Versatility in Monte-Carlo Simulations:

an Application Within Physics and Economics



By Peyton Capristo

Department of Physics

Allegheny College

April 2021

Versatility in Monte-Carlo Simulations an Application Within Physics and Economics

Allegheny College Senior Comprehensive Project

By: Peyton Capristo

Department of Physics

Allegheny College

Meadville, Pennsylvania

April 2021

Dr. James C. Lombardi Jr.	Date	Dr. Michael Michaelides	Date

Peyton Capristo

Acknowledgments

Dr. Lombardi - Thank you for all the help and support along the way to completing this comprehensive project. I know I got stuck plenty of times and gave me the guidance I needed. But the biggest thanks I can give is taking interest in a project that is based outside of your field of study.

Dr. Michaelides - Thank you for the advice and guidance throughout the project, and for continuing to push me to work harder.

The Brothers of Phi Kappa Psi Penn Beta - Thank you for all the love and support throughout the years without you I would not be where I am today. I owe many of you more than you ever know.

Family - Thank you for supporting me throughout my life.

Table of contents

0. Abstract……...…1

1. Introduction……2

2. Background……...3

2.1.0 Derivatives……..3

2.1.1 Forwards…….3

2.1.11 Hedging……...5

2.1.12 Speculation…….6

2.1.13 Arbitrage……....6

2.1.2 Futures……7

2.1.3 Options……....8

2.1.31 European vs American …….8

2.1.32 Call Option…….9

2.1.33 Put Option…….11

2.1.4 Parameters of Options……12

2.1.41 Stock Price……13

2.1.42 Strike Price……...13

2.1.43 Time to Expiration……...13

2.1.44 Volatility of Stock Price……...14

2.1.45 Risk Free Interest Rate……15

2.1.46 Dividends Expected to be Paid……...15

2.2.0 Stochastic Processes……16

2.2.1 Wiener Process……18

2.2.2 Itôs process……...21

2.3.0 Brownian Motion……22

2.4.0 Random Walk……..26

2.5.0 Black-Scholes Model……...31

2.6.0 Monte-Carlo Simulations……35

3. Methodology……39

3.1 Monte-Carlo Simulations……....39

3.2 Photon Moving Through Sun……40

3.3 Monte-Carlo Simulation for Options……46

3.4 Relationship Mapping……50

4. Results……...52

4.1 Photon Visual Code……..52

4.2 Photon Statistics Code…….....54

4.3 Time Elapsed Code……..55

4.4 Option Code…….....58

5. Conclusion……....65

6. Future Work……66

7. Citations……....67

8. Appendix……...68

8.1 1-D Random Walk Code……68

8.2 2-D Random Walk Code……68

8.3 3-D Random Walk Code……69

8.4 Photon Visual Code……70

8.5 Photon Statistics Code…….71

8.6 Time Elapsed Code……..73

8.7 Option Code…….74

List of Tables

Table 1 **Photon Statistics Values……...54**

Table 2 **Base Parameters Statistics Values……..60**

Table 3 **Base Parameters Black Scholes Values……..61**

Table 4 **Altered Parameters Statistics Values…….63**

List of Figures

Figure 1 **Profit of Call Option……...10**

Figure 2 **Profit of Put Option……....12**

Figure 3 **1D Random Walk Graph Example 1…….27**

Figure 4 **1D Random Walk Graph Example 2…….28**

Figure 5 **2D Random Walk Graph……29**

Figure 6 **3D Random Walk Graph……....30**

Figure 7 **Example Monte-Carlo Simulation 1……..36**

Figure 8 **Example Monte-Carlo Simulation 2……..37**

Figure 9 **Example Monte-Carlo Simulation 3……..38**

Figure 10 **Base Parameters……..47**

Figure 11 **Expected BSM Value Movements……..49**

Figure 12 **A Collection of Photon Movements……53**

Figure 13 **Time Displacement vs Distance……..56**

Figure 14 **Log of Time Displacement vs Log of Distance……..57**

Figure 15 **Base Parameter Option Paths…….59**

Figure 16 **Collection of Altered Parameters……...62**

Abstract

For this comprehensive project, Monte-Carlo simulations are applied to two different fields of study. Those fields consist of physics and economics. The physics simulation observes different attributes of a photon traveling through the sun. A visual walk, some statistics, and a question of time for this process to conclude will be analyzed. While the economics simulation explores Monte-Carlo simulations applied to european call and put options. The objective of this paper is to show how Monte-Carlo simulations are a method that can be used not just in a singular area of study. The programming language python is utilized to accomplish this goal. Results show a successful monte carlo simulations in both a physics and an economics application.

1.0 Introduction

There are a vast variety of applications for Monte-Carlo simulations. The method is not bound to one individual field of study. They utilize random sampling and mathematical models in order to obtain results from complex systems. This paper explores two specific applications of these simulations, one involving photons traversing the inside of the sun and another involving european call and put options in the financial sector.

Background Theory

2.1.0 Derivatives

Derivatives are a must learn for anyone who is planning on or working directly with the financial sector, and their importance in finance is continually increasing. “A derivative can be defined as a financial instrument whose value depends on (or derives from) the values of other, more basic, underlying variables” (Hull, 1). Most often the value of a derivative comes from the price of traded assets. However, the underlying variables that define the value of the derivative can be associated with any a whole range of determinants from prices of livestock to weather. Derivatives are used in hedging, speculation, and arbitrage. There are many different kinds of derivative markets. For example, there are insurance derivatives, credit derivatives, foreign exchange derivatives, the list has many more applications. It is worth noting that derivatives are not perfect, and received a great deal of criticism in 2007 for help causing a terrible recession. However, since then, more regulations have been added to help control the criticism. Derivatives can take the shape of many different forms but the most common are forwards, futures, and options.

2.1.1 Forwards

Typically consumers are accustomed to on-the-spot transactions, known as spot contracts. This kind of transaction is effective immediately upon agreement. The New York Stock Exchange is a spot market. An investor buys or sells a stock at a specified price immediately when the transaction is initiated. Forward contracts do not operate in this manner. How a forward contract differs is where the agreement is set to buy or sell an asset at some designated date in the future for a set price. The date that is set is known as the maturity date, and the set price is

known as the strike price. For notation purposes, the maturity date is designated by T, while the strike price is designated by S_T. The markets in which forwards are traded are OTC (over-the-counter) markets. Over-the-counter markets do not have a central market to go through. The participants in OTC markets just interact with each other. Therefore a forward contract is a private agreement. The main contributors to OTC are banks, large financial institutions, and corporations to name a few. Essentially how forwards work is that there are two parties, the buyer and seller. The buyer takes the long position and agrees to buy the asset at a set date in the future, while the seller takes the short position and agrees to sell the asset at the same time for the same price. A long position means that the consumer owns the assets. Whereas the short position is a sale of an asset that is not owned. Generally speaking the payoff from a forward contract, from the long position, is based on the strike price minus the delivery price. The delivery price is the fixed price that is set when a forward contract is made. For notation purposes the delivery price is designated by K. So for long position, the payout is equal to:

$$S_T - K.$$

This relation is due to the holder of the assets having an obligation to buy the asset worth S_T for K. For the short position, the opposite is true for the payoff, therefore payoff is equal to the delivery price minus the strike price:

$$K - S_T.$$

This relation is due to the seller of the assets having an obligation to sell the asset worth K for S_T. Due to there being zero transition costs associated with entering a forward contract, the payoff received for either position can result in a positive or negative gain. Forward contracts are used a

lot for currency exchange markets due to hedging. Every large company uses forwards to help better their institutions by trying to save money.

2.1.11, Hedging

Hedging uses derivatives in an effort to diminish the amount of risk that investors face against the possibility of market movement in the future. For example, a United States company could be receiving a shipment from a European company worth €1 million euros in 3 months, where payment is due on delivery. The value of the U.S. dollar compared to the European euro could differ greatly in 3 months. The U.S. company could hedge in an effort to reduce the risk of losing money due to currency exchange rates. So if when the agreement was made the value of the euro was expected to rise against the U.S. dollar the company would utilize this tactic. Looking at the numbers if the ratio was 1 when the agreement was made the U.S. company would purchase €1 million euros for $1 million american dollars. 3 months later if the ratio for euros to dollars is 1.5 then the U.S. company would owe $1.5 million dollars to the European Company based on the exchange. Since the U.S. company invested in the foreign currency change they had to pay only the equivalent of $1 million dollars, and therefore saved $500 thousand dollars. It is worth noting that the opposite effect can take place where if the currency rate goes the opposite direction then the U.S. company would end up paying more than if they didn't hedge. Foreign currency exchange is not the only form of hedging that takes place. Individual products all have markets of their own, so products like grain, glass, metal, and so on all can enter forward contracts, and companies can hedge in order to save money for the needed supplies. Essentially the objective is trying to avoid adverse movements for a price of an asset. Hedging is a way companies use derivatives, specifically forward contracts, to invest.

2.1.12, Speculation

Speculation is different from hedging in that hedging is trying to avoid exposure to negative price movements, speculation is taking an active position in the market. Speculation is all about betting that the price of an asset will either increase or decrease and capitalizing from that bet. An example of speculation is investors predicting the currency exchange markets. If said investors have strong evidence that one form of currency, say the euro, is going to increase or decrease in value against the dollar, they are going to try to capitalize on this interaction by buying the other currency and putting it in a location to earn interest.

2.1.13, Arbitrage

Arbitrage involves concurrently joining into transitions on two or more different markets that result in a riskless profit. To illustrate arbitrage, take a stock that is both traded in the New York Stock Exchange and the London Stock Exchange. The stock trading on the New York Stock exchange has a price of $140. While the same stock trading simultaneously on the London Stock Exchange has a price of £100. The exchange rate between dollars to pounds will be $1.4300 for this example. If an investor simultaneously buys 100 shares of the stock on the New York Stock Exchange then sells them on the London stock exchange, given the exchange rate the investor would receive a riskless profit of $300. This is how arbitrage works. (Hull, 16-17). Note that there is no transaction cost associated with this example. Transaction cost could very much alter the profit, especially for smaller investors. Larger investors such as banks are more likely to participate in arbitrage since their transaction cost associated with the deal is significantly lower

for both the stock market and the foreign exchange market. These types of transactions do not last long, due to supply and demand affecting the market. Once the stock is bought demand drives up the price and once the stock is sold the value is lowered. Eventually, due to this interaction, the two different prices on the different exchanges will equate to one another. Due to there being arbitrages who are chomping at the bit to take advantage of these transactions, it is safe to assume that very little of these transactions exist, and thus say there is zero opportunity for arbitrage in the market.

2.1.2, Futures

A future contract is very similar to a forward contract in that there is a maturity date and strike price agreed upon in the future between the buyer and seller. What differentiates futures from forwards though is futures are traded on an exchange while forwards are not. What an exchange does is specify certain standardized features on the future contract. In essence, it guarantees that both parties will honor the contract. The contract outlines what is the asset being traded, the quantity of the asset, when the delivery date is, and the location of the delivery. The assets could be financial such as stocks, bonds, currencies, and so forth. Or the asset can be physical goods such as livestock, precious metals, timber, and many others. These also differ from forwards since they are traded on exchanges. They are not traded on over-the-counter markets but rather specific markets such as the Chicago Board of Trade (CBOT) and the Chicago Mercantile Exchange (CME). If the asset is a physical good, otherwise known as a commodity, there may be differences in the quality of one item to the next. This has led to grades, better explained as the quality of a certain commodity needing to reach certain parameters in order to obtain the price. It is important that future contracts for commodities specify the grade of the

product if necessary. The quantity that is traded also needs to be specified since if the quantity is too many investors may consider hedging or speculating smaller positions and be unable to use the exchange. On the contrary, if the quantity is too low, the cost associated with trading may be too expensive and not worth it. The delivery agreement with futures is necessary because it factors shipping expenses into the negotiation. As the delivery date approaches the future price of the asset starts to converge to the spot price of the asset. This relation happens since there is an arbitrage opportunity with future prices and spot prices that differ, and supply and demand forces will equal them out.

2.1.3, Options

Options are similar to forward and future contracts where there is a maturity date and strike price. The difference between them, however, is options give the holder the right to do something. Essentially this is different from the obligation since the holder does not have to trade the asset at the strike price at the maturity date wherein futures and forwards the holder has to. The trader can instead trade it at market value. It is worth noting that options have a cost associated with them in entering the agreement where forwards and futures do not. Options are traded in both over-the-counter markets and in exchange markets. The two main types of options are call and put options. In addition, there is the classification of the option. Is the option a European or American option? The future of the payouts of the asset associated with options depends on these classifications.

2.1.31, European vs American

European Options are options that the decision to trade the assets can be exercised only at the maturity date. This is different from American options since the decision to trade the assets can be made at any time before the maturity date. European vs American options have nothing to do with geographical location. The majority of options that are traded in exchange markets are american-style options. However, these options are much harder to analyze than their European counterparts. This is due to the flexibility with decision-making for the American options. If investors of American options chose to exercise the right to sell the assets before the maturity date, they run the risk of losing profits if the price continues to rise. In addition, if they wait too long, they could end up losing money due to the price dropping. This leads American options significantly harder to model compared to European options. This interaction has led to properties of American options stemming from European options in an effort to better analyze them.

2.1.32, Call options

"A call option gives the holder the right to buy the underlying asset by a certain date for a certain price" (Hull). What does a call option look like? For simplification purposes assume there is a European option being traded. The option has a strike price specified at $100 and a maturity date of 3 months. In addition, the current stock price of the asset is $98. Next, the price of the option to purchase one share of the asset is $5. Normally the asset is bundled in lots, meaning there is a specified number of stocks that need to be purchased with the option. Generally, 100 shares are traded in lots. This is the contract size of the option. So, the investor purchases 100 shares of the asset for $5 each. Therefore the initial investment is $500. Since the option is European, the investor can not exercise the option before the maturity date. So 3 months of time

passes and the investor now has to make a decision on whether or whether not to exercise the option. If the investor chooses to exercise the option, they are able to buy the stock regardless of its new price for the specified strike price when the option was sold. So if the new price of the assets goes below $100, the investor will not exercise the option and will be out the initial investment of $500. If the asset price increases relative to the initial price, the investor will exercise the option since they can buy the assets for a lower value than it is currently traded on the market. How investors profit from call options assumes the same example but the stock price increases to $115 from $98. The investor would exercise the options since they can purchase 100 shares of the asset for $100 instead of $115. Then if the investor immediately sells the shares it's a profit of $15 per share, or $1500 total, assuming there is zero transaction cost. Even including the initial investment of $500 the net profit is still $1,000. This example can be graphically shown to see where an investor is losing and gaining money based on the current price.

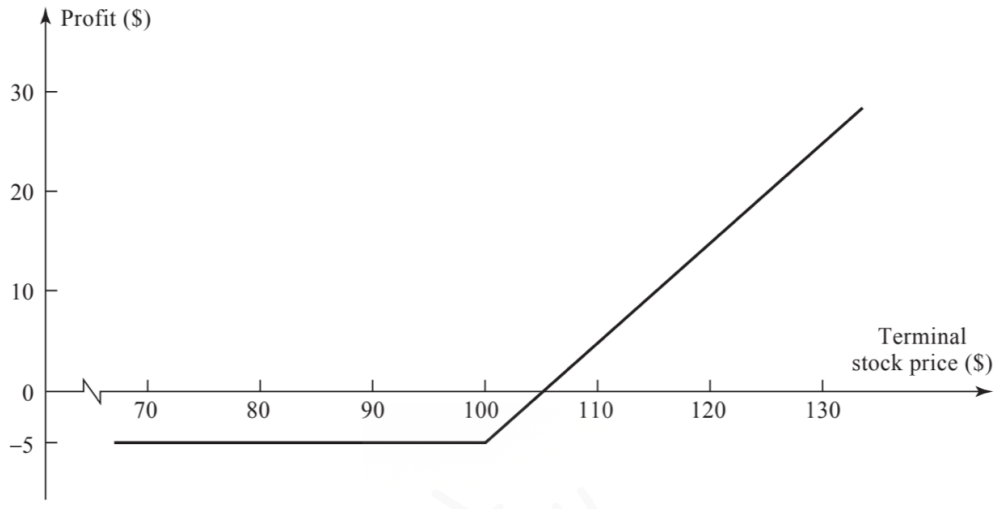


Figure 1 **Profit of Call Option**: The x-axis is in dollars of stock price. The y-axis is in dollars of the profit earned. This graph shows the profit of buying a European on one share of stock. The option price is $5, and the strike price is $100. (Hull, 210)”

The investor is not making any money until the graph crosses the horizontal axis.

2.1.33, Put option

"A put option gives the holder the right to sell the underlying asset by a certain date for a certain price" (Hull). So the main difference is that in a call option the investor is hoping the price to increase while in the put option the investor wants the price to decrease. What does a put option look like? Again the option will be European to simplify the model. The option has a strike price specified at $70 and a maturity date of 3 months. In addition, the current stock price of the asset is $65. Next, the price of the option to sell one share of the asset is $7. Again a lot of 100 shares are purchased by the investor at $7 each. Therefore the initial investment is $700. Since the option is European, the investor can not exercise the option before the maturity date. So 3 months of time passes and the investor now has to make a decision on whether or whether not to exercise the option. If the investor chooses to exercise the option, they are able to buy the stock regardless of its new price for the specified strike price when the option was sold. So if the new price of the assets goes above $70, the investor will not exercise the option and will be out the initial investment of $700. If the asset price decreases relative to the initial price, the investor will exercise the option since they can buy the assets at market value and sell the asset for a higher value than it is currently traded on the market. How investors profit from put options assumes the same example but the stock price decreases from $65 to $55. The investor will buy 100 shares of the stock at $55 and sell it at the strike price of $70 to earn a profit of $15 per share, or $1500 total, assuming there is zero transaction cost. Even including the initial

investment of $700 the net profit is still $800. This example can be graphically shown to see where an investor is losing and gaining money based on the current price.

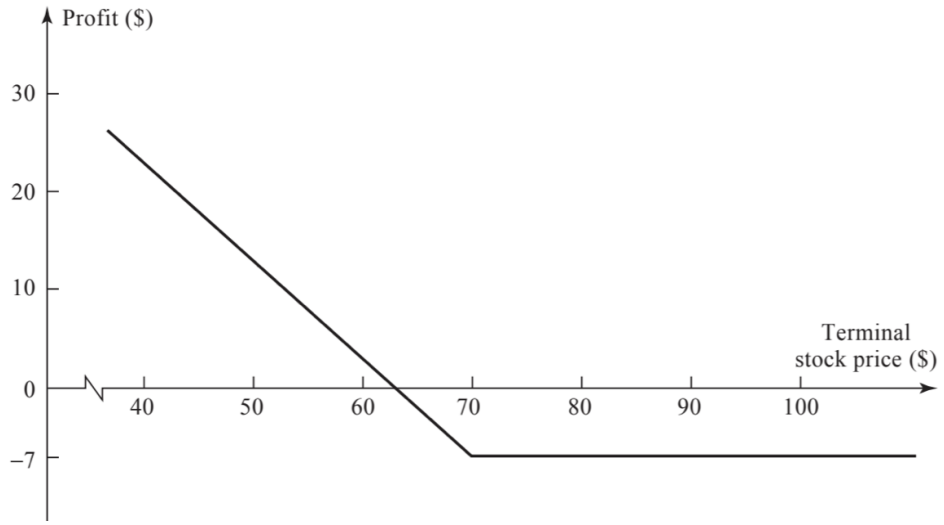


Figure 2 **Profit of Put Option**: The x-axis is in dollars of stock price. The y-axis is in dollars of the profit earned. This graph shows the profit of buying a European put option on one share of stock. The option price is $7, and the strike price is $70" (Hull, 211).

The investor is making money until the graph crosses the horizontal axis.

2.1.4 Parameters of options

There are many different parameters that affect how options move. The components that comprise said parameters include the current stock price, the current strike price, the time to expiration, the volatility of the stock price, the risk interest rate, and the dividends that will be paid. The notation for each of the following needs to be defined. The current stock price will be represented by S_o. The strike price was defined earlier and will not change, which is S_t. Notice t is lowercase because it represents a set time. The time to expiration is represented by the variable

T. $T$is capital because it changes as the time to the maturity date changes. The volatility of the stock price is designated by the Greek letter σ. The risk-free interest rate is designated by R_f, and lastly, there is no need to represent the dividends that need to be paid since those are not an obligation and can not be accurately modeled, yet can affect the price of the stock option that is being observed.

2.1.41, Stock Price, S_o

The stock price is very self-explanatory, it represents the current value of whatever the stock is. In call options, assuming all other variables remain constant, if the stock price increases the price of the call option will increase regardless if it is a European or American option. The opposite relationship is true if looking at a put option. If the stock price decreases the price of the put option will decrease regardless if the option is European or American.

2.1.42, Strike Price, S_t

The strike price was defined earlier in section 1.1. The relation between the strike price and the stock price in the pricing of options is the opposite. In call options, assuming all other variables remain constant, if the strike price increases the price of the call option will decrease regardless if it is a European or American option. While looking at a put option, if the stock price decreases the price of the put option will increase regardless if the option is European or American.

2.1.43, Time to expiration, T

The time to expiration is how long the stock has until the maturity date is reached. The classification on if the option is American or European has importance within this parameter. Starting with American call options, the value of the option is observed to not decrease in value as the time to expiration increases for both put and call options. This is in part due to the ability of the owner of the asset to have the right to exercise the option. Observing two different options, one with a long time to expiration and one with a short time to expiration. The long-time option has, at the minimum, the same amount of time to exercise the option as the short option. Therefore, the longer option has to be just as valuable as the shorter option. Observing European options, the identity is unknown. It is expected for both put and call options that the value of the option should increase as the time to maturity increases however this is not necessarily the case in practice. A way this relationship can be explained is through there is no exercising right for the asset holder, as well as if dividends are planned on being paid. Both these could cause the price to decrease or increase as the time to expiration increases. So, the relation for European options can not be defined through T.

2.1.44, Volatility of Stock Price, σ

"The volatility, σ, of a stock is a measure of our uncertainty about the returns provided by the stock" (Hull, 323). In other words, as the volatility of a stock increases the more the stock is susceptible to increased price fluctuations. With higher volatility, the stock can perform very well or very poorly. If the volatility is low the more likely the stock will not fluctuate in price. Regardless of european or american options, how volatility affects call and put options, is different than normal stocks. For call and put options volatility is considered a positive effect on the options. This is because with options the most the owner can lose is the price of the stock. So

for call options, the owner benefits if the price increases, and for put options, the owner benefits if the price decreases. Therefore traders of assets welcome volatility since it increases the value of the option while offering lower losses than trading away straight stocks, since the risk of losing money is inherently lower.

2.1.45, Risk-Free Interest Rate, R_f

The risk-free interest rate is considered to be the interest rate on an asset that has zero risk associated with it. Financial managers use risk-free assets in portfolios in order to generate higher returns within the portfolio. When combined with other assets that contain a higher risk than the R_f value, portfolios then generate higher returns. When observing the risk-free interest rate for options, the effect on the value of options is not as defined. "As interest rates in the economy increase, the expected return required by investors from the stock tends to increase. In addition, the present value of any future cash flow received by the holder of the option decreases. The combined impact of these two effects is to increase the value of call options and decrease the value of put options" (Hull, 234). However, assuming all other variables remain constant, "In practice, when interest rates rise (fall), stock prices tend to fall (rise). The combined effect of an interest rate increase and the accompanying stock price decrease can be to decrease the value of a call option and increase the value of a put option. Similarly, the combined effect of an interest rate decrease and the accompanying stock price increase can be to increase the value of a call option and decrease the value of a put option." (Hull, 234) This relationship is found for both European and American options.

2.1.46, Dividends expected to be paid.

Dividends of stocks are expected to be paid on the ex-dividend date. The ex-dividend date is the day a company sets that dividends are expected to be paid for their stockholders. With options, the paying of dividends decreases the value of call options and increases the value of put options, regardless if the option is European or American. This is in part because options do not receive dividend payments. The payment size and quantity also play a role in how the value of options is affected.

2.2.0, Stochastic Processes

Stochastic processes have limitless applications for what they are used in. Essentially, a stochastic process observes the relation of randomness in a system that changes over time. In fact, Stochastic, in its very essence is just jargon for random. A stochastic process "can describe such a system by defining a family of random variables, $\{X_t\}$, where X_t measures, at time t, the aspect of the system which is of interest" (Coleman, 1). A very basic example is observing the number of customers in a store throughout some observed time Δt. (Delta representing a change in). At any point in time t the value of X_t can be any whole number. There may be some reasons that draw customers to stores at a specific time, but overall the number of customers in the store at a time, t, is random. There are two types of timed stochastic processes. Those two types are discrete-time and continuous-time. A discrete-time stochastic process is a stochastic process where only at certain fixed points in time can the value of the variable change, where in contrast to a continuous-time stochastic process where a change in the variable can take place at any time in the process. Not only are there differences in time models for stochastic processes there are also two different variable classifications. Those types are discrete variables and continuous variables. In a discrete-variable stochastic process, only certain discrete underlying values are

possible, whereas, in a continuous-variable stochastic process, the variable is able to take any value within a certain range defined by the model. It is worth mentioning that a Markov process is a kind of stochastic process where the future value of the variable is only predicted from the value of the current variable. The history of the data associated with the variable does not affect the future, as well as the path the data takes, is independent of future movements.

Physics, at its core, is the fundamental study of how the universe works. Stochastic processes occur throughout many different physical applications, but specifically in the movement of particles that undergo random walk from various collisions. This property occurs in astrophysics how photons travel from the core of the sun to the sun's convection zone by colliding with other atoms, as well as in biophysics when particles move through a variety of fluids. Later, the stochastic process of how a photon travels through the sun will be modeled. Due to the random nature at the core of the fundamental universe, the applications of stochastic processes venture out into many different fields of study. In fact, how the stock market behaves is fundamentally observed through a stochastic process. Due to this, physical applications can be modeled into trying to determine future outcomes of the stock market. This has led to the growing field of econophysics, which observes this very nature. When observing the stock market, essentially it falls in line with chaos theory. Chaos theory is shown in complex systems where there is an array of randomness that is determined by a number of underlying attributes, repetition, and links to other systems. While specifically observing the stock market, any change in information can cause drastic changes in prices. For example, the volatility is affected by anything from a change in grain prices, to new political news, too far and few in between. In fact, “chaos theory has shown that unpredictable time series can arise from deterministic nonlinear systems” (Mantegna & Stanley, 5). The family of random variables, however, does not

necessarily need a direct tie to physics. As briefly mentioned earlier, a family of random variables can be observed in times customers enter stores, how a gambler loses and gains money, essentially if there is a random nature in the system a stochastic process is represented. Additionally, if accurately modeled these processes can help us understand the real world better than if the randomness was chosen to be excluded (Coleman, 1). There are different types of stochastic processes but in essence, random walk and Brownian motion are essential to stochastic processes. A stochastic process is known as the Wiener process, which utilizes continual time, uses Brownian motion to observe stock prices.

2.2.1, Wiener Process

A Wiener process is a continuous-time Markov process with a normal distribution applied (the distribution implies a mean change of zero and a variance rate of 1.0). Wiener processes are used in physics to describe how a particle moves that undergoes a large number of small molecular shocks and is sometimes referred to as Brownian motion "Expressed formally, a variable z follows a Wiener process if it has the following two properties:

Property 1. The change z during a small period of time Δt is

$$\Delta z = \epsilon\sqrt{\Delta t}$$

where ϵ has a standard normal distribution $\phi(0, 1)$.

Property 2. The values of Δz for any two different short intervals of time, Δt, are independent.

It follows from the first property that z itself has a normal distribution with

$$\text{Mean of } \Delta z = 0$$

$$\text{Standard Deviation of } \Delta z = \sqrt{\Delta t}$$

$$\text{variance of } \Delta z = \Delta t$$

The second property implies that z follows a Markov process" (Hull, 302). If the model is changed from a short change in time to a long change in time,T, for the value of z. This is expressed by $z(T) - z(0)$. The long model time model has to express the sum of the changes in z in N small changes in time of length Δt, where

$$N = \frac{T}{\Delta t}$$

This means that

$$z(T) - z(0) = \sum_{i=1}^{N} \epsilon_i \sqrt{\Delta t} \qquad 2.2.1$$

Where ϵ_i $(i = 1, 2, \ldots, N)$is normally distributed between 0 and 1 ($\Phi(0, 1)$). Given the second property of the Wiener process the $\epsilon_i$are independent of one another. Given this similar expressions can be made about the mean, variance, and standard deviation from equation 2.2.1

$$\text{Mean of } [z(T) - z(0)] = 0$$

$$\text{Standard Deviation of } [z(T) - z(0)] = \sqrt{T}$$

$$\text{variance of } [z(T) - z(0)] = N\Delta t = T$$

Notice how the equations are consistent. "The mean change per unit time for a stochastic process is known as the drift rate, and the variance per unit time is known as the variance rate" (Hull, 303). The Wiener process above makes assumptions that there is a zero drift rate and a variance of 1.0. Zero drift rate implies there is zero change in the value of $z$for future returns. A 1.0 variance implies that the variance in a change in z for a time $T$is equal to T. This might not be the case for every process so to generalize this the Wiener process takes the shape of

$$dx = a\,dt + b\,dz \qquad 2.2.11$$

Equation 2.2.11 is the generalized Wiener process. Not that the variables $a$and b are both constants. x has an expected drift rate of a per unit of time, this is represented by the $a\,dt$ term. If the second term of equation 2.2.11, $b\,dz$, was removed. The equation becomes

$$dx = a\,dt$$

This means that

$$\frac{dx}{dt} = a$$

Integrating this with respect to time becomes

$$x = x_0 + at$$

The $b\,dz$ term represents noise in the system for any path x. Combining the $b$term to the Wiener process gives the amount of noise in the system. The constant b times a Wiener process the variance rate becomes $b^2$for a certain rate of unit time. "In a small time interval t, the change x in the value of x is given by equations (14.1) and (14.3) as" (Hull, 305).

$$\Delta x = a\Delta t + b\epsilon\sqrt{\Delta t}$$

$\epsilon$is normally distributed between 0 and 1. Therefore Δx must have a normal distribution

$$\text{Mean of } \Delta x = a\Delta t$$

$$\text{Standard Deviation of } \Delta x = b\sqrt{\Delta t}$$

$$\text{variance of } \Delta x = b^2\Delta t$$

The generalized Wiener process can make similar arguments to the Wiener process above where the normal distribution for a change in x caused by T

$$\text{Mean of change in} x = aT$$

$$\text{Standard Deviation of change in } x = b\sqrt{T}$$

$$\text{variance of change in } x = b^2T$$

Overall the generalized wiener process express the same properties of distribution, as well as enforcing the fact that there is an expected drift rate of a, a well as a variance rate of b^2

2.2.2, Itôs process

An Itôs process is just a generalized Wiener process. The equation is expressed as

$$dx = a(x,t)dt + b(x,t)dz \quad 2.2.21$$

Here the difference is a and b are expressed as functions of x and t. "Both the expected drift rate and variance rate of an Itô process are liable to change over time. They are functions of the current value of x and the current time, t. In a small time interval between t and $t + \Delta t$, the variable changes from x to $x + \Delta x$, where

$$\Delta x = a(x,t)\Delta t + b(x,t)\epsilon\sqrt{\Delta t}$$

This equation involves a small approximation. It assumes that the drift and variance rate of x remain constant, equal to their values at time t, in the time interval between t and $t + \Delta t$" (Hull, 306). Note that the history is not taken into account only the current variables. The mathematician Kiyoshi Itô was the founder of these processes.

In 1951 he was able to determine an underlying equation to find the differential of a time-dependent stochastic process. Itô's lemma expresses how a function G of x and t follows the differential equation

$$dG = \left(\frac{\delta G}{\delta x}a + \frac{\delta G}{\delta t} + \frac{1}{2}\frac{\delta^2 G}{\delta x^2}b^2\right)dt + \frac{\delta G}{\delta x}bdz \quad 2.2.22$$

This equation is very similar to equation 2.2.21 thus a function G follows an Itô's process. Therefore, this relationship explains how dz represents a Wiener process. In addition, the drift rate can be deduced as

$$\frac{\delta G}{\delta x}a \ + \ \frac{\delta G}{\delta t} \ + \ \frac{1}{2}\frac{\delta^2 G}{\delta x^2}b^2$$

As well as the variance rate to be defined at

$$(\frac{\delta G}{\delta x})^2 b^2$$

Later Itô's limit will be directly applied to asset pricing.

2.3.0, Brownian Motion

Brownian motion was first observed by a botanist named Robert Brown in 1828. He was able to observe how pollen had a random movement when dispersed in water, and thus Brownian motion was born. In better words, Brownian motion happens when an object undergoes small random fluctuations. A basic visual example of what Brownian motion looks like is the movement of heavily intoxicated people, as well as sharks. A heavily intoxicated person has trouble walking in a straight line. The movement of the steps is not consistent and randomly fluctuates over distance and time in small repetitive increments. Since the discovery, the field of Brownian motion has been expanded and used for modeling many different aspects than that of pollen. Brownian motion has been expanded to include "modeling of stock prices, of thermal noise in electrical circuits, of certain limiting behavior in queueing and inventory systems, and of random perturbations in a variety of other physical, biological, economic, and management systems" (Karatzas & Shreve, 47).

A famous household name of science, Albert Einstein, really brought forward the idea of Brownian motion. He was able to show that Brownian motion in one dimension follows the equation

$$[\Delta x(t)]^2 = 2Dt$$

For this equation $[\Delta x(t)]^2$ is the mean squared displacement, or in other words, the difference in the position against time is equal to some diffusion constant D multiplied by time. This application was directly related to thermal physics in how a suspended particle travels through fluids due to the thermal properties of the fluid. From there the diffusion constant was calculated through means by the Boltzmann constant k_b, temperature$T$, and Stokes friction coefficient γ. Shown,

$$D = \frac{k_b T}{\gamma}$$

The diffusion coefficient exists for all Brownian motion problems. The diffusion coefficient represents the distribution of the randomness and is an integral part of the properties behind Brownian motion. The diffusion coefficient is not just shown with particle distributions. Einstein was aware that this diffusive expression does not hold true for small time scales. Fast forward and Paul Langevin was able to revise Einstein's Brownian motion equation. He devised a stochastic differential equation that is known as the Langevin equation (Mo). Today the standard Langevin equation for the velocity of a Brownian motion particle is represented by

$$mv = -\gamma v + \sqrt{2D}\epsilon(t)$$

m is represented of the mass, v is represented of velocity, γ and D are the same as above, and ϵ is represented by a stochastic process. The key point of the Langevin equation is that there is a drag force, a restoring force, and a stochastic force. [iopscience]

As mentioned earlier Brownian motion is an integral part of how the stock market's movement behaves. As shown in physics, a model can be created to solve the movement of Brownian motion. For this, we need to accurately model the process of a stock price. It is tantalizing to assume that the stock market behaves as a generalized Wiener process. This assumption however is intrinsically flawed. The major problem is that there is an assumption of a

constant drift rate. In other words, "the expected percentage return required by investors from a stock is independent of the stock's price. If investors require a 14% per annum expected return when the stock price is \$10, then, *ceteris paribus*, they will also require a 14% per annum expected return when it is \$50" (Hull, 306). A better assumption to model to replace this flaw in the model is that the expected return is constant, not the drift rate. In other words, the expected return is equal to the expected drift rate divided by the stock price. Observing notation, S is representative of a stock price at a certain time t. The expected drift rate for S is assumed to move by some constant parameter. The constant parameter will be represented by the Greek letter μ. Therefore the expected drift rate should be equal to μS. What this means is given a short change in a time interval, Δt, the expected increase for S is $\mu S \Delta t$. The constant parameter μ is not some arbitrary constant but rather the expected rate of return on the stock. If there is no uncertainty included in the model, the dz term goes to 0. This shows that

$$\Delta S = \mu S \Delta t$$

Taking the limit of this equation as $\Delta t \to 0$, shows that,

$$dS = \mu S dt$$

Note that the limit is applied to take a very short time interval so the $\Delta t$is essentially equal to zero. Moving the Sterm to the other side gives

$$\frac{dS}{S} = \mu dt$$

Since, assuming zero uncertainty, the model needs to show the stock price growing at a continuously compounded rate of μ per unit of time. To do this the integral of the above equation needs to be taken from bounds between 0 and time T. These bounds are specified since the model shows a change in time from start to finish.

$$S_T = S_0 e^{\mu T} \qquad 2.3.1$$

Worth noting that S_T and S_0 represent the stock price at time T and 0 respectively for equation 2.3.1. Although the above equation is a good model it is obvious that uncertainty is embedded in the pricing of stocks. "A reasonable assumption is that the variability of the return in a short period of time, Δt, is the same regardless of the stock price. In other words, an investor is just as uncertain about the return when the stock price is \$50 as when it is \$10. This suggests that the standard deviation of the change in a short period of time Δt should be proportional to the stock price and leads to the model" (Hull, 307)

$$dS = \mu S dt + \sigma S dz \quad (2.3.11)$$

Moving *S* to the other side gives,

$$\frac{dS}{S} = \mu dt + \sigma dz \quad (2.3.2)$$

For notation purposes, The stock's expected rate of return is represented by μ, the volatility of the stock price is defined by σ, and (although not shown in equation 2.3.2) the variable σ^2 is referred to as its variance rate. Equation 2.3.2 is the most utilized model by the industry for stock pricing. This is due to the model making assumptions that represent real-world behavior, specifically in a world that involves risk. The model that was developed is considered geometric Brownian motion. Geometric Brownian motion is defined as a continuous-time stochastic process in which the logarithmic of the randomly varying load reveals drift in Brownian motion. If we want to observe the model for discrete-time rather than continuous-time, the model changes to

$$\frac{\Delta S}{S} = \mu \Delta t + \sigma \epsilon \sqrt{\Delta t} \quad (2.3.3)$$

Moving the *S* to the other side gives

$$\Delta S = \mu S \Delta t + \sigma S \epsilon \sqrt{\Delta t} \quad (2.3.4)$$

Understand the notation ΔS is defined as the change in the price of the stock, Δt is defined as a change in time (more so a small change in time for this model), ϵ has a standard normal distribution, μ defined as the expected rate of return for the given time interval, and lastly, $\sigma$is the volatility in the stock price. For the rest of the model, these parameters will be constants. Understanding equation 2.3.3 better, each combination of variables represents another parameter. The $\frac{\Delta S}{S}$is the discrete approximation of the return of the stock for the specified time parameter Δt. The $\mu\Delta t$ term represents the expected value of the return. The $\sigma\epsilon\sqrt{\Delta t}$ term is the stochastic component in the model. σ^2 was referenced earlier as the variance rate, when combed with a time variable Δt, $\sigma^2\Delta t$ represents the variability in a stochastic component. Since the variability is in the stochastic component it applies to the whole return. So another way to model this is

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

In other words, the discrete approximation of stock returns is normally distributed with respect to the expected value and the variability.

2.4.0, Random Walk

Random walk is synonymous with Brownian motion, yet there is a key difference. Both Brownian motion and random walk are a form of stochastic processes. Where they differ from each other is as Browinina motion is continuous-time with continuous space, random walk is discrete-time in discrete space. It is worth noting that Brownian motion models are derived from random walk models. Essentially random walk observes random changes in a variable that are not correlated. Random walk has applications in a variety of fields, yet in physics, random walk is observed mainly through how atomic particles travel. An example of this is the path a photon

travels from the core of the sun to the sun's convection layer. This process takes some time between 10,000 and 170,000years[10]. Later, I will use a Monte-Carlo Simulation to observe how this interacts. In economics, random walk is shown in how markets move, although Brownian motion is better to use in a model.

Random walks can be shown in every dimension. A 1-D random walk shows a change in movement only along one direction. An example of one dimension random walk is the stock market since it moves along one dimension, a price dimension, relative to time. With the help of Dr. Lombardi, a python code was created to visually show a random walk in one direction (code is shown in the appendix). A movement along x is shown on a graph against time.

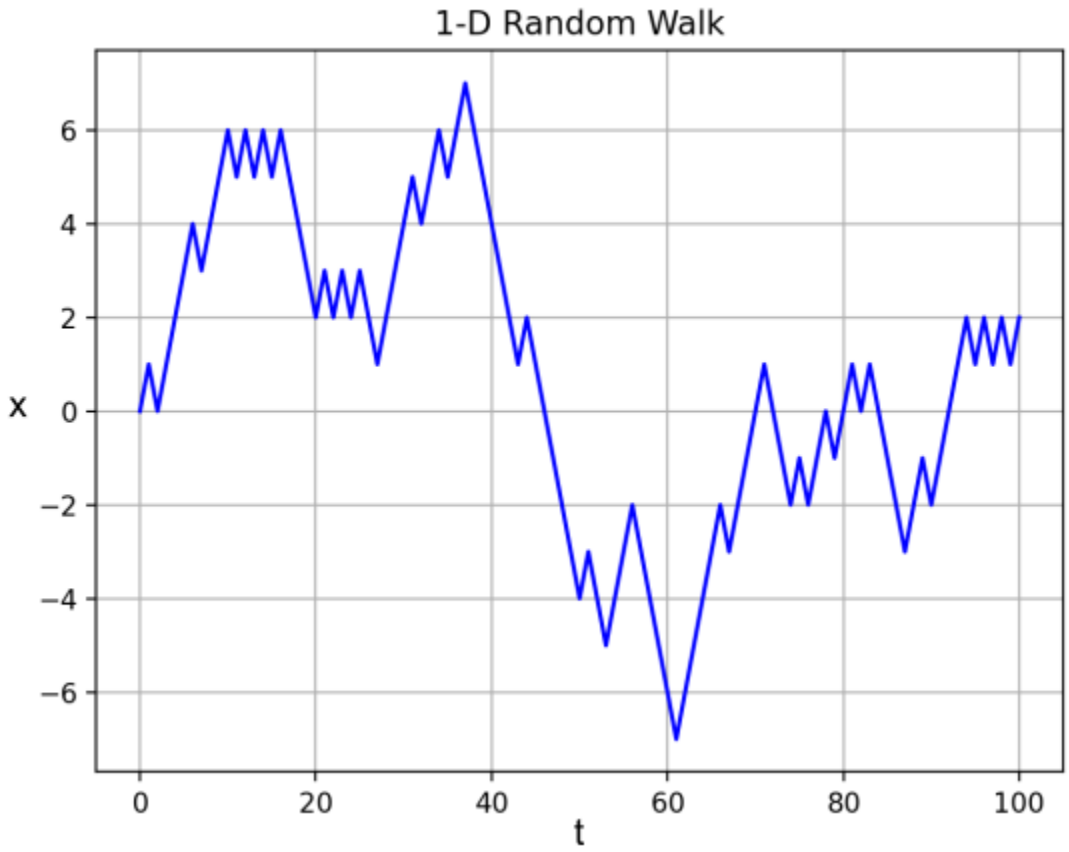


Figure 3 **1D Random Walk Graph Example 1**: Trial 1 for a diagram of an arbitrary one-dimensional random walk vs time. It is shown how as time increases the value of *x* can only move up or down, there is no correlated data behind this movement.

To show the process is random when the simulation is run again a new graph will be created.

(Note if the code had a random number seed, the graphs would be the same.)

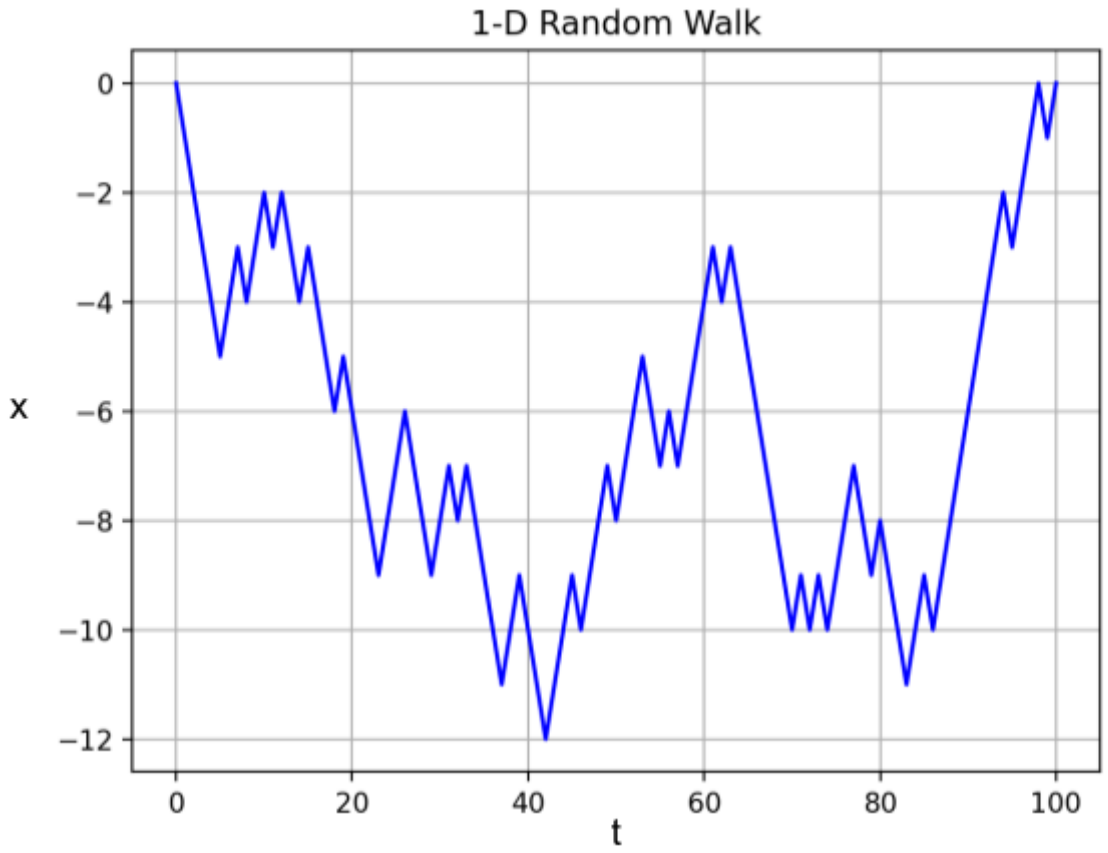


Figure 4 **1D Random Walk Graph Example 2**: Trial 2 of a diagram of an arbitrary one-dimensional random walk vs time. It is shown how as time increases the value of x can only move up or down. There is no correlated data behind this movement.

Notice how the graphs differ since there is nothing bounding the data to specific movements; the direction they move in the x dimension is random with time.

An example of a random walk in 2-D can be represented by how a pigeon walks along the ground, or an intoxicated person gets from point a and point b. Instead of being bound by one axis, there is now another dimension that defines the movement. Random walk in 2-D can be represented in different coordinate systems. In python, I created a code that visualizes random walk in two dimensions (code attached in appendix). The output shows random movement against an x and y-axis. The code uses 1000 time steps.

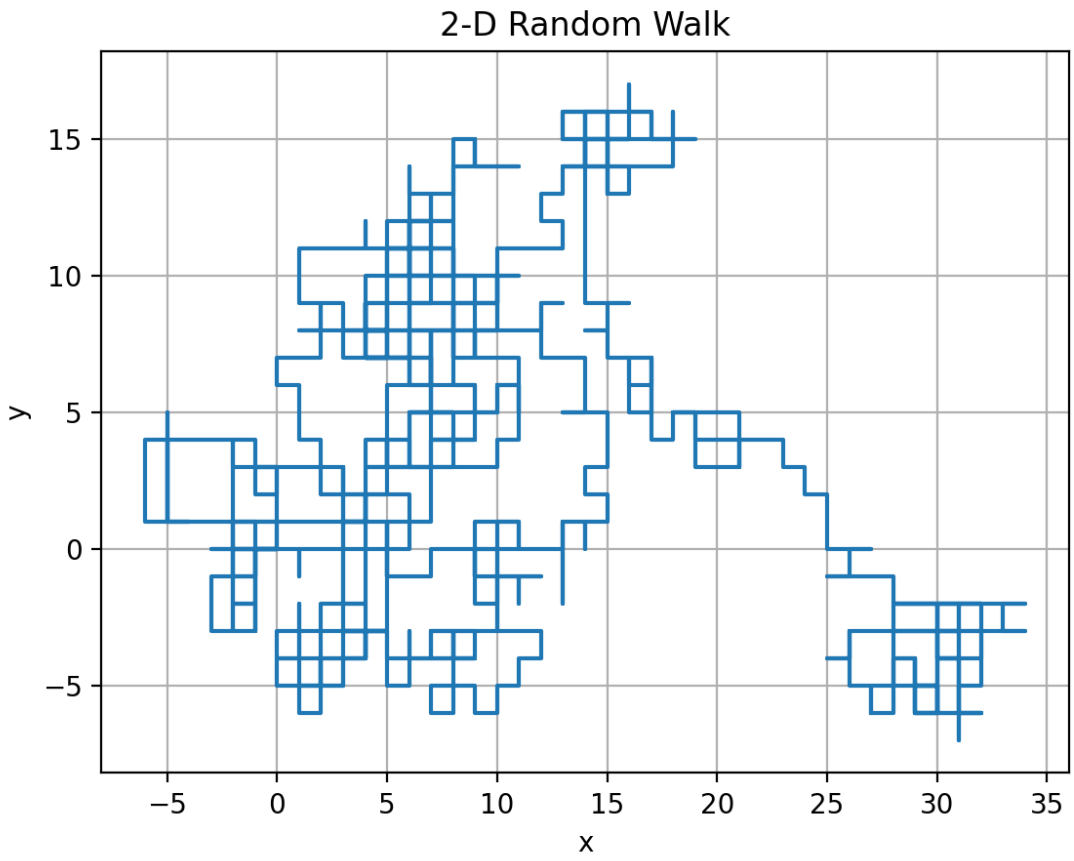


Figure 5 **2D Random Walk Graph**: This is a random walk graph in cartesian coordinates in 2 dimensions. The x-axis represents movement in x-direction. The y-axis represents movement along the y-axis. The run starts at the origin of the graph and moves randomly until the desired number of time steps is complete. It is shown how as the number of steps increases the location

of the point moves randomly along the x and y-axis, there is no correlated data behind this movement.

An example of a random walk in 3-D can be represented by how a gas particle travels through a room. For motion in 3 dimensions there, as in 2-D, is yet another dimension the movement can take place. Also as in 2-D, 3-dimensional random walks can happen in multiple coordinate systems. In python, I created a code that visualizes random walk in three dimensions (code attached in appendix). The output shows random movement against the x, y, and z-axis. The code uses 1,000-time steps.

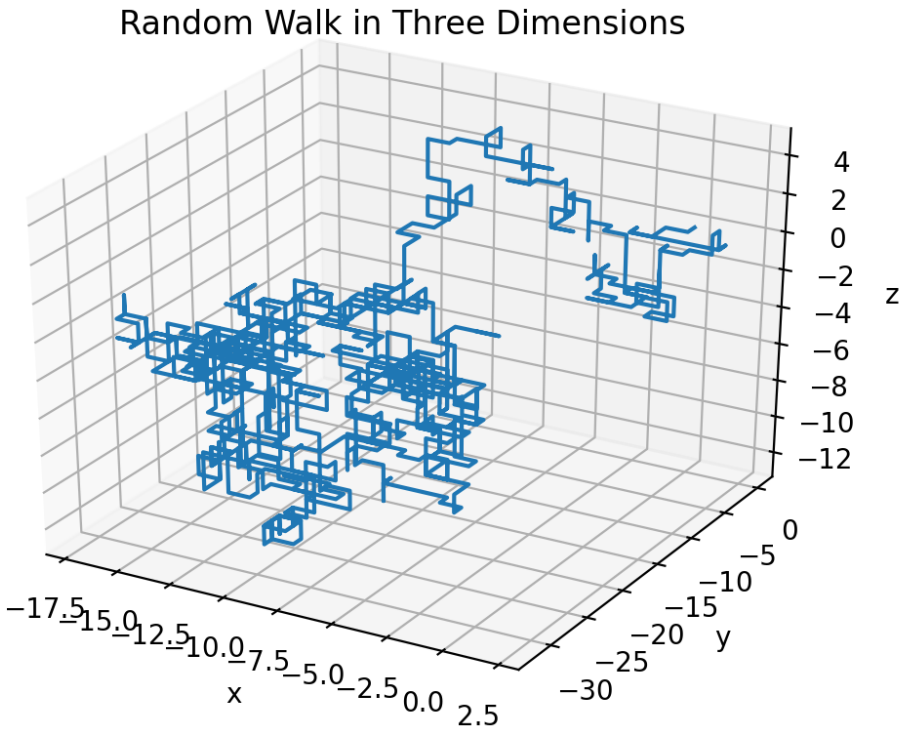


Figure 6 **3D Random Walk Graph**: This is a random walk graph in cartesian coordinates in 3 dimensions. The x-axis represents movement along the x-axis. The y-axis represents movement along the y-axis. The z-axis represents movement along the z-axis. The run starts at the origin of the graph, and moves randomly until the desired number of time steps is complete. It is shown how as the number of steps increases the location of the point moves randomly along the x, y and z-axis, there is no correlated data behind this movement.

Random walk can be applied to more dimensions, however for our purposes only random walk up to three dimensions is necessary.

2.5.0, Black-Scholes Model

The Black-Scholes model was developed by three economists, Fisher Black, Myron Scholes, and Robert Merton in the 1970s. This model suggests the pricing of european stock options. The model they generated has been very influential on the pricing of options and is widely used today for evaluations. What makes the Black-Scholes model unique is “Black and Scholes used the capital asset pricing model to determine a relationship between the market’s required return on the option and the required return on the stock” (Hull 319). Given how price and time are both needed this was not an easy connection to make. Myron took the Black-Scholes model and took it a different direction. Myron’s approach involved setting up a riskless portfolio consisting of the option and the underlying stock. That argument that is generated from this analysis is the return on the portfolio over a short period of time has to be the risk-free return. Although the approach still is dependent on a stochastic process time given the underlying value of the portfolio created, the model is not bound by CAPM assumptions. Other assumptions must be included into the model to fully understand it. Those assumptions are as follows:

- The σand μvariables developed for the price of a stock in early sections are still constant.
- The short selling of securities with full use of proceeds is permitted.
- There are no transaction costs or taxes.
- No dividends are paid during the lifetime of the derivative
- There are no riskless arbitrage opportunities.
- Security trading is continuous.
- The risk-free rate of interest, r, is constant and the same for all maturities (Hull 329).

Exploring the derivation of the Black-Scholes model, using equation 3.11 will give a strong starting place for the price evaluation of a stock. That equation is,

$$dS = \mu S dt + \sigma S dz$$

Given that equation follows a time-dependent stochastic process, apply Itô's Lemma to the above equation.

$$df = \left(\frac{\delta f}{\delta S}\mu S + \frac{\delta f}{\delta t} + \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\right)dt + \frac{\delta f}{\delta S}\sigma S dz \qquad 2.5.0$$

Observing the notation of the variables, f represents the price of an economic derivative. An important note is that f is contingent on S. Equation 2.5.0 is represented by continual time. This equation can be altered to be represented by discrete-time. First, though the pricing of stock options is needed for discrete-time which is,

$$\Delta S = \mu S \Delta t + \sigma S \Delta z \qquad 2.5.1$$

Applying Itô's Lemma to the equation above,

$$\Delta f = \left(\frac{\delta f}{\delta S}\mu S + \frac{\delta f}{\delta t} + \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\right)\Delta t + \frac{\delta f}{\delta S}\sigma S \Delta z \qquad 2.5.2$$

The Δf and Δs represent a change in f and s for a given change in time, Δt. Given that we applied Itô's Lema the Wiener process defined in the model that underlies the variable for f and

s are the same. Essentially, Δz for discrete-time of stock option pricing model and equation 5 both can be explained by

$$\Delta z = \epsilon\sqrt{\Delta t}$$

The takeaway from this is that the Wiener process can be eliminated from the model from the construction of the stock portfolio along with the derivative. "The holder of this portfolio is short one derivative and long an amount $\delta f/\delta S$ of shares. Define Π as the value of the portfolio. By definition" (Hull 330)

$$\Pi = -f + \frac{\delta f}{\delta S}S \qquad 2.5.3$$

Once again a Δ can be applied to show a change in the value of a portfolio for a discrete unit of time. So the equation for the value of a portfolio becomes

$$\Delta\Pi = -\Delta f + \frac{\delta f}{\delta S}\Delta S \qquad 2.5.4$$

Substituting equation 2.5.1 and 2.5.2 into equation 2.5.3 gives

$$\Delta\Pi = -\left(\left(\frac{\delta f}{\delta S}\mu S + \frac{\delta f}{\delta t} + \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\right)\Delta t + \frac{\delta f}{\delta S}\sigma S\Delta z\right) + \frac{\delta f}{\delta S}(\mu S\Delta t + \sigma S\Delta z)$$

$$\Delta\Pi = -\frac{\delta f}{\delta S}\mu S\Delta t - \frac{\delta f}{\delta t}\Delta t - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\Delta t - \frac{\delta f}{\delta S}\sigma S\Delta z + \frac{\delta f}{\delta S}\mu S\Delta t + \frac{\delta f}{\delta S}\sigma S\Delta z$$

$$\Delta\Pi = -\frac{\delta f}{\delta t}\Delta t - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\Delta t$$

$$\Delta\Pi = \left(-\frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\right)\Delta t \qquad 2.5.5$$

Equation 2.5.5 is intriguing since the Δz terms canceled in the derivation, the portfolio is intrinsically implied to be riskless. To fully understand the model some of the assumptions defined earlier are applied here. Given the assumptions the portfolio created earns the same rate of return as other short-term risk-free securities. securities. If the portfolio receives a rate of return that is greater than this return, a riskless profit could be made by arbitragers by borrowing

money to buy the portfolio. If the portfolio receives a rate of return that is less than this return, riskless profit, a riskless profit could be made by arbitrage by shorting the portfolio and buying risk-free securities. This rationale aligns with the equation

$$\Delta\Pi = r\Pi\Delta t \quad 2.5.6$$

Note r in this equation is defined to be the risk-free interest rate. If a similar approach is taken to equation 2.5.6 as equation 2.5.5, the Black-Scholes-Metron differential equation can be created by substituting equation 2.5.3 and equation 2.5.1 into it. This gives,

$$\left(-\frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2\right)\Delta t = r\left(-f + \frac{\delta f}{\delta S}S\right)\Delta t$$

$$-\frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2 = r\left(-f + \frac{\delta f}{\delta S}S\right)$$

$$-\frac{\delta f}{\delta t} - \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2 = -rf + \frac{\delta f}{\delta S}rS$$

$$rf = \frac{\delta f}{\delta t} + \frac{\delta f}{\delta S}rS + \frac{1}{2}\frac{\delta^2 f}{\delta S^2}\sigma^2 S^2 \quad 2.5.7$$

As mentioned this is the Black-Scholes-Metron differential equation. From here, the solutions for this differential equation could be any derivative that is bound by the variable S (the price of the stock). The boundary conditions that get applied to equation 2.5.7 represents the type of financial derivative that is solved. The main type of derivative in question here is the european call and put option. The boundary conditions that define those are for the call option

$$f = max(S - K, 0) \text{ when } t = T$$

For the put option

$$f = max(K - S, 0) \text{ when } t = T$$

Applying these boundary conditions to the black-Scholes-differential equation gives the model for the pricing of european call and put options. The call option is

$$c = S_0 N(d_1) - Ke^{-rt} N(d_2) \quad 2.5.8$$

The put option is

$$p = Ke^{-rt} N(d_2) - S_0 N(d_1) \quad 2.5.9$$

Where d_1 and d_2 are

$$d_1 = \frac{ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

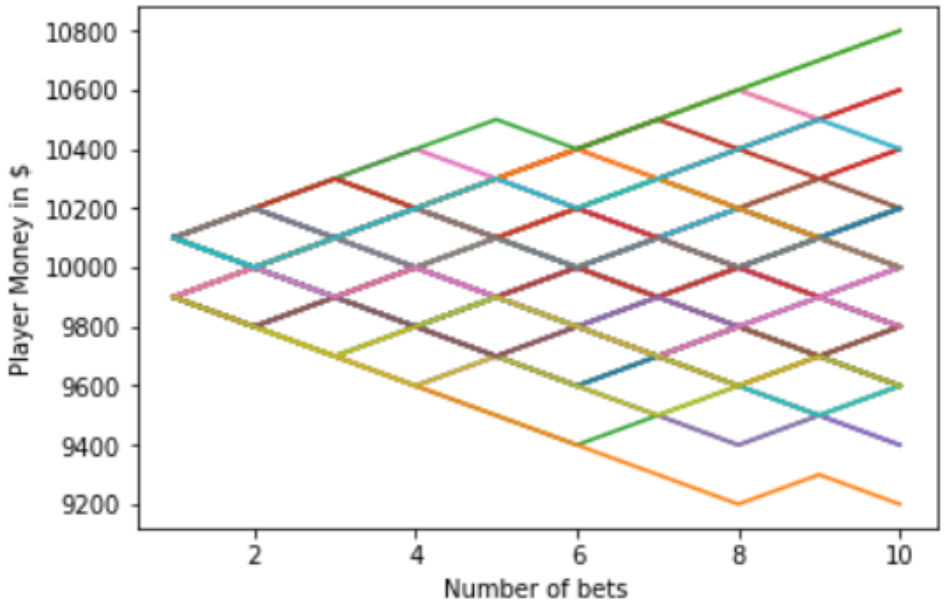
$$d_2 = \frac{ln(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

Notation-wise, c is the price of a call option and p is the price of a put option. "The function $N(x)$ is the cumulative probability distribution function for a variable with a standard normal distribution. In other words, it is the probability that a variable with a standard normal distribution will be less than x (Hull, 344).

2.6.0, Monte-Carlo Simulations

The Monte-Carlo method was first introduced in War World II and used in the Manhattan Project to develop the first nuclear bomb. John von Neumann and Stanislaw Ulam suggested it to observe the interactions of how neutrons travel through radiation shielding. They named the simulation after the casino in Monaco. This is due to the relation between the random nature of how the simulation works matched with the randomness that happens inside a casino. Monte-Carlo simulations have a variety of applications. They are used in fields such as physics, economics, engineering, art, mathematics, environmental science, law, and many more. Essentially what a Monte-Carlo simulation entails is a model of a dynamic system to observe the outcome of a process that involves random variables. The framework of the simulations is what

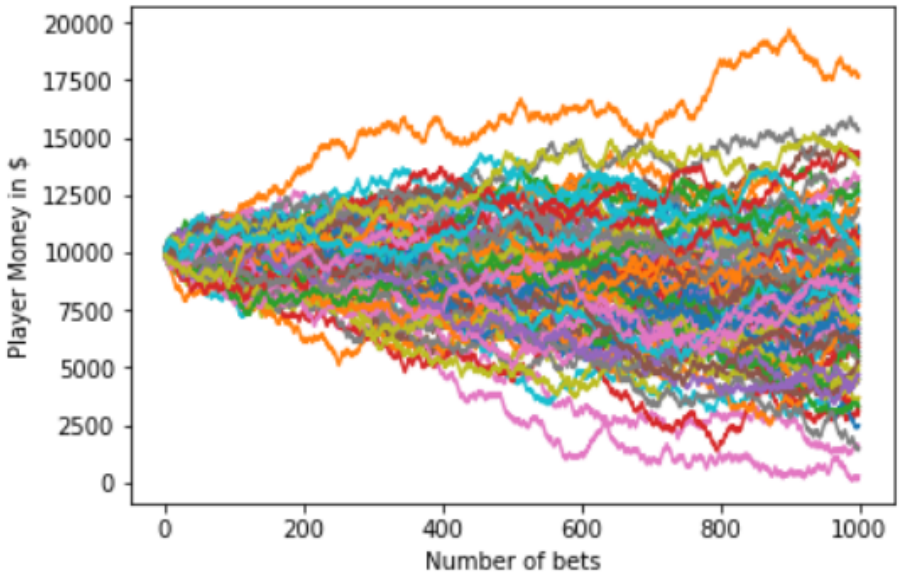
are our desired outputs? What will these outputs be used for? How accurate/precise must the outputs be? How exactly can/must we model? How exactly can/must we define the inputs? How do we model the underlying processes? [Harrison] Answering these questions is key to advising a successful Monte-Carlo simulation. These simulations can take many forms but observing a basic example of one assumes there is a casino dice game where if the player rolls a 1-51 the casino wins and if the dice ends up between 52-100 the player wins. The player initially starts with $10,000 and a bet of $100 is made every time the player plays the game. If all outcomes are equally likely a Monte-Carlo simulation can be made to show how the outcomes are affected after time.




```
The player starts the game with $10,000 and ends with $9958.0
```

Figure 7 **Example Monte-Carlo Simulation 1**: Shows the potential outcomes of the casino game after 10 bets. The player ends up with an average of $9,958. That's a $42 loss[9]

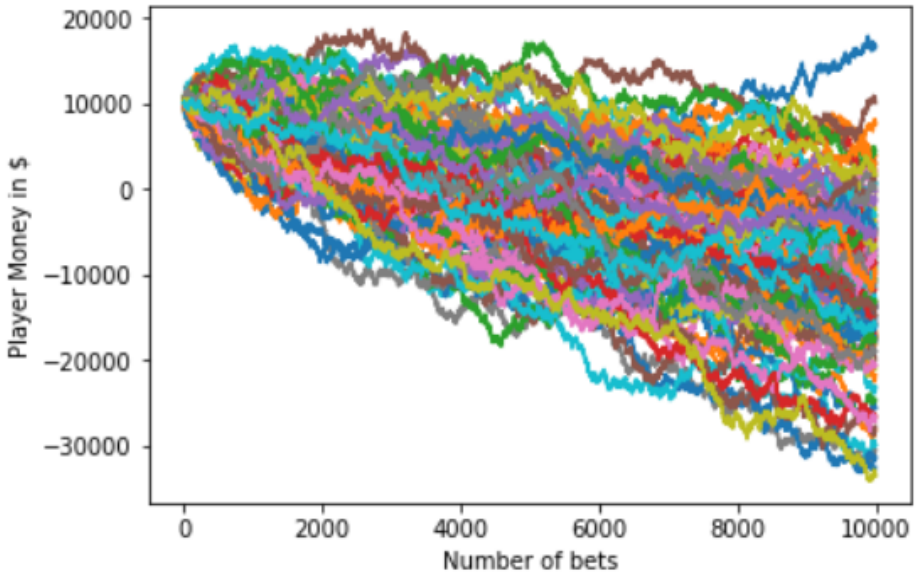
If the simulation time is increased from 10 bets to 1,000 bets




```
The player starts the game with $10,000 and ends with $8034.0
```

Figure 8 **Example Monte-Carlo Simulation 2**: Shows the potential outcomes of the casino game after 1,000 bets. The player ends up with an average of $8,034. That's a $1,966 loss. [9]

If the simulation time is increased from 1,000 bets to 10,000 bets




```
The player starts the game with $10,000 and ends with $-10700.0
```

Figure 9 **Example Monte-Carlo Simulation 3**: Shows the potential outcomes of the casino game after 10,000 bets. The player ends up with an average of $-10,700. That's a $20,700 loss. [9]

Observing the statistics from each run, average, extremes, and standard deviation can be made from each graph to show the possible outcomes. From this simulation, it is clear that the player is better off stopping the game earlier than if they continue to play. This type of simulation will be applied to options to see how they are priced as time evolves.

3.0 Methodology

3.1 Monte-Carlo Simulations

As mentioned earlier Monte-Carlo simulations are a method that can be utilized universally by many fields of study. The main objective of these simulations is to model a dynamic system to observe the outcome of a process that involves random variables. To code these simulations I use the programming language Python. Python is a very versatile coding language with endless possibilities and is extremely useful in computational work. This is mainly in part due to its vast libraries that have direct applications to scientific computing. I intend to show a relation between physics and economics through Monte-Carlo simulations.

For Monte-Carlo simulations themselves, the method follows a general structure. What are the inputs? How do we need to model the underlying process? A replication needs to be implemented, and lastly understanding the outputs produced. Observing these more individually, starting with defining inputs, generally, there is some sort of distribution that the data needs to follow. In some cases, the distribution is a standard normal while in others the distribution may be non-uniform. Understanding how the inputs follow the respected distribution is vital to the completion of the Monte-Carlo method. Just because there is some distribution the data follows doesn't mean the simulation is complete. The underlying process the simulation takes is vital in understanding the meaning of the method. For instance, there is some mathematical model that the distribution or data needs to travel through in order to produce a result. This mathematical model will look different depending on the goal of the Monte-Carlo simulation. Since some distribution and mathematical process has been defined, a replication is now needed. These types of simulations deal with generating random values to progress the model. In order to generate

accurate outputs, the simulation needs to be repeated a large number of times. This ensures that there is a large distribution of data to pull results from. Without the replication, these simulations could have inaccurate results. Lastly, with all of the above steps implemented a result should be generated. Understanding the meaning of the result is vital for any research but with Monte-Carlo simulations, the replication of the simulation leads to an array of data to understand.

3.2 Photon Traveling Through Sun

One possible analysis of the many in a Monte-Carlo simulation in physics is how a photon travels through the sun. Here, a photon is born at the core and takes a random walk until it hits the radiation zone where it gets trapped, then carried to the surface, then finally released into space. Modeling this process can be rather difficult, yet some realistic simplifications in order to find some statistics. Three different Monte-Carlo simulations are created to find the statistics in question. One will focus on a visual representation of a two-dimensional walk. Another will focus on the average in the number of steps the simulation takes, its standard deviation, expected error, average time elapsed, and an arbitrary proportionality variable called alpha. The last will focus on time vs displacement.

First, for the visual walk code, the boundary condition is set so the walk has a discrete stopping point. This variable is defined as radius_to_rz and is equal to the actual radius of the sun from its core to its radiation zone which is $4.697x10^{8}$m. Next, the step size of the photon needs to set. On average this step size is equal to 2mm. This is known as the mean free path. In the code, this is set as a variable equal to mfp. For the code, the mean free path is scaled by a factor of one billion. This allows the simulation to run within a reasonable amount of time. A

Python library called turtle is used to produce the image. Essentially, turtle is a library that replicates a pencil on a piece of paper. This is very effective in producing a clean visual trace for a random walk. The library even has the possibility to show in real-time the photon traversing the sun. The last thing is to define the model itself. For these simulations, it needs to be taken into account that the photon has an equal probability of moving in any direction. Given this, circular coordinates are utilized in the code. Now, the photon is able to move in any direction for its mean free path. To assure this a random angle is generated using python's library called random. This simulation is in two dimensions given turtle's abilities. Although the photon can be modeled in three dimensions (which is used in the statistics simulation), the model actually does not matter. What that means is the photon should take the same number of steps in two-dimensions as it does three-dimensions. With this code, the distribution that is generated is a uniform distribution between 0 and 1. This is because the simulation is operating in circular coordinates. When following the mathematical model multiplying $2\pi$to the generated value gives movement in any radial direction. Now since the distribution is uniform there is an equal probability that the photon can travel in any direction. This process generated the angle the photon will travel. The rest of the mathematical model follows geometry. The angle is split into an x component by cosine and a y component by sine and multiplied by a step size (mean free path) to give it a length. The position is calculated through the pythagorean theorem. To illustrate all as an equation.

$$x = cos(angle) * mfp$$

$$y = sin(angle) * mfp$$

$$position = \sqrt{x^2 + y^2}$$

To continually track the photon all the way to the edge x and y are added to each other that way the next value starts at where the previous value stopped.

$$x = x + cos(angle) * mfp$$

$$y = y + sin(angle) * mfp$$

Since the mathematical model has been defined a regression needs to be set. For this simulation since one image being produced the replication is continually running the simulation to generate an array of different images. The output gives one instance of a random walk of a photon traveling through the sun.

Next observing the statistics code called Photon Statistics 3D most of the code is similar to the visual walk code. The mean free path and stopping point remains the same. The main difference between the two is that this code is transcribed into three dimensions. Since the code is in three dimensions the distribution and mathematical process involved are different from the visual code. For the distribution, there are actually 2 different distributions that are generated one for the variable cos_phi and one for the variable theta. The distribution for theta is the same as in the visual code. Meaning it is uniformly distributed between 0 and 1 then multiplied by 2π. The distribution for cos_phi is still uniform but from -1 to 1. This is because if the distribution is not bound from -1 to 1 the movement would not distribute throughout the sphere. It could add to the other variable. The cos_phi is observing colatitude. While theta is observing longitude. The mathematical process involved for this Monte-Carlo method is to take the generated numbers, turn them into angles and do the walk. The three-dimensional process is very similar to the two-dimensional. The main difference is there's an additional axis of movement to take into consideration. So the angle is split into an x, a y, and a z component. This is again done with

cosine and sin, but with theta and phi. So once cos_phi is found sin_phi needs to be found this is accomplished through

$$sin_phi = \sqrt{1 - cos_phi^2}$$

Theta is found the same way as the angle above by multiplying the value by a factor of 2π. The rest of the mathematical model is similar to the visualization model just with the addition of the cos_phi and sin_phi. In three dimensions this is illustrated by.

$$x = x + sin_phi * cos(theta) * mfp$$

$$y = y + sin_phi * sin(theta) * mfp$$

$$z = z + cos_phi * mfp$$

$$position = \sqrt{x^2 + y^2 + z^2}$$

It can be seen that these equations are just the visual code translated into three dimensions all the same methods apply. Such as multiplying by the mean free path in order to receive a step size. Adding x, y, or z to the previous equation to start at the previous step's stopping point. With this model, there is still an assumption that the photon can travel in any direction. What is important with Monte-Carlo simulations is that there is a large number of replications in order to get accurate results. So for this simulation, there are 100,000 replications that are run in order to generate a large distribution of results and greatly increase accuracy in the data. The outputs desired are certain statistics such as the average amount of steps the simulation takes, the standard deviation of the steps taken, the standard error, an arbitrary value called alpha, and the average time taken for the photon to reach its stopping point. The average tells us the summation of all the steps divided by the number of replications. The equation is defined as

$$avg = \Sigma \frac{x_i}{n}.$$

The standard deviation shows how much the values of the sample differ from the average. This is found by

$$std = \sqrt{\frac{\Sigma(x_i - x_{avg})^2}{n-1}}. \quad 3.21$$

The standard error is a measure of the statistical accuracy and is found by

$$std\ error = \frac{std}{\sqrt{N}}. \quad 3.22$$

Theoretically, the distance traveled should scale based on the square root of time. This is explored more in the next photon code labeled Time Elapsed Code.

$$d = l\sqrt{N} \quad 3.23$$

Where d is total distance and l is the step size (or mean free path), and N is the total number of steps. Trying to explore if this is actually true a proportionality coefficient will be added.

$$d = alpha * l\sqrt{N} \quad 3.24$$

Rearranging for alpha gives

$$alpha = \frac{d}{l\sqrt{N}} \quad 3.25$$

This alpha value should be theoretically one. If this is the case then the distance traveled is truly proportional to mean free path times the number of steps.

The last code in observing the photon is a time displacement code. What this code aims to address which the previous codes fail to do is calculate the amount of time the photon actually takes to traverse the inside of the sun. What differentiates this program from the visual and statistics code is a different stopping distance and mean free path. Since we want to observe how long it takes to travel over time, a list of different distances are initialized. To get more accurate results the mean free path is changed back to 0.002 m. For each distance, a hundred simulations

are run at each distance in order to make the data more accurate. Lately, the model follows the same underlying movement process as the statistics code. This means the photon is modeled with the three-dimensional movement for the code. Therefore it has the same distribution for the inputs and mathematical model as the three-dimensional code. What differs is since the model has to run through a loss the regression is set at a lower number in order to speed up the time as mentioned above that number is 100. The desired output is the average time elapsed is a value for how long the photon took to traverse the sun's interior. The amount of time the photon travels in a single mean free path is found from

$$distance = rate * time.$$

Rearranging for time and substituting the appropriate variables for distance and rate gives

$$time\ per\ step = mean\ free\ path\ /\ speed\ of\ light.$$

Mean free path is substituted for distance since that is how far the photon travels, and the speed of light is substituted for rate since that is the velocity the photon travels. Time per step's units turn out to be seconds. This is since the mean free path is in meters and the speed of light is in meters per second. Unit analysis on the equation removes meters and just be in seconds. Once the time per step is calculated, an estimate can be made based on the total number of steps the simulation takes. Time elapsed is found by

$$time\ elapsed = time\ per\ step * number\ of\ steps. \quad 3.26$$

This calculates the total time taken the photon takes in the simulation to travel a certain distance in seconds. The variable time per step unit was calculated to be in seconds. The variable number of steps is unitless. Performing unit analysis in the above equation leads time elapsed to be in seconds. Based on this a graph should be able to be generated and an equation can be found to solve for the actual time elapsed for a photon to travel from its core to the radiation zone.

3.3 Monte-Carlo Simulation for Options

Moving on to the economic application of this research project, the objective of this research project is to show how Monte-Carlo simulations can be applied to economics. I will be specifically observing applications within option markets. This project is going to focus on european call and put options. It is important to note that this is not the only application for Monte-Carlo simulations in the field of economics. Essentially the outcome of this project shows potential outcomes the price a european call and put option can take based on changing different parameters.

Earlier it was mentioned the parameters that go into an option. Specifically, whether the option is european or american, the current stock price, the strike price, the risk-free interest rate, the volatility, the time to expiration of the option, and the dividends expected to be paid. For this model european options are observed. This is for two reasons. The first is that we are utilizing the Black-Scholes-Metron model. This model observes european options, not american ones, and is an accurate and widely used model today for option pricing. The second is that european options are simpler to model since the right to exercise the option can happen only on the expiration date. Next, the other parameters need to be taken into account, all are utilized in the model except dividends expected to be paid. Including this variable complicates the model so this variable is excluded. Observing the values of the parameters for my simulation they are as follows.

```
So = 95
K = 100
r = 0.1
sigma = 0.2
mu = r # Make assumation that the expected return from th
T = 0.5
```

Figure 10 **Base Parameters**: This figure is the parameters that are set into the code. So is the current stock price and its units are expressed in dollars. K is the strike price and its units are also dollars. r is the risk-free interest rate it is since r is a rate, its units are per year. Sigma is the volatility which also has units of per year. mu is the expected rate of return from the underlying asset in the model. The model makes an assumption that the expected rate of return is equal to the risk-free rate and therefore has units of per year. T is the time to expiration which units are in years. These values assigned to the variables will be known as the base parameters.

The base parameters are set at what they are for a few different reasons. The current stock price and the strike price are $95 and $100 dollars respectively; this allows us to see a large spread in the graphs. It provides a good visual where the runs are not within a few dollars of each other and when variables are changed to not get out of control. This $5 difference in the variables also gives relatively equal Black-Scholes pricing values for the call and put model. This allows the data to be equally beneficial for a call and put option model; it does not benefit either one more than the other.

The risk-free interest rate is 0.1 or in other words 10% per year. The risk-free interest rate is purely a theoretical rate that does not exist in the real world because every given investment has some underlying risk. The current treasury bill is primarily used for the risk-free rate. This is because the bill is backed by the U.S. Government. Even though 10% per year for the model is a tad high to assume. It is important to take away what happens when the variable's values are

changed. Setting the risk-free interest rate at 10% per year allows the model to produce better visual results when the variable is changed and still not be unreasonable for a base model.

The volatility is set at 0.2 which equals 20% per year. “Stocks typically have a volatility between 15% and 60%” (Hull, 323). So setting the base parameter at 20% per year is a realistic assumption for the base model. It allows the base model to start with a distribution spread that is pretty equal on its extremes and nothing too high or low. It also is a great starting point for when the parameter is altered in the model.

The time is set at 0.5 or in other words half a year. Since there are only 252 trading days in a year, for the model, 0.5 is multiplied by 252. This corrects the value to give the time to expiration a more accurate representation of the real world. Half a year is a safe value for the model since it gives the simulation plenty of time to change its prices.

For the simulation itself, in the model, a probabilistic distribution is generated for each run that has a standard normal distribution. This means the distribution has a mean of 0 and a standard deviation of 1. For the mathematical model, the current stock price is calculated based on the Black-Scholes model. The equation follows,

$$S_i = S_{i-1} * e^{\mu - \frac{1}{2}\sigma^2 * dt + \sigma\sqrt{dt} * Z}. \qquad 3.31$$

Equation 3.31 calculates the next price of the stock option based on the previous price multiplied by a function, a time step, and a standard normal distribution. A time step is found by dividing the time to expiration by the number of steps ($T * 252$). This process is replicated N a number of times. The number of times the simulation is replicated is 100,000. This provides a large distribution of option paths to take data from. Once that is concluded the maximum ending value, the minimum ending value and the average of all ending values is found.

Expectations of the results are observed in figure 11.

Variable	*European call*	*European put*	*American call*	*American put*
Current stock price	+	−	+	−
Strike price	−	+	−	+
Time to expiration	?	?	+	+
Volatility	+	+	+	+
Risk-free rate	+	−	+	−
Amount of future dividends	−	+	−	+

Figure 11 **Expected BSM Value Movements**: (Hull, p. 232) This image shows what would happen to the option price for a European call, a European put, an American call, and an American put. A plus (+) assumes a rise, a minus (-) assumes a fall, and a question mark (?) means the relation is unknown (can be either positive or negative). The conclusion shows if there is a rise in the value of the variable what would happen to the option if all other variables were held constant.

Observing each variable in figure 11, an increase in the current stock price increases both call option values since the position taken is the buyer's position. It would decrease the put since the position taken is the seller's position. The strike price decreases the calls since the higher it is the more unconfident the price will go over that value, while the put options it offers a safer investment for the owner. The time to expiration is unknown for european options; this is due to the fact that the right to exercise the option can only be taken on the expiration date so it is dependent on the final value. This could both benefit and hurt the owner. While in the American options, since it can be exercised at any point, the more time the better. Volatility is valuable across the board if it increases cause it drastically changes the value more. This could lead to a benefit for both call and put options with minimal losses. The risk-free rate is the amount of money acquired assuming zero risk, this variable technically does not exist but if the risk-free rate increases it betters calls since it increases the price of stocks. I am not going to go into dividends since they are not included in my model.

3.4 Relation Mapping

A large part of this research project is to show connections between physics and economics through Monte-Carlo simulations. Therefore, I want to illustrate the similarities between the econ code and the physics statistics code (See appendix for codes). These codes are very similar, they both illustrate random walks, just in different dimensions. The econ code is one-dimensional while the physics code is two-dimensional. Obviously, both codes import some libraries, initialize some variables, and have similar python language expressions like print statements. I want to focus on the direct similarities in the programs themselves. Mapping the similarities to start is a variable labeled N. This variable shows the number of replications the simulations undergo. In both codes, this variable is set equal to 100,000. This provides a large distribution of data to add better accuracy to the programs. Next, I want to point out the process where each code has an underlying mathematical equation to move the random walk. In addition, there are random numbers and data that are generated in each of the two codes for the simulations. Lastly, the goal is to generate some statistical data to take away what each code does. Although each one is empirically different, the process remains the same between the two codes. Set the number of replications, set a process for the code to advance in time, add the random competent, and generate data. It is worth noting that the main difference between the two codes is that the econ code has an exact stopping point while the physics code has a discrete stopping point. What this means is there is always the same amount of time steps for each run in the econ code while in the physics code each run is different but every run stops at a distinct boundary. It can be shown that the overarching symmetry of these simulations are very similar. If this was to explore more applications other than physics and economics the simulation would still follow a similar pattern. The pattern is bound by what was referenced in section 3.1. Which

is What are the inputs? How do we need to model the underlying process? A replication needs to be implemented, and lastly understanding the outputs produced. These are the key to any Monte-Carlo simulation.

4.0 Results

4.1 Photon Visual Code

The below figure is collected from the photon visual walk code. To assure the code is truly random a random number generator is not implemented. This allows for different images to be produced each time the simulation is run. Figure 12 shows four different runs from the photon visual code. Visually observing the runs, image a and b took a longer amount of steps. Image c took the shortest amount of steps, and image d took the longest amount of steps. Although the photon started at the same point (the center) on each image the photon took completely different paths for each image and never ended in the same final location.

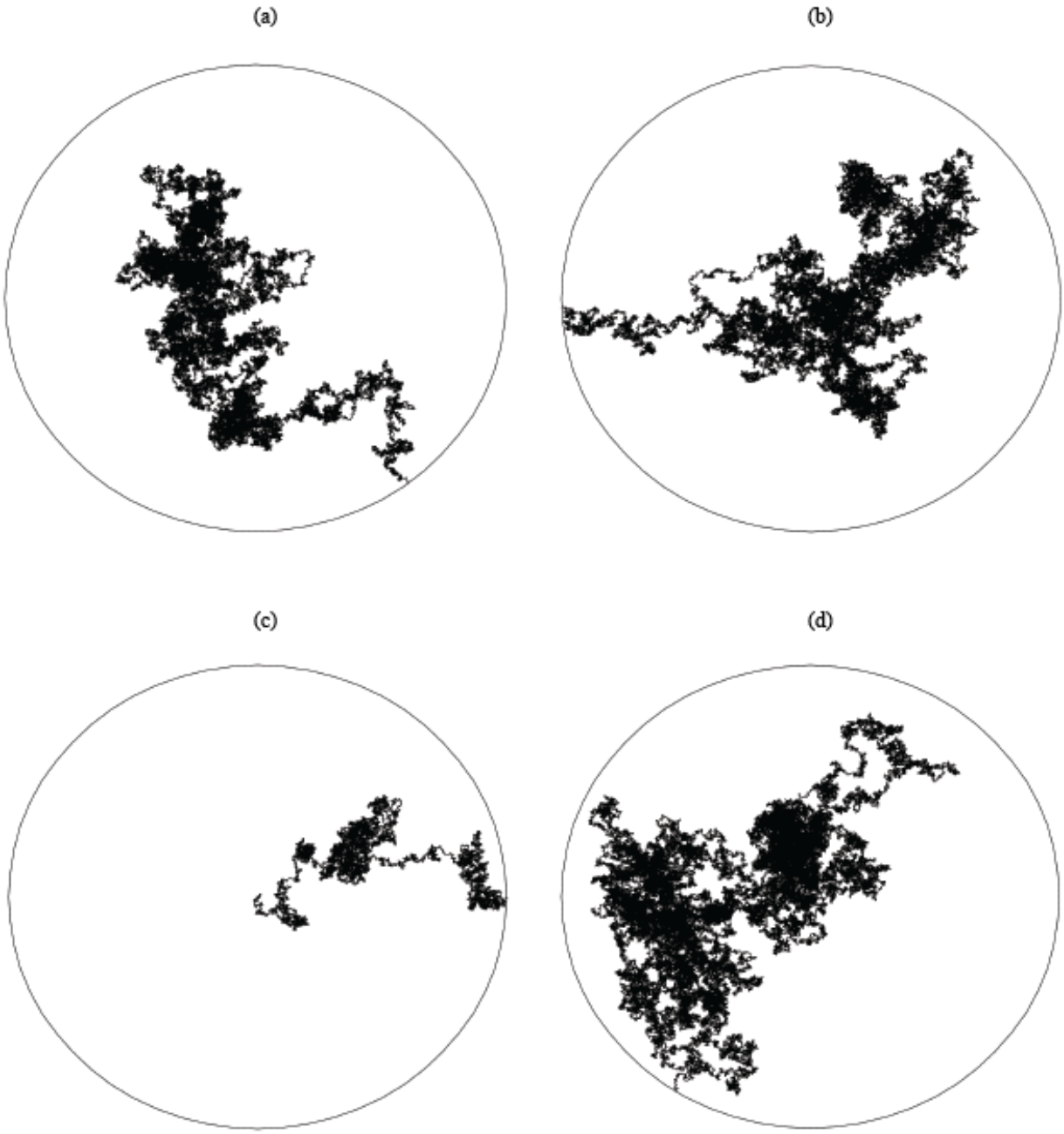


Figure 12 **A Collection of Photon Movements:** The images are in cylindrical coordinates so the x-axis and y-axis both are representing meters in distances. There are four different runs in the figure labeled a, b, c, and d. Each image represents a two-dimensional random walk of a photon traversing the sun. Comparing the individual images together it is clearly visual that each image is not the same in any way. Each run is unique from the next.

Although there are only four images that were produced in figure 12, each image represents one replication of the Monte-Carlo simulation. If more images were wished to be produced, continue to run the simulation individually to increase the number of regressions, therefore, increasing the number of images produced. Each image is different from the next. See the code titled Photon Visual Code in the appendix to observe the code.

4.2 Photon Statistics Code

This code observes the average number of steps, the standard deviation of the average, the standard error, a proportionality coefficient arbitrarily labeled alpha, and average time elapsed. Three runs are observed with each run containing 100,000 replications. That way we can see if the data is repeatable. Essentially there are 300,000 data points in the table below.

	Average # of Steps	Standard Deviation	Standard Error	Alpha Value	Average Time Elapsed (s)
Run 1	55,397	36,799	111.05	0.9978	369.31
Run 2	55,310	34,938	110.48	0.9986	368.73
Run 3	55,336	34,933	110.47	0.9984	368.90
Averages	55,348	35,557	110.67	0.9982	368.98

Table 1 **Photon Statistics Values**: This table shows specific statistical values for three different runs as well as the average of all three runs. Values are found from equations 3.21-3.26 in the methodology.

Observing these numbers a photon takes roughly 55,350 steps to traverse the sun from the core to the radiation zone. Note this number is found with a mean free path of two million. The standard deviation from the runs on average was about 35,550. This number explains how

these simulations can drastically vary on the number of steps it takes for the simulation to conclude. This supports figure 12 where image c showed a smaller amount of steps, while image d showed a larger amount of steps. Moving on to the standard error the average was 110. This number tells us that the simulation is rather accurate. Since the standard error measures accuracy within a sample distribution. This number tells us that when we run the simulation again it will be very likely close to 55,350 steps. This is evident based on the three runs. Moving on to the alpha value this is a theorized proportionality rate. This value should be one since this value is found to be theoretically 1 3.24. For the simulation run, the theorized value turns out to be correct. The average alpha value was essentially one. In addition, if you take an error into effect there is about a plus-minus 0.002 error. With that term, the alpha value falls within one. The last variable observed in this experiment is the time elapsed. Time elapsed is observed more in the next code however for all intents and purposes a photon moving at a mean free path of two million traveling from the core to the radiation zone would take roughly 368 seconds (or 6.31 minutes) to get there. See the code titled Photon Statistics 3D code in the appendix to observe the code.

4.3 Time elapsed code

This code determines the time that elapsed for the photon to reach a specific boundary. Figure 13 shows a base run of the time displacement code (in appendix). This simulation gives a general shape. It is intuitive that the longer radius of the sun the longer the photon takes to travel. The limitation is that these distances are very small.

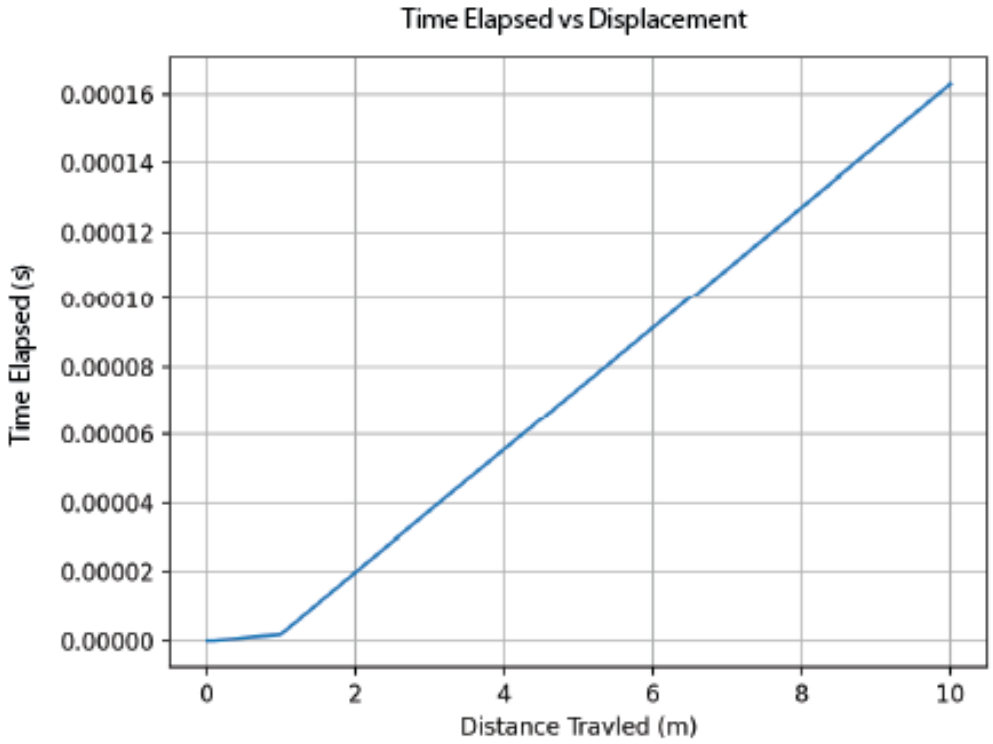


Figure 13 **Time Displacement vs Distance:** This figure represents the time it takes for a photon to travel a different distance. The x-axis represents the distance traveled in meters. The y-axis represents the time elapsed in seconds. There are four points on this graph that are generating this shape.

Visually from the graph, this is not in a straight line. Taking the data points on plotting them on a log log scaled graph should produce a straight line. Since it is hard to see the general shape of figure 13 is parabolic.

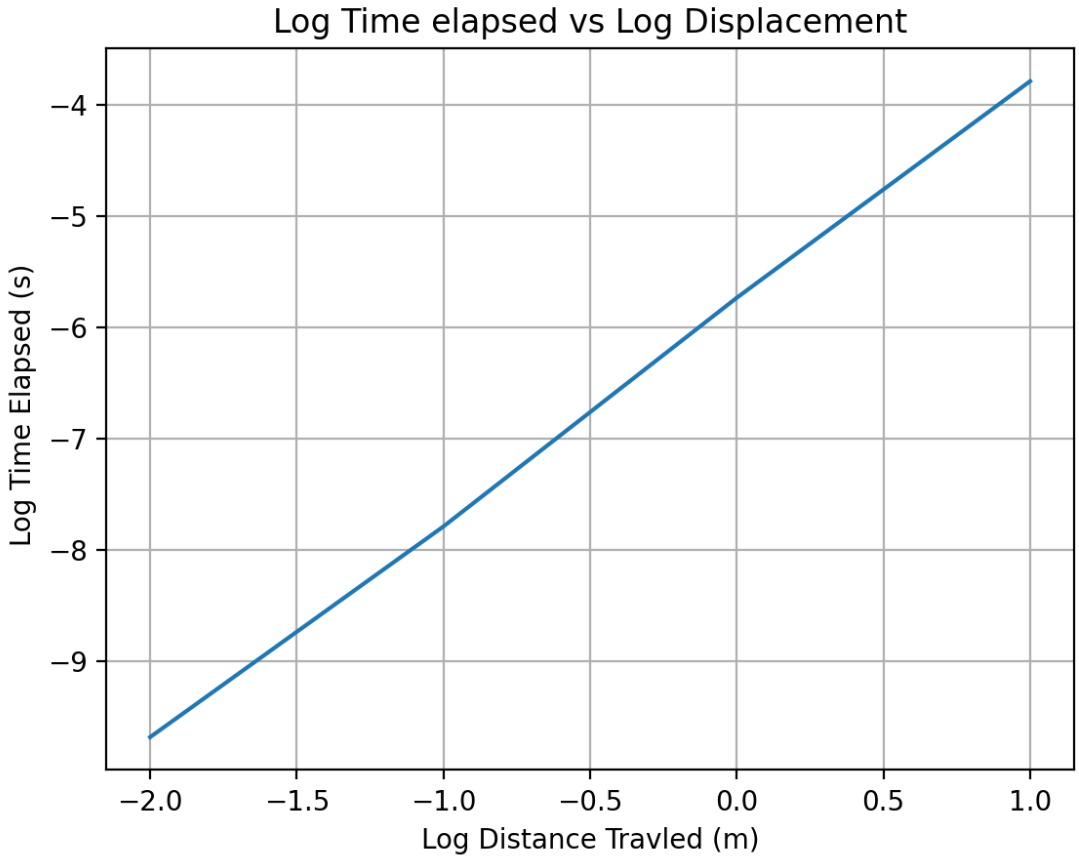


Figure 14 **Log Time Displacement vs Log Distance:** This figure represents the time it takes for a photon to travel a different distance. The x-axis represents the distance traveled in log meters. The y-axis represents the time elapsed in log seconds. This figure is the same data points as figure 13 just on a different scale.

Visually figure 14 is a straight line unlike its same figure with a non-log scale in figure 11. From here a calculation can be made to find how long it would take for a photon of mean free path of 2 millimeters to travel from the center to the radiation zone. Through simple slope-intercept form. It is important to take into effect that this is log scaled, therefore,

$$\log(y) = m \log(x) + b$$

First, find the value for the slope. From the graph the at log x =0.0 log y = -5.73 (This is also the intercept). At log x = 1.0, y = -3.79. Therefore slope is equal to 1.94. Given that these numbers

are rounded this is theoretically closer to a slope of 2. Now that slope and intercept are known we can plug any distance and figure out the time the photon should take or vise versa. So plugging in the distance to the radiation zone for x which is equal to $4.697x10^8$m. should give us the time taken in seconds to travel that distance. Since log scale the log of this number needs to be accounted for. The log of $4.697x10^8$ is about equal to 8.672. Plugging in the slope, the log of x, and intercept, the time elapsed for a photon to reach the radiation zone would take 11.614 log seconds. Converting this away from a log value gives 411,149,721,105 seconds or in better terms 13,037 years. This value is within the theoretical value of 10,000 and 170,000years[10]. It is on the lower value since the mean free path was a constant 2mm throughout the experiment. The mean free path of a photon differs depending on where it is located within the sun's interior. Accounting for a changing mean free path would increase the time it takes for the photon to reach the radiation zone.

4.4 Option Code

For this code, a random number generator is implemented. This allows for the graphs to be replicated as well as what happens to the same graph if the variable changes. The next figure shows the base parameter run

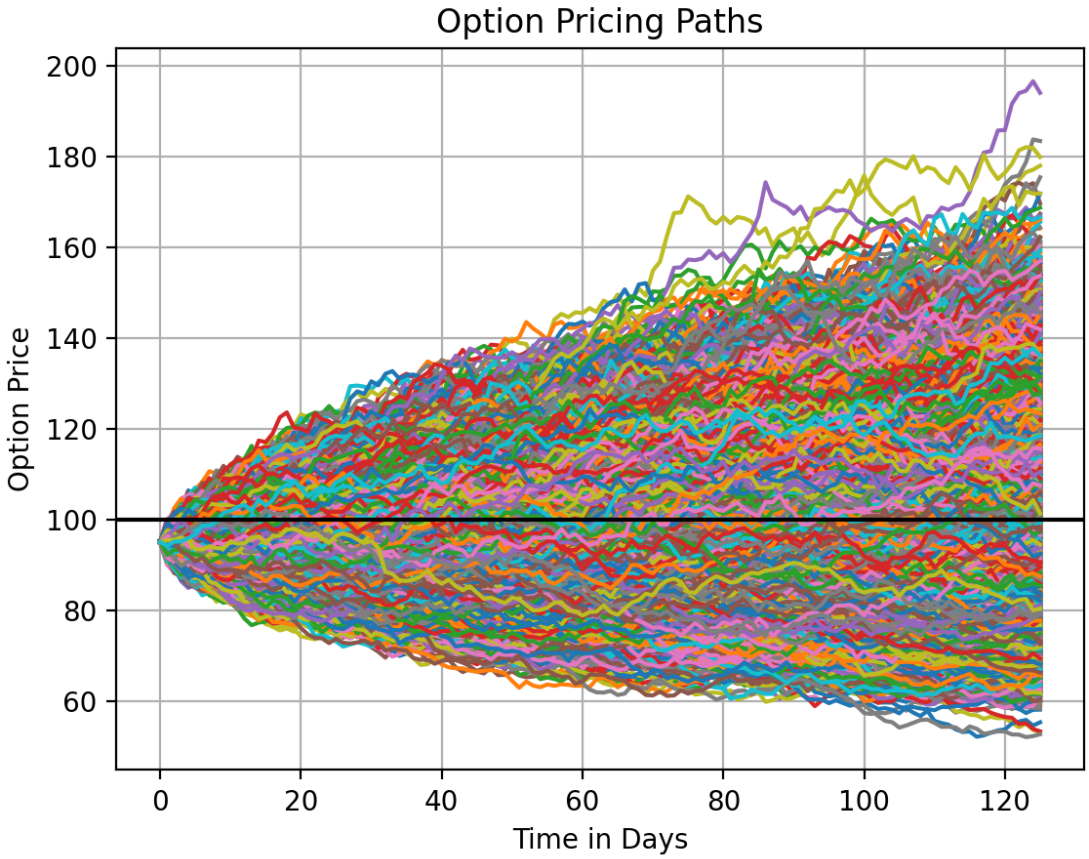


Figure 15 **Base Parameter Option Paths:** The figure illustrates 100,000 different paths a stock-based option could take. Each stochastic line represents one potential outcome the path could take. The x-axis represents time in days and the y-axis represents dollar value. The base parameters in the M-C simulation are the following: So = 95, K = 100, sigma = 0.2, rf = 0.1, T = 0.5. The number of replications is 100,000 runs.

Observing the data from the base parameters,

	Base Values
Average Final Stock Value ($)	99.83
Maximum Ending Stock Value ($)	194.08
Minimum Ending Stock Value ($)	52.62

Table 2 **Base Parameters Statistics Values**: This table shows the average final stock values, the maximum point, and the minimum point for figure 15. All values are represented in dollars.

From the table, the average final value was $99.83. This falls in line with what is expected. Looking at figure 15, The distribution of the final points stays closer to the starting points than the extremes, this is shown by looking at the final values. There is empty space between the final lines at the extreme values. The maximum ending value for the base run was $194.08 and the minimum ending value was $52.62. Note that this is one value from a total of 100,000 final values. The probability of hitting either of these values is 1 in 100,000 or in other words 0.001%. You can see especially in the maximum value space in between that point from the next line. Therefore, although the value can reach that point it is not where the concentration of data resides.

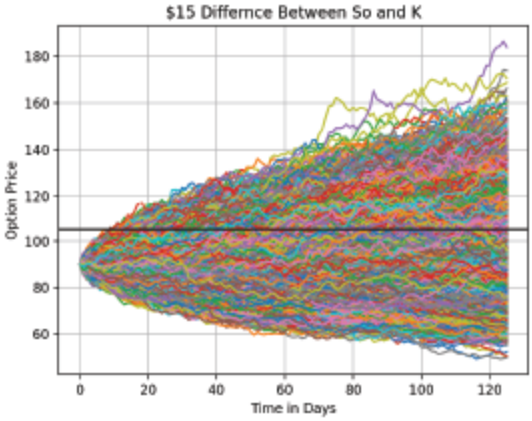
Looking at the Black-Scholes model for the base parameter,

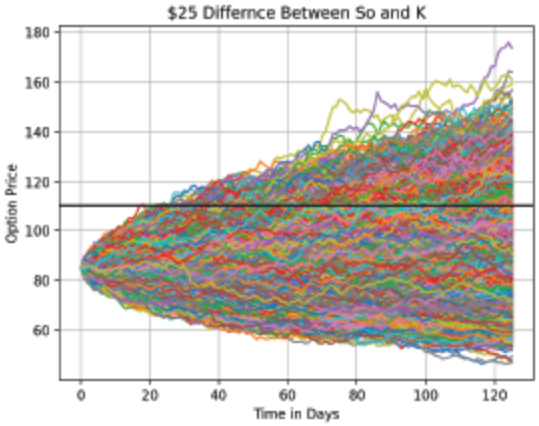
	Black-Scholes Values
Initial BSM Call price	5.30
Average Final Call Price	10.34
Sum and discounted Call Price	5.27
Initial BSM Put price	5.42
Average Final Put Price	5.62
Sum and discounted Put Price	5.42

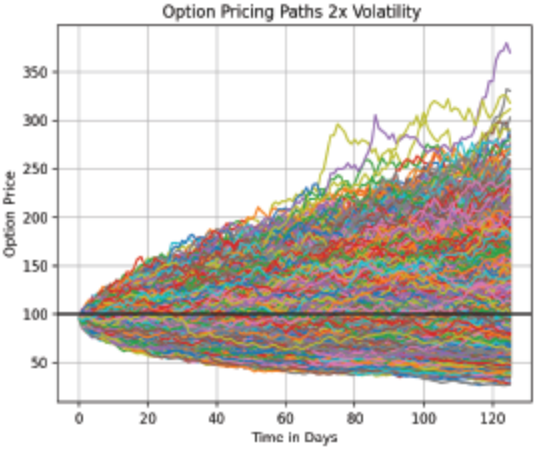
Table 3 **Base Parameters Black Scholes Values**: This table shows the Initial Black-Scholes call and put values, the Average Final Call and Put values, and the sum and discounted call and put values for Figure 15.

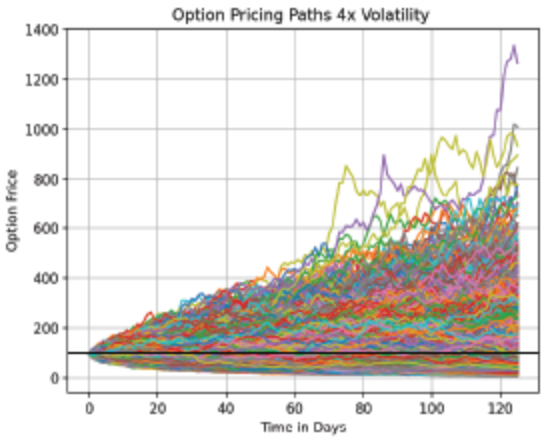
These values are reported to confirm the correctness and robustness of the Monte-Carlo simulation. They are not important for interpretation purposes, and will not be reported after this point. The Initial BSM value should be close to the sum and discounted value, given there is not something wrong with the simulation. Why the average final value is greater than the others is due to the time value of money. The value of money theoretically always is increasing in value. This can be considered inflation.

Moving on to when the parameters have changed the results are observed in Figure 16.


$15 Differnce Between So and K
Option Price
Time in Days

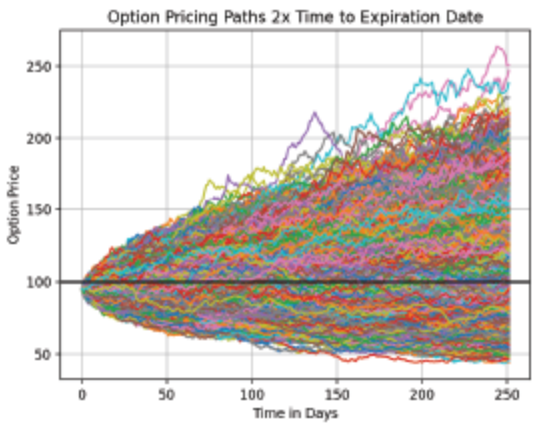

$25 Differnce Between So and K
Option Price
Time in Days


Option Pricing Paths 2x Volatility
Option Price
Time in Days


Option Pricing Paths 4x Volatility
Option Price
Time in Days


Option Pricing Paths 2x Risk Free Rate
Option Price
Time in Days


Option Pricing Paths 4x Risk Free Rate
Option Price


Option Pricing Paths 2x Time to Expiration Date
Option Price
Time in Days

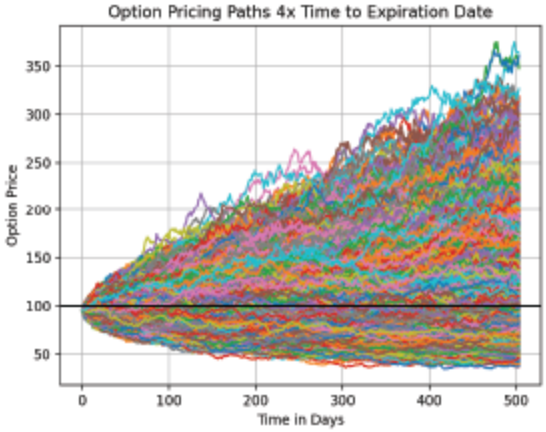

Option Pricing Paths 4x Time to Expiration Date
Option Price
Time in Days

Figure 16 **Collection of Altered Parameters:** This figure shows all the simulations for when a variable is changed. The axis and lines are the same here as they were for the base run figure. The x-axis represents time in days and the y-axis represents dollar value. Each individual graph contains 100,000 lines where each individual line represents one potential outcome. Total there are 800,000 lines collected in this figure. The individual images included a difference in the variable current stock price and a strike price of $15 and $25 difference, a change in volatility by a factor of 2 and 4, a change in risk-free rate by a factor of 2 and 4, and a change in the time to expiration by a factor of 2 and 4. They are labeled in the title individually to differentiate the altered variables. It can be seen how applying the random number generator makes the graphs have the same option price lines with different value outcomes.

Observing the data from these graphs, the average final stock value, the maximum ending stock value and minimum ending stock value is reported.

	Average Final Stock Value ($)	Maximum Ending Stock Value ($)	Minimum Ending Stock Value ($)
Base Values	99.83	194.08	52.62
$15 Difference in So and K (So = 90, K= 105)	94.58	183.87	49.85
$25 Difference in So and K (So = 85, K= 110)	89.33	173.66	47.08
2x Change in Volatility (sigma = 40%)	99.85	369.91	27.19
4x Change in Volatility (sigma = 80%)	99.87	1266.10	6.84
2x Change in Risk Free Rate (r = 20%)	104.92	203.96	55.29
4x Change in Risk Free Rate (r = 40%)	115.86	225.23	61.06
2x Change in Time to Expiration (T = 1.0)	104.98	250.64	44.15
4x Change in Time to Expiration (T = 2.0)	115.87	363.40	37.06

Table 4 **Altered Parameters Statistics Values**: This table shows the average final stock values, the maximum point, and the minimum point for figure 16. All values are in dollars.

Figure 16 and Table 3, show what happens to the data and when one parameter is changed from the base parameters. Since the simulations follow a standard normal distribution, the average stopping values should be around the starting value, and for the most part, they are. Looking at the extremes the best and worst-case scenario can be found. For instance, observing the 4x volatility numbers. The maximum run's final value was $1,266 while the minimum run's final value was $6. These are drastically different values, but what is important to take away is that these are 1 run out of 100,000. The average for all the runs, even with the huge extreme values, was still $99.

Looking at the difference in So and K the farther apart the value of the variables get the more the final outcome values decrease. The greater the volatility increases the average ending value maintains roughly the same value from the base but the extremes vastly differ as volatility increases. With the risk-free interest rate the greater the value increases the greater the final outcomes increases. This can be explained since the return for an asset that has zero risk would increase. Lastly, as the time to expiration increases the average final value increases while the extremes spread farther apart. Since a random seed was implemented in order to replicate the results later to come back to visually all the graphs follow the same path. What is important to take away though is the distribution of the final values. It can visually be seen that the image $25 difference between So and K is way more condensed in its final values than the image for 4x volatility.

5 Conclusion

The objective of this research is to serve proof as to the variety of purposes that the Monte-Carlo Method can be utilized in. Specifically within the physics and economics realm through the random walk of a photon and european call and put options. There are plenty of other applications for the Monte-Carlo method within physics and economics than what was observed. In addition, a plethora of other fields utilizes this type of simulation within their area of study. This paper is not meant to serve as the sole narrative for Monte-Carlo simulations within physics and economics but rather as a proof of concept.

6 Future Work

Given this project's structure, there are different paths to explore in the future. If looking at the photon code, future work can be expanded into adding more statistics. Another thing is to look and see if different optical depths affect the movement of the photon. The statistics code may also be altered to assume a non-uniform mean free path throughout the entire simulation. From there, properties of light can be analyzed to see how this process relates to light hitting the earth.

If looking at option code this project strictly followed european options. Monte Carlo simulations can be applied to other options as well, so future work could involve observing how other options such as american and bermudan options behave. Options are not the only thing that a Monte-Carlo simulation can be applied to in economics. A student could take the topic and apply it to another economic problem. This project can be expanded to add a drift rate to observe the time value of money in the model. Dividends expected to be paid could also be included in the current model. A simplification for the Black-Scholes model was made to get from their base to the final call and put equations. A derivation showing that would benefit this project greatly.

7 References

(1) Coleman R. (1974) What is a Stochastic Process?. In: Stochastic Processes. Problem Solvers, vol 14. Springer, Dordrecht. https://doi.org/10.1007/978-94-010-9796-3_1

(2) Mantegna, R. N.; Stanley, H. E. *Introduction to Econophysics, an: Correlations and Complexity in Finance*; Cambridge University Press, 200.

(3) Hull, J. C. (2018). *Options, futures, and other derivatives*. Harlow: Pearson Educational Limited.

(4) Karatzas I., Shreve S.E. (1998) Brownian Motion. In: Brownian Motion and Stochastic Calculus. Graduate Texts in Mathematics, vol 113. Springer, New York, NY. https://doi.org/10.1007/978-1-4612-0949-2_2

(5) Karatzas, I.; Shreve, S. E. Brownian Motion. In *Brownian Motion and Stochastic Calculus*; Springer New York: New York, NY, 1998; pp 47–127. https://link.springer.com/chapter/10.1007/978-1-4612-0949-2_2

(6) Mo, J. Short Timescale Brownian Motion and Applications. University of Texas at Austin August 2015. https://doi.org/10.15781/T2305W https://repositories.lib.utexas.edu/handle/2152/32541

(7) ShieldSquare Captcha https://iopscience.iop.org/article/10.1088/0143-0807/32/3/002/meta#ejp382436s2 (accessed Nov 10, 2020).

(8) Harrison R. L. (2010). Introduction To Monte Carlo Simulation. *AIP conference proceedings*, *1204*, 17–21. https://doi.org/10.1063/1.329563

(9) Joseph, R. The house always wins : Monte Carlo Simulation - Towards Data Science https://towardsdatascience.com/the-house-always-wins-monte-carlo-simulation-eb82787da2a3 (accessed Dec 12, 2020).

(10) Whitcombe, T. (2015, May 07). The life and times of a photon. *Prince George Citizen* Retrieved from https://search.proquest.com/newspapers/life-times-photon/docview/1679656193/se-2?accountid=8268

8 Appendix

8.1 1-D random walk code:

```
import random as rnd # will use random number generator
import matplotlib.pyplot as plt

dx = 1
x = 0
t = 0
xx = [x]
yy = [t]

while t < 100:
   if rnd.random() < 0.5:
       x = x + dx
   else:
       x = x - dx
   t += 1
   xx.append(x)
   yy.append(t)

plt.plot(yy, xx, '-b') # Prepare a plot of sine wave using time and
displacement arrays
plt.title('1-D Random Walk') # Give a title for the sine wave plot
plt.xlabel('x') # Give x axis label for the sine wave plot
plt.ylabel('t') # Give y axis label for the sine wave plot
plt.grid(True, which='both')  # Show grid lines in both directions

plt.show() # Finally, show the plot
```

8.2 2-D random walk code:

```
import random as rnd # will use random number generator
import matplotlib.pyplot as plt
import numpy as np

n = 1000
x = 0
y = 0
xx = [0]
yy = [0]

for i in range (1,n):
   newstep = rnd.randint(1, 4)
   if newstep == 1:
       xx.append(xx[i - 1])
       yy.append(yy[i - 1] + 1)
```

```
    elif newstep == 2:
        xx.append(xx[i - 1] + 1)
        yy.append(yy[i - 1])

    elif newstep == 3:
        xx.append(xx[i - 1])
        yy.append(yy[i - 1] - 1)

    elif newstep == 4:
        xx.append(xx[i - 1] - 1)
        yy.append(yy[i - 1])

plt.plot(xx, yy) # Prepare a  plot of a 2-D randomwalk
plt.title('2-d random walk') # Give a title for the spacetime diagram
plt.xlabel('x') # Give x axis label for the spacetime diagram
plt.ylabel('y') # Give y axis label for the spacetime diagram
plt.grid(True, which='both')  # Show grid lines in both directions

plt.show() # Finally, show the plot
```

8.3 3-D random walk code:

```
import random as rnd # will use random number generator
import matplotlib.pyplot as plt

n = 1000
xx = [0]
yy = [0]
zz = [0]

for i in range (1,n):
    newstep = rnd.randint(1, 6)
    if newstep == 1:
        xx.append(xx[i - 1])
        yy.append(yy[i - 1] + 1)
        zz.append(zz[i - 1])

    elif newstep == 2:
        xx.append(xx[i - 1] + 1)
        yy.append(yy[i - 1])
        zz.append(zz[i - 1])

    elif newstep == 3:
        xx.append(xx[i - 1])
        yy.append(yy[i - 1])
        zz.append(zz[i - 1] + 1)

    elif newstep == 4:
        xx.append(xx[i - 1])
        yy.append(yy[i - 1] - 1)
```

```
        zz.append(zz[i - 1])

    elif newstep == 5:
        xx.append(xx[i - 1] - 1)
        yy.append(yy[i - 1])
        zz.append(zz[i - 1])

    elif newstep == 6:
        xx.append(xx[i - 1])
        yy.append(yy[i - 1])
        zz.append(zz[i - 1] - 1)

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(xx, yy, zz)
plt.title('Random Walk in Three Dimensions')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

plt.show() # Finally, show the plot
```

8.4 Photon Visual Code

```
#*********************************************************************************
****
#
# Name: Peyton Capristo
# Pledge: Peyton Capristo...
#
# This program will perform a monte carlo simulation on a photon traveling
# through the sun
#
# Inputs: mfp = mean free path of a photon in meters
#         radius_to_rz = distance away from center of the sun to the sun's
#                        radiation zone in meters
#
# Outpus: counter = the number of steps the photon took
#         myS = Turtle screen to display the random walk
#
#*********************************************************************************
****


# Make an assumption that the typical mean free path of a photon is 2mm.
# center of sun to radiation zone is 0.714 solar radii =(approx) 4.967x10^8 m

# import libraries
import turtle
import random as random
import math as math

# Initialize some variables
mfp = 0.002 * 1000000000
radius_to_rz = 4.697*10**8
position_tracker = 0
```

```
x = 0
y = 0
counter = 0

# Sets up the turtle screen
myS = turtle.Screen()
myS.tracer(0)
myS.title("Random Walk") # Titles the screen
border = radius_to_rz * 1.1 #Sets up a variable for the boundary of the page
myS.setworldcoordinates(-border, -border, border, border) # Sets up the
boundaries on the page itself


# Set up boundary circle
turtle.hideturtle()
turtle.speed(0)
turtle.penup()
turtle.goto(0, -4.697*10**8)
turtle.pendown()
turtle.circle(radius_to_rz)
turtle.penup()
turtle.goto(0, 0)
turtle.pendown()

# Do the random walk
while position_tracker < radius_to_rz:
   angle = random.random()*2*math.pi   # generates a random angle
   x = x + math.cos(angle) * mfp  # multiply by a large number so the program
runs shorter than 30,000-100,000 years.
   y = y + math.sin(angle) * mfp  # multiply by a large number so the program
runs shorter than 30,000-100,000 years.
   turtle.goto(x,y)   #for drawing run this, if looking for stats comment out
   position_tracker = math.sqrt(x**2 + y**2)

   counter += 1

# Prints the number of times the photon moved one step
print(counter)

# Updates and displays the turtle screen
myS.update()
myS.mainloop()
```

8.5 Photon Statistics 3D Code

```
#*********************************************************************
****
#
# Name: Peyton Capristo
# Pledge: Peyton Capristo...
#
# This program will perform a monte carlo simulation on a photon traveling
# through the sun and calculate the average number of steps, the standard
# deviation of that, the standard error, a arbitrary drift value
#
```

```
# Inputs: mfp = mean free path of a photon in meters
#         radius_to_rz = distance away from center of the sun to the sun's
#                        radiation zone in meters
#
# Outpus: avg_steps_taken = the average number of steps the photon took
#         std = standard deviation
#         alpha = an arbitrary drift value
#         standard_error = the standard error in the run
#         time_elapsed = time for the photon to complete the run
#
#*************************************************************************
****


# Make an assumption that the typical mean free path of a photon is 2mm.
# center of sun to radiation zone is 0.714 solar radii =(approx) 4.967x10^8 m


# import libraries
import random as random
import math as math
import numpy as np

final = []
N = 100000
c = 3.0*10**8

for i in range (1, N):
   # Initialize some variables
   mfp = 0.002 * 1000000000
   radius_to_rz = 4.697*10**8
   position_tracker = 0
   x = 0
   y = 0
   z = 0
   counter = 0
# Do the random walk
   while position_tracker < radius_to_rz:
       cos_phi = random.uniform(-1, 1)  # generates a random angle
       sin_phi = math.sqrt(1 - cos_phi ** 2)
       theta = random.random() * 2 * math.pi   # generates a random angle
       x = x + sin_phi * math.cos(theta)  * mfp  # multiply by a large number
so the program runs shorter than 30,000-100,000 years.
       y = y + sin_phi * math.sin(theta) * mfp  # multiply by a large number
so the program runs shorter than 30,000-100,000 years.
       z = z + cos_phi * mfp
       position_tracker = math.sqrt(x**2 + y**2 + z**2)

       counter += 1

   final.append(counter)

# Prints a list with each of the counts for the runs
# print("final list = ", final)
# calculate and print average of the counts
avg_steps_taken = sum(final) / len(final)
print("avg = ", avg_steps_taken)
```

```
# Calculates a prints standard deviation of the run
std = np.std(final)
print("std = ", std)

alpha = radius_to_rz / (mfp * np.sqrt(avg_steps_taken))
print("alpha value =", alpha) # proportionality coefficient

standard_error = std / np.sqrt(N)
print("standard error =", standard_error)

time_per_step = mfp / c
time_elapsed = time_per_step * avg_steps_taken
print("time_elapsed =", time_elapsed)
```

8.6 Time Elapsed

```
#*********************************************************************
****
#
# Name: Peyton Capristo
# Pledge: Peyton Capristo...
#
# This program will perform a monte carlo simulation on a photon traveling
# through the sun and find how long the photon takes to travel a certain
distance
#
# Inputs: mfp = mean free path of a photon in meters
#         distances = a list containing different distance for the photon to
travel
#
# Outpus: counter = the number of steps the photon took
#         time_elapsed = The average time taken per step
#
#*********************************************************************
****


# Make an assumption that the typical mean free path of a photon is 2mm.
# center of sun to radiation zone is 0.714 solar radii =(approx) 4.967x10^8 m


# import libraries
import random as random
import math as math
import numpy as np
import matplotlib.pyplot as plt

c = 3.0*10**8
N = 100
distances = [0.01, 0.1, 1, 10]
total_times = []

for distance in distances:
   final = []
   for i in range(0, N):
       # Initialize some variables
```

```
        mfp = 0.002
        position_tracker = 0
        x = 0
        y = 0
        z = 0
        counter = 0
# Do the random walk
        while position_tracker < distance:
            cos_phi = random.uniform(-1, 1)  # generates a random angle
            sin_phi = math.sqrt(1 - cos_phi ** 2)
            theta = random.random() * 2 * math.pi   # generates a random angle
            x = x + sin_phi * math.cos(theta)  * mfp  # multiply by a large
number so the program runs shorter than 30,000-100,000 years.
            y = y + sin_phi * math.sin(theta) * mfp  # multiply by a large
number so the program runs shorter than 30,000-100,000 years.
            z = z + cos_phi * mfp
            position_tracker = math.sqrt(x**2 + y**2 + z**2)

            counter += 1
        final.append(counter)
    avg_steps_taken = sum(final) / len(final)

    time_per_step = mfp / c
    time_elapsed = time_per_step * avg_steps_taken
    print("time_elapsed =", time_elapsed)
    total_times.append(time_elapsed)

# Set ups and displays the graph
plt.plot(distances, total_times) # Prepare a plot of time elapsed vs displacement
plt.title('Time elapsed vs Displacement') # Give a title for the graph
plt.xlabel('Distance Travled (m)') # Give x axis label for the graph
plt.ylabel('Time Elapsed (s)') # Give y axis label for the graph
plt.grid(True, which='both')  # Show grid lines in both directions

plt.show() # Finally, show the plot
```

8.7 Option Code

```
#*******************************************************************************
# Name: Peyton Capristo
# Pledge: Peyton Capristo...
#
# This program will perform a Monte-Carlo Simulation of a European call option
# given a certain set of inputs and produce a graph with different outcomes.
#
# Inputs: So = Current Stock price
#         K = Strike price
#         r = risk free interest rate
#         sigma = volatility of the asset
#         mu = expected return
#         T = Time to expiration
#
# Outputs: C = price of black scholes call option
#          S[i] = path of the option price
#
```

```
#*************************************************************************
*

# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats

# Initialize a random seed to reproduce data
np.random.seed(1)

# Now need to define the Black Scholes Metron Call option Pricing Model

def BSM_Call (So, K, r, sigma, T):
   d1 = (np.log(So/K) + (r + sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
   d2 = d1 - sigma * np.sqrt(T)
   C = (So * stats.norm.cdf(d1)) - (K * np.exp(-r * T) * stats.norm.cdf(d2))
   return C

def BSM_Put (So, K, r, sigma, T):
   d1 = (np.log(So/K) + (r + sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
   d2 = d1 - sigma * np.sqrt(T)
   P = (K * np.exp(-r * T) * stats.norm.cdf(-d2)) - (So * stats.norm.cdf(-d1))
   return P

# Define Inputs
So = 95
K = 100
r = 0.1
sigma = 0.2
mu = r # Make assumption that the expected return from the underlying asset is
the risk-free interest rate
T = 0.5


# Number of simulation is equal to 100000
N = 100000

# Number of trading days in a year =  252
NumberOfSteps = int(T * 252)
dt = T / NumberOfSteps

# Initialize an array S and its starting point to be equal to the current
stock option price
S = np.zeros((NumberOfSteps, N))
S[0] = So

# This for loop runs the simulation starting at the first point until the
expiration date for an option path
# It does this for the number of simulations N desired
for i in range (1, NumberOfSteps):
   Z = np.random.standard_normal(N)
   S[i] = S[i-1] * np.exp((mu - 0.5 * sigma ** 2) * dt + (sigma * np.sqrt(dt)
* Z))

# Averages the Final S[i] values
```

```
Avg_Si = sum(S[i]) / len(S[i])

# Finds highest and lowest ending value
Max_Si = np.amax(S[i])
Min_Si = np.amin(S[i])

# Collects all the final data points and calculates the BSM value
All_Final_Call_Prices = BSM_Call(S[i], K, r, sigma, T)

# Averages the final BSM values
Avg_Final_Call_Price = sum(All_Final_Call_Prices) / len(All_Final_Call_Prices)

# Sums and Discounts all the values
Sum_and_Discounted_Call_Price = np.exp(-r * T) * 1 / N *
np.sum(np.maximum(S[-1] - K, 0))

# Collects all the final data points and calculates the BSM value
All_Final_Put_Prices = BSM_Put(S[i], K, r, sigma, T)

# Averages the final BSM values
Avg_Final_Put_Price = sum(All_Final_Put_Prices) / len(All_Final_Put_Prices)

# Sums and Discounts all the values
Sum_and_Discounted_Put_Price = 1 / N * np.sum(np.maximum(K - S[-1], 0)) *
np.exp(-r * T)


# print("All_Final_Call_Price = ", All_Final_Call_Price) #If want list of all
final uncomment out
print("Average Final Stock Values =", Avg_Si)
print("Maximum Ending Stock Value =", Max_Si)
print("Minimum Ending Stock Value =", Min_Si)
print("\nInitial BSM call price =", BSM_Call(So, K, r, sigma, T))
print("Average Final Call Price =", Avg_Final_Call_Price)
print("Sum and discounted Call Price =", Sum_and_Discounted_Call_Price)
print("\nInitial BSM put price =", BSM_Put(So, K, r, sigma, T))
print("Average Final Put Price =", Avg_Final_Put_Price)
print("Sum and discounted Put Price =", Sum_and_Discounted_Put_Price)


# Set ups and displays the graph
plt.plot(S[:,0:100000]) # Prepare a  plot of option price movements
plt.axhline(K, c="k", xmin=0,      # Plot a Horizontal line at the strike
price
           xmax=NumberOfSteps,
          label="Strike Price")
plt.title('$25 Differnce Between So and K') # Give a title for the graph
plt.xlabel('Time in Days') # Give x axis label for the graph
plt.ylabel('Option Price') # Give y axis label for the graph
plt.grid(True, which='both')  # Show grid lines in both directions

plt.show() # Finally, show the plot
```