IMPLEMENTATION OF KALMAN FILTER FOR PREDICTION OF STOCK MARKET

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Harsh Kumar Birthare
(2001EE20)

under the guidance of

Dr. Abhinoy Kumar Singh

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY PATNA
PATNA - 801103, BIHAR

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Implementation of Kalman Filter for Prediction of Stock Market**" is a bonafide work of **Harsh Kumar Birthare** (Roll No. **2001EE20**), carried out in the Department of Electrical Engineering, Indian Institute of Technology Patna under my supervision and that it has not been submitted elsewhere for a degree.*

[illegible]

Supervisor: **Dr. Abhinoy Kumar Singh**

Assistant/Associate Professor,

Department of Electrical Engineering,

Indian Institute of Technology Patna, Bihar.

May, 2024

Patna.

Acknowledgements

I would like to express my sincere gratitude to Professor Dr. Abhinoy Kumar Singh for giving me this project and for providing key insights into the theoretical portion required for the completion of this project without which the project would not have been possible.

Abstract

This thesis deals with the implementation of Kalman Filter to predict the stock price based on the historical data set of some of the companies. The Kalman Filter is a recursive optimal state estimator and hence is implemented in this thesis with Python 3.11.3 Programming Language. The stock prices are considered to be a maneuvering system and hence the formulation of state space model is done. Based on this Model we have developed a Python recursive loop to implement the model. The model is has been able to reduce the relative error percentage to around 0.5%. Hence the model is successfully implementing the Kalman Algorithm to predict the Apriori as well as the posteriori (i.e, the optimal) stock prices.

Keywords : Kalman Filter, State Space Model, Dynamic Systems, Recursive Loop, Prediction and Update Steps, Apriori Estimate, Optimal State.

Table of Contents

List of Symbols

Sr. No.	Symbol	Definition
1	x_n	State Vector at time index n
2	Ψ_n	State Transition Matrix
3	B_n	Control Input Matrix
4	u_n	Control Input Vector
5	μ_n	Process Noise
6	α_n	Measurement Vector
7	H_n	Measurement Matrix/ Observation Matrix
8	γ_n	Measurement Noise
9	x_o	Initial Condition
10	P_o	Initial Condition
11	$x_{n\|n-1}$	Predicted State at time index n given all measurements up to n-1
12	$P_{n\|n-1}$	Predicted Covariance Matrix
13	$P_{n-1\|n-1}$	Previous Estimated Covariance Matrix
14	Q_n	Process Noise Covariance Matrix
15	K_n	Kalman Gain
16	R_n	Measurement Noise Covariance Matrix
17	$x_{n\|n}$	Updated State Estimate
18	I	Identity Matrix
19	$x_{n-1\|n-1}$	Previous Estimated State

List of Figures

List of Tables

Chapter 1
Introduction

The stock market has always been popular, given all the uncertainties and fluctuations in this arena, the thrill and excitement to invest in the companies and achieve the maximum profit is always the most popular demand of any investor. The success rate for a profitable investment is subject to many constraints in this domain, let's recognise that the stocks are inherently unpredictable, anything can happen at any point of time. The stocks can boom or then can collapse without any prior warning. Thus successful investing in this market requires knowledge, risk management abilities, and a whole lot of research and experience.

Getting to a method to predict the stock values thus becomes a viable tool, an important asset for an investor to do well thought investments for getting the maximum profits on his investments. Algorithm based prediction has got attention of many companies due to increasing computational power of the modern microprocessors. Stock market prediction algorithms utilize various mathematical models, statistical techniques, and ML algorithms for the analysis of the historical data and identification of patterns that are probable toS indicate future price movements. These algorithms aim to forecast stock prices, to an extent the overall market behaviour, helping the investors in making well thought and calculated decisions minimising the risk. It is still recommended that the stock market is inherently unpredictable and none of the tool guarantee the absolute success of any method, hence it should be considered one of many factors before investing.

Various algorithm-based stock market prediction methods uses mathematical models, ML based algorithmic implementation, and statistics to forecast future stock prices. In domain of Machine

Learning, algorithms like Support Vector Machines (SVM), Random Forests, and Neural Networks can learn complex relationships between input features (such as past stock prices, trading volumes, and economic indicators) and stock price movements, making them powerful tools for prediction. Algorithms like Hidden Markov Models (HMM) and K-means clustering identify recurring patterns in stock price data, helping predict future price movements based on historical patterns. One such algorithm is the Kalman Algorithm, a powerful mathematical tool used for stock market prediction by estimating future stock prices based on observed historical data.

1.0.1 Objective of the Thesis

The fundamental aim of this thesis is to implement the Kalman Filter Algorithm for predicting the stock market. The stock market is an arena full of fluctuations and uncertainty and thus we consider in the hypothesis of the algorithm that the stock prices are viewed as a maneuvering system. Hence the Kalman Algorithm is implemented.

We want to analyse this algorithms performance in predicting the stock prices given the historical data set. We want to know if this algorithm is practical for real world applications in financial market where the investments are taken seriously and people want accurate predictions.

We want to generate output based on the historical data of certain companies listed in the stock exchange. These plots will then be analysed and we will quantify whether our algorithm is working fine or not.

1.0.2 Thesis Walk Through

Let's see what this thesis contains:

(i) This chapter provides the introduction to the Thesis, we see foundational reason and motivation for writing the thesis.

(ii) The Chapter 2 deals with the fundamental theory of Kalman Filters that is relevant to this thesis, where we discuss the prediction and update steps of the Filters in a given state space model.

(iii) In Chapter 3, we implement the Kalman Filter in Python 3.11.3 in Pycharm Environment, where we first propose the hypothesis of the algorithm, the dynamics system equations, the state space model, the lemma to derive necessary relations, and implements the code.

(iv) Then Chapter 4 deals with the output of the model based on the given historical stock data. We will be taking the stock data from Yahoo! Finance for the companies namely Apple Inc., Reliance Industries Limited, State Bank of India, and Bharti Airtel, finally we analyse the output for the accuracy of the model.

(v) Chapter 5 concludes the thesis mentioning some points worth noting at the completion of the Thesis.

Chapter 2
The Kalman Filter

2.0.1 Introduction

The Kalman Filter is a technique used in the control systems to obtain the optimal estimate of the states of any dynamic system when the measurements are subjected to noise. In 1950, Rudolf E. Kalman first developed this theory and since then it has gained popularity due to its versatile nature.

At its core it is a recursive algorithm, thus can be implemented with a recursive loop in python. An advantage of this Filter is that it is able to incorporate new measurement in an efficient method.

Hence, for dynamic systems Kalman Filter is a powerful tool which balances accuracy and efficiency making it suitable for various fields.

2.0.2 State Space Model

The dynamic state space model used in the Kalman Filter is described below as:

(i) State Transition Equation

$$x_n = \Psi_n x_{n-1} + B_n u_n + \mu_n$$

This equation describes how the state parameters are evolving over the time steps. Noting here that to get the state at time step **n**, all we need is the state at the previous time step **n-1**.

(ii) Measurement Equation

$$\alpha_n = H_n x_n + \gamma_n$$

This equation describes a formal relation between the state vector, the observation matrix H_n and the measurement error vector γ_n. It's important to mention here that the measurement vector α_n is not evolving over time steps, instead it is obtained for each time step separately.

The measurement vector α_n is not directly derived from the above equation which is a general inference anyone can get, instead it is obtained independently from the measurement sensor readings.

The above equation just establishes a formal relation between the model parameters with the observation vector.

2.0.3 Prediction Step

The prediction step is used calculate the Apriori estimated prediction of the state vector x_n at time step **n** from the state vector we have at time step **n-1**.

In this step we also predict the error covariance matrix P at time step **n** from the previous time step **n-1**.

(i) State Prediction:

$$x_{n|n-1} = \Psi_n x_{n-1|n-1} + B_n u_n$$

This step calculates the predicted state estimate at time time step **n** from the state estimate given at time step **n-1**. It is important to note here that this equation does not include the process noise vector μ like the state transition equation we saw previously. We generally do not include the error/noise part in the pure state estimate prediction, we handle it separately in the Covariance Prediction step, which we are going to see in the next step.

(ii) Covariance Prediction

In this step we calculate the predicted covariance at time step **n** by using the covariance matrix a previous step **n-1**.

$$P_{n|n-1} = \Psi_n P_{n-1|n-1} \Psi_n^T + Q$$

Now observe this equation, the P represent the predicted covariance matrix at time step **n** and is calculated from the covariance matrix at time step **n-1**, note here that his matrix is predicted and not directly estimated, it handles the error in the estimated state by directly including the matrix Q which is the covariance matrix of the process noise μ.

2.0.4 Update Step

(i) The Kalman Gain

$$K_n = P_{n|n-1} H_n^T (H_n P_{n|n-1} H_n^T + R_n)^{-1}$$

The Kalman Gain at time step **n** is calculated here. The Kalman Gain is the factor the Kalman Filter is known for. The Kalman Gain K_n is the equivalent of Controller Gain K in the deterministic control systems, only that the Kalman Controller Gain K_n is used for Stochastic Systems.

The Kalman Gain can be thought of as the factor which determines which parameter is to be trusted more, the estimated or the observed. The factor favours the parameter which has minimum error in its values. If observation error γ is lower than the process error/noise μ, then the Kalman Gain favours observation covariance matrix R and if the error/process noise μ is lower than the measurement noise, then Kalman Gain favours the process noise variance matrix Q. Hence which ever parameter has lower error associated with it contributes more to the estimated state.

Hence the Kalman Filter estimate is the optimal state estimate.

(ii) The State Update

$$x_{n|n} = x_{n|n-1} + K_n(\alpha_n - H_n x_{n|n-1})$$

This equation updates the Apriori estimation of the state vector at time step **n** by optimising this estimation by incorporating the Kalman Gain and the Residual.

(ii) Covariance Update

$$P_{n|n} = (I - K_n H_n) P_{n|n-1}$$

Here the Apriori covariance matrix is getting updated resulting in the posteriori covariance matrix. I is the identity matrix.

2.0.5 Conclusive Note

Here we conclude that the Kalman Filter is really efficient approach for optimal state estimation and the Apriori estimates can predict without need to incorporate the current measurement. The posteriori estimation gives optimal state estimate. Thus this model can be used to predict stock prices where the measurements are obtained from the historical stock data and thus the estimates can be evaluated.

Chapter 3

Implementation in Python

3.0.1 The Algorithm Hypothesis

The stock market is known for its uncertainty and high fluctuations in the equities and various other parameters (i.e. volatility, etc.) along with the time varying nature. This problem is suitable with dynamic real time tracking problem characteristics of the Kalman Filter. In 2015, in the research paper (given in references section), they first introduced this hypothesis of viewing the stock prices as the maneuvering system. What is a maneuvering system? As we know that a physical object is governed by the classical mechanics laws, i.e., the laws of motion, so when an object is accelerated with acceleration a, the motion of this object is governed by three fundamental equations ($v = u + at$, $x = x_o + ut + 1/2at^2$). This analogy of the stock market's dynamic nature and the Kalman Filters real time tracking ability, when viewed with stock markets fluctuations by taking the stock prices as the positional analogue, the velocity of the object with the velocity of stock prices(i.e. the change in the stock prices per unit time steps) and the acceleration can be viewed analogues to the stock prices changing factor.

The acceleration of the stock in our case we will be taking as the white noise. White noise is random signal having constant power spectral density due to having equal intensity at different frequencies. Thus applying the acceleration to the stock prices, we can predict the stock values. The acceleration is white noise, due to the unpredictable nature of the stock value. Thus Kalman Filter with this hypothesis should be able to predict the stock values efficiently.

3.0.2 The Dynamic System

Let's consider the dynamic system:

$$X(t+1) = \Phi \cdot X(t) + \Gamma \cdot w(t) \tag{3.1}$$

$$y(t) = H \cdot X(t) + v(t) \tag{3.2}$$

Here the state transition matrix is Φ, which we took as Ψ in Chapter 2, measurement matrix is H, $w(t)$ is the model noise, the state vector is $X(t+1)$, the measurement vector is $y(t)$ (which was taken as α), the measurement noise is $v(t)$, note here that $w(t)$ and $v(t)$ are two separate white noises with zero mean and variance matrices Q and R respectively.

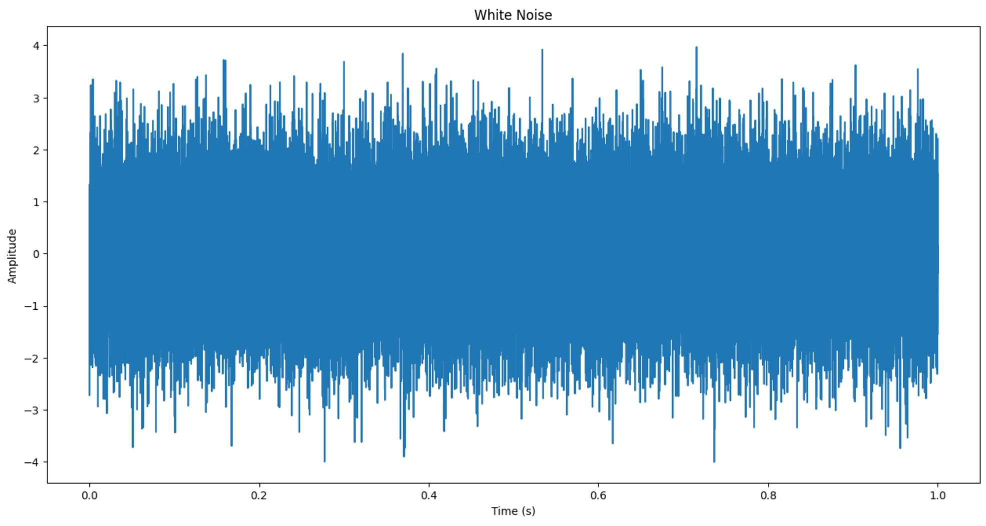

Fig 3.1 White Noise Signal

This dynamic system allows us to formulate the state space model of stock market. The general equation of the Kalman Filter uses the input control matrix to map the impact of input on the overall changes in the state variables of the model.

3.0.3 The State Space Model

As we described earlier in the hypothesis, we are viewing the stock market prices as the maneuvering system due to uncertainty and fluctuations in the stock market.

Using the analogy with the mechanical model of an object under acceleration, the maneuvering equations for the Stock Prices can be formulated as:

$$x(t+1\,|\,t+1) = x(t) + T \cdot \dot{x}(t) + \frac{1}{2}a(t)T^2 \tag{3.3}$$

$$\dot{x}(t+1) = \dot{x}(t) + T \cdot a(t) \tag{3.4}$$

$$y(t) = x(t) + v(t) \tag{3.5}$$

$$\Gamma = \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} \tag{3.6}$$

$$\Phi = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \tag{3.7}$$

$$w(t) = \alpha(t) \tag{3.8}$$

$$X(t) = \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix} \tag{3.9}$$

$$H = [1 \quad 0] \tag{3.10}$$

Thus system can be modelled into equation (3.1) and (3.2) based on equation (3.3) to (10).

We have all the required four matrices for our implementation. Now we need to derive the recursive relations for us to be able to implement this model in python. This is discussed in the next section.

3.0.4 Formulating the Lemma For Recursive Loop

Now we have the lemma derived from the eq (1) and eq (2), which is the foundation of our algorithm implementation.

$$\hat{X}(t+1|t+1) = \hat{X}(t+1|t) + K(t+1)\cdot\varepsilon(t+1) \tag{3.11}$$

$$\hat{X}(t+1|t+1) = \Phi\cdot\hat{X}(t|t) \tag{3.12}$$

$$\varepsilon(t+1) = y(t+1) - H\cdot\hat{X}(t+1|t) \tag{3.13}$$

$$K(t+1) = P(t+1|t)\cdot H^T\cdot[H\cdot P(t+1|t)\cdot H^T + R]^{-1} \tag{3.14}$$

$$P(t+1|t) = \Phi\cdot P(t|t)\cdot\Phi^T + \Gamma\cdot Q\cdot\Gamma^T \tag{3.15}$$

$$P(t+1|t+1) = [I - K(t+1)\cdot H]\cdot P(t+1|t) \tag{3.16}$$

The initial values are given as:

$$\hat{X}(0|0) = E\cdot\hat{X}(0) = \mu_0 \tag{3.17}$$

$$P(0|0) = E[(X(0)-\mu_0)(X(0)-\mu_0)^T] = P_0 \tag{3.18}$$

The equation (3.1)~(3.18) are as per Reference [R.1].

The equations given above entails that process is recursive. The Kalman Filter can handle recursive approach. Thus we conclude that these formulae can be implemented in Python with a recursion loop.

3.0.5 Training Data

Before we move on to the approach we will follow for developing the python code, first let us discuss the nature and dynamics of the training data used for training our python model. It is important to understand it here so that during the formation of algorithm we can parallel understand the processing of each training sample.

So, the data we are using is from Yahoo Finance. Yahoo Finance is the platform which used for various financial purposes, it gives news on finance, data, and tools for personal finance management. One of its domain is tracking the stock market prices. It archives the historical stock data of various listed companies in the stock exchange authorities. From this platform we are taking the historical data of the opening, and closing prices of the stock of few companies for analysis purposes. Exact training samples are discussed in the further section.

3.0.6 Approaching the Problem

So now the problem that we arises before us, what methodology is most suited for this state space model to be implemented in python? So the problem can be solved by handling it in three phases. We are using Pycharm Software for developing our model. So first we need to understand all the important python libraries which are going to be used in our model. These libraries are used multiple times in our code. The main purpose of the libraries is to enhance the computation complexity of the code and the minimise the time complexity. The libraries are names:

(i) NumPy: This library provides us the ability to perform complex scientific computations with an object oriented approach. It is used to calculate the array operations here.

(ii) Matplot: At its core it's a library used for visualisations like graphing, animating and creating static plots. The output in will be obtained by the dot pyploter.

(iii) Pandas: It processes data set. The historical data sets are going to be processed with this libray.

3.0.7 The Python Implementation

In our case, we are using Python 3.11.3 for executing our code. All the libraries use are already discussed in previous section.

Hence the final code is as follows:

```
import matplotlib.pyplot as ploter
import numpy as npy
import pandas as pds

read_data = pds.read_csv('BHARTIARTL.NS.csv')
observed_prices = read_data['Close'].values

Sample_time = 1
num_of_obv = len(observed_prices)
Q = npy.array([[0.01, 0], [0, 0.01]])
R = 0.57 ** 3
Φ = npy.array([[1, Sample_time], [0, 1]])
Γ = npy.array([[0.5 * Sample_time ** 2], [Sample_time]]).T
H = npy.array([[1, 0]])
Est_P = npy.eye(2)
white_noise = npy.random.randn(num_of_obv)
X = npy.zeros((2, num_of_obv))
Est_X = npy.zeros((2, num_of_obv))


for time_step in range(1, num_of_obv):
    Est_P = npy.dot(npy.dot(Φ, Est_P), Φ.T) + Q
    Est_X[:, time_step] = npy.dot(Φ, Est_X[:, time_step - 1]) + npy.dot(Γ, white_noise[time_step])

    Kalman_Gain = npy.dot(npy.dot(Est_P, H.T), npy.linalg.inv(npy.dot(npy.dot(H, Est_P), H.T) + R))

    Est_P = npy.dot((npy.eye(2) - npy.dot(Kalman_Gain, H)), Est_P)
    epsilon_residual = observed_prices[time_step] - npy.dot(H, Est_X[:, time_step])
    Est_X[:, time_step] = Est_X[:, time_step] + npy.dot(Kalman_Gain, epsilon_residual)
```

```
ploter.figure(figsize=(15, 7))
ploter.xlabel('Time Step of Each Closing Price')
ploter.ylabel('Stock Price')
ploter.plot(observed_prices, label='Actual Historical Stock Prices', color='blue', alpha=0.7,
marker='o', markersize=4)
ploter.plot(Est_X[0], label='Kalman Filter Estimate', color='red', alpha=0.7, marker='o',
markersize=4)

ploter.title('Kalman Filter Estimate vs Actual Historical Stock Prices')


for iter in range(num_of_obv):
    ploter.annotate(f'{Est_X[0, iter]:.2f}', (iter, Est_X[0, iter]), textcoords="offset points",
    xytext=(-10,11), ha='center', fontsize=7)

    ploter.annotate(f'{observed_prices[iter]:.2f}', (iter, observed_prices[iter]),
    textcoords="offset points", xytext=(-10,-25), ha='center', fontsize=7)


ploter.legend()
ploter.grid(True)
ploter.show()
```

Let's look and understand the code line by one:

1] import matplotlib.pyplot as ploter

2] import numpy as npy

3] import pandas as pds

```
4] read_data = pds.read_csv('AAPL.csv')

5] observed_prices = read_data['Close'].values

6] Sample_time = 1

7] num_of_obv = len(observed_prices)

8] Q = npy.array([[0.01, 0], [0, 0.01]])

9] R = 0.57 ** 3

10] Φ = npy.array([[1, Sample_time], [0, 1]])

11] Γ = npy.array([[0.5 * Sample_time ** 2], [Sample_time]]).T

12] H = npy.array([[1, 0]])

13] Est_P = npy.eye(2)

14] white_noise = npy.random.randn(num_of_obv)

15] X = npy.zeros((2, num_of_obv))

16] Est_X = npy.zeros((2, num_of_obv))


17] for time_step in range(1, num_of_obv):

18]     Est_P = npy.dot(npy.dot(Φ, Est_P), Φ.T) + Q

19]     Est_X[:, time_step] = npy.dot(Φ, Est_X[:, time_step - 1]) + npy.dot(Γ,
white_noise[time_step])

20]     Kalman_Gain = npy.dot(npy.dot(Est_P, H.T),
npy.linalg.inv(npy.dot(npy.dot(H, Est_P), H.T) + R))

21]     Est_P = npy.dot((npy.eye(2) - npy.dot(Kalman_Gain, H)), Est_P)

22]     epsilon_residual = observed_prices[time_step] - npy.dot(H, Est_X[:,
time_step])
```

```
23]     Est_X[:, time_step] = Est_X[:, time_step] + npy.dot(Kalman_Gain,
epsilon_residual)


24] ploter.figure(figsize=(15, 7))

25] ploter.xlabel('Time Step of Each Closing Price')

26] ploter.ylabel('Stock Price')

27] ploter.plot(observed_prices, label='Actual Historical Stock Prices',
color='blue', alpha=0.7,

    marker='o', markersize=4)

28] ploter.plot(Est_X[0], label='Kalman Filter Estimate', color='red', alpha=0.7,
marker='o',

markersize=4)

29] ploter.title('Kalman Filter Estimate vs Actual Historical Stock Prices')

30] for iter in range(num_of_obv):

31]     ploter.annotate(f'{Est_X[0, iter]:.2f}', (iter, Est_X[0, iter]),
textcoords="offset points",

        xytext=(-10,11), ha='center', fontsize=7)

32]     ploter.annotate(f'{observed_prices[iter]:.2f}', (iter,
observed_prices[iter]),

        textcoords="offset points", xytext=(-10,-25), ha='center', fontsize=7)

33] ploter.legend()

34] ploter.grid(True)

35] ploter.show()
```

Explanation:

1, 2, 3] So the code begins with importing the required libraries into the virtual environment. These libraries and their functions are already discussion in section 3.0.5.

4] We are importing the historical stock data of one of the companies (in the line 4 it is AAPL that is the stock imprint of Apple Inc.) into the virtual environment.

5] In this line we read and store all the closing prices from the starting data of the data.

6] Here we are defining the sampling time of reading. So in the historical data, there are some inherent gaps between some consecutive dates, we are taking them to be consecutive as it will not affect our output in any form.

7] Here number of stock data that we are supplying his recorded which is same as length of the CSV file.

8] Here covariance matrix of the process noise Q is defined.

9] Here the covariance matrix of the measurement noise R is defined.

10] This line defines the state transition matrix in a 2x2 matrix as describe in the hypothesis (eq. 3.7), defining the evolution of states over time.

11] Γ is the input control matrix modelling the control inputs on the transition of states in a 2x1 matrix (eq. 3.6).

12] H, the obervation matrix is defined here, 1x2 matrix.

13] The estimate of covariance matrix P is initialised here.

14] White noise is generated here using the NumPy radn function with zero mean and as decribed in 3.0.2.

15] This initialize the state vector X in a 2x"no_of_obv" matrix.

16] This line initialize the state estimate of X.

17] This line begins the recursive Kalman Filter.

18 and 19] These lines implement the prediction step.

20] This calculates the Kalman Gain.

21, 22, and 23] This is the Update step, estimate updation takes place based on current measurement.

24 to 35] This section of the code is used to plot using matplot library with appropriate adjustments.

3.0.8 A Conclusive Note on the Code

By this time we have successfully implemented Kalman Filter with Python. This model is ready to be trained and operated on the historical stock data. The code runs perfectly on the system and all the further execution is done on Python 3.11.3.

Chapter 4
Output & Analysis

4.0.1 The Historical Stock Data

The stock data we are using is from 'Yahoo! Finance'. For analysis purpose, we are using the historical stock data of Apple Inc., Reliance Industries, and State Bank of India. One of data subset of data for Apple Inc. used for generating the output is given below:

```
Date,Open,High,Low,Close,Adj Close,Volume

2024-03-01,179.550003,180.529999,177.380005,179.660004,179.660004,73488000

2024-03-04,176.149994,176.899994,173.789993,175.100006,175.100006,81510100

2024-03-05,170.759995,172.039993,169.619995,170.119995,170.119995,95132400

2024-03-06,171.059998,171.240005,168.679993,169.119995,169.119995,68587700

2024-03-07,169.149994,170.729996,168.490005,169.000000,169.000000,71765100

2024-03-08,169.000000,173.699997,168.940002,170.729996,170.729996,76114600

2024-03-11,172.940002,174.380005,172.050003,172.750000,172.750000,60139500

2024-03-12,173.149994,174.029999,171.009995,173.229996,173.229996,59825400

2024-03-13,172.770004,173.190002,170.759995,171.130005,171.130005,52488700

2024-03-14,172.910004,174.309998,172.050003,173.000000,173.000000,72913500
```

The entire datasheet used for the analysis is given in the appendix.

4.0.2 The Generated Output

The output plots for discussed companies are given:

(i) AAPL (Apple Inc.)

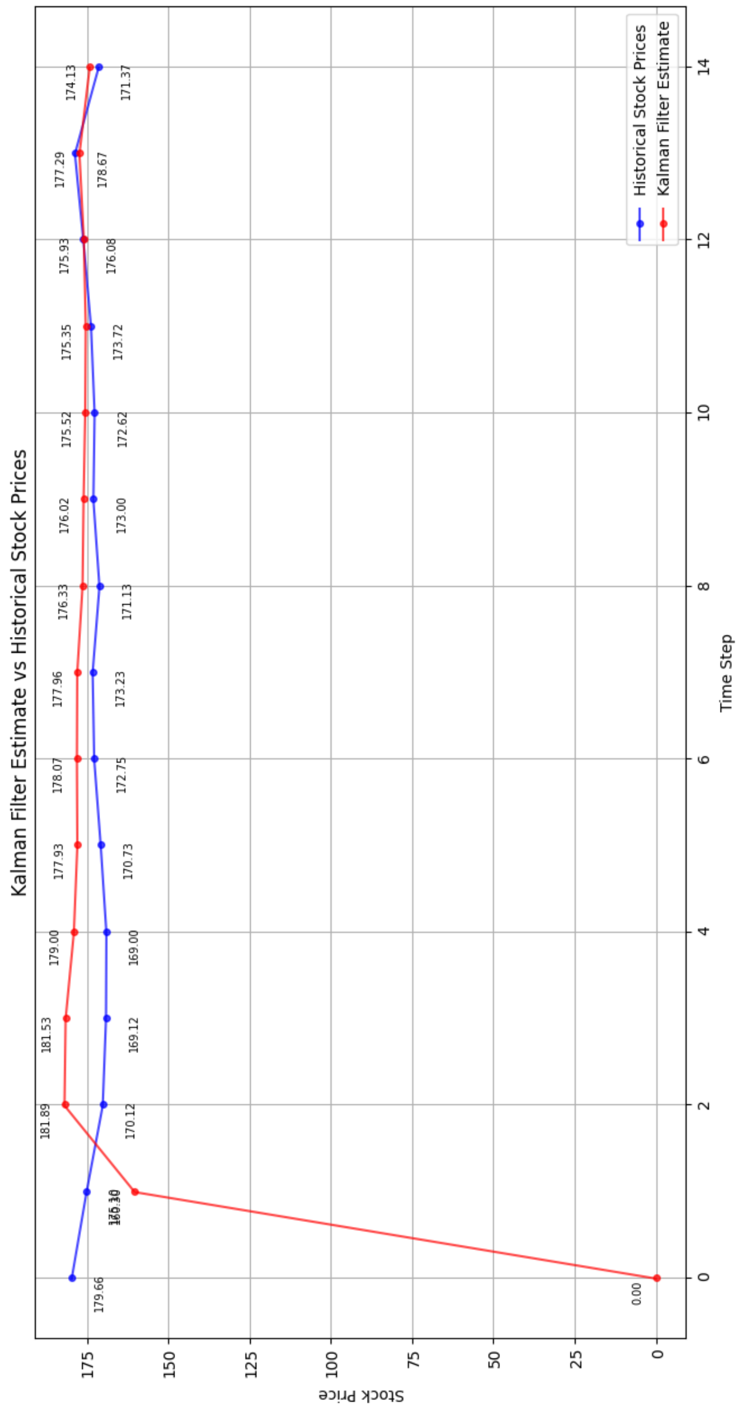

Fig 4.1

(ii) State Bank of India (SBI)

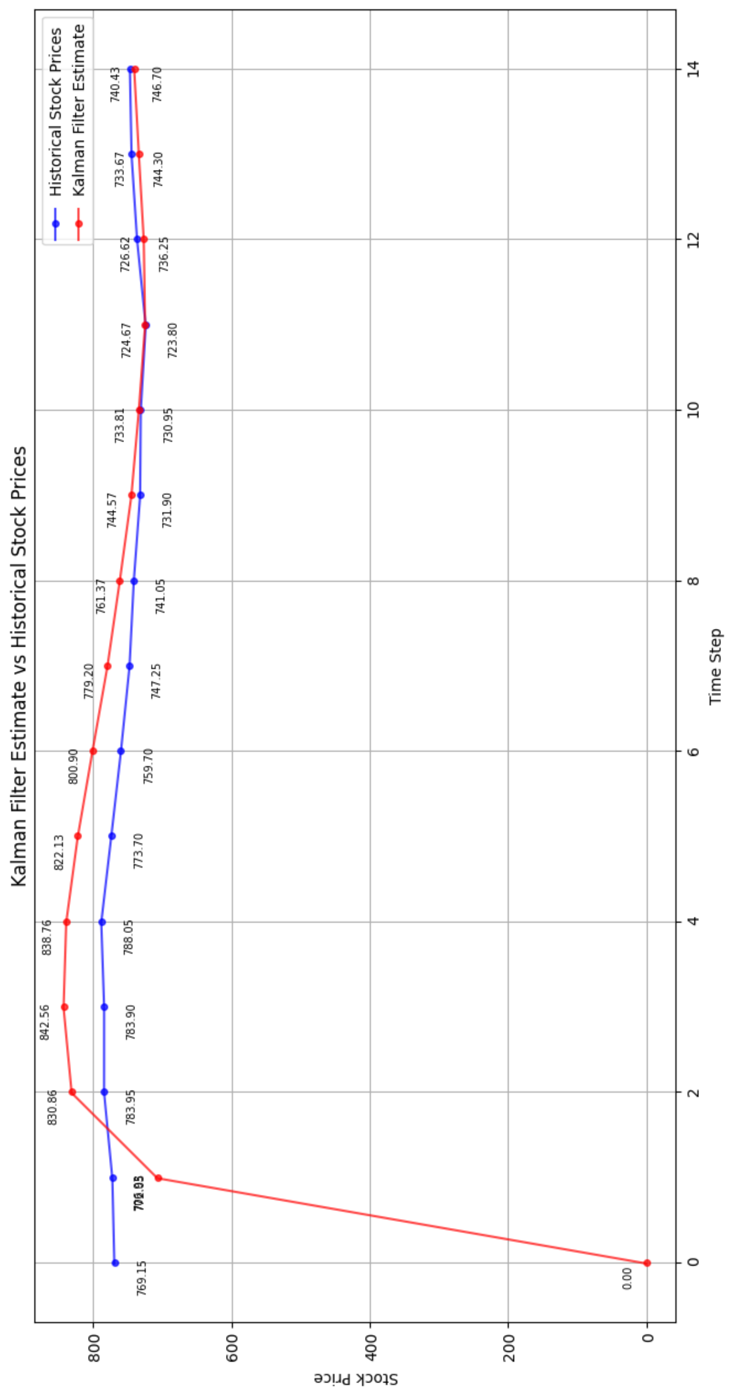

Fig 4.2

(iii) BHARTIAIRTL.NS (Airtel)

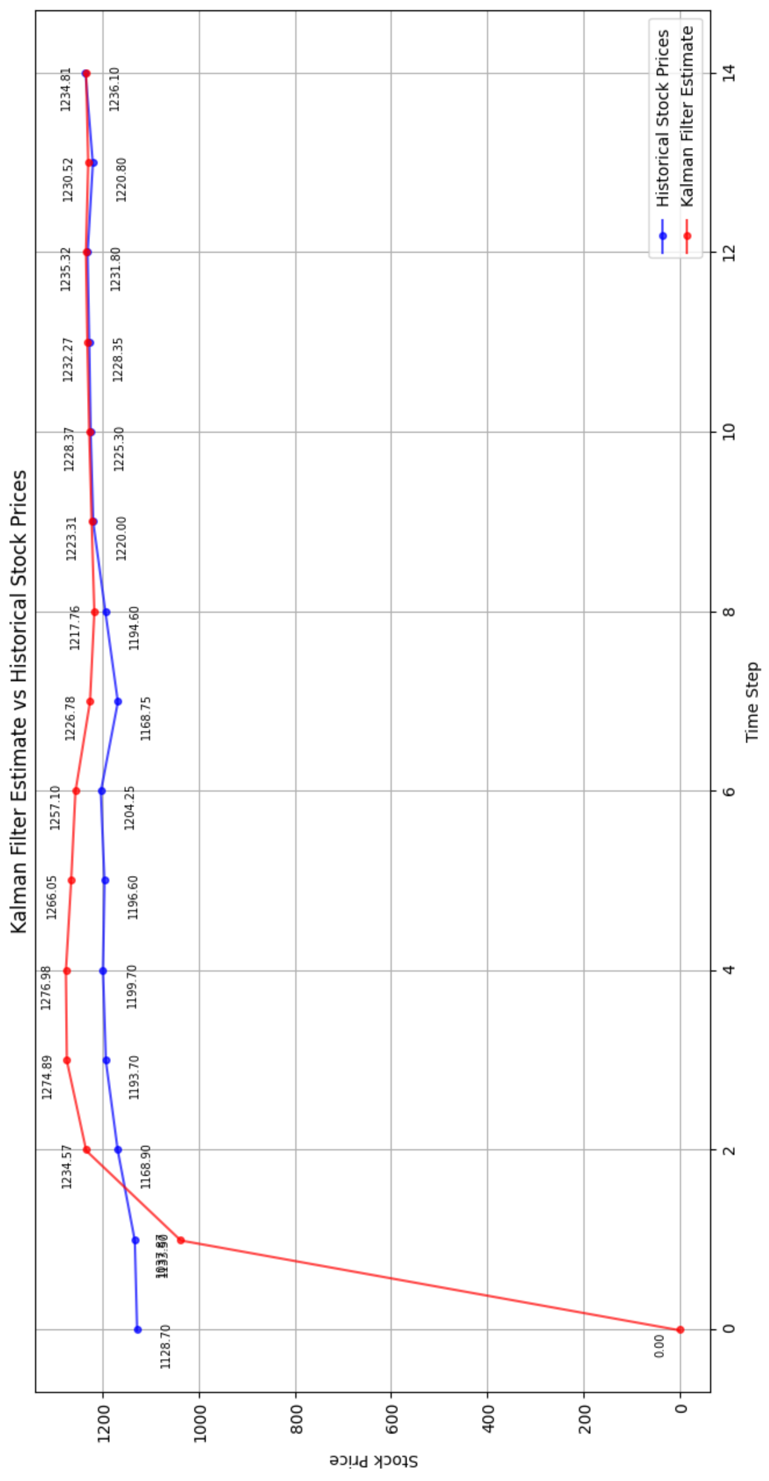

Fig 4.3

(iv) RELIANCE.NS (Reliance Industries Limited)

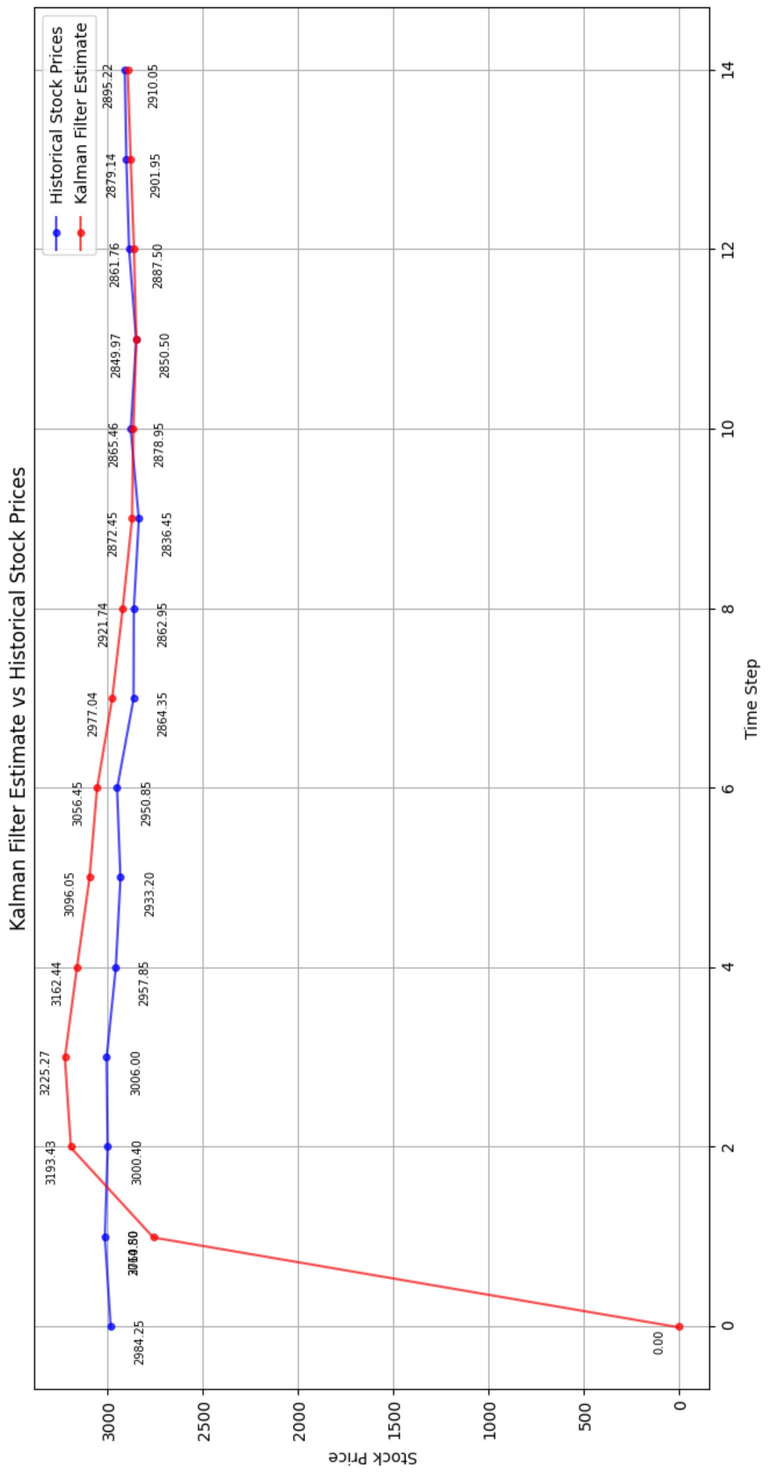

Fig 4.4

4.0.3 Analysis of the Output

(i) For Apple Inc.

Time Step	Observed Price (USD)	Kalman Filter Estimate (USD)	Absolute Error	Relative Error (%)
0	179.66	0	Not Applicable	Not Applicable
1	175.10	160.30	14.80	8.45
2	170.12	181.89	11.77	6.92
3	169.12	181.53	12.41	7.34
4	169.00	178.99	9.99	5.91
5	170.73	177.93	7.20	4.22
6	172.75	178.06	5.32	3.08
7	173.23	177.96	4.73	2.73
8	171.13	176.33	5.20	3.04
9	173.00	176.02	3.02	1.75
10	172.62	175.52	2.90	1.68

Table 4.1 AAPL (Apple Inc.)

(ii) For State Bank of India

Time Step	Observed Price (INR)	Kalman Filter Estimate (INR)	Absolute Error	Relative Error (%)
0	769.15	0	Not Applicable	Not Applicable
1	772.05	706.87	65.18	8.44
2	783.95	830.49	46.54	5.94
3	783.90	841.81	57.91	7.39
4	788.05	837.74	49.69	6.30
5	773.70	821.23	47.53	6.14
6	759.70	800.43	40.73	5.36
7	747.25	778.81	31.57	4.23
8	741.05	760.87	19.82	2.67
9	731.90	744.50	12.60	1.72
10	730.95	734.43	3.47	0.47

Table 4.2 SBIN.NS (State Bank of India)

(iii) For Airtel

Time Step	Observed Price (INR)	Kalman Filter Estimate (INR)	Absolute Error	Relative Error (%)
0	1128.70	0	Not Applicable	Not Applicable
1	1133.50	1037.87	95.63	8.44
2	1168.90	1234.57	65.67	5.62
3	1193.70	1274.89	41.19	3.45
4	1199.70	1276.98	77.28	6.44
5	1196.60	1266.04	69.44	5.80
6	1204.25	1257.10	52.85	4.39
7	1168.75	1226.78	58.03	4.97
8	1194.60	1217.76	23.16	1.94
9	1220.00	1223.31	3.31	0.27
10	1225.30	1228.37	3.07	0.25

Table 4.3 BHARTIARTL.NS (Bharti Airtel Limited)

(iii) For Reliance Industries

Time Step	Observed Price (INR)	Kalman Filter Estimate (INR)	Absolute Error	Relative Error (%)
0	2984.25	0	Not Applicable	Not Applicable
1	3014.80	2760.45	65.18	8.44
2	3000.40	3193.44	46.64	5.95
3	3006.00	3225.72	57.91	7.39
4	2957.85	3162.97	49.69	6.30
5	2933.20	3095.97	47.54	6.14
6	2950.85	3055.99	40.73	5.36
7	2864.35	2976.33	31.57	4.23
8	2862.95	2921.15	19.81	2.67
9	2836.45	2871.34	12.60	1.72
10	2878.95	2865.43	3.47	0.47

Table 4.4 RELIANCE.NS (Reliance Industries Limited)

4.0.4 Conclusive Note

Till now we have successfully tested our model with the actual historical data set. The model's accuracy is reflected for the analysis. Thus we can state that our model is working properly and is good for other datasets for any other organization. Hence, we have implemented our hypothesis successfully with Kalman Filter in Python.

Chapter 5
General Conclusions

1. Thus in Chapter 4, we has successfully implemented our algorithm hypothesis with Python 3.11.3. Given any historical stock data for any companies, we can input it into our model to predict the stock prices and the efficiency of this discussed already in the Chapter 5.
2. Our model can handle the large amount of data sets perfectly given the complexity of the code is reasonable and thus computational complexity with growing input data is not very high.
3. We can infer from the outputs generated that the the models begin with an error % of around 8.5% and is able to reduce it in subsequent predictions. We have taken 10 samples from the data set of each companies from the month of March 2024.
4. The dataset obtained from Yahoo! Finance are attached in the appendix of the thesis.
5. Hence we are able to implement the Kalman Filter algorithm in Python by considering it as a maneuvering system successfully. Hence processing Kalman Gain is working as a good controller for the algorithm to work efficiently.
6. In initial chapters we have discussed the background of the Kalman Filters and discussed why this Optimal State Estimator is good for dynamic systems like the stock market.
7. As we discussed in our objective section, we are concluding that this algorithm is practical for real world applications and can be a really viable source of information for any investor.

Appendix

1) Historical Stock Data Sheet For Apple

```
Date,Open,High,Low,Close,Adj Close,Volume

2024-03-01,179.550003,180.529999,177.380005,179.660004,179.660004,73488000

2024-03-04,176.149994,176.899994,173.789993,175.100006,175.100006,81510100

2024-03-05,170.759995,172.039993,169.619995,170.119995,170.119995,95132400

2024-03-06,171.059998,171.240005,168.679993,169.119995,169.119995,68587700

2024-03-07,169.149994,170.729996,168.490005,169.000000,169.000000,71765100

2024-03-08,169.000000,173.699997,168.940002,170.729996,170.729996,76114600

2024-03-11,172.940002,174.380005,172.050003,172.750000,172.750000,60139500

2024-03-12,173.149994,174.029999,171.009995,173.229996,173.229996,59825400

2024-03-13,172.770004,173.190002,170.759995,171.130005,171.130005,52488700

2024-03-14,172.910004,174.309998,172.050003,173.000000,173.000000,72913500

2024-03-15,171.169998,172.619995,170.289993,172.619995,172.619995,121664700

2024-03-18,175.570007,177.710007,173.520004,173.720001,173.720001,75604200

2024-03-19,174.339996,176.610001,173.029999,176.080002,176.080002,55215200

2024-03-20,175.720001,178.669998,175.089996,178.669998,178.669998,53423100

2024-03-21,177.050003,177.490005,170.839996,171.369995,171.369995,106181300
```

2) Historical Stock Data Sheet For State Bank of India

```
Date,Open,High,Low,Close,Adj Close,Volume
2024-03-01,752.000000,772.000000,751.950012,769.150024,769.150024,17348526
2024-03-04,774.400024,777.000000,769.000000,772.049988,772.049988,9789141
2024-03-05,769.500000,786.950012,769.099976,783.950012,783.950012,18088847
2024-03-06,783.650024,790.299988,772.900024,783.900024,783.900024,26710525
2024-03-07,790.000000,793.400024,783.000000,788.049988,788.049988,15497868
2024-03-11,790.000000,792.799988,770.549988,773.700012,773.700012,16778340
2024-03-12,770.000000,777.750000,757.349976,759.700012,759.700012,21529705
2024-03-13,758.650024,763.700012,743.000000,747.250000,747.250000,27950252
2024-03-14,749.900024,750.799988,734.049988,741.049988,741.049988,19730882
2024-03-15,739.250000,746.549988,723.000000,731.900024,731.900024,29792241
2024-03-18,727.099976,737.900024,722.099976,730.950012,730.950012,18145126
2024-03-19,730.000000,734.349976,721.150024,723.799988,723.799988,15205043
2024-03-20,725.150024,738.950012,719.799988,736.250000,736.250000,25405455
2024-03-21,742.000000,750.599976,740.549988,744.299988,744.299988,15161161
2024-03-22,743.849976,748.799988,741.400024,746.700012,746.700012,15535921
```

3) Historical Stock Data Sheet For Bharti Airtel

```
Date,Open,High,Low,Close,Adj Close,Volume
2024-03-01,1128.000000,1140.750000,1117.400024,1128.699951,1128.699951,5927490
2024-03-04,1132.000000,1149.250000,1132.000000,1133.500000,1133.500000,5855090
2024-03-05,1133.050049,1179.750000,1133.050049,1168.900024,1168.900024,5375008
2024-03-06,1173.949951,1196.599976,1157.349976,1193.699951,1193.699951,5464839
2024-03-07,1209.849976,1213.599976,1189.449951,1199.699951,1199.699951,8903981
2024-03-11,1186.000000,1218.300049,1186.000000,1196.599976,1196.599976,6679208
2024-03-12,1200.000000,1208.000000,1197.800049,1204.250000,1204.250000,4249548
2024-03-13,1212.000000,1212.000000,1151.699951,1168.750000,1168.750000,7528096
2024-03-14,1173.949951,1199.699951,1163.550049,1194.599976,1194.599976,8569080
2024-03-15,1190.000000,1222.800049,1187.849976,1220.000000,1220.000000,10115416
2024-03-18,1225.000000,1230.550049,1215.199951,1225.300049,1225.300049,4141080
2024-03-19,1216.199951,1240.400024,1214.599976,1228.349976,1228.349976,7703719
2024-03-20,1228.250000,1239.000000,1223.650024,1231.800049,1231.800049,6596032
2024-03-21,1238.000000,1242.900024,1214.099976,1220.800049,1220.800049,9952250
2024-03-22,1226.949951,1245.000000,1222.000000,1236.099976,1236.099976,7840007
```

4) Historical Stock Data Sheet For Reliance Industries

```
Date,Open,High,Low,Close,Adj Close,Volume

2024-03-01,2927.000000,3000.000000,2925.000000,2984.250000,2984.250000,6066463

2024-03-04,2980.949951,3024.899902,2974.449951,3014.800049,3014.800049,5012210

2024-03-05,3011.550049,3014.800049,2972.100098,3000.399902,3000.399902,3553834

2024-03-06,2986.899902,3018.000000,2957.000000,3006.000000,3006.000000,3902838

2024-03-07,3005.949951,3006.199951,2951.100098,2957.850098,2957.850098,4157863

2024-03-11,2978.000000,2978.000000,2927.000000,2933.199951,2933.199951,5638565

2024-03-12,2933.199951,2976.000000,2930.050049,2950.850098,2950.850098,4716339

2024-03-13,2959.550049,2966.199951,2855.550049,2864.350098,2864.350098,6761067

2024-03-14,2879.399902,2897.050049,2851.000000,2862.949951,2862.949951,9285551

2024-03-15,2851.899902,2866.449951,2825.800049,2836.449951,2836.449951,9611909

2024-03-18,2840.000000,2883.449951,2833.050049,2878.949951,2878.949951,4584696

2024-03-19,2857.500000,2875.199951,2834.500000,2850.500000,2850.500000,4137882

2024-03-20,2855.899902,2890.000000,2848.050049,2887.500000,2887.500000,4244403

2024-03-21,2905.050049,2915.800049,2889.350098,2901.949951,2901.949951,6503468

2024-03-22,2899.949951,2920.000000,2894.699951,2910.050049,2910.050049,9763804
```

References

[R.1] Yan Xu, Guosheng Zhang, Application of Kalman Filter in the Prediction of Stock Price, Beijing Institute of Graphic Communication, Beijing, 2015.

[R.2] https://medium.com/intro-to-artificial-intelligence/kalman-filter-in-stock-trading-552e1e4b2dfb (10-03-2024)

[R.3] Deng Zi-Li. *Optimal Filtering Theory and Applications: Modern Time Series Analysis Method*. Harbin: Harbin Institute of Technology Press, 2000.

[R.4] Stocks & Commodities V. 28_1 (44-47)_ Predicting Market Data Using The Kalman Filter by R. Martinelli & N. Rhoads.

[R.5] Wan, E.A.; Van Der Merwe, R. (2000). "The unscented Kalman filter for nonlinear estimation". *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. p. 153.

[R.6] Särkkä, Simo; Hartikainen, Jouni; Svensson, Lennart; Sandblom, Fredrik (2015-04-22). "On the relation between Gaussian process quadratures and sigma-point methods".

[R.7] https://thekalmanfilter.com/kalman-filter-explained-simply/ (11-03-2024)

[R.8] *Simon, D. (2006).* Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. *Wiley-Interscience. Archived from* the original *on 2010-12-30*. Retrieved 2006-07-05*.*

[R.9] Einicke, G.A. (2019). Smoothing, Filtering and Prediction: Estimating the Past, Present and Future (2nd ed.).

[R.10] Liu, W.; Principe, J.C. and Haykin, S. (2010). Kernel Adaptive Filtering: A Comprehensive Introduction. John Wiley.

[R.11] Burkhart, Michael C. (2019). "Chapter 1. An Overview of Bayesian Filtering". *A Discriminative Approach to Bayesian Filtering with Applications to Human Neural Decoding*. Providence, RI, USA: Brown University.

[R.12] Irwin, Scott H.; Park, Cheol-Ho (2007). "What Do We Know About the Profitability of Technical Analysis?". Journal of Economic Surveys. 21 (4): 786–826.

[R.13] Thawornwong, S.; Enke, D. Forecasting Stock Returns with Artificial Neural Networks, Chap. 3. In: Neural Networks in Business Forecasting, Editor: Zhang, G.P. IRM Press, 2004.

[R.14] Levine, Ross (2002). "Bank-Based or Market-Based Financial Systems: Which Is Better?". Journal of Financial Intermediation. **11** (4): 398–428.

[R.15] Tversky, A. & Kahneman, D. (1974). "Judgement under uncertainty: heuristics and biases". Science. 185 (4157): 1124–1131.

[R.16] Shiller, Robert (2005). Irrational Exuberance (2nd ed.). Princeton University Press. ISBN 978-0-691-12335-6.

[R.17] Nier, Erlend Walter. "Financial Stability Frameworks and the Role of Central Banks: Lessons from the Crisis" (PDF). International Monetary Fund. Archived (PDF) from the original on March 4, 2016.

[R.18] https://www.kalmanfilter.net/kalmanmulti.html (10.03.2024)

[R.19] Gilson, Ronald J.; Black, Bernard S. (1998). "Venture Capital and the Structure of Capital Markets: Banks Versus Stock Markets". *Journal of Financial Economics*. 47.

[R.20] Diamond, Peter A. (1967). "The Role of a Stock Market in a General Equilibrium Model with Technological Uncertainty". *American Economic Review*. 57 (4): 759–776.

[R.21] George R. Adams (March 1977). "New York Stock Exchange National Register of Historic Places Inventory-Nomination", National Park Service.

[R.22] McCulloch, W; Pitts, W (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*. **5** (4): 115–133.

[R.23] https://www.intechopen.com/chapters/63164 (15.03.2024)