May 2018

Analysis of Dynamic Mode Decomposition

ARCHANA MUNIRAJU
University of Wisconsin-Milwaukee

ANALYSIS OF DYNAMIC MODE DECOMPOSITION

by

Archana Muniraju

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Engineering

at

University of Wisconsin Milwaukee

May 2018

ABSTRACT

ANALYSIS OF DYNAMIC MODE DECOMPOSITION

by

Archana Muniraju

The University of Wisconsin-Milwaukee, 2018
Under the Supervision of Professor Zeyun Yu

In this master thesis, a study was conducted on a method known as Dynamic mode decomposition(DMD), an equation-free technique which does not require to know the underlying governing equations of the complex data. As a result of massive datasets from various resources, like experiments, simulation, historical records, etc. has led to an increasing demand for an efficient method for data mining and analysis techniques. The main goals of data mining are the description and prediction. Description involves finding patterns in the data and prediction involves predicting the system dynamics. An important aspect when analyzing an algorithm is testing. In this work, DMD-a data based technique is used to test different cases to find the underlying patterns, predict the system dynamics and for reconstruction of original data. Using real data for analyzing a new algorithm may not be appropriate due to lack of knowledge of the algorithm performance in various cases. So, testing is done on synthetic data for all the cases discussed in this work, as it is useful for visualization and to find the robustness of the new algorithm. Finally, this work makes an attempts to understand the DMD's performance and limitations better for the future applications with real data.

TABLE OF CONTENTS

1 Introduction **1**

1.1 A brief history . . . 2

1.2 Examples . . . 3

1.2.1 Foreground and background separation using DMD . . . 3

1.2.2 DMD on infectious Disease data . . . 6

1.2.3 DMD on neural recordings . . . 9

2 Dynamic Mode Decomposition(DMD) **13**

2.1 DMD Related Decomposition and Techniques . . . 13

2.1.1 Arnoldi Method . . . 13

2.1.2 Principal Component Analysis(PCA) . . . 15

2.1.3 Singular Value Decomposition(SVD) . . . 17

2.2 Mathematical Introduction to DMD . . . 21

2.2.1 Algorithm . . . 21

2.2.2 Eigendecomposition . . . 23

2.2.3 Reconstruction . . . 24

3 Testing Examples for DMD **28**

3.1 Interpretation of DMD modes and Eigenvalues . . . 28

3.1.1 Interpretation of DMD Modes Φ . . . 28

3.1.2 Interpretation of DMD Eigenvalues Λ . . . 28

3.1.3 Interpretation of continuous-time Eigenvalues Ω . . . 29

3.2 CASE 1: DMD on Simple Data . . . 32

3.3 CASE 2: DMD on data set with modes oscillating at different frequency 34

3.4 CASE 3: DMD on data with Circle growing and decaying 36

3.5 CASE 4: DMD on Linear-Motions 37

3.5.1 Rectilinear Motion 37

3.5.2 Curvilinear Motion 41

3.6 CASE 5: DMD on synthetic data that mimics Human Heart Beat 44

3.7 CASE 6: DMD for Transient time behavior 46

4 Conclusion **48**

References **49**

LIST OF FIGURES

1.1 The result for DMD and RPCA background/foreground separation in the ”Parked Vehicle” video. 5

1.2 The result for DMD and RPCA background/foreground separation in the ”Abandoned Bag” video. 6

1.3 Google Flu data and output of DMD. 7

1.4 Pre-Vaccination measles data and output of DMD. 8

1.5 Type 1 paralytic polio data and output of DMD. 9

1.6 An illustration of coupled spatial-temporal modal decomposition by DMD on ECoG data. 10

1.7 A summary for extracting spindle networks based on the DMD spectra of windowed ECoG data. 12

2.1 Describing principal component analysis. 16

2.2 Singular Value Decomposition. 20

3.1 Spectrum plot of Eigenvalue Λ and Eigenvalue Ω 30

3.2 Time Evolution of Eigenvalues Ω 30

3.3 Data Visualizing and Eigenplot for case1. 32

3.4 Temporal evolution for case1 analysis. 33

3.5 Original data for case2 analysis. 34

3.6 Reconstructed data for case2 analysis. 34

3.7 Temporal evolution for case2 analysis. 35

3.8 Temporal evolution for case3 analysis. 36

3.9 Visualizing data for growing and decaying circles 36

3.10 Reconstruction for Rectilinear motion. 39

3.11 Eigenvalue plot for Rectilinear motion. 40

3.12 Reconstruction for curvilinear motion with SVD truncation $r = 1$. 41

3.13 Reconstruction for curvilinear motion with SVD truncation $r = 10$. . . . 42

3.14 Eigenvalue plot for curvilinear motion. 43

3.15 Eigenvalue plot for case5 analysis. 45

3.16 Data visualization and reconstruction for transient behavior. 46

3.17 Eigenvalue plot for Transient behavior. 47

Chapter 1

Introduction

The data-based techniques have been rapidly developed over the past two decades and widely used in numerous industrial sectors nowadays. The core of data-based methods is to take full advantage of the enormous amounts of available data, aiming to acquire the useful information within. Compared with the well-developed model-based approaches, data-based techniques provide efficient alternative solutions.Data-based modal decomposition is frequently employed to investigate complex dynamical systems. When dynamical operators are too sophisticated to analyze directly because of nonlinearity and high dimensionality, then in such cases data-based techniques are often more practical. Such methods numerically analyze empirical data produced by experiments or simulations, yielding modes containing dynamically significant structures.

In this thesis work, a data-based decomposition method referred to as the dynamic mode decomposition(DMD) will be discussed and applied to synthetic data, that attempts to extract dynamic information from the data without depending on the availability of a model, but instead is based on a sequence of snapshots of data. DMD method can be applied equally well to numerically generated and experimentally measured data. A brief description of the underlying idea of this method and related applications will be discussed. Few example cases will be analyzed to demonstrate the potential of this technique.

1.1 A brief history

Introduced in the fluid mechanic's community, Dynamic Mode Decomposition(DMD) traces its origins to Bernard Koopman in 1931 and be a particular case of Koopman theory. Initially, DMD was used to analyze the time evolution of fluid flows. In recent years DMD has quickly gained its popularity and has emerged as a significant tool for analyzing dynamics of nonlinear systems. When we see DMD in the last few years alone, it has made immense progress in both theory and application. When we look into the theory of DMD, it has seen innovations around compressive architectures, multi-resolution analysis, and de-noising algorithms. DMD has not only made its progress in fluid dynamics, but also it has been applied to new domains such as neuroscience, epidemiology, robotics, and the current application of video processing and computer vision. In the latter part, there will be a brief discussion on applications of DMD on different domains to understand the potential of DMD.

The dynamic mode decomposition(DMD) is an equation-free, data-driven matrix decomposition method. This method can provide accurate reconstructions of coherent spatiotemporal structures arising in nonlinear dynamical systems or short-time future estimates of such systems. The DMD method offers a regression technique for the least-square fitting of snapshots to a linear dynamical system. Dynamic mode decomposition approximates the modes of the Koopman operator, which is a linear, infinite-dimensional operator that represents nonlinear, infinite-dimensional dynamics without linearization and is the adjoint of the Perron-Frobenius operator. The method can be viewed as computing eigenvalues and eigenvectors of a linear model that approximates the underlying dynamics, even if those dynamics are nonlinear. Unlike Proper Orthogonal Decomposition(POD) and balanced POD, this decomposition yields growth rates and frequencies associated with each mode, which can be found from the magnitude and phase of each corresponding eigenvalue. If the data generated by a linear dynamical operator, then the decomposition recovers the leading eigenvalues and eigenvectors of that operator. If the data generated is periodic, then the decomposition is equivalent to a temporal discrete Fourier transform (DFT).

Describing a nonlinear system by a superposition of modes whose dynamics are governed by eigenvalues may seem dubious at first place. After all, one needs a linear operator to talk about eigenvalues. However, we can see in (Tu, Rowley, Luchtenburg, Brunton, & Kutz, 2013) that DMD is closely related to a spectral analysis of the Koopman operator. The Koopman operator is a linear but infinite-dimensional operator whose modes and eigenvalues capture the evolution of observables describing any (even nonlinear) dynamical system. The use of its spectral decomposition for data-based modal decomposition and model reduction was proposed first in (Mezić, 2005). DMD analysis can be a numerical approximation to Koopman spectral analysis, and it is in this sense that DMD applies to nonlinear systems. In fact, the terms DMD mode and Koopman mode are often used interchangeably in this literature.

1.2 Examples

1.2.1 Foreground and background separation using DMD

As the demand for the accurate and real-time video surveillance techniques increases. The algorithms that can remove background variations in a video stream are at the frontline of modern data-analysis research. Most of the modern computer vision applications require algorithms to be implemented in real-time and that are robust enough to handle diverse, complicated, and cluttered backgrounds. A variety of iterative techniques and methods have already been developed to perform background/foreground separation. In this paper(Grosek & Kutz, 2014), the authors introduce the method of dynamic mode decomposition (DMD) for robustly separating video frames into the background (low-rank) and foreground (sparse) components in real-time. Also, using the Advanced Video and Signal based Surveillance (AVSS) Datasets, specifically the Parked Vehicle - Hard and Abandoned Bag -Hard videos, the DMD separation procedure is compared and contrasted against the RPCA procedure.

In this application, the data matrix for DMD consists of snapshots of video frames. As video frames are 2D by nature, they are reshaped into 1D column vector and united into a single data matrix X. Applying DMD to this dataset yields oscillatory time components(DMD modes and eigenvalues) that have contextual implications. The DMD modes that are near the origin are the stationary background pixels as they represent dynamics that are unchanging or changing slowly or low-rank components of the data matrix X. In constrast the DMD modes that are bounded away from the origin are faster and represent the foreground motion in the video or the sparse components of the data matrix. When RPCA algorithm is applied to data matrix, it perfectly separates the given data X according to $X = L + S$, where L has a low-rank, and S is sparse. The low-rank matrix L will yield the video of just the background, and the sparse matrix S will yield the video of the moving foreground objects. The DMD method can attempt to reconstruct any given frame, or even possibly future frames, by calculating xDMD(t) at the corresponding time t. The AVSS video stream data is analyzed individually using both the RPCA and DMD methods.

Frame numbers 500, 1000, and 2000 of the entire video stream is depicted in Figures 1.1 and 1.2, along with their separation results for easy comparison. DMD method seems to eliminate more spurious pixels in its sparse results that may pertain to the background when compared to the RPCAs sparse results. When the dataset contains various motions, such as walking, standing, and sitting thier shadows are relevant in this video because they do move with their respective foreground objects, and tend to change the background significantly enough to be viewed as extensions of the moving objects themselves. The RPCA and DMD methods struggle with the dataset of this kind, but DMD

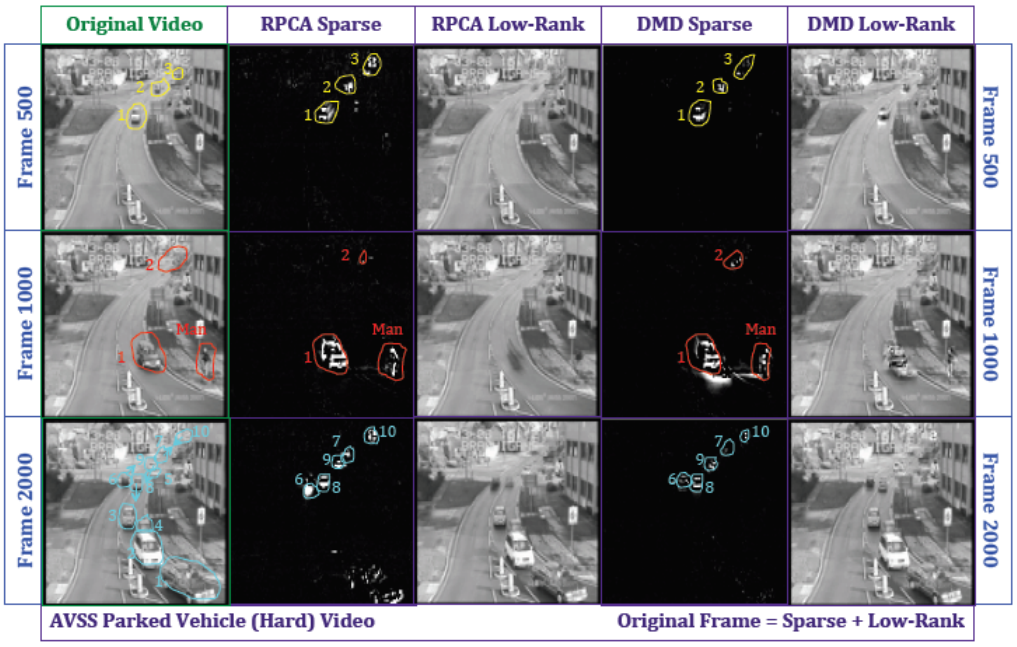


Figure 1.1: The results for DMD and RPCA background/foreground separation are illustrated for 3 specific frames in the Parked Vehicle video. The 30 frame video segment that contains frame #500 has 3 vehicles driving in the same lane toward the camera. The frame #1000 has a man stepping up to a crosswalk as a vehicle passes by, and a second car in the distance, barely perceptible, starts to come into view. The frame #2000 has 3 vehicles stopped at a traffic light at the bottom of the frame with another 2 vehicles parked on the right side of the road, and 5 moving vehicles, 2 going into the distance and 3 coming toward the vehicles waiting at the light, the last vehicle being imperceptible to the eye at this pixel resolution: $n = 11520$.

method is generous in depicting these movement trails in its sparse results, while the RCPA method is more conservative. The RPCA and DMD methods have a comparably good background/ foreground separation results but the real difference in effectiveness between the RPCA and DMD algorithms is found in the amount of computational time needed to complete the background/foreground separation. From the results, it was clear that the DMD method was about two to three orders of magnitude faster than its RPCA.

Finally, it was concluded that DMD method could be used for foreground/background separation in videos with excellent computational efficiency. The results produced by DMD method are on par with the quality of separation achieved with the RPCA method for realistic video scenarios. But the results obtained by DMD were orders of magnitude faster. When dealing with high-resolution images or many frames per video segment,

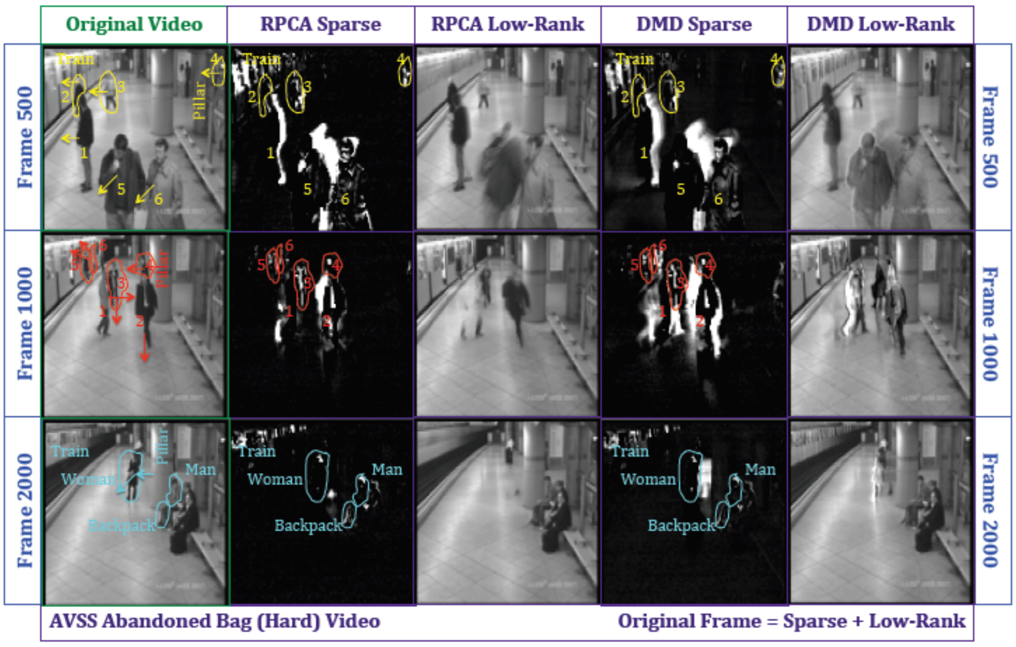


Figure 1.2: The results for DMD and RPCA background/foreground separation are illustrated for 3 specific frames in the Abandoned Bag video. The 30 frame video segment that contains frame #500 has 3 people stepping slightly closer to an arriving train, and another person walks behind a support pillar, toward the train in the upper right area of the frame. The two people closest to the camera walk toward the bottom of the frame. The frame #1000 has 4 people walking in different directions in the middle of the frame, one of which comes out from behind a support pillar, while, farther down the platform, 2 people enter the train. The frame #2000 has a woman walk out from behind a support pillar, moving to the left, and then turned somewhat toward the camera. The train is moving and the man sitting closest to the support pillar adjusts his backpack.

working with DMD and RPCA can be problematic because of limited memory size and reduced computational speed.

1.2.2 DMD on infectious Disease data

There is a rapid increase in surveillance systems for infectious disease, capacity for digital storage and computational resources. A stronger understanding of the underlying process of infectious disease spread becomes vital to fight against the spread of infectious disease in the human population. This paper (Proctor & Eckhoff, 2015), demonstrates has how the DMD method can help in the analysis of infectious disease data. DMD method is used on three different infectious disease sets including Google Flu Trends data, pre-vaccination measles in the UK, and paralytic poliomyelitis wild-type-1 cases in Nigeria. After applying DMD to this datasets, the dynamic modes obtained describe how spatial locations are related. Each element in a single dynamic mode has two essential pieces of information. First being the magnitude of the element(absolute value) provides a measure of the spatial locations participation in the mode. Second is the angle between the real and imaginary component of the element provides a measure of a locations phase of oscillation relative to others for that modes frequency.

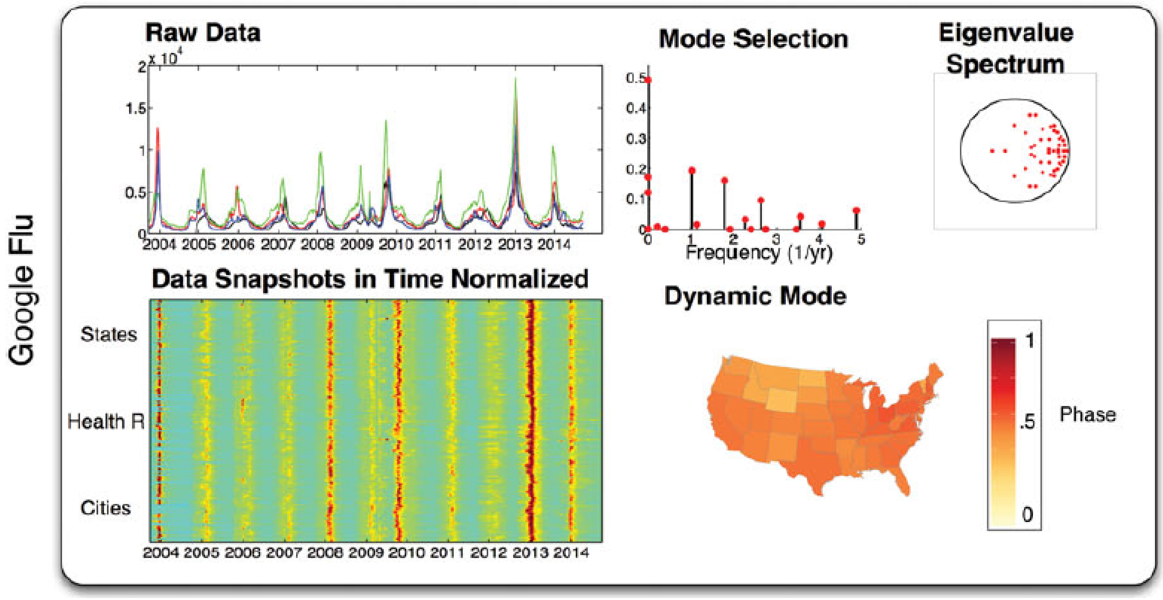


Figure 1.3: Example1: Google Flu data and output of DMD

The first example of infectious disease data for DMD analysis is obtained from Googles Flu Trends tool. This tool provides data for every seven days. Also, data for DMD analysis is focused solely on the state information to visualize every element of the dynamic mode on the map of the US. In figure 1.3, four traces of the raw from Alaska (black), California (red), Texas (green) and New York (blue) are shown for comparison. In figure 1.3 the output of DMD is shown to the right of the data visualizations. The eigenvalue spectrum indicates some modes that are within the unit circle meaning fast decaying eigenvalues and modes that do not contribute to the broader structure of the dynamics. The phase of the dynamic mode associated with the one-year frequency is plotted on the

US map.

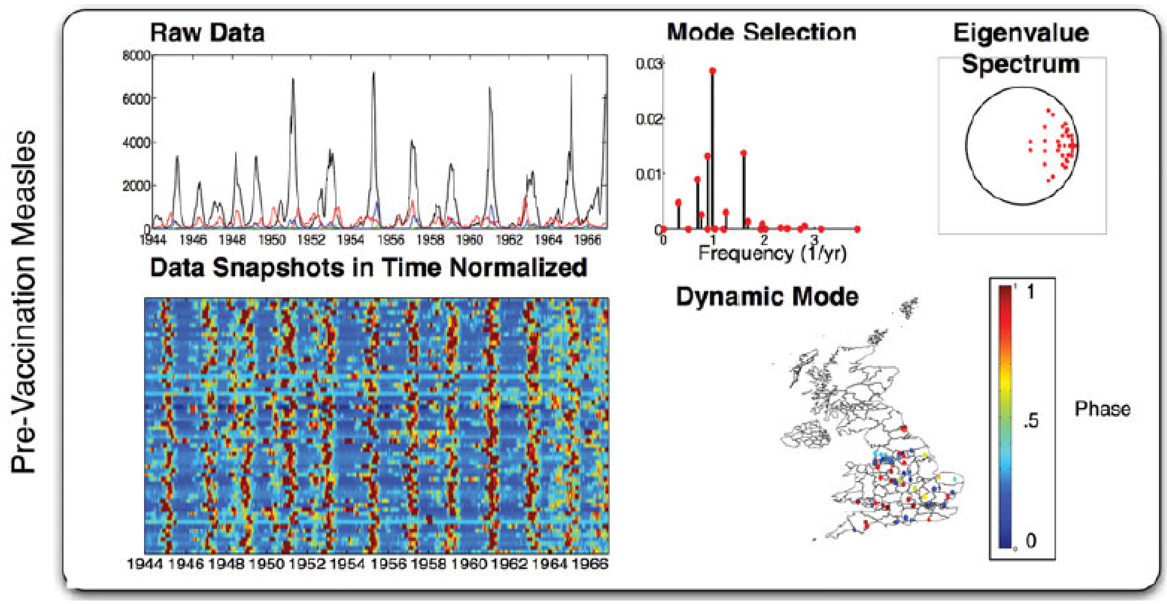


Figure 1.4: Example2: Pre- vaccination measles data and output of DMD

In the second example, DMD analysis is performed on infectious disease data-set of pre-vaccination measles cases in the UK. In figure 1.4, four traces of the raw data or unprocessed data from four cities in the UK: London(black), Liverpool (red), Colchester (green) and Cardiff (blue), are shown for comparison. In the dataset, sixty cities have been included, and first 10years data is used for DMD analysis. DMD output is shown on the right-hand side of figure 1.4. Individual markers are placed down at each citys latitude and longitude with the color indicating the phase. The phase differences for measles in the UK span a larger range than theGoogles Flu case. For example, the difference between London and Warrington is approximately 0.39, almost five months.

The final example involves the analysis of data about wild-type1 polio paralytic cases from Nigeria. The figure 1.5 shows raw data and visualization of all of the sub-province (LGA) level spatial divisions. The four LGAs plotted are Kano (black), Katsina (red), Akko (green), and Funakaye (blue). Here for DMD analysis, they focus on 5-years of data and considers LGAs with more than five cases. The output of the DMD analysis is shown to the right of the data visualizations. When the eigenvalue spectrum of this

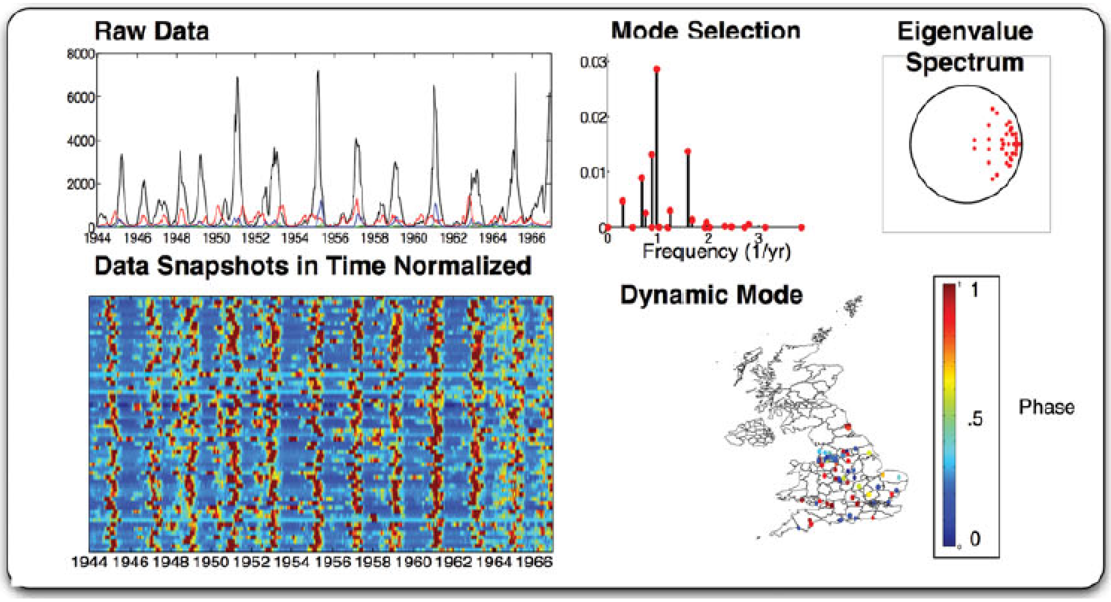


Figure 1.5: Example3: Type 1 paralytic polio data and output of DMD

case is compared to the eigenvalue spectrum of last two cases, it is different and having more eigenvalues near the unit circle and evenly spaced. This is characteristic of a signal decomposition with a broad frequency content.

1.2.3 DMD on neural recordings

In this paper (Brunton, Johnson, Ojemann, & Kutz, 2016), Dynamic mode decomposition(DMD) was applied to a large dataset containing neural recordings to analyze the large-scale data. In this case, data is obtained from tens or thousands of electrodes simultaneously recording dynamic brain activity over minutes to hours. So the neuroscience community requires this big data to be understood and visualized. The dynamic data acquired from the recording of neural activity are characterized by coherent patterns across both space and time. Here DMD method is validated on sub-dural electrode array recordings from human subjects performing a known motor activation task. Also, DMD in combination with machine learning is used to develop a method to extract sleep spindle networks. Dynamic mode decomposition is a robust method which can be used in the analysis and interpreting of large-scale recordings of neural activity. Few distinct patterns describe the activity of the complex networks of neurons. Identifying these spatial-temporal patterns helps in the reduction of complex measurements. The reduction can be made by projecting onto coherent structures and therefore it gets easier to build dynamical models and apply machine learning tools for pattern analysis.

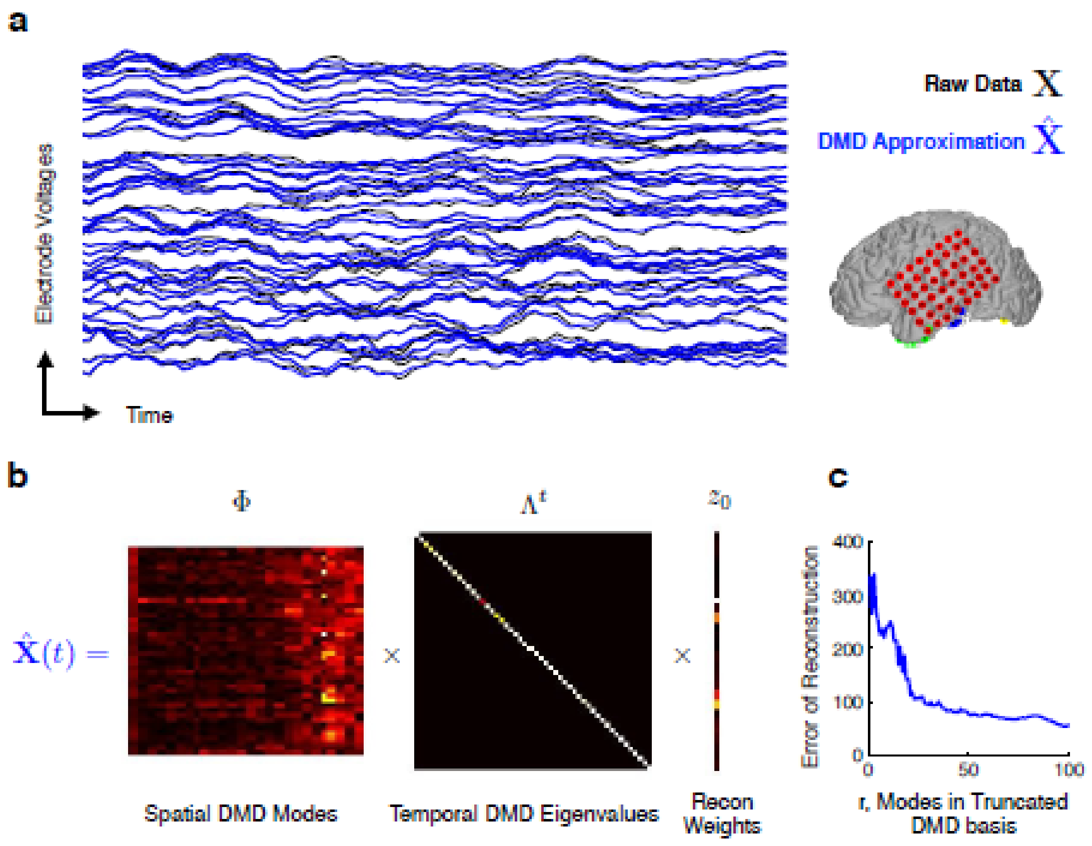


Figure 1.6: An illustration of coupled spatial-temporal modal decomposition by DMD applied to a 500 msec window of 45-channel ECoG data sampled at 200 Hz. The raw data X (black traces in 'a') is approximated as $\hat{X(t)}$ (blue traces in a). Voltages are shown in normalized units. 'b' shows that $\hat{X}$ is the product of spatial modes Φ and temporal eigenvalues Λ^t. 'c' shows the decrease in reconstruction error as more modes are included in a truncated, low-rank DMD basis.

The Dynamic mode decomposition's applicability to large-scale neural recordings is demonstrated by analyzing sub-dural electrode array recordings from human subjects in two different contexts. First, DMD is used to derive sensorimotor maps based on a simple movement task. Next, DMD in combination with machine learning techniques is used to detect and characterize spindle networks present during sleep. In neuroscience, we are often interested in electrode arrays that have tens of channels sampled at hundreds of samples per second, so for a data matrix X over a one-second window of data,

$n < (m-1)$. Where n is the number of observable locations, and m is the number of snapshots in time. For instance, these measurements may be voltages from n channels of an electrode array sampled every Δt. As the data matrix is $n < (m-1)$ the application of the DMD algorithm to arrays of large-scale recordings of neuronal activity may require an adaptation to the normal procedure. The modified versions of the data matrices are constructed, appending to the snapshot measurements with time-shifted versions of themselves, thus augmenting the number of channels to be hn. For our multichannel recordings, smallest integer h is chosen so that $hn > 2M$.

In the first case of sensorimotor mapping, DMD was analyzed on datasets recorded from sub-dural electrode arrays implanted in two patients monitored for seizure localization. To verify DMD approach to spatial-temporal pattern analysis sensorimotor maps were derived based on a simple motor repetition task where the subjects were instructed to move either their tongue or the hand contralateral to the implanted electrode array. An analysis of DMD spectrum was performed and derived motor maps of the tongue and hand movements that are firmly consistent with some of the previously described results. DMD spectra were computed for each epoch of the motor screen, and DMD modes within a low frequency band (8 to 32 Hz) and a high-frequency band (76 to 100 Hz) were averaged to derive mean modes in each band. In the second case, DMD was used to extract and identify sleep spindle networks in human ECoG recordings. DMD spectra of windowed ECoG data is used in the extraction of spindle network. The DMD modes corresponding to larger than expected power in the spindle band (11 to 17 Hz) were collected in a library, whose elements were then clustered to determine stereotypes of the spindle networks in the recording. Each spindle network stereotype represents the strength of spatial correlations between electrodes and may be visualized on a grid of electrode location.

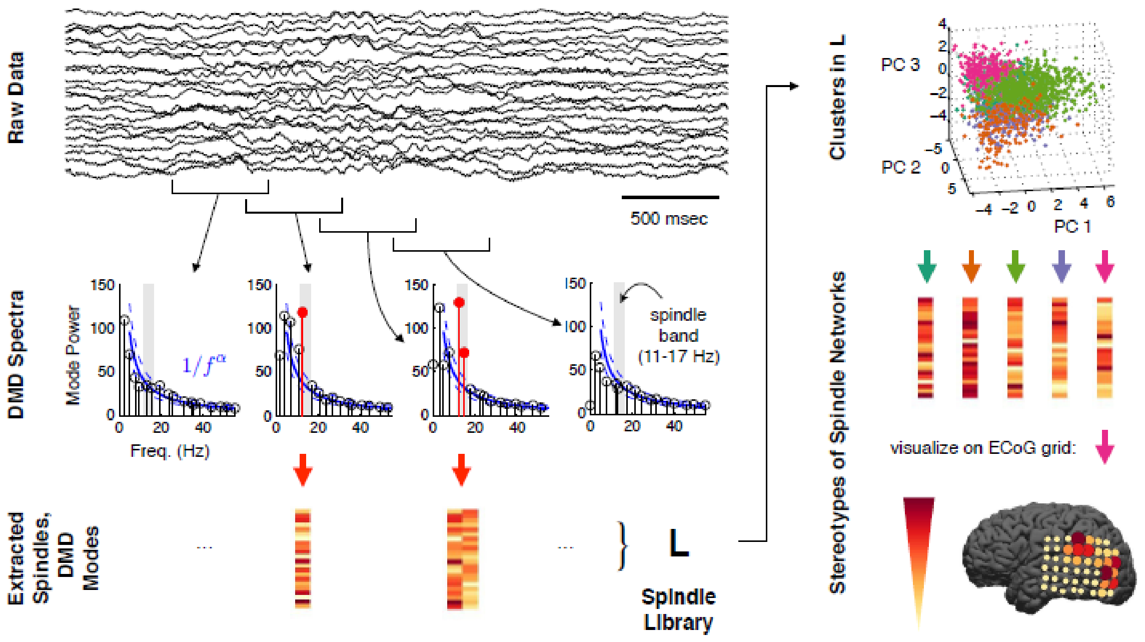


Figure 1.7: A summary of how spindle networks were extracted based on the DMD spectra of windowed ECoG data. DMD spectra were computed for windows of raw data, and DMD modes in the spindle band (11 to 17 Hz, gray) whose power significantly exceeded the expected $1/f^{\alpha}$ distribution (blue lines) were extracted and gathered into L. This library L was then clustered using a Gaussian mixtures model to discover distinct spindle networks, and the centroids of these clusters were stereotypes of spindle networks that could be visualized on the ECoG grid. Each spindle network represented correlated groups of electrodes with activity in the spindle band.

Finally, this paper presented the novel adaption of a modal decomposition technique known as dynamic mode decomposition (DMD) to analyze and visualize large-scale neural recordings. The neural recordings are often characterized by coherent patterns of activity in space (across many channels) at multiple temporal frequencies. Thus this paper demonstrated how DMD could be used to extract the coherent patterns in the data. This extraction through DMD was validated on a simple motor task where selective and separable regions of sensorimotor cortex were identified. Later it was used to detect and analyze sleep spindle networks from large-scale human ECoG recordings.

Chapter 2

Dynamic Mode Decomposition(DMD)

2.1 DMD Related Decomposition and Techniques

2.1.1 Arnoldi Method

The Arnoldi Method makes use of iterative schemes to extract global stability modes. The large stability matrices which are a result of global stability analyses for flows in complex geometries have put forward considerable strain on computational resources. It becomes costly to use direct methods, the method of choice for simple problems, and iterative schemes have thus to be employed to extract the global stability modes.The Arnoldi Method has been particularly successful in this respect.In this method, reduction of the high-dimensional System matrix is obtained by projecting it on a lower-dimensional subspace know as Krylov subspace. In this way, we can make an efficient computation of the dominant eigenvalues and corresponding eigenvectors of the full system.

Basic Arnoldi algorithm

Let's say we have a square matrix A of order n if Arnoldis method is carried out for n steps then an Orthogonal reduction to Hessenberg form is achieved, $AV = VH$. Where H is an upper Hessenberg matrix of order n with positive sub-diagonal elements, and V

is an orthogonal matrix. The first column of V uniquely determines the matrix V and H, a unit-norm vector $v1 = Ve1$ that is called the initial vector. For some initial vectors, Arnoldis method fails after m steps, and in that case, the algorithm produces an nxm matrix Vm with orthogonal columns and an upper Hessenberg matrix of order m, H_m, that satisfy:

$$AV_mV_mH_m = 0$$

Basic Algorithm(Saad, 1992)

```
Input: Matrix $A$, number of steps m, and initial vector $v_1$ of norm 1
Output: $(V_m, H_m, f, \beta)$ so that $AV_mV_mH_m = fe_m^*$, $\beta = \|f\|_2$
For $j = 1, 2, ..., m - 1$
$w = Av_j$
Orthogonalize $w$ with respect to $V_j$ (obtaining $h_{1:j,j}$
$h_{j+1,j} = \|w\|_2$
If $h_{j+1,j} = 0$, stop
$v_{j+1} = w/h_{j+1,j}$
end
$f = Av_m$
Orthogonalize $f$ with respect to $V_m$ (obtaining $h_1 : m, m)$
$\beta = \|f\|_2$.
```

In practical situations, the number of steps m, required to obtain good approximations may be too large. When the number of steps m is too large, it results in storage problems, and also the computational cost grows in every step. Moreover, the Arnoldi Method is based on a QR decomposition resulting in a Hessenberg matrix H, in which eigenvalues represent or approximate the eigenvalues of the linear operator system matrix A. This results in a stable algorithm, in which the system matrix A should be known. In physical experiments, this system matrix A is not available. DMD can be interpreted as an extension of the classical Arnoldi algorithm used to determine eigen-elements of large size problems.

2.1.2 Principal Component Analysis(PCA)

Principal Component Analysis (PCA) is a dimensionality-reduction technique that is used to transform a high-dimensional dataset into a low-dimensional subspace, i.e., into a new coordinate system. It is a way of identifying patterns in a high dimensional data. In simplest terms, we can define PCA has a method to transforms the original interrelated variables into a new set of uncorrelated variables called Principal Components. In the new coordinate system, the first axis corresponds to the first principal component, which is the component that explains the highest amount of the variance in the data. Here variance is defined as the amount of information expressed by each principal component.

Necessary steps to compute PCA are as follows:

1. Data acquisition. Typically, a number of measurements are collected in a single experiment, and these measurements are arranged into a row vector.

2. Subtracting the Mean. For proper functioning of PCA, we have to subtract the mean from each of the data dimensions. This produces a data set whose mean is zero.

3. Calculate the Covariance Matrix.

4. Calculate the Eigenvectors and Eigenvalues of the covariance matrix.

5. Choosing principal components and forming a feature vector.

Lets say the original dataset has two variables, $x1$ and $x2$ with the data spread as shown in the figure 2.1a. Principal components are calculated using PCA, and the first principal component explains the highest amount of variance. The line in the figure 2.1b signifies the first principal component. Here we have the 2D data, and if we need to reduce the data to 1D, then we project the data onto first principal component as shown in the figure 2.1c (Hamilton, 2014 (accessed April 10, 2018)).

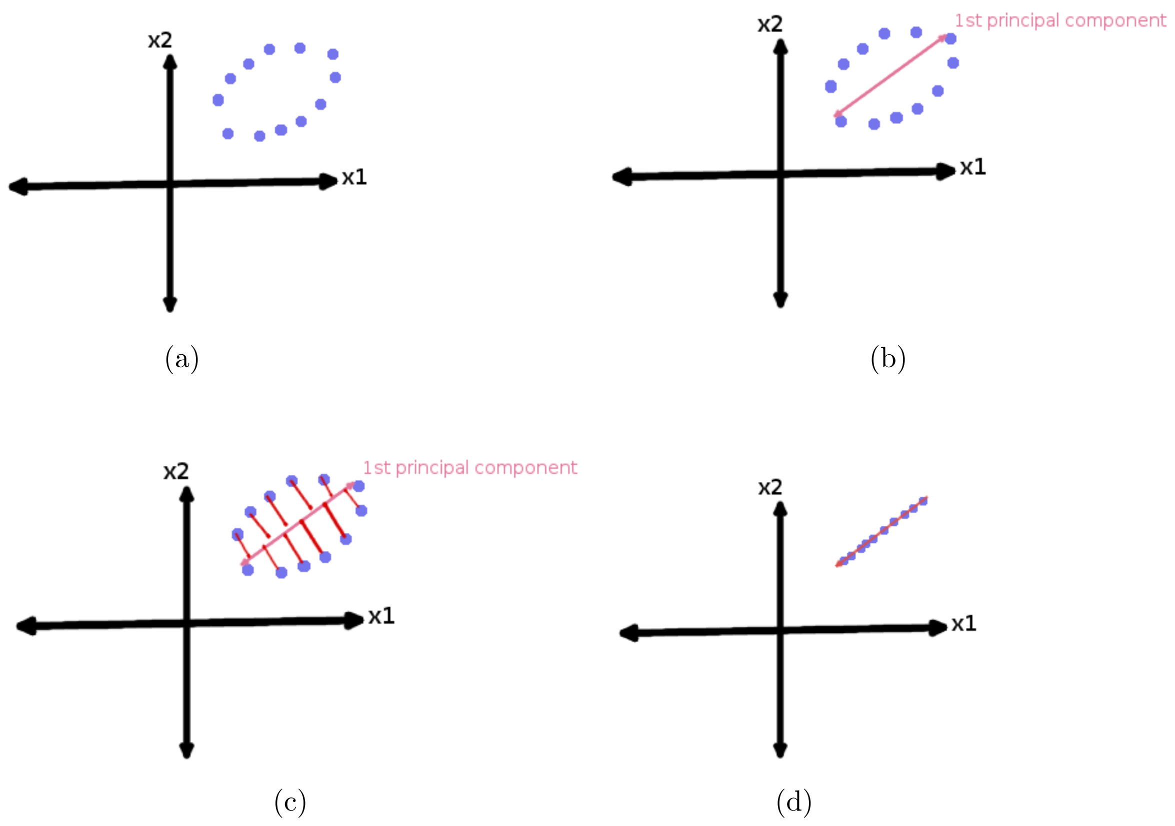


Figure 2.1: 2.1a:Represents 2D data. 2.1b: First principal component. 2.1c:Projecting Data. 2.1d:Data after projection.

The eigenvalues are ordered from largest to smallest so that it gives us the components in order of significance. Here comes the dimensionality reduction part. If we have a dataset with n variables, then we have the corresponding n eigenvalues and eigenvectors. It turns out that the eigenvector corresponding to the highest eigenvalue is the principal component of the dataset and it is our call as to how many eigenvalues we choose to proceed our analysis with. To reduce the dimensions, we choose the first p eigenvalues and ignore the rest. We do lose out some information in the process, but if the eigenvalues are small, then the information lost is negligible.

Mathematically, the principal components are the eigenvectors of the covariance matrix of the original dataset. The covariance matrix is symmetric; the eigenvectors are orthogonal. The principal components (eigenvectors) correspond to the direction (in the original n-dimensional space) with the greatest variance in the data. Each eigenvector has a corresponding eigenvalue. The larger eigenvalue corresponding to the eigenvector(principal component) indicates that that principal component explains a large amount of the variance in the data.

When it comes to applications, PCA is mainly used as a dimensionality reduction technique in domains like facial recognition, computer vision, and image compression. Also, it is used for finding patterns in data of high dimension in the field of bioinformatics, finance, data mining, psychology, etc.

2.1.3 Singular Value Decomposition(SVD)

The singular value decomposition (SVD) is one of the most important matrix decomposition technique of the computational era. Singular value decomposition is a stable and efficient method, and this can be used to split the system into a set of linearly independent components. With each component having their own energy contribution. Numerically stable matrix decomposition can be obtained from SVD method and can be used in various applications. Singular value decomposition is a numerical technique used to diagonalize matrices in numerical analysis. SVD can be seen from different points of view. On the one hand, it can be viewed as a method for transforming correlated variables into a set of uncorrelated ones that reveal the various relationships among the original data items. Also, SVD can be seen as a method which identifies and organizes the dimensions along which data points exhibit the most variation. This leads to one more way of viewing SVD, which is that once we have identified as where the most changes(variation) are, then we can find the best possible approximation of the original data points using very few dimensions. Hence, SVD can be seen as a method for data reduction. The SVD method will provide a foundation for many other techniques including classification methods, the Proper orthogonal decomposition (POD) and the Dynamic mode decomposition (DMD).

The Singular Value Decomposition is a data-driven method in which patterns are discovered purely obtained from the data, without the addition of any expert knowledge or intuition. This technique is helpful in determining the dominant patterns underlying a high-dimensional system. The low-rank approximation for high-dimensional data can be made using SVD. The SVD may be thought of as a numerically stable computation that provides a hierarchical representation of the data regarding a new coordinate system defined by dominant correlations within the data. The SVD has many applications beyond dimensionality reduction of high-dimensional data. It is useful in computing the pseudo-inverse of non-square matrices, Face recognition, providing solutions to underdetermined or overdetermined matrix equations that are either minimum norm solutions or solutions that minimize the sum-squared error. Also, SVD is used in a principled approach to de-noising data sets. The SVD is also essential to characterize the input and output geometry of a linear map between vector spaces.

Definition

Let's consider a large data set X:

$$X = \begin{bmatrix} | & | & \dots & | \\ x_1 & x_2 & \dots & x_m \\ | & | & \dots & | \end{bmatrix} \tag{2.1}$$

Where $x_1, x_2...., x_m$ are the column vectors and may represents measurements from simulations or experiments. For example, In DMD case columns may represent images that have been reshaped into column vectors with the number of elements equal to the number of pixels in the image. The column vectors may also represent the state of a physical system that is evolving in time, such as the fluid velocity at each point in a discretized simulation or each measurement location in a wind-tunnel experiment. The index r is a label indicating the rth distinct set of measurements. Often the state-dimension n is vast, on the order of millions or billions in the case of fluid systems. The columns are usually called snapshots, and m is the number of snapshots in X. For many systems nxm, resulting in a tall-skinny matrix, as opposed to a short-fat matrix when $n << m$.

SVD breaks a nxm matrix A into three matrices U, Σ and V such that

$$A = U\Sigma V^*$$

. The matrix U is a nxr orthogonal matrix whose column vectors are called the left singular vectors of A, V is a rxm orthogonal matrix whose column vectors are the right singular vectors of A, and Σ is a rxr diagonal matrix having the singular values of A ordered decreasingly. Columns of U form an orthogonal basis for the column space of A. Since only the first r concepts can be considered are semantic important (the singular values are high), we can approximate the decomposition as:

$$A = U_r \Sigma_r V_r^*$$

where U_r contains the first r most important concept vectors, Σ_r contains the respective singular values and $\Sigma_r V_r$ contains the pseudo-document vectors represented using the first r concept vectors. In other words, by SVD the original m-dimensional vectors are projected into a vector space of dimension $r(r << m)$. The SVD approximation (so-called rank-r SVD) can be created either by trimming the full-SVD matrices or by usage of a special method designed to perform directly the rank-r SVD.

Theorem (SVD): Let A be an mxn rank-r matrix. Be $\sigma_1, ..., \sigma_y$ eigenvalues of a matrix $\sqrt{AA^T}$. There exist orthogonal matrices $U = (u_1, ...u_r)$ and $V = (v_1, ..., v_r)$, whose column vectors are orthonormal, and diagonal matrix $\Sigma = diag(\sigma_1, ..., \sigma_r)$. The decomposition $A = U\Sigma V^T$ is referred to as a singular value decomposition of matrix A and numbers $\sigma_1, ..., \sigma_r$ are singular values of the matrix A. Columns of $U(orV)$ are referred to as left (or right) singular vectors of matrix A. Now we have a decomposition of the original matrix A. Needless to say, the left and right singular vectors are not sparse. We have at most r nonzero singular numbers, where rank-r is the smaller of the two matrix dimensions. Because the singular values usually decrease quickly, we only need to take k greatest singular values and corresponding singular vector coordinates and create a r-reduced singular decomposition of matrix A.

Definition: Let us have r, $0 < k < r$ and SVD of A.

$$A = U\Sigma V^T = (U_r U_0) \begin{pmatrix} \Sigma_r & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix}$$

$A = U_r \Sigma_r V_r^T$ is referred to as a r-reduced SVD (rank-r SVD).

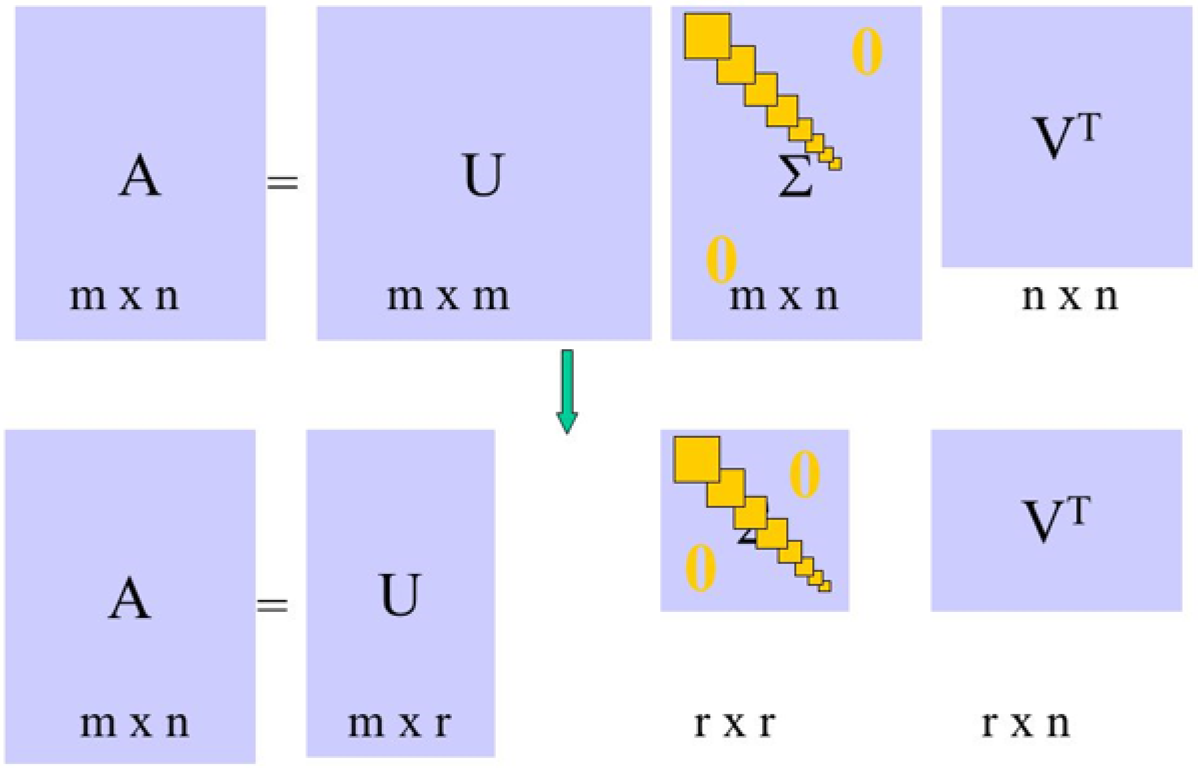


Figure 2.2: Singular Value Decomposition.

Image Source (Shang-Hua, 2015(accessed April 10, 2018))

Truncation and hard thresholding

Truncation is an essential factor to consider when dealing with singular value decomposition. Deciding on how many singular values to retain, i.e., where to truncate, is one of the most critical and contentious decisions when using the SVD. There are many factors, including specifications on the desired rank of the system, the magnitude of noise, and the distribution of the singular values. Often, one truncates the SVD at a rank r that captures a pre-determined amount of the variance or energy in the original data, such as 90% or 99% truncation. This is the commonly used technique for truncation. Other methods involve identifying elbows or knees in the singular value distribution, which may denote the transition from singular values that represent important signal from those that represent noise (Brunton-Kutz-Proctor, 2015(accessed April 10, 2018)). Truncation may be viewed as a hard threshold on singular values, where values greater than a threshold t are kept while remaining singular values are cutoff. Recent work provides an optimal truncation value, or hard threshold, under certain conditions providing a principled approach to obtaining low-rank matrix approximations using the SVD.

2.2 Mathematical Introduction to DMD

The DMD algorithm highlights its connections to the Arnoldi algorithm and Koopman operator theory as initially it was formulated regarding a companion matrix. Here mathematical introduction of an SVD-based DMD algorithm is presented which is more numerically stable, and is now generally accepted as the defining DMD algorithm.

2.2.1 Algorithm

Consider the data obtained from n observable locations at times $k\Delta t$, where the data at single snapshot k is saved into a single column vector x_k and this is done for every time step k.

Assuming that the data being generated by linear dynamics and collecting data from m snapshots in time, we may construct two data matrices X_1 and X_2 as follows:

$$X_1 = \begin{bmatrix} | & | & \cdots & | \\ x_1 & x_2 & \cdots & x_{m-1} \\ | & | & \cdots & | \end{bmatrix} \tag{2.2}$$

$$X_2 = \begin{bmatrix} | & | & \dots & | \\ x_2 & x_3 & \dots & x_m \\ | & | & \dots & | \end{bmatrix} \tag{2.3}$$

Where X_2 is the time-shifted matrix and since we assume that the data is generated from linear dynamics we can write:

$$X_2 = AX_1 \tag{2.4}$$

The matrix A is referred to as the linear operator or Koopman operator and therefore the approximating operator A can be defined as:

$$A = X_2 X_1^{\dagger} \tag{2.5}$$

where $\dagger$ is the Moore-Penrose pseudoinverse. DMD is defined as the eigenvalue analysis of the Matrix A, decomposed into eigenvalues Λ and eigenmodes Φ . Also when we apply DMD algorithm to the data which is generated by nonlinear dynamics, then we assume that there exists an operator A that approximates those dynamics. The DMD eigenvalues Λ and modes Φ is expected to approximate the eigenvectors and eigenvalues of A.

To obtain the approximation of A, it requires the pseudoinverse of X_1 to be solved. An efficient and accurate way to solve the pseudoinverse is by using the singular value decomposition(SVD). SVD of the matrix X_1 is given by the relation:

$$X_1 = U\Sigma V^* \tag{2.6}$$

The matrix U is orthogonal and contains POD modes. And Σ is a diagonal matrix, where the diagonal values represent the contribution of each mode.Also matrix U and V are unitary matrices.

When the Matrix A is large, computing the eigendecomposition of the matrix A may become very expensive. So we solve the eigenvalue problem for the matrix $\widetilde{A}$ which is an approximation to matrix A. The approximation matrix $\widetilde{A}$ can be found using equations 2.5 and 2.6 and by truncating the SVD of matrix X_1 to r singular values.

$$\begin{aligned} AX_1 &= X_2 \\ U^*AX_1 &= U^*X_2 \\ U^*A(U\Sigma V^*) &= U^*X_2 \\ U^*AU\Sigma V^* &= U^*X_2 \\ U^*AU &= U^*X_2V\Sigma^{-1} \\ \widetilde{A} &= U^*X_2V\Sigma^{-1} \end{aligned}$$

In the next section eigendecomposition of $\widetilde{A}$ will be solved. By solving the eigenvalue problem of $\widetilde{A}$, one can approximate the eigenvalues of (unknown) matrix A.

2.2.2 Eigendecomposition

The eigenvalues of the matrix $\widetilde{A}$ represents the inter-snapshot of Propagation Matrix's A dynamics(promotor De Mulder, 2016 (accessed April 10, 2018)). This eigenvalue problem of $\widetilde{A}$ is characterised by relation:

$$\widetilde{A}y_i = \Lambda_i y_i \tag{2.7}$$

$$Y^{-1}\widetilde{A}Y = M \tag{2.8}$$

where $Y = [y_1, y_2.....y_m]$, Λ_i and Φ_i are the eigenvalues and eigenmodes, stored in Eigenvalue Matrix M(diagonal matrix).

Transforming the eigenvalue problem to Matrix $\widetilde{A}$ results in the following relation:

$$\widetilde{A}y_i = (U^* AU)y_i = \Lambda_i y_i$$
$$Y^{-1}\widetilde{A}Y = Y^{-1}(U^* AU)Y = M$$

Or

$$A(Uy_i) = \Lambda_i(Uy_i) \tag{2.9}$$
$$A(UY) = (UY)M \tag{2.10}$$

From the equation 2.9 the eigenvectors of the Matrix A are clearly identifi

ed as $\Phi_i = Uy_i$. Hence, dynamic eigenmodes are constructed by Eigenmode Matrix $UY = \Phi$ and the eigenvalue problem is thus rewritten as:

$$\Phi^{-1}A\Phi = M$$

These constructed eigenvectors $\Phi_i = (UY)_i$ or dynamic modes are expressed in the POD-basis of X_1 and are characterized by their eigenvalues Λ_i. In fact these eigenvectors Φ_i and corresponding eigenvalues Λ_i are an approximated eigenpair of the system matrix A. Hence, the obtained eigenvectors and eigenvalues are only an approximation. Moreover it is stated that the lower eigenvalues are approximated from the right hand side, while the higher eigenvalues are approximated from the left hand side.

2.2.3 Reconstruction

By this step, we have calculated the eigenvalues Λ_i and eigenmodes Φ_i . It is known that the eigenvectors Φ_i represent the spatial distribution and eigenvalues Λ_i represent the dynamical behavior in time, intrinsic to the considered data field. Moreover, it is possible to reconstruct the data field based on the eigenmodes and eigenvalues. More specifically, the reconstruction takes the form of a linear combination of spatial eigenmodes Φ_i, eigenvalues Λ_i and an initial amplitude. The end result of this linear combination is given in equation below:

$$\hat{X}(t) = \Phi exp(\Omega t) z \tag{2.11}$$

Where $\Omega = log(\Lambda)/\Delta t$ is the continuous time eigenvalue, t is time and z is the set of weights to match the first time point measured such that $x_1 = \Phi z$.

The Standard DMD and Exact DMD algorithms are summarized as follows(Tu et al., 2013)

Algorithm1: Standard DMD

1. Arrange the data $[x_1, x_2....., x_m]$ into two matrices

   $X_1 = [x_1, x_2, ..., x_{m-1}]$ $\qquad$ $X_2 = [x_1, x_2, ..., x_m]$

2. Compute the Singular value Decomposition(SVD) of X_1, writing

   $X_1 = U\Sigma V^*$

   where U is nxr, Σ is diagonal and rxr, V is mxr, and r is the truncated value.

3. Define the matrix $\widetilde{A}$ as

   $\widetilde{A} = U^* X_2 V \Sigma^{-1}$

4. Compute eigenvalues and eigenvectors of $\widetilde{A}$

   $\widetilde{A}y = \Lambda y$

5. The DMD mode corresponding to the DMD eigenvalue Λ is then given by

   $\Phi = Uy$

In Algorithm 1 the data is assumed to be a sequential set of data vectors. Whereas in case of Algorithm 2 there is no such restriction, instead we consider data pairs $(x_1, y_1), ...(x_m, y_m)$. In this case, we define data matrices as follows.

$$X = \begin{bmatrix} | & | & \dots & | \\ x_1 & x_2 & \dots & x_m \\ | & | & \dots & | \end{bmatrix} \tag{2.12}$$

$$Y = \begin{bmatrix} | & | & \dots & | \\ y_1 & y_2 & \dots & y_m \\ | & | & \dots & | \end{bmatrix} \tag{2.13}$$

To relate this method to the standard DMD procedure, we may assume that:

$$y_k = Ax_k$$

For the data set arranged as data pairs A is given by,

$$A = YX^{\dagger}$$

Where $X^{\dagger}$ is the pseudoinverse of X. The dynamic mode decomposition of the data pair (X, Y) is given by the eigendecomposition of A. That is, the DMD modes and eigenvalues are the eigenvectors and eigenvalues of the matrix A.

Algorithm2: Exact DMD

1. Arrange the data pairs $[(x_1, y_1), ...(x_m, y_m)]$ into matrices X and Y , as

   $X = [(x_1....x_m)]$ $\qquad$ $Y = [y_1....y_m]$

2. Compute the Singular value Decomposition(SVD) of X_1, writing

   $X_1 = U\Sigma V^*$

   where U is nxr, Σ is diagonal and rxr, V is mxr, and r is the truncated value.

3. Define the matrix $\widetilde{A}$ as

   $\widetilde{A} = U^* X_2 V \Sigma^{-1}$

4. Compute eigenvalues and eigenvectors of $\widetilde{A}$

   $\widetilde{A}y = \Lambda y$

5. The DMD mode corresponding to Λ is then given by

   $\Phi = (1/Y)V\Sigma^{-1}y$

Algorithm 2 is almost identical to Algorithm 1. The only difference lies in the way that the DMD modes are defined. The eigenvectors with $\Lambda = 0$ can also be found using Algorithm2, but DMD modes corresponding to zero eigenvalues are usually not of interest since they do not play a role in the dynamics.

Chapter 3

Testing Examples for DMD

3.1 Interpretation of DMD modes and Eigenvalues

During the analysis of DMD, the fundamental part is to understand the DMD modes and Eigenvalues. In this section, the interpretation of DMD results is presented.

3.1.1 Interpretation of DMD Modes Φ

The DMD modes or Eigenvectors Φ represent the spatial distributions proper to the system or data field. These spatial modes are separated patterns on which the data is decomposed. This is important to estimate the influence range, the spatial distribution, or magnitude of the specific mode. The DMD modes Φ are also scaled by an amplitude z which can be seen in 2.11, which is retrieved from the decomposition. This amplitude is important to determine which mode is more dominant, or in other words, which mode is more represented in the decomposition. The amplitude gives quantitative information about the presence of a specific mode.

3.1.2 Interpretation of DMD Eigenvalues Λ

The DMD Eigenvalues Λ_i are the discrete eigenvalues, and they represent the temporal, dynamical behavior of their corresponding DMD mode Φ_i. To interpret the temporal behavior, the eigenvalues Λ are transformed to an equivalent continuous-time eigenvalues Ω

or also called exponential eigenvalues. The spectrum plot of the eigenvalues Λ is realized by a complex valued Cartesian coordinate system, which is called the complex plane. The DMD eigenvalues Λ are in the vicinity of the unitary circle. The unitary circle itself represents stable or stationary flow structures. DMD modes or Eigenmodes corresponding to eigenvalues inside the circle are decaying modes, Eigenmodes corresponding to eigenvalues outside the circle is growing modes, and Eigenmodes corresponding to eigenvalues on the circle are stable modes. The interpretation of continuous-time eigenvalues Ω_i is discussed in the next section.

3.1.3 Interpretation of continuous-time Eigenvalues Ω

As mentioned, the DMD eigenvalues Λ are transformed to continuous-time eigenvalues Ω. This transformation is essential for the interpretation of the dynamical properties. The continuous-time eigenvalues Ω can be easily interpreted by their corresponding real and imaginary part respectively. First, the DMD eigenvalue Λ is converted to continuous-time eigenvalue Ω using the equation 3.1

$$\Omega = log(\Lambda)/\Delta t \tag{3.1}$$

This results in a complex value $\Omega = \beta + j\omega$.

$$e^{\Omega t} = e^{(\beta + j\omega)t}$$

$$e^{\Omega t} = e^{\beta t}(cos(\omega t) + jsin(\omega t))$$

$$e^{\Omega t} = e^{\beta t}cos(\omega t) + je^{\beta t}sin(\omega t) \tag{3.2}$$

- If $\beta < 0$, then it represents that the DMD mode is decaying.
- If $\beta > 0$, then it represents that the DMD mode is growing.
- If $\beta = 0$, then it represents that the DMD mode is oscillating.

A sample of the spectrum plot for discrete Eigenvalues Λ and continuous-time Eigenvalues Ω is as shown in the Figure:3.1.

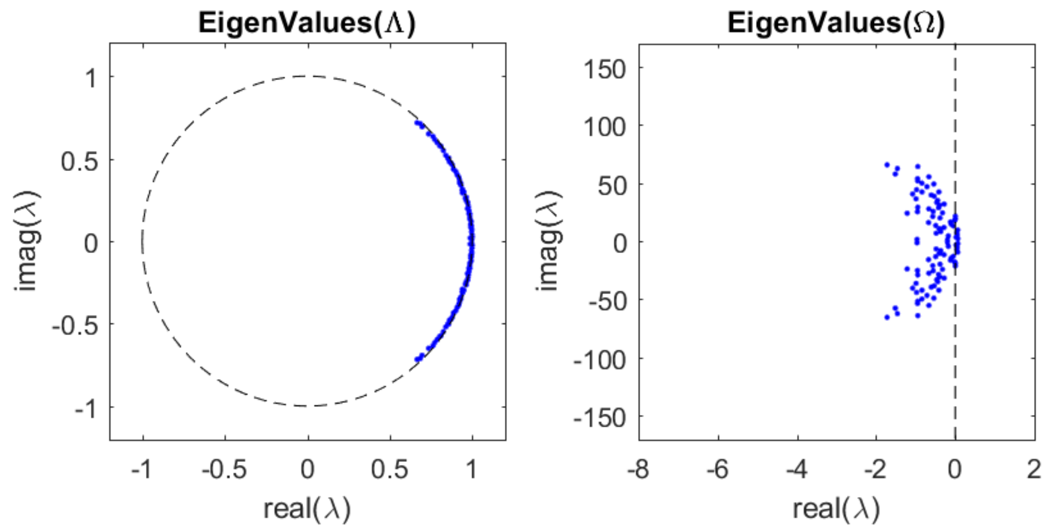


Figure 3.1: Spectrum plot of Eigenvalue Λ and Eigenvalue Ω

For better interpretation of the system dynamics, the time-evolution of eigenvalues Ω are plotted. We can plot the eigenvalues Ω either by using the real or the imaginary part of 3.2. A sample of the time-evolution is show in the Figure:3.2.

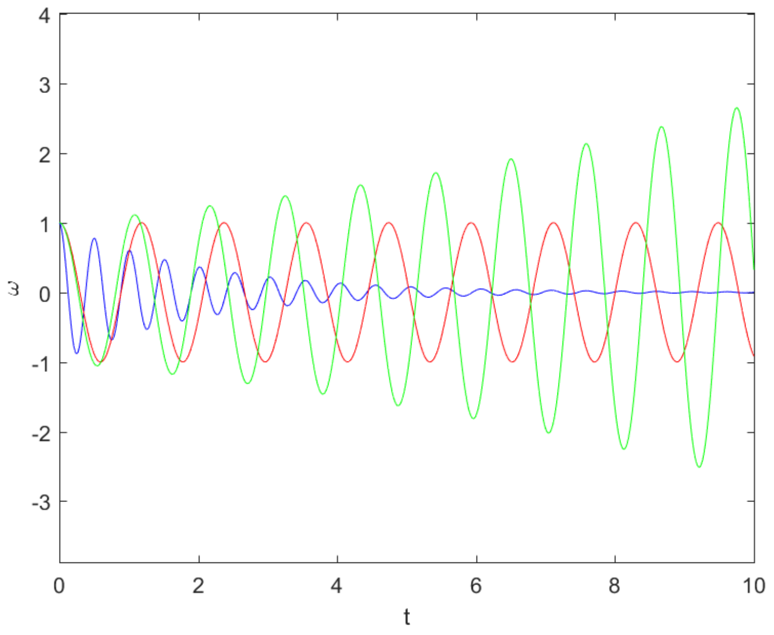


Figure 3.2: Time Evolution of Eigenvalues Ω

The time-evolution plot shown in the Figure 3.2 is plotted for three different Eigenvalues Ω. Where,

$$\Omega_1 = -0.5004 + j12.4150$$

$$\Omega_2 = 0.0000 - j5.3000$$

$$\Omega_2 = 0.1000 - j5.8000$$

Every DMD mode will have its corresponding Eigenvalue(time dynamics). The DMD mode corresponding to Ω_1 will be decaying over time. This can be seen by the blue waveform in the Figure 3.2. The DMD mode corresponding to Ω_2 will be oscillating over time. This can be seen by the red waveform in the Figure 3.2. The DMD mode corresponding to Ω_3 will be growing over time. This can be seen by the green waveform in the Figure 3.2.

The temporal mode stability for discrete and continuous-time Eigenvalues is summarized in the Table 3.1. Also, the Figure 3.1 is considered in summarizing this table.

Discrete-time Eigenvalue Λ	Continuous-time Eigenvalue Ω	Mode Evolution	Spectrum plot
$\|\Lambda\| < 0$	$\Re(\Omega) < 0$	Decaying	Λ: Inside unit circle,Ω: LHS of straight line
$\|\Lambda\| = 0$	$\Re(\Omega) = 0$	Oscillating	Λ: On the unit circle,Ω:On the straight line
$\|\Lambda\| > 0$	$\Re(\Omega) > 0$	Growing	Λ: Outside unit circle,Ω: RHS of straight line

Table 3.1: Temporal Mode Stability

3.2 CASE 1: DMD on Simple Data

DMD is an equation free modeling technique. The data required for DMD analysis in the real world does not need to know the governing equations for the data. Here to generate the synthetic data few equations were used. The primary purpose of testing this data was to identify the underlying patterns in the data and to predict the dynamics of these patterns over time. The coding was done in Matlab for all the cases and the code from (Nthan Kutz Research Group, 2015(accessed April 10, 2018)) was used as a reference.

Generating Synthetic Data

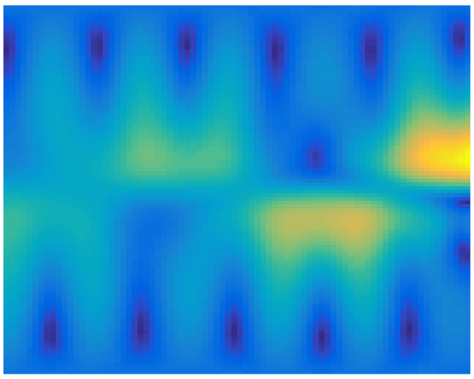
The following three equations were used in generating testing data for case 1. These three equations were added to obtain the final data matrix. This data serves as the input for the case1 analysis.

$$Equation_1 = (20 - 0.2 * X_m^2) * \exp(2.3j * T_m)$$

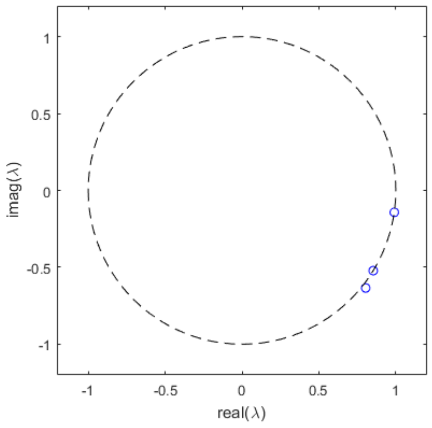
$$Equation_2 = X_m * exp(0.6j + T_m)$$

$$Equation_3 = 5 * 1/cosh(X_m/2) * tanh(X_m/2) * 2exp((0.1 + 2.8j) * T_m)$$

Where X_m and T_m are the space and time coordinates.



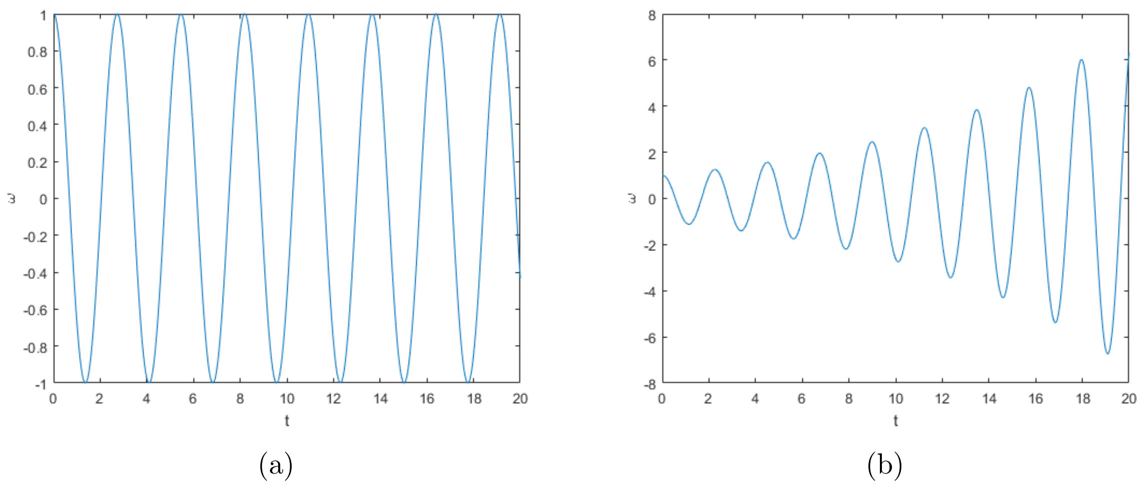
(a)



(b)

Figure 3.3: 3.3a Visualizing Data, 3.3b Eigenvalue plot

In the data provided for DMD, there were three dominant modes. After applying DMD to this dataset, DMD was successful in determining those three dominant modes. This can be seen in the eigenvalue plot shown in the figure 3.3b. The eigenvalues Λ_i represent the temporal, dynamical behavior of their corresponding eigenmodes Φ_i. In order to interpret the temporal behavior, the discrete eigenvalues Λ_i are transformed to an equivalent continuous-time eigenvalue Ω_i. The temporal evolution corresponding to each mode is shown below.



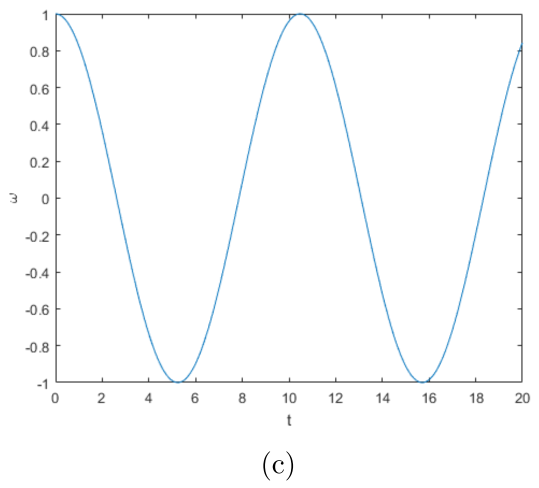


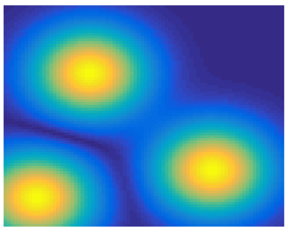
Figure 3.4: 3.4a:Temporal Evolution of Mode1, 3.4b:Temporal Evolution of Mode2, 3.4c: Temporal Evolution of Mode3.

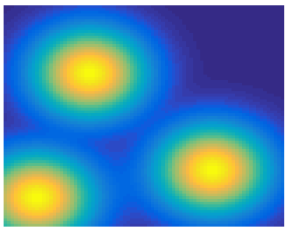
It can be seen clearly from the figure 3.4 that two out of three modes are oscillating and one mode is growing over time. Thus DMD was successful in determining the underlying patterns(modes) and their corresponding temporal evolution in the data.

3.3 CASE 2: DMD on data set with modes oscillating at different frequency

In case2 analysis, DMD was fed with the data set having modes oscillating at a different frequency. In this particular case, three modes oscillating at 0.5Hz, 2.5Hz and 5Hz were generated. The purpose of testing this case was to check whether DMD can capture these modes oscillating at the different frequency but having the same amplitude. After applying DMD to this data set, DMD was able to recover those three modes with the frequencies approximately equal to the original frequencies.

From the below figures, we can see that the reconstructed snapshots are similar to the original data snapshots.


(a)


(b)

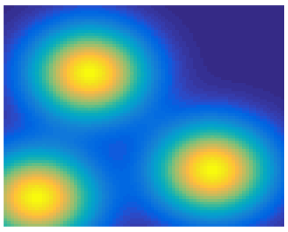
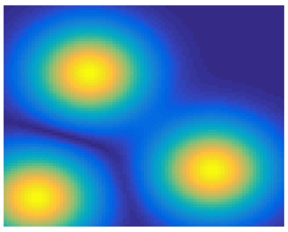
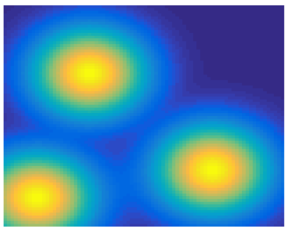

(c)

Figure 3.5: 3.5a: Data from 20th snapshot, 3.5b: Data from 250th snapshot, 3.5c: Data from 800th snapshot


(a)


(b)

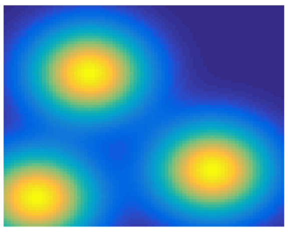
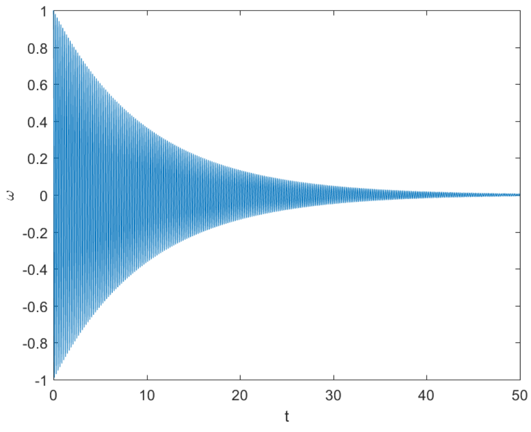

(c)

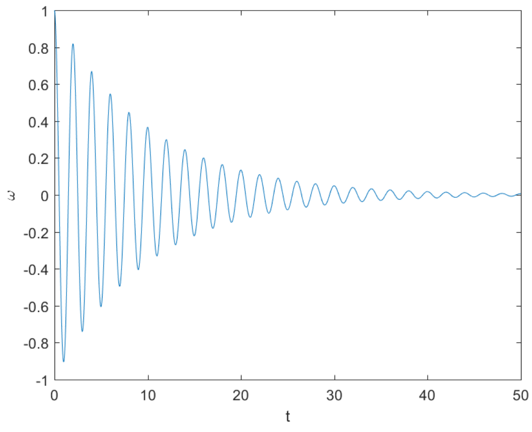
Figure 3.6: 3.6a: Reconstructed data from 20th snapshot, 3.6b: Reconstructed data from 250th snapshot, 3.6c: Reconstructed data from 800th snapshot

The continuous-time eigenvalues obtained for this particular case are: $\Omega_1 = -0.1004+$

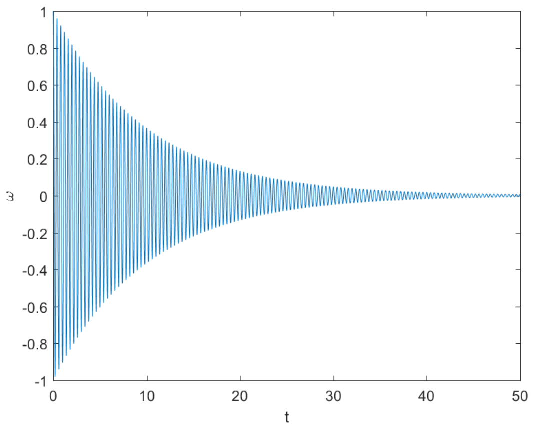
31.4150i, $\Omega_2 = -0.1004+3.1427i$, $\Omega_3 = -0.1005+15.7078i$ Therefore giving the frequency values of 4.9999Hz, 0.5002Hz, 2.5000Hz which are same as the original frequencies.The temporal evolution of all three modes are as shown in the below figures and all three modes are decaying over time.



(a)



(b)



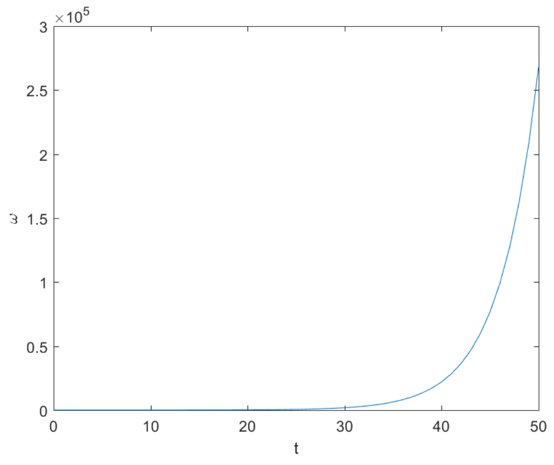
(c)

Figure 3.7: 3.7a: Temporal Evolution of Mode1, 3.7b: Temporal Evolution of Mode2, 3.7c:Temporal Evolution of Mode3.

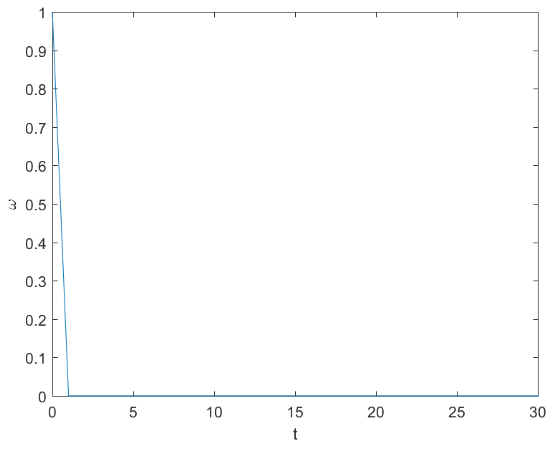
Thus in case 2 analysis, DMD was successful in capturing the three modes oscillating at different frequencies, and the reconstructed data almost approximated the original data. Also, time dynamics for these modes is shown in the figure 3.7.

3.4 CASE 3: DMD on data with Circle growing and decaying

In the case3 analysis, DMD is applied to two different datasets. The first dataset consists of a single mode which will be strictly growing over time. Also, the second dataset consists of single mode which is decaying over time. In both the cases, there was no frequency content. The data visualization for the circle growing and decaying is shown in the figure. In both the cases its quite hard to identify the pattern visually, as they appear to be similar.



(a)



(b)

Figure 3.8: 3.8a: Temporal Evolution of Growing circle case, 3.8b: Temporal Evolution of decaying circle case

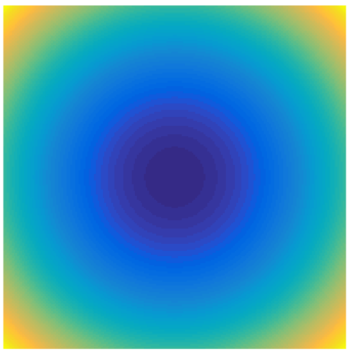


Figure 3.9: Visualizing data for growing and decaying circles

When DMD was applied to both the data sets, DMD was able to identify a single dominant mode in each case. The continuous-time eigenvalue in both the cases were real numbers. After plotting their respective time behavior; it showed the underlying patterns which were growing and decaying over time.

3.5 CASE 4: DMD on Linear-Motions

If we observe, we will find that everything in the universe is in motion. However, different objects move differently. Some objects move along a straight line, some move in a curved path, and some move in some other way. In the translational motion, a body moves along a line without any rotation. The line may be straight or curved. Translatory motion is further divided into Rectilinear motion, curvilinear motion and random motion.

A simple dataset with Rectilinear and Curvilinear motion of an object(solid circle) was generated. DMD was applied to this dataset to figure out whether DMD can be used for classification of different motion patterns.

3.5.1 Rectilinear Motion

In case of rectilinear motion, the object was made to move along the vertical, horizontal and diagonal path. In each case, the object was made to move at different speeds from snapshot to snapshot. Later DMD was applied to each case to observe the reconstruction of the dataset.

Generating synthetic Data

When generating the dataset for horizontal movement of the object(circle), the centerY was kept constant varying the centerX from snapshot to snapshot. Also to get the object move at different speed(N), the value of N was chosen for each case. The equation that was used to get centerX for each snapshot is shown below:

$$centerX = (N * SnapshotNumber)/imageSizeX$$

Next, the dataset for vertical movement of the object(circle) was generated by keeping the centerX constant and varying the centerY from one snapshot to other snapshot. Even in this case to get the object move at different speed(N), the value of N was chosen for each case. The equation that was used to get centerY for each snapshot is shown below:

$$centerY = (N * SnapshotNumber)/imageSizeY$$

Finally for generating the dataset for diagonal movement of the object both centerX and centerY was varied from snapshot to snapshot. The equation that was used to get centerX and centerY for each snapshot is shown below:

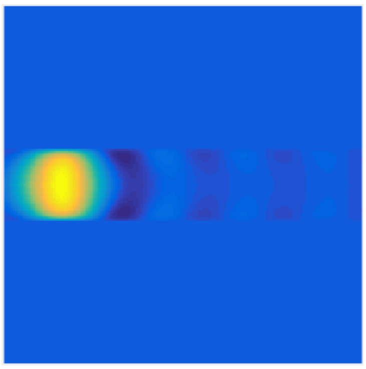
$$centerX = (N * SnapshotNumber)/imageSizeX$$

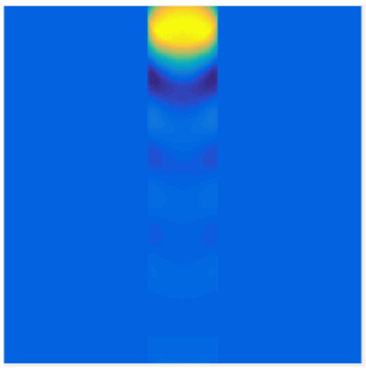
$$centerY = (N * SnapshotNumber)/imageSizeY$$

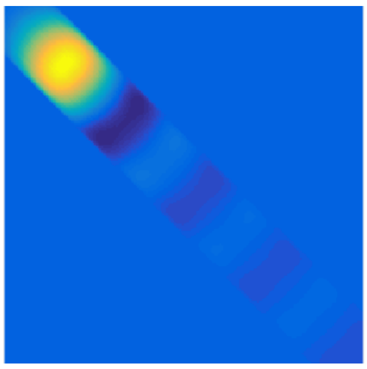
In the figure 3.10, the first column shows the reconstructed data for the object moving horizontally from slow (3.10a)to fast(3.10j) pace. The second column represents the reconstructed data for the object moving vertically from slow(3.10b) to fast(3.10k) pace. The third colum represents the reconstructed data for the diagonal case. For all these cases the SVD value was truncated to $r = 10$.

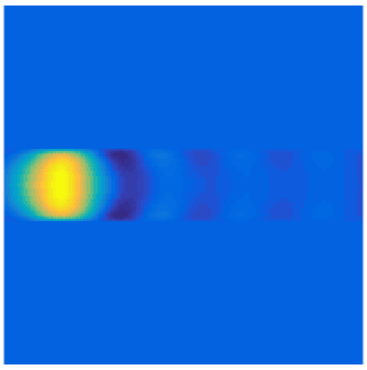
Also, when we compare the results in the first row to the last row in the figure 3.10 we can see that the reconstructed data improves to approximate the original data for all the three cases. Thus DMD will require the larger number of modes to reconstruct the original data at lower speeds.

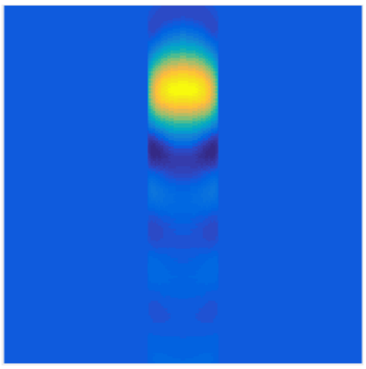
In the below figures the reconstructed data at different speeds for all the three cases with SVD truncation $r = 10$ is shown.

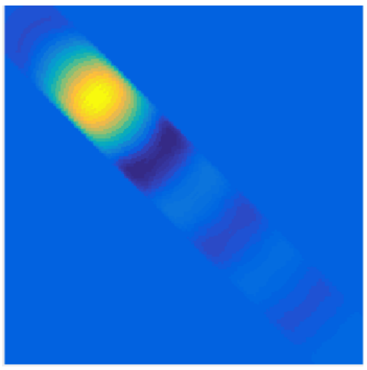

(a)

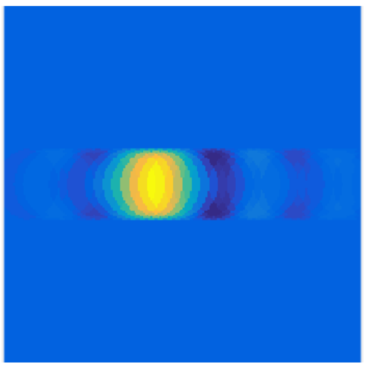

(b)

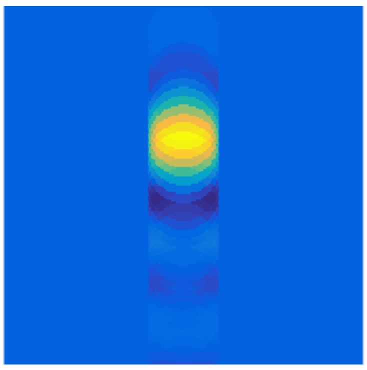

(c)

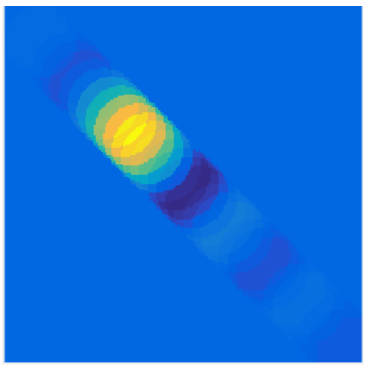

(d)

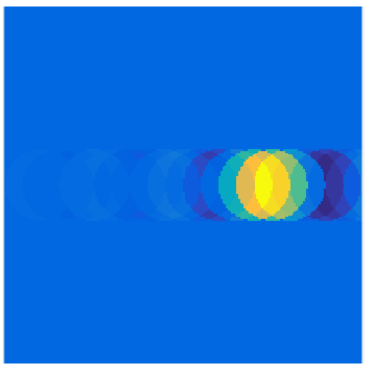

(e)

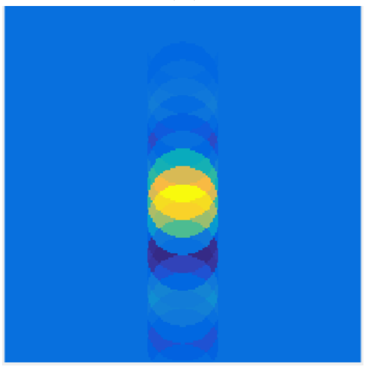

(f)


(g)


(h)


(i)


(j)


(k)

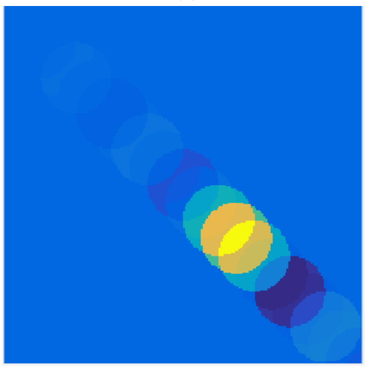

(l)

Figure 3.10: 3.10a,3.10d,3.10g,3.10j: Reconstructed data for horizontal case from slow to fast pace. 3.10b,3.10e,3.10h,3.10k:Reconstructed data for vertical case from slow to fast pace. 3.10c,3.10f,3.10i,3.10l:Reconstructed data for diagonal case from slow to fast pace.

The eigenvalue plot at different speeds for the case of rectilinear motion is as shown below.

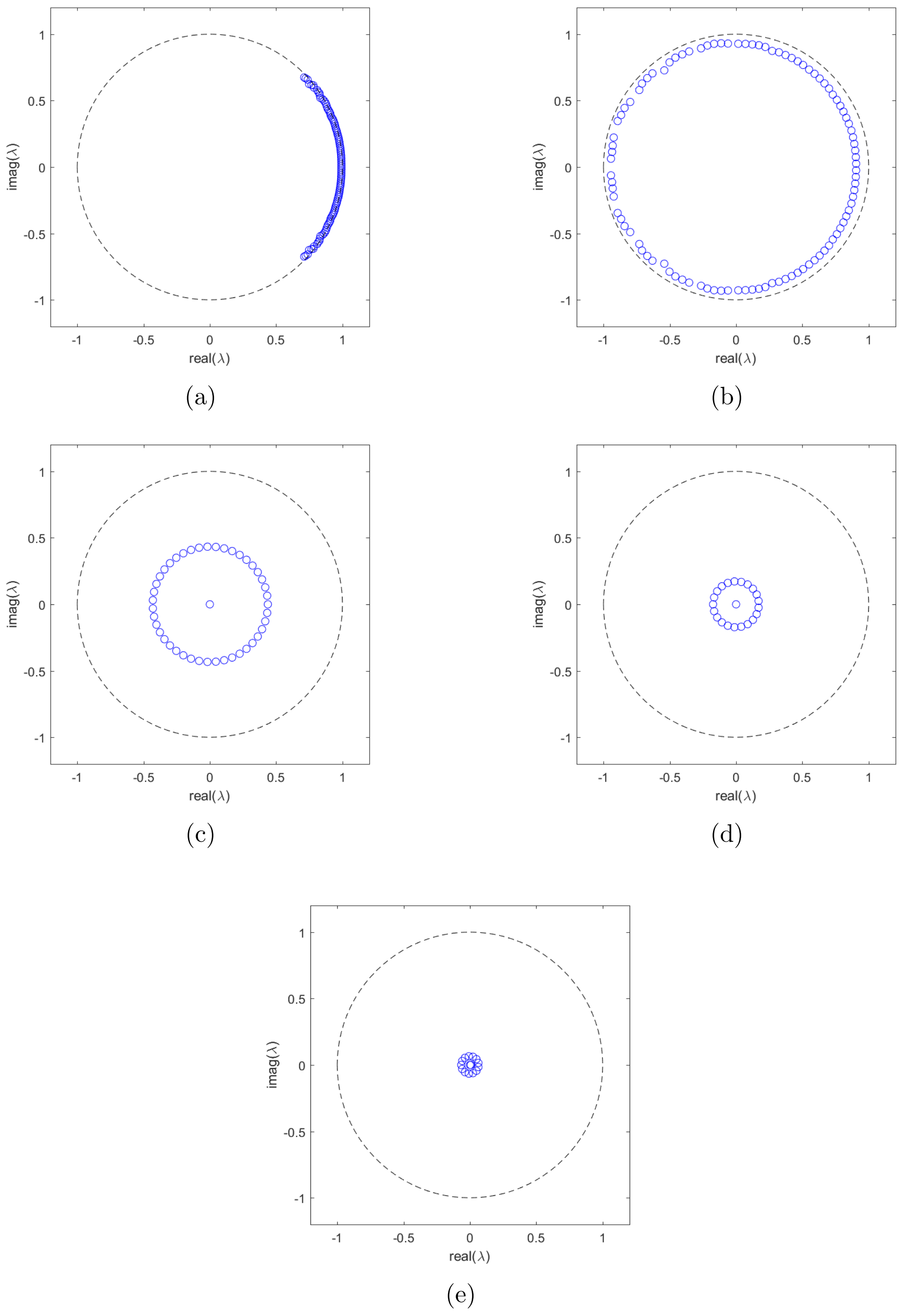


Figure 3.11: 3.11a to 3.11e: EigenValue plot for rectilinear motion from slow to fast motion

3.5.2 Curvilinear Motion

In case of curvilinear motion, the object was made to traverse along the circular path. In each case, the object was made to move at different speeds from snapshot to snapshot. Later DMD was applied to each case to observe the reconstruction of the dataset.

Generating synthetic data When generating the dataset for the curvilinear motion, the centerX and centerY values were obtained for each snapshot using the following set of equations. Also to get the object move at different speed(N), the value of N was chosen for each case.

$$centerX = ImageCenter + (rotatingradius * cos(SnapshotNumber * 2 * pi/N))$$

$$centerY = ImageCenter + (rotatingradius * sin(SnapshotNumber * 2 * pi/N))$$

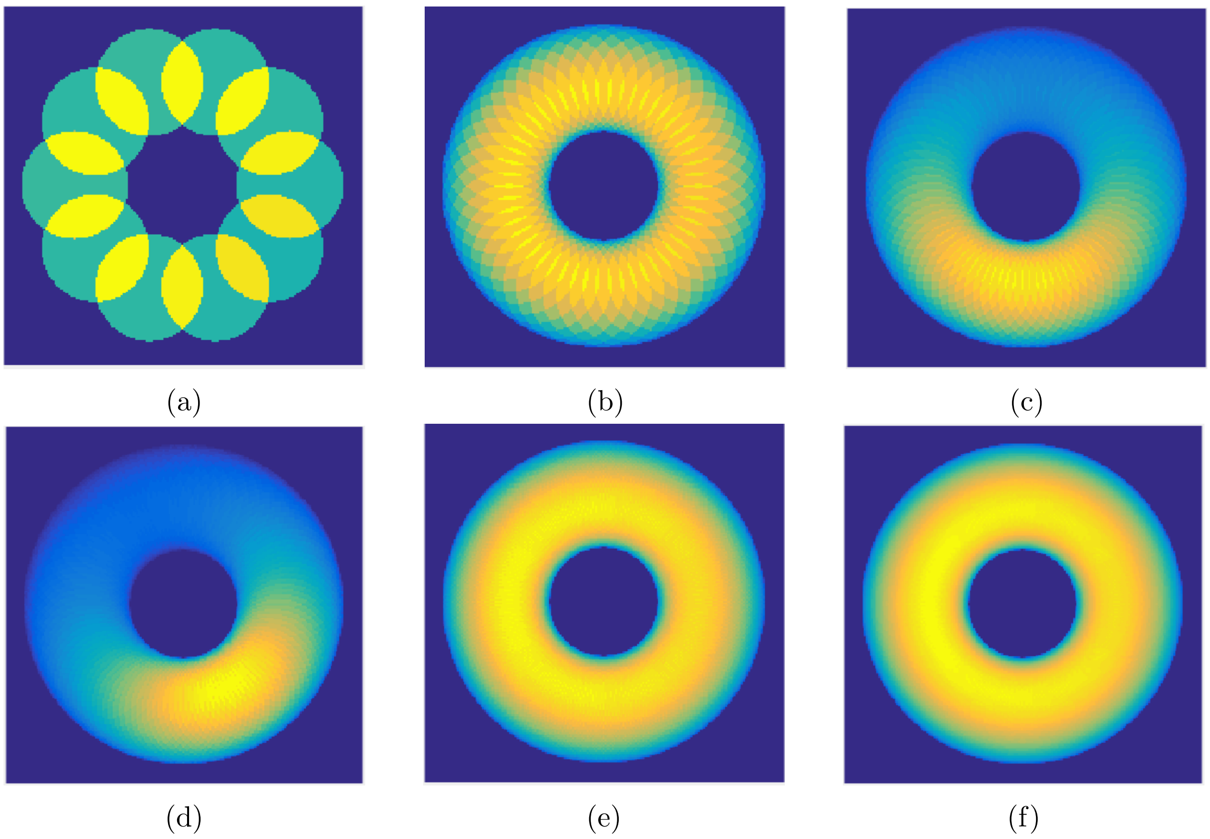


Figure 3.12: 3.12a to 3.12f: Reconstructed data for curvilinear case from fast to slow pace with svd truncation $r = 1$

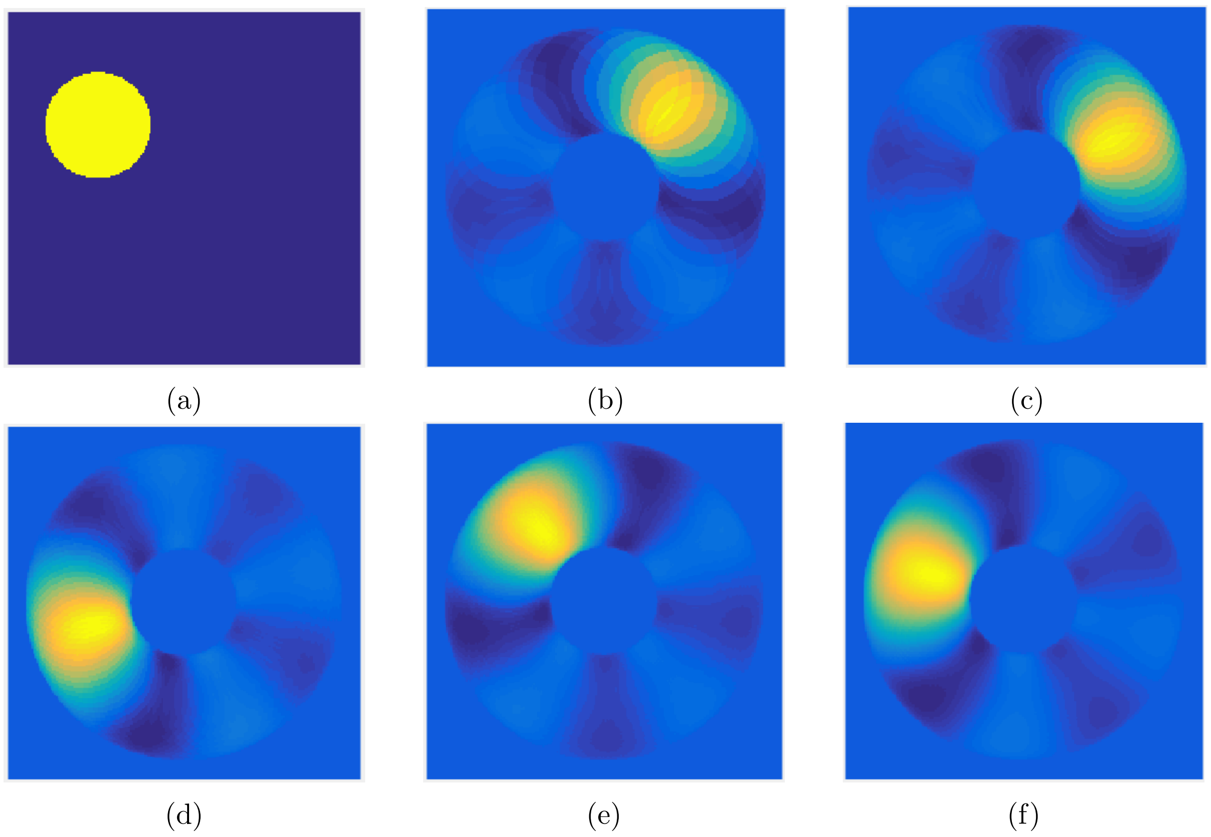


Figure 3.13: 3.13a to 3.13f: Reconstructed data for curvilinear motion with SVD truncation $r = 10$

When we compare the figures 3.12 and 3.13 we can see that as we increase the SVD truncation values from $r = 1$ to $r = 10$, the reconstructed data from DMD improves greatly. Also, when we compare figures 3.12a , 3.12f and 3.13a, 3.13f we can see that at relatively faster speed, DMD reconstructed data approximated the original data. Whereas at slower speed DMD will require $r >>> 10$ to approximate the original data.

The Eigenvalue plot from the figure 3.11 and 3.14 reveals different patterns at faster and slower speeds. At faster speed eigenvalues are spread out, and at the slower speed, the eigenvalues are arranged compactly. Further investigation of these patterns will be made in the future for classification purpose.

The eigenvalue plot for the curvilinear motion at different speeds is as shown below.

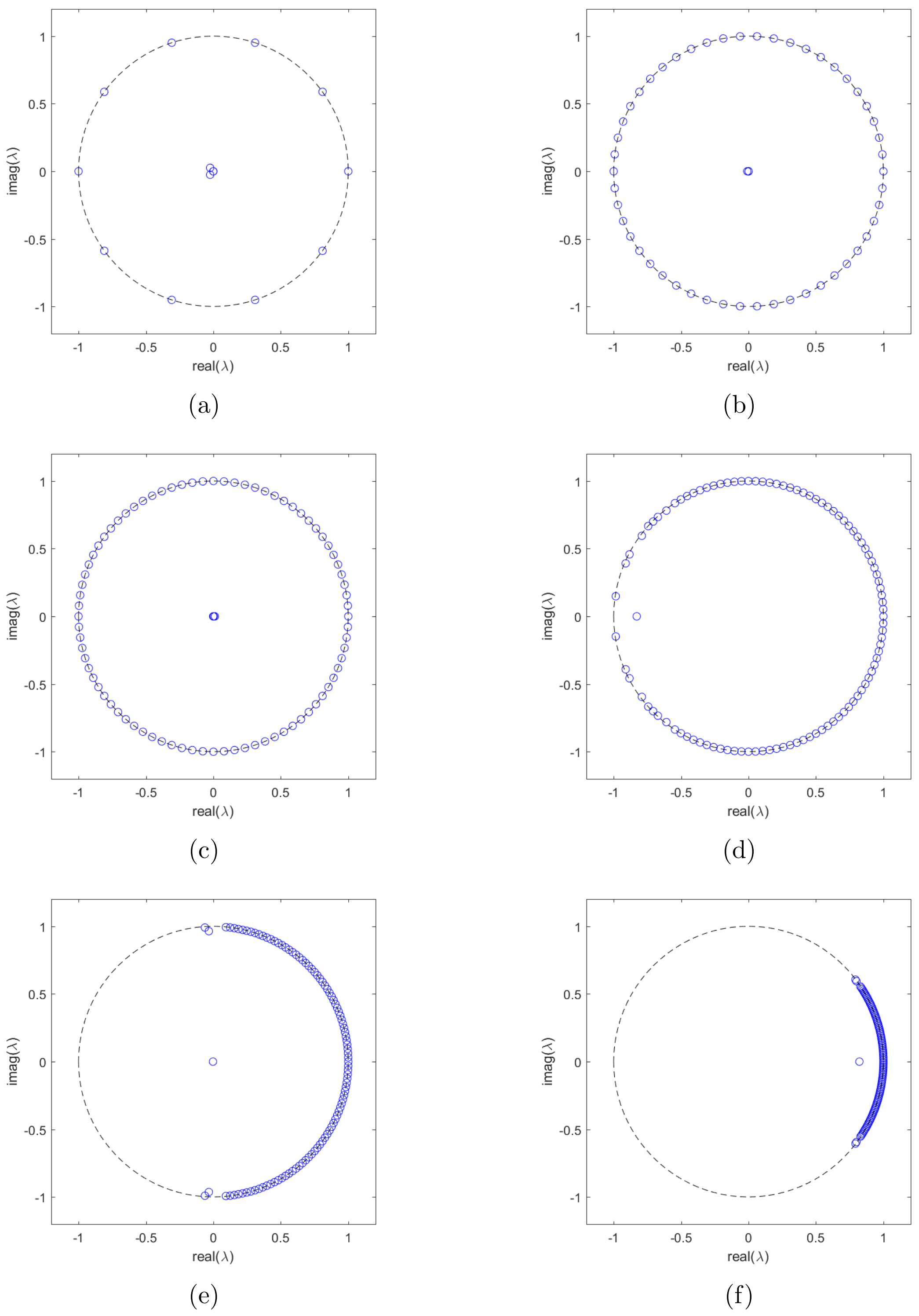


Figure 3.14: 3.14a to 3.14f: EigenValue Plot for Curvilinear motion from fast to slow pace.

Conclusion for CASE4 Analysis

The following conclusion is made while analyzing the DMD on the rectilinear and curvilinear motion. The datasets for both the cases on which DMD was tested was a simple one consisting of a single mode translating along the spatial domain as the system evolves. Although one would think that the DMD would handle this cleanly, the opposite happens. Instead of picking up a single, well-defined singular value, the SVD picks up many. Also, as we tested DMD at different speeds, we could see that when the motion of the object was very slow, SVD picks up almost all the values. Which can be seen in the eigenvalue plot of both the rectilinear and curvilinear motion. But when the motion of the object starts improving the speed, the number of SVD values picked gets lower and lower. It turns out that when the motion is slow, almost all the DMD modes are needed to correctly approximate the system.

3.6 CASE 5: DMD on synthetic data that mimics Human Heart Beat

In the case 5 analysis, the idea was to test DMD whether it can be used for classification. DMD was applied to a synthetic data that mimics human heartbeat at different rates.

Generating Data

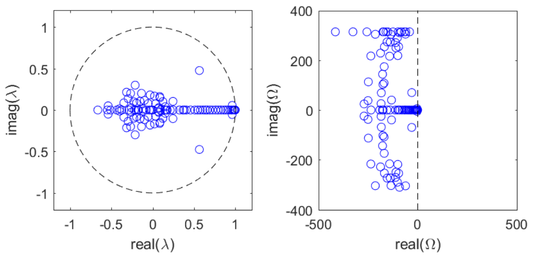
When generating this dataset, a binary circle was created which decreases and increases over time(somewhat mimic human heartbeat). The following equations for radius was used to achieve this:

$$radius = 0.1 + 0.05 * exp(+0.1 * t) * cos(2 * pi * t) \tag{3.3}$$

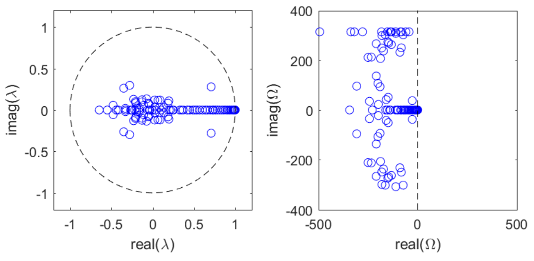
$$radius = 0.1 + 0.05 * exp(-0.1 * t) * cos(2 * pi * t) \tag{3.4}$$

$$t = Constant * SnapshotNumber * SmaplingTime(dt) \tag{3.5}$$

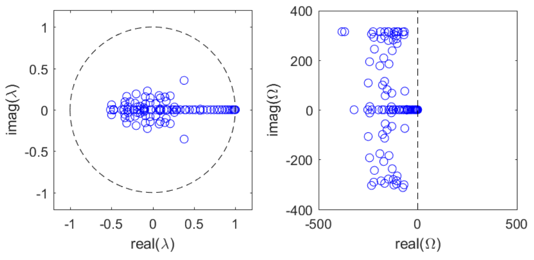
In the equation (3.5) the value of constant term was changed so that when t is used in the radius equation, it produces different patterns. Meaning pattern mimics human heartbeat at slow to fast rates. When the radius with the positive exponential value was used, circle mimics human heartbeat and also it increases its size over time.



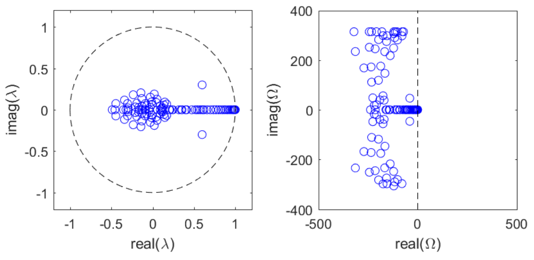
(a)



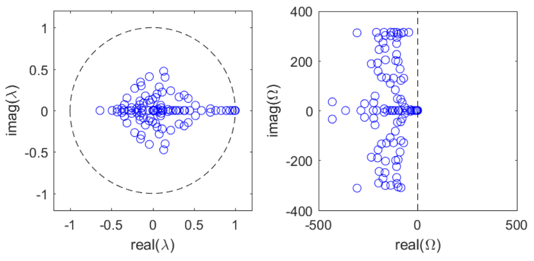
(b)



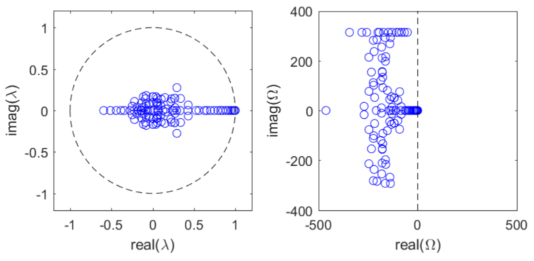
(c)



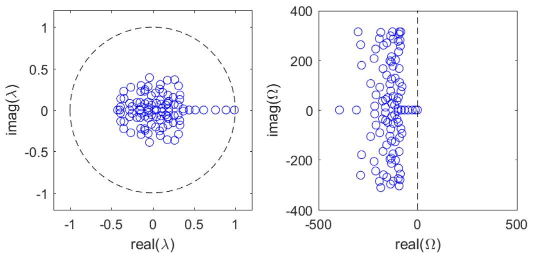
(d)



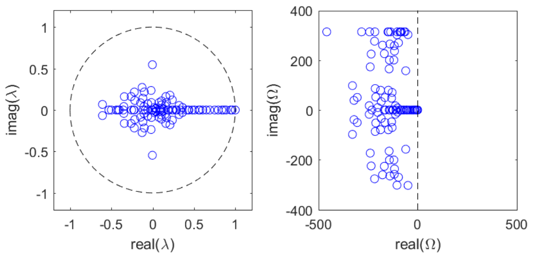
(e)



(f)



(g)



(h)

Figure 3.15: 3.15a, 3.15c,3.15e,3.15g: EigenValue plot for from slow to fast pace with increasing size of the circle in the data. 3.15b,3.15d,3.15f,3.15h: EigenValue plot for from slow to fast pace with decreasing size of the circle in the data.

In contrast, when the radius with the negative exponential value was used, circle mimics human heartbeat and also it decreases its size over time. This was created just to obtain a rough pattern to analyze normal and abnormal heart rates.

The Eigenvalue spectrum for both discrete and continuous-time Eigenvalues are plotted for different patterns is as shown in the figures 3.15. From the Spectrum plot 3.15, it is quite clear that there are slightly different patterns of arrangements of eigenvalues. We can see the cluster of eigenvalues inside the unitary circle representing a decaying dynamics. Well in real data it wasn't an entirely decaying pattern, so DMD fails to capture the time dynamics of this data and also it fails to reconstruct the original data. Further investigation on this case will be made in the future work to see whether we can use these clustered eigenvalues to obtain unique descriptors to classify different motion patterns.

3.7 CASE 6: DMD for Transient time behavior

In the case6 analysis, DMD was applied to the data set having a single transient time dynamics. The data displays gaussian popping in and out of existence. The following equation was used in Matlab to generate this dataset.

$$D = exp(-(((Xm)/4)^2)) * exp(-(((Tm-5)/2)^2)) \tag{3.6}$$

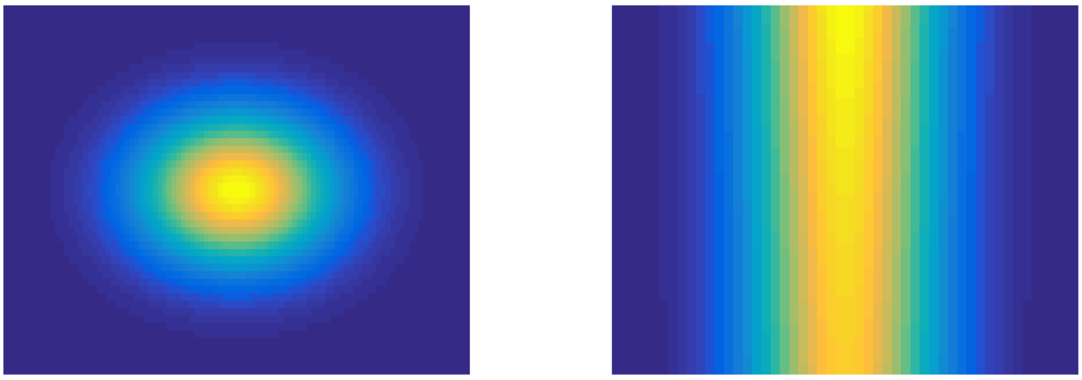

(a) (b)

Figure 3.16: Data visualization and reconstruction for transient behavior.

The spectrum plot for discrete and continuous time eigenvalues is as shown in the figure 3.17.

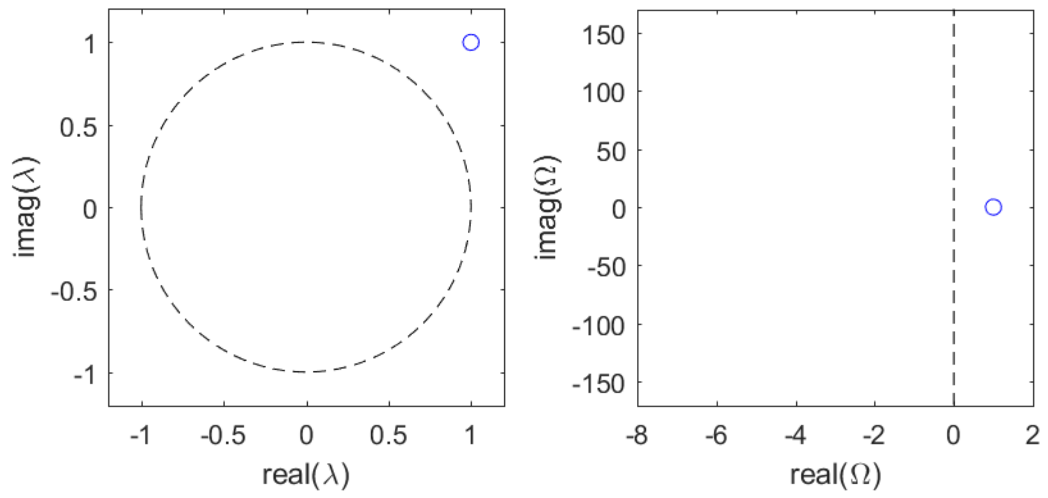


Figure 3.17: Eigenvalue plot for Transient behavior.

From the results, we can say that DMD it was unable to decompose the data accurately. DMD was successful in identifying the dominant mode, but it failed to predict the time-behavior of the system. From the Eigenvalue plot, we can see that the mode has growing time dynamics which is not true for this data. In this case, ideal decomposition of the data could consist of the superposition of single mode with various eigenvalues. DMD was not successful in providing the same DMD mode various times with different eigenvalues with the single application of SVD (Taylor, 2016 (accessed April 10, 2018)).

This drawback of DMD maybe alleviated by its extension called Multi-Resolution DMD(mrDMD). The multi-resolution DMD (mrDMD) attempts to solve the transient time behavior problem using a recursive application of the DMD.

Chapter 4

Conclusion

In this work, a powerful decomposition technique known as Dynamic mode decomposition was presented. The concept of dimensionality reduction is advantageous to build models based on dynamics of the data if there exists a subspace or low-dimensional structure in the high-dimensional data. DMD plays a significant role in dimensionality reduction provided there exist a subspace in the big data. Therefore DMD can reconstruct the original data with fewer dimensions. When compared to other modal decomposition techniques, DMD not only provides modes but also helps in predicting the system dynamics(how modes evolve). Also, it's an equation-free technique which can be applied to complex nonlinear systems. DMD has certain disadvantages as discussed in example cases, despite its limitation DMD is a powerful tool in analyzing and predicting dynamical systems. As it finds application in various domains such as computer vision, neuroscience, biology, finance, etc, it would be great for the data scientists from different backgrounds to have good knowledge of DMD to explore its true potential.

References

Brunton, B. W., Johnson, L. A., Ojemann, J. G., & Kutz, J. N. (2016). Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of neuroscience methods*, *258*, 1–15.

Brunton-Kutz-Proctor. (2015(accessed April 10, 2018)). Singular value decomposition [Computer software manual]. Retrieved from `http://faculty.washington.edu/sbrunton/me565/pdf/CHAPTER1.pdf`

Grosek, J., & Kutz, J. N. (2014). Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*.

Hamilton, L. D. (2014 (accessed April 10, 2018)). Introduction to principal component analysis [Computer software manual]. Retrieved from `http://www.lauradhamilton.com/introduction-to-principal-component-analysis-pca`

Mezić, I. (2005). Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, *41* (1-3), 309–325.

Nthan Kutz Research Group, g. (2015(accessed April 10, 2018)). Dynamic mode decomposition [Computer software manual]. Retrieved from `http://faculty.washington.edu/kutz/page1/page13/`

Proctor, J. L., & Eckhoff, P. A. (2015). Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International health*, *7*(2), 139–145.

promotor De Mulder, T. (2016 (accessed April 10, 2018)). Dynamic mode decomposition in hydrodynamics: Influence of noise and amp missing data [Computer software manual]. Retrieved from `http://lib.ugent.be/catalog/rug01:002300477`

Saad, Y. (1992). *Numerical methods for large eigenvalue problems.* Manchester University Press.

Shang-Hua, T. (2015(accessed April 10, 2018)). Sinular value decomposition and its applications [Computer software manual]. Retrieved from `http://slideplayer.com/slide/5189063/`

Taylor, R. (2016 (accessed April 10, 2018)). Dynamic mode decomposition in python [Computer software manual]. Retrieved from `http://www.pyrunner.com/weblog/2016/07/25/dmd-python/`

Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., & Kutz, J. N. (2013). On dynamic mode decomposition: theory and applications. *arXiv preprint arXiv:1312.0041*.