DMD, EDMD, etc.

September 1, 2020

This document is about the updates on DMD, EDMD, etc.

- Section 1: EDMD for Multiple Observables.
- Section 2: EDMD with Random Fourier Features.
- Section 3: Learning Contracting Vector Fields.
- Section 4: Connection between Kernel Regression and EDMD.
- Section 5: Randomized DMD.
- Section 6: EDMD with Reduced Basis Functions.
- Section 7: DMD Based Feature Extraction.
- Section 8: Fitting the Data Surface and Apply DMD.
- Section 9: Kernel EDMD with Regularization.
- Section 10: Learn the Eigenfunctions and Perform Prediction.
- Section 11: Possible Research Directions.
- Section 12: Modeling Cardiac Data with Multiple Observables.
- Section 13: Experiments with Burgers' Equation.
- Section 14: Helmholtz-Hodge Decomposition.
- Section 15: EDMD for Vector Fields.
- Section 16: Hierarchical Function Approximation.

1 EDMD for Multiple Observables

1.1 Formulation

Consider observables $\boldsymbol{X}_i \in \mathcal{M} \subset \mathbb{R}^{M\times C}$, where M is the number of spatial grids, and C is the number of measured quantities. Suppose there exists $f : \mathcal{M} \to \mathcal{M}$ such that

$$\boldsymbol{X}_{i+1} = f(\boldsymbol{X}_i).$$

Define the Koopman operator $\mathcal{K}$ to be acting on vector-valued function $\varphi : \mathbb{R}^{M\times C} \to \mathbb{R}^{1\times C}$, i.e.,

$$(\mathcal{K}\varphi)(\boldsymbol{X}) = \varphi \circ f(\boldsymbol{X}).$$

Thus $\mathcal{K}$ is a linear operator. Denoted $\{\mu_k\}$ and $\{\varphi_k\}$ to the the Koopman eigenvalues and eigenfunctions.

Basis Functions. Consider basis functions $\{\psi_l\}_{l=1}^{L}$. For example, the components in the feature map corresponding to a matrix-valued kernel. Denote

$$\Psi(\boldsymbol{X}) = \begin{bmatrix} \psi^{(1)}(\boldsymbol{X}) \\ \vdots \\ \psi^{(L)}(\boldsymbol{X}) \end{bmatrix} \in \mathbb{R}^{L\times C}.$$

For example, consider

$$\boldsymbol{K}(x, y) = (1 + xy)^2 \cdot \boldsymbol{A}$$

for some positive definite matrix $\boldsymbol{A}$, then

$$\Psi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x \\ x^2 \end{bmatrix} \otimes \boldsymbol{A}^{1/2}.$$

Procedure. The kernel based reformulation can be similarly done as in [**?**]. Suppose there are $(N + 1)$ snapshots observed, the observed tensor is $\mathcal{A} \in \mathbb{R}^{M\times C\times(N+1)}$ with $\boldsymbol{X}_i = \mathcal{A}^{(:,:,i)}$. Denote $\boldsymbol{A}_0 \in \mathbb{R}^{M\times(CN)}$ to be the matricization of $\mathcal{A}^{(:,:,1:N)}$. The procedure of EDMD for multiple observables, is as the following.

- Construct $\widehat{\boldsymbol{G}} \in \mathbb{R}^{(CN)\times(CN)}$, made of $N \times N$ blocks of equal size. The (i, j)-th block is

  $$\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \Psi(\boldsymbol{X}_i)^T\Psi(\boldsymbol{X}_j) \in \mathbb{R}^{C\times C}.$$

  Similarly $\widehat{\boldsymbol{A}} \in \mathbb{R}^{(CN)\times(CN)}$ is defined such that the (i, j)-th block is

  $$\boldsymbol{K}(\boldsymbol{Y}_i, \boldsymbol{X}_j) = \Psi(\boldsymbol{Y}_i)^T\Psi(\boldsymbol{X}_j) \in \mathbb{R}^{C\times C};$$

- Compute the eigendecomposition $\widehat{\boldsymbol{G}} = \boldsymbol{Q}_0\boldsymbol{\Lambda}_0\boldsymbol{Q}_0^T$;

- Construct the matrix

  $$\widehat{\boldsymbol{K}} = (\boldsymbol{Q}_0\boldsymbol{\Lambda}_0^{-1/2})^T \cdot \widehat{\boldsymbol{A}} \cdot (\boldsymbol{Q}_0\boldsymbol{\Lambda}_0^{-1/2}),$$

  and compute its eigenvalues $\{\mu_k\}$ and eigenvectors $\{\widehat{\boldsymbol{v}}_k\}$. Construct $\widehat{\boldsymbol{V}}$ such that $\widehat{\boldsymbol{V}}^{(:,k)} = \widehat{\boldsymbol{v}}_k$;

- The k-th (approximated) Koopman mode is

  $$\boldsymbol{\xi}_k = \boldsymbol{A}_0(\boldsymbol{Q}_0\boldsymbol{\Lambda}_0^{-1/2})\widehat{\boldsymbol{w}}_k,$$

  where $\widehat{\boldsymbol{w}}_k^T$ is the k-th row of $\widehat{\boldsymbol{V}}^{-1}$.

1.2 Experiments

1.2.1 GEMS

We experiment on Velocity - x and Velocity - y of the GEMS data. For $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{38523\times 2}$, the matrix kernel is chosen to be

$$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{Y}) = k(\boldsymbol{X}, \boldsymbol{Y}) \cdot \boldsymbol{I} \in \mathbb{R}^{2\times 2},$$

where $k(\boldsymbol{X}, \boldsymbol{Y})$ is a scalar kernel. For quadratic and Gaussian kernel, we record at various ranks the prediction errors for the two measures.

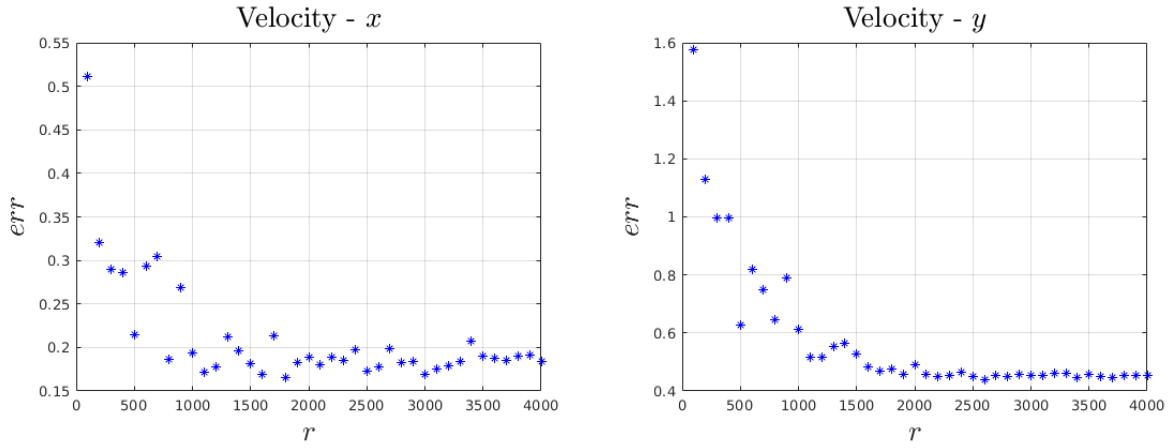

Figure 1: For quadratic kernel, prediction errors for velocities.

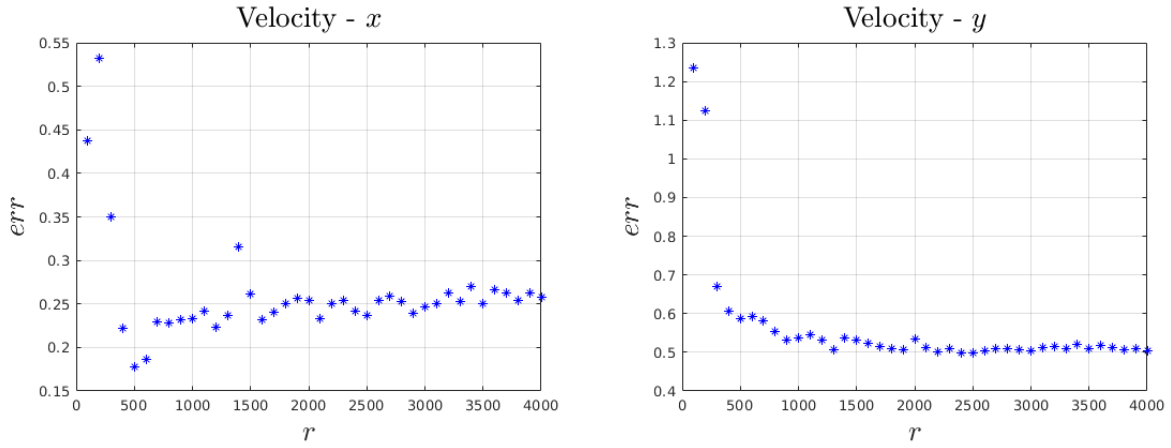

Figure 2: For Gaussian kernel, prediction errors for velocities.

Comparing to modeling the velocities separately, with (best) prediction errors (0.2000,0.4678), modeling them together using quadratic kernel shows some improvement, e.g., (0.1776,0.4396) at rank 2600. Modeling using Gaussian seem to overfit in this case.

2 EDMD with Random Fourier Features

Consider the GEMS data of measure Velocity - y, EDMD with Gaussian kernel achieves the best prediction error. The kernel matrix used in the calculation can be computed using random Fourier features. The number of features D determines the approximation accuracy. Recall that $N = 38523$ is the number of spatial grids. For different values of D, the corresponding prediction errors at different truncation ranks are shown in Figure 3, where $D = \infty$ denotes EDMD without approximation.

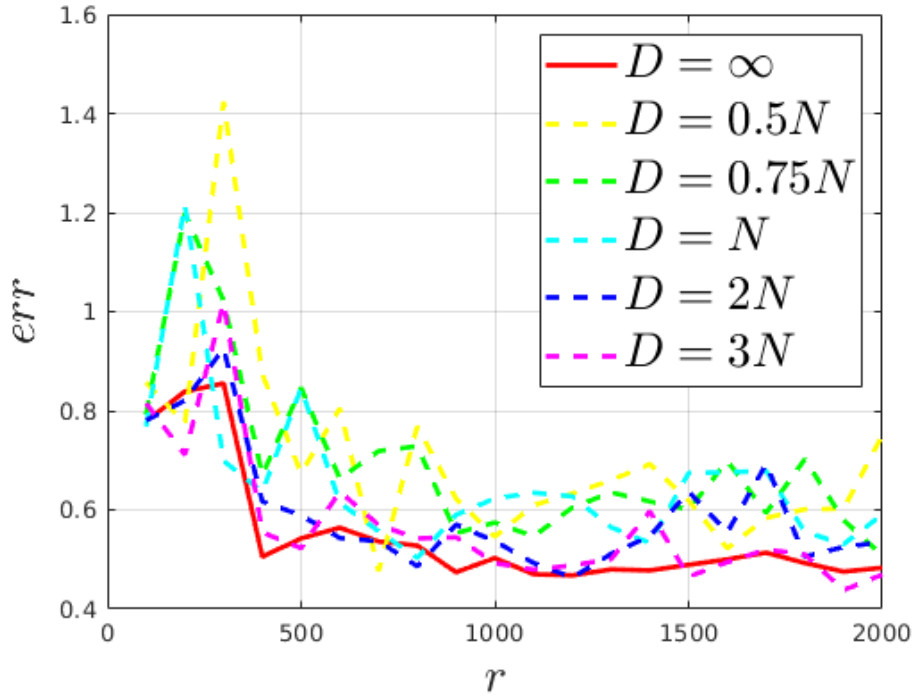

Figure 3: The prediction errors of EDMD with different number of random features on Velocity - y data.

It can be seen that when D increases, the prediction performance approaches EDMD without approximation. When $D = 3N$, the predictions may be better at a few points, but these are only caused by the approximation error of the kernel. The approximation error, acting like noise, may cause fluctuation in the prediction performance.

3 Learning Contracting Vector Fields

The paper [**?**] considers the problem of learning the function f that maps spatial locations along a trajectory to the corresponding derivatives. Such f is assumed to be in a RKHS space $\mathcal{H}_K$. The contributions are

- f is further confined to be in a RKHS space $\mathcal{H}_K^Z \subset \mathcal{H}_K$, where $Z = \{\boldsymbol{x}_*^i, i = 1 \dots k\}$ contains the desired equilibria, and
$$\mathcal{H}_K^Z = \{f \in \mathcal{H}_K : f(\boldsymbol{x}_*^i) = \mathbf{0}, \boldsymbol{x}_*^i \in Z\}.$$
Proposition 2 shows the relationship between the corresponding matrix-valued kernels of the two RKHS spaces, i.e., $\mathcal{H}_K$ and $\mathcal{H}_K^Z$.
- f is also subject to the contraction constraints, which help ensure that trajectories starting from nearby points will converge to the same end point.
- A heuristic method is proposed for solving the optimization problem based on random Fourier approximation to the matrix-valued kernels. Two types of kernels are considered, i.e., Gaussian separable kernels and curl-free kernels.

How can we view the GEMS data as trajectories? If we consider the data of velocities, then the data may be organized to

$$\{\boldsymbol{x}_t^i \in \mathbb{R}^2 \mid t = 1 \dots 10001, i = 1 \dots 38523\},$$

i.e., 38523 trajectories, each with 10001 time steps. However, the trajectories are much more complicated than the *Angle*, *CShape*, *GShape*, *JShape* considered in [**?**]. For $i = 31979$, the ground truth trajectory, and the prediction by EDMD with multiple observables (EDMD-MO) are shown in Figure 4. We can see that the predictions are accurate from $t = 1$ to $t = 8000$, and not reliable from $t = 8001$ to $t = 10001$.

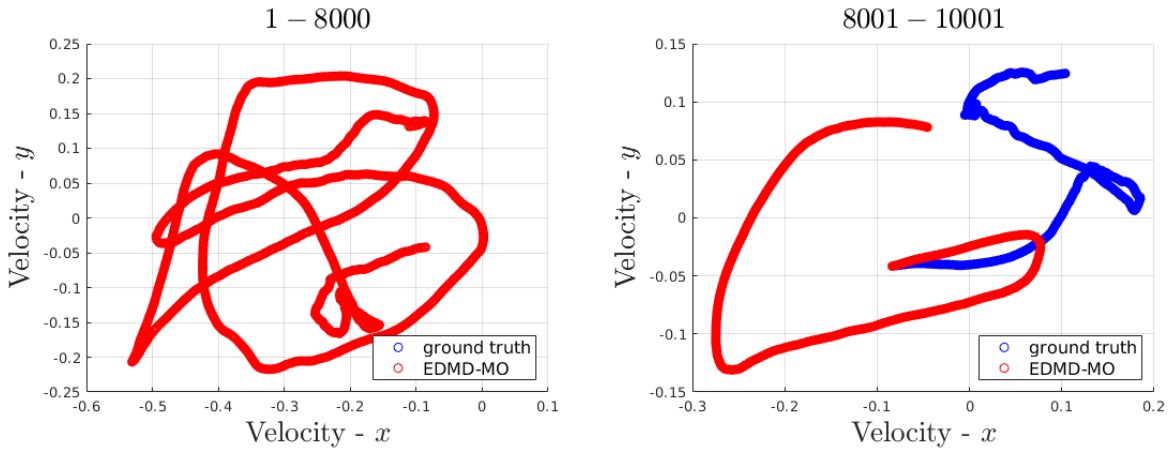

Figure 4: Ground truth trajectories about velocities at $i = 31979$, and the predition by EDMD with multiple observables (EDMD-MO).

Recall that EDMD-MO uses all the 38523 spatial grids for the training. It is then compared with regularized matrix-valued kernel regression, i.e., solving

$$\min_f \sum_{t=1}^{7999} \|f(\boldsymbol{x}_t^i) - \boldsymbol{x}_{t+1}^i\|_2^2 + \lambda \|f\|_{\mathcal{H}_K}^2, \tag{1}$$

and prediction by the learned f. We consider K to be either Gaussian with identity matrix, or the curl-free kernel. We can see from Figure 5 and 6 that curl-free kernel is more accurate for 1 step ahead prediction, and (1) is not suitable for long-term prediction.

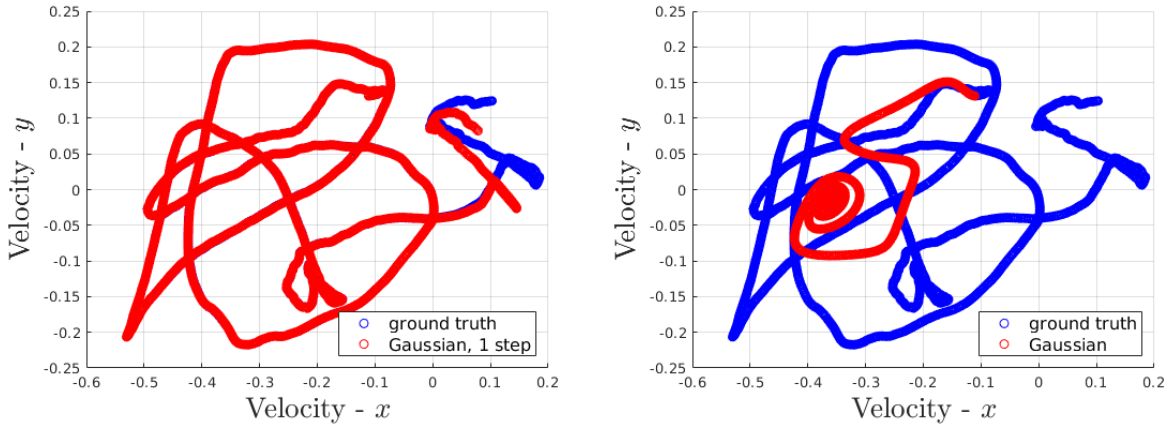

Figure 5: Result about Gaussian kernel with identity matrix. Left: 1 step ahead prediction. Right: prediction from the beginning.

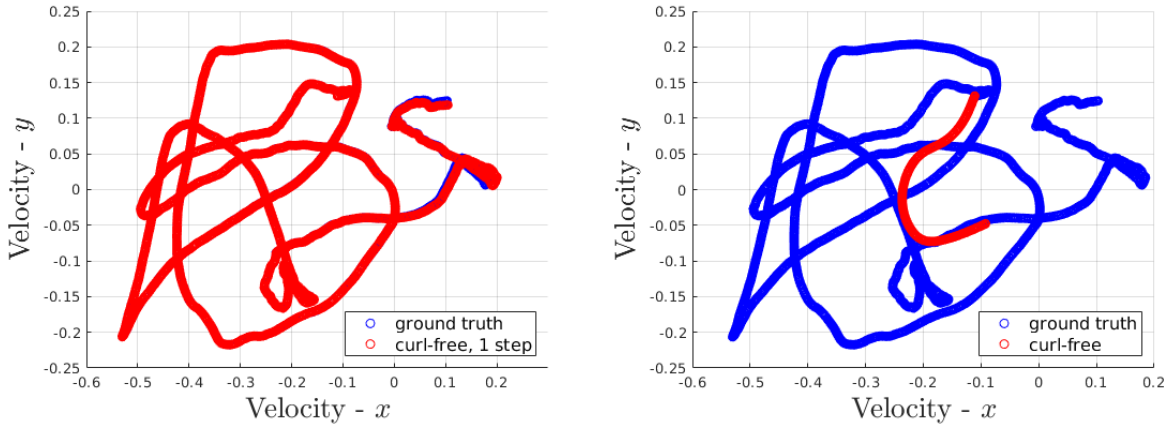

Figure 6: Result about curl-free kernel. Left: 1 step ahead prediction. Right: prediction from the beginning.

4 Connection between Kernel Regression and EDMD

Suppose $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, where $\boldsymbol{x}_i = \boldsymbol{X}^{(:,i)}$, and $\boldsymbol{Y} \in \mathbb{R}^{N\times M}$, where $\boldsymbol{y}_i = \boldsymbol{Y}^{(:,i)}$. Consider the regularized kernel regression

$$\min_{\{f_j\}} \sum_{j=1}^{N} \left(\sum_{i=1}^{M} \|f_j(\boldsymbol{x}_i) - \boldsymbol{y}_i^{(j)}\|_2^2 + \lambda_j \|f_j\|_{\mathcal{H}_K}^2 \right).$$

The EDMD formulation is

$$\min_{\boldsymbol{K}} \sum_{i=1}^{M} \|\boldsymbol{K} \cdot \psi(\boldsymbol{x}_i) - \psi(\boldsymbol{y}_i)\|_2^2.$$

Define $\hat{\boldsymbol{y}}_i = \psi(\boldsymbol{y}_i) \in \mathbb{R}^L$. Then EDMD solves

$$\min_{\boldsymbol{K}} \sum_{l=1}^{L} \left(\sum_{i=1}^{M} \|\boldsymbol{K}^{(l,:)} \cdot \psi(\boldsymbol{x}_i) - \hat{\boldsymbol{y}}_i^{(l)}\|_2^2 \right).$$

Functions $\{\boldsymbol{K}^{(l,:)} \cdot \psi\}_l$ correspond to $\{f_j\}$. If one adds regularization, the regularized EDMD becomes

$$\min_{\boldsymbol{K}} \sum_{l=1}^{L} \left(\sum_{i=1}^{M} \|\boldsymbol{K}^{(l,:)} \cdot \psi(\boldsymbol{x}_i) - \hat{\boldsymbol{y}}_i^{(l)}\|_2^2 + \lambda_l \|\boldsymbol{K}^{(l,:)} \cdot \psi\|_{\mathcal{H}_K}^2 \right).$$

However, the basis functions in ψ are not necessarily in the RKHS space.

5 Randomized DMD

The paper [**?**] proposes an approach called randomized DMD (RDMD). Suppose the data matrix is $\boldsymbol{X} = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_m] \in \mathbb{R}^{n\times m}$, and the choice of rank is r. RDMD performs the following steps to approximately calculate the exact DMD.

- Construct $\boldsymbol{\Omega} \in \mathbb{R}^{m\times k}$, and form $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$. Let $[\boldsymbol{Q}, \sim] = \text{qr}(\boldsymbol{Y})$.
- Define $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{X} \in \mathbb{R}^{k\times m}$. Split $\boldsymbol{B}$ into $\boldsymbol{B}_L = \boldsymbol{B}^{(:,1:m-1)}$ and $\boldsymbol{B}_R = \boldsymbol{B}^{(:,2:m)}$.
- Calculate the truncated SVD $[\widetilde{\boldsymbol{U}}, \boldsymbol{\Sigma}, \boldsymbol{V}] = \text{svd}(\boldsymbol{B}_L, r)$.
- Form the linear map $\widetilde{\boldsymbol{A}}_{\boldsymbol{B}} = \widetilde{\boldsymbol{U}}^*(\boldsymbol{B}_R\boldsymbol{V}\boldsymbol{\Sigma}^{-1}) \in \mathbb{R}^{r\times r}$.
- Calculate the eigen-decomposition of $[\widetilde{\boldsymbol{W}}_{\boldsymbol{B}}, \boldsymbol{\Lambda}] = \text{eig}(\widetilde{\boldsymbol{A}}_{\boldsymbol{B}})$.
- The DMD modes are $\boldsymbol{W} = \boldsymbol{Q}(\boldsymbol{B}_R\boldsymbol{V}\boldsymbol{\Sigma}^{-1})\widetilde{\boldsymbol{W}}_{\boldsymbol{B}}$.

SketchyCoreDMD is compared with RDMD on a dataset about fluid flow behind a cylinder on the 2D spatial grids of size 499×199, across 151 time steps, thus $\boldsymbol{X} \in \mathbb{R}^{89351\times 151}$. According to the parameter choice in [**?**], $(r, k) = (15, 25)$. For SketchyCoreDMD, we try $(r, k, s, p) = (15, 25, 51, 0.35)$ and $(r, k, s, p) = (15, 31, 63, 0.65)$. The reported times in Table 1 and the the Koopman eigenvalues shown in Figure 7 are averaged over 20 random tests.

Table 1: Comparisons of computation time.

	DMD	RDMD	SketchyCoreDMD	
time	0.7340	0.0533	0.1498	0.3189

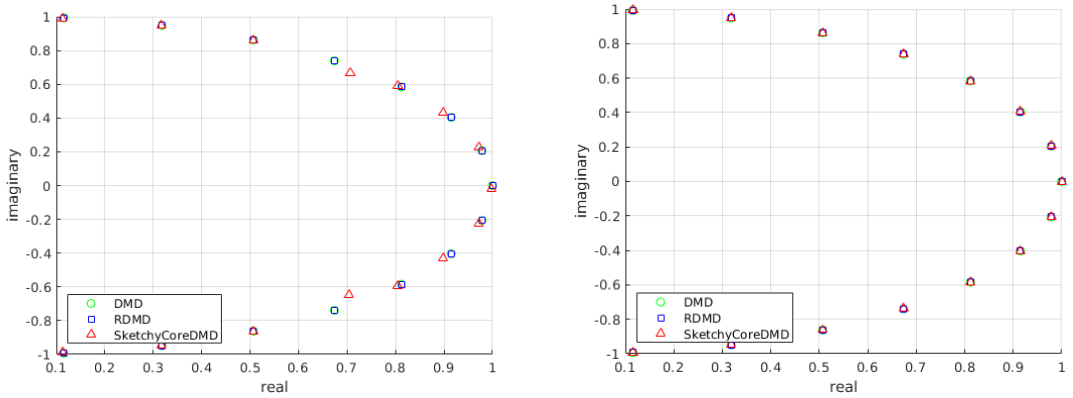

Figure 7: Comparisons of first 15 Koopman eigenvalues. Left: $(r, k, s, p) = (15, 25, 51, 0.35)$ for SketchyCoreDMD; Right: $(r, k, s, p) = (15, 31, 63, 0.65)$ for SketchyCoreDMD

With the same r and k, RDMD is not only faster, but also more accurate. To reach the same accuracy, SketchyCoreDMD needs a larger k.

6 EDMD with Reduced Basis Functions

We consider the original EDMD [**?**], instead of the kernel variant [**?**]. The basis functions are assumed to come from a subset of the basis functions in the feature map of a scalar kernel. The original EDMD performs the following steps.

- The stack of basis functions evaluated at $\boldsymbol{x}$ is denoted by

$$\boldsymbol{\psi}(\boldsymbol{x}) = \begin{bmatrix} \psi_1(\boldsymbol{x}) \\ \vdots \\ \psi_K(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^K.$$

  Assembly the matrices

$$\boldsymbol{\Psi}_{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{\psi}(\boldsymbol{x}_M)^T \end{bmatrix} \in \mathbb{R}^{M \times K} \quad \text{and} \quad \boldsymbol{\Psi}_{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{y}_1)^T \\ \vdots \\ \boldsymbol{\psi}(\boldsymbol{y}_M)^T \end{bmatrix} \in \mathbb{R}^{M \times K}.$$

- Compute the linear map $\boldsymbol{K} = \boldsymbol{\Psi}_{\boldsymbol{x}}^{\dagger} \boldsymbol{\Psi}_{\boldsymbol{y}} \in \mathbb{R}^{K \times K}$.
- Compute the eigendecomposition of $\boldsymbol{K}$ to get eigenvectors $\{\boldsymbol{v}_k\}$, and Koopman eigenvalues $\{\mu_k\}$.
- The k-th approximated Koopman eigenfunction is $\varphi_k(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{x})^T \boldsymbol{v}_k$. Define $\boldsymbol{\Phi}_{\boldsymbol{x}} = \boldsymbol{\Psi}_{\boldsymbol{x}} \boldsymbol{V}$.
- Find a matrix $\boldsymbol{\Xi}$ such that $\boldsymbol{X} = \boldsymbol{\Phi}_{\boldsymbol{x}} \boldsymbol{\Xi}$, where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_M^T \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\xi}_1^T \\ \vdots \\ \boldsymbol{\xi}_K^T \end{bmatrix}.$$

- The prediction of $\boldsymbol{x}$ is computed as

$$\boldsymbol{y} = \sum_k (\mu_k \varphi_k(\boldsymbol{x})) \cdot \boldsymbol{\xi}_k.$$

For a proof of concept, we test on the *Angle* shape (2D trajectory of 1000 time steps) from the LASA dataset [**?**]. The basis functions are from the feature map of a quadratic kernel. They are:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1 x_2 & x_2 & x_2^2 \end{bmatrix}.$$

With 1000 or 900 time steps for training, the predictions of EDMD are shown in Figure 8. In the case of 900 time steps, the predictions are not good. Next, consider reduced basis functions

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_2 & x_2^2 \end{bmatrix}.$$

With 1000 or 900 time steps for training, the predictions of EDMD are shown in Figure 9. We can see that the prediction are closer to the ground truth for reduced basis functions.

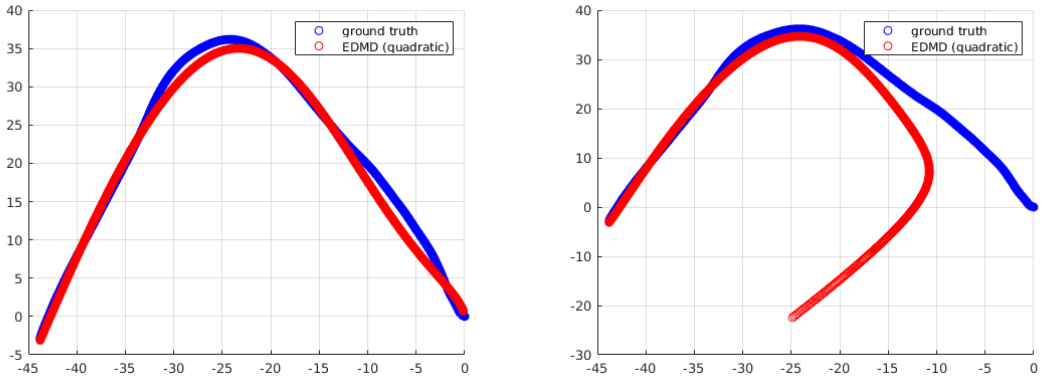

Figure 8: For EDMD with full basis functions, prediction from training using 1000 time steps (Left) and first 900 time steps (Right).

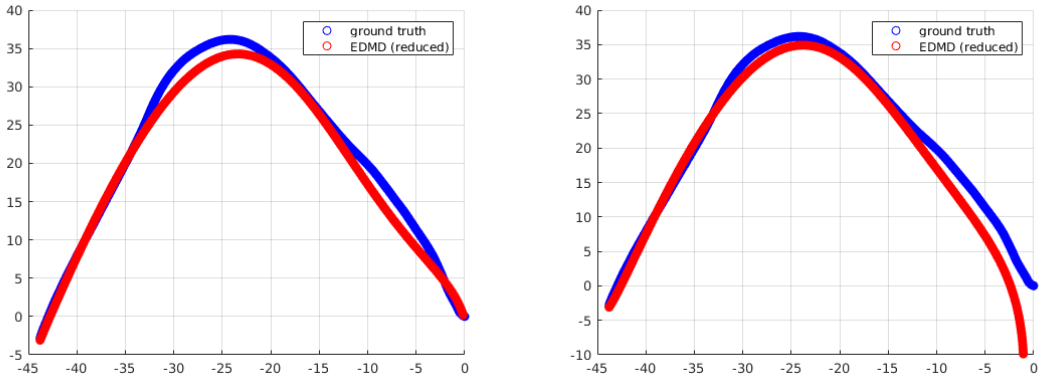

Figure 9: For EDMD with reduced basis functions, prediction from training using 1000 time steps (Left) and first 900 time steps (Right).

7 DMD Based Feature Extraction

Suppose we have a paired data $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, and $\boldsymbol{Y} \in \mathbb{R}^{N\times M}$. Let $\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{X}^\dagger \in \mathbb{R}^{N\times N}$ be the optimal solution to

$$\|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{Y}\|_F^2.$$

DMD finds the eigenvalues and eigenvectors of $\boldsymbol{A}$. For the case that N is much larger than M, it is not wise to directly compute $\boldsymbol{Y}\boldsymbol{X}^\dagger$.

Note that $\boldsymbol{X}$ is usually of low-rank. Denote $[[\boldsymbol{X}]]_K = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ to be the best rank K approximation to $\boldsymbol{X}$, and suppose $\boldsymbol{X} \approx [[\boldsymbol{X}]]_K$. We need to find the non-zero eigenvalues and the corresponding eigenvectors of $\boldsymbol{A}$. To do so, $\boldsymbol{A}$ is transformed to

$$\widetilde{\boldsymbol{A}} = \boldsymbol{U}^*\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}^*\boldsymbol{Y}\boldsymbol{X}^\dagger\boldsymbol{U} = \boldsymbol{U}^*\boldsymbol{Y}(\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^*)\boldsymbol{U} = \boldsymbol{U}^*\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1} \in \mathbb{R}^{K\times K}.$$

Denote the eigendecomposition of $\widetilde{\boldsymbol{A}}$ to be $\widetilde{\boldsymbol{W}}\boldsymbol{\Lambda}\widetilde{\boldsymbol{W}}^{-1}$. Here $\boldsymbol{\Lambda}$ contains the eigenvalues $\{\lambda_k\}_{k=1}^K$, which are proven to be the non-zero eigenvalues of $\boldsymbol{A}$. There are different ways to get the corresponding eigenvectors of $\boldsymbol{A}$ from $\widetilde{\boldsymbol{W}}$.

- **Standard DMD.** Refer to [**?**], note that

  $$\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{w}}_k = \lambda_k\widetilde{\boldsymbol{w}}_k \Rightarrow \boldsymbol{U}\boldsymbol{U}^*\boldsymbol{A}(\boldsymbol{U}\widetilde{\boldsymbol{w}}_k) = \lambda_k(\boldsymbol{U}\widetilde{\boldsymbol{w}}_k) \approx \boldsymbol{A}(\boldsymbol{U}\widetilde{\boldsymbol{w}}_k) = \lambda_k(\boldsymbol{U}\widetilde{\boldsymbol{w}}_k),$$

  where $(\boldsymbol{U}\widetilde{\boldsymbol{w}}_k)$ is the k-th computed eigenvector. The last approximation is due to

  $$\boldsymbol{U}\boldsymbol{U}^*\boldsymbol{A} = \boldsymbol{U}\boldsymbol{U}^*\boldsymbol{Y}\boldsymbol{X}^\dagger \approx \boldsymbol{Y}\boldsymbol{X}^\dagger = \boldsymbol{A},$$

  where $\boldsymbol{U}\boldsymbol{U}^*\boldsymbol{Y} \approx \boldsymbol{Y}$ since $\boldsymbol{U}\boldsymbol{U}^*\boldsymbol{X} \approx \boldsymbol{X}$, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ have similar column spaces. Denote $\boldsymbol{W} = \boldsymbol{U}\widetilde{\boldsymbol{W}}$ to be the horizontal stack of the computed eigenvectors.

- **Exact DMD.** Refer to [**?**], note that

  $$\begin{aligned}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{w}}_k = \lambda_k\widetilde{\boldsymbol{w}}_k \Rightarrow \boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{w}}_k &= \lambda_k(\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{w}}_k)\\ &= (\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^*)\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{w}}_k = \lambda_k(\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{w}}_k)\\ &\approx \boldsymbol{A}(\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{w}}_k) = \lambda_k(\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{w}}_k).\end{aligned}$$

  The k-th eigenvector is chosen to be $\frac{1}{\lambda_k}\boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{w}}_k$. Denote $\boldsymbol{W} = \boldsymbol{Y}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{W}}\boldsymbol{\Lambda}^{-1}$ to be the horizontal stack of the computed eigenvectors.

Suppose $\boldsymbol{X} = \boldsymbol{D}^{(:,1:M)}$, and $\boldsymbol{Y} = \boldsymbol{D}^{(:,2:M+1)}$ for some data matrix $\boldsymbol{D}$. Given the eigenvectors $\boldsymbol{W} \in \mathbb{C}^{N\times K}$, find the vector of coefficients $\boldsymbol{b} \in \mathbb{C}^K$ such that

$$\boldsymbol{D}^{(:,1)} \approx \boldsymbol{W}\boldsymbol{b}.$$

Then the prediction of $\boldsymbol{D}^{(:,2)}$ is

$$\boldsymbol{D}^{(:,2)} \approx \boldsymbol{A}\boldsymbol{D}^{(:,1)} \approx \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{T}\boldsymbol{W}\boldsymbol{b} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{b},$$

where $\boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{T}$ is the partial eigendecomposition of $\boldsymbol{A}$ corresponding to the eigenvalues in $\boldsymbol{\Lambda}$. Inductively,

$$\boldsymbol{D}^{(:,m)} \approx \boldsymbol{W}\boldsymbol{\Lambda}^{(m-1)}\boldsymbol{b}.$$

Thus the data representation from the learnt Koopman modes and Koopman eigenvalues can be written as

$$\begin{bmatrix} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_K \end{bmatrix} \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_K \end{bmatrix} \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^M \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \lambda_K & \cdots & \lambda_K^M \end{bmatrix}. \tag{2}$$

Suppose $\boldsymbol{D}$ is a video about sport actions. The approximated Koopman modes $\{\boldsymbol{w}_k\}_{k=1}^K$ are about different motions. The approximated Koopman eigenvalues $\{\lambda_k\}_{k=1}^K$ describe the frequencies of the motions. The eigenvalues on the unit circle correspond to some recurrent motions, and the eigenvalues inside the unit circle corresponds to some "fast" motions presented in fewer video frames.

- Is the decomposition (2) unique? Note that the right matrix is a Vandermonde matrix.
- Although computed differently, the data representation of EDMD from the learnt Koopman modes and Koopman eigenvalues can also be written in the form of (2).
- How to classify, e.g., videos of sport actions based on the computed Koopman modes and eigenvalues? The paper [**?**] considers such a problem and define the distances using the Binet-Cauchy kernel developed in [**?**].

8 Fitting the Data Surface and Apply DMD

The data is usually recorded on 2D or 3D grid points. The idea is to interpolate the data using some local basis functions, and apply DMD to the coefficients of the basis functions.

8.1 Experiments on GEMS Data

The idea with bilinear and biquadratic interpolations has been tried on the data of the flow behind a cylinder. DMD suffices for this data, and these new methods work as well. In order to apply the idea to GEMS dataset, we have to deal with scattered data points not on the grids, as shown in Figure 10.

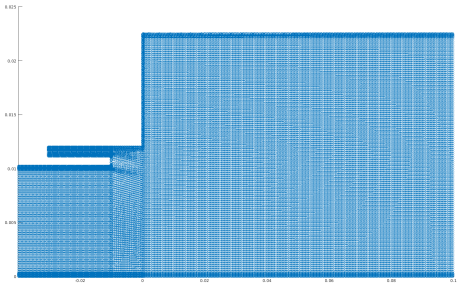

Figure 10: Scattered data points of the GEMS dataset.

We choose to work with evenly distributed grid points, as shown in Figure 11. Denser grid points are put along the boundaries. The values at those grid points are computed using Delaunay triangulation and the natural neighbor interpolation method [**?**].

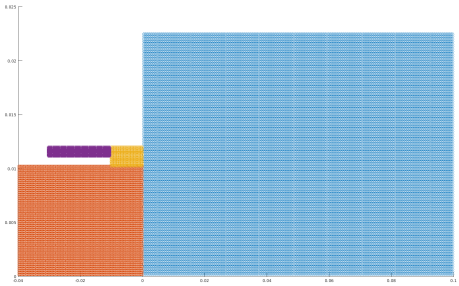

Figure 11: Evenly distributed grid points for the GEMS dataset.

However, experiments with the interpolated data produce more errors.

8.2 Experiments on Cardiac Data

We then tried interpolate every frame of the Cardiac data using B-splines, apply DMD to the coefficients of the B-splines, and interpolate the data back. The following figure shows the results of cubic and fifth order B-splines, compared with the results of DMD, and EDMD with different kernels.

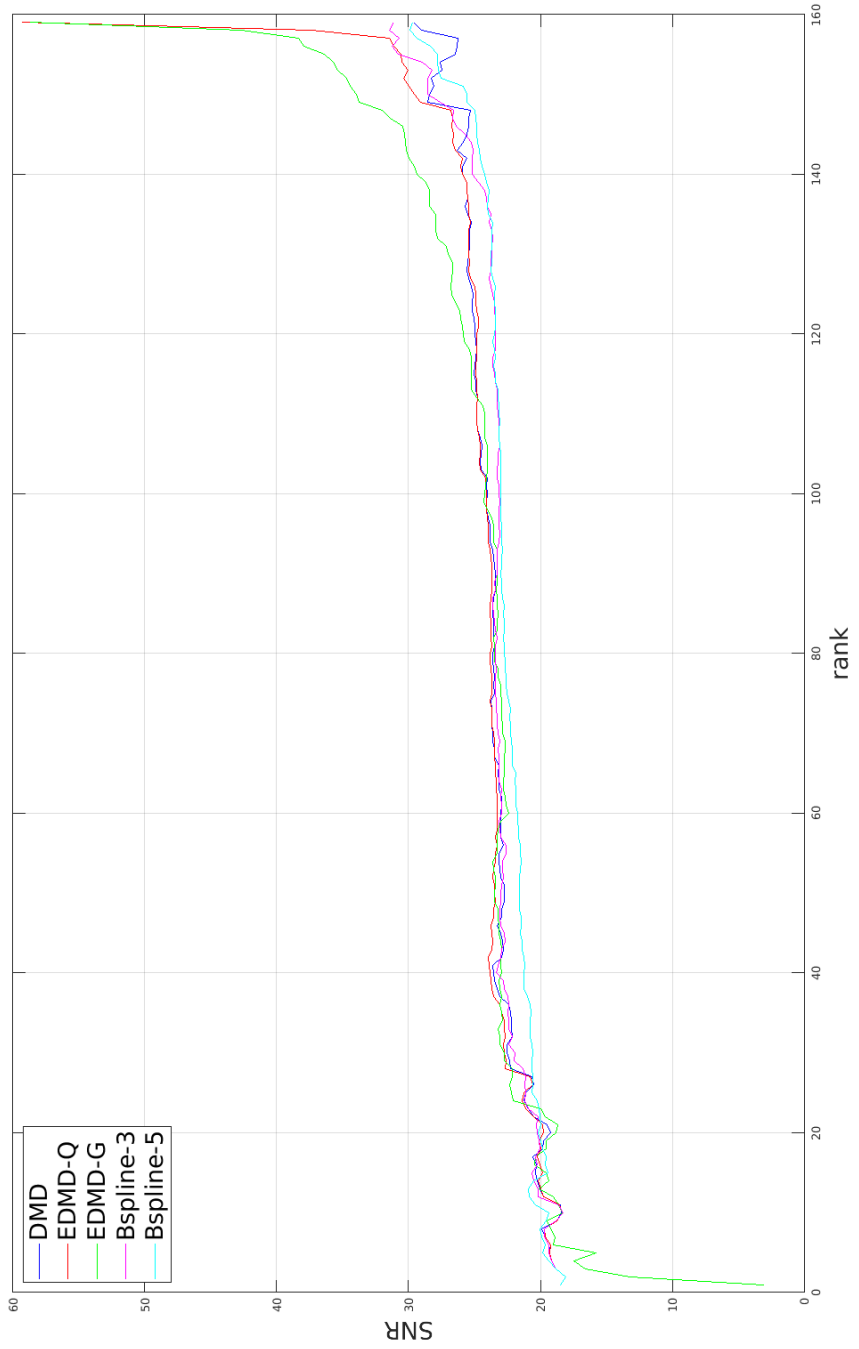

Figure 12: Comparisons of approximation accuracy at different ranks.

9 Kernel EDMD with Regularization

Suppose $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, where $\boldsymbol{x}_i = \boldsymbol{X}^{(:,i)}$, and $\boldsymbol{Y} \in \mathbb{R}^{N\times M}$, where $\boldsymbol{y}_i = \boldsymbol{Y}^{(:,i)}$. Let $\psi : N \to N_k$ be the feature map. Denote

$$\boldsymbol{\Phi}_x = \begin{bmatrix} \psi(\boldsymbol{x}_1)^T \\ \vdots \\ \psi(\boldsymbol{x}_M)^T \end{bmatrix} \in \mathbb{R}^{M\times N_k} \quad \text{and} \quad \boldsymbol{\Phi}_y = \begin{bmatrix} \psi(\boldsymbol{y}_1)^T \\ \vdots \\ \psi(\boldsymbol{y}_M)^T \end{bmatrix} \in \mathbb{R}^{M\times N_k}.$$

Consider the following minimization

$$\|\boldsymbol{K}\boldsymbol{\Phi}_x - \boldsymbol{\Phi}_y\|_F^2 + \alpha\|\boldsymbol{K}\|_F^2.$$

The closed-form solution is

$$\boldsymbol{K} = (\boldsymbol{\Phi}_x\boldsymbol{\Phi}_x^T + \alpha\boldsymbol{I})^{-1} \cdot (\boldsymbol{\Phi}_y\boldsymbol{\Phi}_x^T) \in \mathbb{R}^{M\times M},$$

where $(\boldsymbol{\Phi}_x\boldsymbol{\Phi}_x^T) \in \mathbb{R}^{M\times M}$ and $(\boldsymbol{\Phi}_y\boldsymbol{\Phi}_x^T) \in \mathbb{R}^{M\times M}$ are two kernel matrices. However, this minimization is meaningless. The regularized version of EDMD is to minimize

$$\|\boldsymbol{\Phi}_x\boldsymbol{K} - \boldsymbol{\Phi}_y\|_F^2 + \alpha\|\boldsymbol{K}\|_F^2.$$

Such regularization favors simple $\boldsymbol{K}$, and it is not related to regularization in the RKHS space. We do not have a formulation in the RKHS space, see Section 4 of this document. The closed-form solution is

$$\boldsymbol{K} = (\boldsymbol{\Phi}_x^T\boldsymbol{\Phi}_x + \alpha\boldsymbol{I})^{-1} \cdot (\boldsymbol{\Phi}_x^T\boldsymbol{\Phi}_y) \in \mathbb{R}^{N_k\times N_k}. \tag{3}$$

In general, $N_k \gg M$. For example, if ψ corresponds to the feature map of a quadratic polynomial kernel, $N_k = \frac{N^2+3N+2}{2}$. $\boldsymbol{K}$ in (3) is rank-deficient since

$$\text{rank}(\boldsymbol{K}) \le \text{rank}(\boldsymbol{\Phi}_x) = r \le M.$$

We need to find the eigenvalues and eigenvectors of $\boldsymbol{K}$. Since N_k is very large, we should not work on $\boldsymbol{K}$ directly. Similar as [**?**, Propostion 1], we can prove the following.

Lemma 1. *Let the (truncated) SVD of $\boldsymbol{\Phi}_x$ be*

$$\boldsymbol{\Phi}_x = \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Z}^T,$$

where $\boldsymbol{Q} \in \mathbb{R}^{M\times r}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$, and $\boldsymbol{Z} \in \mathbb{R}^{N_k\times r}$. 1) If μ and $\hat{\boldsymbol{v}}$ are an eigenvalue and eigenvector of

$$\hat{\boldsymbol{K}} = (\boldsymbol{\Sigma}^2 + \alpha\boldsymbol{I})^{-1}\boldsymbol{\Sigma}\boldsymbol{Q}^T \cdot \boldsymbol{\Phi}_y\boldsymbol{\Phi}_x^T \cdot \boldsymbol{Q}\boldsymbol{\Sigma}^{-1} \in \mathbb{R}^{r\times r},$$

then μ and $\boldsymbol{Z}\hat{\boldsymbol{v}}$ are an eigenvalue and eigenvector of $\boldsymbol{K}$; 2) if $\mu \neq 0$ and $\boldsymbol{v}$ are an eigenvalue and eigenvector of $\boldsymbol{K}$, then μ and $\boldsymbol{Z}^T\boldsymbol{v}$ are an eigenvalue and eigenvector of $\hat{\boldsymbol{K}}$.

From this lemma, we can find the non-zero eigenvalues and the corresponding eigenvectors of $\boldsymbol{K}$ through computing with $\hat{\boldsymbol{K}}$.

10 Learn the Eigenfunctions and Perform Prediction

[**?**] provides a method to learn the eigenfunctions and perform prediction. We try to understand the method using the example of classical Van der Pol oscillator.

$$\begin{aligned}\dot{x}_1 =& 2x_2\\ \dot{x}_2 =& -0.8x_1 + 2x_2 - 10x_1^2x_2\end{aligned}$$

The authors learn 10 eigenfunctions for each variable. There are in total 20 eigenfunctions to be learned.

10.1 Training Data

100 different trajectories of length 501 are generated. The sampling time is $T_s = 0.01$ second. The initial conditions of the trajectories are sampled uniformly over a circle of radius 0.05. Denote the dataset as $\{\boldsymbol{x}_k^j\}$, where $k \in \{0, \ldots, 500\}$, and $j \in \{1, \ldots, 100\}$.

10.2 Learn the Eigenvalues

Stack the trajectories horizontally, and apply dynamic mode decomposition to get two eigenvalues, which can be expressed as

$$e^{\lambda T_s} \quad \text{and} \quad e^{\bar{\lambda} T_s}$$

for some λ. Define the following set

$$\Lambda = \{a_1\lambda + a_2\bar{\lambda} \mid a_1 + a_2 \leq 3,\ a_1, a_2 \in \mathbb{N}\},$$

which contains 10 distinct values. The values in Λ are used as the initial approximation to the eigenvalues. We are going to separately refine the 10 eigenvalues for each variable. Denote the eigenvalues to be optimized for the first variable as $\{\lambda_{1,j}\}_{j=1}^{10}$. Denote the collection of trajectories about the first variable as

$$\boldsymbol{X}_1 \in \mathbb{R}^{501\times 100}.$$

The optimal $\{\lambda_{1,j}\}_{j=1}^{10}$ should satisfy

$$\text{Range}\left(\begin{bmatrix} 1 & \cdots & 1 \\ e^{\lambda_{1,1}T_s} & \cdots & e^{\lambda_{1,10}T_s} \\ \vdots & \cdots & \vdots \\ (e^{\lambda_{1,1}T_s})^{500} & \cdots & (e^{\lambda_{1,10}T_s})^{500} \end{bmatrix}\right) := \text{Range}(\boldsymbol{V}_1) \approx \text{Range}(\boldsymbol{X}_1).$$

To enforce such property, one needs to solve a non-convex optimization. The authors propose a gradient type method that locally optimize $\{\lambda_{1,j}\}_{j=1}^{10}$ around the initial values. Denote the collection of trajectories about the second variable as

$$\boldsymbol{X}_2 \in \mathbb{R}^{501\times 100}.$$

The optimal $\{\lambda_{2,j}\}_{j=1}^{10}$ should satisfy

$$\text{Range}\left(\begin{bmatrix} 1 & \cdots & 1 \\ e^{\lambda_{2,1}T_s} & \cdots & e^{\lambda_{2,10}T_s} \\ \vdots & \cdots & \vdots \\ (e^{\lambda_{2,1}T_s})^{500} & \cdots & (e^{\lambda_{2,10}T_s})^{500} \end{bmatrix}\right) := \text{Range}(\boldsymbol{V}_2) \approx \text{Range}(\boldsymbol{X}_2).$$

The eigenvalues $\{\lambda_{2,j}\}_{j=1}^{10}$ are optimized similarly.

10.3 Define the Eigenfunctions

For the learned eigenvalues, suppose

$$\left\| \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \end{bmatrix} - \begin{bmatrix} \boldsymbol{V}_1 & \\ & \boldsymbol{V}_2 \end{bmatrix} \boldsymbol{G} \right\|_F^2$$

is the smallest for $\boldsymbol{G} \in \mathbb{C}^{20\times 100}$, which contains the corresponding optimal initial values for the trajectories. Rename the eigenvalues $\{\lambda_{1,j}\}_{j=1}^{10}$ and $\{\lambda_{2,j}\}_{j=1}^{10}$ as

$$\lambda_1, \cdots, \lambda_{10}, \lambda_{11}, \cdots, \lambda_{20}.$$

Then 20 eigenfunctions $\{\phi_i\}_{i=1}^{20}$ are defined along the trajectories corresponding to the initial values, i.e.,

$$\phi_i(\boldsymbol{x}_k^j) = e^{\lambda_i T_s k} \boldsymbol{G}(i,j).$$

For the other values in the state space, the authors told us to use linear approximation based on the values defined on $\{\boldsymbol{x}_k^j\}$.

10.4 Prediction

For each trajectory, perturb each snapshot with a random noise uniformly distributed over $[-0.05, 0.05] \times [-0.05, 0.05]$. Each eigenfunction can be evaluated at the perturbed data. Now with all the perturbed snapshots from all the trajectories, compute a matrix $C \in \mathbb{C}^{2\times 20}$ that brings the values of the eigenfunctions back to values in the state space. Then for any initial point $\boldsymbol{x}_0$, its prediction at time t is

$$\boldsymbol{C} \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_{20} t} \end{bmatrix} \begin{bmatrix} \phi_1(\boldsymbol{x}_0) \\ \vdots \\ \phi_{20}(\boldsymbol{x}_0) \end{bmatrix}.$$

11 Possible Research Directions

Some possible research directions are discussed below.

11.1 Faster Computation

- For kernel based EDMD, the combination with random Fourier features can be demonstrated on the dataset about Burger equation. Unlike GEMS, this dataset can be modeled well by EDMD using small ranks.
- The only approach to compute tensor based (E)DMD [**?**] is via a sequence of CUR decompositions with some volume preserving sampling strategy. Maybe one could design a better algorithm that outperforms the current one.

11.2 Meaningful Feature Extraction

The approximation of (E)DMD to a dataset can be written as $\boldsymbol{W}\boldsymbol{B}\boldsymbol{V}^T$, see Section 7. When applied to cardiac or sport videos, the columns of $\boldsymbol{W}$ usually lack interpretability, which is bad for the purpose of feature extraction. One possible way to achieve meaningful feature extraction is to enforce some constraints on $\boldsymbol{W}$, such as orthogonality, sparsity, etc. Let $R(\cdot)$ be a regularizer, the new problem to solve is

$$\min_{\boldsymbol{W},\boldsymbol{B},\boldsymbol{V}} \|(\boldsymbol{W}\boldsymbol{B}\boldsymbol{V}^T)\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + R(\boldsymbol{W}) \quad \text{s.t.} \quad \boldsymbol{B} \text{ is diagonal, } \boldsymbol{V} \text{ is a Vandermonde matrix.}$$

The problem is nonconvex, and we could use the results from (E)DMD as initialization.

11.3 Better Prediction

There are two main problems in better prediction. One is whether the eigenfunctions are rich enough to model the evolution of the dynamics. For GEMS modeled by EDMD with Gaussian kernel, even one-step prediction has errors that are not negligible. This suggests that the eigenfunctions learned are not rich enough. The most recent paper about learning invariant set is [**?**].

Given rich enough basis functions, the other problem is to prevent overfitting using regularization. In the computation of EDMD, the first overfitting happens at forming eigenfunctions from basis functions. Another overfitting may happen during computing the Koopman modes.

12 Modeling Cardiac Data with Multiple Observables

We test on Cardiac videos. In this datasets, most videos have 30 frames. The video we test on is of size $216 \times 256 \times 30$. We first see how well DMD, EDMD-Q, and EDMD-G can model the video with different number of (approximated) eigenfunctions r. We see from the figure below that the video can be modeled well with $r = 15$. Also, EDMD is better than DMD, and EDMD-Q is better than EDMD-G.

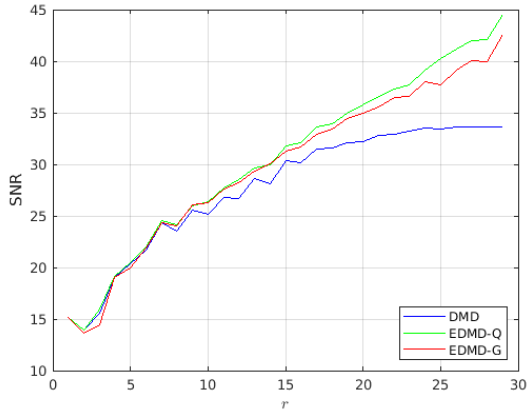

Figure 13: Performances of (E)DMD in modeling the original video.

In order to infer velocities from the video, we use the Lucas-Kanade method [**?**]. We first try to model the velocity-x (horizontal) and velocity-y (vertical) separately, which are both of size $216 \times 256 \times 29$. We see from the figure below that EDMD is better than DMD, and EDMD-G is better than EDMD-Q. However, the quality of approximation is not very good since the SNR is less than 20.

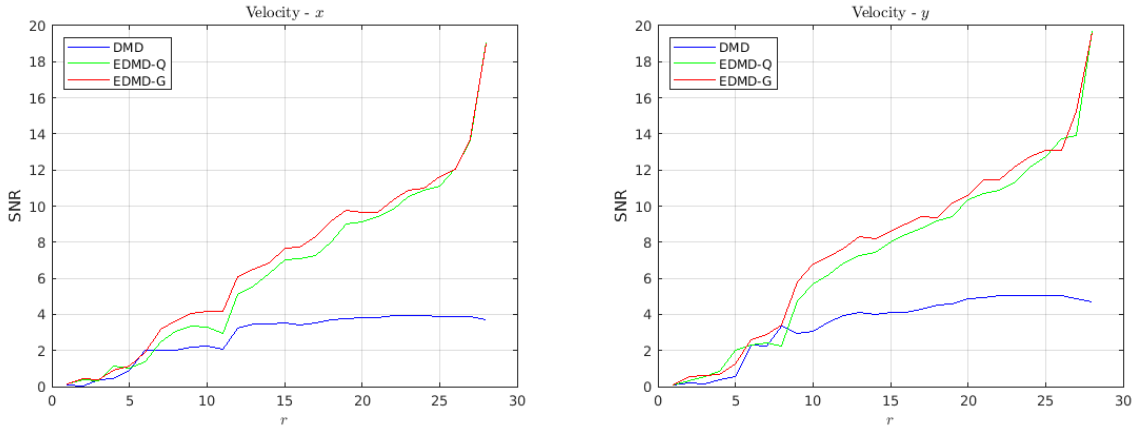

Figure 14: Performances of (E)DMD in modeling the velocities separately.

The velocities are harder to model than pixel intensities. We try to model them together using EDMD with multiple observables. The matrix kernel is the product of a scalar kernel multiplied by the 2×2 identity matrix. However, we see from the figure below that there is no gain than modeling separately.

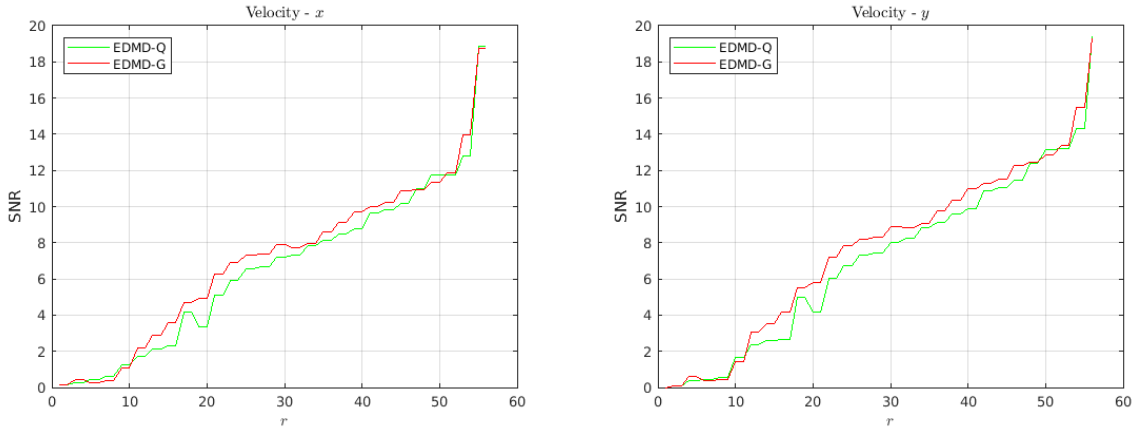

Figure 15: Performances of EDMD in modeling the velocities together.

Why modeling together is not better than modeling separately for Cardiac videos? One possible reason is the matrix kernels we use are not good enough.

13 Experiments with Burgers' Equation

We consider the Burgers's equation studied in [**?**]. Suppose the spatial domain is $[0, 1]$, and the time domain is also $[0, 1]$. The considered viscous Burgers' equation is

$$\frac{\partial}{\partial t}x(\omega, t) = -x(\omega, t)\frac{\partial}{\partial \omega}x(\omega, t) + 0.1\frac{\partial^2}{\partial \omega^2}x(\omega, t),$$

where ω is the spatial location. Dirichlet boundary conditions are imposed such that

$$x(0, t) = u(t) \quad \text{and} \quad x(1, t) = -u(t)$$

for some function $u(t)$. Discretization with the finite difference method on an equidistant grid with mesh width 2^{-7} leads to the system of parametrized nonlinear ODEs

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{F}\boldsymbol{x}^2(t) + \boldsymbol{B}u(t),$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{129\times 1}$. The initial condition is $\boldsymbol{x}(0) = \boldsymbol{0}$. The system of ODEs is solved using the **semi-implicit Euler scheme** with time step size $\delta t = 10^{-4}$, and the values of the function u on these time steps are randomly generated. The output data is of size 129×10000, and the performances of DMD and EDMD are shown in the following figure.

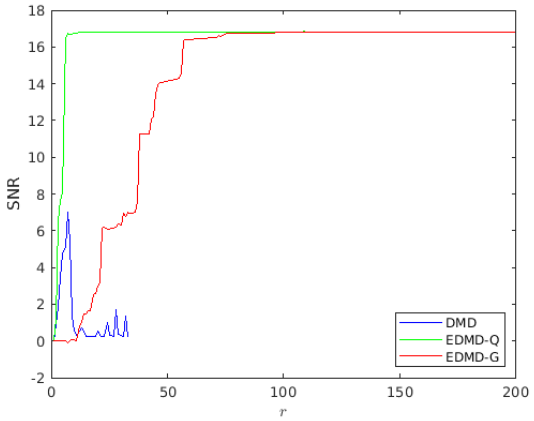

Figure 16: Performances of (E)DMD in modeling Burgers' equation.

DMD and EDMD-Q both blow up after what are shown in the figure. EDMD-G tries to model using more (approximated) eigenfunctions, but these do not help. EDMD-Q performs better than EDMD-G since it quickly reaches the smallest error. However, the quality of approximation is not very good. One possible reason is that the semi-implicit Euler scheme is not good enough.

14 Helmholtz-Hodge Decomposition

We test on Cardiac videos. In this datasets, most videos have 30 frames. The video we test on is of size $216 \times 256 \times 30$. In order to infer velocities from the video, we use the Lucas-Kanade method [**?**] implemented in MATLAB. We then get a sequence of 29 velocity fields. For each velocity field, we apply the Helmholtz-Hodge Decomposition (HHD) [**?**] to get the divergence-free (incompressible), curl-free (irrational), and harmonic parts. The code of [**?**] is in Python. For the 21st velocity filed, we visualization the magnitude of the original flow and its decomposition as the following.

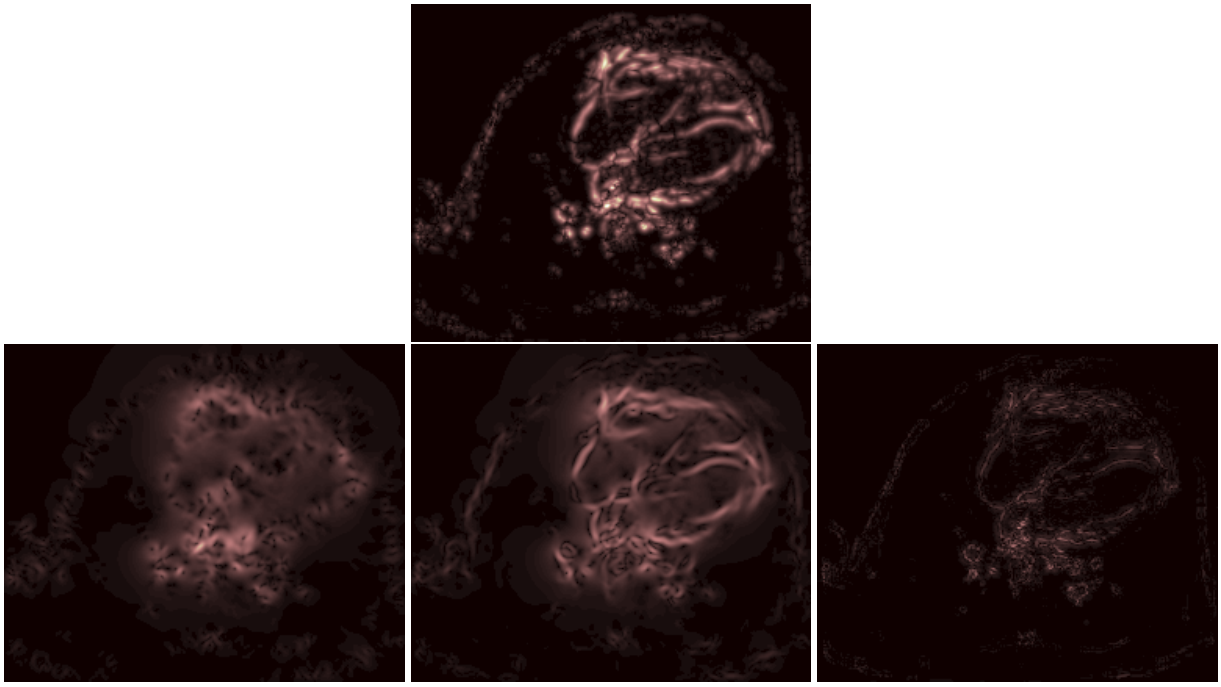

Figure 17: Original flow (top), divergence-free (bottom, left), curl-free (bottom, middle), and harmonic (bottom, right) parts.

We then try to apply EDMD with suitable kernels to the divergence-free, and curl-free parts.

15 EDMD for Vector Fields

As shown in the previous section, we can decompose a sequence of vector fields into three parts. We then try to model the divergence-free, and curl-free parts separately. We would like to use the basis functions from divergence-free and curl-free matrix kernels. The previous approach in section 1, EDMD for multiple observables, does not allow us to use matrix kernels other than a scalar kernel multiplied by a PSD matrix. The following is the derivation of a modified approach that allows one to use divergence-free and curl-free matrix kernels. For the ease of presentation, we will only consider the divergence-free kernel.

Consider observables $\boldsymbol{X}_i \in \mathbb{R}^{2\times N}$, where N is the number of spatial locations. Suppose there exists f such that

$$\boldsymbol{X}_{i+1} = f(\boldsymbol{X}_i).$$

Define the Koopman operator $\mathcal{K}$ to be acting on vector-valued function $\varphi : \mathbb{R}^{2\times N} \to \mathbb{R}^{2\times 1}$, i.e.,

$$(\mathcal{K}\varphi)(\boldsymbol{X}) = \varphi \circ f(\boldsymbol{X}).$$

Thus $\mathcal{K}$ is a linear operator. For $n \in \{1, \ldots, N\}$, let $\{\psi_l^{(n)}\}_{l=1}^{\infty}$ be the basis functions in the feature map corresponding to the divergence-free kernel used by the n-th spatial location. Denote

$$\Psi(\boldsymbol{X}) = \begin{bmatrix} \cdots & \psi_l^{(1)}(\boldsymbol{X}^{(:,1)}) & \cdots & \cdots & \cdots & \psi_l^{(N)}(\boldsymbol{X}^{(:,N)}) & \cdots \end{bmatrix}.$$

Then $\Psi(\boldsymbol{X})\Psi(\boldsymbol{Y})^T$ is equal to

$$\sum_{n=1}^{N} k^{(n)}(\boldsymbol{X}^{(:,n)}, \boldsymbol{Y}^{(:,n)}),$$

i.e., the sum of the matrix kernels at all the spatial locations. Each $\{k^{(n)}\}$ is a divergence-free kernel with possibly different kernel scale parameter. Suppose there are $(M+1)$ snapshots. Define

$$\boldsymbol{\Psi}_{\boldsymbol{X}} = \begin{bmatrix} \Psi(\boldsymbol{X}_1) \\ \vdots \\ \Psi(\boldsymbol{X}_M) \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Psi}_{\boldsymbol{Y}} = \begin{bmatrix} \Psi(\boldsymbol{X}_2) \\ \vdots \\ \Psi(\boldsymbol{X}_{M+1}) \end{bmatrix}.$$

We model the evolution by solving

$$\min_{\boldsymbol{K}} \ \|\boldsymbol{\Psi}_{\boldsymbol{X}}\boldsymbol{K} - \boldsymbol{\Psi}_{\boldsymbol{Y}}\|_F^2.$$

In the calculation, we construct $\widehat{\boldsymbol{L}} \in \mathbb{R}^{(2M+2))\times(2M+2))}$, made of $(M+1)\times(M+1)$ blocks of equal size. The (i,j)-th block is equal to

$$\sum_{n=1}^{N} k^{(n)}(\boldsymbol{X}_i^{(:,n)}, \boldsymbol{X}_j^{(:,n)}).$$

Then $\widehat{\boldsymbol{G}} = \widehat{\boldsymbol{L}}^{(1:2M,1:2M)}$, and $\widehat{\boldsymbol{A}} = \widehat{\boldsymbol{L}}^{(3:2M+2,1:2M)}$.

15.1 Comparisons

We compare the modified approach, denoted by **Divergence Free**, with the approach in section 1 with Gaussian kernel, denoted by **Separable Gaussian**. The first dataset is the divergence-free component of the velocity fields from a Navier-Stokes simulation. The data is of size $50 \times 100 \times 2 \times 200$. However, we can

see from the following figure that **Separable Gaussian** seems to provide a richer basis, and perform better than **Divergence Free**. One reason is that **Divergence Free** does not include the cross-terms between spatial locations.

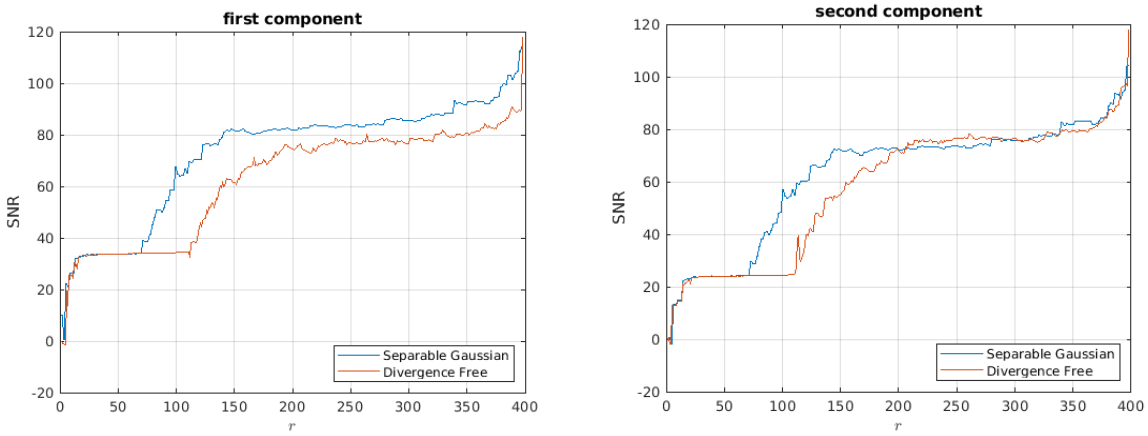

Figure 18: Comparisons on Navier-Stokes dataset.

The second dataset is the divergence-free component from the optical flow of a Cardiac video. The data of is of size $216 \times 256 \times 2 \times 29$. **Separable Gaussian** is better, but none of them is good enough.

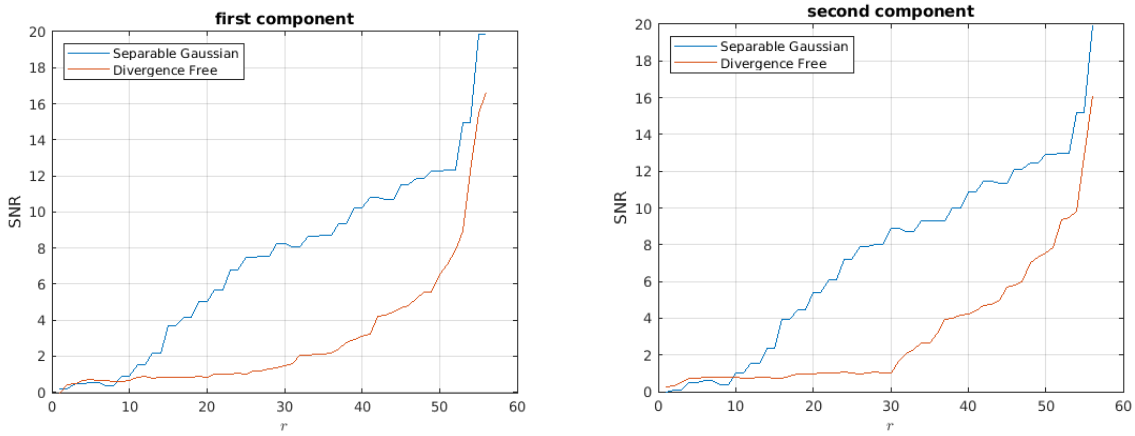

Figure 19: Comparisons on Cardiac dataset.

16 Hierarchical Function Approximation

The idea of hierarchical function approximation may be applied to EDMD. Before doing that, let's consider the following simple example. Randomly generate two groups of 2D points. As shown in the following figure, one group is of radius 1, the other is of radius 2.

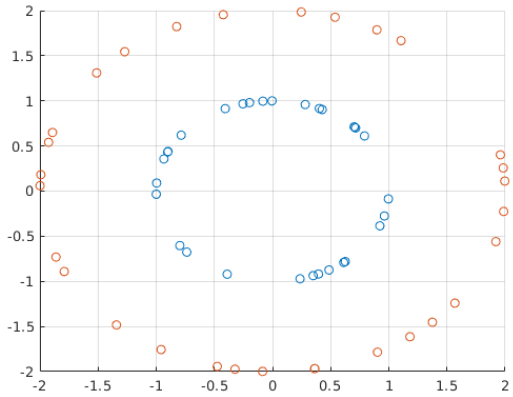

Figure 20: Two groups of points with different radii.

The goal is to find a function that maps points in the first group to 1, and points in the second group to 2. If we use the basis functions

$$1, x_1, x_2, x_1^2, x_1x_2, x_2^2$$

to model, the error is 0, and the function learned is

$$\frac{1}{3}(x_1^2 + x_2^2) + \frac{2}{3}.$$

In a hierarchical manner, if we first use

$$1, x_1, x_2$$

to model, the error is 14.6980, and the function learned is

$$1.4989 - 0.0155x_1 - 0.0615x_2.$$

Fix the coefficients before

$$1, x_1, x_2,$$

we then model the residue using

$$x_1^2, x_1x_2, x_2^2.$$

The error is 10.8708, and the function learned is

$$1.4989 - 0.0155x_1 - 0.0615x_2 + 0.0939x_1^2 + 0.0037x_1x_2 + 0.0798x_2^2.$$

References

[1] Isaac Amidror. Scattered data interpolation methods for electronic imaging systems: a survey. *Journal of electronic imaging*, 11(ARTICLE):157–76, 2002.

[2] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[3] Harsh Bhatia, Valerio Pascucci, and Peer-Timo Bremer. The natural Helmholtz-Hodge decomposition for open-boundary flow analysis. *IEEE transactions on visualization and computer graphics*, 20(11):1566–1578, 2014.

[4] N Benjamin Erichson, Lionel Mathelin, J Nathan Kutz, and Steven L Brunton. Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(4):1867–1891, 2019.

[5] Keisuke Fujii, Yuki Inaba, and Yoshinobu Kawahara. Koopman spectral kernels for comparing complex dynamics: Application to multiagent sport plays. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 127–139. Springer, 2017.

[6] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[7] Milan Korda. Computing controlled invariant sets from data using convex optimization. *arXiv preprint arXiv:1912.03256*, 2019.

[8] Milan Korda and Igor Mezić. Learning Koopman eigenfunctions for prediction and control: the transient case. *arXiv preprint arXiv:1810.08733*, 2018.

[9] Feliks Nüske, Patrick Gelß, Stefan Klus, and Cecilia Clementi. Tensor-based EDMD for the Koopman analysis of high-dimensional systems. *arXiv preprint arXiv:1908.04741*, 2019.

[10] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[11] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[12] Vikas Sindhwani, Stephen Tu, and Mohi Khansari. Learning contracting vector fields for stable imitation learning. *arXiv preprint arXiv:1804.04878*, 2018.

[13] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[14] SVN Vishwanathan, Alexander J Smola, and René Vidal. Binet-cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73(1):95–119, 2007.

[15] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[16] Matthew O Williams, Clarence W Rowley, and Ioannis G Kevrekidis. A kernel-based method for data-driven koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.