ME493 - Methods of Data-Driven Control

Benjamin Aziel

2024-06-03

Table of contents

Preface

Documentation of the independent study ME493: Methods of Data-Driven Control, taken during Spring 2024.

Instructor: Dirk M. Luchtenburg **Co-conspirators:** Sohaib Bhatti, Khushant Khurana, Eunkyu (Q) Kim, Sonam Okuda **Email:** dirk.luchtenburg@cooper.edu

Description: This independent study provides an introduction to state-of-the-art methods employed in the field of data-driven engineering. Data-driven methods such as the proper orthogonal decomposition and the dynamic mode decomposition are used in a variety of fields including fluid dynamics, climate analysis, mixing problems, the study of infectious diseases, etc. Data-driven methods rely heavily on concepts from linear algebra, calculus, probability, and statistics. We will review these essential elements and develop a hands-on understanding of a selection of data-driven methods. The study culminates with an application of some method(s) to a practical problem which is to be carefully detailed in a technical report.

Referenced texts include the following:

- Strang (2019)
- Brunton and Kutz (2022)
- Boyd and Vandenberghe (2018)
- Géron (2022)
- Sutton and Barto (2018)

1 The Four Fundamental Subspaces

In the Strang-ian view of linear algebra, an m by n matrix $\mathbf{A}$ is associated with four fundamental subspaces - two of $\mathbb{R}^m$ and two of $\mathbb{R}^n$. It's easiest to illustrate this using an example matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \end{bmatrix}$$

where:

$$\mathbf{r}_1 = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad \mathbf{r}_2 = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

$$\mathbf{c}_1 = \begin{bmatrix} 1 & 4 \end{bmatrix}^T \quad \mathbf{c}_2 = \begin{bmatrix} 2 & 5 \end{bmatrix}^T \quad \mathbf{c}_3 = \begin{bmatrix} 3 & 6 \end{bmatrix}^T$$

Column Space

The column space (or range) $R(\mathbf{A})$ contains all linear combinations of the column vectors of $\mathbf{A}$.

It only takes two linearly independent vectors to span $\mathbb{R}^2$, and we have three! Our column space is $\mathbb{R}^2$.

Row Space

The row space $R(\mathbf{A}^T)$ contains all linear combinations of the column vectors of $\mathbf{A}^T$ (or equivalently, the row vectors of A).

Transposed row vectors $\mathbf{r}_1^T$ and $\mathbf{r}_2^T$ span the following plane:

$$\left\{ \begin{bmatrix} x_1 + 4x_2 \\ 2x_1 + 5x_2 \\ 3x_1 + 6x_2 \end{bmatrix} : \mathbf{x} \in \mathbb{R}^2 \right\}$$

Null Space

The null space $N(\mathbf{A})$ contains all solutions $\mathbf{u}$ to $\mathbf{Au} = \mathbf{0}$.

Solving this equation yields the nontrivial solution:

$$\mathbf{u} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Our null space is the span of $\mathbf{u}$, or the following line:

$$\left\{ \begin{bmatrix} x \\ -2x \\ x \end{bmatrix} : x \in \mathbb{R} \right\}$$

Left Null Space

The left null space $N(\mathbf{A}^T)$ contains all solutions $\mathbf{v}$ to $\mathbf{A}^T\mathbf{v} = \mathbf{0}$.

This equation has no nontrivial solutions, so the left null space is the zero subspace.

The "big picture of linear algebra," as Gil Strang puts it, is that for an m by n matrix $\mathbf{A}$:

- The column space/range $R(\mathbf{A})$ is perpendicular to the left null space $N(\mathbf{A}^T)$ in $\mathbb{R}^m$
- The row space $R(\mathbf{A}^T)$ is perpendicular to the null space $N(\mathbf{A})$ in $\mathbb{R}^n$

Rank

The rank r of a matrix $\mathbf{A}$ is the number of independent rows/columns, i.e., the row space and column space/range have the same dimension r.

- The dimension of the null space $N(\mathbf{A})$ is $n - r$ and the dimension of the left null space $N(\mathbf{A}^T)$ is $m - r$.

2 Singular Value Decomposition

2.1 Overview

The singular value decomposition (SVD) is a fundamental matrix factorization with numerous applications in data analysis and scientific computing. Mathematically, the SVD of an $m \times n$ matrix $\mathbf{X}$ is a factorization of the form:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ diagonal matix with non-negative real numbers on the diagonal, and $\mathbf{V}$ is an $n \times n$ orthogonal matrix.

The SVD is particularly useful for analyzing large, high-dimensional datasets that can be well-approximated by matrices of much lower rank. By extracting the dominant patterms in the data, the SVD enables efficient dimensionality reduction, noise removal, and data compression. It is the foundation of techniques like principal component analysis (PCA) and is widely applied in fields such as signal processing, machine learning, and image analysis.

Proper orthogonal decomposition (POD) modes are a set of orthogonal basis functions that optimally represent a given dataset in a least-squares sense. They are obtained by performing an SVD on a data matrix. POD modes form an orthonormal basis, meaning the modes are mutually orthogonal and have unit norm.

The modes are ranked by their energy content, with the first mode capturing the most energy and subsequent modes capturing progressively less energy.

One last thing- the pseudoinverse can be computed using the singular value decomposition of a matrix. If $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*$ is the SVD of $\mathbf{X}$, then the pseudoinverse is given by:

$$\mathrm{X}^+ = \mathbf{V}\Sigma^{-1}\mathbf{U}^*$$

2.2 POD Analysis of Low Reynolds Number Pitching Airfoil DNS

The dataset analyzed contains direct numerical simulations (DNS) of two-dimensional stationary and pitching flat-plate airfoils at a Reynolds number of 100. The dataset includes time-resolved snapshots of the velocity field, lift and drag coefficients, and airfoil kinematics spanning 40-100 convective time units. The cases consist of a stationary airfoil and eight different pitching frequencies. This dataset is part of a database intended to aid in the conception, training, demonstration, evaluation, and comparison of reduced-complexity models for fluid mechanics, created by Aaron Towne and Scott Dawson.

The dataset also includes a MATLAB function that provides a simple implementation of Dynamic Mode Decomposition (DMD), a data-driven method we'll get to later.

To analyze the DNS data using POD, I first extracted the velocity components from the provided snapshots. I then computed the mean-corrected snapshots by subtracting the mean of each snapshot and arranged them into a matrix $\mathbf{X}$.

Next, I performed an economy-sized SVD on $\mathbf{X}$. The squared singular values are plotted here, as a function of the mode index.

This plot reveals how many individual POD basis vectors there are. The rapid decay of the singular values indicates that the flow is well-approximated by a low-rank subspace, with the first 16 modes capturing the majority of the energy.

Visualizing the first six POD modes for both u_x and u_y revealed the spatial structure of the dominant flow patterns. The oscillatory modes have a characteristic wavelength that can be estimated from the spatial distribution of the mode amplitudes.

Here are the temporal amplitudes associated with those spatial modes:

Finally, I reconstructed the snapshots using a rank-4 approximation, which captures the most energetic flow structures. Comparing the reconstructed snapshots with the original data showed that the low-rank approximation successfully recovers the essential flow physics, such as the presence of coherent structures and the overall flow patterns.

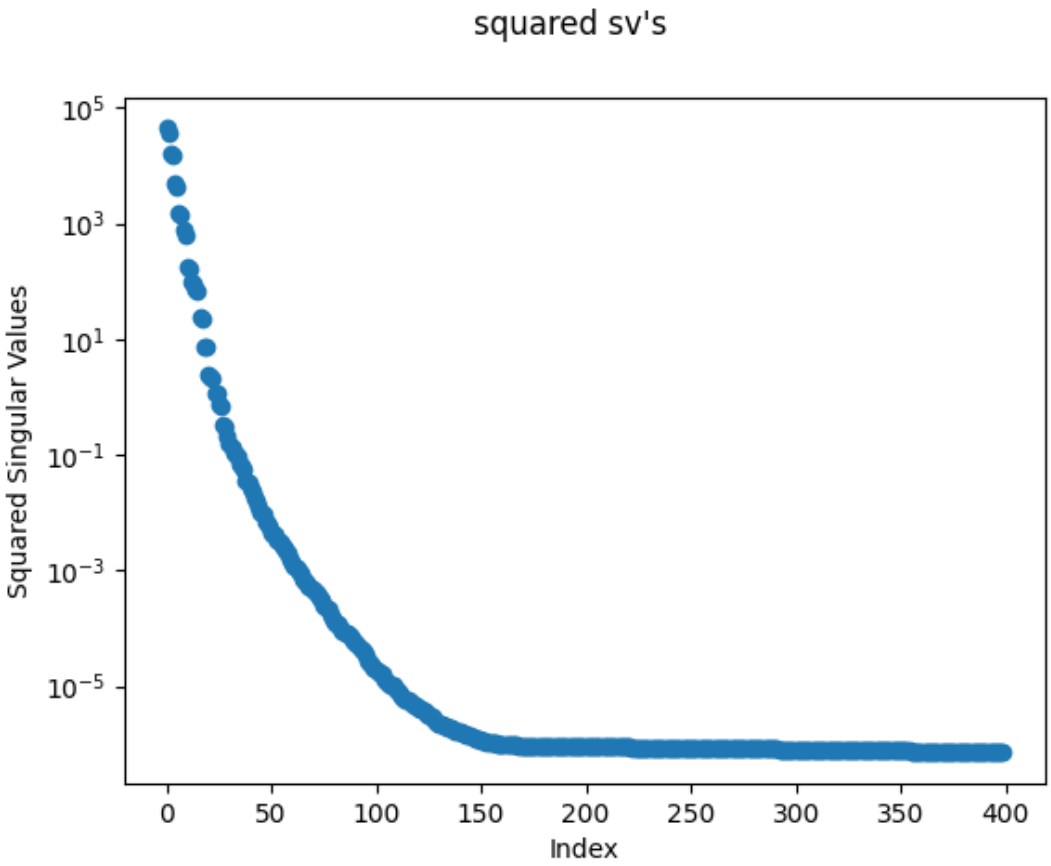


Figure 2.1: squared_sv

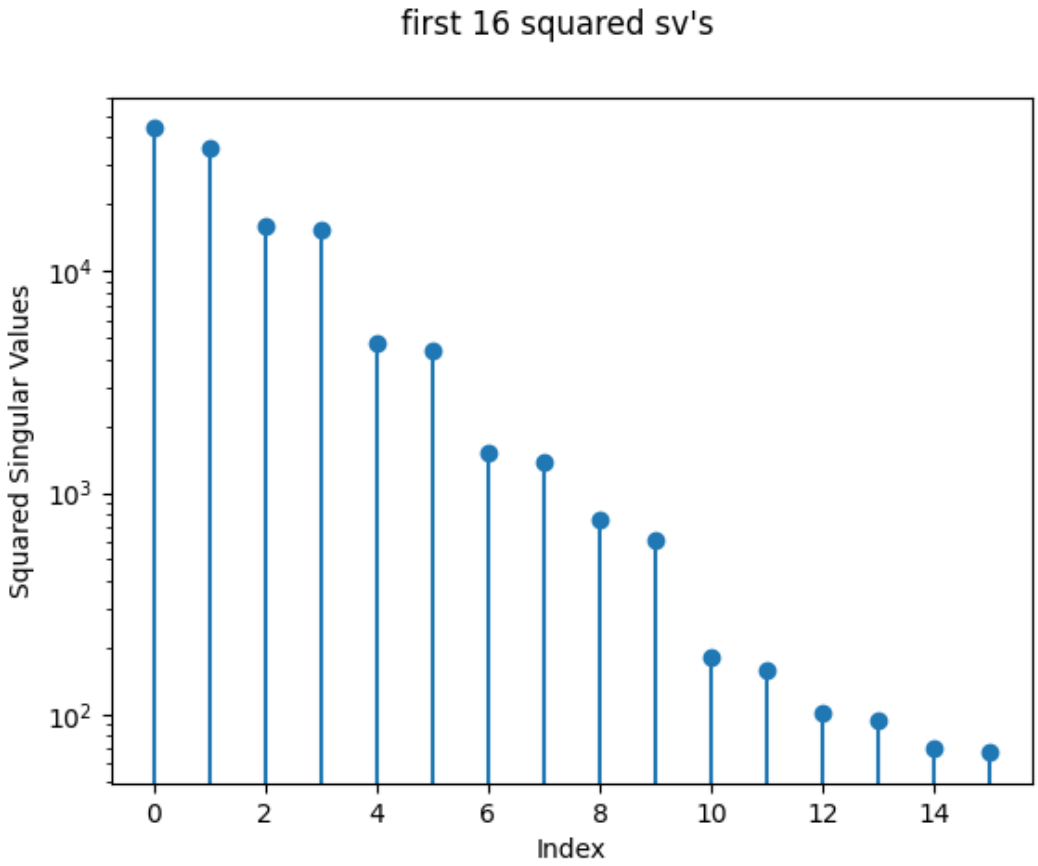


Figure 2.2: squared_sv_truncated

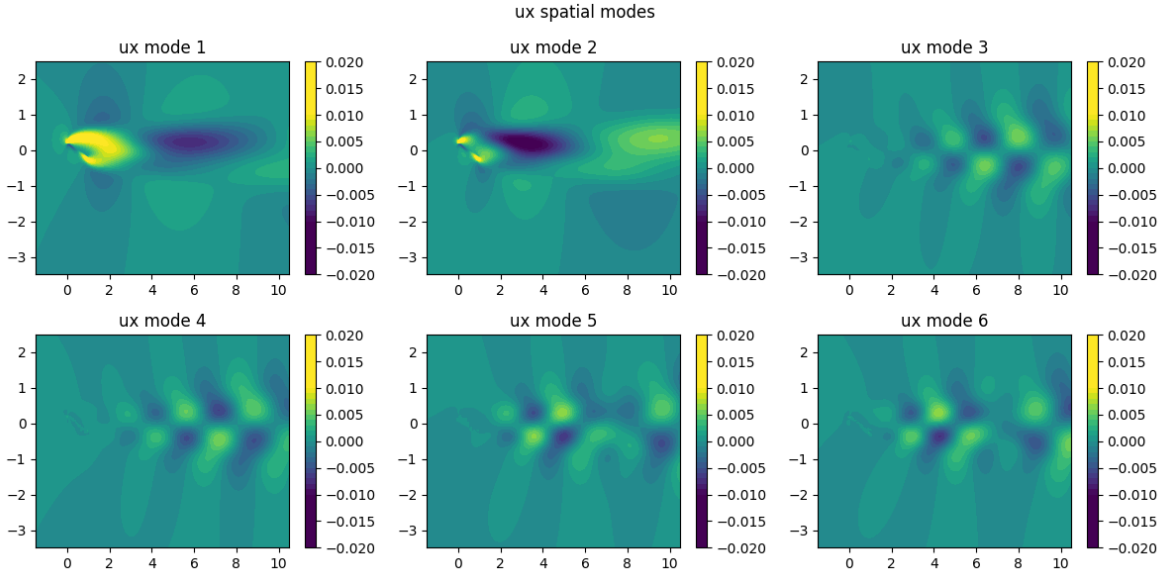


Figure 2.3: ux_spatial_modes

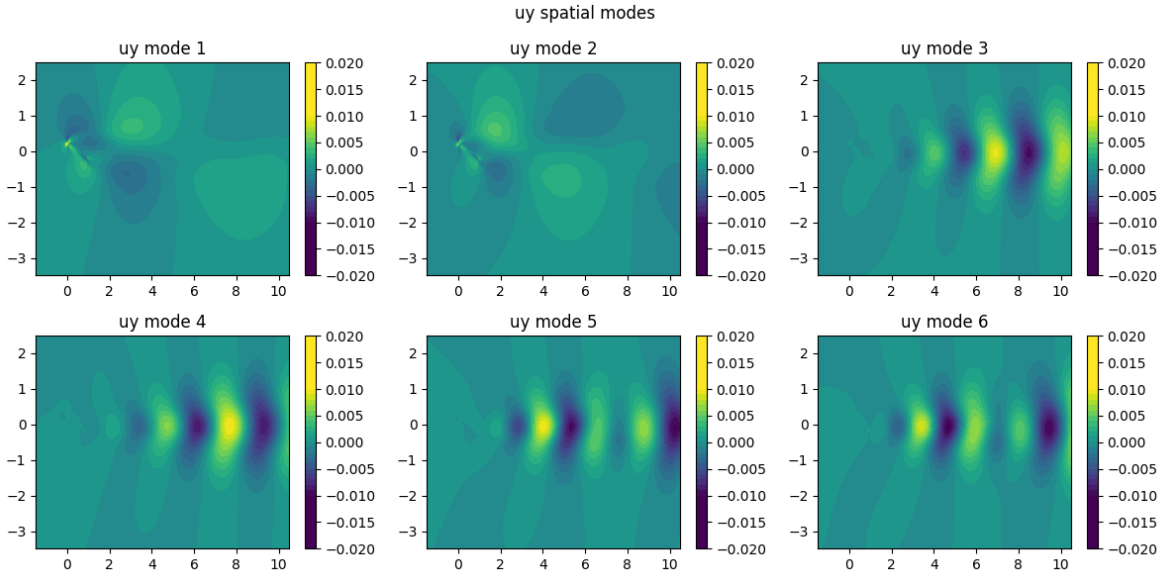


Figure 2.4: uy_spatial_modes

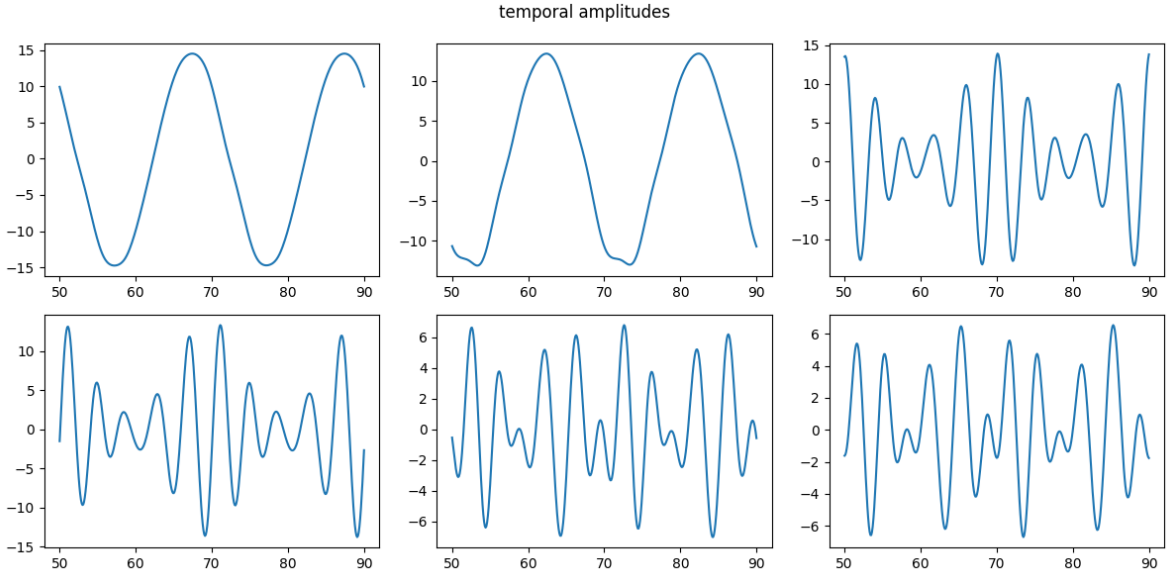


Figure 2.5: temporal_amplitudes

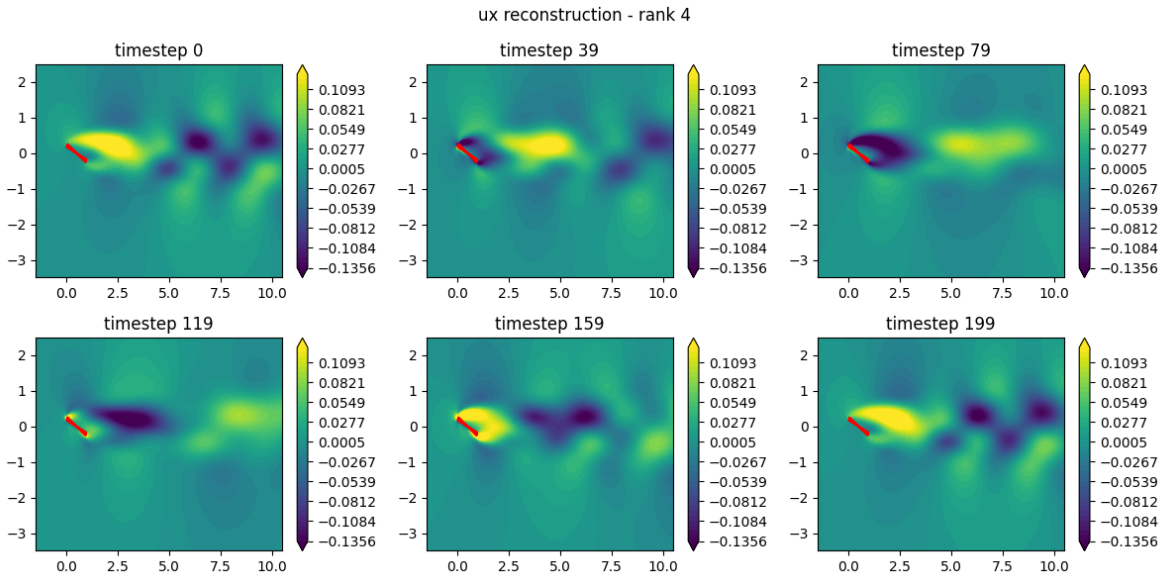


Figure 2.6: ux_reconstruction

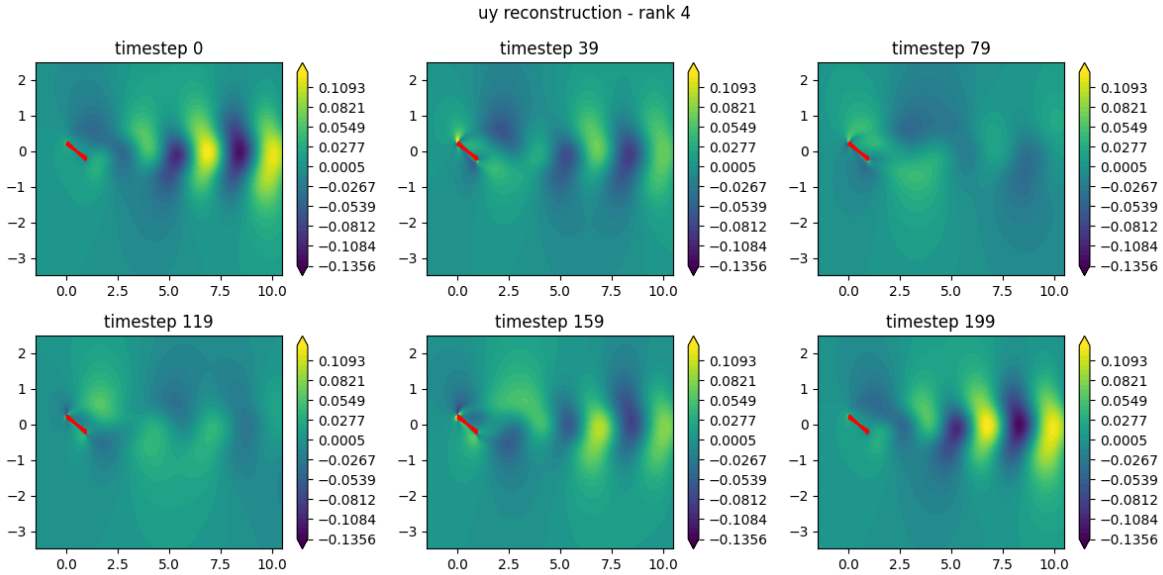


Figure 2.7: uy_reconstruction

3 Data-Driven Dynamical Systems

3.1 Overview: Dynamic Mode Decomposition

Dynamic mode decomposition (DMD) is a dimensionality reduction technique developed in the fluid dynamics community to extract spatiotemporal coherent structures from high-dimensional data. Unlike principal component analysis (PCA) or proper orthogonal decomposition (POD), which focus solely on spatial correlations or energy content, DMD provides a modal decomposition where each mode is associated with a specific oscillation frequency and growth/decay rate. This allows DMD to capture the temporal evolution of the system, in addition to reducing the dimensionality. DMD achieves this by approximating the Koopman operator, which describes the linear dynamics governing the system, rather than just identifying the dominant spatial patterns like PCA/POD. As a result, DMD modes can be more physically meaningful than the orthogonal modes generated by PCA, as they directly correspond to the intrinsic temporal behaviors of the system.

DMD operates on a series of data snapshots, typically obtained from simulations or experiments. These snapshots are organized into a matrix $\mathbf{V}$, where:

$$\mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_N] \in \mathbb{R}^{M \times N}$$

Each $\mathbf{v}_i \in \mathbb{R}^M$ represents a snapshot at a specific time. DMD attempts to find a linear operator $\mathbf{A}$ that approximates the evolution of the system from one snapshot to the next.

First, the snapshot data is arranged into two matrices $\mathbf{X}$ and $\mathbf{X}'$, where $\mathbf{X}$ contains the first N-1 snapshots and $\mathbf{X}'$ contains the last N-1 snapshots.

The SVD of $\mathbf{X}$ is taken. To deal with high-dimensional data, a reduced order representation may be used, by truncating the SVD to the first r modes.

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^* \approx \mathbf{U}_r\Sigma_r\mathbf{V}_r^*$$

The full matrix $\mathbf{A}$ is found by computing the pseudo-inverse of $\mathbf{X}$:

$$\mathbf{A} = \mathbf{X}'\mathbf{V}_r\Sigma_r^{-1}\mathbf{U}_r^*$$

The next step involves computing the matrix $\tilde{\mathbf{A}}$, which is the projection of $\mathbf{A}$ onto the subspace spanned by the columns of $\mathbf{U}_r$.

$$\tilde{\mathbf{A}} = \mathbf{U}_r^* \mathbf{A} \mathbf{U}_r = \mathbf{U}_r^* \mathbf{X}' \mathbf{V}_r \Sigma_r^{-1}$$

Then, we "eigendecompose" (is that a word?) $\tilde{\mathbf{A}}$ to find its eigenvalues Λ and eigenvectors $\mathbf{W}$. These eigenvalues and eigenvectors are used to compute the DMD modes Φ, which are given by:

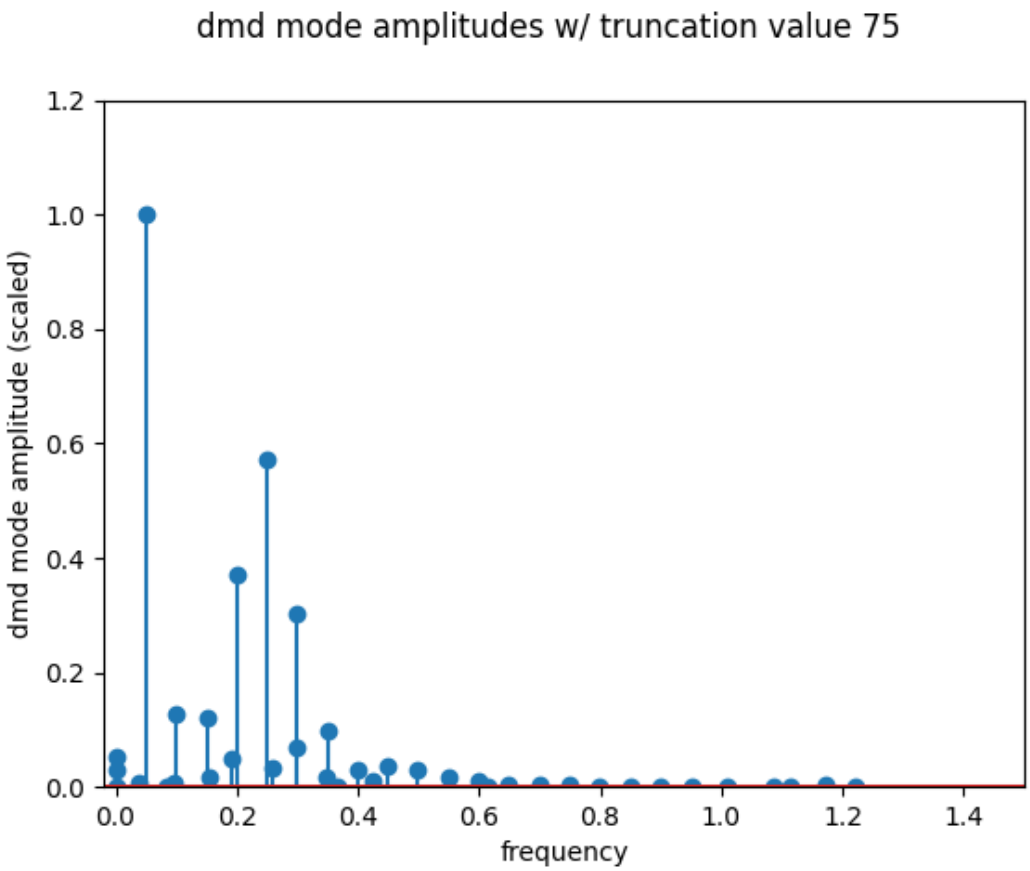
$$\Phi = \mathbf{X}' \mathbf{V}_r \Sigma_r^{-1} \mathbf{W}$$

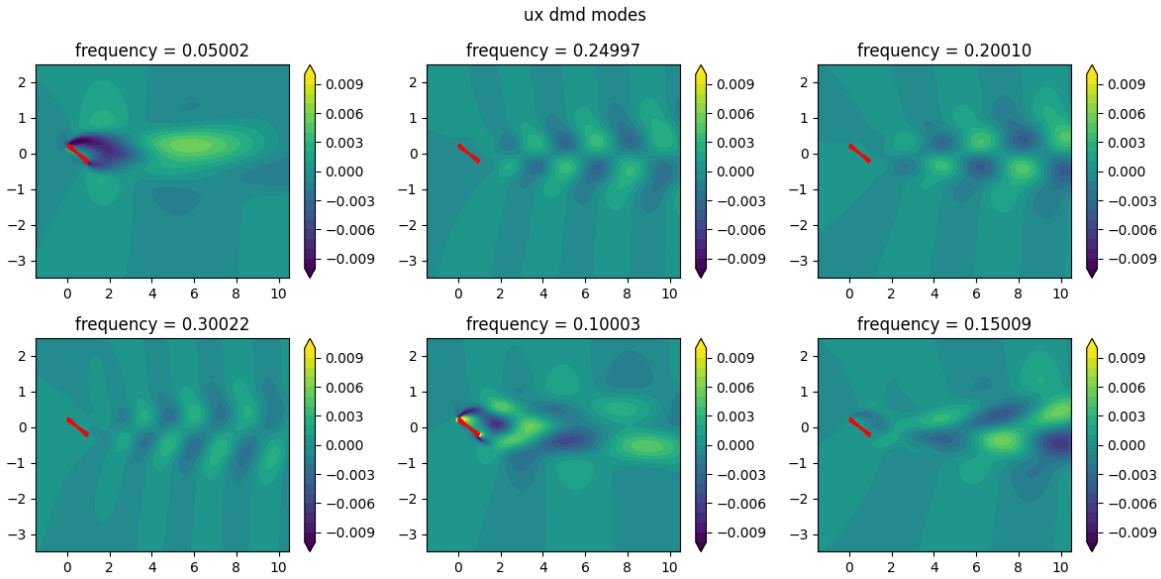
Notably, these high-dimensional DMD modes are also the eigenvectors of the full $\mathbf{A}$ matrix corresponding to the eigenvalues in Λ.

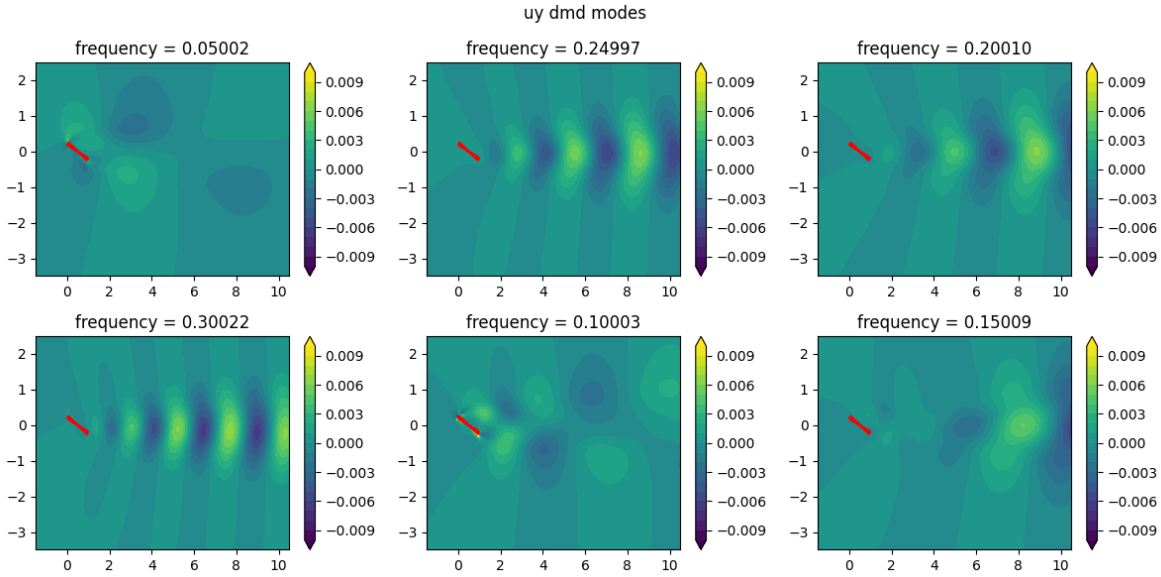
3.2 DMD Analysis of Low Reynolds Number Pitching Airfoil DNS

We now return to the dataset analyzed using POD, but now we use the dynamic mode decomposition to analyze this data.

The modal amplitudes, and first six u_x and u_y modes are displayed below. It is clearly apparent that the modes are **not** identical to the POD modes. Due to the non-orthogonal nature of its modes, DMD-based representations can be less concise compared to those generated by PCA. However, DMD modes often provide greater physical insight, as each mode corresponds to a damped or driven sinusoidal time behavior.


dmd mode amplitudes w/ truncation value 75
dmd mode amplitude (scaled)
frequency
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


ux dmd modes
frequency = 0.05002
frequency = 0.24997
frequency = 0.20010
frequency = 0.30022
frequency = 0.10003
frequency = 0.15009



Here, the spatial modes reveal the dominant patterns or structures within the flow. Each spatial mode corresponds to a specific frequency and growth/decay rate, providing insights into how different flow features contribute to the overall dynamics and how they change spatially across the domain.

3.3 Overview: Sparse Identification of Nonlinear Dynamics

The SINDy (Sparse Identification of Nonlinear Dynamics) algorithm is a technique designed to uncover the governing equations of a dynamical system from observed data. It works by identifying a sparse (or concise) representation of the system's dynamics, focusing on the most relevant terms that describe the system's behavior.

The process begins with collecting time-series data of the state variables of the system. From this data, a comprehensive library of candidate functions, which may include polynomials, trigonometric functions, and other nonlinear terms, is constructed. These functions represent potential components of the system's underlying equations.

SINDy then employs sparse regression techniques to sift through this library, selecting only the most pertinent functions. The goal is to find a minimal set of terms that can accurately describe the dynamics, resulting in a sparse representation of the system's governing equations. This approach ensures that the resulting model is both interpretable and parsimonious, capturing the essential dynamics without unnecessary complexity.

3.4 SINDy Implementation on Temporal Amplitudes

We attempt to fit a dynamic system $\dot{\mathbf{x}} = f(\mathbf{x})$ to the POD temporal amplitudes using the SINDy algorithm. We select a number of temporal amplitudes to consider (6, in this case) and construct the matrix of temporal amplitudes tempamps by scaling the right singular vectors $\mathbf{V}$ with the singular values Σ.

Next, we compute the time derivatives of the temporal amplitudes using finite differences. We then pool the data to form a library of candidate functions Θ, including polynomial terms up to the second order, using a function that constructs a matrix where each column is a candidate term for the dynamics, such as constant, linear, and quadratic terms of the state variables.

We perform sequential thresholding least squares to find a sparse matrix Ξ that best fits the time derivatives $\dot{\mathbf{x}}$ to the candidate functions in Θ. The sparsification process involves iteratively zeroing out small coefficients (below the threshold value) and refitting the remaining coefficients.

The system is then integrated using cumulative trapezoidal numerical integration. This was initially attempted using Runge-Kutta methods, but these methods proved ineffective likely because the system is multi-scale (and rather complicated), leading to instability and inaccuracies in the solutions.

We then compare these SINDy-derived amplitudes with the original POD temporal amplitudes by plotting them. The plots show the effectiveness of the SINDy model in capturing the dynamics.

Usage of the PySINDy library was also experimented with - it yielded identical results.

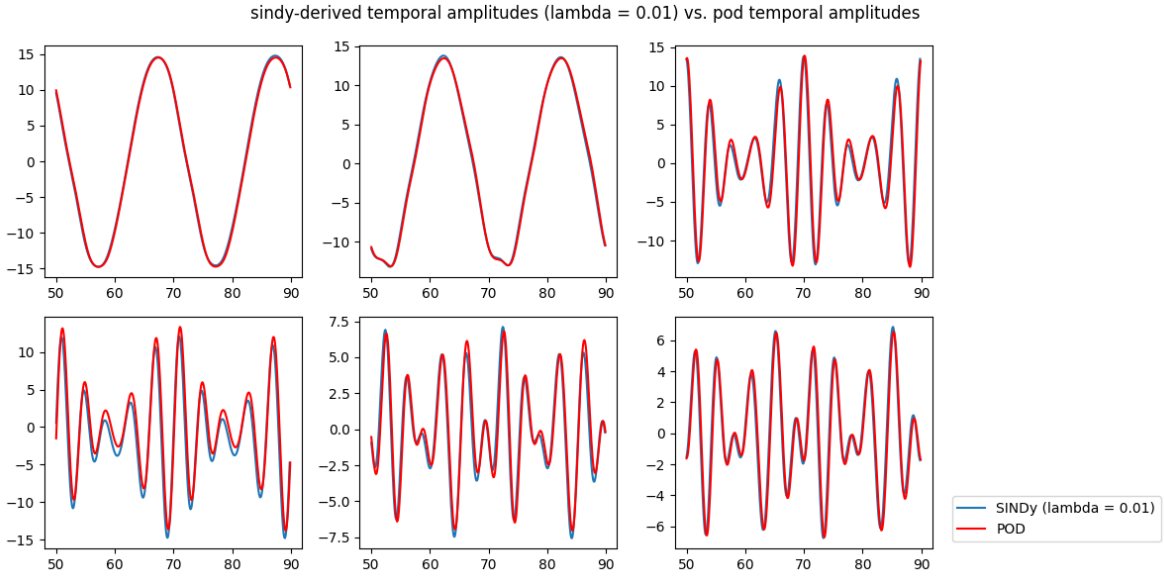


Figure 3.1: sindy_fit

4 System Identification Techniques

4.1 Overview: Eigensystem Realization Algorithm

The Eigensystem Realization Algorithm (ERA) is a method used in system identification to derive a state-space model from time-domain data, typically impulse or step response data.

> **Hankel Matrix**
>
> A Hankel matrix is a structured matrix where each ascending diagonal from left to right is constant.

Two Hankel matrices are constructed from the collected data. Let $\mathbf{y}$ be the sequence of observed data points. Then the Hankel matrix $\mathbf{H}_0$ and the shifted Hankel matrix $\mathbf{H}_1$ are defined as:

$$\mathbf{H}_0 = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \\ y_2 & y_3 & \cdots & y_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & y_{n+1} & \cdots & y_{n+m-1} \end{bmatrix}$$

$$\mathbf{H}_1 = \begin{bmatrix} y_2 & y_3 & \cdots & y_{m+1} \\ y_3 & y_4 & \cdots & y_{m+2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n+1} & y_{n+2} & \cdots & y_{n+m} \end{bmatrix}$$

The singular value decomposition (SVD) is performed on $\mathbf{H}_0$ (and truncated to keep only the most significant singular values):

$$\mathbf{H}_0 = \mathbf{U}\Sigma\mathbf{V}^* \approx \mathbf{U}_r\Sigma_r\mathbf{V}_r^*$$

The reduced matrices are used to construct the state-space matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$:

$$\mathbf{A} = \Sigma_r^{-0.5}\mathbf{U}_r^*\mathbf{H}_1\mathbf{V}_r\Sigma_r^{-0.5}$$
$$\mathbf{B} = \Sigma_r^{0.5}\mathbf{V}_r^*[:, 0]$$

$$\mathbf{C} = \mathbf{U}_r[0, :]\Sigma_r^{0.5}$$

When constructing the Hankel matrices, collecting sufficient data ensures balanced controllability and observability Gramians $\mathcal{O}_d\mathcal{O}_d^*$ and $\mathcal{C}_d\mathcal{C}_d^*$. If data is insufficient, ERA only approximates the balance. Alternatively, collecting enough data for the Hankel matrices to reach numerical full rank (where remaining singular values are below a threshold) yields a low-order ERA model.

4.2 Atomic Force Microscope Transfer Function Recovery

We use the Eigensystem Realization Algorithm (ERA) to perform system identification on a transfer function representing the dynamics of an atomic force microscope.

$$G(s) = \frac{k\omega_2^2\omega_3^2\omega_5^2\left(s^2 + 2\zeta_1\omega_1 s + \omega_1^2\right)\left(s^2 + 2\zeta_4\omega_4 s + \omega_4^2\right)e^{-s\tau}}{\omega_1^2\omega_4^2\left(s^2 + 2\zeta_2\omega_2 s + \omega_2^2\right)\left(s^2 + 2\zeta_3\omega_3 s + \omega_3^2\right)\left(s^2 + 2\zeta_5\omega_5 s + \omega_5^2\right)}$$

with $\omega_i = 2\pi f_i$, $k = 5$,

$f_1 = 2.4$ kHz,	$f_2 = 2.6$ kHz,	$f_3 = 6.5$ kHz,	$f_4 = 8.3$ kHz,	$f_5 = 9.3$ kHz,
$\zeta_1 = 0.03$,	$\zeta_2 = 0.03$,	$\zeta_3 = 0.042$,	$\zeta_4 = 0.025$,	$\zeta_5 = 0.032$

and $\tau = 10^{-4}$ seconds.

A Padé approximant is used to handle the time delay, as the control library does not support time delays directly.

The impulse response of the system is computed over a specified time range.

Hankel matrices are constructed using the impulse response data. The SVD is used to decompose the Hankel matrix into components that facilitate the identification of the system's state-space representation.

The state-space matrices **A**, **B**, and **C** are calculated using the truncated SVD components and fractional matrix powers. This step essentially reduces the system to a simplified model while preserving its significant dynamics.

The squared singular values of the Hankel matrix are plotted to identify the most significant components.

A new state-space model is created using the ERA results. Bode plots of the original and reconstructed systems are generated and compared. These plots visualize the frequency response

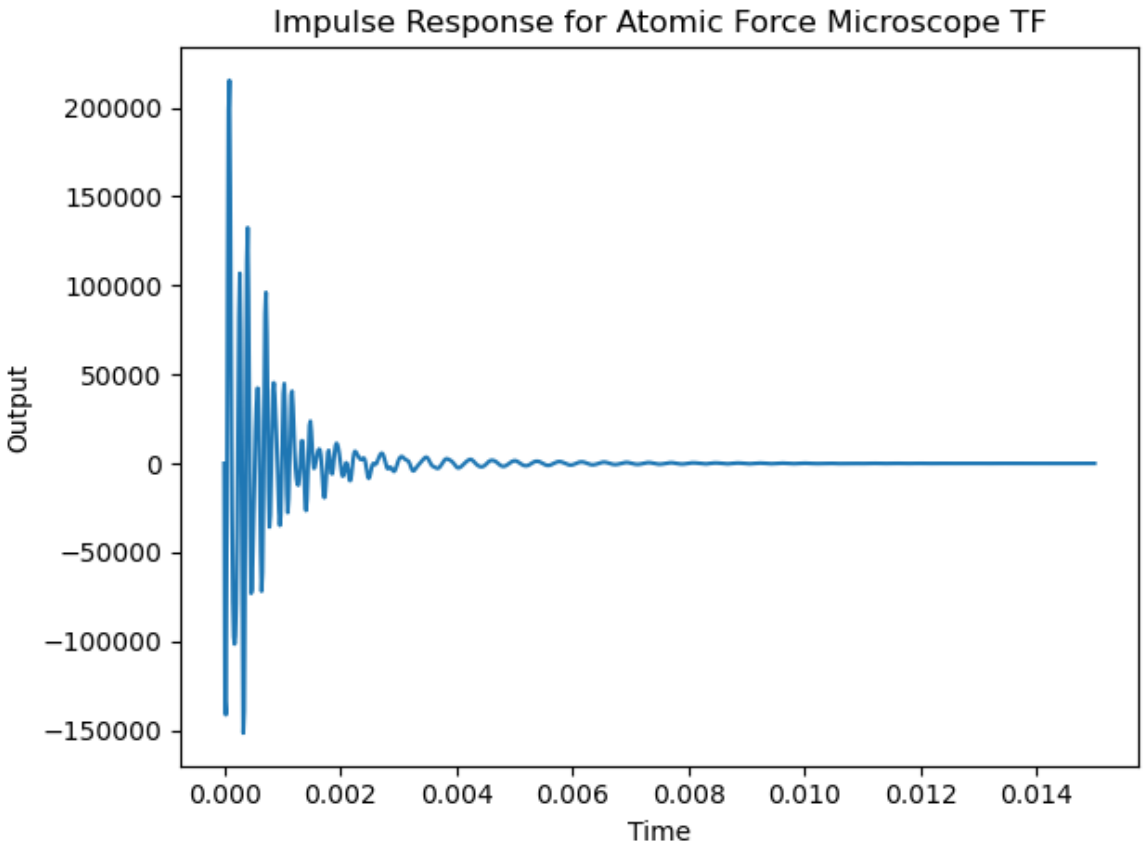


Figure 4.1: Impulse response of the provided transfer function.

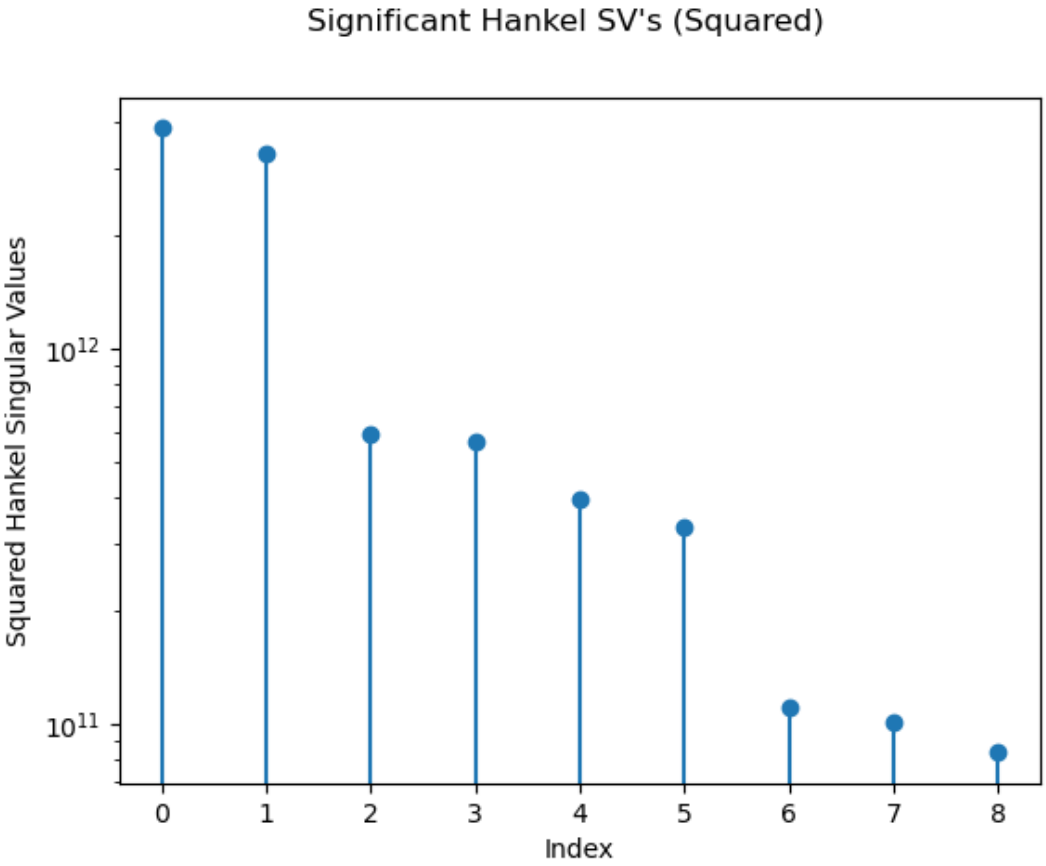


Figure 4.2: Most significant Hankel squared singular values.

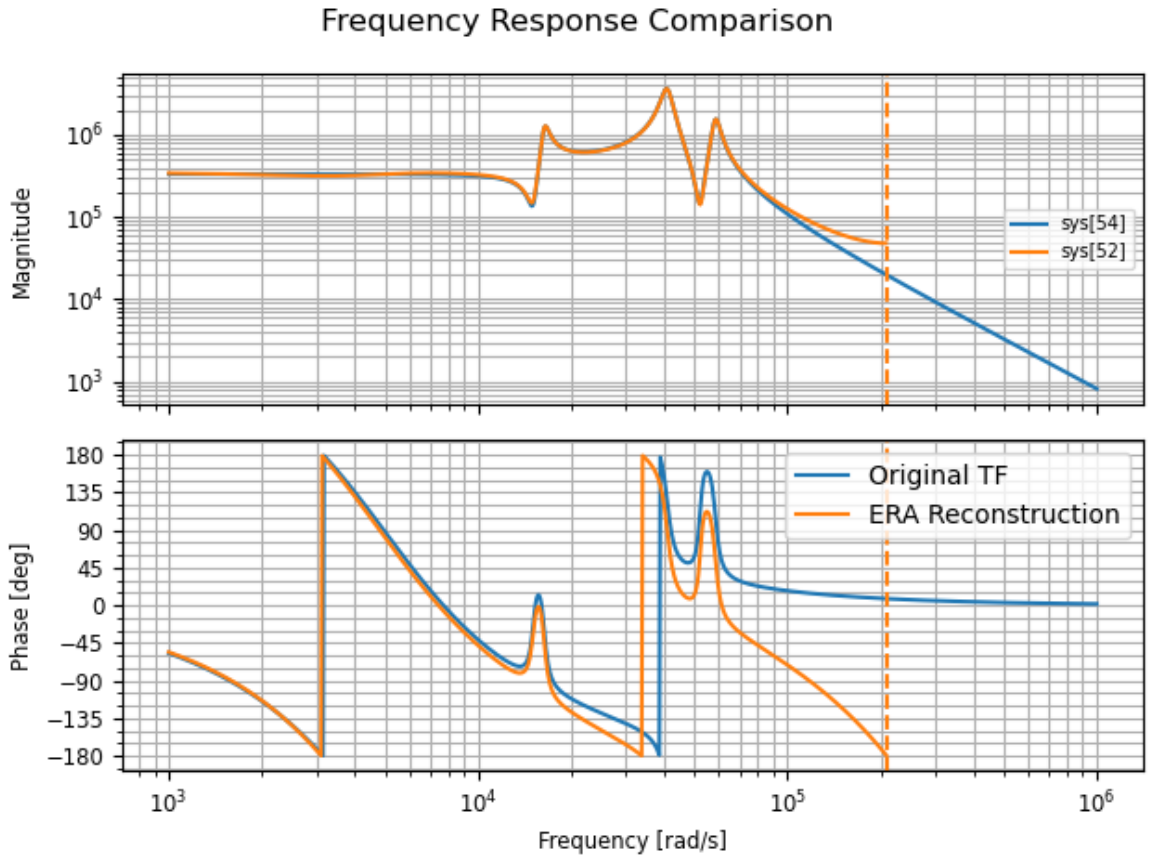


Figure 4.3: Frequency responses of the original and reconstructed images.

of both systems, ensuring that the reconstructed model accurately represents the original system's dynamics.

Due to the use of the Padé approximant for handling the time delay in the system, there are minor discrepancies in the phase of the frequency response. While it effectively captures the overall behavior of the delay, it can introduce slight inaccuracies in the phase response, especially at higher frequencies. Consequently, the phase response of the system might not perfectly align with the actual phase characteristics, although the magnitude response remains largely unaffected.

4.3 Overview: DMD with Control

Dynamic Mode Decomposition with control (DMDc) is a modification of the standard DMD algorithm designed to handle input-output systems where actuation or control inputs are present. In short, the DMDc method tries to find the best-fit linear operators $\mathbf{A}$ and $\mathbf{B}$ that approximately describe the following dynamics based on measurement data:

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

$\mathbf{X}$ and $\mathbf{X}$' are defined as they were for standard DMD. A matrix of the actuation input history is assembled, defined as follows:

$$\Upsilon = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_N \end{bmatrix}$$

(Υ is used in lieu of $\mathbf{U}$ to disambiguate between the $\mathbf{U}$ matrix in the SVD.) The dynamics are now written:

$$\mathbf{X}' \approx \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \Upsilon \end{bmatrix} = \mathbf{G}\Omega$$

$\mathbf{G} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$ can be isolated using least-squares regression:

$$\mathbf{G} \approx \mathbf{X}'\Omega^{+}$$

The singular value decomposition of $\Omega = \begin{bmatrix} \mathbf{X}^* & \Upsilon^* \end{bmatrix}^*$ is taken:

$$\Omega = \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^*$$

$\tilde{\mathbf{U}}$ is split into two matrices to provide bases for $\mathbf{X}$ and Υ:

$$\tilde{\mathbf{U}} = \begin{bmatrix} \tilde{\mathbf{U}}_1^* & \tilde{\mathbf{U}}_2^* \end{bmatrix}$$

The state matrices $\mathbf{A}$ and $\mathbf{B}$ can now be constructed:

$$\mathbf{A} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\Sigma}^{-1}\tilde{\mathbf{U}}_1^*$$
$$\mathbf{B} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\Sigma}^{-1}\tilde{\mathbf{U}}_2^*$$

$\tilde{\mathbf{U}}$ provides a reduced basis for the input space, whereas $\hat{\mathbf{U}}$ from:

$$\mathbf{X}' = \hat{\mathbf{U}}\hat{\Sigma}\hat{\mathbf{V}}^*$$

provides a reduced basis for the output space. These bases allow us to reduce the order of $\mathbf{G}$ by projecting onto this basis:

$$\tilde{\mathbf{G}} = \hat{\mathbf{U}}^* \mathbf{G} \begin{bmatrix} \hat{\mathbf{U}} \\ \mathbf{I} \end{bmatrix}$$

and the corresponding projected matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are:

$$\tilde{\mathbf{A}} = \hat{\mathbf{U}}^* \mathbf{A} \hat{\mathbf{U}}$$

$$\tilde{\mathbf{B}} = \hat{\mathbf{U}}^* \mathbf{B}$$

The key difference from standard DMD is the augmented data matrix $\mathbf{G}$, which incorporates both state and control input data. This allows DMDc to disambiguate the underlying dynamics from the effects of actuation, providing an accurate input-output model represented by the modes Φ and eigenvalues Λ.

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\Lambda$$

$$\Phi = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\Sigma}^{-1}\tilde{\mathbf{U}}_1^*\hat{\mathbf{U}}\mathbf{W}$$

5 2024SP Project: Population Dynamics

For my final project, I implemented the SINDyC algorithm to identify the governing eqquations of a modified Lotka-Volterra predator-prey model with an external forcing term that artificially inflates the prey population. I then implemented a Model Predictive Control (MPC) strategy to control the system, using the equations identified by the SINDy algorithm.

$$\dot{x}_1 = \alpha x_1 - \beta x_1 x_2$$
$$\dot{x}_2 = -\gamma x_2 + \delta x_1 x_2 + F(t)$$

where α represents the max prey per capita growth rate, β represents the effect of the presence of predators on the prey death rate, γ represents the predator per capita death rate, and δ represents the effect of the presence of prey on predator birth rate.

Time-series data for the prey and predator population are generated by numerically solving the equations for a specified input. Options are provided to add Gaussian noise to the data and visualize the noisy and noiseles time-series.

Next, the SINDyC algorithm is set up using the PySINDy library. The algorithm is configured with a finite difference differentiation method, a polunomial feature library of degree 2, and a sequential thresholded least-squares optimizer. The governing equations are represented as sparse coefficients for the candidate library functions.

The MPC controller is configured using the do_mpc library. The state variables are the prey and predator populations, and the control input is an unspecified external action.

The objective function for the MPC controller is defined as minimizing the sum of squared population rates of prey and predators, effectively aiming to stabilize the populations of both species. The MPC controller and a simulator are initialized with the same initial conditions for the prey and predator populations. A simulation loop is run for 20 time steps, where the MPC controller computes the optimal control input at each step, and the simulator updates the system state based on the control input.

Figures can be found on my repository.

References

Boyd, Stephen, and Lieven Vandenberghe. 2018. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press.

Brunton, Steven L, and J Nathan Kutz. 2022. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Vol. 2. Cambridge University Press.

Géron, Aurélien. 2022. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc.

Strang, Gilbert. 2019. *Linear Algebra and Learning from Data*. SIAM.

Sutton, Richard S, and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.