C programming approach for prediction of Stock prices based on Brownian Motion theory and Monte Carlo simulation

for the Bachelor of Science (General) Degree

By
0.G.V.V Bandara
sc/2019/10723

Supervisor:
Dr H.L Jayathilake

Department of Mathematics
University of Ruhuna
Matara.

2020

Acknowledgment

This project has been capable of completing with the kind support of many people. Hereby I would like to pass my sincere gratitude. First and foremost my sincere gratitude goes to our supervisor Dr. H.L Jayathilake for providing us with the required guidance and resources to complete this project. Next I would like to pass my gratitude to our instructors Miss Nimanthi and Miss Dilmi for providing us with the required guidance and resources in completing the project.I would also like to pass my gratitude to all my lecturers and instructors who enlightened us with the c programming knowledge. My sincere gratitude also goes out to our lecturers of the science faculty for providing us with the required guidance and resources which made this possible. At last but but not least this would not have been capable without the contribution,co-operation of my group members so I would like to pass my gratitude to them as well.

Abstract

The project carries out the objective of developing a code to predict the stock price using the C programming language. By using two quantities Drift and the Random Change the code aims to predict the stock price of a future day. The user is able to input data of the stock prices of a certain stock for previous days. The code assists the user throughout the process of entering data. The code obtains the entered data and performs the calculation which in return outputs the predicted stock price. The code in the end outputs a plot as an image which enables the user to visualize the stock price variation

Contents

List of Figures

Chapter 1

Introduction

1.1 Background of the study

1.1.1 Stock market

The stock market,share market and also referred to as the equity market aggregates the buyers and sellers of stocks.This includes securities listed on public stock exchange and shares that are sold privately on crowd funding platforms. securities are tradable through trading software or through a broker by any individual. registered individuals will have the ability to buy stock based on the prediction of where the price would fluctuate which in return would provide a profit if the stock price increases. buying and selling of stocks occurs such that the trader takes advantage of the voltility of the stock prices. A rise in the purchased stock price creates a profit and the fall creates a loss. Stock prices tend to be very volatile and depend on the company or ownership that sells the stocks. The objective and the nature of the fluctuation of previous stock prices pay a key role in deciding the nature of the stock of a buisness.A broker an intermediate in the process enables the trader to buy and sell stocks by providing a trading platform and other tools. Brokers require a small commission to complete the trading process but private assets cannot be traded without the interference of a broker since the broker lists the available stocks for the trader. some brokers have realtime charts that display the nature of the stock prices. The random nature of stock prices are

regarded to be brownian in nature. Investors, traders and brokers have all expressed their concerns over the volatility of the stock market.The perception that prices move a lot are considered to be merely a reflection of the historically high level of stock indexes. There is little evidence that the volatility increased due to computerized trading.The volatility is apparent when analyzing variation of stock price in large companies.

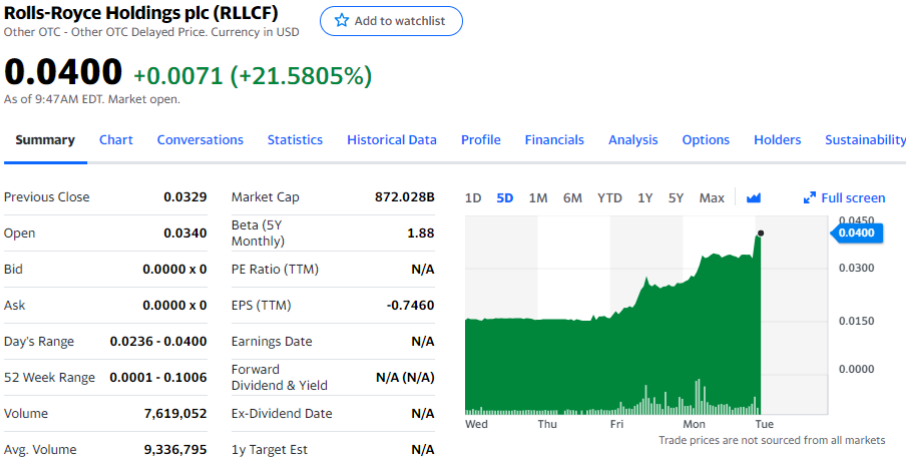

Figure 1.1: five day variation of stock prices of Rolles Royce

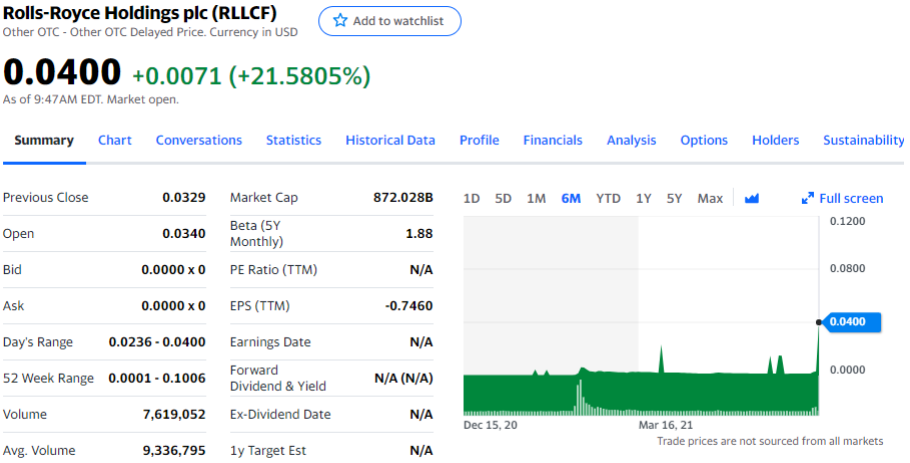

Figure 1.2: six month variation of stock prices of Rolles Royce

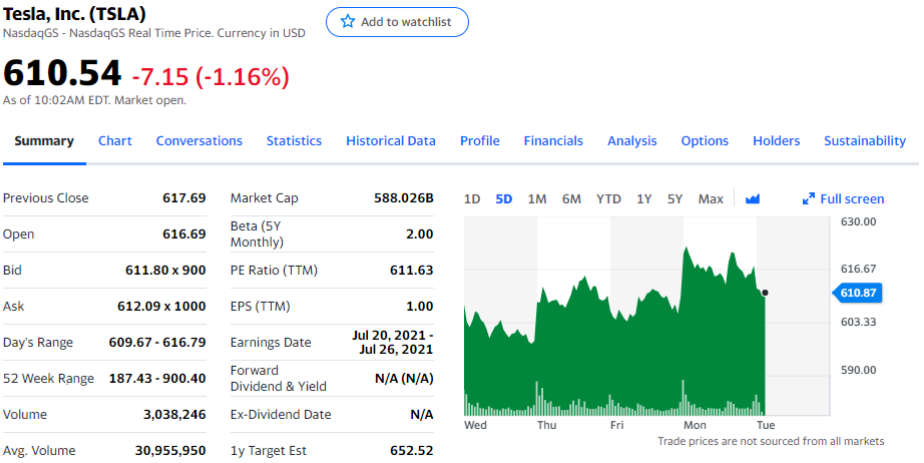

Figure 1.3: five day variation of stock prices of tesla

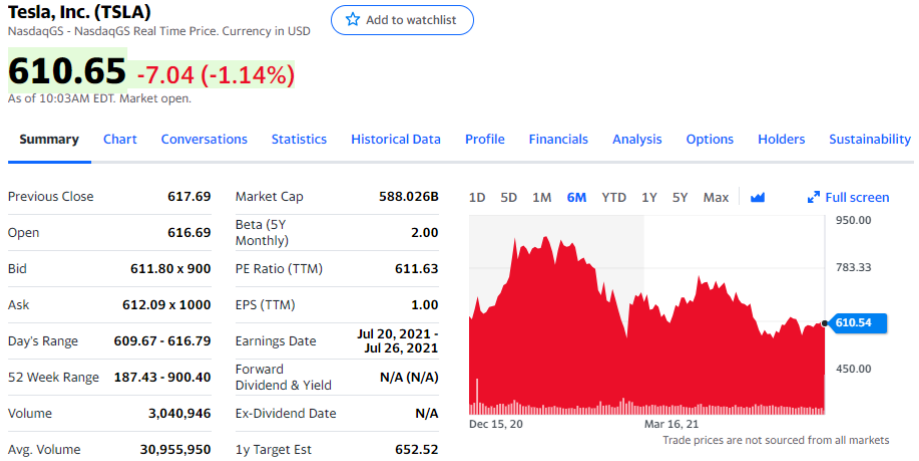

Figure 1.4: six month variation of stock prices of tesla

1.1.2 Brownian Motion Theory

This pattern of motion typically consists of random fluctuations in a particle's position inside a fluid sub-domain, followed by a relocation to another sub-domain. Each relocation is followed by more fluctuations within the new closed volume. The fluids linear and angular momentum remains null over time.Such fluids have no preferred direction of flow.The idea that the molecules of a liquid or gas are constantly in motion is a key factor in the kinetic theory.It was then naturally assumed that this type of motion can be imposed onto large

particles.The German physicist Albert Einstein proposed a quantitative theory related to Brownian Motion in the later years. Reasoning on the basis of statistical mechanics, he showed that for such a microscopic particle the random difference between the pressure of molecular bombardment on two opposite sides would cause it to constantly wobble back and forth. A smaller particle, a less viscous fluid, and a higher temperature would each increase the amount of motion one could expect to observe. Over a period of time, the particle would tend to drift from its starting point, and, on the basis of kinetic theory. It is possible to compute the probabilty(p) that a particle moves a distance(x) in a given time(t).probability(p) plotted against (x) represents a bell shaped gaussian curve.If the coefficient of diffusion was considered to be (D)

$$p = \frac{\exp(\frac{-x^2}{4\mathrm{Dt}})}{2\sqrt{\pi \mathrm{Dt}}} \tag{1.1}$$

Assets have continuous prices evolving continuously in time and are driven by Brownian motion processes. An assumption is that asset prices have no jumps, that is there are no surprises in the market.Similar to the motion of a microscopic particle in Einstein's theory, the stock price drifts from the initial position.

1.1.3 Monte-Carlo simulation

Monte Carlo Simulation, also known as the Monte Carlo Method or a multiple probability simulation, is a mathematical technique, which is used to estimate the possible outcomes of an uncertain event.Unlike a normal forecasting model, Monte Carlo Simulation predicts a set of outcomes based on an estimated range of values versus a set of fixed input values.Monte-Carlo simulation builds a model of possible results by utilizing a probability distribution curve.The probability distribution curve is uniform normal distribution curve. The results are calculated over and over using different random numbers between the maximum and the minimum. The exercise is repeated thousands of times to get a large number of outcomes.As the number of inputs increase the number of forecasts grow which enables projecting outcomes furthur in time with more accuracy.

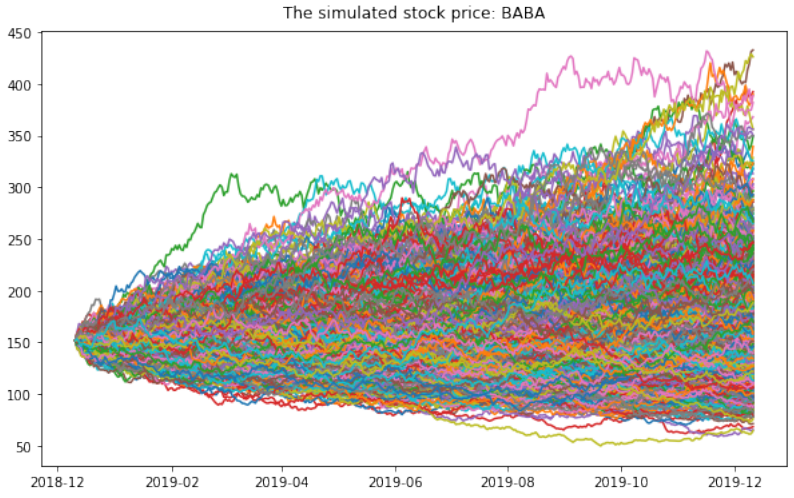

Figure 1.5: Monte carlo simulation

1.2 Objectives

1.2.1 Major Object

C is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system. By design, C provides constructs that map efficiently to typical machine instructions. It has found lasting use in applications previously coded in assembly language.

i C programming is utilized in this occasion in order to create a code to execute statistical equations. The programmer has the ability to program the whole calculation into the C code.This takes away the pain of performing the same calculation multiple times from the user

ii When utilizing the program the user will interact with an environment which provides the calculated information. The user is only required to enter the accurate information/input data as instructed and guided by the code. The computer will do the calculation and provide the user with the required results and the predicted output data.

iii without the program the user would require to allocate a larger amount of time to engage in calculation of the steps of the calculation. The code will do the calculation utilizing the power of the computer which would reduce the time drastically.

iv utilizing the C program the user is able to get accurate information. If the calculation were to be done manually errors or mistakes can occur. The use of the c code eiliminates the problem.

Chapter 2

Problem Statement

2.1 volatility of stock price

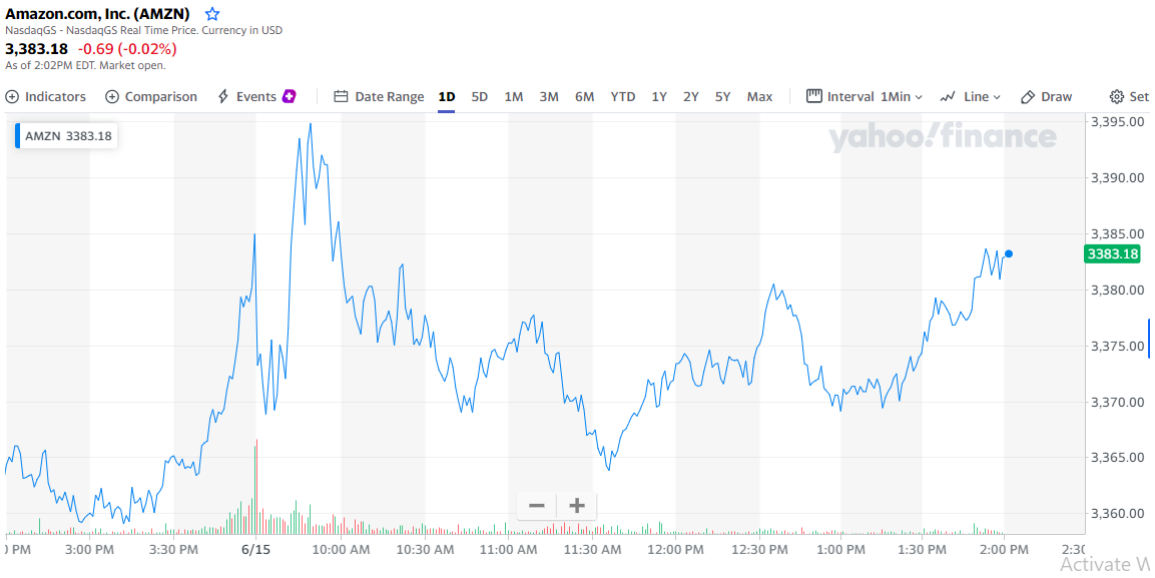

Figure 2.1: one day variation of Amazon stock price

The random change is what makes the stock price more unpredictable. The volatile nature also depends on the company in which the stocks are purchased. Some stocks tend to have lesser volatility and more consistency.The stocks with greater volatility are the hardest to predict since they tend to fluctuate more within small time frames

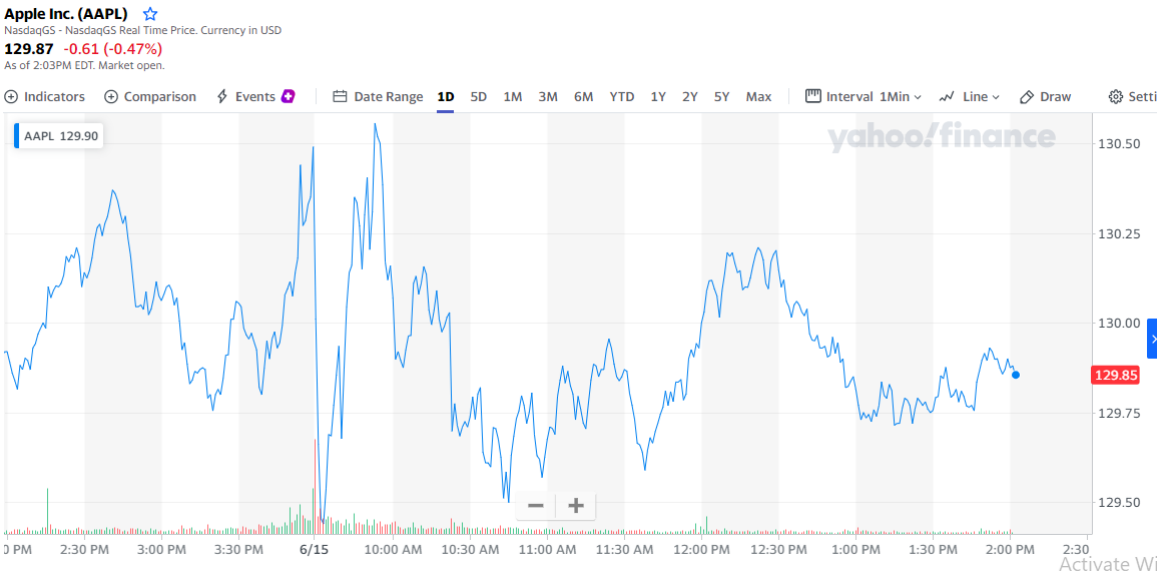

Figure 2.2: one day variation of apple stock price

The stock price has a lesser volatility and the stock price continuously rises or falls within short periods of time and large fluctuations are observed when analyzing a longer time frame.

The stock price of apple appears to be highly volatile and sudden rise and fall of stock price can be observed in shorter time frames.Even during longer when analyzing longer time frames the stock price seems to be inconsistent.

2.2 predictability depends on user experience

Buying and selling of stock prices are done by individuals that are referred to as traders. When buying and selling stocks there is always a risk involved. The stock prices could continuously decrease and reach a low or continously increase and reach a high.The stock price can fluctuate in a very unpredictable manner which can create loss during trading rather than a profit. The trader identifies the risk and trading is done while being aware of the risk that is involved. Brokers provide charts that display historical variation of stock prices and also provide realtime variation of stock prices. The trader analyses the charts using various techniques and makes he decision to buy or sell a stock. analyzing method may vary depending on the stock type and company.predicting stock price in this manner requires proper background study and research on the stock and companies and historical data. The decision making ability depends on the experience,which requires a large time to be skilled in the process

Chapter 3

Methodology

3.1 Drift part of formula

Drift is the general change in the stock price over a long period of time.The general trend of the graph is to go up as the value of the stock keeps increasing as the asset gains more value. The short term rise and fall represent the random daily volatility.we say that

$$\text{Drift} = \text{average daily return} - \frac{\text{variance}}{2} \tag{3.1}$$

variance is a measure of how much each value varies from the mean.Periodic daily return is a variable that relates a previous days price to the current days price.

$$\text{Periodic daily return} = ln\Big(\frac{\text{days price}}{\text{previous days price}}\Big) \tag{3.2}$$

If the days are considered backwards starting from the current day, which means "days price" refers to each of the days taken backwards consecutively and "previous days price would refer to the previous days of each of those days. The purpose of the code is to predict the price of a certian day so data would be considered from the day before the day which the price would be predicted. "previous day one" refers to the first day before the day that the stock price is to be predicted.

$$\text{Periodic daily return one} = ln\Big(\frac{\text{previous day one price}}{\text{previous day two price}}\Big) \tag{3.3}$$

$$\text{Periodic daily return two} = ln\Big(\frac{\text{previous day two price}}{\text{previous day three price}}\Big) \tag{3.4}$$

$$\text{Periodic daily return three} = ln\Big(\frac{\text{previous day three price}}{\text{previous day four price}}\Big) \tag{3.5}$$

$$\vdots \tag{3.6}$$

$$\vdots \tag{3.7}$$

$$\vdots \tag{3.8}$$

$$\text{Periodic daily return n} = ln\Big(\frac{\text{previous day n price}}{\text{previous day n+1 price}}\Big) \tag{3.9}$$

For the calculation of the variance, the mean of the periodic daily return values are required. The mean of the periodic daily return values is referred to as the "average daily return"

$$\text{average daily return} = \frac{\text{sum of periodic daily return values}}{\text{number of periodic daily return values"n"}} \tag{3.10}$$

The general formula for the variance is..

$$\text{variance} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{\text{number of x values}} \tag{3.11}$$

When applying this formula to or instance, x bar is the average daily return value and
x are the periodic daily return values.The drift can be calculated utilizing the average daily return value and variance

$$\text{drift} = \text{average daily return} - \frac{variance}{2} \tag{3.12}$$

3.2 Random change part of formula

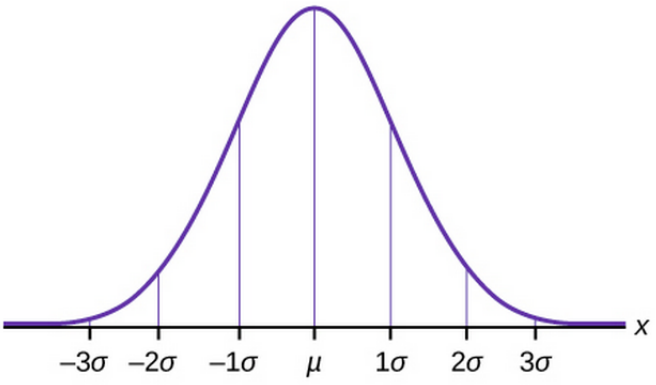

Figure 3.1: normal distribution curve

A random number generated by the computer is obtained in order to perform the calculation. The computer generated vale is modified. Generally the variation of the value about the mean value could be large or small but generally it's a value in between. It's known that the stock prices tend to follow a normal distribution

statistics say that 68 percent of values lie between standard deviation. The standard deviation and the variance are related as

$$\text{standard deviation} = \sqrt{\text{variance}} \tag{3.13}$$

The random number is generated by the computer.Let the random number be "q". A function "t" is defined as follows

$$\text{t} = \sqrt{\ln\left(\frac{1}{q^2}\right)} \tag{3.14}$$

Another function "x" is defined as follows

$$\text{x} = \text{t} - \frac{\left(c_0 + c_1 t + c_2 t^2\right)}{\left(1 + b_1 t + b_2 t^2 + b_3 t^3\right)} \tag{3.15}$$

The constants of x are defined as

$$c_0 = 2.515517 \tag{3.16}$$

$$c_1 = 0.802853 \tag{3.17}$$

$$c_2 = 0.010328 \tag{3.18}$$

$$b_1 = 1.432788 \tag{3.19}$$

$$b_2 = 0.189269 \tag{3.20}$$

$$b_3 = 0.001308 \tag{3.21}$$

Utilizing the above functions the final equation to find the "Random Change" is constructed.

$$\text{Random change} = \text{standard deviation} \times \text{x} \tag{3.22}$$

3.3 Final formula

The final formula intends to predict a days stock price based on previous days stock price data. Combining both the previously obtained elements "random change" and "drift" the final formal is constructed as

$$\text{today's stock price} = \text{yesterday's stock price} \times \exp(\text{drift+random change}) \tag{3.23}$$

Chapter 4

Discussion

The code has the ability to predict stock prices utilizing the inputs. In order to test the accuracy of the code in predicting stock prices of a real stock available in the stock market, the code is tested with actual data. Here data of the variation of stock prices of the company tesla will be used to test the code for accuracy of prediction. The following series of images depicts how the code calculated and predicted the stock price. To graphically depict the output the code has the capability to plot a graph that shows the variation of stock prices.

4.1 analysis of real data sample

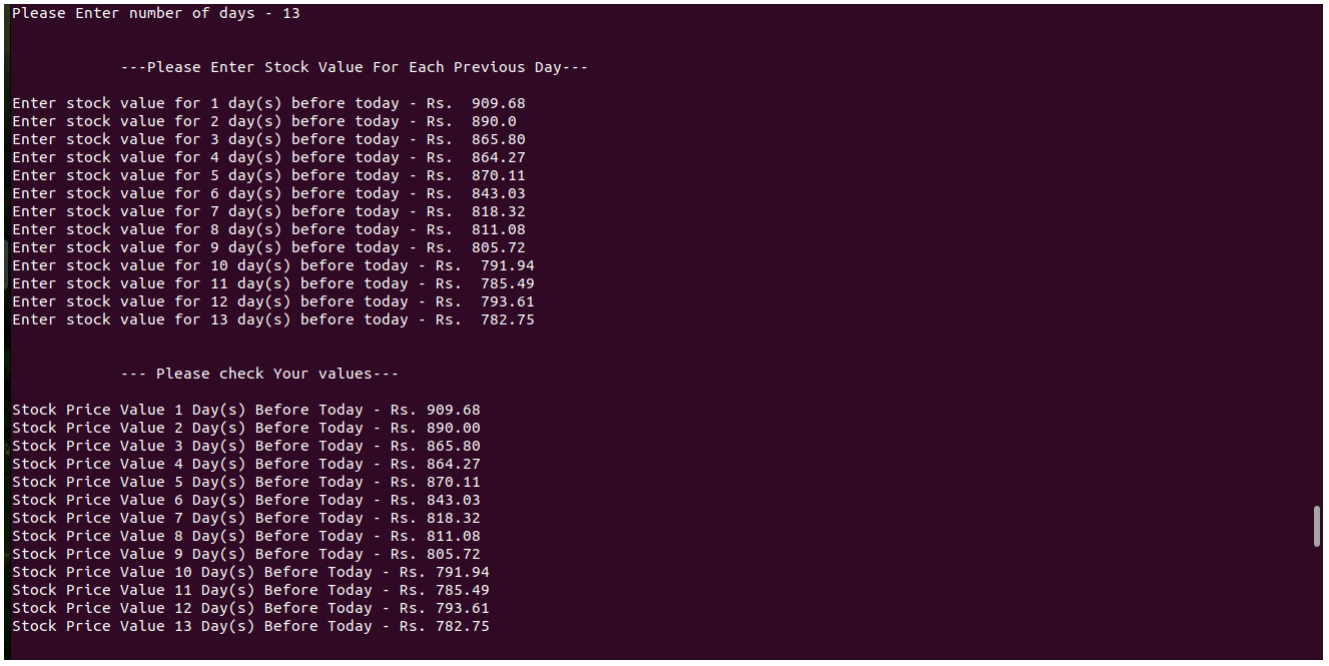

Figure 4.1: execution of the graph and entering real values of stock price changes of the company tesla

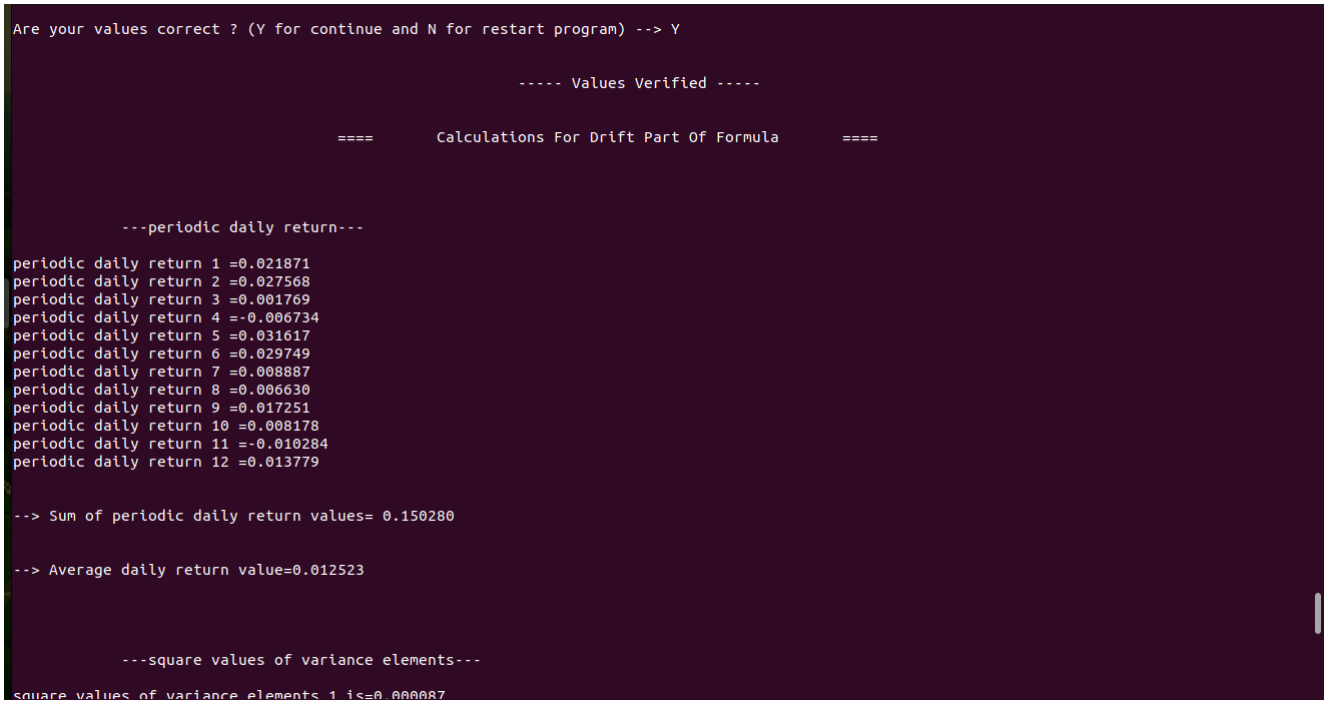

Figure 4.2: checking whether the entered values are acceptable as stock prices and the drift part of the formula is displayed

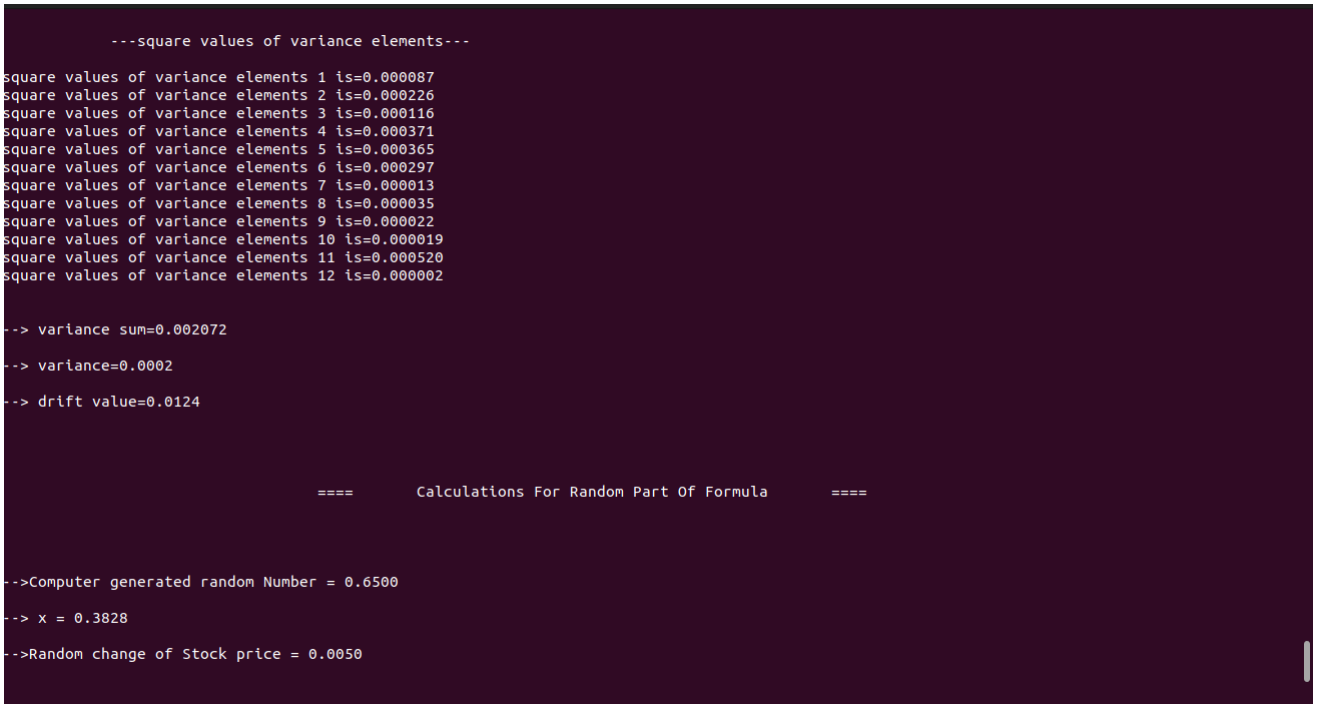

Figure 4.3: obtained results for the calculation of random variables of the code

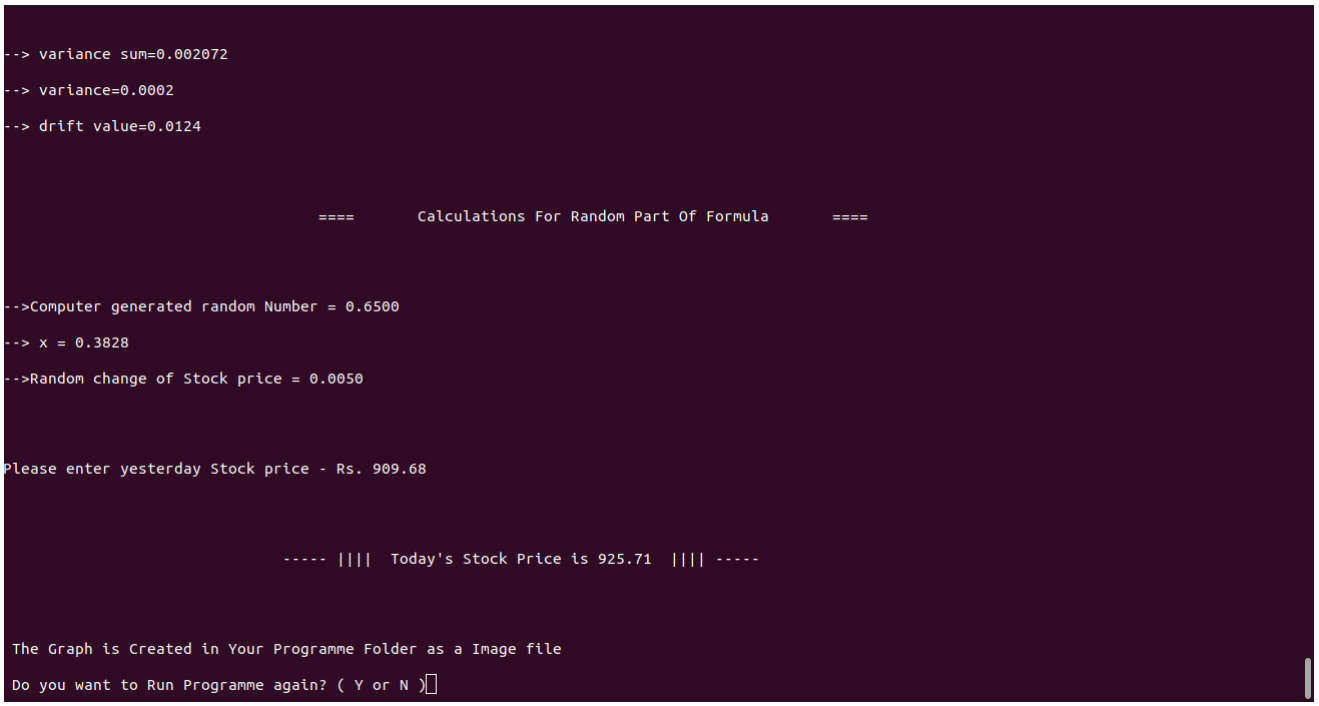

Figure 4.4: displaying the predicted stock price and execution of graph plotting process

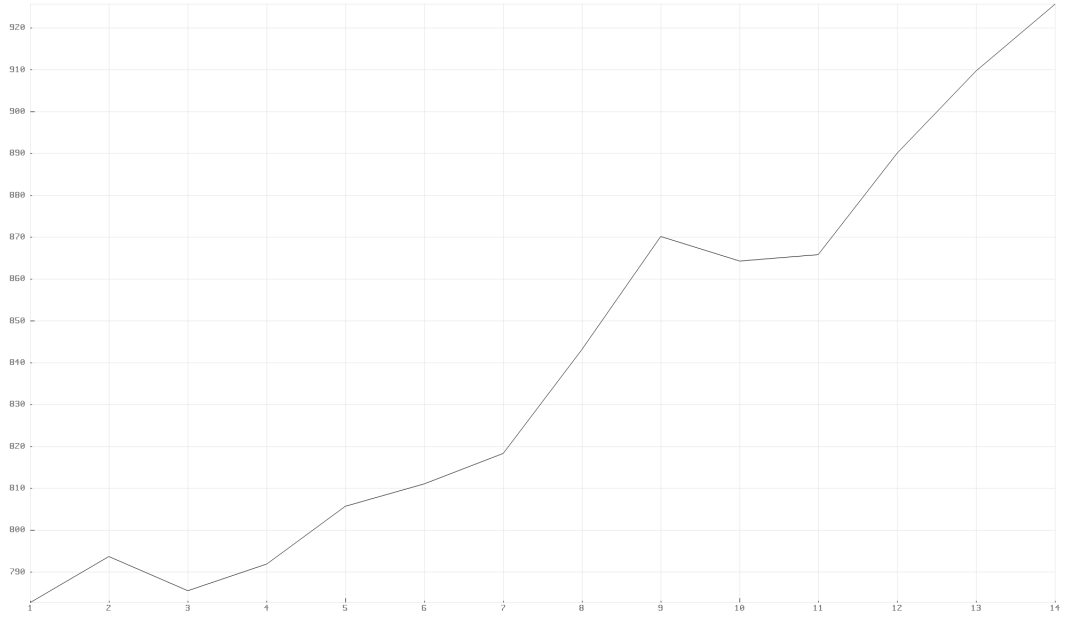

Figure 4.5: The plot of the stock price variation obtained from the code

4.2 Error calculation to test for accuracy

The percentage error calculation for the predicted stock price value of the data

$$\text{Predicted value using code} = 925.71 \tag{4.1}$$

$$\text{Actual value} = 1024.86 \tag{4.2}$$

$$\text{percentage error} = \frac{1024.86 - 925.71}{1024.86} \times 100 \tag{4.3}$$

$$\text{percentage error} = 9.67 \tag{4.4}$$

Chapter 5

Conclusion

The development of the code has been successfully completed. The initial suggestion was to develop a code that would predict the stock price of a day. The code obtained has the ability to predict the stock price with a less than 10 percent error. The code is easy to use by the user and the user is only required to enter the data. The code also tests for accurate values that can be considered as stock prices thereby assisting the user in entering the values. The code also plots a graph "Stock price variation" vs "the days" which enables the user to visualize the change in stock prices over a prolonged period and come to certain conclusions by visual examination of the general trend of the stock price variation. The code developed can be of great assistance to anyone who requires to obtain a prediction regarding the stock price of a future day with minimum error thereby concluding the success of the code

Chapter 6

Appendix

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include "pbPlots.h"
#include "supportLib.h"
#include <unistd.h>

//Function declaration
void drift_calculation (double a_1[], int n, double* Dr, double* Vr);
double random_calculation (double var);
double Final_formula (double drift, double random_change);
void Graph_creation (double a_1[], int n_2 , double tsp);

//main Function
int main()
{

    int i,n_1;
    double a[1000], temp, drift_1, rand_1, var_1;
    char verif, verify, run_1 ,run_2;

start_1:
        //obtaining number of days
            printf("Please Enter number of days - ");
            scanf("%d",&n_1);

        //loop for scanning daily stock price values
            printf("\n\n            ---Please Enter
            Stock Value For Each Previous Day---\n\n");
            for(i=n_1; i>0;)
            {
                printf("Enter stock value for %d day(s)
                before today - Rs.  ",n_1-i+1);
                scanf("%lf",&temp);

            //Chacking for minus numbers
                if(temp > 0.00)
                {
                    a[i-1] = temp;
                    temp=0;
                    i--;
                }
                else
```

```
                {
                    printf("\n !!!!!! Please enter a
                    valid value !!!!!!\n");
                    temp=0;
                }
            }
        //loop for printing the daily stock price values
            printf("\n\n            --- Please check
            Your values---\n\n");
            for(i=n_1; i>0; i--)
            {
                printf("Stock Price Value %d Day(s) Before
                Today - Rs. %.2f\n",n_1-i+1,a[i-1]);
            }

start_2:
        //Verification
            printf("\n\nAre your values correct ? (Y for
            continue and N for restart program) --> ");
            scanf(" %c",&verify);
            if(verify=='Y')
            {
                goto verified;
            }
            else
            {
                if(verify=='N')
                {
                    printf("\n\n
                     --- Programme Restarted---\n\n");
                    goto start_1;
                }
                else
                {
                    goto start_2;
                }
            }
verified:
            printf("\n\n
                                    ----- Values Verified -----\n\n");
        //Calling Function for calculate drift part
            drift_calculation (a,n_1, &drift_1, &var_1);
            printf("--> drift value=%.4lf\n\n",drift_1);
        //Calling Function for calculate Random part
            rand_1 = random_calculation (var_1);
            printf("\n\n-->Random change of
            Stock price = %.4lf\n\n",rand_1);
        // Calling function to final part
            double today_stock_price = Final_formula(drift_1,rand_1);
```

```
        // Printing Final result
            printf(" \n\n\n\n
             ----- ||||  Today's Stock Price is %.2lf
             |||| -----\n\n\n\n",today_stock_price);
        // Calling function to create Graph
            Graph_creation (a,n_1,today_stock_price);
start_3:
        // ASk to run again
            printf("\n\n Do you want to Run Programme again? ( Y or N )");
            scanf(" %c",&run_1);
        // check for answer
            if(run_1=='Y')
            {
                printf("\n\n
                                              --- Program Restarted---\n\n");
                goto start_1;
            }
            else
            {
                if(run_1=='N')
                {
                    printf("\n\n
                                                  --- Thank You ---\n\n");
                    goto end;
                }
                else
                {
                    goto start_3;
                }
            }

end:
return 0;
}

// Defining function for drift part calculation
void drift_calculation (double a_1[],int n,double* Dr,double* Vr)
{
            int i;
            double b[1000],c[1000],k[1000],m[1000],sum,ave,vars,temp;
            double drift,var;
            printf("\n
             ====       Calculations For Drift Part Of Formula
                   ====\n\n\n");
        //loop for calculating periodic daily return
            for(i=n; i>0; )
            {
                b[n-i+1]=a_1[i-1]/a_1[i-2];
                c[n-i+1]=logf(b[n-i+1]);
                i--;
```

```
    }
//loop for printing average daily return values
    printf("\n\n            ---periodic daily return---\n\n");
    for(i=1; i<n; i++)
    {
        printf("periodic daily return %d =%f\n",i,c[i]);
    }
    printf("\n\n");
//summation of periodic daily return values
    for(i=1; i<n; i++)
    {
        sum=sum+c[i];
    }
    printf("--> Sum of periodic daily return values= %f\n",sum);
    printf("\n\n");
//average daily return
    ave=sum/(n-1);
    printf("--> Average daily return value=%f\n",ave);
    printf("\n\n");
//calculating variance
    for(i=1; i<n; i++)
    {
        k[i]=c[i]-ave;
    }
//square of elements
    for(i=1; i<n; i++)
    {
        m[i]=pow(k[i],2);
    }
//printing square of variance elements
    printf("\n\n            ---square values
    of variance elements---\n\n");
    for(i=1; i<n; i++)
    {
        printf("square values of variance
        elements %d is=%f\n",i,m[i]);
    }
//summation of matrix elements
    for(i=1; i<n; i++)
    {
        vars=vars+m[i];
    }
//sum of square elements
    printf("\n\n--> variance sum=%f\n\n",vars);
//variance
    var=vars/(n-1);
    printf("--> variance=%.4lf\n\n",var);
//calculating drift
    drift=ave-(var/2);
    *Dr= drift;
    *Vr = var;
```

```
        return;
}
// Defining function for random part calculation
double random_calculation (double var)
{
            printf("\n\n\n
                   ====       Calculations For Random Part Of Formula
                        ====\n\n\n");
            double x,j,o,l,t;
            // x - x for random probability | j - c part of formula
            | o - b part of formula | l - square root of log natural
            | t - time
            double ran,sqr,nat_log,inv_sqr,stnd_dev,random_change;
             // ran - random number | sqr - Square of random number
             | natlog - Natural log | stnd_dev - Standard deviation
             | random_change - final random change
        // generating random number
            srand(time(0));
            t = rand() % 100;
            ran = t/100;
        // print random number
            printf("\n\n-->Computer generated random Number = %.4lf",ran);
        // Square of random number
            sqr = ran*ran;
            //printf("\n y is %.4lf",sqr );
        // reciprocal of square of random number
            inv_sqr = 1/sqr;
            //printf("\n g is %.4lf",invsqr );
        // log natural of reciprocal of square of random number
            nat_log = log(inv_sqr);
            //printf("\n l is %.4lf",natlog );
        // square root of log natural of reciprocal of square of random number
            l = sqrt(nat_log);
            //printf("\n p is %.4lf",a );
        // c part of formula
            j = 2.515517 + (0.802853*l) + (0.010328*(l*l));
            //printf("\n c is %.4lf",c );
        // b part of formula
            o = 1 + (1.432788*l) + (0.189269*(l*l)) + (0.001308*(l*l*l));
            //printf("\n b is %.4lf",b );
        // calculate x
            x = (j/o)-l;
        // print x
            printf("\n\n--> x = %.4lf",x );
        // calculating standard deviation
            stnd_dev = sqrt(var);
        // calculating random change
```

```
            random_change = stnd_dev * x;
        // return random change
            return random_change ;
}
// Defining fuction for Final Formula
double Final_formula (double drift,double random_change)
{
     double yest_stock_price,p,q,tsp;
        // addition
            p = drift + random_change;
        // printf("\n p is %.4lf",p );
        // calculating exponential
            q = exp(p);
        // printf("\n q is %.4lf",q );
        // obtaining yesterday stock price
            printf("\n\n\nPlease enter yesterday Stock price - Rs. " );
            scanf("%lf",&yest_stock_price);
            //printf("\n yest is %.4lf",yest_stock_price );
        // calculating today stock price
            tsp= yest_stock_price * q;
            return tsp;
}
// Defining function for Create Graph
void Graph_creation (double a_1[],int n_2 ,double tsp)
{
            int  x_range, y_range,i ;
            double x_r[1000];
        // Loop for elements of x-axis
            for (i=0; i<=n_2; i++)
            {
                x_r[i]=i+1;
            }
        // Adding calculated stockprice as last element of the array
            a_1[n_2] = tsp;
        // assigning element number in arrays
            x_range = n_2+1;
            y_range = n_2+1;
        // Image creating function
            RGBABitmapImageReference
            *imageref = CreateRGBABitmapImageReference();
        // Plot (function, image width ,
        image height , x array , number of elements in x array ,
         y array , number of elements in y array)
            DrawScatterPlot(imageref, 2400, 1400, x_r, x_range,
            a_1, y_range);
        //
            size_t length;
```

```
            double *pngData = ConvertToPNG(&length, imageref->image);
        // Create Graph .png File
            WriteToFile(pngData, length, "plot.png");
            printf("\n The Graph is Created in Your
             Programme Folder as a Image file ");
            return ;
}
```

The custom libraries "pbPlots.h" and "supportLib.h" are utilized in developing this code. The custom libraries are available on github and free to use by anyone. The custom libraries Available at: `https://github.com/InductiveComputerScience/pbPlots/tree/v0.1.7.1/C`

Bibliography

[1] Kac, M. (2018). Random Walk and the Theory of Brownian Motion. pp.(369–391).

[2] P (2020). Practical Numerical C Programming : Finance, Engineering, and Physics Applications. Berkeley, Ca: Apress, Imprint Apress.

[3] Wikipedia Contributors (2019c). Stock market. [online] Wikipedia. Available at: `https://en.wikipedia.org/wiki/Stock_market`.

[4] Schwert, G.W. (2018). Stock Market Volatility,G. William Schwert. pp.(23–34).

[5] Wikipedia Contributors (2019a). Brownian motion. (online) Wikipedia. Available at: `https://en.wikipedia.org/wiki/Brownian_motion`.

[6] Wikipedia Contributors (2019b). C (programming language). (online) Wikipedia. Available at: `https://en.wikipedia.org/wiki/C_(programming_language)`.

[7] Yahoo Finance (2021). Yahoo Finance - Business Finance, Stock Market, Quotes, News. (online) Yahoo Finance. Available at :`https://finance.yahoo.com/`.

[8] `www.google.com`. (n.d.). Tesla Inc (TSLA : NASDAQ) Stock Price , News - Google Finance. (online) Available at:`https://www.google.com/finance/quote/TSLA:NASDAQ`.

[9] Stack Overflow. (n.d.). graph - How to plot data by c program? (online) Available at: `https://stackoverflow.com/questions/14311640/how-to-plot-data-by-c-program[Accessed6Nov.2021]`.