Spatial Temporial Koopman decompsition for crypto-valuation prediction

Anirudh Bhaskar

Anirudh Vadakedath

Rohith Ramakrishnan

Abstract

A prediction analysis applied on a dynamical system such as crypto currency using the principles of spatio temporial koopman decomposition for a higher dimensions. We go through the basic theory with our data-driven approach using higher order dynamic mode decomposition taking certain dimensions from sentiment analysis from news and tweets generated on the topic "Bitcoin" and "Dogecoin" which are famous crypto-currency in circulation in the economy.

Introduction on Mathematical Concepts used -

Koopman Operator

In the context of dynamical systems, we can interpret the data as knowledge of some variable(s) related to the state of the system. To put this into mathematical form is to assume that data is evaluation of functions of the state, theses functions are also known as observables of the system. In the case of a fluid, the unforced motion of an incompressible fluid inside a box constitutes a dynamical system, and the state space is the set of all smooth velocity fields on the flow domain that satisfy the incompressibility condition and according to the rule

of evolution, the Eular equations, the state changes with time. In this case pressure or velocity at a given point in the flow domain, total kinetic energy of the flow or the velocity at a set of points can be considered as the observables on the system. Considering a continuous-time dynamical system as $x^{t+1} = T(x^t)$, let $g : S \to \Re$ be a real-valued observable dynamical system. The collection of all such observables form a linear vector space. The Koopman operator, given as U, is a linear transformation on the vector space given by $\mathrm{Ug}(x) = \mathrm{goT}(x)$, where '*o*' denotes the composition operation.

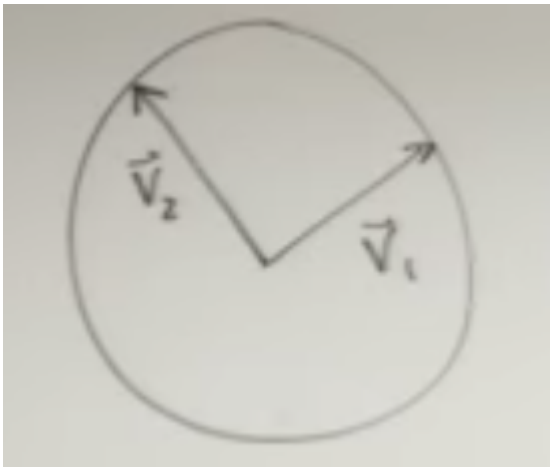
Singular Value Decomposition

Consider $\vec{x} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ and $A = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix}$, then $\vec{y} = A\bar{x} = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$. $\vec{y}$ is basically a rotated and stretched form of $\vec{x}$.

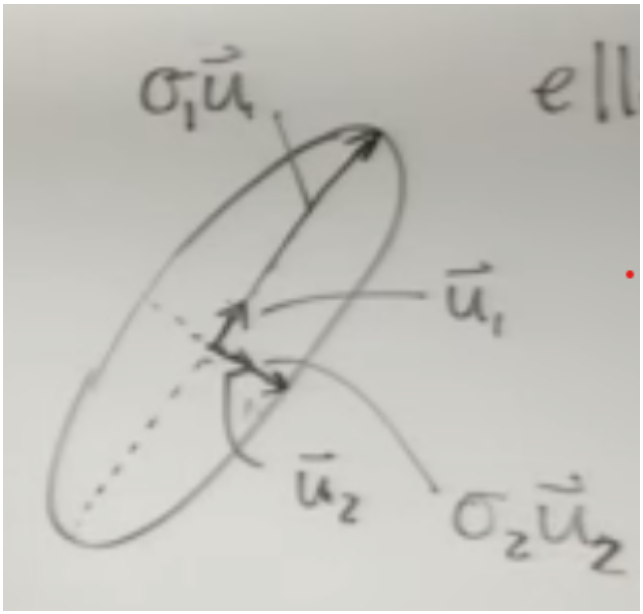
Now $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is a rotation matrix and $\begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}$, where α is a *positive real number*, is a stretching/ translational matrix and A is composed of these two.

Now consider a circle in 2-dimensions with its orthogonal direction co-ordinates being v_1 and v_2.

(in n-dimensions it would be $v_1, v_2, \ldots, v_n$)



When an A_{mxn}, is operated on this circle it becomes an ellipse (*it is rotated and then stretched*):



where u_1 and u_2 are the unit vectors in those directions and σ_1 and σ_2 are the distances along the unit vectos.

$\therefore$ $v_1, v_2, \ldots, v_n$ vector space gets transformed to $u_1, u_2, \ldots, u_n$ with $\sigma_1, \sigma_2, \ldots, \sigma_n$ as stretching factors, when multiplied by A. The $\sigma_1, \sigma_2, \ldots, \sigma_n$ are known as the ***singular values***.

Therefore in vector terms: $A\vec{v_j} = \sigma_j\vec{u_j},\ j = 1, 2, ..., n$.

$$\begin{bmatrix} A \end{bmatrix}_{mxn} \begin{bmatrix} \vec{v_1} & \vec{v_2} & \ldots & \vec{v_n} \end{bmatrix}_{nxn} = \begin{bmatrix} \vec{u_1} & \vec{u_2} & \ldots & \vec{u_n} \end{bmatrix}_{mxn} \begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_n \end{bmatrix}_{nxn}$$

$\therefore \mathrm{AV} = \hat{U}\hat{\Sigma}$

from V and $\hat{U}$ it could be seen that these are unitary transformation matrices and orthogonal too. Thus $V^{-1} = V^*$and $\hat{U}^{-1} = \hat{U}^*$. On right multiplication with V^{-1} we get

$A = \hat{U}\hat{\Sigma}V^* \longrightarrow$ reduced SVD of A.

Since $\hat{U}$ is of size mxn, m-n extra silent columns can be added to make it mxm and the $\hat{\Sigma}$ can be stretched with m-n rows of 0's to make its size mxn.

$\therefore A_{mxn} = U_{mxm}\Sigma_{mxn}V^*_{nxn} \longrightarrow$ **Singular Value Decomposition** of $\boldsymbol{A}$, where

($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$).

To get the U, Σ and V matrices we follow these steps:

$$A^TA = (U\Sigma V^*)^T(U\Sigma V^*)$$
$$A^TA = V\Sigma U^*U\Sigma V^*$$
$$A^TA = V\Sigma^2V^*$$
$$A^TAV = (V\Sigma^2V^*)V$$
$$\therefore A^TAV = \Sigma^2V$$

This is the same as $Av = \lambda v$, thus by solving this eigen-value problem we would get the Σ^2 and V values.

$$AA^T = (U\Sigma V^*)(U\Sigma V^*)^T$$
$$AA^T = U\Sigma V^*V\Sigma U^*$$
$$AA^T = U\Sigma^2U^*$$
$$AA^TU = (U\Sigma^2U^*)U$$
$$\therefore AA^TU = \Sigma^2U$$

```
clear all;close all;clc;
A = randi(10,3);
[u1, s1, v1] = svd(A);
[v2,ss21] = eig(A'*A);
[u2, ss12] = eig(A*A');
s2 = sqrt(ss12);
u1
```

```
u1 = 3×3
   -0.4864    0.6727   -0.5576
   -0.6741   -0.6949   -0.2503
   -0.5559    0.2542    0.7915
```

```
u2
```

```
u2 = 3×3
    0.5576    0.6727    0.4864
    0.2503   -0.6949    0.6741
   -0.7915    0.2542    0.5559
```

```
disp("*****")
```

```
*****
```

```
s1
```

```
s1 = 3×3
   18.4219         0         0
         0    4.7969         0
         0         0    3.2591
```

```
s2
```

```
s2 = 3×3
    3.2591         0         0
         0    4.7969         0
         0         0   18.4219
```

```
disp("*****")
```

```
*****
```

```
v1
```

```
v1 = 3×3
   -0.8159    0.2077    0.5396
   -0.2920    0.6576   -0.6945
   -0.4991   -0.7242   -0.4759
```

```
v2
```

```
v2 = 3×3
   -0.5396   -0.2077    0.8159
    0.6945   -0.6576    0.2920
    0.4759    0.7242    0.4991
```

A^TA and AA^T are hermition matrices thus the eigen values are ***always*** *positive, real and distinct*.

Dynamic Mode Decomposition

A method to reduce the dimensionality of an algorithm. It is seen as an application extracted from the idea of Koopman Spectral analysis.

We consider this method to be more data-driven and on futher explain DMD for higher order dimension non-linear systems.

Basic idea

Non-linear to best linear approximation,

$$\frac{dx}{\mathrm{dt}} = f(\underline{x}, t, \mu) \rightarrow \frac{dx}{\mathrm{dt}} = A\underline{x}$$

We measure x for any time-point,

$$x_j = \underline{x}(t_j)$$

So,

$$X = [x_1 \;\; x_2 \;\; x_3 \;\; \cdots\cdots \;\; x_{m-1}]$$
$$X' = [x_2 \;\; x_3 \;\; x_4 \;\; \cdots\cdots \;\; x_m]$$

Our objective is to find the best matrix that takes X to $X' \ldots\ldots X' = \mathrm{AX}$

Here matrix **A** is seen as an operator that takes us Δt times into the future from $X \,\mathrm{to}\, X'$

Now from the equation we can calculate X by multiplying its respective pseudo inverse on both sides,

Final equation $\longrightarrow$ $A = X'X^{+}$

This equation is the exact depiction of working of DMD but cannot be practically implemented in data-driven cases. Given our X is has N columns multiplied by N rows of its pseudo inverse we get NxN matrix **A** which is computationally not feasible. So we can define it as an algorithm.

Step 1: Decompose X using SVD

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

We reduce the rank using rank truncation and use that as the subspace for building the model

$$X = U_r\Sigma_r V^*_r$$

Step 2:

$$\begin{aligned} A &= X'X^+ \\ &= X'V\,\Sigma^{-1}U^* \end{aligned}$$

We will be working with a similarity transform of A dentoted as $\tilde{A}$

$$\tilde{A} = U^* A\,U$$

$$\tilde{A}_{\text{rxr}} = U^* X'\,V\,\Sigma^{-1}$$

Step 3:

we do an eigenvalue decomposition

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

where W is [eigenvectors] and Λ is $\begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_r \end{bmatrix}$

Step 4:

Bring it back to the original higher rank

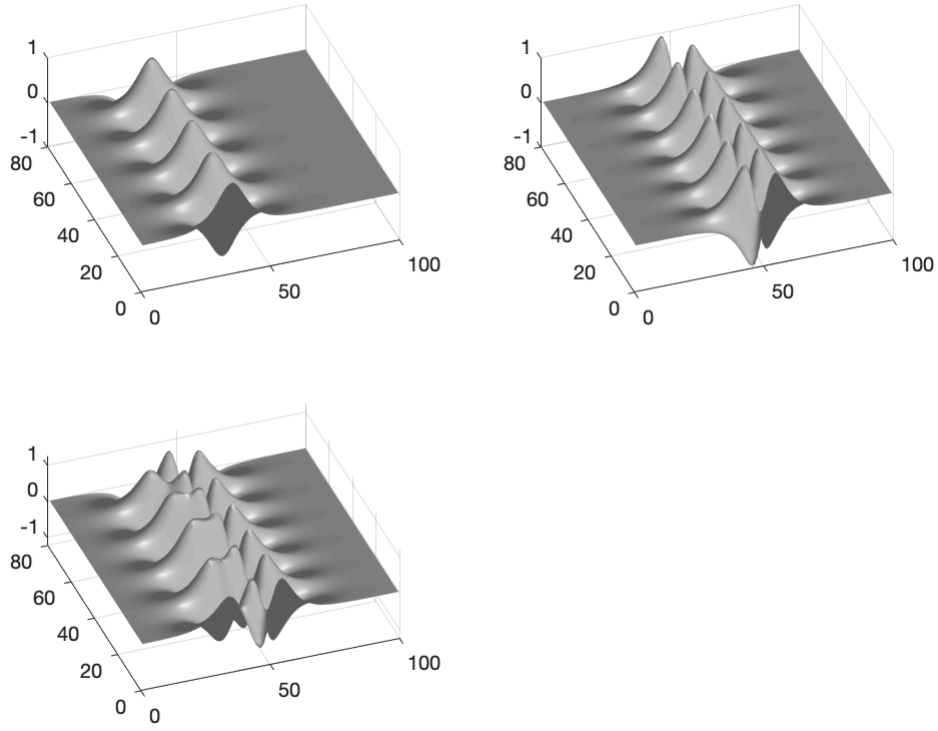
$$\phi = X'\,V\,\Sigma^{-1}\,W \longrightarrow \textbf{DMD modes}$$

$$X(t) = \phi\, e^{\Omega t}\, b = \sum_{k=1}^{r} \phi_k\, e^{w_k t}\, b_k \qquad \text{where} \quad b = \frac{\phi}{x(0)}$$

```
x=linspace(-10,10,100);
t=linspace(0,4*pi,80); dt=t(2)-t(1);
[X,T]=meshgrid(x,t);
```

```
f1=sech(X+3).*(1*exp(i*2.3*T));
f2=(sech(X).*tanh(X)).*(2*exp(i*2.8*T));
```

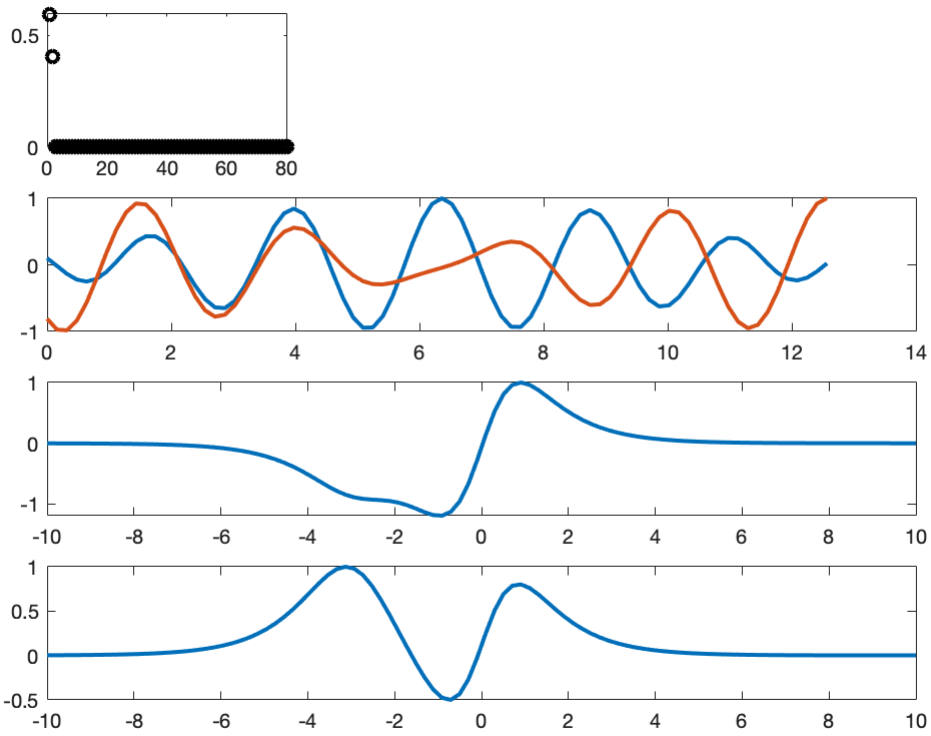
```
f=f1+f2;

subplot(2,2,1), surfl(real(f1)), shading interp, colormap(gray), view(-20,60)
subplot(2,2,2), surfl(real(f2)), shading interp,colormap(gray), view(-20,60)
subplot(2,2,3), surfl(real(f)), shading interp,colormap(gray), view(-20,60)
```




```
[u,s,v]=svd(f');

figure(2)
subplot(4,3,1), plot(diag(s)/sum(diag(s)),'ko','Linewidth',[2])
subplot(4,1,2), plot(t,v(:,1)/max(v(:,1)),t,v(:,2)/max(v(:,2)),'Linewidth',[2])
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

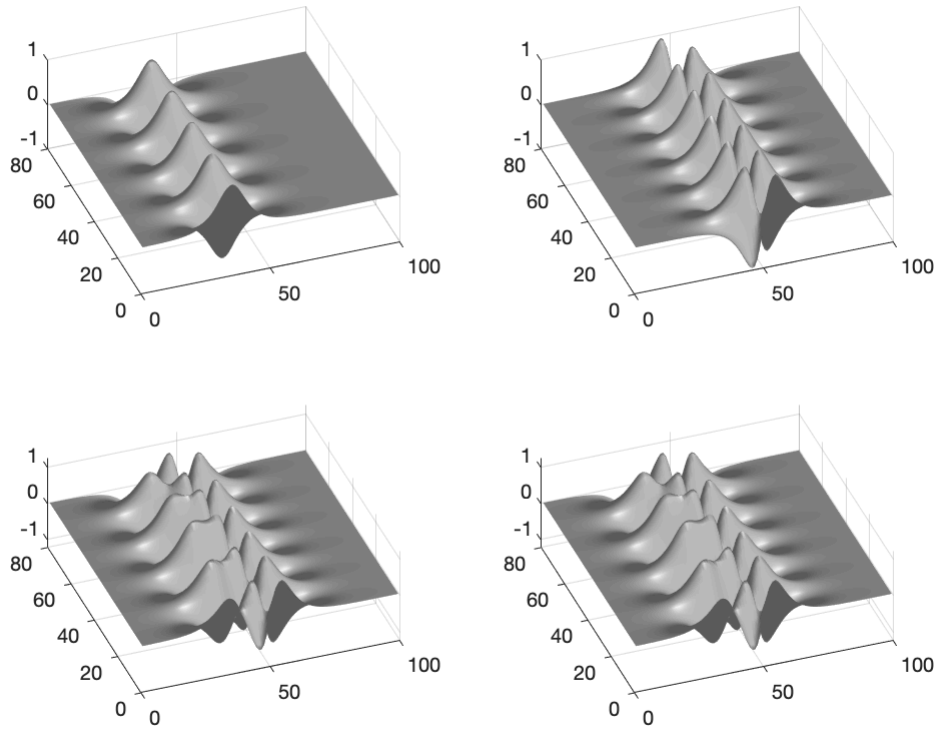
```
subplot(4,1,3), plot(x,real(u(:,1))/max(real(u(:,1))),'Linewidth',[2])
subplot(4,1,4), plot(x,real(u(:,2))/max(real(u(:,2))),'Linewidth',[2])
```



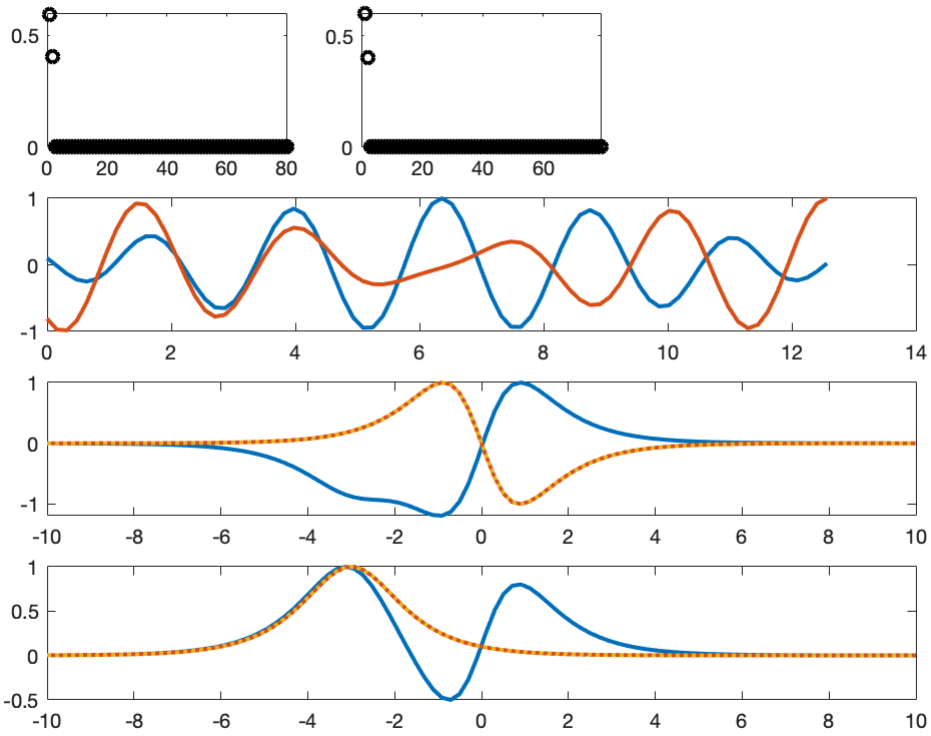

```
X = f.'; X1 = X(:,1:end-1); X2 = X(:,2:end);
r=2;
```

```
[U2,Sigma2,V2] = svd(X1,'econ'); U=U2(:,1:r); Sigma=Sigma2(1:r,1:r); V=V2(:,1:r);
Atilde = U'*X2*V/Sigma;
[W,D] = eig(Atilde);
Phi = X2*V/Sigma*W;

mu = diag(D);
omega = log(mu)/dt;

u0=f(1,:).';
y0 = Phi\u0;  % pseudo-inverse initial conditions
u_modes = zeros(r,length(t));
for iter = 1:length(t)
     u_modes(:,iter) =(y0.*exp(omega*t(iter)));
end
u_dmd = Phi*u_modes;
```

```
figure(1)
subplot(2,2,4)
surfl(real(u_dmd.')), shading interp,colormap(gray), view(-20,60)
```




```
figure(2),
subplot(4,3,2), plot(diag(Sigma2)/sum(diag(Sigma2)),'ko','Linewidth',[2])
subplot(4,1,3), hold on, plot(x,real(-Phi(:,1)/max(Phi(:,1))),'Linewidth',[2])
subplot(4,1,4), hold on, plot(x,real(Phi(:,2)/max(Phi(:,2))),'Linewidth',[2])

subplot(4,1,3), plot(x,-sech(x).*tanh(x)/max(sech(x).*tanh(x)),':','Linewidth',[2])
subplot(4,1,4), plot(x,sech(x+3)/max(sech(x+3)),':','Linewidth',[2])
```



Higher Order Dynamic Mode Decomposition

Lets assume a dynammic system given by the vector state variable (), of size , we consider a set of snapshots at equispaced values of *t*

$v_k \equiv v(t_k)$

where $t_k = t_1 + (k-1).\Delta t$, for $k = 1, \ldots, K$

Assuming the dynamical system is infinite-dimensional, the finite- dimensional state vector V generally results from spatial discretization. The snapshots

are organized in a snapshot matrix (whose columns are the snapshots), as

$V_1^k = [v_1 \quad v_2 \quad \ldots \quad v_k]$

Metrics: *relative root mean square* (RMS) error, defined as

$$\text{error} = \sqrt{\frac{\sum_{k=1}^{K} \| v_k^{\text{approx}} - v_k \|_2^2}{\sum_{k=1}^{K} \| v_k \|_2^2}}$$

Algorithm:

SVD :

For dimensionality reduction, truncated SVD is applied to V_1^k, which yields,

$V_1^k \approx \mathrm{P}\Sigma Q^T \equiv P\widehat{\mathrm{T}_1^K}$ (or $\upsilon_k \equiv P\widehat{t_k}$)

where $\widehat{\mathrm{T}_1} = \Sigma Q^T$

where the matrix $\widehat{\mathrm{T}_1^K}$ is known as the dimension-reduced snapshot matrix and their columns, the *dimension-reduced snapshots*, $\widehat{t_k}$.

DMDd:

For ordinary DMD, we consider the Koopman Assumption which can be formulated as,

$\widehat{t}_{k+1} = \widehat{R}^t . \widehat{t}_k$

Where the M x M matrix $\widehat{R}^t$ is computed from dimension-reduced snapshots via pseudoinverse.

Reduced Snapshots can be represented by,

$\widehat{t_{k+d}} \approx \widehat{R}_1 \widehat{t_{k+d-1}} + \widehat{R}_2 \widehat{t_{k+d-2}} + \ldots\ldots\ldots + \widehat{R}_k \widehat{t_k}$

The enlarged snapshots can be represented by,

$$\widetilde{t_k} \equiv \begin{bmatrix} \widehat{t_k} \\ \widehat{t_{k+1}} \\ \ldots \\ \widehat{t_{k+d-2}} \\ \widehat{t_{k+d-1}} \end{bmatrix}$$

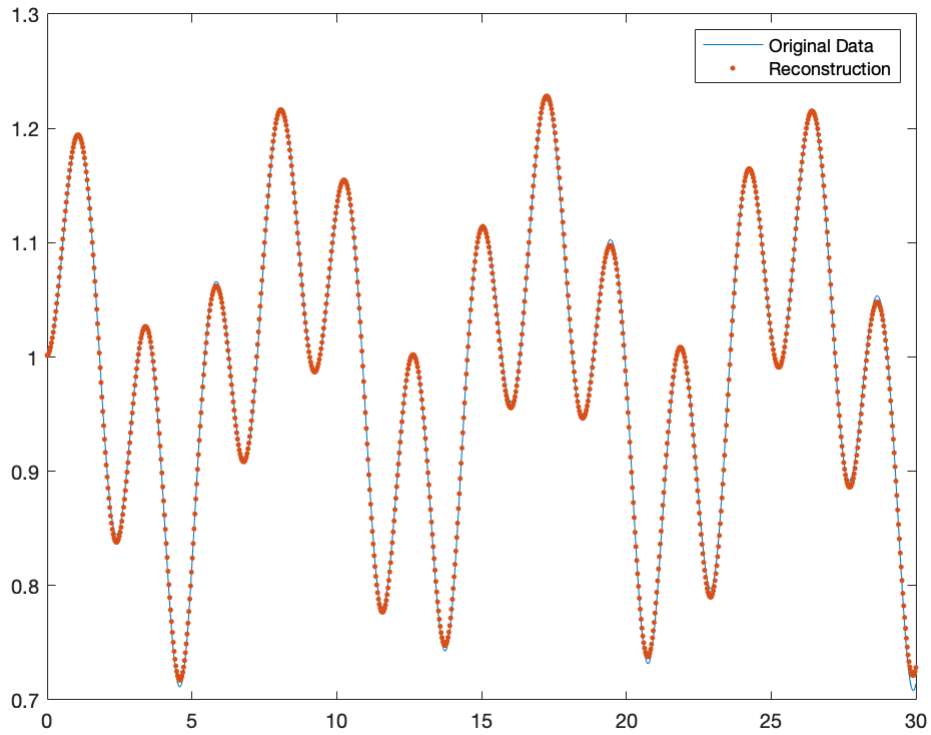
SVD is applied to $\widetilde{t_k}$, a second dimension reduction that gives a new set of enlarges dimension-reduced snapshots, $\widehat{\widetilde{t_k}}$

which will satisfy - $\widehat{\widetilde{t_{k+1}}} = \widetilde{R}^t . \widehat{\widetilde{t_k}}$, where $\widetilde{R}^t$ is computed via the pseudo-inverse as defined above.

```
t = linspace(0,30,1000);
V = sqrt(1 + (1/2).*sin(t).*sin(sqrt(3)*t));
d = 800;
varepsilon1 = 10^-10;
varepsilon = 10^-4;
figure
[Vreconst,~] =DMDd_SIADS(d,V,t,varepsilon1,varepsilon)
```

```
Vreconst = 1×1000 complex
   1.0012 + 0.0000i   1.0020 + 0.0000i   1.0034 + 0.0000i   1.0057 + 0.0000i ···
```

```
plot(t,V)
hold on
plot(t,real(Vreconst),'.')
legend("Original Data","Reconstruction")
```

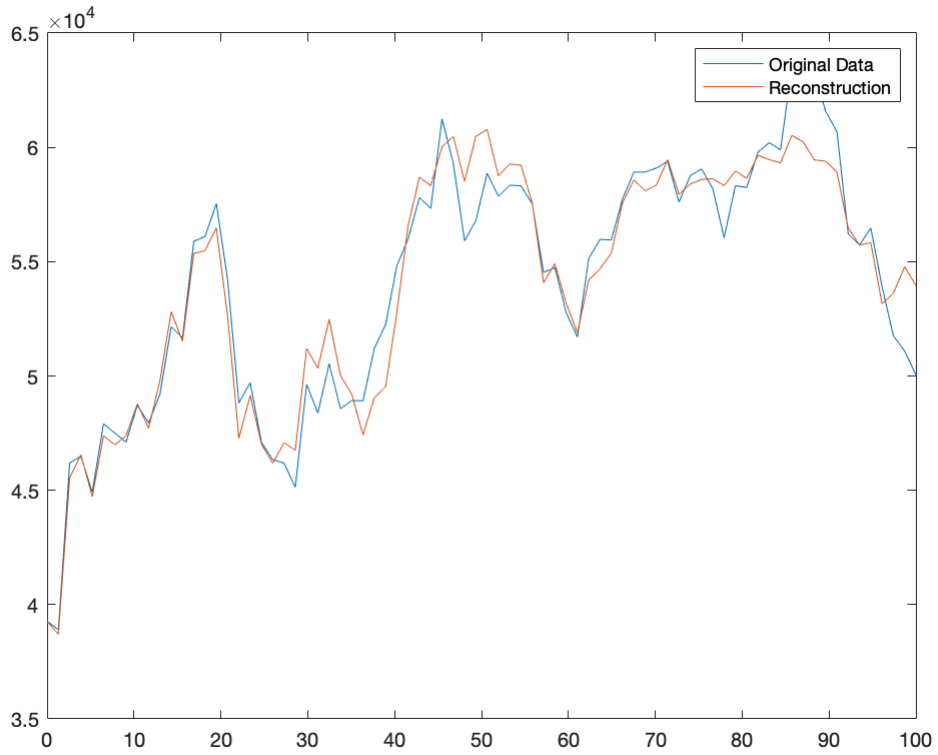



```
% Import data from text file
% Script for importing data from the following text file:
%
```

```
%    filename: /Volumes/Rohith/College/SpatioTemporalKoopman/data/BTC-USD_daily-06_02-24_04.csv
%
% Auto-generated by MATLAB on 03-Jun-2021 10:10:32

% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 7);

% Specify range and delimiter
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["Var1", "Var2", "Var3", "Var4", "Close", "Var6", "Var7"];
opts.SelectedVariableNames = "Close";
opts.VariableTypes = ["string", "string", "string", "string", "double", "string", "string"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Specify variable properties
opts = setvaropts(opts, ["Var1", "Var2", "Var3", "Var4", "Var6", "Var7"], "WhitespaceRule", "pr
opts = setvaropts(opts, ["Var1", "Var2", "Var3", "Var4", "Var6", "Var7"], "EmptyFieldRule", "au

% Import the data
tbl = readtable("/Volumes/Rohith/College/SpatioTemporalKoopman/data/BTC-USD_daily-06_02-24_04.c

% Convert to output type
Close = tbl.Close;

% Clear temporary variables
clear opts tbl

t = linspace(0,100,length(Close));
d = 46;
varepsilon1 = 10^-10;
varepsilon = 10^-4;
[Vreconst,~] =DMDd_SIADS(d,Close',t,varepsilon1,varepsilon)
```

```
Vreconst = 1×78 complex
10^4 ×
   3.9266 + 0.0000i   3.8704 - 0.0000i   4.5549 - 0.0000i   4.6551 - 0.0000i ···
```

```
figure
plot(t,Close)
hold on
plot(t,real(Vreconst))
legend("Original Data","Reconstruction")
```



Spatio Temporal Koopman Decomposition

For simplicity pupose we will consider one spatio-variable X and a temporal-variable T for and snapshot V_1^k of dimension J x K.

Lets consider a toy model with discritised values $x_j = x_1 + j\Delta x$ and $t_j = t_1 + k\Delta t$, for j = 1,...,J and k = 1,....,K

STKD can be represented by

$$v(x_j, t_k) = \sum_{m,n} a_{\mathrm{mn}}\, u_{\mathrm{mn}}\, e^{(v_m + ik_m)x_j + (\delta_n + i\omega_m)t_k}$$

u_n are conveniently normalized spatial modes, a_n > 0 are the mode amplitudes, and and ; v_m and ik_m are the associated

growth rates and frequencies, respectively.

STEP : 1 -

As in HODMD, SVD is performed on the Snapshot V_1^k with a tunable value ϵ_1 which determines the number of SVD modes M.

$$V_1^k = P \sum Q^T \equiv \left(\hat{X}_1^J\right)^T \hat{T}_1^K,$$

$\widehat{X}_1^J = \sqrt{\sum} P^T$

$\widehat{T}_1^J = \sqrt{\sum} Q^T$

These matrices are the reduced spatial and temporal snapshots and their columns are $\widehat{x}_j$and $\widehat{t}_k$ are the reduced spatial and temporal snapshots.

STEP : 2 -

The HODMD algorithm is applied to thr spatial and temporal components

$\widehat{x}_j \approx \sum_{m=1}^{M} \widehat{a}_m^x . \widehat{q}_m^x e^{(v_m + ik_m)x_j}$, for j=1,...,J

$\widehat{t}_k \approx \sum_{n=1}^{N} \widehat{a}_n^t . \widehat{q}_n^t e^{(\delta_n + i\omega_n)t_k}$, for k=1,...,K

The numbers of retained modes, M and N, are the *spec- tral spatial and temporal complexities*, respectively, and are selected in terms of the spatial

and temporal amplitudes, $\widehat{a}_m^x$ and $\widehat{a}_n^t$. M and N are the smallest values of the indexes and such that

$\frac{\widehat{a_{M+1}}^x}{\widehat{a_1}^x} < \varepsilon_2$

$\frac{\widehat{a_{N+1}}^x}{\widehat{a_1}^x} < \varepsilon_2$

Illustration of the STKD in a toy model,

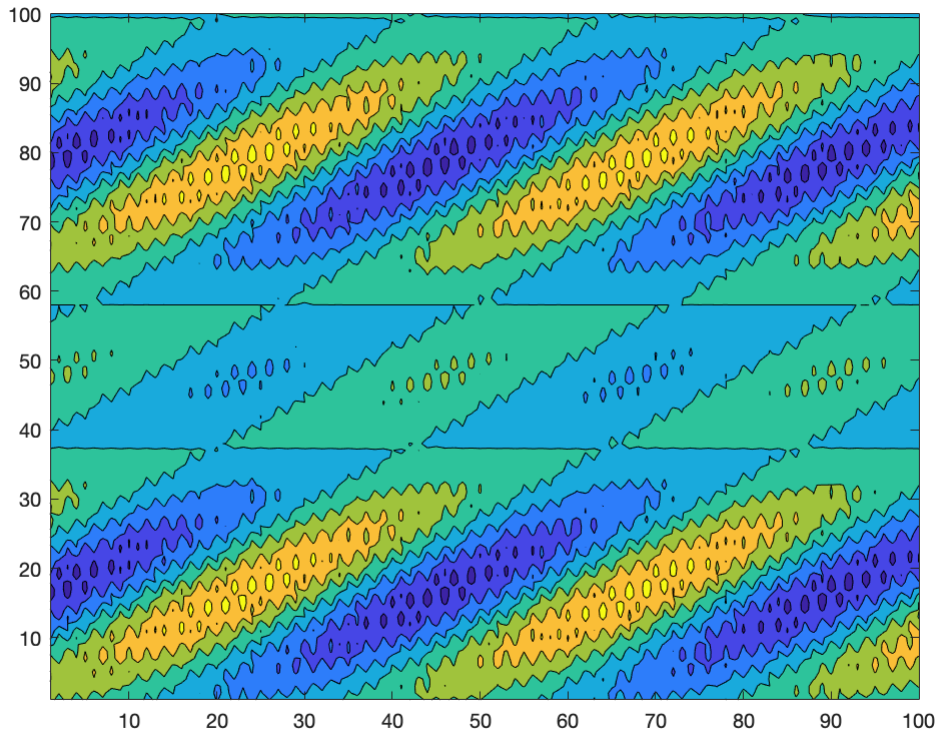
u (,) = (0.5 + sin).[2.cos($k_1 . x - w_1 . t$) + (0.5).cos($k_2 . x - w_2 . t$)]

```
x = linspace(0,10,100);
t = linspace(0,100,100);
U = zeros(length(x));

varepsilon1 = 10^-8;
varepsilon2 = 10^-7;

for i=1:length(x)
    for j=1:length(t)
        X = x(i); T = t(j);
        U(i,j) = (0.5 + sin(X)) * (2*cos(2*X - (2*pi/45)*T) + 0.5*cos(10*X - sqrt(10)*T));
    end
end
figure
[Vreconst,Modes,Amplitudes,Amplitudesx,GrowthRatex,Frequencyx,Amplitudest,GrowthRatet,Frequency
CalculateDMDdSdT(10,1,x,t,U,varepsilon1,varepsilon2);
```

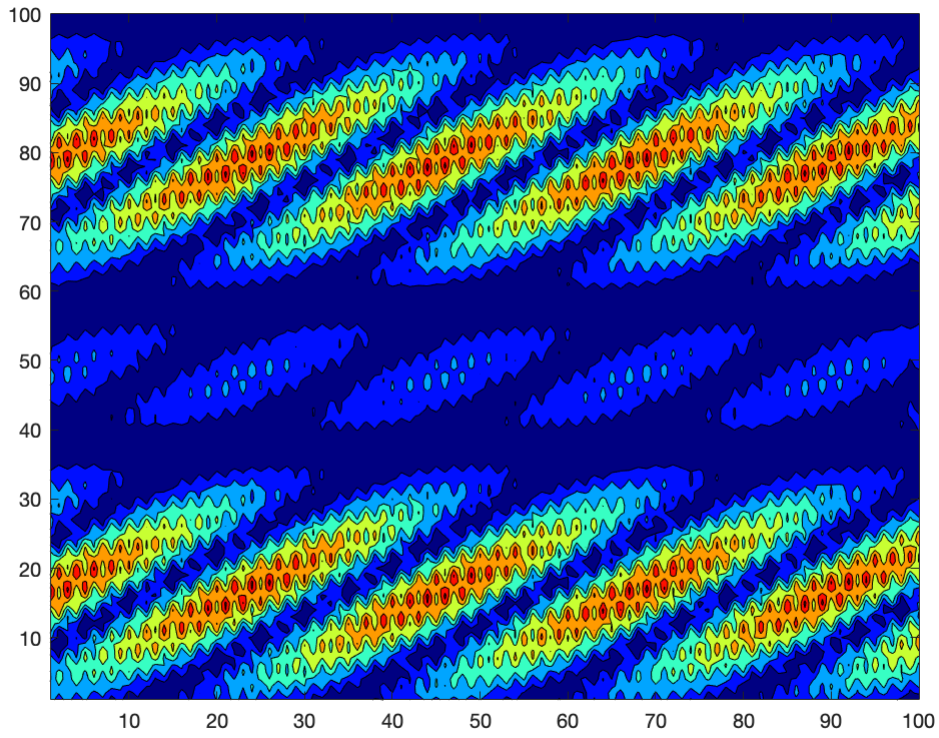
```
Spatial Dimension: singular values
```

```
     4

Spatial dimension reduction
    12

size of a
ans = 1×2
    12     1
size of Q
ans = 1×2
     4    12
Spectral complexity
    12
ans =
'Spectral complexity'
kk3 = 4
    48

    12

Number of spatial modes
    12

Number of temporal modes
     4
```

```
contourf(U)
```




```
figure
contourf(abs(Vreconst))
shading('interp')
colormap(jet)
```

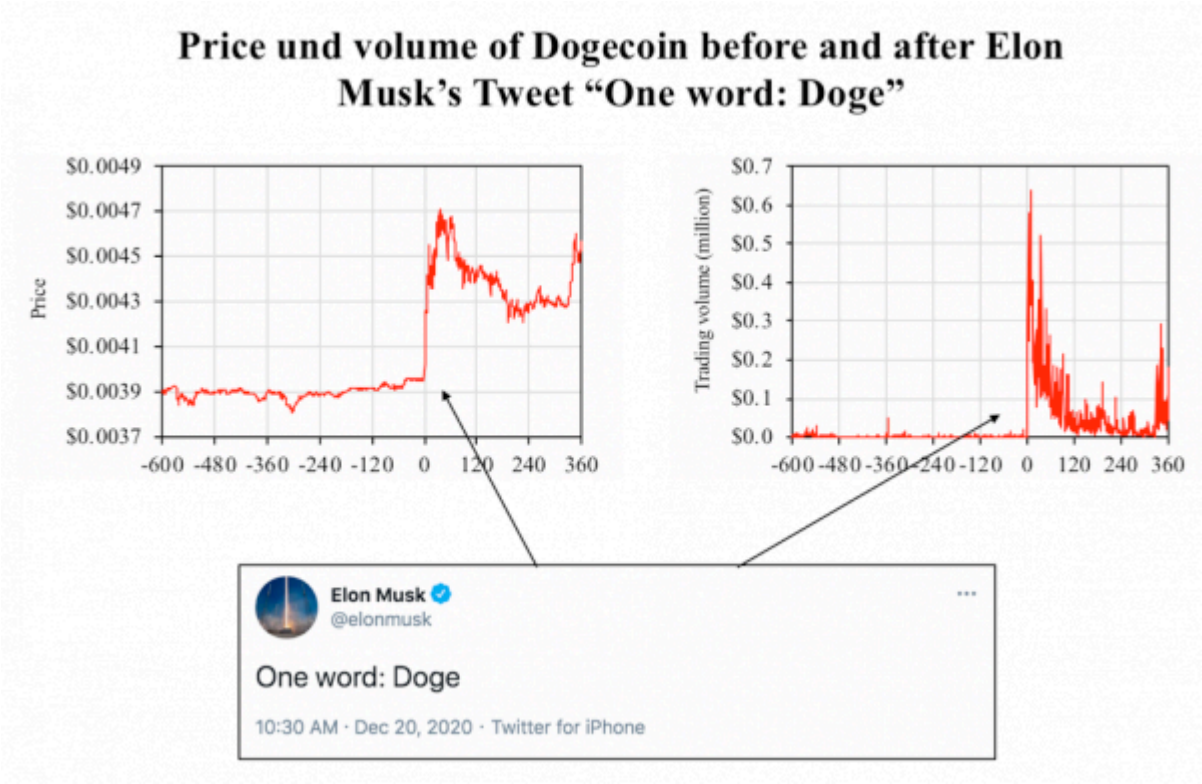


Stock Prediction via STKD

The information shared on social media generates sentiments and reactions in investors that influence their decisions to buy or sell financial assets, especially on the stock market [1]. News shared on social media, whether true or not, cause changes in the trends of international stock market indices [2]. Some studies confirm the relationship between sentiments generated by social media in investors and market trends, and the current study continues this path in exploring pandemic periods with a lexicon-based approach to detect polarity in financial news on Twitter.

Our hypothesis is that sentiments expressed via Tweets and News can cause a potential change in stock and could be a crucial factor for predicting very volatile prices such as **Cryptocurrency**.In the recent times, we saw hoe Elon Musks Tweets single handed brought down the price of Bitcoin and sky-rocketted Dogecoin's price.

In the experiment performed we will be collecting News and Tweets and analysiong the sentiments of the same which will act as the spatial component of our analysis and temporal being time.



Bitcoin prediction with Tweets

For this experiment tweets between 6th February to 24th April 2021 were collected along with the closing price of Bitcoin in the same Date range. The collected tweet swill have numerous tweets for any given day hence an mean of all the sentiments for the day was taken.

The sentiment analysis was done in python using *'nlptown/bert-base-multilingual-uncased-sentiment'*

```
path = os.path.join(os.getcwd(), 'data', 'Bitcoin_tweets.csv')
data = pd.read_csv(path)

data.drop(columns=['user_name', 'user_location', 'user_description', 'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'hashtags', 'source', 'is_retweet'], inplace=True)
text = data['text'].fillna("#").tolist()


tokenizer = AutoTokenizer.from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment')

model = AutoModelForSequenceClassification.from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment')

def sentiment_score(review):
    tokens = tokenizer.encode(review, return_tensors='pt')
    result = model(tokens)
    return int(torch.argmax(result.logits))+1

sent=[]
for i in tqdm(text):
    sent.append(sentiment_score(p.clean(i)))
```

```
data['sent'] = pd.DataFrame(sent)
```

The above code corresponds to the python script for extracting sentiments.

SInce the data isnt very huge, a single snapshot will will very sparse.

```
clc
clear all

load sent_snp.mat
V = sent;
s = size(V);
Time = linspace(0,1,s(2));
Exis = linspace(0,1,s(1));


% APPLY STKD TO V
[J,K]=size(V);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% STKD  PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SET PARAMETERS STKD
% Tolerances
% SVD
varepsilon1=1e-8;
% DMD
varepsilon2=1e-7;
% Set index d: dSpace for spatial analysis and dTime for temporal analysis
dSpace=2;
dTime=1;

[Vreconst,Modes,Amplitudes,Amplitudesx,GrowthRatex,Frequencyx,Amplitudest,GrowthRatet,Frequency
CalculateDMDdSdT(dSpace,dTime,Time,Exis,V,varepsilon1,varepsilon2);
```

```
Spatial Dimension: singular values
    18

Spatial dimension reduction
    20
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  2.692283e-17.
size of a
ans = 1×2
    20     1
size of Q
ans = 1×2
    18    20
Spectral complexity
    20
ans =
'Spectral complexity'
kk3 = 18
   360
```

```
   360

Number of spatial modes
    20

Number of temporal modes
    18
```

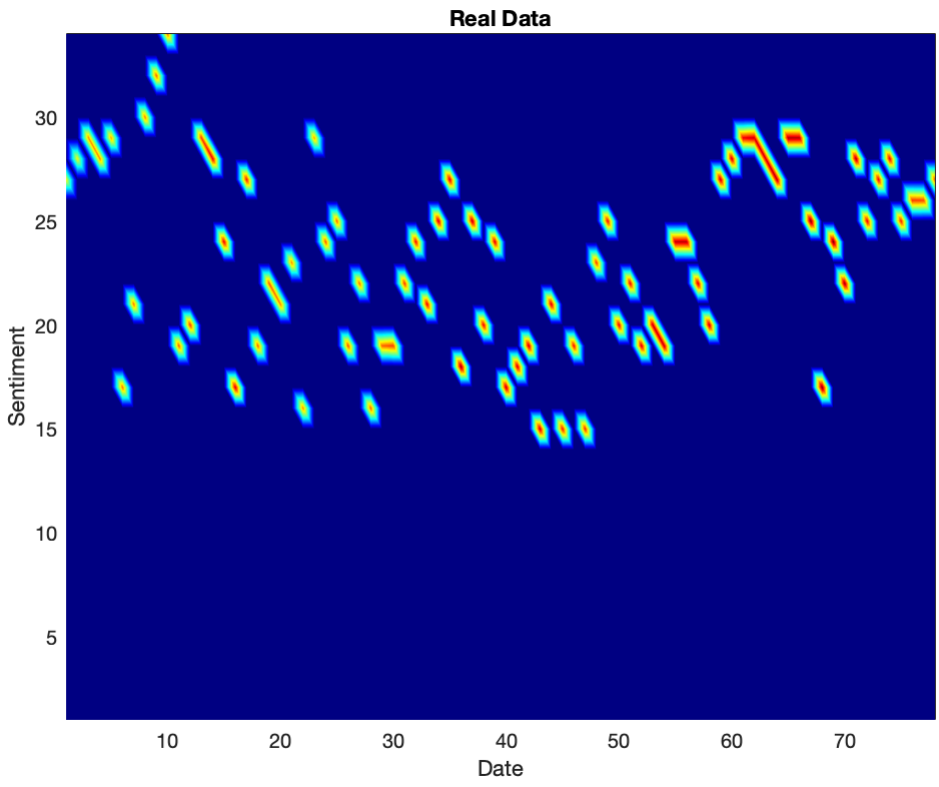
```
% CALCULATE RRMS ERROR
dif=Vreconst-V;
errorRMS=norm(dif(:))/norm(V(:));
disp('Reconstruction error: RRMSE')
```

```
Reconstruction error: RRMSE
```

```
disp(errorRMS)
```

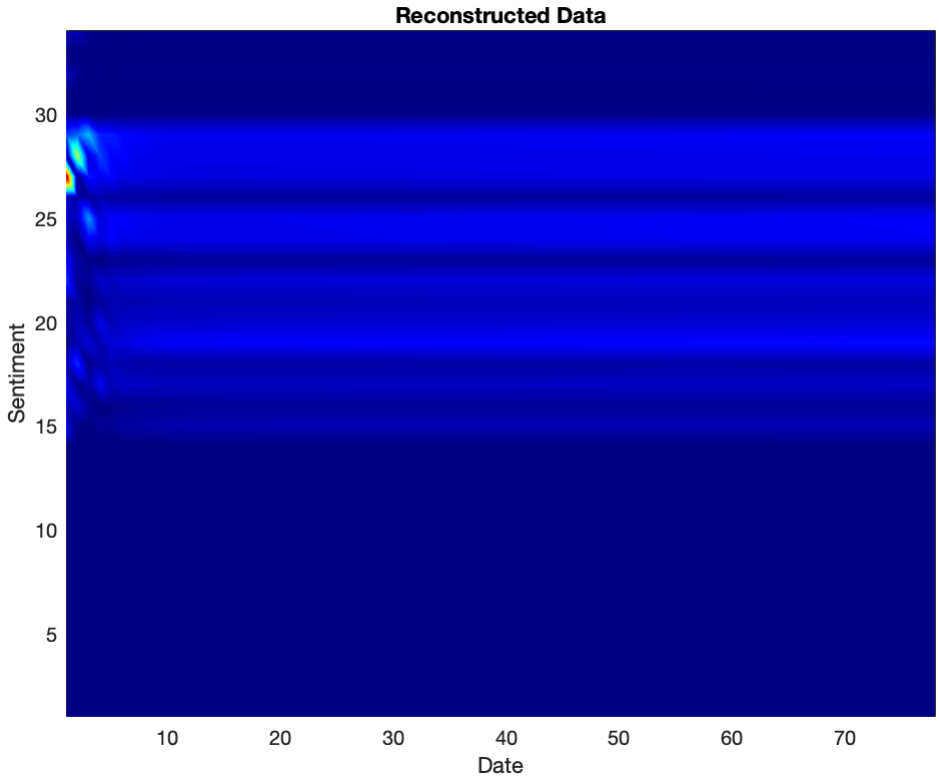
```
    0.9540
```

```
% PLOT RECONSTRUCTION
figure
pcolor(V)
shading('interp')
colormap(jet)
xlabel('Date')
ylabel('Sentiment')
title("Real Data")
```

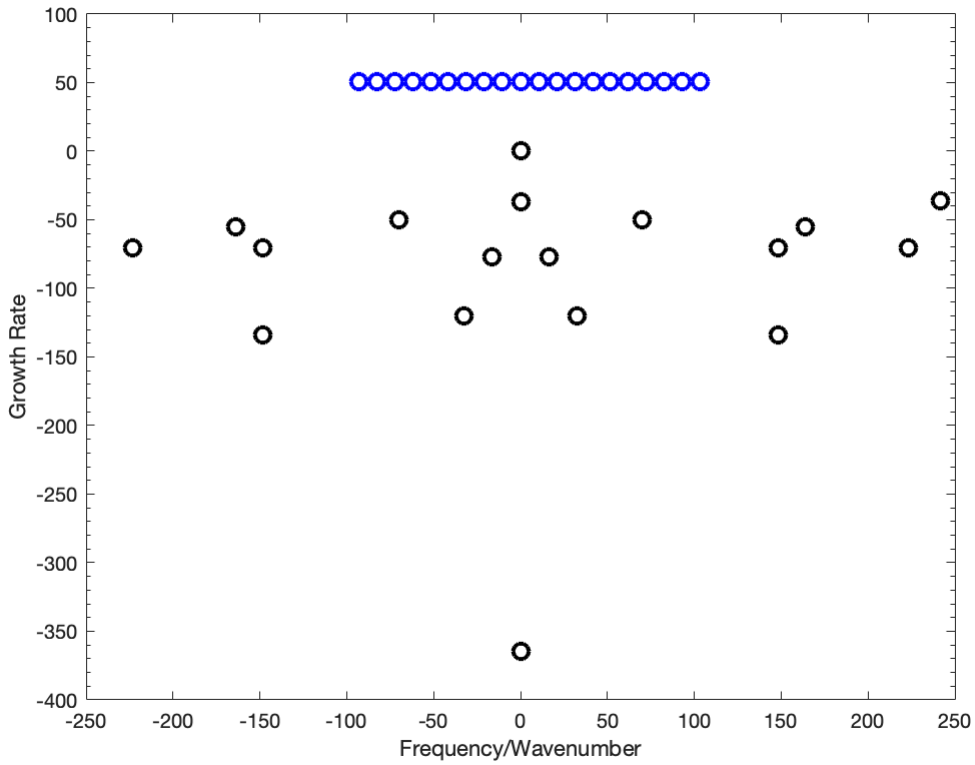



```
h9=figure;
```

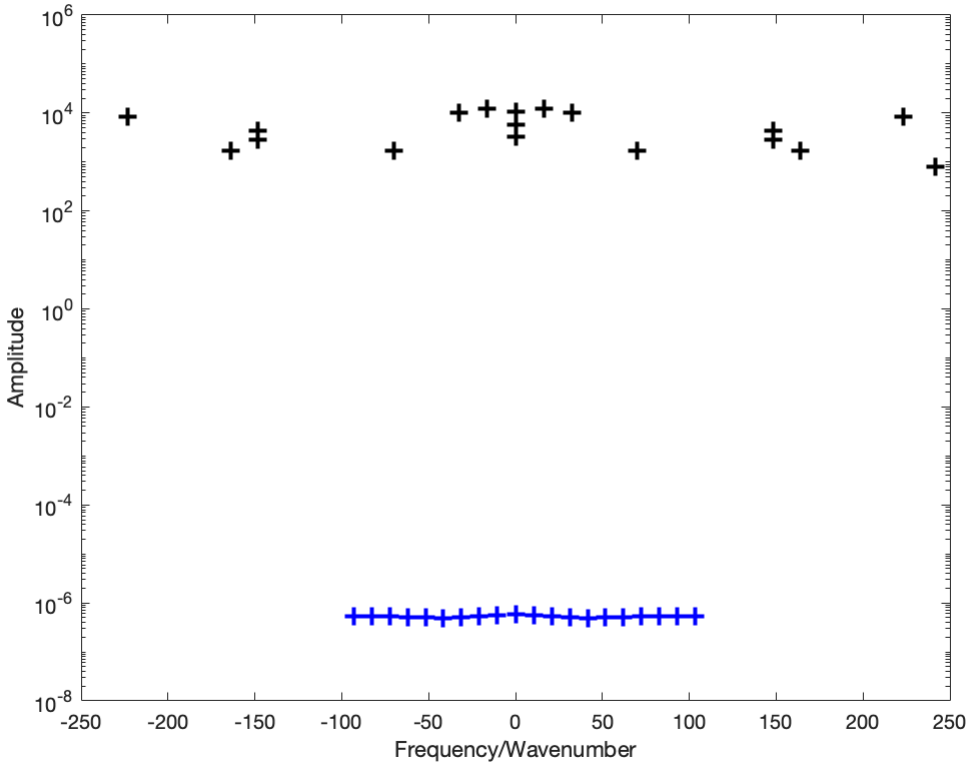
```
axes9 = axes('Parent',h9,'FontSize',14,'FontName','Agency FB');
box(axes9,'on');
pcolor(abs(Vreconst))
shading('interp')
colormap(jet)
xlabel('Date')
ylabel('Sentiment')
title("Reconstructed Data")
```




```
% Plot: frequency/wavenumber vs. amplitude, vs. growth rate
figure1 = figure;
axes1 = axes('Parent',figure1);
hold(axes1,'on');
box(axes1,'on');
semilogy(Frequencyt,GrowthRatet,'o','linewidth',2,'color','k','MarkerSize',8);
semilogy(Frequencyx,GrowthRatex,'o','linewidth',2,'color','b','MarkerSize',8);
set(axes1,'YMinorTick','on');
xlabel('Frequency/Wavenumber')
ylabel('Growth Rate')
```




```
figure1 = figure;
axes1 = axes('Parent',figure1);
hold(axes1,'on');
box(axes1,'on');
semilogy(Frequencyt,Amplitudest,'+','linewidth',2,'color','k','MarkerSize',8);
semilogy(Frequencyx,Amplitudesx,'+','linewidth',2,'color','b','MarkerSize',8);
set(axes1,'YMinorTick','on','YScale','log');
xlabel('Frequency/Wavenumber')
ylabel('Amplitude')
```




```
figure
```

Dogecoin prediction with News Sentiments

Similar to the Bitcoin approach, news was collected for the past 2 years - 1st January 2019 to 3rd June 2021.

```
frames=[]
delta = timedelta(days=1)
start_date = date(2019, 1, 1)
end_date = date(2021, 6, 3)
for i in tqdm(range(880)):
    s = start_date.strftime("%Y-%m-%d")
    e = (start_date+delta).strftime("%Y-%m-%d")
    response = datanews.news(q='Doge',from_date=s,to_date=e)
    articles = response['hits']
    d = pd.DataFrame(articles)
    frames.append(d)
    start_date+=delta
    if start_date>end_date:
      break
    time.sleep(1.5)
```

The full notebook can be found here : https://colab.research.google.com/drive/1YeivODqLAZtZiSBbuot7vV_T8ZkFzwEF?usp=sharing

```
clear all
clc
%Import data from text file
% Script for importing data from the following text file:
```

```
%
%    filename: /Volumes/Rohith/College/SpatioTemporalKoopman/scripts/Matrix.csv
%
% Auto-generated by MATLAB on 02-Jun-2021 23:38:47

% Set up the Import Options and import the data
opts = delimitedTextImportOptions("NumVariables", 304);

% Specify range and delimiter
opts.DataLines = [1, Inf];
opts.Delimiter = ",";

% Specify column names and types
opts.VariableNames = ["e00", "e00_1", "e00_2", "e00_3", "e00_4", "e00_5", "e00_6", "e00_7", "e0
opts.VariableTypes = ["double", "double", "double", "double", "double", "double", "double", "do

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% Import the data
Matrix = readtable("/Volumes/Rohith/College/SpatioTemporalKoopman/scripts/Matrix.csv", opts);

% Convert to output type
Matrix = table2array(Matrix);

% Clear temporary variables
clear opts

V = 100*Matrix;
s = size(V);
Time = linspace(0,1,s(2));
Exis = linspace(0,1,s(1));



% APPLY STKD TO V
[J,K]=size(V);
for i=1:J
    for j = 1:K
        if(isnan(V(i,j)))
           V(i,j) = 0;
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% STKD  PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SET PARAMETERS STKD
% Tolerances
% SVD
varepsilon1=1e-20;
% DMD
```

```
varepsilon2=1e-18;
% Set index d: dSpace for spatial analysis and dTime for temporal analysis
dSpace=25;
dTime=25;

[Vreconst,Modes,Amplitudes,Amplitudesx,GrowthRatex,Frequencyx,Amplitudest,GrowthRatet,Frequency
CalculateDMDdSdT(dSpace,dTime,Time,Exis,V,varepsilon1,varepsilon2);
```

```
Spatial Dimension: singular values
    50

Spatial dimension reduction
    26

size of a
ans = 1×2
    26     1
size of Q
ans = 1×2
    50    26
Spectral complexity
    26

Spatial dimension reduction
   280

size of a
ans = 1×2
   280     1
size of Q
ans = 1×2
    50   280
Spectral complexity
   280
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  4.209482e-17.
        7280

        7280

Number of spatial modes
    26

Number of temporal modes
   280
```

```
% CALCULATE RRMS ERROR
dif=Vreconst-V;
errorRMS=norm(dif(:))/norm(V(:));
disp('Reconstruction error: RRMSE')
```

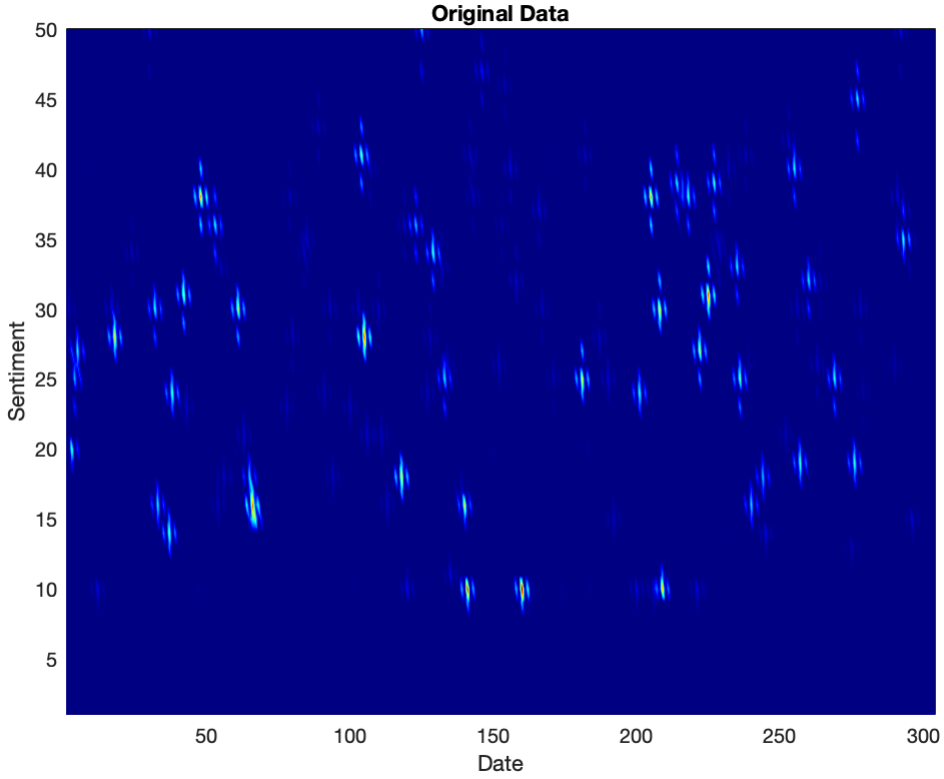
```
Reconstruction error: RRMSE
```

```
disp(errorRMS)
```

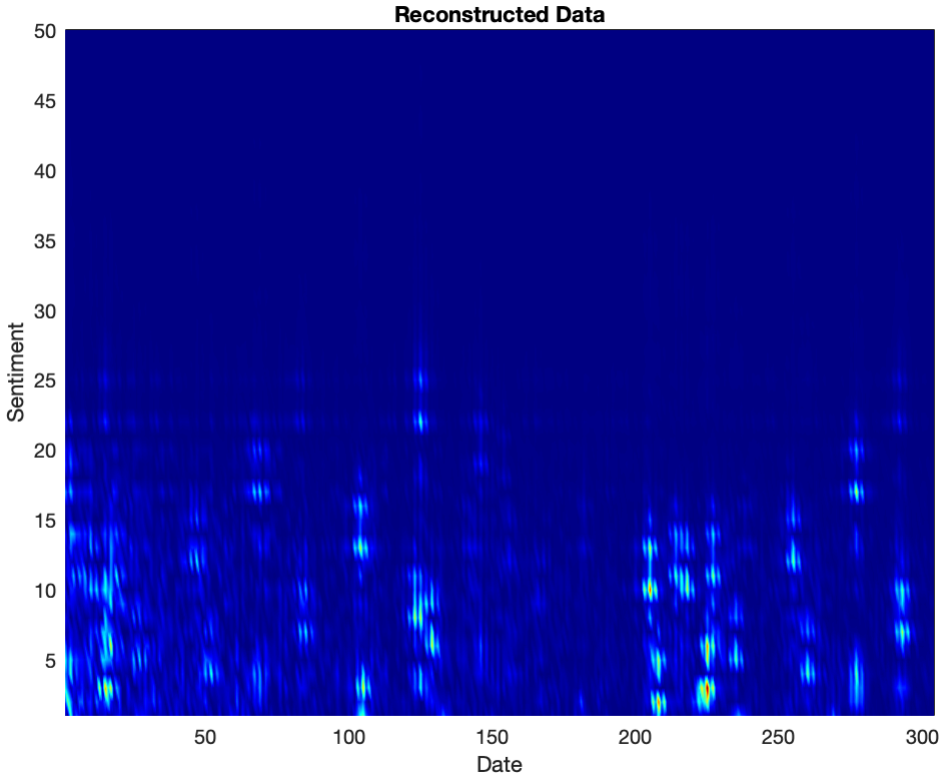
```
    0.9998
```

```
% PLOT RECONSTRUCTION
figure
pcolor(V)
shading('interp')
```

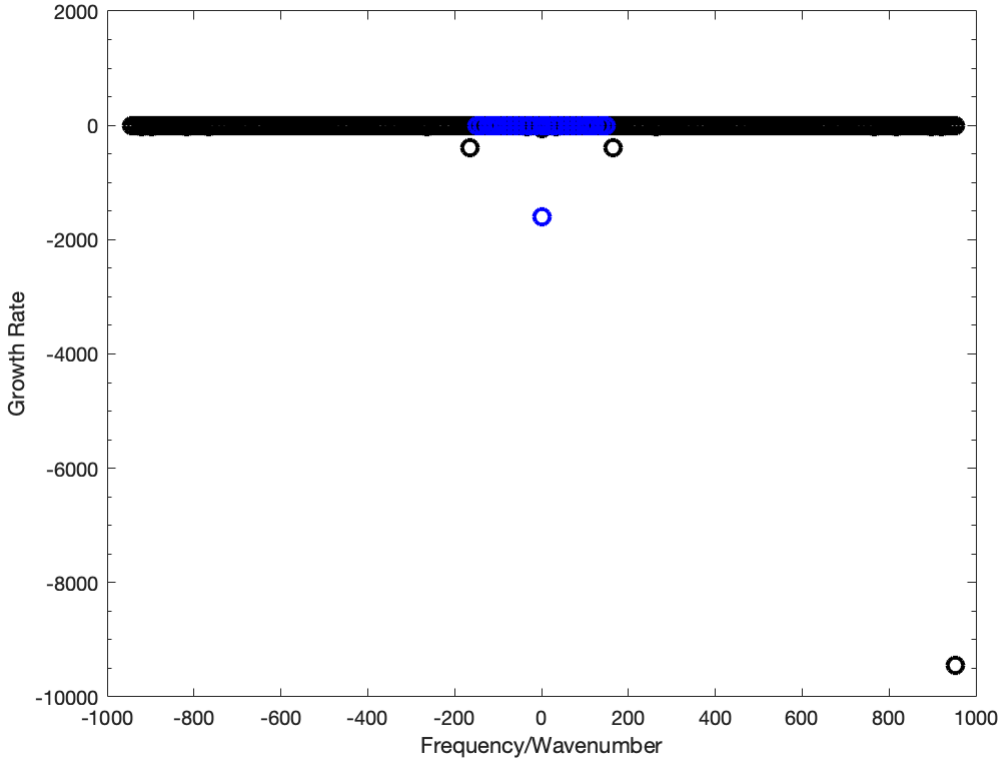
```
colormap(jet)
xlabel('Date')
ylabel('Sentiment')
title("Original Data")
```



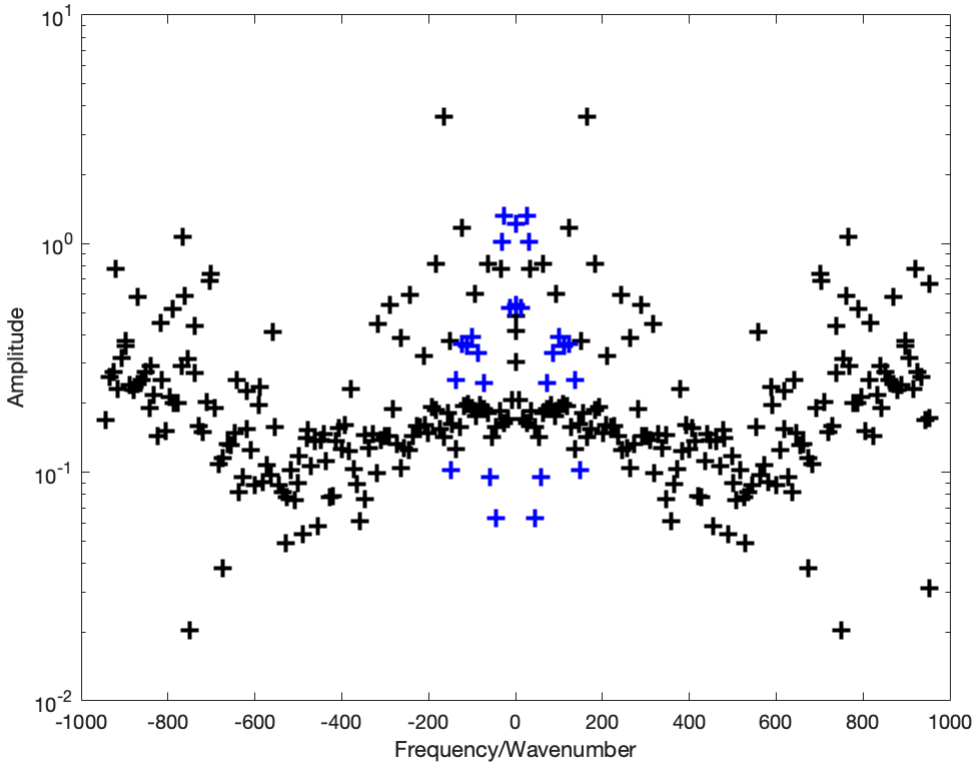

```
h9=figure;
axes9 = axes('Parent',h9,'FontSize',14,'FontName','Agency FB');
box(axes9,'on');
pcolor(abs(Vreconst))
shading('interp')
colormap(jet)
xlabel('Date')
ylabel('Sentiment')
title("Reconstructed Data")
```




```
% Plot: frequency/wavenumber vs. amplitude, vs. growth rate
figure1 = figure;
axes1 = axes('Parent',figure1);
hold(axes1,'on');
box(axes1,'on');
semilogy(Frequencyt,GrowthRatet,'o','linewidth',2,'color','k','MarkerSize',8);
semilogy(Frequencyx,GrowthRatex,'o','linewidth',2,'color','b','MarkerSize',8);
set(axes1,'YMinorTick','on');
xlabel('Frequency/Wavenumber')
ylabel('Growth Rate')
```




```
figure1 = figure;
axes1 = axes('Parent',figure1);
hold(axes1,'on');
box(axes1,'on');
semilogy(Frequencyt,Amplitudest,'+','linewidth',2,'color','k','MarkerSize',8);
semilogy(Frequencyx,Amplitudesx,'+','linewidth',2,'color','b','MarkerSize',8);
set(axes1,'YMinorTick','on','YScale','log');
xlabel('Frequency/Wavenumber')
ylabel('Amplitude')
```



Conclusion

From the above experiments we can see that the snapshots (Data Matrix) is very sparse and STKD is not very sucessful in reconstrcting the snapshot. A few reasons could be/scope of improvement:

- Incorrect Choice of Space Variable
- Use of improved Sentiment Analyser
- Hypertuning parameters of STKD
- Restriction in Data Collection
- More weightage to Tweets from celebrities (eg:Elon Musk's tweet would be more important than any unknown individual)

Despite the result, this experiment was conducted to analyse if sentiments from twitter and News could potentially aid in predicting Volatile Markets. These experiments would be a base for further improvement and analysis.

Reference

[1] Kearney C, Liu S. Textual sentiment in finance: a survey of methods and models. Int Rev Financ Anal. 2014;33:171–85.

[2] Jiao P, Veiga A, Walther A. Social media, news media and the stock market. J Econ Behav Organ. 2020;176:63–90.

[3] Arbabi H. Introduction to Koopman operator theory of dynamical systems. 2018.

Appendix

```
function  [Vreconst,deltas,omegas,amplitude,modes] =DMDd_SIADS(d,V,Time,varepsilon1,varepsilon)


%%%%%%%%%%%%%%%%%%%%%%%%%  DMD-d %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This function solves the HODMD algorithm presented in                %%%
%%% Le Clainche & Vega, SIAM J. on Appl. Dyn. Sys. 16(2):882-925, 2017  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% %% INPUT: %%
%%% d: parameter of DMD-d (higher order Koopman assumption)
%%% V: snapshot matrix
%%% Time: vector time
%%% varepsilon1: first tolerance (SVD)
%%% varepsilon: second tolerance (DMD-d modes)
%%% %% OUTPUT: %%
%%% Vreconst: reconstruction of the snapshot matrix V
%%% deltas: growht rate of DMD modes
%%% omegas: frequency of DMD modes(angular frequency)
%%% amplitude: amplitude of DMD modes
%%% modes: DMD modes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[J,K]=size(V);

% STEP 1: SVD of the original data

[U,Sigma,T]=svd(V,'econ');
sigmas=diag(Sigma);
n=length(sigmas);

NormS=norm(sigmas,2);
kk=0;
for k=1:n
    if norm(sigmas(k:n),2)/NormS>varepsilon1
        kk=kk+1;
    end
end

U=U(:,1:kk);

% Create reduced snapshots matrix
hatT=Sigma(1:kk,1:kk)*T(:,1:kk)';
[N,~]=size(hatT);

% Create the modified snapshot matrix
tildeT=zeros(d*N,K-d+1);
for ppp=1:d
```

```
    tildeT((ppp-1)*N+1:ppp*N,:)=hatT(:,ppp:ppp+K-d);
end

% Dimension reduction
[U1,Sigma1,T1]=svd(tildeT,'econ');
sigmas1=diag(Sigma1);

Deltat=Time(2)-Time(1);
n=length(sigmas1);

NormS=norm(sigmas1,2);
kk1=0;
for k=1:n
    RRMSEE(k)=norm(sigmas1(k:n),2)/NormS;
    if RRMSEE(k)>varepsilon1
        kk1=kk1+1;
    end
end


U1=U1(:,1:kk1);
hatT1=Sigma1(1:kk1,1:kk1)*T1(:,1:kk1)';

% Reduced modified snapshot matrix
[~,K1]=size(hatT1);
[tildeU1,tildeSigma,tildeU2]=svd(hatT1(:,1:K1-1),'econ');

% Reduced modified Koopman matrix
tildeR=hatT1(:,2:K1)*tildeU2*inv(tildeSigma)*tildeU1';
[tildeQ,tildeMM]=eig(tildeR);
eigenvalues=diag(tildeMM);

M=length(eigenvalues);
qq=log(eigenvalues);
deltas=real(qq)/Deltat;
omegas=imag(qq)/Deltat;

Q=U1*tildeQ;
Q=Q((d-1)*N+1:d*N,:);
[NN,MMM]=size(Q);

for m=1:MMM
    NormQ=Q(:,m);
    Q(:,m)= Q(:,m)/norm(NormQ(:),2);
end

% Calculate amplitudes
Mm=zeros(NN*K,M);
Bb=zeros(NN*K,1);
aa=eye(MMM);
for k=1:K
    Mm(1+(k-1)*NN:k*NN,:)=Q*aa;
    aa=aa*tildeMM;
    Bb(1+(k-1)*NN:k*NN,1)=hatT(:,k);
```

```
end

[Ur,Sigmar,Vr]=svd(Mm,'econ');
a=Vr*(Sigmar\(Ur'*Bb));

u=zeros(NN,M);
for m=1:M
    u(:,m)=a(m)*Q(:,m);
end
amplitude=zeros(M,1);

for m=1:M
    aca=U*u(:,m);
    amplitude(m)=norm(aca(:),2)/sqrt(J);
end

UU=[u;deltas';omegas';amplitude']';
UU1=sortrows(UU,-(NN+3));

UU=UU1';
u=UU(1:NN,:);
deltas=UU(NN+1,:);
omegas=UU(NN+2,:);
amplitude=UU(NN+3,:);
kk3=0;

for m=1:M
    if amplitude(m)/amplitude(1)>varepsilon
        kk3=kk3+1;
    else
    end
end

u=u(:,1:kk3);
deltas=deltas(1:kk3);
omegas=omegas(1:kk3);
amplitude=amplitude(1:kk3);

% Reconstruction of the original snapshot matrix
hatTreconst=zeros(N,K);
for k=1:K
    hatTreconst(:,k)= ContReconst_SIADS(Time(k),Time(1),u,deltas,omegas);
end

Vreconst=U*hatTreconst;

% Calculation of DMD modes
modes=zeros(J,kk3);
amplitude0=zeros(kk3,1);
for m=1:kk3
    NormMode=norm(U*u(:,m),2)/sqrt(J);
    amplitude0(m)=NormMode;
    modes(:,m)=U*u(:,m)/NormMode;
end
```

```
end
```