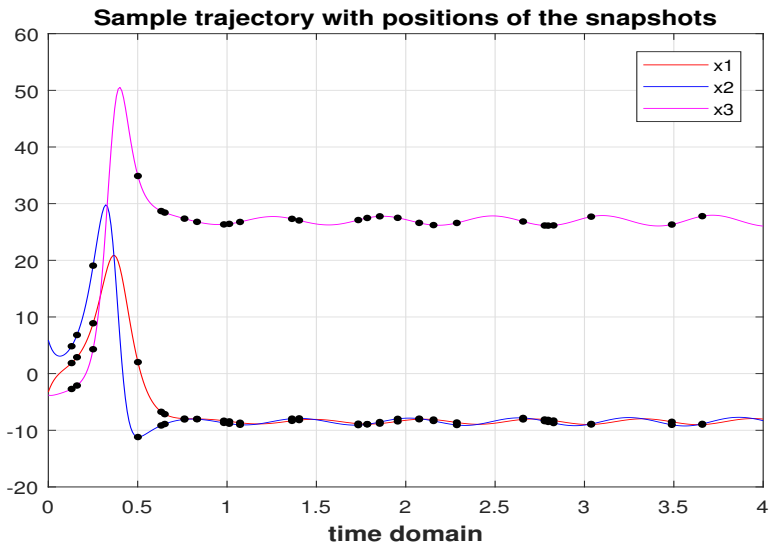
	1	x_3	x_2	x_1	x_3^2	x_2x_3	x_2^2	x_1x_3	x_1x_2	x_1^2
F_1:	$e{-5}$	$e{-6}$	**1.0000e1**	**−1.0000e1**	$e{-7}$	$e{-6}$	$e{-7}$	$e{-6}$	$e{-6}$	$e{-7}$
F_2:	$e{-5}$	$e{-6}$	**−1.0000e0**	**2.8000e1**	$e{-6}$	$e{-5}$	$e{-9}$	**−1.0001e0**	$e{-6}$	$e{-6}$
F_3:	$e{-4}$	**−2.6667e0**	$-5.5e{-6}$	$e{-6}$	$e{-6}$	$e{-6}$	$e{-8}$	$e{-5}$	**1.0000e0**	$e{-6}$

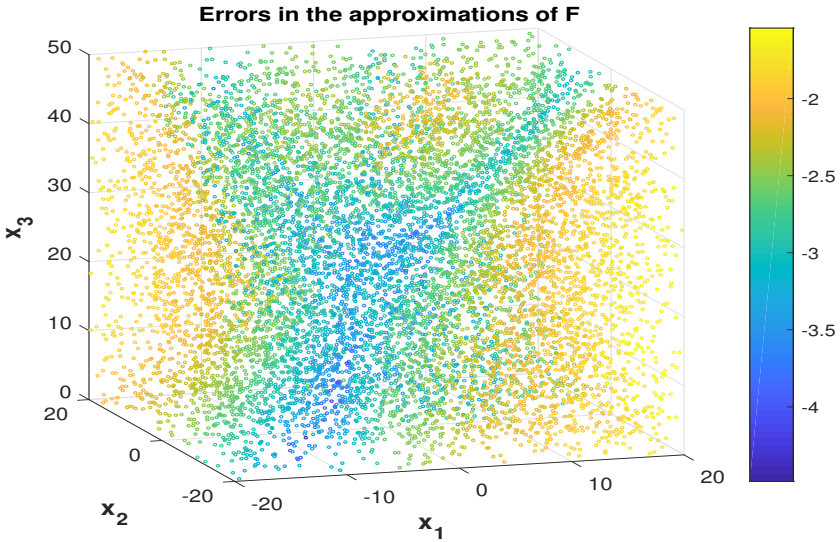
.

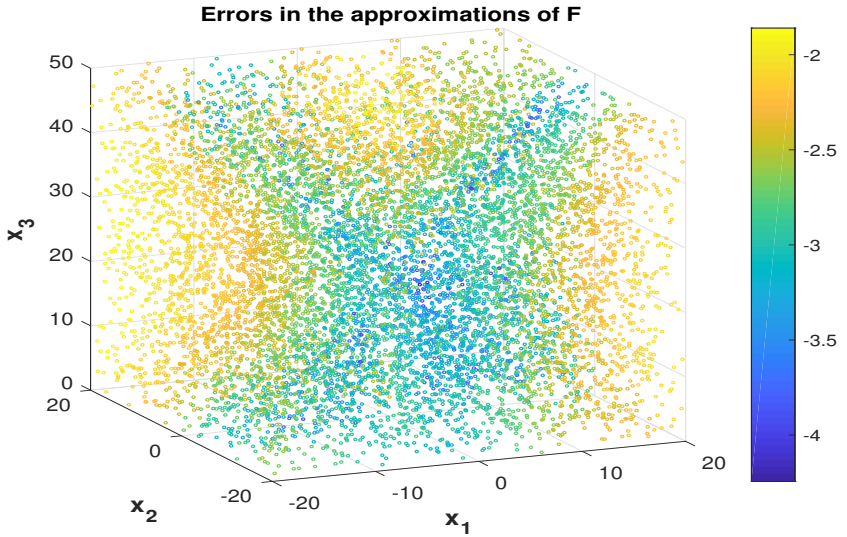
[3] Of course, using the pseudoinverse explicitly is not recommended. We use it here for illustrative purposes only.

$m = 3$; $\epsilon_k = \max_{i=1,2,3} |F_i(\mathbf{x}_k) - F_i(\mathbf{x}_k)| / \|\mathbf{F}(\mathbf{x}_k)\|_\infty$









$\log_{10} \epsilon_k$ using 150×27 samples.

An Example

Now we use the data snapshots from the previous example, and increase the total degree to $m = 9$, thus increasing N from $N = 20$ to $N = 220$. Recall that $K = 3025$. Surprisingly, the computed coefficients are all complex, and are completely off; the euclidean norms of the real and the imaginary parts of the vector of the computed coefficients are $O(10^6)$.

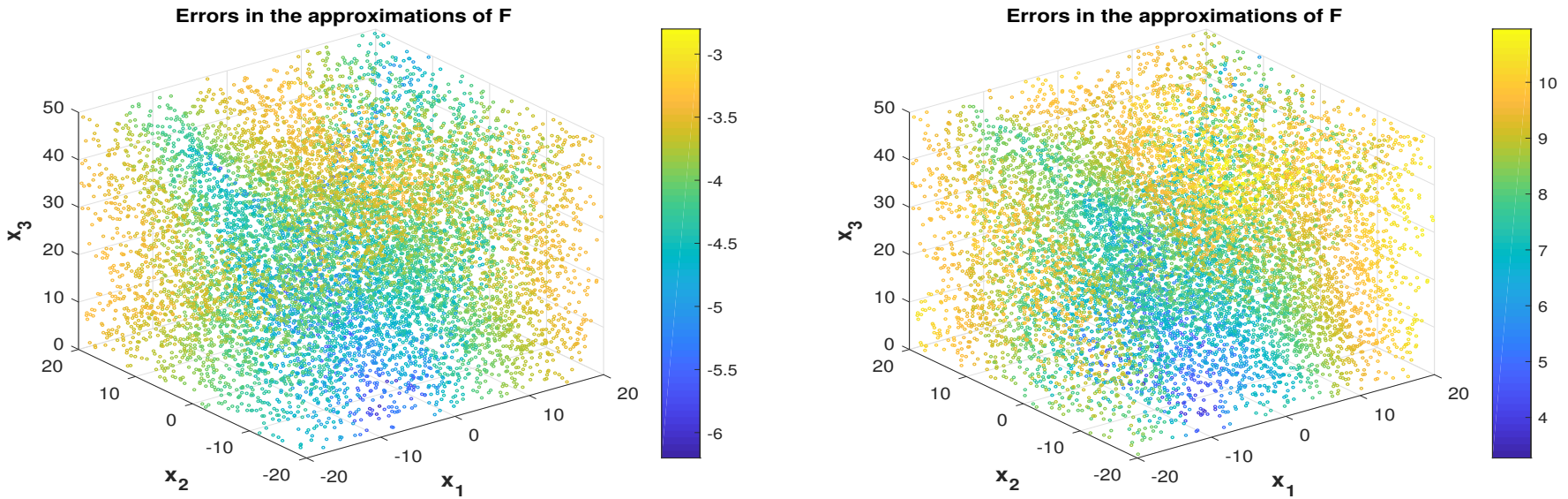


Figure: $\log_{10} \epsilon_k$. *Left panel*: $m = 3$. *Right panel*: $m = 9$.

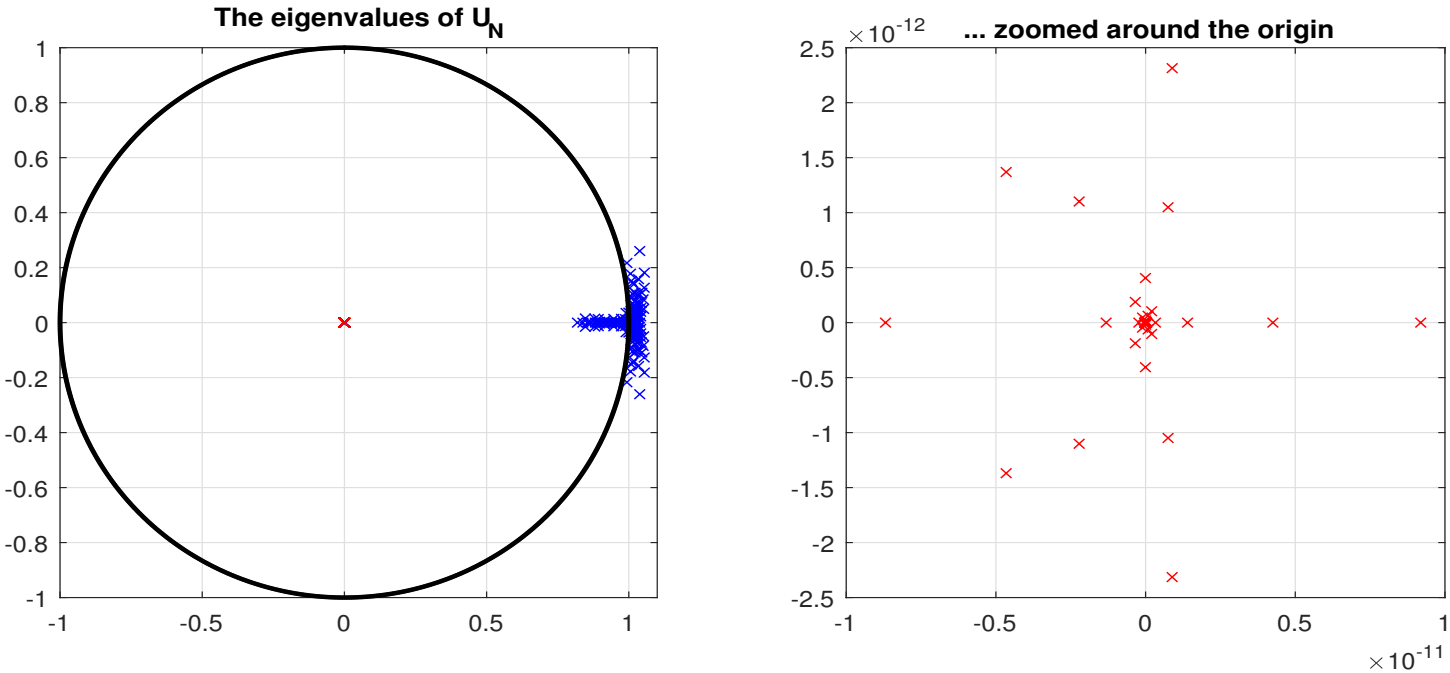
$\log U_N \equiv \log(O_X^\dagger O_Y)$ is difficult

- Computing the matrix logarithm becomes difficult, in particular when the dimensions increase (for better approximation).
- Hence, the better the method theoretically, the more numerically unstable it becomes for practical computation.
- Numerical implementation (available in the literature) fails even when using state of the art tools (Matlab). Often it works only for small time intervals with high resolution sampling.
- However, the approach is appealing as it does not use the derivatives and it is better suited for real applications where the sampling may not be fine enough for sufficiently accurate approximations of the derivatives.

```
Warning:  Principal matrix logarithm is not defined for A
with nonpositive real eigenvalues.  A non-principal matrix
logarithm is returned.
```

$\log U_N \equiv \log(O_X^\dagger O_Y)$ is difficult to compute

A closer inspection of the eigenvalues of U_N confirms that U_N has problematic (real negative) eigenvalues.



($m = 9,\ N = 220.$) *Left panel*: The (computed) eigenvalues of the matrix representation of the computed compression $U_N = \texttt{pinv}(O_X) * O_Y$ of $\mathcal{K}^t$. The red cross at the origin indicates a cluster of eigenvalues. *Right panel*: Zoomed neighborhood of the origin, showing many absolutely small eigenvalues, quite a few of whom are negative real. $O_X \backslash O_Y$ even worse.

Even $O_X^\dagger O_Y$ may be difficult to compute

Using `pinv()` is not advisable. What if we use direct LS solver (Matlab's `backslash`). One conspicuous difference is that instead of the cluster of absolutely small eigenvalues of $\texttt{pinv}(O_X) * O_Y$, $O_X \backslash O_Y$ has zero eigenvalue of multiplicity 56. This multiple zero eigenvalue is a consequence of the sparsity structure of $O_X \backslash O_Y$.

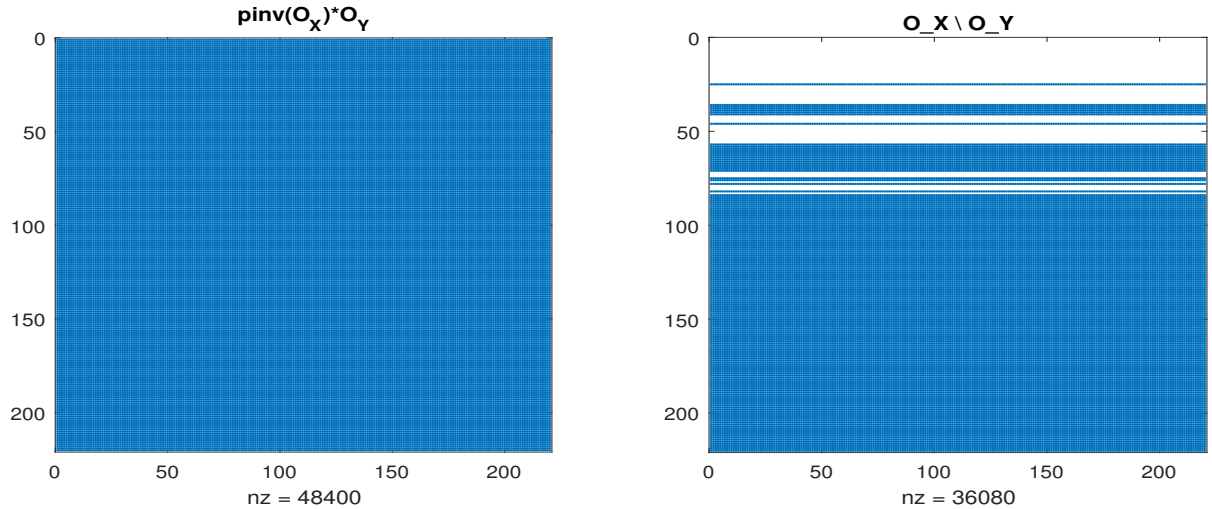


Figure: The sparsity structure of $\texttt{pinv}(O_X) * O_Y$ and $O_X \backslash O_Y$. The backslash operator uses the rank revealing (column pivoted) QR factorization and, by truncation, returns sparse rank deficient solution. As a result, computation of the matrix logarithm fails.

Even $O_X^\dagger O_Y$ may be difficult to compute

In this example we use monomials and the first basis vector is the constant. Since $\mathcal{K}^t \wp_1 = 1 \cdot \wp_1$, $[\Phi_N \mathcal{K}^t_{|\mathcal{F}_N}]_{\mathcal{B}} [\wp_1]_{\mathcal{B}} \equiv U_N e_1 = e_1 = 1 \cdot [\wp_1]_{\mathcal{B}}$, which clearly follows from the solution of the least squares problem. On the other hand, the first column of U_N is computed as shown in the Figure.

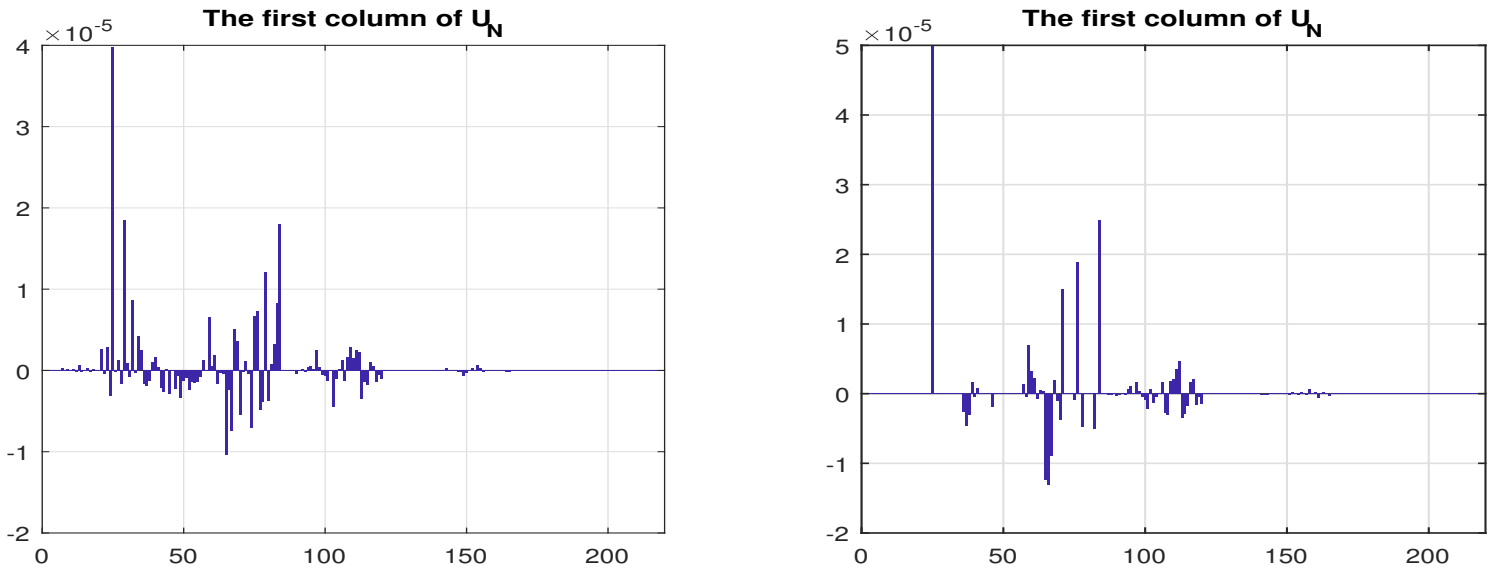
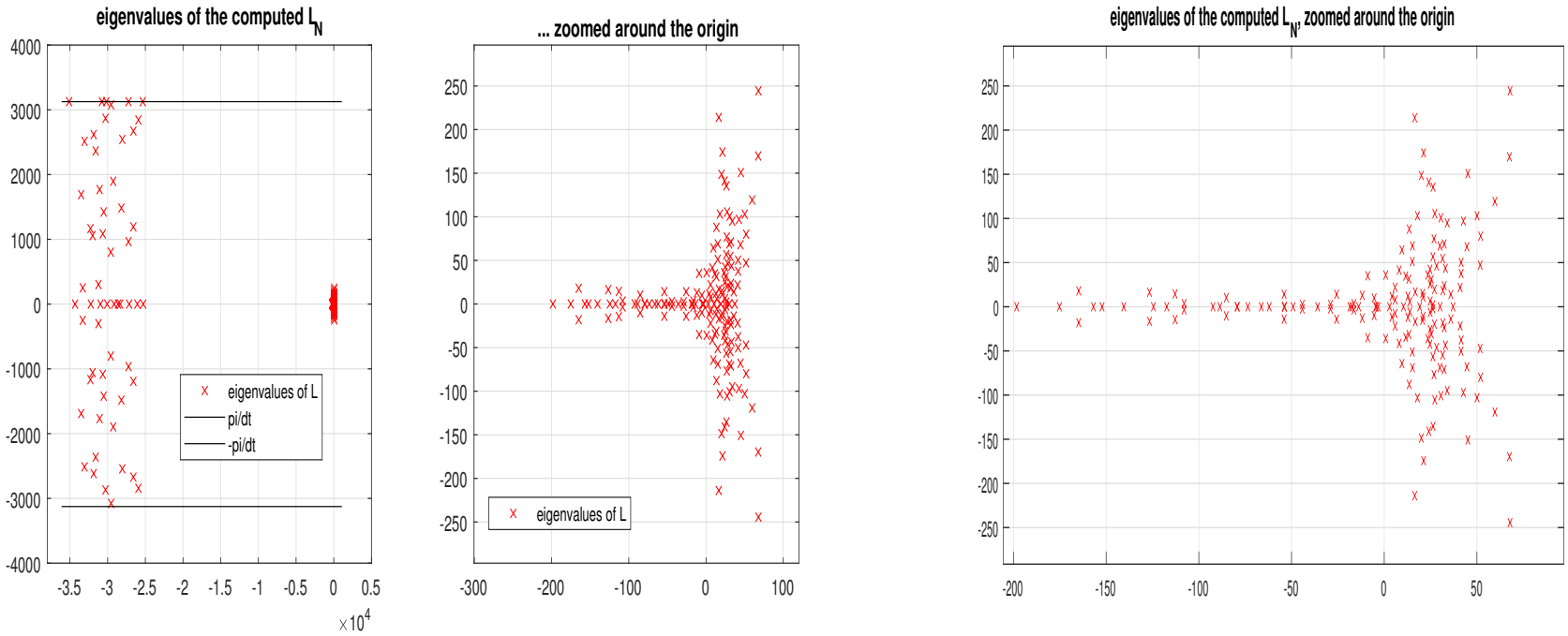


Figure: ($m = 9$.) Left: the first column of U_N, computed in Matlab as $\texttt{pinv}(O_X) * O_Y$. Its norm is $\|U_N(:,1)\|_2 \approx 5.7007e-05$. Right: the first column of $U_N = O_X \backslash O_Y$, with norm $\|U_N(:,1)\|_2 \approx 6.6258e-05$. The true value of $U_N(:,1)$ should be $e_1 = (1, 0, \ldots, 0)^T$.

$\log U_N \equiv \log(O_X^\dagger O_Y)$ is difficult



(Lorenz, $m = 9$.) The (computed) eigenvalues of the matrix L_N. Note that some of them are at the boundary of the strip $\{z \in \mathbb{C} \ : \ -\pi/\delta t < \Im(z) < \pi/\delta t\}$, i.e. the eigenvalues of $\log U_N$ are at the boundary of $\{z \in \mathbb{C} \ : \ -\pi < \Im(z) < \pi\}$. The right panel shows te distribution of the eigenvalues closer to the origin.

Computing $\log U_N$ with preconditioning

If S is any nonsingular matrix, then

$$\log(O_X^\dagger O_Y) = S \log(S^{-1}(O_X^\dagger O_Y)S)S^{-1} \equiv S \log((O_X S)^\dagger O_Y S)S^{-1}. \quad (8)$$

Note that replacing $O_X^\dagger O_Y$ with the similar matrix $S^{-1}(O_X^\dagger O_Y)S$ corresponds to changing the basis for matrix representation of the compressed Koopman operator. Clearly, the key is to compute the preconditioned matrix $S^{-1}(O_X^\dagger O_Y)S$ without first computing $O_X^\dagger O_Y$. (Once we compute and store $O_X^\dagger O_Y$ explicitly in floating point arithmetic, it may be then too late even for exact computation.)

The conditions on S are:

(i) it should facilitate more accurate computation of the argument $S^{-1}(O_X^\dagger O_Y)S = (O_X S)^\dagger O_Y S$ for the matrix logarithm;

(ii) it should have preconditioning effect for computing the logarithm of $S^{-1}(O_X^\dagger O_Y)S$;

(iii) the application of S and S^{-1} should be numerically efficient.

Simple diagonal preconditioning

Algorithm: $[L_N] = \text{Inf\_Generator\_QRSC}(O_X, O_Y, T, N)$

Input: O_X, O_Y, T

1: $S = \text{diag}(1/\|O_X(:,i)\|_2)$
2: $[Q_X, \widehat{R}_X] = \text{qr}(O_X S)$ {*QR factorization*}
3: $\widehat{U}_N = Q_X^T (O_Y S) \widehat{R}_X^{-1}$ {$\widehat{U}_N$ *is similar to* $O_X^\dagger O_Y$.}
4: $\widehat{L}_N = \log(\widehat{U}_N)$
5: $L_N = (1/T) S (\widehat{R}_X^{-1} \widehat{L}_N \widehat{R}_X) S^{-1}$

Output: $L = (1/T)\log(O_X^\dagger O_Y)$

- Simple to deploy.
- It is a simple preconditioner for LS solvers used to compute $O_X^\dagger O_Y$.
- Useful in many cases but of limited use in more difficult cases.
- Must be careful when scaling noisy data.

Preconditioning using RR QR factorization

$[L_N] = \text{Inf\_Generator\_QRCP}(O_X, O_Y, T, N)$

Input: O_X, O_Y, T

1: Reorder the snapshots by simultaneous row permutation of O_X and O_Y; see the remark below.

2: $[Q_X, R_X, \Pi_X] = \text{qr}(O_X)$ {*Rank revealing QR factorization*}

3: $\widehat{U}_N = Q_X^T (O_Y \Pi_X) R_X^{-1}$ {$\widehat{U}_N$ *is similar to* $O_X^\dagger O_Y$.}

4: $\widehat{L}_N = \log(\widehat{U}_N)$

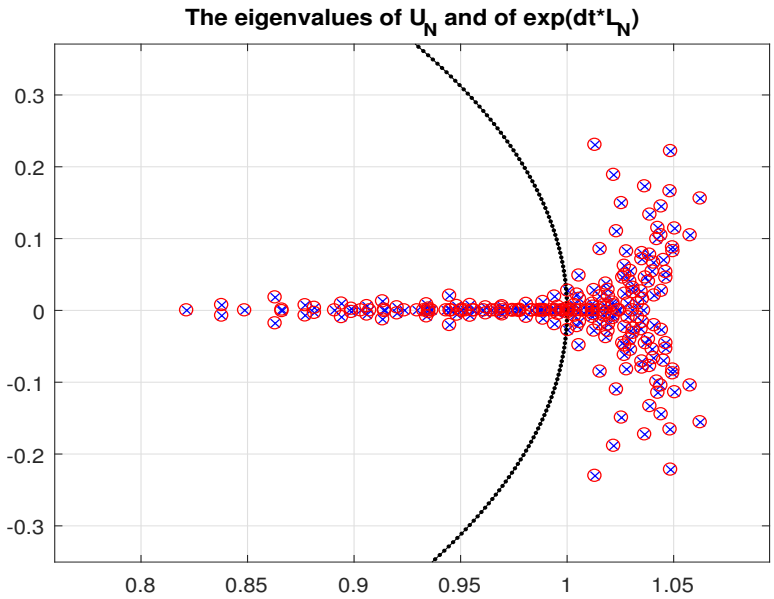
5: $L_N = (1/T)\Pi_X (R_X^{-1} \widehat{L}_N R_X) \Pi_X^T$

Output: $L = (1/T)\log(O_X^\dagger O_Y)$

Remark on row pivoting in the QR factorization

For the numerical accuracy of the QR factorization, an additional row pivoting may be needed to obtain the rows ordered so that their ℓ_∞ norms are decreasing, see e.g. Cox and Higham (1998). If Ψ is a permutation matrix that encodes the row pivoting, then $(\Psi O_X)^\dagger = O_X^\dagger \Psi^T$, so that $(\Psi O_X)^\dagger (\Psi O_Y) = O_X^\dagger O_Y$. This means that the row pivoting in the QR factorization is equivalent to a reordering of the data. The column pivoting corresponds to reordering the basis' functions.

Test Example; $m = 9$ revisited



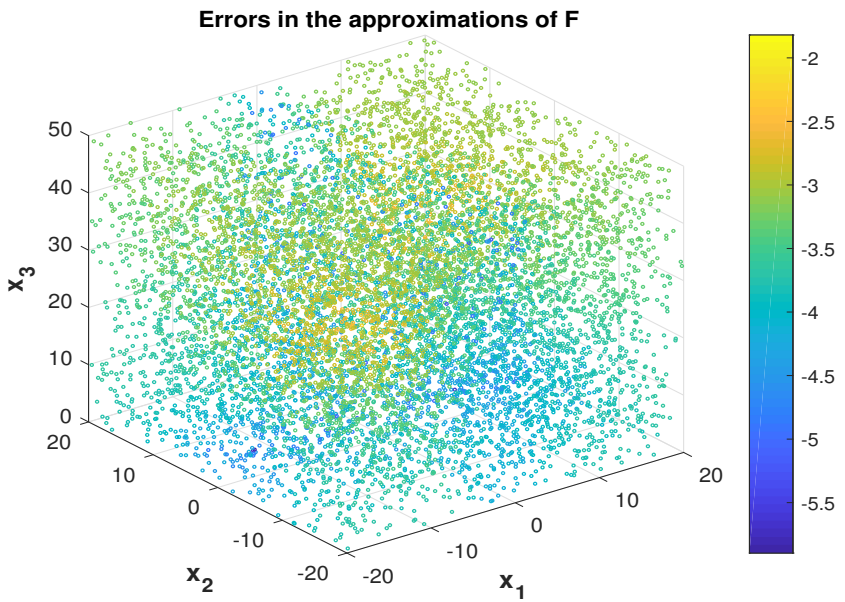
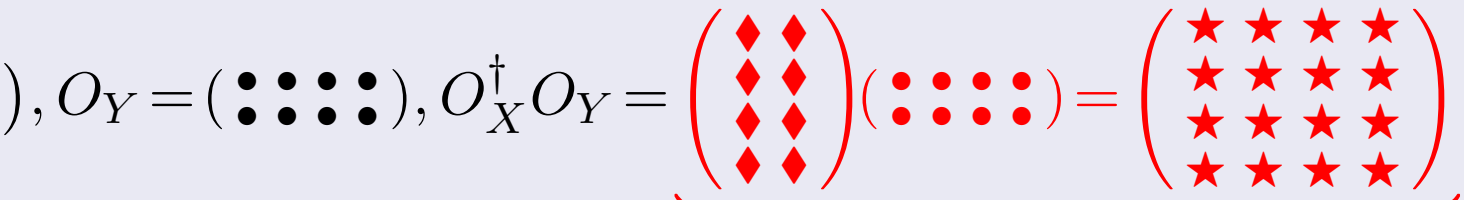


Figure: *Left panel*: The (computed) eigenvalues of $U_N \in \mathbb{R}^{220\times 220}$ ($\times$), and the eigenvalues of $exp(\delta t L_N)$ ($\circ$). The maximal relative difference between the matching eigenvalues is computed as $8.1 \cdot 10^{-9}$. Here U_N is computed as $U_N = \Pi R^{-1} Q^T O_Y$ without any truncation of R. The diagonal entries of R span, in absolute value, the range between $1.9 \cdot 10^{16}$ and $1.0 \cdot 10^{1}$. This reveals the condition number of U_N which is computed as $7.4 \cdot 10^{16}$ by the Matlab function `cond()`. *Right panel*: The values of $\log_{10} \epsilon_k$ for 12000 randomly selected points in the box $[-20, 20] \times [-20, 20] \times [0, 50]$.

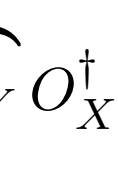
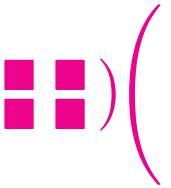
Dual formulation (Mauroy and Gonsalves)

$N > K$

Large dimension N of $\mathcal{F}_N$, number of snapshots $K < N$.

$O_X = ($  $), O_Y = ($ $), O_X^\dagger O_Y =$ 

$\log U_N$ does not exist

$$O_X \overbrace{O_X^\dagger O_Y}^{U_N} O_X^\dagger = ($$  $)($  $)($  $) = ($ $)($ $) = O_Y O_X^\dagger$$

Dual formulation is based on the logarithm of $U_K = O_Y O_X^\dagger$.

The matrix U_K is the matrix Rayleigh quotient of U_N with respect to the range of $O_X^\dagger$, i.e. $U_K = O_X U_N O_X^\dagger$ and $U_N O_X^\dagger = O_X^\dagger U_K$.

Learning better subspaces

- The dual formulation is based on a particular K-dimensional subspace of $\mathcal{F}_N$ (the span of $\wp_1, \ldots, \wp_N$).
- The problem of numerical ill-conditioning of the compression of the infinitesimal generator remains the key issue in both formulations.

We can setup a more general framework:

- Seek other subspaces, and not necessarily of dimension K.
- Both the subspace and its dimension $\widehat{N}$ should be determined with respect to the numerical conditioning of the matrix representations at the finite sequence $\mathbf{x}_1, \ldots, \mathbf{x}_K$.
- A basis of such a $\widehat{N}$-dimensional subspace $\mathcal{F}_{\widehat{N}}$ of $\mathcal{F}_N$ is written as $(\psi_1, \ldots, \psi_{\widehat{N}}) = (\wp_1, \ldots, \wp_N)\mathcal{S}$, where $\mathcal{S}$ is $N \times \widehat{N}$ selection operator, i.e. matrix, of rank $\widehat{N}$.

Pruning the dictionary

$[\mathcal{S}, \widehat{\ell}, \widehat{r}, \Pi_1, \Pi_2] = \text{Subspace\_Selection}(O_X, \mathcal{S}_0, \widehat{N}, \texttt{tol})$

1: Reorder the snapshots: simultaneous row permutation of O_X, O_Y;
2: Bring the selected functions forward to the leading ℓ positions: $Q_X = Q_X \mathcal{S}_0$. Implement $\mathcal{S}_0$ as a sequence of swaps to avoid excess data movement (in the case of large dimensions).
3: $[Q_1, R_1, \Pi_1] = \text{qr}(O_X(:, 1:\ell))$ {*Rank revealing QR factorization with column pivoting. Overwrite* $R_1 = (R_{11}^T, \mathbf{0})^T$ *over the leading* ℓ *columns of* O_X*.*}
4: Determine the numerical rank $\widehat{\ell}$ of R_{11} and in the case $\widehat{\ell} < \ell$ set $R_{11}(\widehat{\ell}+1:\ell, \widehat{\ell}+1:\ell) = \mathbf{0}$.
5: $O_X(:, \ell+1:N) = Q_1^* O_X(:, \ell+1:N)$.
6: $[Q_2, R_2, \Pi_2] = \texttt{qr}(O_X(\widehat{\ell}+1:K, \widehat{\ell}+1:N))$. {*Rank revealing QR factorization with column pivoting.* $R_2 = (R_{22}, R_{23})$ *overwrites* $O_X(\widehat{\ell}+1:K, \widehat{\ell}+1:N)$}
7: Determine the numerical rank $\widehat{r}$ of R_{22}. Set $\widetilde{N} = \widehat{\ell} + \widehat{r}$.
8: $\mathcal{S} = (\mathcal{S}_0(\Pi_1 \oplus \mathbb{I}_{N-\ell})(\mathbb{I}_{\widehat{\ell}} \oplus \Pi_2))(:, 1:\min(\widehat{N}, \widetilde{N}))$;

Output: $\mathcal{S}$

Example (dual method)

The errors $\epsilon_k^{(i)}$ for the intervals $[0, 0.1]$ and $[0, 0.18]$; in the case of the time interval $[0, 0.19]$, the method broke down and the reconstructed values were computed as `NaN`'s.

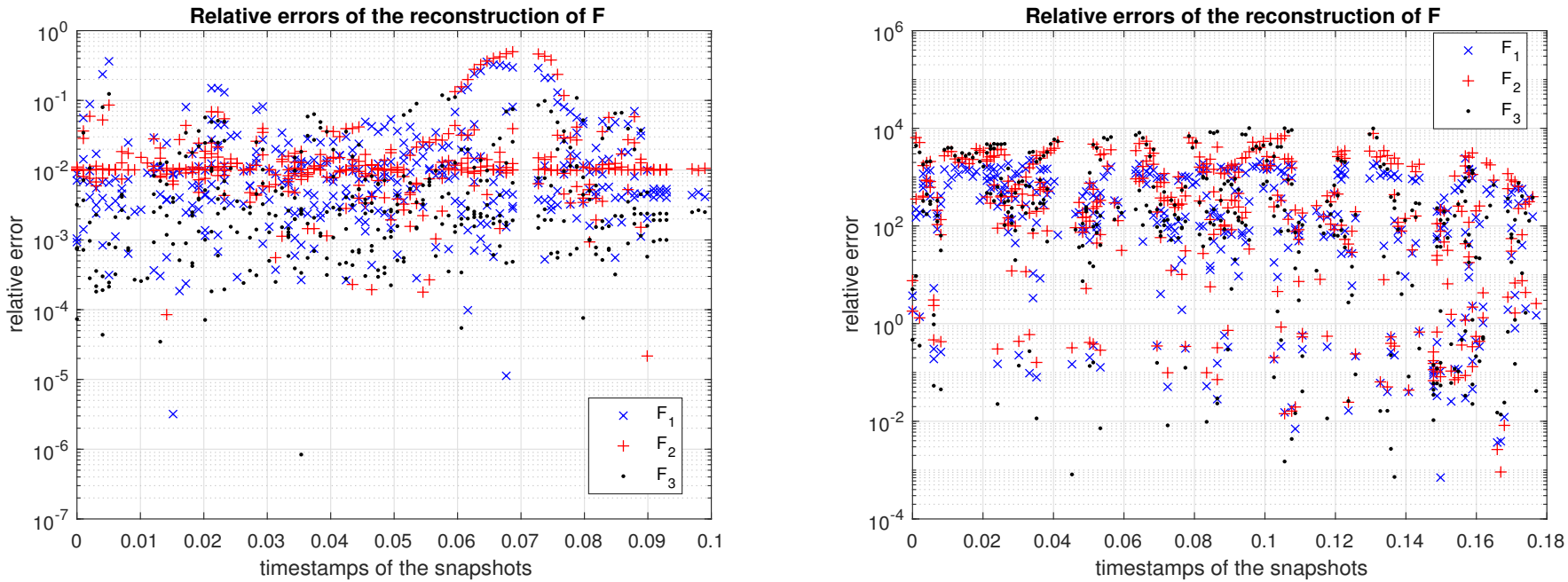
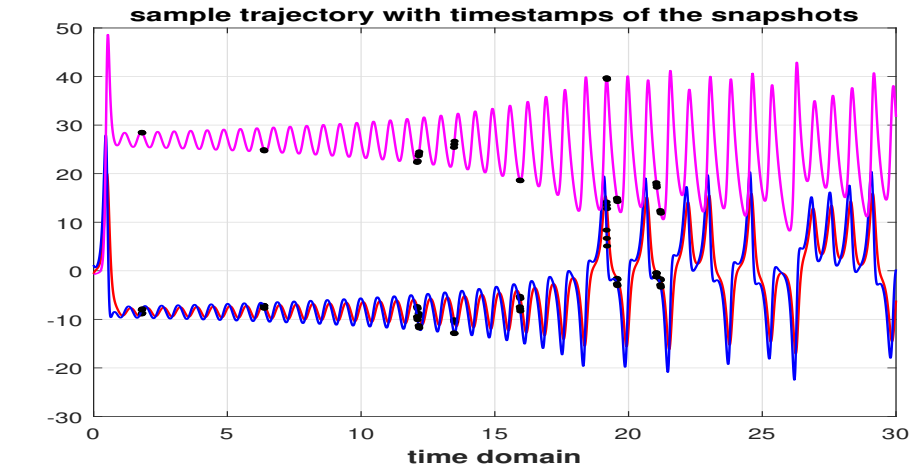
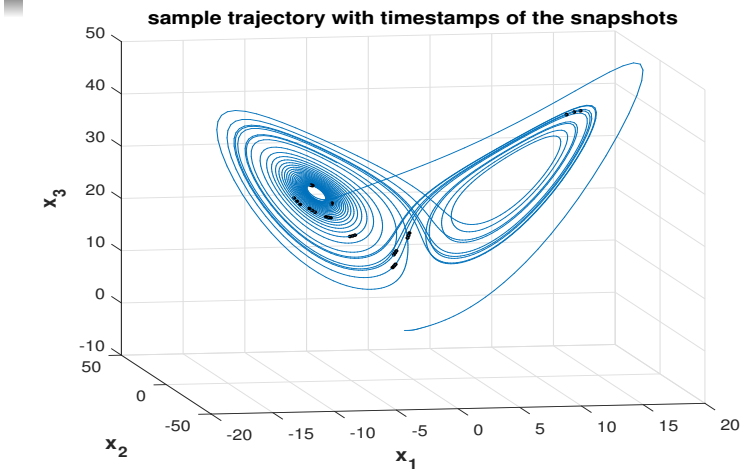
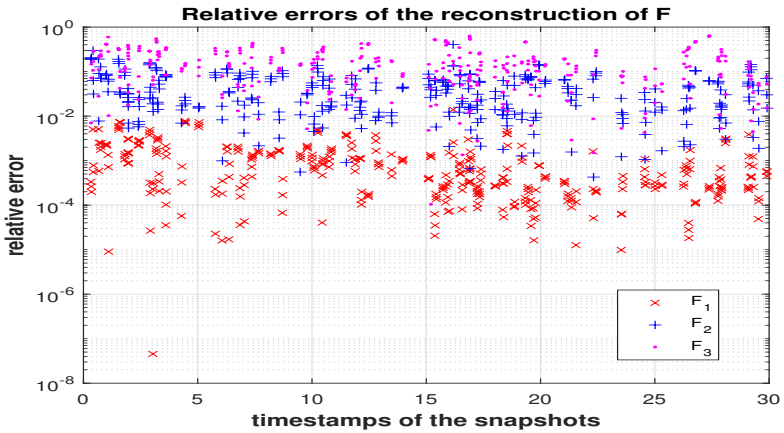


Figure: The errors $\epsilon_k^{(i)} = \frac{|\widetilde{F_i(\mathbf{x}_k)} - F_i(\mathbf{x}_k)|}{\|F(\mathbf{x}_k)\|_\infty}$. Left panel: time interval $[0, 0.1]$. Right panel: time interval $[0, 0.18]$. $\delta t = 0.001$.

Example (new, pruning+preconditioning)







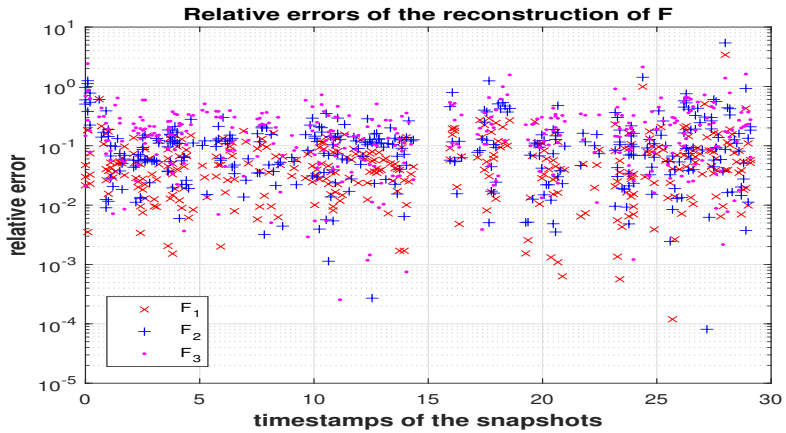


Figure: *First row:* The time stamps of $\mathbf{x}_1, \ldots, \mathbf{x}_{360}$, illustrated on the first out of 12 generated trajectories. Three consecutive snapshots, with time lag 0.01, are taken at ten randomly selected and fixed time instances. *Second row:* The first plot shows the relative errors $\epsilon_k^{(i)}$ with $\delta t = 0.01$, and the second plot for $\delta t = 0.1$.

Other approaches: SINDY

Sparse Identification of Nonlinear Dynamics (SINDy) is a popular method for data driven identification. For details see

- Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz: Discovering governing equations from data by sparse identification of nonlinear dynamical systems, PNAS, vol. 113, no. 1, April 12, 2016, pp. 3932-3937.
- `https://faculty.washington.edu/kutz/page26/`

Exercise

Read he above paper on SINDY and use the software toolbox to test it on your favorite data set.

References

- S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *PNAS, vol. 113, no. 1, April 12, 2016, pp. 3932-3937.*
- Z. Drmač, I. Mezić, and R. Mohr. Koopman lifting and identification using finite dimensional approximations of the infinitesimal generator – a numerical implementation of the Mauroy-Goncalves method. *Mathematics 2021, 9(17), 2075*.
- N. Higham. Functions of Matrices: Theory and Computation (Chapter 11), *SIAM 2008*.
- A. Mauroy, J. Goncalves. Koopman-based lifting techniques for nonlinear systems identification, *IEEE Transactions on Automatic Control 65 (6) 2020, 2550–2565*.