AM3611: Final Project
Geometric Brownian Motion Simulations of Tesla Motors

Submitted to: Professor Colin Denniston
Written by: Mina Yuan and Natalie To

Date: December 7th 2018
Pages: 20

Introduction

Beating the market is an ongoing controversial topic between portfolio managers, investors, and speculators. Following the Efficient Market Hypothesis (EMH), no forms of technical and fundamental analysis can help investors to consistently beat the market performance, as the future stock prices follow a random walk while past prices have incorporated all expected information. Despite the theory, various hedge funds have been able to show positive returns which let investor wonder how efficient the stock market is. At the very least, the Geometric Brownian Motion (GBM) is a widely recognized fundamental model that conforms with the weak form of EMH has been used by traders and analysts in the financial industry to provide them with a holistic view of potential future exposure of the financial institution's assets.

This paper uses Monte Carlo with the GBM method to model the stochastic price movements of Tesla Motors (Tesla) closing prices over the last year. The prediction is drawn on historical financial data of Tesla of the past five years in the NASDAQ's stock exchange. In 2018, Tesla has made major progress with the company turned a profit in its 2018 Q3 earning for the second time in its history. Its stock soared 7% in the market one day after its release and raised more than 34% one month later to its current price of $344 from $260. Aside from the positive news, Tesla has been under turmoil and scrutinized by the sentimental shareholders in last few months with unsuccessful stock buybacks, sudden personnel changes in high-level executives, and the CEO Elon Musk's attempt at marijuana on a popular podcast show. Given Musk is public opinions and the worrisome sentiments about the company's debts and profits, Tesla has experienced extremely volatile price movements in the market with its 52-week range at 244.59 - 387.46 (Bloomberg). As evidenced, Tesla has been considered to be a growth stock with high volatility. The model is also applied to a smaller and less volatile company, Danaos Corporation's stock. Hence, this paper's results examined the appropriateness of the Geometric Brownian Motion model in forecasting volatile stocks by evaluating Mean Squared Errors (MSE) and Mean Absolute Percentage Errors (MAPE). The paper further extended the model in financial application to support investment portfolio management with Value at Risk calculations.

Literature Review

The Geometric Brownian motion is a continuous stochastic process in which the logarithm of a randomly changing factor follows the Brownian motion with a constant drift (Ross, 2014). Various researchers has applied this model in various fields of studies such as stock prices, natural resources prices, and demand for products or services. For instance, Marathe and Ryan concluded that GBM process is applicable to data that shares the criteria of normality and independence such as electric utility data and passenger data (Marathe, Ryan). More recently, in a study done by Mosiato and Moreno-Okuno, they explored both GBM and the Variance-Gamma method to model the fossil fuels prices and noted the limitation of GBM that it can not capture extreme movements in the international market (Mosiato, Moreno-Okuno, 2018).

Nevertheless, the GBM model and its variations have been widely accepted in the financial industry and are valid for modelling the growth of equity assets over time. That notwithstanding, the GBM is not completely realistic model give that stock market is highly uncertain and volatile while the models assume otherwise.

Summary

This study was an attempt to evaluate the appropriateness of the Geometric Brownian motion in modelling volatile stocks daily and based the Black-Scholes-Merton option pricing model and portfolio management decisions on the simulations.

Among other discoveries, the paper found out that the MSE value in the simulations is consistently high, which implies the flaws in using a constant the historical returns of stock considering the impact of seasonal jumps. Base on the MAPE value and the plots of simulation, the model serves as an accurate measure of the general trend of the stocks and the value at risk in equity investment. Additionally, the model could be modified with Merton Jump Diffusion to adjusts stock prices for unexpected market movements.

Conceptual Framework

Geometric Brownian Motion Concept

Before understanding the importance of Geometric Brownian Motion, we need to understand what a stock is and how people use it today. Davison defines a stock as a "fractional ownership, with limited liability in a company" (Davison 157). Investors and traders spend a great deal trying to understand the markets and to predict their movements. One way to better their judgment is to create statistical models that could simulate the possible paths a stock could take which would help them minimize their risk and optimize their portfolio's position. This paper will speak more about the famous model, Geometric Brownian Motion, a discovery made by Robert Brown during the 1820s when he noticed the random path a pollen took on the surface of a liquid (Wikipedia). He realized that the continuously moving particles in the water acted on the pollen and forced it to move in a jittery pattern. This motion resembled a stock price moving up and down in a jittery pattern due to its investors' decisions to buy or sell the stock. Robert Brown was able to take the idea and form the stochastic differential equation below.

$$dS = \mu S dt + \sigma S dw, \; S(0) = S_0$$

Here, S is the stock price and it changes upon t time, μ is the drift, and dw is a random increment of a Wiener process. Lastly, σ is the volatility and there is an initial condition where $S(0)$ is the stock price at time zero. As Davison says, "a Wiener or diffusion process based on the behavior of a binomial tree model as modeled by a random walk. These processes are also often known as Brownian motions" (Davison 227).

The first step to solving the stochastic differential equation is to divide both sides by S and recognize that one can now integrate the left side.

$$\frac{dS}{S} = \mu dt + \sigma dw$$

$$\frac{dS}{S} = d(\ln S) + \frac{1}{2}\sigma^2 dt$$

After integrating the left side one can isolate dt and integrate the entire equation from 0 to time t.

$$\int_0^t d(\ln S) = \int_0^t \left(\mu - \frac{1}{2}\sigma^2\right) du + \int_0^t \sigma \, dW_u$$

$$\ln S = \ln S_0 + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t$$

$$S = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t} e^{\sigma W_t}$$

This is the closed form of the Geometric Brownian Motion. A more detailed solution to the stochastic differential equation can be found in Davison's textbook on pages 241 to 243. The closed form will be used in the program to simulate Tesla Motor's stock price using five years of historical data from Nasdaq. As well, we will consider 252 trading days in a year instead of 365 days because trading days are more volatile compared to calendar days (Hull 328).

In using this closed form, we also need to solve for the drift and volatility. Since, the non-closed form of the stochastic equation, S has a log-normal distribution we can assume a series of log returns, r_i that are independently and identically distributed normally distributed random variables (Lid´en 15).

$$r_i = \log\left(\frac{S(t_i)}{S(t_{i-1})}\right) \sim Log \ Normal \ \left(\mu - \frac{1}{2}\sigma^2, \sigma^2 \Delta t\right)$$

As well, the sample variance is $S^2\mu$ and $\hat{\mu}$ is the sample mean. These two Maximum Likelihood Estimators below will be used in the model to simulate Tesla's stock price paths.

$$\hat{\mu} = \frac{\sum_{i=0}^{n} r_i}{n}$$

$$S_\mu^2 = \frac{1}{n-1} \sum_{i=0}^{n} (r_i - \hat{\mu})$$

The last part is to generate a random number within a normal distribution of mean 0 and standard deviation of 1, this is the Wiener process and will be further discussed in the computation algorithm section. After collecting all of the variables and creating a framework we can now implement our model into a C++ program.

Determining Error

It is clear to expect that a stock with higher volatility will have a larger return compared to a stock with less volatility, but would volatility affect the accuracy of a model? After a certain number of simulations have been executed from our program we can estimate the mean squared error below to find out how accurate our findings are compared to the historical data.

$$\sqrt{\frac{\sum_{i=0}^{n} (\hat{y}_i - y)^2}{n}}$$

Here, i is the number of paths up till n paths, $\hat{y}_i$ is the predicted stock price at time t and lastly, y is the historical stock price at time t. We expect the mean squared error to get smaller as n simulations increase.

Another way to measure the accuracy of GBM is to calculate the mean absolute percentage error. In the formulas below, t is the number of paths up till n paths, A_t is the actual stock price at time t and lastly, F_t is the estimated stock price at time t. From this statistical measurement, it is expected that an M percentage below 50% is a reasonable forecast and that below 20% is an even better approximation (Siti, Maheran 91).

Option Pricing with $M = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$ Black-Scholes-Merton

A further step one can take after simulating their Brownian

Motion is taking an average of their simulations at time T to estimate the price of a call or put option. We can calculate options using the equations below.

$$Call\ Price = max(ST - K, 0)e^{-rT}$$
$$Put\ Price = max(K - ST, 0)e^{-rT}$$

Here, ST is the stock price at time T and K is the strike price. r is the risk-free interest rate and the T is time at maturity. Because a call is a bet on the stock price going up the return will be $ST - K$ whereas a put is a bet on the stock price falling so it is $K - ST$. These equations can be check using the Black Scholes-Merton formula from Chapter 15 in the John C.Hull Option textbook. A model discovered in the 1970s by Fischer Black, Myron Scholes, and Robert Merton that is another form of the Put-Call Parity which will be defined later on.

$$C = S_0 \Phi(d_1) - Ke^{-rT} \Phi(d_2),$$

$$d_1 = \frac{\left(\ln\left(\frac{S_0}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)T\right)}{\sigma\sqrt{T}}, d_2 = d_1 - \sigma\sqrt{T},$$

Above is the Black-Scholes-Merton equation for a call option where C is the price of a call, S_0 is the price of the stock at time 0 and K is the strike price. r is the risk-free interest rate and T is the time at maturity. The z-score sign is a property of the Black Scholes-Merton that assumes short periods of time are normally distributed (Hull 322).

$$\frac{\Delta S}{S} \sim \phi(\mu \Delta t, \sigma^2 \Delta t)$$

From this equation, one can simplify it to a linear model called the Put-Call Parity which is "a principle that defines the relationship between the price of European put options and European call options of the same class, that is, with the same underlying asset, strike price and expiration date" (Investopedia). It is an important rule in the finance industry because it ensures zero arbitrage when pricing derivatives. If this equation failed then there is arbitrage.

$$C(K) + PV(K) = P(K) + S_0,$$

Computational Algorithm

Numerical Model Implementation Plan

The program is built in seven separate files. Two files create the *DoubArray* class and another two files make the *DoubMatrix* class. *DoubArray* is a class that creates arrays of type doubles that can perform mathematical operations such as addition and multiplication. The class also overloads the squared brackets to allow elemental access to position i when calling *array[i]*. *DoubMatrix* is a similar class to *DoubArray,* however, it collects data in a matrix and can perform tasks such as matrix multiplication. The main called *main.cpp* includes the main implementation function and other non-class functions*.* Lastly, the a class named Brownian emcompasses the Geometric Brownian Motion and Monte Carlo method that simulate any given stock's potential price level paths across a time period based on historical data.

The computation of Geometric Brownian Motion stock price simulation consists of data collection, parameter estimations, and stock price simulation. Additionally, the program incorporates essential functionalities to evaluate the appropriateness of GBM through MSE and further functions that price European options and estimate the value at risk of the stock at a 1% chance based on the simulations.

In the beginning, the main function calls *getData* to read and store historical stock prices in a *DoubArray*. Using the processed data, the function *log_return_rate* computes and fills an array of daily stock returns for all data.

The GBM process then starts by creating a *Brownian class* object through the default constructor with member variables of the mean and standard deviation of the stock in which it estimates through a member function *mu_sd_estimator*. This member function is built around the MLE estimators and takes the array of daily stock return and sets the member variables *m_mean* and *m_sd* of the Brownian class. With all variables computed, the program begins simulating the GBM stock price using Monte Carlo simulation for *n* stock price paths where each

path is consisted of *n* days through the member function *simulate*. This function uses nested for loops to achieve the effect of Monte Carlo simulation. The simulation generates many price paths that would reflect different market scenarios using inputs of the historical return rate, standard deviation, and from a random number generator with the standard normal distribution to be the shock described as the Wiener process. If the program runs as expected, it would return a *DoubMatrix* object and write the matrix to an output file as a database.

The matrix object stores simulated stock prices of each day of each path, each column is a path and each row is the next price. Afterward, the main function uses the outside function *AvgSt* to take the average of the simulated stock prices on the last day over all paths. The average end price of the stock is an input of an outside function *Option_Pricing* along with the risk-free interest rates, time to maturity, and option strike price to calculate the present value of the financial contracts. Note that the risk-free rate is the interest rate of a risk free asset such as 1 year Canada Government Bond.

Using the *AvgSt* (renamed as *avg*), we can now calculate the price of an European call with a strike price of $300, one-year maturity, and risk-free rate of 0.0144 with our *estimate_option_price* function. The function first reads if the argument *option* is true or false. If the variable equals 1 then it is a call, otherwise, a put option. Once the type of derivative is determined, it executes the option pricing equations mentioned previously and returns the value.

$$Call\ Price\ =\ max(ST-K,0)e^{-rT}$$
$$Put\ Price\ =\ max(K-ST,0)e^{-rT}$$

After the program calculates the call option, it is called again to calculate a put option with a strike price of $350, one-year maturity, and a risk-free rate of 0.0144.

The function *Mean_Squared_Error* is also called within the main function. Grabbing the matrix of GBM simulations and the historical price at day 1260 as arguments, it loops through the estimated stock prices at maturity and subtracts the actual price at maturity from each one. Then

summates all values to the power of 2 and divides itself by the number of simulations. Lastly, the value is square-rooted and returned at the end of the function.

The next function is *MAPE* that takes in a matrix of simulated stock prices and sums each final simulated price and historical price difference in a loop. It is then averaged and multiplies by 100 to return a percentage.

Another feature in the program is forecasting the worst loss possible-value at risk at 1% chance for investors who own one share of the stock, which can be applied to understand the potential loss for a larger investment. Recall that the Monte Carlo simulation provides various possible end prices of day N, the function *VaR* takes in the simulation matrix, and sort the end values to find the 1% lowest stock prices. The value at risk is returned as the difference between stock price today and the day N.

This program uses five libraries: fstream, iostream, random, array, and algorithm. The fstream library is used to read and write to files in addition to the terminal output. The iostream library enables the program to write output such as option prices, MSE, value at risk to the terminal directly. The random library helped the program to generate uniformly distribution numbers that follow a standard normal distribution that is needed in the GBM simulation. Finally, the array and algorithm libraries are used to create a resizable array with the ability to sort the elements that the VaR calculation needs.

Data

The data was obtained through NASDAQ's stock exchange on November 13th, 2018. The data includes Tesla's daily stock prices over the last 5 years, 252 trading days each year. **Figure 1** is a screenshot is the original format of the data file before removing its dates and headers to be imported to the program.

Results

The output below illustrated the successful implementation of the Geometric Brownian Motion through Monte Carlo simulation. The simulated prices and paths are written a file in a data frame format. The program is implemented twice, once with 500 stock price simulation, and the second time with 50 stock simulations. The terminal output **(Figure 2)** showed option prices and model errors in the 500 simulations implementation, the simulations are plotted with R in **Figure 3**, and the simulation matrix is written to a file in **Figure 4**.

Since graph of 500 simulations is extremely dense, we will analyzed **Figure 5** of 50 stock price simulations in this study. Base on the plots of 50 stock price simulations, we observed the majority fall within the price range of $200 - $600. There is a few outliers follow an upper price range above $800. Compared to the historical data, it only fluctuates between $250 - $350 which implies that GBM can capture the historical path, but assumes a lot of extreme values as well. Note that the starting price at day 0 is $315.40, and the average of the simulated price at day 252 is $389.48. The horizontal and slightly upward trends among the paths imply a log-normal return that has a longer tail on the right side of distribution of stock price level. This is expected because the constant quantity μ in the GBM model is positive although small. Overall, the volatile movements and trends from the simulation align with Tesla's stock price movement in the past year **(Figure 6)**, which is volatile but drops down to a price level with a slight gain of $23.

From **Figure 2** we can see that the price of a call option from our estimated data is $2.37542 which is a reasonable price, given that our option is one year long. Because a derivative is a financial tool that does not directly invest in the stock itself it would not be priced around the current stock price. Calculating the formula by hand we can verify that our calculation is correct and therefore the program's logic is accurate. Overall, our program suggests that purchasing a call on Tesla one year ago from today, we could have earned $Payoff = 2.37542e^{rT} = \$89.47$ where r is the 0.0144 risk-free rate and T is 252 days in a year. However, the stock moves in a random path and so GMB simulations cannot dictate how the stock will move.

The resulted put option price is $0 from **Figure 2**. This is a reasonable price because our strike price of $350 is less than our projected average stock price of $387.477 at time T, 252 days maturity. Calculating the formula by hand we can verify that our calculation is correct and therefore the program's logic is accurate. Overall, our program suggests that purchasing a put on Tesla one year ago from today, we would not have earned any payoff. Again, remember that the stock moves in a random path and so GMB simulations cannot dictate how the stock will move.

The mean square error **(Figure 2)** of $185.242 from 500 stock price simulations is relatively large compared to the stock value at $338 approximately. The potential reasons include the substantial errors in its outlier paths which valued the stock at more than $600 with few paths above the $1,200 point. Due to the large error size, one might consider GBM not an accurate measure of any particular price on a particular day. Moreover, the mean squared absolute percentage error is 39.1362%, which is within the reasonable forecast range determined by Siti and Maheran. Nevertheless, the plots demonstrated GBM as an appropriate measure of trends in short-term stock returns as both the actual and simulated paths show a volatile but flatter shape in them.

Further investment decision can be made using VaR **(Figure 2)**, which is a loss of $39.8956 from day 0 asset value at a 1% chance. This evaluation should be taken into consideration of market risk and capital adequacy in any portfolio manager's investment decision of Tesla. An appropriate strategy such as investing in bond-liked securities should be taken to hedge the high volatility of this stock. Speculators may use VaR to understand their potential loss and only buy in at a considerably lower price point such as $250 to avoid extreme loss.

Another step we took was to evaluate the program's logic by plotting the 50 simulated prices in a histogram. The graph is expected to resemble a log-normal distribution because one of the conditions of the Geometric Brownian Motion model S stock prices has a log-normal distribution. Observing **Figure 7** confirms that the program's logic is accurately built.

Lastly, we applied the method to a smaller company called Danaos whose stock price level is less volatile and less affected by the market movements in order to compare against the volatile stock Tesla. Danaos has been consistently falling for the last five years from $6 to $1 **(Figure 8).** From the terminal output of Danaos' 500 simulations over the last year in **Figure 9**, we observed that the simulated average end price of $1.14 is relatively close to the real stock price currently trading at $1. However, the MSE and MAPE are extremely high. The potential reasons are the constant drift and volatility parameters in the GBM method, which failed to reflect the actual drift and volatility of the stock last year. The high MSE and MAPE from Danaos' simulations suggested that the GBM is not an accurate model to represent future stock prices.
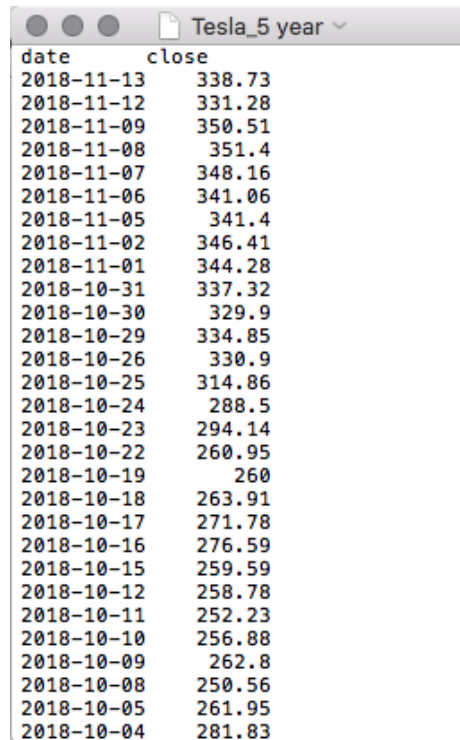
Conclusion

In conclusion, the Geometric Brownian Motion is a model that well-represents the random paths a stock price such as Tesla Motor's could take within a given time frame and can give asset managers a holistic view of their portfolio's risk. However, due to its assumptions of a constant mean as time t reaches maturity, it fails to recognize the assets' changing volatility and therefore cannot predict an accurate individual path and therefore, is not an appropriate model to estimate stock prices. Each simulation has an equal chance of happening. A calculated mean squared error of $185.242 and the mean absolute percentage error of 37.92% on the GMB model is another indicator that the GMB is not an accurate representation of Tesla's stock.

Other models such as Merton Jump Diffusion and Ornstein-Uhlenbeck Process could be similarly created within the program as a derived class from the *Brownian* class. Testing these models could lead to better stock price approximations, but overall, further research is needed to make better estimations.

Visuals and Graphs

Figure 1

Data. 5 Years Tesla Motor Stock Price (11-13-2008 to 11-13-2018)

date	close
2018-11-13	338.73
2018-11-12	331.28
2018-11-09	350.51
2018-11-08	351.4
2018-11-07	348.16
2018-11-06	341.06
2018-11-05	341.4
2018-11-02	346.41
2018-11-01	344.28
2018-10-31	337.32
2018-10-30	329.9
2018-10-29	334.85
2018-10-26	330.9
2018-10-25	314.86
2018-10-24	288.5
2018-10-23	294.14
2018-10-22	260.95
2018-10-19	260
2018-10-18	263.91
2018-10-17	271.78
2018-10-16	276.59
2018-10-15	259.59
2018-10-12	258.78
2018-10-11	252.23
2018-10-10	256.88
2018-10-09	262.8
2018-10-08	250.56
2018-10-05	261.95
2018-10-04	281.83

Source: Nasdaq

File name: Tesla_5 Year

Figure 2

*Terminal. Mac terminal output 500 stock simulations*

```
Minas-MacBook-Air:Project minaxxan$ g++ main.cpp DoubMatrix.cpp DoubArray.cpp Brownian.cpp -o test -std=c++11
Minas-MacBook-Air:Project minaxxan$ ./test
The average simulated ST: 389.477
Call Price: 2.37543
Put Price: 0
Mean Squared Error: 185.242
VaR at 1% probability: $-39.8956
Mean Absolute Percentage Error: 39.1362
```

Note: the terminal output includes the average of end prices across all simulations, the call option price, the put option prices, the MSE of simulations, and the VaR based on 500 simulations .

Figure 3

Plot. 500 GBM simulations on 1 year Tesla Motors Stock Price
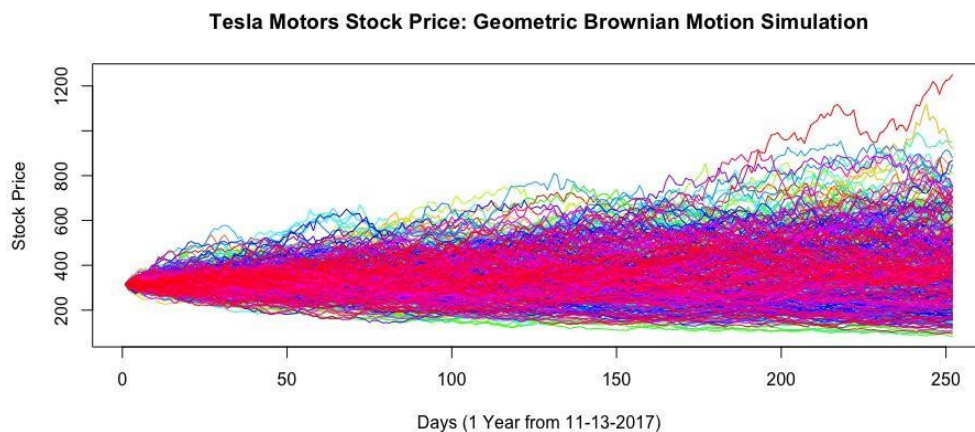
Source: file: Tesla_GBM2.dat

Note: Day 0 = 11/13/2017, Day 252 = 11/13/2018

Figure 4

*Data. Tesla stock price GBM simulations, 252 trading days, 50 simulations*

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
1	315.400	315.400	315.400	315.400	315.400	315.400	315.400	315.400	315.400	315.400	315.400	315.400
2	306.387	309.345	303.228	311.342	321.750	319.262	327.669	333.551	317.052	309.447	326.707	315.791
3	302.464	310.519	303.265	319.665	319.030	320.190	327.075	332.987	310.473	301.089	313.231	299.841
4	299.437	301.668	298.307	317.762	312.367	315.317	317.842	323.980	307.646	296.404	313.232	279.164
5	283.080	292.063	297.898	315.415	300.557	322.199	317.787	328.960	308.670	296.735	326.188	273.797
6	280.640	286.864	302.774	318.333	314.291	322.898	326.471	322.235	324.430	298.194	326.255	276.456
7	281.272	289.147	299.069	325.130	330.734	350.809	320.883	322.107	330.312	281.908	327.123	277.443
8	284.975	291.402	296.658	316.390	333.308	358.814	327.314	337.465	348.997	293.814	312.569	269.595
9	297.295	285.971	292.414	310.940	334.983	345.624	317.827	333.907	354.078	289.751	309.587	286.283
10	288.677	285.389	297.366	314.212	324.641	346.985	322.345	323.658	357.009	289.311	314.208	290.354

Showing 1 to 10 of 252 entries

Size: 252 rows (trading days), 50 columns (simulations)

Figure 5

Plot. 50 GBM simulations on 1 year Tesla Motors Stock Price
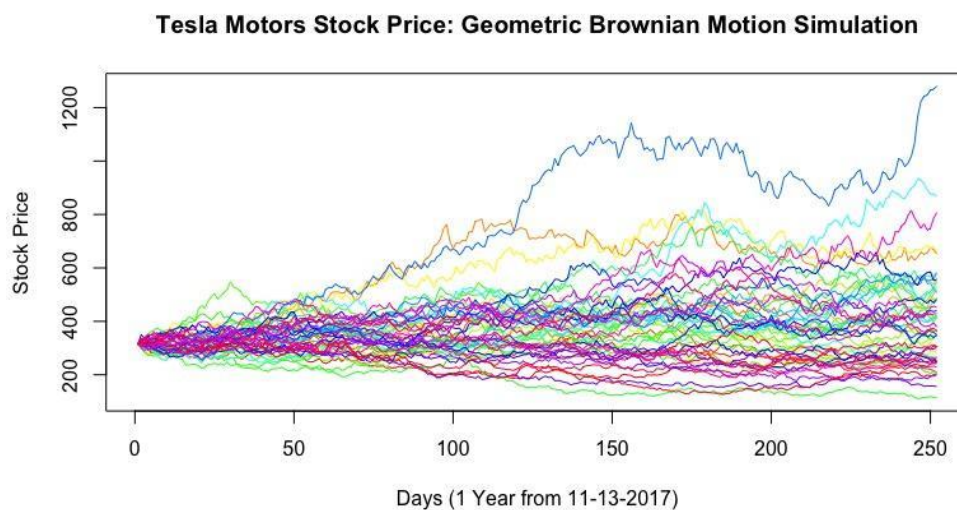
Source: file: Tesla_GBM.dat

Note: Day 0 = 11/13/2017, Day 252 = 11/13/2018

Figure 6

Plot. Actual Tesla Motors stock price in the past year

Figure 7

Plot. Histogram of 50 GBM simulations on 1 year Tesla Motors Stock Price
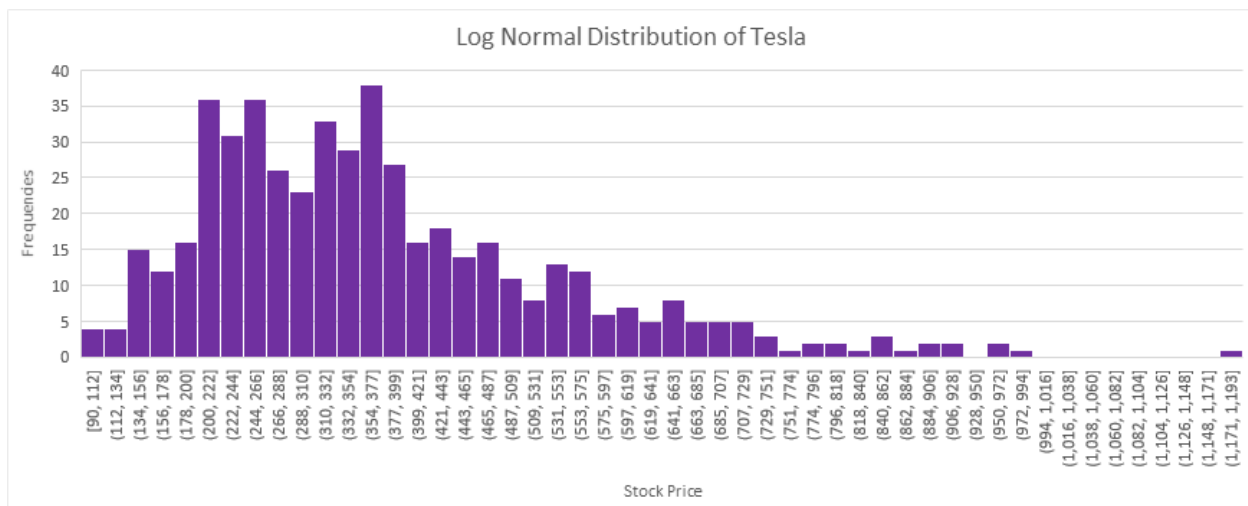
Source: simulation matrix file, plot with Excel

Note: Day 0 = 11/13/2017, Day 252 = 11/13/2018

Figure 8

Plot. 5 Year Historical stock prices of Danos

Source: Google

Figure 9

Terminal. Output of Danaos stock price simulation, 500 simulations

```
The average simulated ST: 1.14223
Mean Squared Error of Danaos: 349.369
Mean Absolute Percentage Error of Danaos: 99.6741
```

Bibliography

Abidin, Siti Nazifah Zainol, and Maheran Mohd Jaffar. A Review on Geometric Brownian Motion in Forcasting the Share Prices in Bursa Malaysia. IDOSI Publications, 2012, A Review on Geometric Brownian Motion in Forcasting the Share Prices in Bursa Malaysia.

Alejandro Mosiño and Alejandro Tatsuo Moreno-Okuno, (2018) ''On modeling fossil fuel prices: geometric Brownian motion vs. variance-gamma process'', Economics Bulletin, Volume 38, Issue 1, pages 509-519, http://www.accessecon.com/Pubs/EB/2018/Volume38/EB-18-V38-I1-P47.pdf

Bloomberg.com, Bloomberg, www.bloomberg.com/quote/TSLA:US.

"Danaos Corporation Common Stock (DAC) Quote & Summary Data." *NASDAQ.com*, Nasdaq, www.nasdaq.com/symbol/dac.

Davison, Matt. Quantitative Finance: a Simulation-Based Introduction Using Excel. CRC Press, 2014.

Dmouj, Abdelmoula. Stock Price Modelling: Theory and Practice. Vrije Universiteit, Nov. 2006, beta.vu.nl/nl/Images/werkstuk-dmouj_tcm235-91341.pdf?fbclid=IwAR0R_y_owzym755USuRMkwmh6rnQB5XT_TdFay1nhzL6m5-XN5iDsuVrdv4.

"European Vanilla Option Pricing with C++ and Analytic Formulae." Algorithmic Trading, Quantitative Trading, Trading Strategies, Backtesting and Implementation | QuantStart, QuarkGluon Ltd., 2018, www.quantstart.com/articles/European-vanilla-option-pricing-with-C-and-analytic-formulae?fbclid=IwAR2578vtOIQNTf6L-158QF6cUjOBeH3Ph2C6AfK--CkwIBftWwKPyMDKw6k.

Hull, John. Options, Futures, and Other Derivatives. Pearson Education Limited, 2018.

Marathe, Rahul R., and Sarah M. Ryan. *On the Validity of the Geometric Brownian Motion Assumption*. Iowa State University , www.public.iastate.edu/~smryan/gbmvalidv2.pdf.

"Mean Absolute Percentage Error." *Wikipedia*, Wikimedia Foundation, 12 Oct. 2018, en.wikipedia.org/wiki/Mean_absolute_percentage_error.

Lid´en, Joel. Stock Price Predictions Using a Geometric Brownian Motion. UPPSALA UNIVERSITET, June 2018, www.diva-portal.org/smash/get/diva2:1218088/FULLTEXT01.pdf?fbclid=IwAR0Gs2tEThsiJ3Jrtb1WCxLogahGF4ZQXTmEDVMPC85QRp0I7weTLW2gK4g.

"Robert Brown (Botanist, Born 1773)." Wikipedia, Wikimedia Foundation, 29 Nov. 2018, en.wikipedia.org/wiki/Robert_Brown_(botanist,_born_1773).

Ross, S. M. (2014). Variations on Brownian Motion. In Introduction to Probability Models (11th edition ed., pp. 612-14). Amsterdam: Elsevier.

Staff, Investopedia. "Put-Call Parity." *Investopedia*, Investopedia, 20 Oct. 2018,
    www.investopedia.com/terms/p/putcallparity.asp.

"Tesla, Inc. Common Stock (TSLA) Quote & Summary Data." *NASDAQ.com*, Nasdaq, 21 Nov. 2018,
    www.nasdaq.com/symbol/tsla.