DYNAMIC MODE DECOMPOSITION

TABLE OF CONTENTS

ABSTRACT	3
THEORY	4
ALGORITHM	5
1.Singular value Decomposition (SVD)	5
2.Linear dynamical system A	6
3.Eigen value decomposition	7
4.Look back up high-dimensional space from low-dimensional space	7
5.Exact solution x	8
MATLAB IMPLEMENTATION (DMD ALGORITHM)	8
APPLICATIONS	14
Finance (Stock Market)	14
Neuroscience	15
Epidemiology	15
Fluid Dynamics	15
Video Processing	15
Thus, we have separated the foreground and back-ground separately from the video. Created DMD modes for both fore-ground and back-ground and extracted its features.	19
CONCLUSION	20

ABSTRACT

In this project, 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. The algorithm working has been implemented in MATLAB in this project. Along with that, an application is implemented. Finally, this project makes an attempt to understand the DMD's performance and its various applications so that it can be applied in a variety of fields in future.

INTRODUCTION

In the past 2 decades data-based techniques have been rapidly developed over the past two decades and have been widely used in numerous industrial sectors nowadays. The core idea behind this is to acquire useful information from the enormous amount of data available. Compared with well-developed model-based approaches, data-based techniques provide efficient alternatives. 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.

Generally, this algorithm was developed in the fluid mechanics community. Initially DMD was used to analyze the time evolution of fluid flows. In recent years its use has been popularized and has been used in analyzing the dynamics of nonlinear systems. The DMD has many innovations around compressive architectures, multi-resolution analysis and de-noising algorithms.

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 squares 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 approximate the underlying dynamics, even if those dynamics are nonlinear. It's 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 is 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).

THEORY

A dynamical system can be represented as $\frac{\mathrm{dx}}{\mathrm{dt}} = f(x, t, \mu),$ where 'x' defines measurements, 't' denotes time, $'\mu'$ is parametric dependence and 'f' indicates a system. This system 'f' is too complex and non-linear. Thought it is not very clear, it is approximated as, $\frac{\mathrm{dx}}{\mathrm{dt}} \approx \mathrm{Ax}$, where 'A' is a linear dynamical system with a low-rank structure.

When a linear dynamical system 'A' is formulated as differential equation, $\frac{\mathrm{dx}}{\mathrm{dt}} = \mathrm{Ax}, \quad x \in R^n,\ n \gg 1$. The differential equation with the linear dynamical system A can be easily solved. Then it's general exponential solution defined as, $x = \nu e^{\lambda t}$, where ν and λ are eigenvectors and eigenvalues of the linear system 'A', respectively. The problem of finding the eigenvectors ν and eigenvalues $\lambda$is a eigen problem defined as $\lambda\, \nu = A\, \nu$.

The eigenvalues λ and eigenvectors ν are found by solving the equations (called characteristic function) below:

$$\det|A - \lambda I| = 0$$
$$(A - \lambda_j I)\nu_j = 0$$

In order to fit the general DMD equation form, the notation of eigenvectors (ν) is changed to eigen function(Φ).

An exact solution of the differential equation is represented as:

$$x = \sum_{j=1}^{n} b_j \Phi_j e^{\lambda_j t}$$

The exact solution of the original dynamic system f is represented above. It preserves the time dynamic of 't'. Now, we know how to express the exact solution x from the linear dynamical system A. However, we are not sure how to express the linear dynamical system A.

Let we can measure $x_j = x(t_j)$ at any time point of j. We make big matrix concatenating the data from 1^{st} snapshot to $(m-1)^{\text{th}}$ snapshot.

$$X = \begin{bmatrix} \\ x_1 & x_2 & . & . & . & . & x_{m-1} \\ \\ \end{bmatrix}$$

Another matrix shifted by 1 time-step is defined as,

$$X' = \begin{bmatrix} \\ x_2 & x_3 & . & . & . & . & x_m \\ \\ \end{bmatrix}$$

In order to get this system, we should follow a data-driven method.

Our objective is to build a linear dynamical system A fitted with $\frac{\mathrm{dx}}{\mathrm{dt}} = \mathrm{Ax}$. The linear dynamical system A takes the data x from current state (j −1) to future state (j). Therefore, the linear dynamical system A is satisfied with the relationship below:

$X' = \mathrm{AX}$, where X' and X are the future state of X and the current state, respectively. The linear dynamical system A can be extracted using a pseudo inverse X' of X: $A = X'\mathrm{pinv}(X)$

We easily think about that, the linear dynamical system A performs a least-square fitting, from the current state X to the future state X'. This method is called by EXACT DMD.

ALGORITHM

1.Singular value Decomposition (SVD)

Let $\overline{X} \in R^{n \times (m-1)}$is dataset of a current state, its SVD is represented as:

$$\bar{X} = U \sum V^*$$

The Dimensions of each matrix are defined as

$$U \in \mathrm{R}^{\mathrm{n\ x\ n}}$$
$$\sum \in \mathrm{R}^{n\ \mathrm{x}\ (m-1)}$$

$$V \in \mathrm{R}^{(m-1)\ \mathrm{x}\ (m-1)}$$

In general, it is difficult to calculate the algorithm because the dimensions of the data $\bar{X}$ are too large. Fortunately, since all systems measuring $\bar{X}$ has a low-rank structure, 'rank -r truncation' is applied SVD

$$\bar{X} = U_r \sum r V_r^*$$

The truncated dimensions are defined as:

$$U_r \in \mathrm{R}^{\mathrm{n\ x\ r}}$$
$$\sum r \in \mathrm{R}^{\mathrm{r\ x\ r}}$$

$$V_{\mathrm{r}} \in \mathrm{R}^{(m-1)\ \mathrm{x\ r}}$$

2.Linear dynamical system A

A linear dynamical system A $\in \mathrm{R}^{\mathrm{n\ x\ n}}$is defined as:

$$A_{\mathrm{n\ x\ n}} = \bar{X}' * \operatorname{pinv}(X)$$

where $\operatorname{pinv}(V)$ defines a pseudo-inverse of $\bar{X}'$.

Since $\bar{X}$ was decomposed by SVD,the pseudo-inverse can be easily calculate as below:

$$\mathrm{pinv}(V) = V_r \sum_r^{-1} U_r^*$$

Then, the linear dynamical system $\mathrm{A}_{\mathrm{n\,x\,n}}$ can be reformulated by feeding the pseudo-inverse $\mathrm{pinv}(V)$:

$$\mathrm{A}_{\mathrm{n\,x\,n}} = \bar{X}' V_r \sum{}_r^{-1}$$

Although the linear dynamical system $\mathrm{A}_{\mathrm{n\,x\,n}}$was calculated, still the linear dynamical system $\mathrm{A}_{\mathrm{n\,x\,n}}$ is too huge.
To project the linear dynamical system

$\mathrm{A}_{\mathrm{n\,x\,n}}$into low-rank subspace, the similarity transform is performed:

$$\widetilde{A}_{\mathrm{r\,x\,r}} = U_r^* A U_r = U_r^* \left(\overline{X'}\, V_r \sum{}_r^{-1} U_r^* \right) U_r = U_r^* \bar{X}' V_r \sum{}_r^{-1}$$

where U_r is low-rank embedding space and $U_r^* U_r = I$. Now, the dimension of the low-rank embedded linear dynamical system

$\widetilde{A}$is defined as:

$$\widetilde{A} \in R^{\mathrm{r\,x\,r}}, \qquad\qquad r << n$$

3.Eigen value decomposition

$\widetilde{A}$ is the low-rank embedded linear dynamical system. Therefore, eigenvalue problem of is cheaply solved:

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

Where *W*=[eigen vectors] and Λ = [eigenvalues]

4.Look back up high-dimensional space from low-dimensional space

In the previous step, the eigenvectors *W* are calculated in the low-dimensional subspace, but not in an original high-dimensional space. The eigenvectors *W* can be returned to the original space by calculating below:

$$\Phi = \overline{X'}\, V_r {\sum}_r^{-1} W$$

where, Φ is DMD mode in the original space. The eigenvalues Λ do not change.

5.Exact solution x

We have performed from defining the linear dynamical system A to calculating the eigenvectors Φ and the eigenvalues Λ

Using the eigenvectors Φ and the eigenvalues Λ , the solution x can be calculated as:

$$\mathrm{x}(t) = \Phi e^{\Omega t} b = \sum \phi_k e^{\omega_k t}\, b_k$$

where Ω = log Λ and b is arbitrary constants. The arbitrary constants b can be decide to solve using initial condition problem:

$$\mathrm{x}(0) = \Phi \mathrm{b},$$

then,

$$\mathrm{b} = \Phi^{\mathrm{pinv}(\Phi)} \mathrm{x}(0)$$

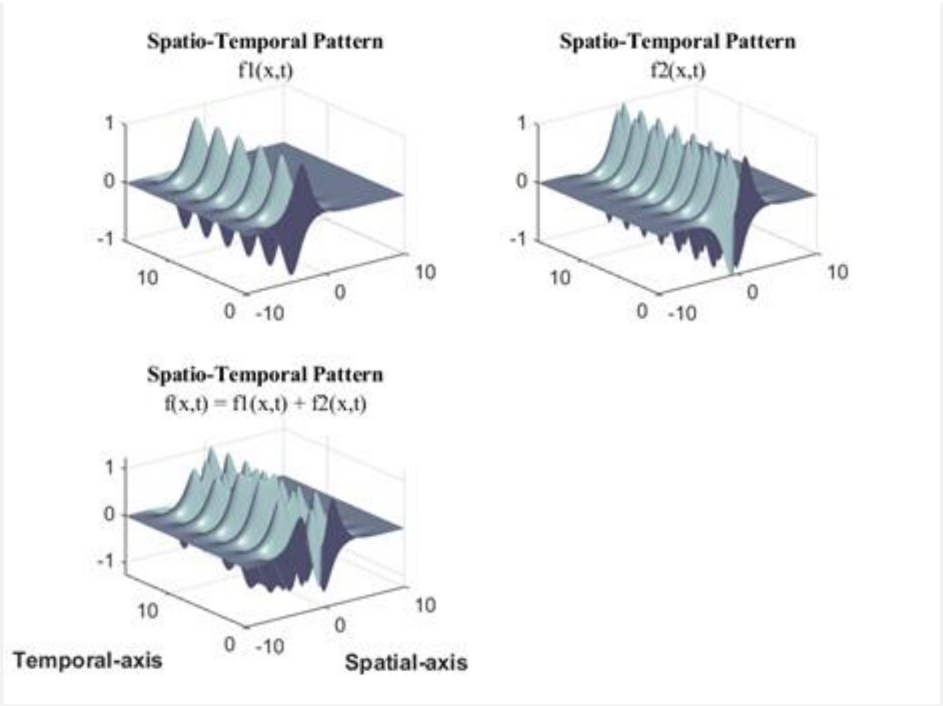
where $\Phi^{\mathrm{pinv}(\Phi)}$ is pseudo-inverse of Φ

MATLAB IMPLEMENTATION (DMD ALGORITHM)

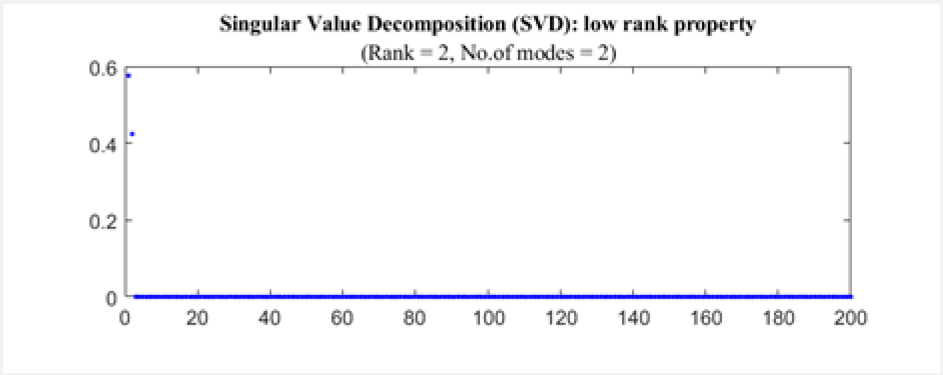
A sample spatio-temporal pattern has been created and DMD modes have been found for it along with extraction of the features of the signal. Along with that, from the modes the original signal has been re-constructed. Also, the future state of the signal has been predicted using the above mentioned DMD algorithm.

```
clc;
close all;
clear all;

% Generating grid
x = linspace(-10,10,400);
t = linspace(0,5*pi,200);
```

```
dt = t(2) - t(1);
[x_grid, t_grid] = meshgrid(x,t);

% Creation of spatio-temporal patterns(Patterns that vary with space and time)
f1 = sech(x_grid+3).*(1*exp(1i*2.3*t_grid));
f2 = (sech(x_grid).*tanh(x_grid)).*(2*exp(1i*2.8*t_grid));
f = f1 + f2;
[u, s, v] = svd(f.');

% Plotting the spatio-temporal patterns f1, f2, f
figure(1)
subplot(2,2,1)
surfl(x_grid,t_grid,real(f1));
shading interp;
colormap bone;
title("Spatio-Temporal Pattern","f1(x,t)",'FontName','times new roman','FontSize',12)

subplot(2,2,2)
surfl(x_grid,t_grid,real(f2));
shading interp;
colormap bone;
title("Spatio-Temporal Pattern","f2(x,t)",'FontName','times new roman','FontSize',12)

subplot(2,2,3)
surfl(x_grid,t_grid,real(f));
shading interp;
colormap bone;
xlabel('Spatial-axis','FontSize',10,'FontWeight','bold');
ylabel('Temporal-axis','FontSize',10,'FontWeight','bold');
title("Spatio-Temporal Pattern","f(x,t) = f1(x,t) + f2(x,t)",'FontName','times new roman','FontSize',12)
```

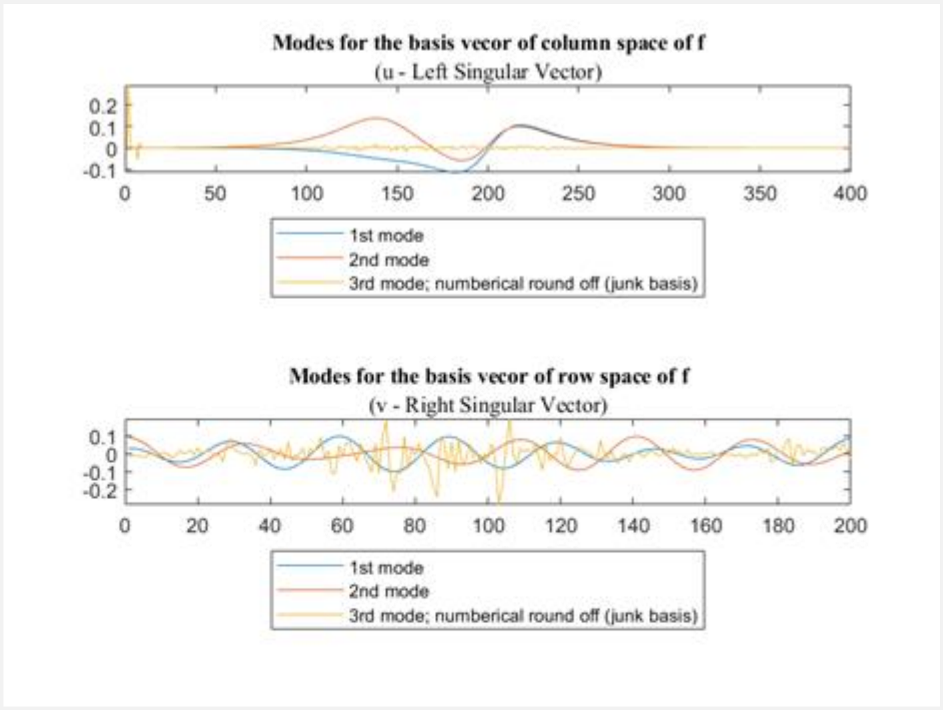


```
figure(2)
subplot(2,1,1);
plot(diag(s) / sum(diag(s)),'b.');
title('Singular Value Decomposition (SVD): low rank property','(Rank = 2, No.of modes = 2)','FontName','times new roman','FontSize',12)
```



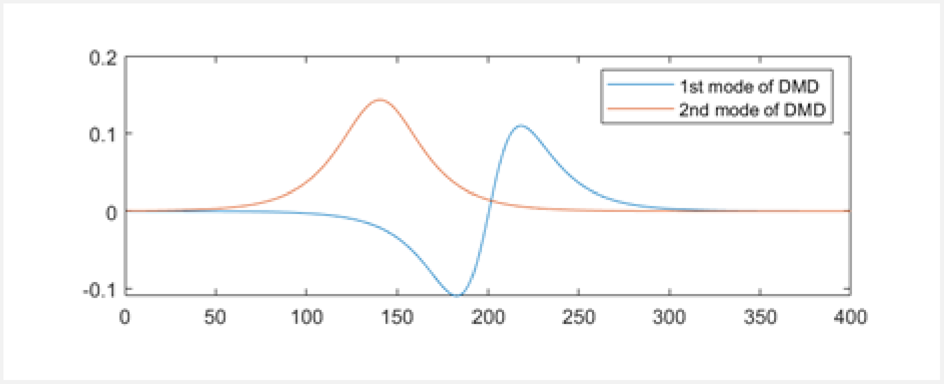
```
figure(3)
subplot(2,1,1)
plot(real(u(:,1:3)));
title('Modes for the basis vector of column space of f','(u - Left Singular Vector)','FontName','times new roman','FontSize',12);
legend('1st mode', '2nd mode', '3rd mode; numerical round off (junk basis)','Location','southoutside');

subplot(2,1,2)
plot(real(v(:,1:3)));
```

```
title('Modes for the basis vector of row space of f','(v - Right Singular Vector)','FontName','times new roman','FontSize',12);
legend('1st mode', '2nd mode', '3rd mode; numerical round off (junk basis)','Location','southoutside');
```



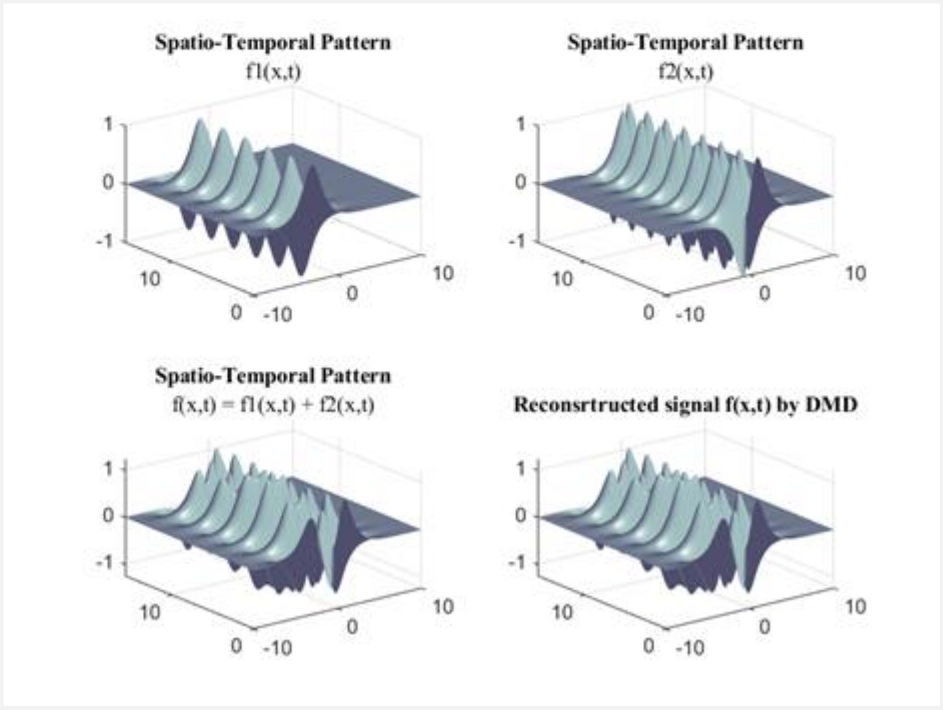
```
% Dynamic Mode Decomposition (DMD) - Algorithm
X = f.';
X1 = X(:,1:end-1);
X2 = X(:,2:end);

% Performing singular value decomposition - SVD
r = 2; % Rank-r truncation
[U, S, V] = svd(X1, "econ");
Ur = U(:,1:r);
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);

% Creating low-rank subspace matrix
%(similarity transform, least-square fit matrix, low-rank subspace matrix)
A_tilde = Ur'*X2*Vr*Sr^(-1);

% Eigen_Decomposition
[eig_vec, eig_val] = eig(A_tilde);

% Real space DMD mode
phi = X2*Vr*Sr^(-1)*eig_vec;   % DMD modes
lamdba = diag(eig_val);       % eigenvalue
omega = log(lamdba)/dt; % log of eigenvalue
```

```
figure(4)
subplot(2,1,1);
plot(real(u(:, 1:2)))
legend('1st mode of SVD','2nd mode of SVD');
plot(real(phi))
legend('1st mode of DMD','2nd mode of DMD');
```



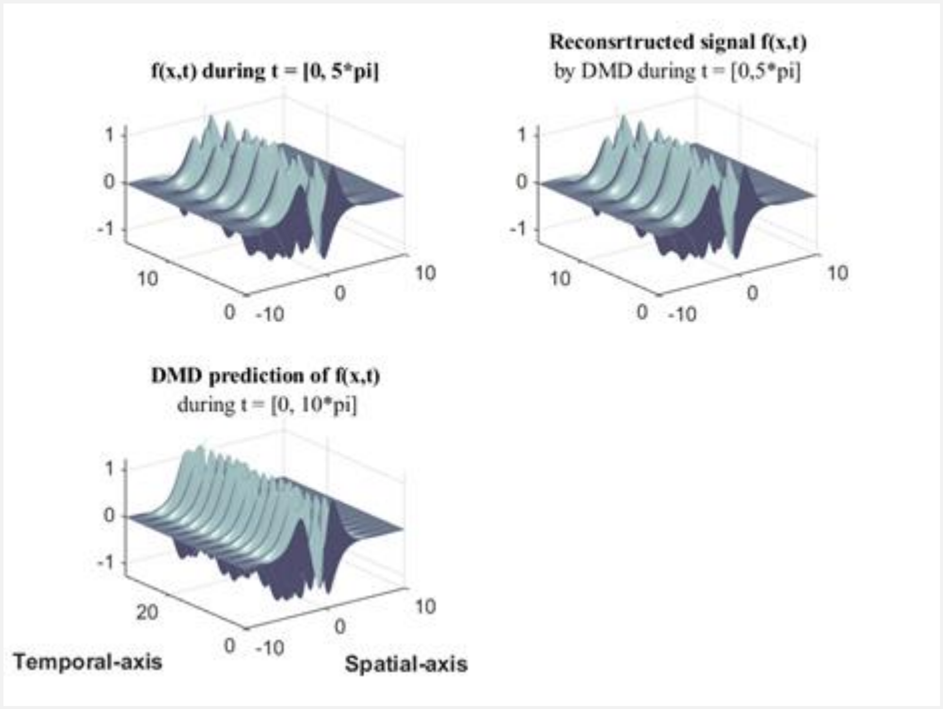
```
% Reconstructing the signal
X1 = X(:,1); % time = 0
b = pinv(phi)*X1; % initial value;
t_dynamic = zeros(r,length(t));
for i = 1:length(t)
    t_dynamic(:,i) = (b.*exp(omega*t(i)));
end
f_dmd = phi*t_dynamic;

% Plot of the reconstructed signal with dmd
figure(5)
subplot(2,2,1)
surfl(x_grid,t_grid,real(f1));
shading interp;
colormap bone;
title("Spatio-Temporal Pattern","f1(x,t)",'FontName','times new roman','FontSize',12)

subplot(2,2,2)
surfl(x_grid,t_grid,real(f2));
shading interp;
colormap bone;
title("Spatio-Temporal Pattern","f2(x,t)",'FontName','times new roman','FontSize',12)

subplot(2,2,3)
surfl(x_grid,t_grid,real(f));
shading interp;
colormap bone;
```

```
title("Spatio-Temporal Pattern","f(x,t) = f1(x,t) + f2(x,t)",'FontName','times new roman','FontSize',12)

subplot(2,2,4)
surfl(x_grid,t_grid,real((f_dmd).'));
shading interp;
colormap bone;
title("Reconstructed signal f(x,t) by DMD",'FontName','times new roman','FontSize',12)
```



```
% Predicting the future state using time dynamics
t_ext = linspace(0,10*pi, 400);
[x_grid_ext, t_grid_ext] = meshgrid(x, t_ext);
t_ext_dynamic = zeros(r, length(t_ext));
for i = 1:length(t_ext)
    t_ext_dynamic(:,i) = (b.*exp(omega*t_ext(i)));
end
f_dmd_ext = phi*t_ext_dynamic;

% Plotting the signals
figure(6)
subplot(2,2,1)
surfl(x_grid,t_grid,real(f));
shading interp;
colormap bone;
title("f(x,t) during t = [0, 5*pi]",'FontName','times new roman','FontSize',12)

subplot(2,2,2)
surfl(x_grid,t_grid,real(f_dmd).');
```

```
shading interp;
colormap bone;
title('Reconstructed signal f(x,t)','by DMD during t = [0,5*pi]','FontName','times new roman','FontSize',12)

subplot(2,2,3)
surfl(x_grid_ext,t_grid_ext,real(f_dmd_ext).');
shading interp;
colormap bone;
xlabel('Spatial-axis','FontSize',10,'FontWeight','bold');
ylabel('Temporal-axis','FontSize',10,'FontWeight','bold');
title('DMD prediction of f(x,t)',' during t = [0, 10*pi]','FontName','times new roman','FontSize',12)
```



APPLICATIONS

It has many applications in various domains such as computer vision, neuroscience, biology, finance and many more.

Finance (Stock Market)

Algorithmic trading schemes are growing in importance in the modern financial world. An application of algorithmic trading could be done by using dynamic mode decomposition. The dynamic mode decomposition is a data analysis tool which is capable of characterizing the dynamical systems in an equation free manner by decomposing the system into low-rank structures, dynamic modes, whose temporal evolution is known. The method enables financial market prediction using dynamic modes.

Neuroscience

There is a broad need in the neuroscience community to understand and visualize large-scale recordings of neural activity, big data acquired by tens or hundreds of electrodes simultaneously recording dynamic brain activity over minutes to hours. Such dynamic datasets are characterized by coherent patterns across both space and time, yet existing computational methods are typically restricted to analysis either in space or in time separately. Here, dynamic mode decomposition (DMD), is used due to large-scale neuronal recordings. Here DMD approach is validated on subdural electrode array recordings from human subjects performing a known motor activation task. Also, DMD can be used in combination with machine learning techniques to develop a novel method to extract sleep spindle networks from the same subjects. DMD is generally applicable as a powerful method in the analysis and understanding of large-scale recordings of neural activity.

Epidemiology

The development and application of quantitative methods to understand disease dynamics and plan interventions is becoming increasingly important in the push toward eradication of human infectious diseases. Here the method is demonstrated on different infectious disease sets. We demonstrate how DMD can aid in the analysis of spatial-temporal disease data. DMD is poised to be an effective and efficient computational analysis tool for the study of infectious disease.

Fluid Dynamics

In fluid dynamics, modal analysis of unsteady fluid flows over moving structures is significant in terms of state estimation and control. However, the underlying algorithm of the DMD requires a fixed spatial domain, which is an obstacle for applying the DMD on the numerically investigated problems using dynamic meshes.

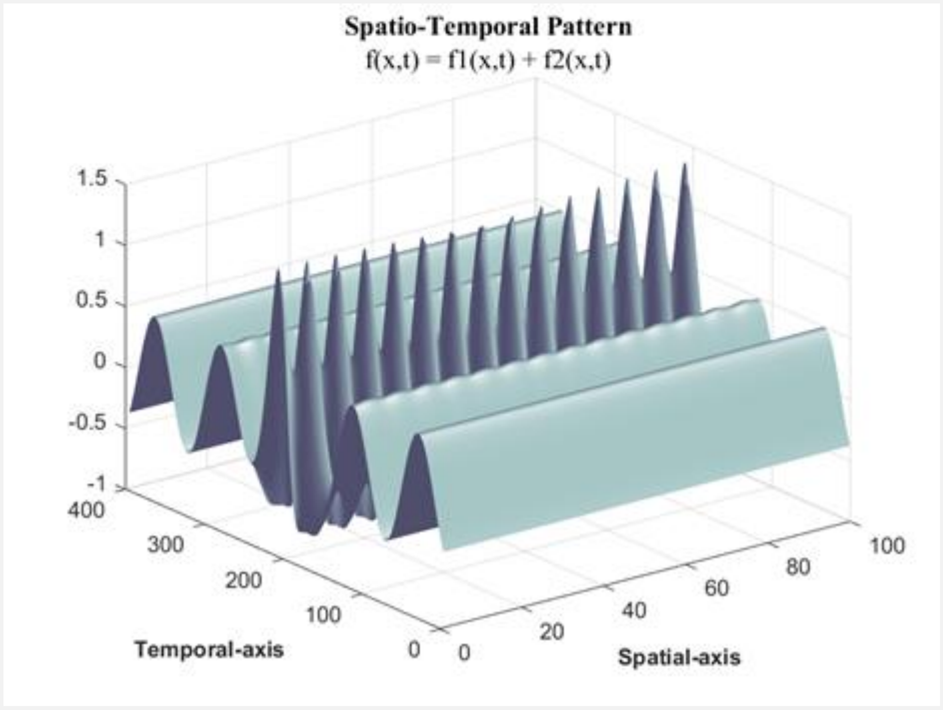
Video Processing

The Dynamic Mode Decomposition (DMD) is a spatiotemporal matrix decomposition method capable of background modeling in video streams. DMD is a regression technique that integrates Fourier transforms and singular value decomposition.

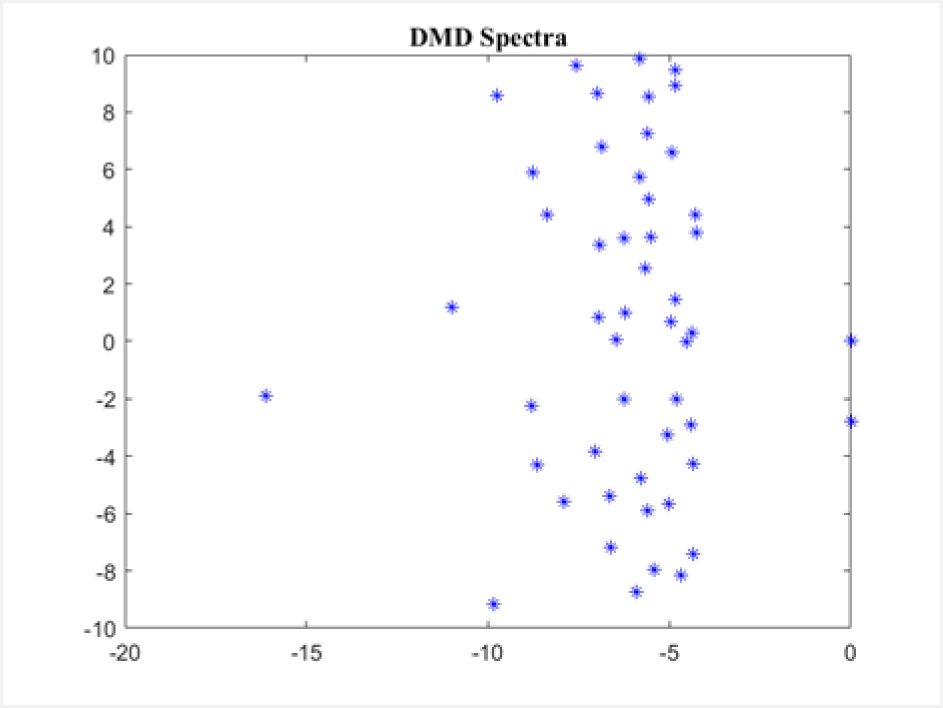
Innovations in compressed sensing allow for a scalable and rapid decomposition of video streams that scales with the intrinsic rank of the matrix, rather than the size of the actual video.

Implementation of Video Processing (MATLAB)

```
% Discretization of space and time
x = linspace(-15,15,400);
t = linspace(0,10*pi,100);
dt = t(2) - t(1);
[X_grid,T_grid] = meshgrid(x,t);

%% Creating spatio-temporal patterns f1 & f2
f1 = 0.5*cos(X_grid) .* (1+0*T_grid);  % time-independent!
f2 = (sech(X_grid).*tanh(X_grid)) .* (2*exp(1j*2.8*T_grid));

% Combining signals and making data matrix
f = (f1 + f2)';

figure(1);
surfl(real(f));
shading interp;
colormap bone;
xlabel('Spatial-axis','FontSize',10,'FontWeight','bold');
ylabel('Temporal-axis','FontSize',10,'FontWeight','bold');
title("Spatio-Temporal Pattern","f(x,t) = f1(x,t) + f2(x,t)",'FontName','times new roman','FontSize',12);
```

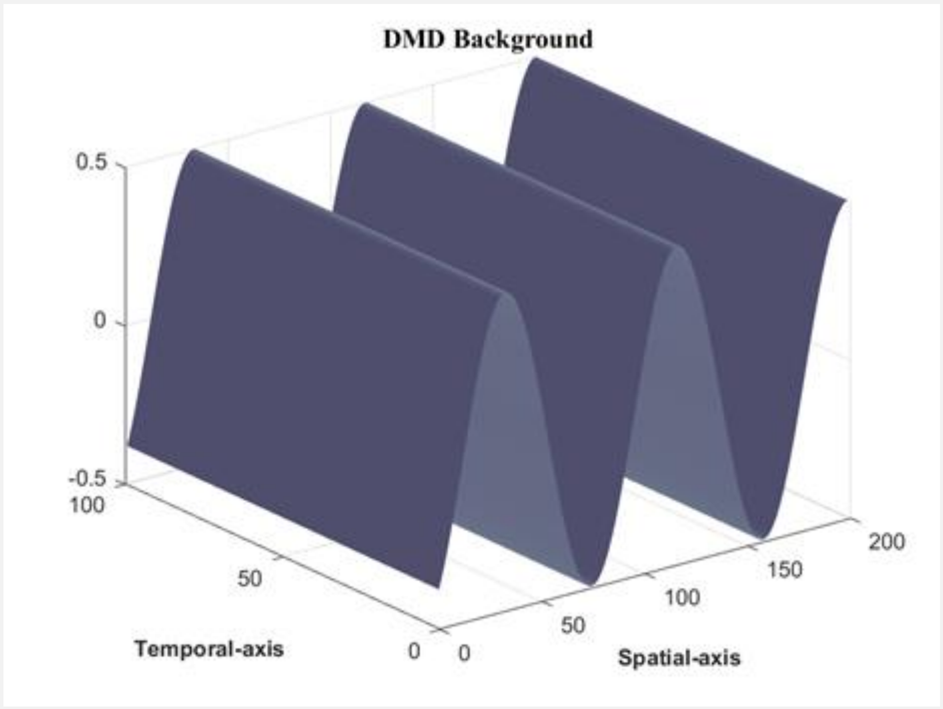


```
% Create data matrices for DMD
X1 = f(:,1:end-1);
X2 = f(:,2:end);

% Performing SVD and rank-50 truncation
r = 50; % rank truncation
[U, S, V] = svd(X1, 'econ');
Ur = U(:, 1:r);
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);

% Build A_tilde and DMD Modes
A_tilde = Ur'*X2*Vr/Sr;

[eig_vec, eig_val] = eig(A_tilde);
Phi = X2*Vr/Sr*eig_vec;  % DMD Modes

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

figure(2);
plot(omega, 'b*');
title('DMD Spectra','FontName','times new roman','FontSize',12);
```



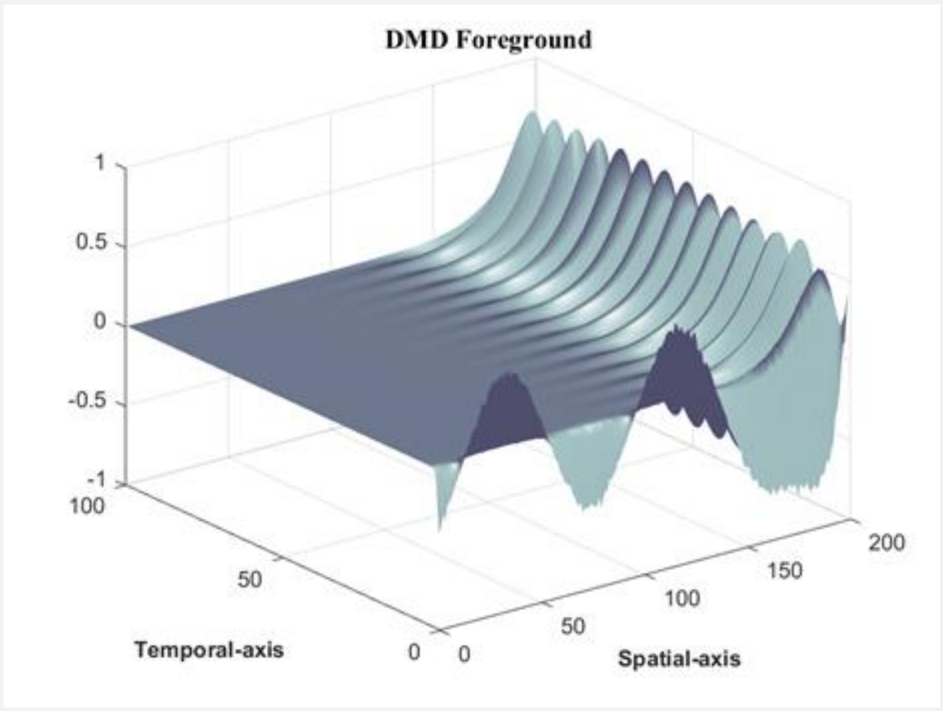
```
% Declaring fore-ground and background
bg = find(abs(omega) < 1e-2);
fg = setdiff(1:r, bg);
```

```
omega_fg = omega(fg); % foreground
Phi_fg = Phi(:,fg); % DMD foreground modes

omega_bg = omega(bg); % background
Phi_bg = Phi(:,bg); % DMD background modes

% DMD Background
b = Phi_bg \ f(:, 1);
X_bg = zeros(numel(omega_bg), length(t));
for tt = 1:length(t)
    X_bg(:, tt) = b .* exp(omega_bg .* t(tt));
end
X_bg = Phi_bg * X_bg;
X_bg = X_bg(1:n,:);

figure(3);
surfl(real(X_bg'));
shading interp;
colormap bone;
xlabel('Spatial-axis','FontSize',10,'FontWeight','bold');
ylabel('Temporal-axis','FontSize',10,'FontWeight','bold');
title('DMD Background','FontName','times new roman','FontSize',12);
```



```
% DMD Foreground
b = Phi_fg \ f(:, 1);
X_fg = zeros(numel(omega_fg), length(t));
```

```
for tt = 1:length(t)
    X_fg(:, tt) = b .* exp(omega_fg .* t(tt));
end
X_fg = Phi_fg * X_fg;
X_fg = X_fg(1:n,:);

figure(4);
surfl(real(X_fg'));
shading interp;
colormap bone;
xlabel('Spatial-axis','FontSize',10,'FontWeight','bold');
ylabel('Temporal-axis','FontSize',10,'FontWeight','bold');
title('DMD Foreground','FontName','times new roman','FontSize',12);
```



Thus, we have separated the foreground and back-ground separately from the video. Created DMD modes for both fore-ground and back-ground and extracted its features.

CONCLUSION

In this project, 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 exists 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. Also, it’s an equation-free technique which can be applied to complex nonlinear systems. Despite its limitations, DMD is a powerful tool in analyzing and predicting dynamical systems. It has many applications in various domains such as computer vision, neuroscience, biology, finance. It would be great to explore it from different backgrounds to explore its true potential.

*****