PROJECT REPORT

Financial Trading

Applied Mathematics
at Lahore University of Management Sciences

02^{nd} October 2021

Contents

1 — Acknowledgments

We would like to thank J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton and Joshua L. Proctor, authors of the book Dynamic Mode Decomposition for providing us with the algorithm and code on which we based the foundation for our research.

2 — DMD applied to Financial Data

Dynamic Mode Decomposition is built on the mathematical techniques Koopman theory which relies on mapping finite-dimensional nonlinear dynamical system to an infinite-dimensional linear system. It is a rigorous data driven and equation free modeling strategy. (i) providing a rigorous mathematical connection with dynamical systems theory, and (ii) adaptively modeling and controlling complex, nonlinear processes

2.1 Stabillity

Financial data is non linear process, while the stock market is assumed to be a complex, dynamical system. DMD decomposes stock portfolio data into low-rank features that behave with a prescribed temporal dynamics. In the process, the least-square fit linear dynamical system allows one to predict short-time future states of the system[1] We will be applying the following constraints and investigating how it impacts conditioning of the system

- Diversifying the Portfolio
- Varying the time window of the sample data

[1]Taken from Kurtz Dynamic Decomposition mode

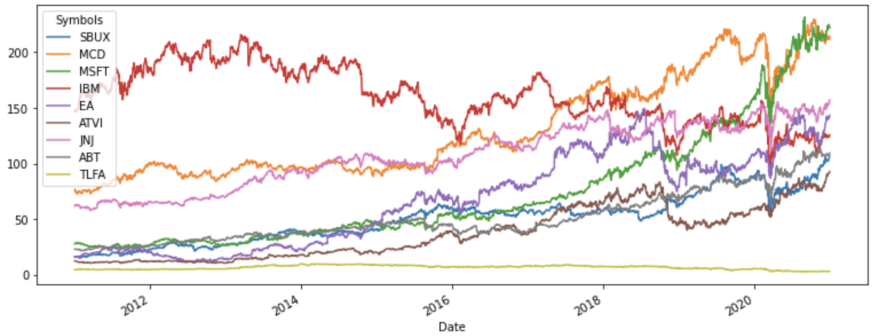


Figure 2.1: Time Series for nine stocks from the past decade
Time series demonstrates that Microsoft stock price has had the most exponential increase in past decade while the price of the leather stock has remained relatively the same.

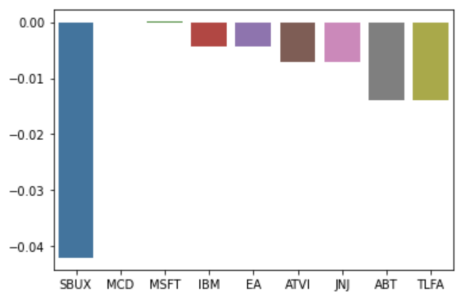


Figure 2.2: Eigenvalues

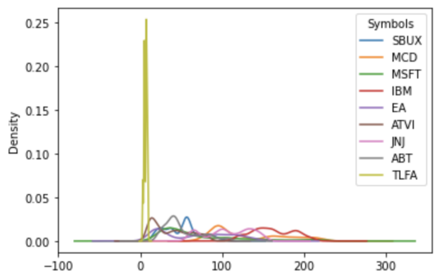


Figure 2.3: KDE

Above Figure shows the risk associated with Starbucks stock is the highest while the lowest is with McDonalds

2.1.1 Diversifying Portfolio

To diversify our portfolio we choose stocks from five industries including Gaming, leather,Technology, Healthcare and Food. In general the diversification led our matrix becoming highly dense and as a result ill - conditioned.

Industry Variation

After the diversification we tried grouping stocks from different sectors to see the impact on condition number of our matrix. On average after normalization the addition of two stocks regardless of the industry causes an increment of 25 to the

	Condition Number	**Normalized Condition Number**
Food, Tech	37.64	25.32
Food,Tech & Game	74.31	50.88
Food,Tech,Game,Health	656.31	80.88

Table 2.1: Impact on Condition Number on diversifying portfolio.

condition number.Non normalization causes an upward spike in the condition numbers

2.1.2 Varying the Time Window

Below are the results from a nine stock portfolio from five industries

	Condition Number	**Normalized Condition Number**
Two years	1026.312	119.737
Five years	665.246	85.050
Ten years	378.069	65.060

Table 2.2: Impact on Condition Number on increasing time frame.

As we increase the time window of the sample data, our problem becomes better conditioned and the matrix structure that is being built improves significantly and more effectively captures the structure for each stock

3 — Conclusion

3.1 Mathematics

3.1.1 Definitions

Definition 1. *A function $f : X \to Y$ is injective if and only if for all $x_1, x_2 \in X$, $x_1 \neq x_2$ implies $f(x_1) \neq f(x_2)$.*

3.1.2 Formulas

Formulas can be included inline, e.g. K_E^{CID} to S or as an own block.

$$otp = f_k(i, k), \qquad i = 1, ..., n$$

3.2 Code Listings

```
start = datetime.datetime(2011, 7, 30)
end = datetime.datetime(2021, 7, 30)
df_SBUX = web.DataReader("SBUX", 'yahoo', start, end)
```

Listing 3.1: Fetching the Data

Out[3]:

Date	High	Low	Open	Close	Volume	[illegible]
2011-07-29	20.445000	19.620001	19.945000	20.045000	31177400.0	16.906878
2011-08-01	20.385000	19.650000	20.219999	19.900000	16245600.0	16.784586
2011-08-02	19.830000	19.290001	19.705000	19.305000	16162600.0	16.282726
2011-08-03	19.660000	18.985001	19.305000	19.639999	14852600.0	16.565285
2011-08-04	19.580000	18.445000	19.375000	18.450001	20348400.0	15.561584

Figure 2.1

```
def cond_numb(start_date):
    df = pdr.DataReader(Stocks, data_source='yahoo', start =
        start_date, end='2021-01-01')['Close']
    X = df.T.values
    X_normalized = preprocessing.normalize(X, norm='l2')
    return "X:", la.cond(X) ,"Normalized X:",la.cond(X_normalized)
```

Listing 3.2: Condition Number

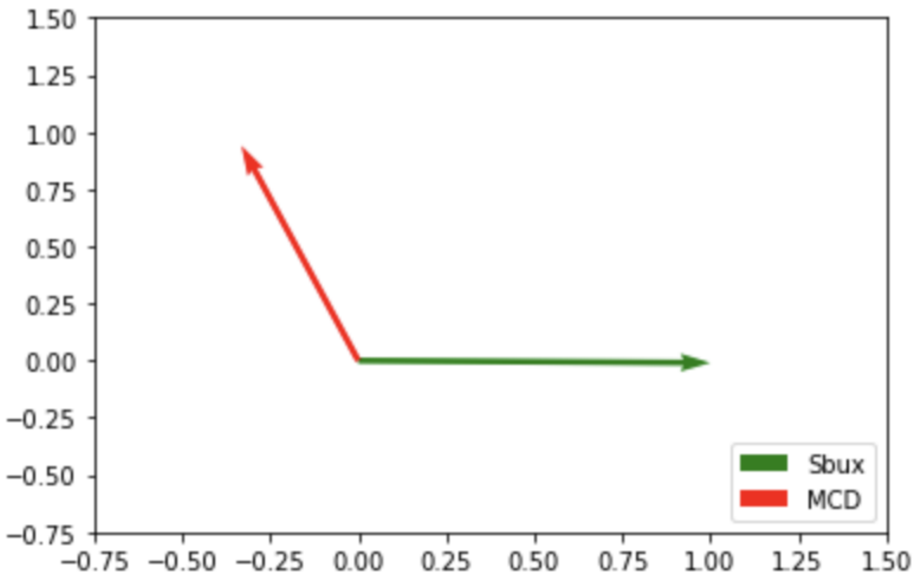
Bibliography

[1] Max Mustermann and Moritz Mustermann. *This is a cool website with funny cat pictures*. June 2016. URL: `https://example.org` (visited on 01/01/2016).

A — Appendix

A.0.1 Figures

Figure A.1: Normalized Eigenvectors multiplied by Eigenvalues of Two stocks



A.0.2 Flow Diagram

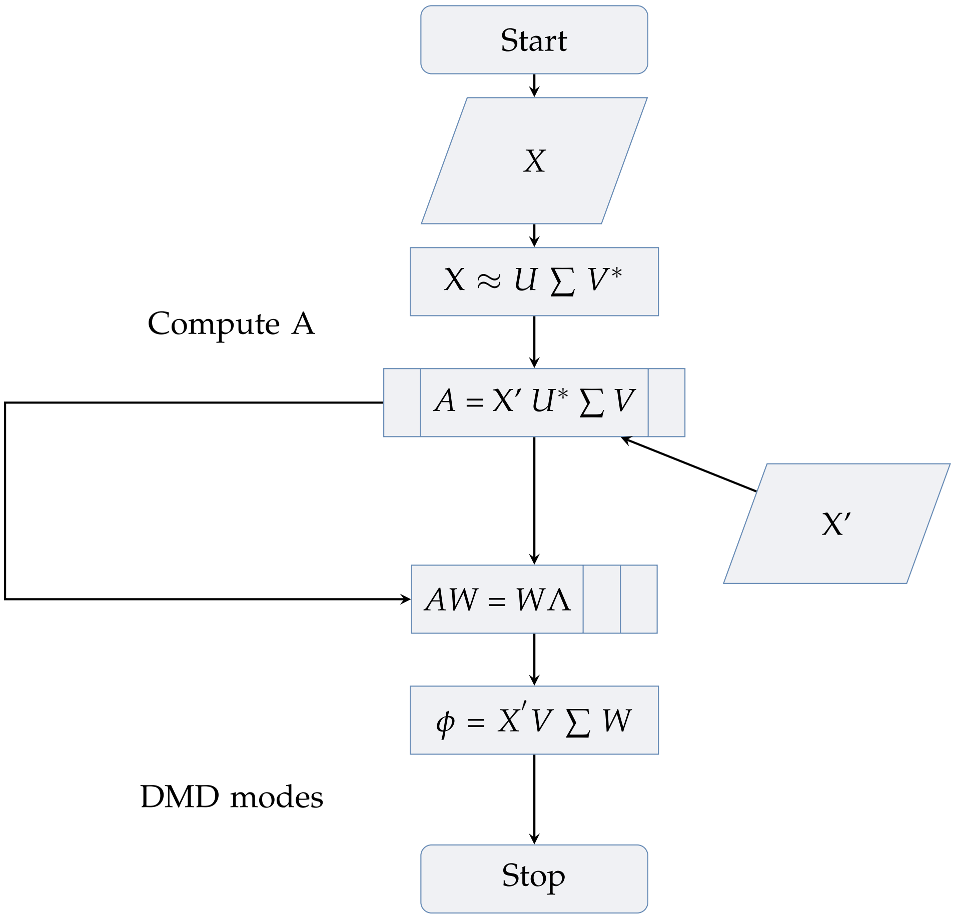


Figure A.2: Flow Diagram

Source: [1]

A.0.3 Algorithims

Listing A.1: computing DMD modes

```
X1 = np.delete(X, -1, 1)
X2 = np.delete(X, 0, 1)

#Normalizing X1 and X2
X1_normalized = preprocessing.normalize(X1, norm='l2')
X2_normalized = preprocessing.normalize(X2, norm='l2')

#Singular Value Decomposition
U, s, vh = np.linalg.svd(X1_normalized, full_matrices=True)
S = np.diag(s)

#Reshaping
UT = U.T
V = vh.T
arr = np.zeros((2,2515))
ST = np.concatenate((la.inv(S),arr), axis =1)

#Computing rank reduced A
Atilde = UT @ X2_normalized @ V @ ST.T

#Eigen Decomposition
eig_vals, eig_vecs = la.eig(Atilde)

#DMD modes
phi = X2_normalized @ V @ ST.T @ eig_vecs
```