Data-driven modeling: Dynamic Mode Decomposition (DMD)

Rafael Harbour

Supervisor: Prof. Naratip Santitissadeekorn

Department of Mathematics
University of Surrey
Guildford GU2 7XH, United Kingdom

Submitted as a report for the Final Year Project for the M.Math. degree at the University of Surrey

May, 2020

Scientific abstract

This report looks into Dynamic Mode Decomposition (DMD) and it's application to time-series data. Chapters 1-4 introduce the methodology of DMD as well as talk about Singular Value Decomposition (SVD), a key component in the DMD algorithm. Chapter 5 applies DMD to a real life setting with influenza data from the United States. In this chapter DMD will be used to reconstruct the data and make predictions of the influenza cases in the following year. DMD will be used on the whole data and on the data separated by the 4 different age groups. In this setting DMD produces an accurate reconstruction of the data although not perfect.

Keywords and AMS Classification Codes: Dynamic Mode Decomposition, SVD, Data Analysis

Acknowledgements

Many thanks to my supervisor Prof. Naratip Santitissadeekorn who has guided me through the entirety of this project. His aid has been extremely helpful, the weekly meetings and his openness to assist with any problem at any time has made this project not only feel more enjoyable but also advance smoothly.

Contents

1 Introduction

Dynamic Mode Decomposition (DMD) is a data based algorithm that requires no knowledge on the complex dynamical systems involved but only requires raw data. DMD works by exploiting low dimensionality in the data and it produces modes that not only can be used to reconstruct past states in the dynamical system but also predict future states. The amazing aspect of DMD is that it does not require any knowledge of the underlying equations governing the dynamical system meaning that it's application is almost limitless as there is usually an abundance of data but it is also very difficult to discover equations that fully explain a complex dynamical system. When you have a large dimensional data matrix $\mathbf{X}$ it can be tricky and time consuming to work with, that is where Singular Value Decomposition (SVD) comes in. SVD creates a low dimensional approximation of $\mathbf{X}$ which still explains the majority of information of the underlying system. The following chapter will explain in more detail the SVD process which will lead into the DMD methodology.

2

Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD)

We want to decompose a $m \times n$ real matrix $\mathbf{A}$ into the sum of rank 1 matrices, this will be important for us later in the DMD methodology. The SVD allows us to decompose $\mathbf{A}$ into r components

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \; r \leqslant n \tag{2.1}$$

where r is the rank of $\mathbf{A}$. The SVD seeks $\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i$ with $\sigma_i > 0$ for all i, and has the requirement that each $\mathbf{u}_i$ is orthogonal to each other in $\mathbb{R}^m$ and each $\mathbf{v}_i$ is also orthogonal to each other in $\mathbb{R}^n$. The vectors u_i and v_i are called left singular vectors and right singular vectors, respectively. The positive real number σ_i is called a singular value of $\mathbf{A}$. A short way to write the above decomposition is

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \tag{2.2}$$

where $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix}$, $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}$ $n \times n$ and $\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$

$\mathbf{\Sigma}$ has dimensions $m \times n$. The matrix $\mathbf{\Sigma}$ only has non-zero entries $\sigma_1 > \sigma_2 > \ldots > \sigma_r$ for $r \leqslant \min(m, n)$. $\mathbf{\Sigma}$ only has entries $\sigma_1 \ldots \sigma_r$ on the main diagonal; all other entries are 0. Since all the columns of $\mathbf{U}$ and $\mathbf{V}$ are orthogonal vectors then $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices such that $\mathbf{U}^{-1} = \mathbf{U}^T$ and $\mathbf{V}^{-1} = \mathbf{V}^T$.

2.0.1. SVD theory

To see where the singular vectors and singular values appear from, we first start by showing that

$$\mathbf{A^T A} = (\mathbf{V\Sigma^T U^T})(\mathbf{U\Sigma V^T}) = \mathbf{V\Sigma^2 V^T} \tag{2.3}$$

$$\mathbf{AA^T} = (\mathbf{U\Sigma V^T})(\mathbf{V\Sigma^T U^T}) = \mathbf{U\Sigma^2 U^T} \tag{2.4}$$

Also

Theorem 2.1. A matrix $\mathbf{X}$ is symmetric if and only if it is orthogonally diagonalizable. $\mathbf{X} = \mathbf{Q\Lambda Q^T}$, $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix.
Hence $\mathbf{A^T A}$ and $\mathbf{AA^T}$ are diagonal matrices. Using the fact that $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices we get

$$\mathbf{A^T AV} = \mathbf{V\Sigma^2} \tag{2.5}$$

$$\mathbf{AA^T U} = \mathbf{U\Sigma^2} \tag{2.6}$$

This is just the eigenvalue/eigenvector equation, with the eigenvectors of $\mathbf{AA}^T$ being the vectors $\mathbf{u}_i$ with corresponding non-zero eigenvalues σ_i^2 and similarly the eigenvectors of $\mathbf{A}^T\mathbf{A}$ being the vectors $\mathbf{v}_i$ with corresponding non-zero eigenvalues σ_i^2. Lastly we want to show that $\mathbf{Av}_i = \sigma_i \mathbf{u}_i$. Since $\mathbf{A}^T\mathbf{Av}_i = \sigma_i^2 \mathbf{v}_i$, this is true if

$$\mathbf{u}_i = \frac{\mathbf{Av}_i}{\sigma_i},$$

and they are mutually orthonormal. It follows that

$$\mathbf{AA}^T \frac{\mathbf{Av}_i}{\sigma_i} = \mathbf{A}\frac{\mathbf{A}^T\mathbf{Av}_i}{\sigma_i} = \mathbf{A}\frac{\sigma_i^2\mathbf{v}_i}{\sigma_i} = \sigma_i^2 \frac{\mathbf{Av}_i}{\sigma_i}.$$

Using the fact that $\mathbf{A^T A}$ and $\mathbf{AA^T}$ are symmetric matrices and the uniqueness eigenvector property of symmetric matrices we get that $\mathbf{u}_i = \frac{\mathbf{Av}_i}{\sigma_i}$. Verifying that $\mathbf{u}_i$ are mutually orthonormal can be seen

$$\mathbf{u}_i^T\mathbf{u}_j = \left(\frac{\mathbf{Av}_i}{\sigma_i}\right)^T \left(\frac{\mathbf{Av}_j}{\sigma_j}\right) = \frac{\sigma_j}{\sigma_i}\mathbf{v}_i^T\mathbf{v}_i = \delta_{ij},$$

$\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Hence, ub_i and vb_i are orthonormal eigenvectors of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{AA}^T$ respectively. Also the singular values are the square root of the non-zero eigenvalues of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{AA}^T$.

2.0.2. Reduced SVD

The above construction of the SVD that only uses non-zero singular values is known as the "reduced" SVD in comparison to the "full" SVD. It is useful to use the reduced SVD when the rank of $\mathbf{A}$ is small and $\mathbf{\Sigma}$ can have many zeros. These zeros will add nothing to the SVD of $\mathbf{A}$. Hence we can create a reduced form of the SVD. Figure 2.0.2. shows the full SVD whilst 2.0.2. demonstrates the reduced SVD.

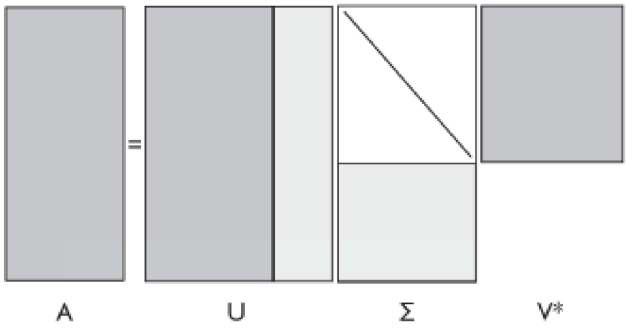

Figure 2.1: Full SVD [1]

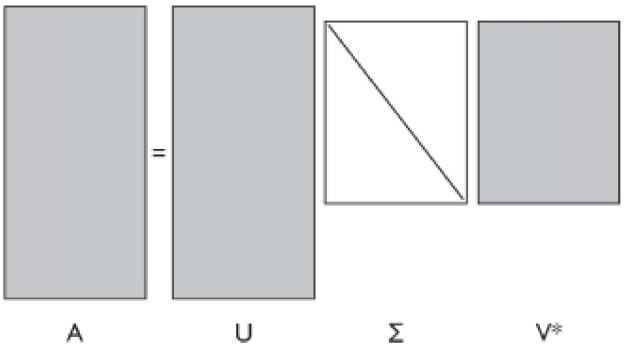

Figure 2.2: Reduced SVD [1]

2.0.3. Eckart Young Theorem

One of the key aspects in why the SVD is such a powerful tool is the Eckart Young Theorem which states

Theorem 2.2. If $\mathbf{B}$ has rank k then $\|\mathbf{A} - \mathbf{A_k}\| \leqslant \|\mathbf{A} - \mathbf{B}\|$ where $\mathbf{A_k} = \sigma_1 \mathbf{u_1} \mathbf{v_1^T} + ... + \sigma_k \mathbf{u_k} \mathbf{v_k^T}$ with $k < r$

The above statement is true for all matrix norms. This is important as it can give us a matrix $\mathbf{A_k}$ where $\mathbf{A_k}$ is the best rank k matrix that approximates $\mathbf{A}$.

2.0.4. Geometric view of SVD

Singular values and vectors can explain to us how vectors would be transformed under a linear transformation of $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$. When $\mathbf{A}$ is applied on a vector $\mathbf{x}$ so $\mathbf{x} \rightarrow \mathbf{A}\mathbf{x}$, $\mathbf{U}$ and $\mathbf{V}$ rotate or reflect the vector $\mathbf{x}$ whilst $\mathbf{\Sigma}$ stretches the vector. This is demonstrated in 2.0.4.

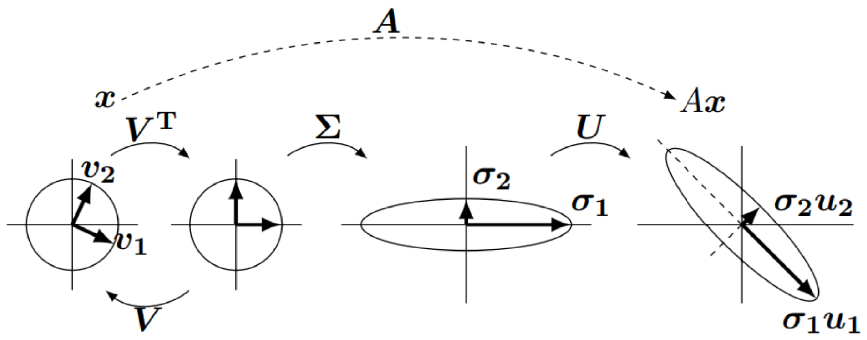

Figure 2.3: Geometric representation of the SVD. Here you can see how $\mathbf{V}$ and $\mathbf{U}$ turn the circle round and how $\mathbf{\Sigma}$ stretches/shrinks the circle. [3]

3

Singular Value Decomposition (SVD) vs Principal Component Analysis (PCA)

Principal Component Analysis is a method for reducing a complex data set to lower dimensions. To understand PCA more take for example an experiment where you want to study the motion of a ball of mass m attached to a massless frictionless spring. The ball is released a small distance away from equilibrium and then oscillates along the x-axis (one dimension). So the dynamics can be expressed as a single variable x. But lets assume we did not know the underlying dynamics of the experiments, hence we decide to measure position of the ball with 3 cameras in arbitrary positions 3.

With this data from the cameras we are not sure if the dynamics of the movement of the ball are 3D, 2D or 1D. But PCA would be able to reduce this data to a 1D system (x-axis) and that is why it is important. PCA will give us the best basis to re-express a data set, in the case of our experiment the dynamics are across the x-axis so we want to find the unit vector along the x-axis $\hat{\mathbf{x}}$.

Lets say we had a data matrix $\mathbf{X}$ $m \times n$, m is the number of measurement types and n is the number of samples, so $\mathbf{X}$ has n data sets of size m. Let $\mathbf{Y}$ be another matrix $m \times n$ related by a linear transformation $\mathbf{P}$, such that

$$\mathbf{PX} = \mathbf{Y} \tag{3.1}$$

$\mathbf{P}$ is a matrix that transforms $\mathbf{X}$ into $\mathbf{Y}$, the rows of $\mathbf{P}$ are a set of new basis vectors for representing the columns of $\mathbf{X}$ this can be seen as follows

$$\mathbf{PX} = \begin{bmatrix} \mathbf{p_1} \\ \vdots \\ \mathbf{p_m} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} & \cdots & \mathbf{x_n} \end{bmatrix} \tag{3.2}$$

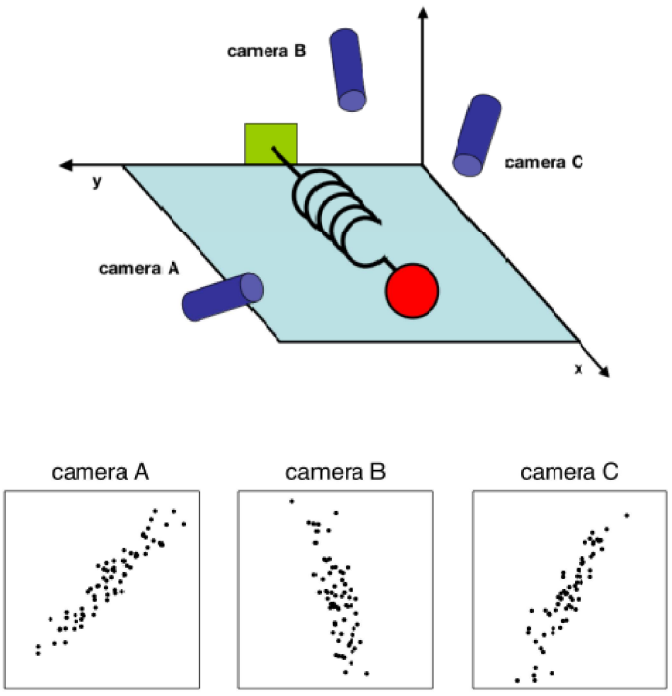

Figure 3.1: Ball on spring, position of the ball recorded by 3 different cameras in 3 different locations [2]

$$\mathbf{Y} = \begin{bmatrix} \mathbf{p_1 x_1} & \cdots & \mathbf{p_1 x_n} \\ \vdots & \ddots & \vdots \\ \mathbf{p_m x_1} & \cdots & \mathbf{p_m x_n} \end{bmatrix} \tag{3.3}$$

Geometrically $\mathbf{P}$ is a rotation and stretch of matrix $\mathbf{X}$ to $\mathbf{Y}$. The row vectors $\mathbf{p_1} \cdots \mathbf{p_m}$ will become the principal components of $\mathbf{X}$.

3.0.1. Noise in data

We want noise in our data to be small. All noise can be quantified relative to the signal strength. A common measure is the signal-to-noise ratio (SNR)

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} \tag{3.4}$$

A high SNR ($>> 1$) indicates high precision in the data whilst a low SNR implies noisy data. Note that the largest direction of variance lies on the best fit line 3.0.1.. We know that in our

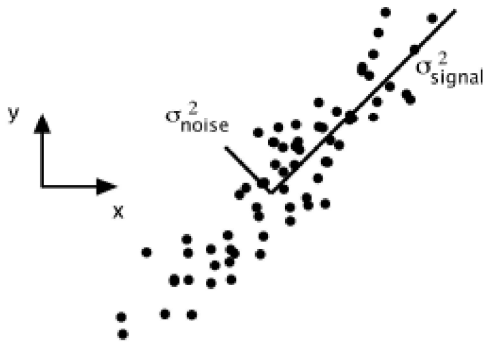

Figure 3.2: Simulated data of (x,y) for our spring/ball experiment for camera A, signal and noise variances σ^2_{signal} σ^2_{noise} are represented by lines on the data [2].

experiment the spring moves in a straight line, any spread deviating from the straight line is noise. Thus from this we can assume that the dynamics of interest lie on the direction of largest variance and highest SNR.

Covariance matrix

Take the covariance matrix of $\mathbf{X}$ to be

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}\mathbf{X}^\mathbf{T} \tag{3.5}$$

then $\mathbf{C_X}$ is a square symmetric $m \times m$ matrix. The diagonal terms of $\mathbf{C_X}$ are the variances of each measurement type (rows of $\mathbf{X}$) and the off-diagonal terms are the covariance between the measurement types. Large diagonal terms in $\mathbf{C_X}$ correspond to interesting nature whilst in the off-diagonal terms large magnitudes imply high redundancy.

In the case of the covariance matrix of $\mathbf{Y}$, $\mathbf{C_Y}$, we would like to minimize redundancy and also maximise the signal (measured by the variance). So we want to diagonalise $\mathbf{Y}$ so the off-diagonal values are zero and the variances in the diagonal entries are ordered from large to small. This is important as it allows us to see how "principal" each principal component $\mathbf{p_i}$ is, higher variance

associated with $\mathbf{p_i}$ the more that principal component represents the data.

$$\begin{aligned}\mathbf{C_Y} &= \frac{1}{n}\mathbf{Y}\mathbf{Y}^\mathbf{T} \\ &= \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^\mathbf{T} \\ &= \frac{1}{n}\mathbf{PX}\mathbf{X}^\mathbf{T}\mathbf{P}^\mathbf{T} \\ &= \mathbf{P}(\frac{1}{n}\mathbf{X}\mathbf{X}^\mathbf{T})\mathbf{P}^\mathbf{T} \\ \mathbf{C_Y} &= \mathbf{P}\mathbf{C_X}\mathbf{P}^\mathbf{T}\end{aligned} \tag{3.6}$$

Since $\mathbf{C_X}$ is symmetric it can be orthogonally diagonalisable (using theorem 2.1). Let $\mathbf{C_X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathbf{T}$ and take $\mathbf{Q}^\mathbf{T} \equiv \mathbf{P}$ hence we get

$$\begin{aligned}\mathbf{C_Y} &= \mathbf{P}\mathbf{C_X}\mathbf{P}^\mathbf{T} \\ &= \mathbf{P}(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\mathbf{T})\mathbf{P}^\mathbf{T} \\ &= \mathbf{P}\mathbf{P}^\mathbf{T}\mathbf{\Lambda}\mathbf{P}\mathbf{P}^\mathbf{T} \\ &= \mathbf{\Lambda}\end{aligned} \tag{3.7}$$

So the choice of $\mathbf{P}^\mathbf{T}$ such that the columns of $\mathbf{P}^\mathbf{T}$ are eigenvectors of $\mathbf{C_X}$ diagonalises the matrix $\mathbf{C_Y}$. The eigenvectors of $\mathbf{C_X}$ are the principal components of $\mathbf{X}$.

3.0.2. How does PCA relate to SVD

Take a new $n \times m$ matrix $\mathbf{B}$ to be

$$\mathbf{B} = \frac{\mathbf{1}}{\sqrt{\mathbf{n}}}\mathbf{X}^\mathbf{T} \tag{3.8}$$

Then

$$\begin{aligned}\mathbf{B}^\mathbf{T}\mathbf{B} &= \frac{\mathbf{1}}{\mathbf{n}}\mathbf{X}\mathbf{X}^\mathbf{T} \\ &= \mathbf{C_X}\end{aligned} \tag{3.9}$$

We know that the principal components of $\mathbf{X}$ are the eigenvectors of $\mathbf{C_X}$. Now if we calculate the SVD of $\mathbf{B}$ we find that $\mathbf{V}$ contains the eigenvectors of $\mathbf{B}^\mathbf{T}\mathbf{B}$. Therefore the columns of $\mathbf{V}$ are the principal components of $\mathbf{X}$, we can use SVD to find the principal components.

4 Dynamic Mode Decomposition (DMD)

Proper Orthogonal Decomposition (POD)

The idea behind POD is to represent a given function $f(\mathbf{x}, \mathbf{t})$ as the sum of ordered orthonormal functions which are called the Proper Orthogonal Modes for the function $f(\mathbf{x}, \mathbf{t})$.

$f(\mathbf{x}, \mathbf{t}) \approx \sum_{j=1}^{N} \mathbf{a}_j(\mathbf{t})\boldsymbol{\phi}_j(\mathbf{x})$, here $\mathbf{x}$ represents the space variable and $\mathbf{t}$ represents the time variable.

$\boldsymbol{\phi}_j(\mathbf{x})$ represent the POD modes and $\mathbf{a}_j(\mathbf{t})$ are the respective weights of those modes.

Dynamic Mode Decomposition (DMD)

DMD is useful for when there is no underlying equations regarding the system but instead there is a large sample of data over a short time span. This sample of data is used to predict the underlying system. The aim is to take advantage of low dimensionality in the data.

With DMD we consider data collected from a dynamical system

$$\frac{d\mathbf{x}}{d\mathbf{t}} = f(\mathbf{x}, \mathbf{t}) \qquad \mathbf{x}(\mathbf{t}) \in \mathbb{R}^N \tag{4.1}$$

$\mathbf{x}(t)$ is a vector containing the data of the dynamical system at time t. Assume that the data is collected at evenly spaced time intervals.

$$t_{i+1} = t_i + \Delta t \qquad \Delta t > 0 \qquad i = 1, 2, ... \tag{4.2}$$

Thus

$$\mathbf{x}(i\Delta t) = \mathbf{x}_i \tag{4.3}$$

Lets say we had collected sample data of a dynamical system over a period of time. This data could be organised into a $N \times M$ matrix $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & ... & \mathbf{x}_M \end{bmatrix} \qquad \mathbf{x}_i \in \mathbb{R}^N \text{ for all } i = \{1, 2, ..., M\} \tag{4.4}$$

The subscripts of $\mathbf{x}_i$ indicate the time t_i at which the snapshots of data where taken. Hence $\mathbf{X}$ has M different data samples at M different times each containing N pieces of data. DMD deals with large amount of data in a short time interval hence $M << N$.
We assume a linear dynamic for $\mathbf{x}_i$

$$\mathbf{x}_{i+1} \approx \mathbf{A}\mathbf{x}_i \tag{4.5}$$

Hence $\mathbf{A}$ maps the data from time $\mathbf{t}_i$ to $\mathbf{t}_{i+1}$. The mapping of $\mathbf{A}$ is linear even though the process that generated $\mathbf{x}_i$ may be nonlinear. We want $\mathbf{A}$ such that $\|\mathbf{x}_{i+1} - \mathbf{A}\mathbf{x}_i\|_2$ is minimized for all i To find $\mathbf{A}$ we start by constructing

$$\mathbf{X}_1^{M-1} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & ... & \mathbf{x}_{M-1} \end{bmatrix} \tag{4.6}$$

and

$$\mathbf{X}_2^{M} = \begin{bmatrix} \mathbf{x}_2 & \mathbf{x}_3 & ... & \mathbf{x}_{M} \end{bmatrix} \tag{4.7}$$

Let $\mathbf{A} \approx \mathbf{X}_2^M \mathbf{X}_1^{M-1^*}$, where $\mathbf{X}_1^{M-1^*}$ is the moore-penrose inverse of $\mathbf{X}_1^{M-1}$. This is because if we find $\mathbf{A}$ we can use $\mathbf{A}$ to give us insight into the underlying system of the data and allow us to predict future states of the system outside our sample data. Firstly we wish to find the eigenvalues and eigenvectors of $\mathbf{A}$. As these values will explain the dynamics of $\mathbf{A}$. We use reduced SVD to decompose $\mathbf{X}_1^{M-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$ $\mathbf{U}$ is $N \times r$ matrix, $\mathbf{A}$ is $N \times N$ matrix and $\mathbf{V}^T$ is $r \times (M-1)$ matrix. Hence

$$\mathbf{A} = \mathbf{X}_2^M \mathbf{X}_1^{M-1^*} = \mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T \tag{4.8}$$

Now we want to use POD, by projecting $\mathbf{A}$ onto a basis spanned by reduced matrix $\mathbf{U}$ we get a reduced matrix of $\mathbf{A}$.

$$\bar{\mathbf{A}} = \mathbf{U}^T\mathbf{A}\mathbf{U} \tag{4.9}$$

Here $\bar{\mathbf{A}}$ is the reduced matrix of $\mathbf{A}$ and is an $r \times r$ matrix, $\mathbf{U}^T$ has dimensions $r \times N$, $\mathbf{A}$ has dimensions $N \times N$ and $\mathbf{U}$ has dimensions $N \times r$.

$$\begin{aligned} \bar{\mathbf{A}} &= \mathbf{U}^T\mathbf{A}\mathbf{U} \\ &= \mathbf{U}^T\mathbf{X}_2^M\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{U} \\ &= \mathbf{U}^T\mathbf{X}_2^M\mathbf{V}\mathbf{\Sigma}^{-1} \end{aligned} \tag{4.10}$$

So the reduced model is $\bar{\mathbf{x}}_{i+1} = \bar{\mathbf{A}}\bar{\mathbf{x}}_i$, we can reconstruct the high dimensional state $\mathbf{x}_i = \mathbf{U}\bar{\mathbf{x}}_i$. Now we try to find the eigenvalues and eigenvectors of $\bar{\mathbf{A}}$ and use them to reconstruct the eigenvalues and eigenvectors of $\mathbf{A}$. Want to solve

$$\bar{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda} \tag{4.11}$$

where $\mathbf{W}$ and $\mathbf{\Lambda}$ are $r \times r$ matrices and each column of $\mathbf{W}$ is an eigenvector whilst $\mathbf{\Lambda}$ has the eigenvalues on the diagonal. $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & ... & \mathbf{w}_r \end{bmatrix}$ and $\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_r \end{bmatrix}$ To reconstruct the eigenvectors of $\mathbf{A}$ first take $\mathbf{\Phi}_1 = \mathbf{U}\mathbf{W}$ then

$$\begin{aligned} \bar{\mathbf{A}}\mathbf{W} &= \mathbf{W}\mathbf{\Lambda} \\ \mathbf{U^T A U W} &= \mathbf{W}\mathbf{\Lambda} \\ \mathbf{A U W} &= \mathbf{U W}\mathbf{\Lambda} \end{aligned} \tag{4.12}$$

So $\mathbf{\Phi}_1$ is a matrix whose columns are eigenvectors of $\mathbf{A}$, these eigenvectors are known as the projected DMD modes. Now take $\mathbf{\Phi}_2 = \mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$ then

$$\begin{aligned} \mathbf{A}\mathbf{\Phi}_2 &= \mathbf{A}\mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W} \\ &= \mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W} \text{ using } \bar{\mathbf{A}} = \mathbf{U}^T\mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1} \\ &= \mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\bar{\mathbf{A}}\mathbf{W} \\ &= \mathbf{X}_2^M \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}\mathbf{\Lambda} \end{aligned} \tag{4.13}$$

$\mathbf{A}\mathbf{\Phi}_2 = \mathbf{\Phi}_2\mathbf{\Lambda}$,therefore $\mathbf{\Phi}_2$ is also a matrix whose columns are eigenvectors of $\mathbf{A}$, these eigenvectors are known as the exact DMD modes.

Approximate values of $\mathbf{x}_i$ at all future times

With the low rank approximations of the eigenvalues and eigenvectors we can now construct the future state of the system for all time. Let

$$\omega_i = \frac{\ln(\lambda_i)}{\Delta t} \tag{4.14}$$

then the solution for all future times is

$$\mathbf{x(t)} \approx \sum_{i=1}^{r} \boldsymbol{\phi}_i \exp(\boldsymbol{\omega}_i \mathbf{t})\mathbf{b}_i = \mathbf{\Phi}\exp(\mathbf{\Omega} \boldsymbol{t})\mathbf{b} \tag{4.15}$$

where $\mathbf{b}_i$ is the initial amplitude of each mode, $\mathbf{\Phi}$ is the matrix whose columns are the DMD eigenvectors $\boldsymbol{\phi}_i$, $\mathbf{\Omega} = \text{diag}(\boldsymbol{\omega})$ is a diagonal matrix whose elements are the eigenvalues ω_i. $\mathbf{b}$ is a vector of coefficients b_i. This equation is for continuous time. For discrete time we have the equation

$$\mathbf{x}_i = \sum_{j=1}^{r} \boldsymbol{\phi}_j \boldsymbol{\lambda}_j^i b_j \tag{4.16}$$

Here $\boldsymbol{\phi}_j$ are the eigenvectors and $\boldsymbol{\lambda}_j$ are the eigenvalues associated with the eigenvectors at time t_j. To calculate the coefficients b_j take $t_1 = 0$ then $\mathbf{x}_1 = \sum_{j=1}^{r} \boldsymbol{\phi}_j \mathbf{b}_j = \mathbf{\Phi}\mathbf{b}$ therefore

$$\mathbf{b} = \mathbf{\Phi}^* \mathbf{x}_1 \tag{4.17}$$

where $\mathbf{\Phi}^*$ is the pseudo inverse of $\mathbf{\Phi}$

4.0.1. Demonstration of DMD

To demonstrate DMD we will look at two mixed spatio-temporal signals and see the efficiency of DMD in decomposing the signal into its temporal and spacial parts. We will also compare the results obtained from the DMD with the results obtained from PCA. Take the two signals to be

$$\begin{aligned} f(\mathbf{x},\mathbf{t}) &= f_1(\mathbf{x},\mathbf{t}) + f_2(\mathbf{x},\mathbf{t}) \\ &= 3\text{sech}(\mathbf{x}+4.3)\exp(i2.8\mathbf{t}) + 4\text{sech}(\mathbf{x})\tanh(\mathbf{x})\exp(i3.6\mathbf{t}) \end{aligned} \tag{4.18}$$

The two frequencies are $\omega_1 = 2.8$ and $\omega_2 = 3.6$ which have distinct spatial structures. We construct two vectors $\mathbf{x}$ size 1×400, that goes from -10 to 10 in 400 evenly spaced intervals, and $\mathbf{t}$ 1×200, that goes from 0 to 4π in 200 evenly spaced intervals. With these vectors we can get a data matrix $\mathbf{X}$ size 400×200, which are values of $f(\mathbf{x},\mathbf{t})$, then we can use DMD on the data matrix $\mathbf{X}$. We can see the individual spatial-temporal signals in figure 4.1 The ω_i

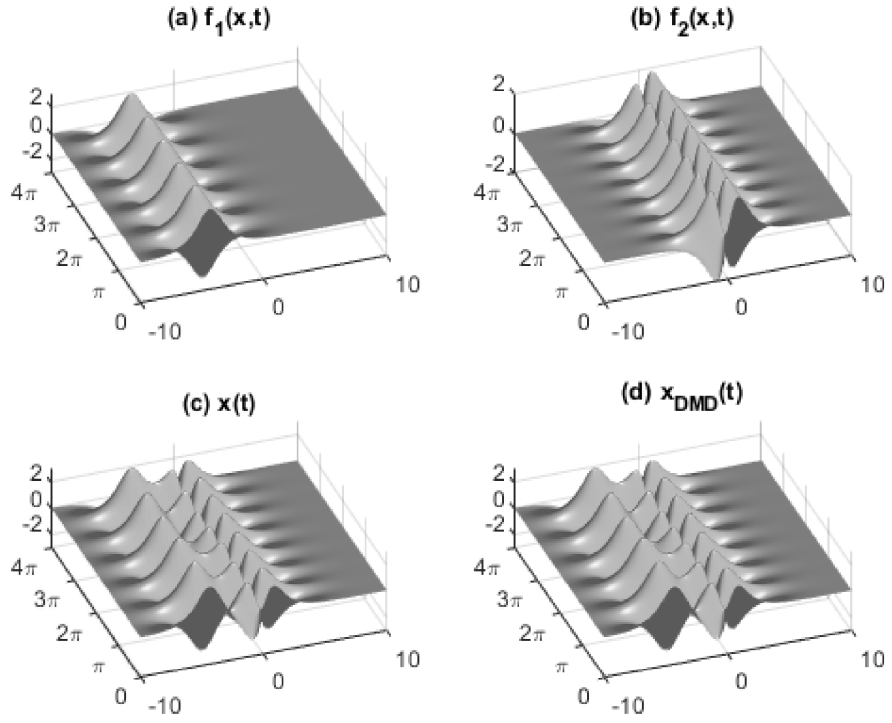

Figure 4.1: An example of the dynamics of the two spatio-temporal signals. (a) $f_1(\mathbf{x},\mathbf{t})$, (b) $f_2(\mathbf{x},\mathbf{t})$, (c) $f(\mathbf{x},\mathbf{t}) = f_1(\mathbf{x},\mathbf{t}) + f_2(\mathbf{x},\mathbf{t})$ and (d) $\mathbf{x}_{DMD}$ reconstruction. We can see that DMD does an excellent job at matching the underlying dynamics of $f(\mathbf{x},\mathbf{t})$. (Matlab code 7.1.)

values obtained by the DMD process match exactly with the underlying frequencies of $f_i(\mathbf{x},\mathbf{t})$.

Now we can compute the PCA modes and compare them with the DMD modes to see how each method approximates the underlying system 4.2. Clearly DMD outperforms PCA in estimating the underlying dynamics of the system.

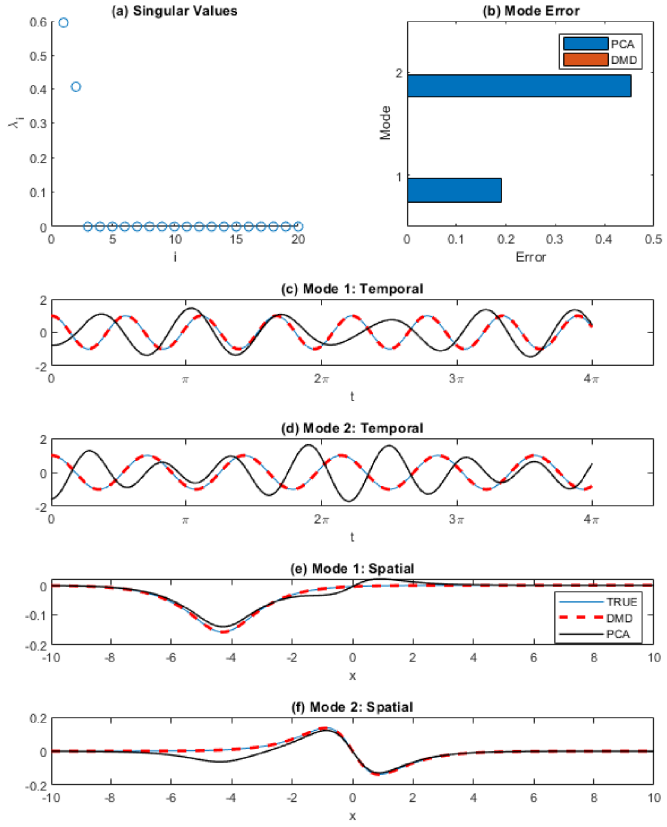

Figure 4.2: Comparison between DMD and PCA. (a) Shows the singular values obtained from the SVD, this shows that the system is 2 dimensional. (b) Demonstrates the error in the spatial modes in comparison with the underlying system, the DMD modes have basically zero error much less than the spatial modes. (c)-(d) Shows the temporal modes, DMD follows the true modes closely while PCA is slightly off. (e) and (f) Compare the spatial modes again the DMD modes follow the true modes almost perfectly whilst the PCA modes trail off slightly. (Matlab code 7.1.1.)

4.0.2. Issues with the DMD method

DMD and PCA rely on the underlying SVD. A weakness to SVD approaches is that they have trouble with invariances in the data. For example DMD and PCA have trouble with translation and/ or rotational invariances. In this case the DMD reconstruction may need more modes than

the rank of the underlying system to achieve a good approximation of the underlying system.

Centering data to improve DMD

DMD assumes a linear system to the dynamical system such that $\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i$ but what if the data could be more closely fit to an affine model. If the mean of $\mathbf{X}$ is not zero then the DMD model could be improved with an extra affine term. DMD with centering can always produce the correct dynamics of a system but DMD without centering can sometimes give out an inaccurate model for the underlying system.

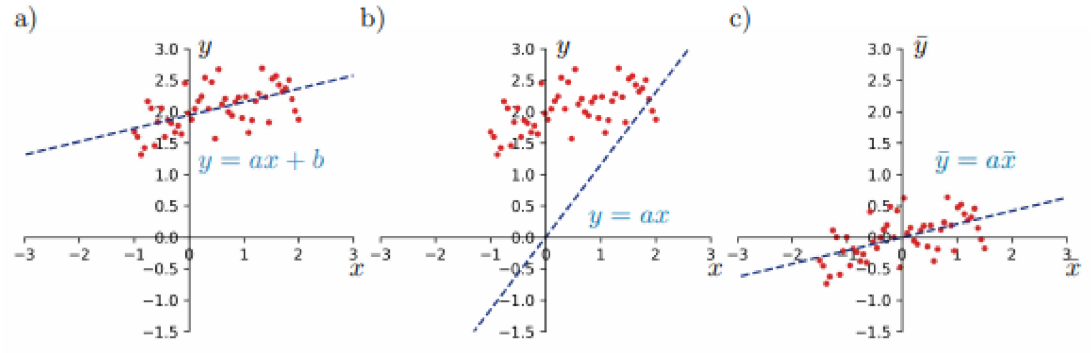

Figure 4.3: The data is generated by an affine model with noise. a) Shows the data being fit with an affine model $y = ax + b$. b) Data is fit with linear model $y = ax$ which leads to a poor fit. c) Centered data fit to linear model leads to a good fit [5].

5 DMD Application on Flu Data

We will be applying DMD to US flu data reported by public health laboratories [4]. The data collects positive (weekly) influenza tests. The data is separated by age groups (0-4 years, 5-24 years, 25-64 years and 65+ years) and 5 different virus types. The data ranges from 1997 to current year and the records start at week 40 of the year (1st week of October).

Organising the data

To run DMD we need to rearrange the data into a matrix $\mathbf{X}$. Most years collected flu data for 52 weeks but some gathered data from a different amount of weeks, these years had to be omitted(years 1997-1998, 2003-2004, 2008-2009, 2009-2010 and 2014-2015 were omitted). Firstly we will sort the data by the total amount of influenza cases by week and we standardize the data by dividing the total number of weekly influenza cases by the yearly influenza cases, so the total yearly influenza cases is 1 for every year. We standardize the data since older years had a lot less flu cases, this may be due to lower population or even that less people where diagnosed/recorded with flu. Throughout this report we will only be using standardized data. With this data we create a matrix $\mathbf{X}$ 52×17 where the rows are the weeks and the columns are the years (organised in ascending order $X(1,1)$ indicates week 1 year 1998-1999). The graph below 5.1 shows that the positive flu cases really start to increase after the holidays (around January) and that the peak amount of people being diagnosed with influenza is around February.

5.0.1. Running DMD on the standardized data

When we initially computed DMD on the data we set the initial amplitude to each mode $\mathbf{b} = \mathbf{\Phi}^{*}\mathbf{x_1}$. This did not give us a very good reconstruction of the true data 5.2. As you can see a large amount of the DMD reconstruction was negative and jittery, this is not good as you cannot have negative amounts of people diagnosed with influenza and also was not a very good fit to the true data, although some years did have a good fit to the data, such as 1998-1999, 2000-2001

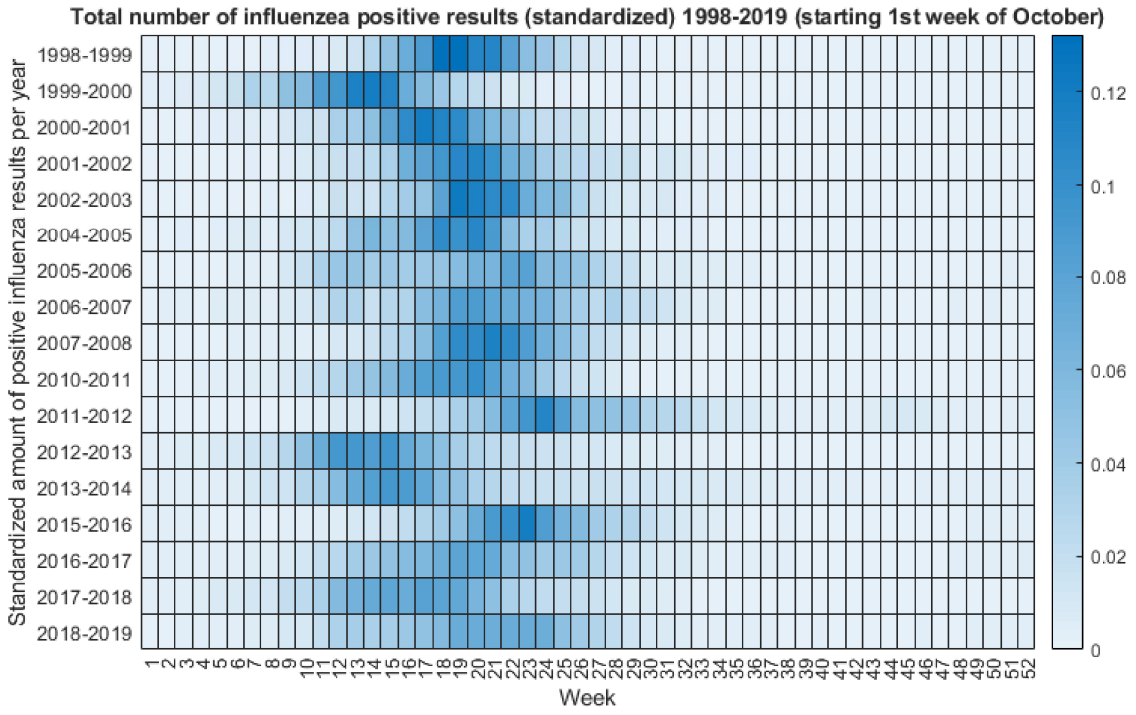

Figure 5.1: Heatmap of the flu data. The darker the blue represents the larger amounts of positive influenza results (standardized). For example in 2011-2012 the peak amount of positive influenza results was around early April 2012 whilst in 2011-2012 the peak was around early January.

and 2015-2016, it was overall a fairly poor fit. To get a better reconstruction we calculated $\mathbf{b}$ to depend not just on the first year $\mathbf{x_1}$ but on all 17 years 5.3. This was done by letting $\dot{\mathbf{b}} = \dot{\mathbf{\Phi}}^* \dot{\mathbf{X}}$ where $\dot{\mathbf{\Phi}} = \begin{bmatrix} \mathbf{\Phi} \\ \mathbf{\Phi\Lambda} \\ \mathbf{\Phi\Lambda^2} \\ \vdots \\ \mathbf{\Phi\Lambda^{16}} \end{bmatrix}$ with dimensions 884×16 and $\dot{\mathbf{X}} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \vdots \\ \boldsymbol{x_{17}} \end{bmatrix}$ with dimensions 1×884. Although there is now no negative values when using full 16 modes, the reconstruction is still not perfect as some years have some sort of lag between the true data and the reconstructions. For example 1998-1999 the DMD reconstruction peaks noticeably earlier than the true data and in the 2011-2012 year the difference is even more evident. There is another alternative way to also get a positive reconstruction, this involves changing all 0 values in X into a very small positive number, then taking the log of every entry in X executing the DMD algorithm on $log(X)$ and then taking the exponential of the values of the DMD reconstruction.

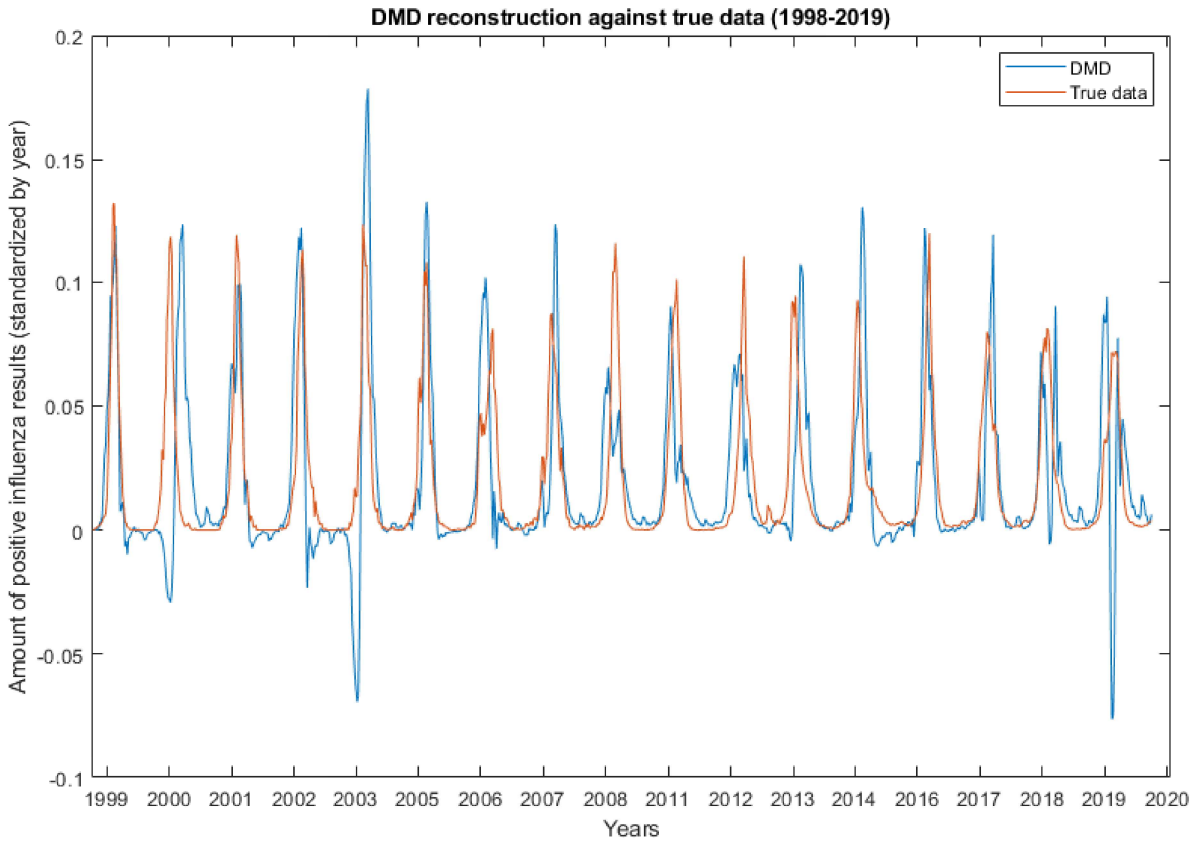

Figure 5.2: True data against DMD reconstruction with maximum (16) DMD modes. As you can see the DMD reconstruction has a large amount of negative values, this is not good as all the true data is positive.

Using DMD for prediction

DMD can be used to make predictions using 4.16. It is also possible to create some amount of error in the DMD predictions/reconstructions by changing the amplitude of each mode $\bar{\mathbf{b}}$ to have a normal distribution with mean $\mathbf{b}$ and variance σ, $\bar{\mathbf{b}} \sim N(\mathbf{b}, \sigma)$. For example 5.4 shows the true data with 5 predictions of the 2018-2019 year, here $\dot{\sigma} = 0.0001$. As seen in the figure the true data is surrounded by the DMD predictions.

Also in 5.5 you can see 5 predictions for the flu data in the upcoming 2019-2020 year. This data is not available yet as the data ranges from 52 weeks starting October every year. But it will be interesting to see if the true data matches the predictions.

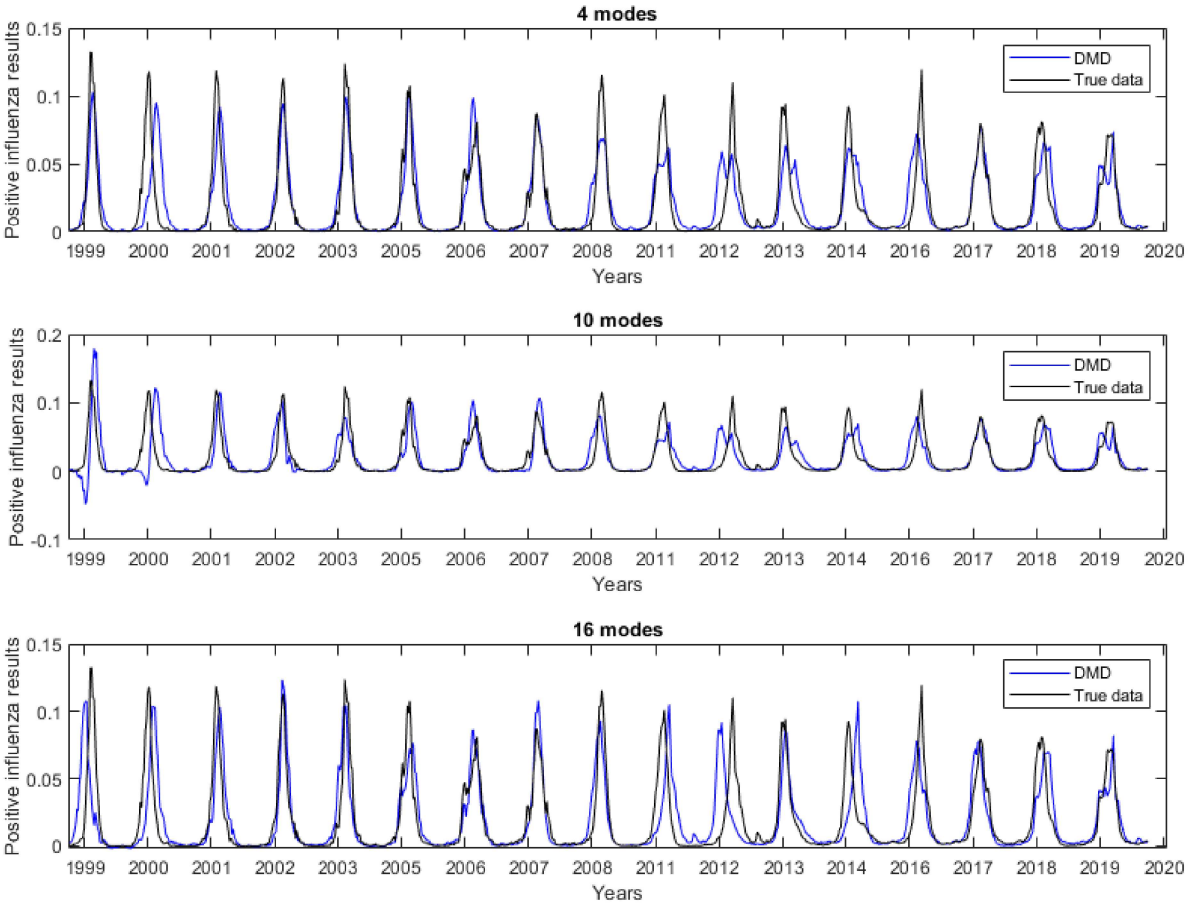

Figure 5.3: True data against DMD reconstruction with maximum 16 DMD modes. Here we have used 4,10 and 16 modes to reconstruct the data. As you can see this DMD reconstruction with the new $\dot{\mathbf{b}}$ is much closer to the true data with significantly lower negative values.

5.0.2. Flu data separated by age groups

It can also be interesting to look at the data split into the four different age groups and run DMD on this new data. The new data matrix $\mathbf{X} = \begin{bmatrix} \mathbf{X11} \\ \mathbf{X22} \\ \mathbf{X33} \\ \mathbf{X44} \end{bmatrix}$ such that $\mathbf{X11}, \mathbf{X22}, \mathbf{X33}$ and $\mathbf{X44}$ have dimension 52×17 and they are organised in ascending age group order such that $\mathbf{X11}$ represents the $0-4$ year age group and $\mathbf{X44}$ represents the 65+ age group, the rows are the weeks and the columns are the years organised in ascending order. From this new $\mathbf{X}$ we subtract the mean from each entry in $\mathbf{X}$ then we run DMD with rank $r = 12$ to get 12 DMD modes. This

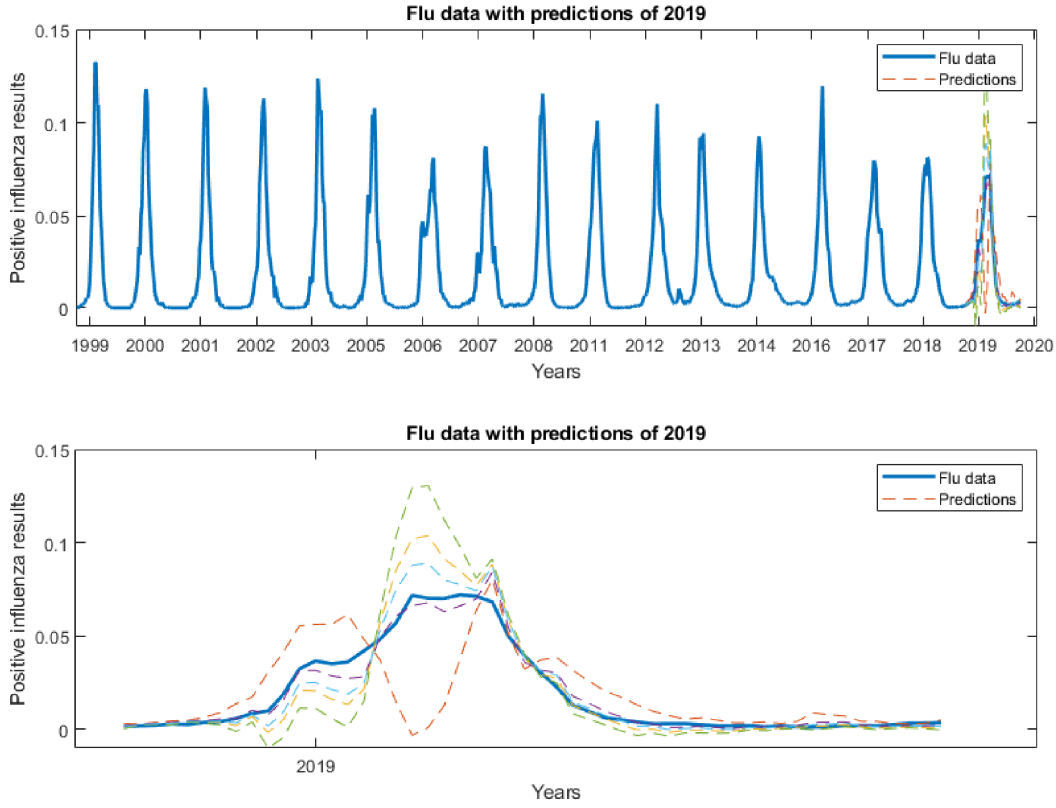

Figure 5.4: Flu data with 5 DMD predictions for the 2018-2019 year

is done to center the data to help improve the DMD as seen in the previous chapter. Figure 5.6 shows the data average against the first DMD mode. Apart from the first DMD mode having some negative values we can see that it is very close to the data average. In figure 5.7 we can see all the years with the age groups and the DMD 12 mode reconstructions.

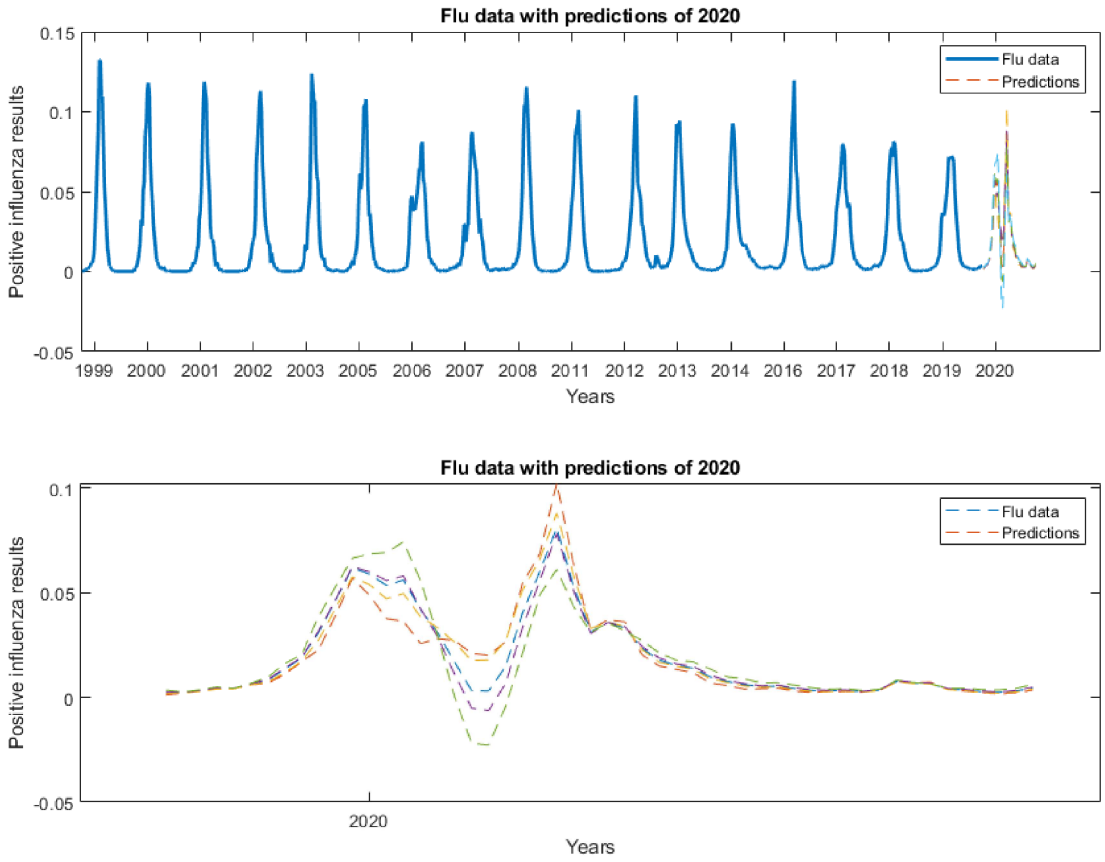

Figure 5.5: Flu data with 5 DMD predictions for the 2019-2020 year, when the data is released then it can be plotted with the predictions to see how close the DMD predictions are to the true data.

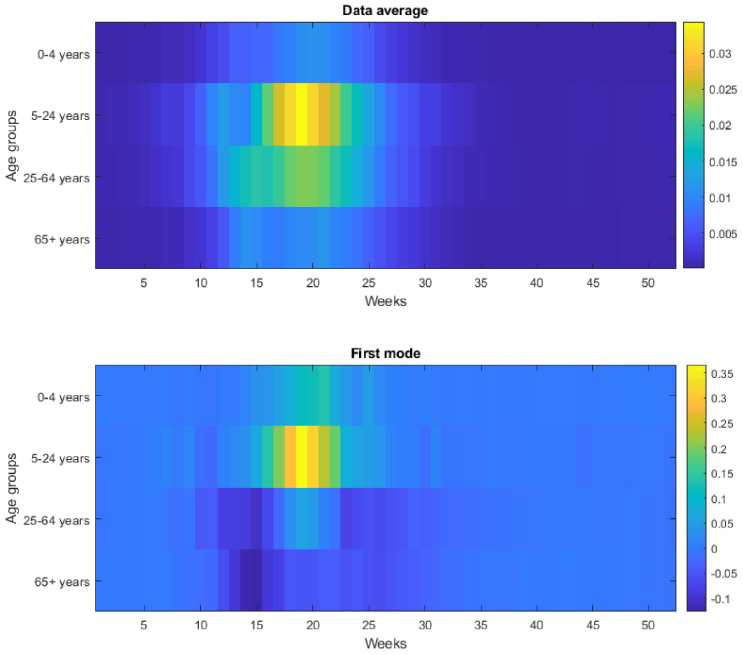

Figure 5.6: Data average and first DMD mode

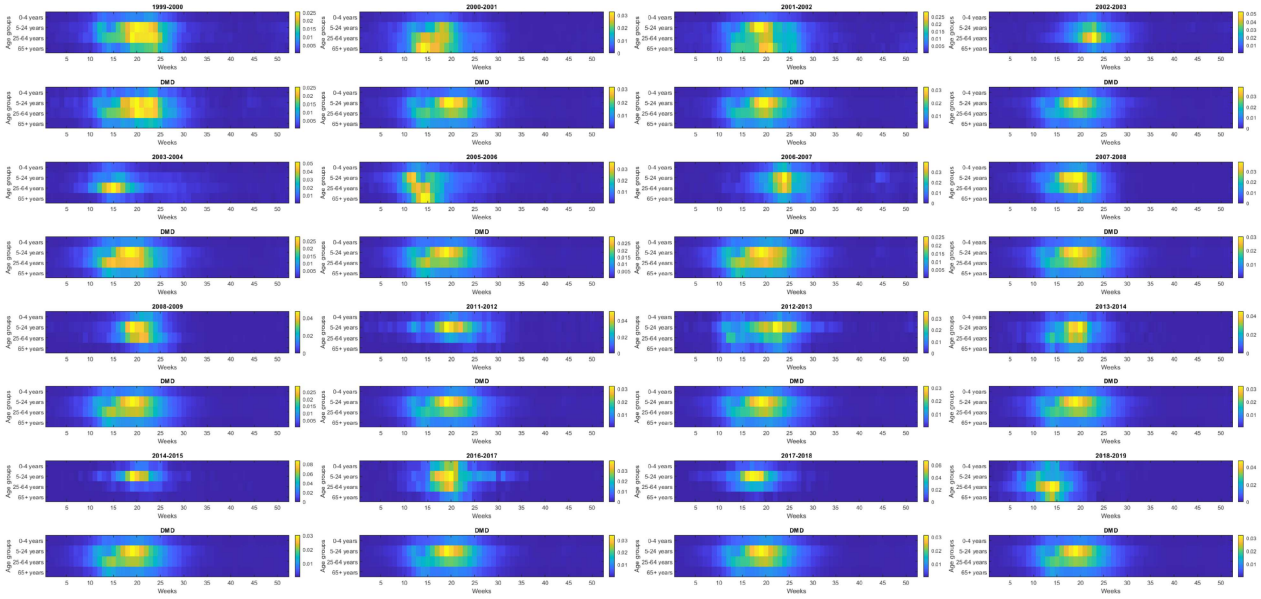

Figure 5.7: Every year of flu data with their corresponding DMD 12 mode reconstruction below the year. The DMD reconstructions seem to stay close to the first mode/ sample data average with slight variations. This means that when there is a year such as 2000-2001 which has a lot of cases for people in the age range 65+ then DMD struggles to reconstruct the data accurately.

6 Conclusions and Outlook

6.0.1. Conclusion

DMD has proven to be a very strong data-based algorithm due to its accessibility,variability and accuracy. In chapter 4 we constructed a function composed of two signals, then we recorded the data. When DMD decomposed the signal into the temporal and spatial parts it did so with nearly zero error in this regard vastly outperforming PCA. With the flu data DMD struggled more possibly due to the randomness involved in the amount of people being positively tested for influenza weekly. Although the DMD reconstruction was not perfect it was still surprisingly accurate even when using a small number of DMD modes as seen in figure 5.3.

6.0.2. Outlook

There are many more things that could have been done, for instance looking into how different values of amplitudes of the DMD modes **b** and how they would compare against each other in creating a DMD reconstruction that was closer/further away from the true data. Also we only observed DMD predictions for the upcoming year and this previous year but it would have been interesting to look into distant DMD predictions for instance in the year 2050-2051 and see how that prediction would differ if at all. Also I did not have time to do predictions for the age group data but that would also have been interesting to see. Or even apply DMD to something else entirely different than to flu data since DMD can almost be applied to anything so long as you have the data. This would obviously take a decent amount of time in not only finding the data but also organising it so it was suitable for DMD. What we can look forward to is the DMD prediction for this 2019-2020 year 5.5, although with the Covid-19 outbreak this year the amount of influenza positive results may go up higher than usual, as more people may be getting tested more when they get ill, or it might also be lower than usual, as less people may be getting infected as the country is in lock down. Regardless if Covid-19 has had an effect on

the distribution of influenza positive results DMD would not have been able to predict this and the prediction will most likely be wrong.

7 This is an Appendix

7.1. Code 1.1

```
%%Define time and space discretizations
xi = linspace(-10,10,400);
t = linspace(0,4*pi,200);
dt = t(2) - t(1);
[Xgrid,T] = meshgrid(xi,t);

%% Create two spatiotemporal patterns
f1 = sech(Xgrid+4.3) .* (3*exp(1j*2.8*T));
f2 = (sech(Xgrid) .*tanh(Xgrid)) .*(4*exp(1j*3.6*T));

%% Combine signals and make data matrix
f = f1 + f2;
X = f.'; %Data Matrix

%% Visualize f1, f2, and f
figure;
subplot(2,2,1);
surfl(real(f1));
shading interp; colormap(gray); view(-20,60);
set(gca, 'Ytick', numel(t)/4 * (0:4)),
set(gca, 'Yticklabel' , {'0','\pi','2\pi','3\pi','4\pi'});
set(gca, 'XTick', linspace(1,numel(xi),3)),
set(gca, 'Xticklabel',{'-10','0','10'});
```

```
24 title('(a) f_{1}(x,t)')
25
26 subplot(2,2,2);
27 surfl(real(f2));
28 shading interp; colormap(gray); view(-20,60);
29 set(gca, 'Ytick', numel(t)/4 * (0:4)),
30 set(gca, 'Yticklabel' , {'0','\pi','2\pi','3\pi','4\pi'});
31 set(gca, 'XTick', linspace(1,numel(xi),3)),
32 set(gca, 'Xticklabel',{'-10','0','10'});
33 title('(b) f_{2}(x,t)')
34
35 subplot(2,2,3);
36 surfl(real(f));
37 shading interp; colormap(gray); view(-20,60);
38 set(gca, 'Ytick', numel(t)/4 * (0:4)),
39 set(gca, 'Yticklabel' , {'0','\pi','2\pi','3\pi','4\pi'});
40 set(gca, 'XTick', linspace(1,numel(xi),3)),
41 set(gca, 'Xticklabel',{'-10','0','10'});
42 title('(c) x(t)')
43
44 %% Create DMD data matrices
45 X1 = X(:, 1:end-1);
46 X2 = X(:, 2:end);
47
48 %% SVD and rank-2 truncation
49 r = 2; % rank truncation
50 [U, S, V] = svd(X1, 'econ');
51 Ur = U(:, 1:r);
52 Sr = S(1:r, 1:r);
53 Vr = V(:, 1:r);
54
55 %% Build Atilde and DMD modes
56 Atilde = Ur'*X2*Vr/Sr;
57 [W, D] = eig(Atilde);
58 Phi = X2*Vr/Sr*W; %DMD Modes
59
```

```
%% DMD Spectra
lambda = diag(D);
omega = log(lambda)/dt;

%% Compute DMD Solution
x1 = X(:, 1);
b = Phi\x1;
time_dynamics = zeros(r, length(t));
for iter = 1:length(t),
    time_dynamics(:,iter) = (b.*exp(omega*t(iter)));
end;
X_dmd = Phi*time_dynamics;

subplot(2,2,4);
surfl(real(X_dmd'));
shading interp; colormap(gray); view(-20,60);
set(gca, 'Ytick', numel(t)/4 * (0:4)),
set(gca, 'Yticklabel' , {'0','\pi','2\pi','3\pi','4\pi'});
set(gca, 'XTick', linspace(1,numel(xi),3)),
set(gca, 'Xticklabel',{'-10','0','10'});
title('(d) x_{DMD}(t)')
```

7.1.1. Code 1.2

```
%PCA computed by SVD
[U, S, V] = svd(X);
pc1 = U(:, 1); % first PCA mode
pc2 = U(:, 2); % second PCA mode
time_pc1 = V(:, 1); %temporal evolution of pc1
time_pc2 = V(:, 2); % temporal evolution of pc2
%Singular values plot
Stot = sum(diag(S));
Sdiag = diag(S);
M=1:20;
y=1:20;
for i=1:20
    M(i)=Sdiag(i)/Stot;
```

```
14 end
15 figure
16 scatter(y,M)
17 title('(a) Singular Values')
18 xlabel('i')
19 ylabel('\lambda_{i}')
20 %Error of DMD and PCA modes
21 c = real(Phi)\x1;
22 DMD2=norm(f2(1,:).'./c(2)-real(Phi(:,2)));
23 DMD1=norm(f1(1,:).'./(c(1))-real(Phi(:,1)));
24 PC2n=norm(f2(1,:).'./c(2)-real(pc2));
25 PC1n=norm(f1(1,:).'./(c(1))-real(pc1));
26 L=[PC1n,DMD1
27       PC2n,DMD2];
28 figure
29 barh(L)
30 title( '(b) Mode Error')
31 xlabel('Error')
32 legend({'PCA','DMD'})
33 ylabel('Mode')
34 %Spatial mode 2
35 figure
36 plot(xi,f2(1,:)./c(2))
37 hold on
38 plot(xi,real(Phi(:,2)), 'r--', 'linewidth',2)
39 hold on
40 plot(xi,real(pc2), 'k', 'linewidth',1)
41 title('Mode 2: Spatial')
42 xlabel('x')
43 %Spatial mode 1
44 figure
45 plot(xi,f1(1,:)./c(1))
46 hold on
47 plot(xi,real(Phi(:,1)), 'r--', 'linewidth',2)
48 hold on
49 plot(xi,real(pc1), 'k', 'linewidth',1)
```

```
50 title('Mode 1: Spatial')
51 xlabel('x')
52 legend('TRUE', 'DMD', 'PCA')
53 %Temporal mode 1
54 figure
55 plot(t,exp(1j*3.6*T))
56 hold on
57 plot(t,time_dynamics(2,:)./b(2), 'r--', 'linewidth',2)
58 hold on
59 plot(t,time_pc1.*sqrt(S(1,1)), 'k', 'linewidth',1)
60 hold off
61 title('Mode 1: Temporal')
62 set(gca,'XTick',0:pi:4*pi)
63 set(gca,'XTickLabel',{'0','\pi','2\pi','3\pi','4\pi'})
64 xlabel('t')
65 %Temporal mode 2
66 figure
67 plot(t,exp(1j*2.8*T))
68 hold on
69 plot(t,time_dynamics(1,:)./b(1), 'r--', 'linewidth',2)
70 hold on
71 plot(t,time_pc2.*sqrt(S(1,1)), 'k', 'linewidth',1)
72 title('Mode 2: Temporal')
73 set(gca,'XTick',0:pi:4*pi)
74 set(gca,'XTickLabel',{'0','\pi','2\pi','3\pi','4\pi'})
75 xlabel('t')
76
77 %plot all figures for DMD/PCA analysis
78 tiledlayout(6,2)
79 %First plot
80 nexttile([2,1])
81 scatter(y,M)
82 title('(a) Singular Values')
83 xlabel('i')
84 ylabel('\lambda_{i}')
85 %Second plot
```

```
nexttile([2,1])
barh(L)
title( '(b) Mode Error')
xlabel('Error')
legend({'PCA','DMD'})
ylabel('Mode')
%Third plot
nexttile([1,2])
plot(t,exp(1j*3.6*T))
hold on
plot(t,time_dynamics(2,:)./b(2), 'r--', 'linewidth',2)
hold on
plot(t,time_pc1.*sqrt(S(1,1)), 'k', 'linewidth',1)
hold off
title('(c) Mode 1: Temporal')
set(gca,'XTick',0:pi:4*pi)
set(gca,'XTickLabel',{'0','\pi','2\pi','3\pi','4\pi'})
xlabel('t')
%Fourth plot
nexttile([1,2])
plot(t,exp(1j*2.8*T))
hold on
plot(t,time_dynamics(1,:)./b(1), 'r--', 'linewidth',2)
hold on
plot(t,time_pc2.*sqrt(S(1,1)), 'k', 'linewidth',1)
title('(d) Mode 2: Temporal')
set(gca,'XTick',0:pi:4*pi)
set(gca,'XTickLabel',{'0','\pi','2\pi','3\pi','4\pi'})
xlabel('t')
%Fifth plot
nexttile([1,2])
plot(xi,f1(1,:)./c(1))
hold on
plot(xi,real(Phi(:,1)), 'r--', 'linewidth',2)
hold on
plot(xi,real(pc1), 'k', 'linewidth',1)
```

```
title('(e) Mode 1: Spatial')
xlabel('x')
legend('TRUE', 'DMD', 'PCA')
%Sixth plot
nexttile([1,2])
plot(xi,f2(1,:)./c(2))
hold on
plot(xi,real(Phi(:,2)), 'r--', 'linewidth',2)
hold on
plot(xi,real(pc2), 'k', 'linewidth',1)
title('(f) Mode 2: Spatial')
xlabel('x')
set(gcf,'color','w');
```

7.2. Flu Data

```
%% Create DMD data matrices
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);
t=1:17;
dt=1;
%% SVD and rank-16 truncation
r = 16; % rank truncation
[U, S, V] = svd(X1, 'econ');
Ur = U(:, 1:r);
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);

%% Build Atilde and DMD modes
Atilde = Ur'*X2*Vr/Sr;
[W, D] = eig(Atilde);
Phi = X2*Vr/Sr*W; %DMD Modes

%% DMD Spectra
lambda = diag(D);
omega = log(lambda)/dt;

```

```
22 %%b approximation
23 for i=1:16
24     A{i}=Phi*D^i;
25 end
26 Phi1=[Phi;A{1};A{2};A{3};A{4};A{5};A{6};A{7};A{8};A{9};A{10};A{11};A
       {12};A{13};A{14};A{15};A{16}];
27 %% Compute DMD Solution
28 x1 = X(:, 1);
29 %b = Phi1\X(:);
30 b = Phi\x1;
31 %b = pinv(Phi)*x1;
32 time_dynamics = zeros(r, length(t));
33 for iter = 1:length(t),
34     time_dynamics(:,iter) = (b.*exp(omega*t(iter)));
35 end;
36 X_dmd = Phi*time_dynamics;
37 %% Reconstruction of true signal
38 F=[];
39 B=[];
40 for j=1:17
41 for i=1:16 %number of modes you want to use
42 F(:,i)= Phi(:,i)*exp(log(D(i,i))*j)*b(i); %For loop reconstructs x(j)
43 end
44 Fm=sum(F'); %sums the columns of F and returns a row vector
45 B(:,j)=Fm; %B(:,j) is the approximation of X(:,j)
46 end
47 plot(real(B(:)), 'b');
48 hold on
49 plot(X(:), 'k')
50 err = sqrt(sum( (real((X(:)-B(:))).^2)))
51
52 %% plot of 17 years of flu data and prediction of 17th year
53 H=[];
54 NP=8; %NP is the number of predictions
55 Var=0.0001; %variance you want to use
56 bb=b+sqrt(Var)*randn(16,NP);
```

```
H=Phi*exp(log(D)*17)*bb;
%plot(X(:), 'linewidth',2)
%hold on
%plot([833:884],real(H), '--')
```

7.3. Flu Data Age Groups "Original code written by Naratip Santitissadeekorn"

```
 %52 wks, 4 age groups, 17 years
%Mat(Mat==0)=1e-7;
Mat_mean = mean(Mat,3);
%---choose 1D or 2D (blanking out the one you do not choose)-----
X=reshape(Mat,52*4,[]); IDplot=2; %seprate age group
%X = squeeze(sum(Mat,2)); IDplot =1; %combine age group
%-------------------------------------------------------------------

%=====Prepare data for DMD
    ================================================
 [ndim,nlength] = size(X);
 nsnap=16;
 XX = X(:,1:nsnap);
 mean_sub=1;
 if mean_sub==1, XX = XX-mean(XX,2); end
 X1 = XX(:,1:end-1);
 %X1=X1-mean(X,2);
 X2 = XX(:,2:end);
 %X2=X2-mean(X,2);



%===run DMD
    ================================================================
r=12;
[U,S,V] = svd(X1);
Ur = U(:,1:r);
Sr = S(1:r,1:r);
Vr = V(:,1:r);
```

```
29 Atilde = Ur'*X2*Vr/Sr;
30 [W,D] = eig(Atilde);
31 Phi = X2*Vr/Sr*W;
32
33
34 lambda = diag(D);
35 omega = log(lambda);
36
37
38 nfit = 1;  %choose the number of snapshot to fit
39 Mtmp = zeros(ndim*nfit,r);
40 for i=1:nfit
41     Mtmp(1+ndim*(i-1):i*ndim,:)=Phi*D^(i-1);
42 end
43 tmpX = XX(:,1:nfit)';
44 b = pinv(real(Mtmp))*tmpX(:);
45
46 %%
47 F= [];
48 nleft = nlength-nsnap;
49 run_length=nsnap-1+nleft;
50 for iter=0:run_length
51     F(:,iter+1) = b.*exp(omega*iter);
52 end
53 X_dmd = real(Phi*F);
54 if mean_sub==1, X_dmd=X_dmd+mean(X(:,1:nsnap),2);  end
55
56 Err = (X_dmd(:,1:nsnap)-X(:,1:nsnap)).^2; Err = sqrt(sum(Err(:)));
57
58
59 % X_dmdn = X_dmd;
60 % for j=1:run_length+1, X_dmdn(:,1) = X_dmd(:,j)/sum(X_dmd(:,j)); end
61
62
63 if IDplot==1
64     figure
```

```
65      plot(X(:),'k--','linewidth',1), hold on
66      plot(X_dmd(1:nsnap*ndim),'r-','linewidth',1)
67      plot(1+nsnap*ndim:nlength*ndim,X_dmd(1+nsnap*ndim:end),'g-','
            linewidth',2)
68      title(['Fitting error=',num2str(Err)],'fontsize',14)
69      set(gcf,'color','w')
70  else
71      figure
72      subplot(211)
73      imagesc(Mat_mean'), colorbar
74      title('Data average')
75      subplot(212)
76      imagesc(reshape(real(Phi(:,1)),52,4)'), colorbar
77      title('First mode')
78      set(gcf,'color','w')
79
80      %XX = exp(XX);
81      %X_dmd= exp(X_dmd);
82      figure
83      choose_yr=1;
84      subplot(2,1,1)
85      imagesc(reshape(XX(:,choose_yr),52,4)'), colorbar
86      title(['Year = ',num2str(choose_yr)])
87      subplot(2,1,2)
88      imagesc(reshape(X_dmd(:,choose_yr),52,4)'), colorbar
89      title(['Year = ',num2str(choose_yr)])
90      set(gcf,'color','w')
91  end
92
93
94  %X_dmd = CLR_inv(X_f');
```

References

[1] J. N. Kutz, Data-Driven Modeling and Scientific Computation: Methods for Complex Systems and Big Data, 2013.

[2] J. Shlens, A Tutorial on Principal Component Analysis 2014.

[3] Gilbert Strang, RES.18-010 A 2020 Vision of Linear Algebra. Spring 2020. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA

[4] Centers for Disease Control and Prevention, https://www.cdc.gov/flu/weekly/fluviewinteractive.htm .

[5] Centering Data Improves the Dynamic Mode Decomposition, Seth M. Hirsh and Kameron Decker Harris and J. Nathan Kutz and Bingni W. Brunton, 2019, eprint = 1906.05973

[6] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, SIAM, 2016 ISBN: 1611974496, 9781611974492