MF4090 Research Project Report

Nathaniel Dureza Volfango

115321436

Supervisor - Cónall Kelly

Coordinator - Spyridon Dendrinos

Estimating the Mean Percentage Return (MPR) of Structured Financial Products

School of Mathematical Sciences
University College Cork
Ireland
27th March 2019

Abstract

Stock market linked savings accounts have become more and more popular due to interest rates on most deposit accounts being close to zero. These financial products offer a higher return by combining exposure to risky assets while still providing some protection against a fall in the stock market. The returns on such products are not guaranteed, so it is useful to be able to estimate their mean percentage return (MPR) under various market conditions. This task requires modelling and simulating the underlying asset and applying an appropriate implementation of the product's payoff structure. In this project, we will look at the work already done by others on the theory of the MPR and discuss further advancements to the theory.

Contents

1 Introduction

Before we proceed, it is important to distinguish the purpose of this project to prevent any misunderstanding. Our intention is not to look at historical price data, determining the best stochastic model that fits that data, estimating the parameters of that underlying model, and then finally forecasting future price data. We refer the reader to [4, 5] for more information in this direction. Instead, our task of estimating the MPR of any structured financial product under various market conditions can be outlined as follows:

1. We make an assumption about the stochastic process that the underlying asset will follow in the future.

2. We choose specific values for the parameters of the model to reflect the market conditions which we assume will hold true in the future.

3. We design an algorithm that simulates the underlying model a given number of times, each time implementing the payoff structure over a certain period as specified in the product description and calculating a percentage return.

4. We calculate the average payoff, giving us the MPR.

The second point is quite important; having parameters in our model that reflect specific characteristics of the market and which we can vary is very useful. This will allow the product designer or potential investor to consider the sensitivity of the MPR under different market conditions and see at which key values of the parameter/s the product will give the same expected return as a risk-free asset.

For example, the Geometric Brownian Motion has a (constant) drift coefficient μ which determines the overall trend in the asset price and a (constant) diffusion coefficient σ which determines the volatility of the asset price. We can also use more complex models that allow us to create more realistic simulations. For example, rather than keeping the diffusion coefficient constant, we could also model volatility with another stochastic process (see Section 3.2). This would make sense since it is universally agreed that volatility in the financial markets rarely remains constant for a long period.[13]

2 Background and Preliminary Work

A significant amount of groundwork has already been laid in [1] where the authors begin by establishing the theory on the MPR and then apply this theory to real savings plans offered by financial institutions. Some time was spent reviewing the work, attempting to replicate the implementations from R into Python, conducting analysis on the implementations, and even making some improvements before progressing into further development of the method. Since my project is a progression from [1], it is important that we give a summary of the work that was done in the paper to provide some context, then we will go through the improvements and preliminary analysis that we did before we embarked on our own additions to the MPR framework.

2.1 Description of the Savings Plans

We have provided the description of each savings plan early on in this paper given that much of the work is based on the rule set of each plan. The authors chose 5 different savings plans of increasing complexity, linked to the FTSE 100 (which we will call the 'Index'), with Plan P1 being the simplest and Plan P5 being the most complicated. We have included descriptions of P1 and P2 to better illustrate the range of complexity between the plans, but since the treatment of these plans does not require any numerical analysis, not much attention is given to them in [1] or in this paper (see Section 6.2.1-6.2.3 for the derivations of the MPR for P1-P3). We will limit the focus of this project and its developments to Plans P3-P5.

P1: A 3-year plan delivering a 13% return is received if the Index is higher at the end of the plan and a return of 0% if it is lower.

P2: A 5-year plan delivering a 40% return if the Index is increased by more than 40% at the end of the plan and a return of 0.5% if the Index is not increased by more than 0.5%. If at the end of the plan the Index is increased by a percentage between 0.5% and 40%, a return of the same percentage is received.

P3: A 5-year plan delivering a 35% return at the end of the plan if the Index was greater than its initial level at the end of Year 3, 4 and 5, but if the Index is lower at the end of any of these three years, a return of 0% is received.

P4: A 6-year plan; the Index is measured at the start of the Plan, and then on a yearly basis thereafter. If the average closing level of the Index for the five business days up to and including the anniversary date is higher than 90% of the initial level, a fixed income of 5% gross will be paid for that year. If the anniversary Index level is equal to or below 90% of the Initial Index Level, no income will be paid. However, should the Index meet the required level on any future anniversary, any previously missed income payments will be added back and paid out.

P5: A 6-year plan; it offers up to 6.5% each year, with income being accrued for each week the Index closes between 4,500 and 9,000 points - if it closes outside of this range, no income will be added for that week. Income payments will be calculated and paid at the end of each quarter, with the total income payment being proportionate to the number of weekly observation dates that the Index is between the 4,500 to 9,000 range. For example for a quarter consisting of 13 weeks, should the Index finish between the range on 11 of 13 Weekly Observation Dates, the quarterly income payment would be 1.375% $\left(\frac{0.065}{4} * \frac{11}{13}\right)$. If the Index is lower than 4,250 points at the end of the plan, the initial investment will be reduced by 1% for every 1% that the final Index level is below the initial Index level.

2.2 Summary of the Foundational Article

Plans P1 and P2 are simple enough that, where the probability distribution of the underlying stochastic process is known, an analytic solution can be derived for the MPR. On the other hand, Plans P4 and P5 require a numerical approach (Monte Carlo) to approximating the MPR. Interestingly, Plan P3 lies on the cusp of simplicity and complexity because a closed form expression can be obtained, with some effort, but due to the nature of the solution, a numerical solution is warranted. The analytic solution for P3's MPR is given by the following expression:

$$\text{MPR} = 0.35\left(1 - N(d(X_0,3)) - \int_{d(X_0,3)}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-0.5\bar{x}_3^2} I(-\sqrt{3}\bar{x}_3 - 4(\mu - 0.5\sigma^2)/\sigma)\, d\bar{x}_3\right), \tag{2.1}$$

where

$$I(x) := N(x) + \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-0.5\bar{x}_4^2} N\left(x - \bar{x}_4 - \frac{\mu - 0.5\sigma^2}{\sigma}\right) d\bar{x}_4,$$

$$d(x,t) := \frac{\log({}^{X_0}/_{x}) - (\mu - 0.5\sigma^2)t}{\sigma\sqrt{t}},$$

$$\bar{x}_3(X_3) = \frac{\log({}^{X_3}/_{X_0}) - 3(\mu - 0.5\sigma^2)}{\sigma\sqrt{3}}, \qquad \bar{x}_4(X_4) = \frac{\log({}^{X_4}/_{X_3}) - (\mu - 0.5\sigma^2)}{\sigma},$$

where $\bar{x}_i$ is a function of X_i, $i = 3, 4$, and it is assumed that the price of the underlying process $X(t)$ follows the Black-Scholes (GBM) model with drift μ and volatility σ (see Equation (2.2)). To avoid distracting the reader, we have left the complete derivation and definition of notations of (2.1) in Section 6.2.3. If we were to increase the complexity of the rule set or change the underlying stochastic model, this would make a closed form solution unobtainable. It is at this point then that we will start using Monte Carlo (MC) simulations to estimate the MPR.

The main body of [1] can be divided into two sections, each one being a treatment of the MPR based on a different stochastic model. In the first section the authors assume that the price of the Index, denoted by $X(t)$, follows the Geometric Brownian Motion (GBM), given by the linear SDE

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t), \quad t \geq 0, \tag{2.2}$$

with initial condition $X(0) = X_0$, where μ and σ are positive constants, $W(t)$ is the standard Brownian motion or Wiener process (Definition 6.1.1) defined on a complete (filtered) probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ (Definitions 6.1.2 and 6.1.3) with the filtration $\{\mathcal{F}_t\}_{t\geq 0}$ (Definition 6.1.4) satisfying the usual conditions, i.e. it is right continuous and increasing while $\mathcal{F}_0$ contains all $\mathbb{P}$-null sets, and t is a yearly time unit. The SDE (2.2) has the well-known explicit solution

$$X(t) = X_0 \exp\Big((\mu - 0.5\sigma^2)t + \sigma W(t)\Big), \quad t \geq 0. \tag{2.3}$$

The analytic solutions were derived for P1, P2 and P3, but the MPR for P4 and P5 was obtained numerically via Monte Carlo simulations.

In the second section, it is assumed that the Index follows the mean reverting Cox-Ingersoll-Ross (CIR) model

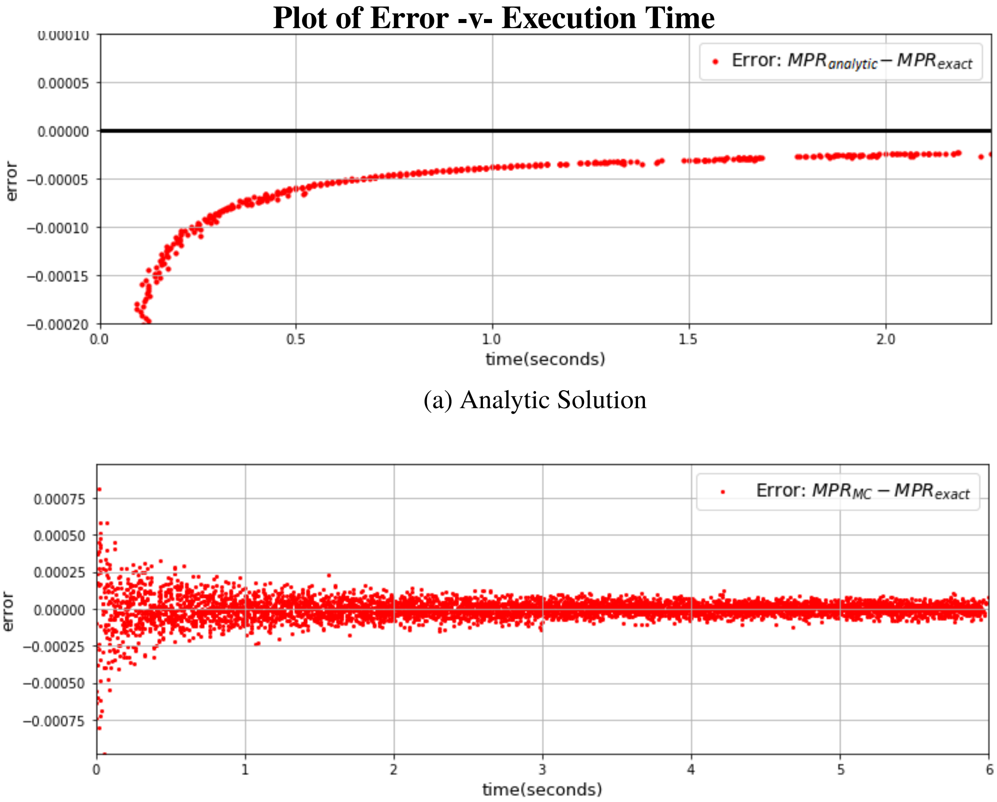
$$dX(t) = \mu(\lambda - X(t))dt + \sigma\sqrt{X(t)}dW(t), \quad t \geq 0. \tag{2.4}$$

The authors show how to use MC simulations to compute the MPR for Plans P3 to P5, using the Euler-Maruyama method to generate independent sample paths of the Index level. They also make note of how the MPR is affected when the parameters μ and σ are varied.

2.3 Preliminary Analysis and Observations

As mentioned above for Plan P3, an analytic solution was derived and a numerical solution implemented. Both solutions have a numerical element, with the analytic solution involving the approximation of nested integrals, and the numerical solution consisting of simulations. It is of interest to see how the two solutions compare with each other in terms of accuracy and execution time.

In general, greater accuracy in a numerical solution requires an increase in the computational cost, and this is true for both methods. For the analytic solution, the accuracy improves as the Riemann sum partitions become finer, while for the numerical solution, we increase the number of MC simulations. To compare the two methods objectively, we will look at the computational cost of each one (as measured by the time taken to calculate the MPR) required to achieve a given level of accuracy, and vice versa.



(a) Analytic Solution

(b) Numerical Solution

Figure 1a and Figure 1b show the plots of the error of the analytic and numerical (MC) solutions, respectively. Here, we have defined the true solution to be the result of calculating the MPR using the analytic solution for a very small step size. We use this to calculate the error of our solutions. Each red dot in the plots represents an MPR value. We note a few key characteristics and comparisons between the two solutions:

- Both solutions converge to the true MPR value as the computational cost t increases.
- The analytic solution converges monotonically and with relatively little variance, while the numerical solution only converges in mean to the true value.
- The variance in the error of the numerical solution decreases as t increases.

- At $t = 2$ seconds, an accuracy of 4 decimal places is already ensured by the analytic solution, whereas the numerical solution cannot guarantee this, and we would need to calculate several more MPR values then recalculate the mean in order to get a satisfactory MPR estimate. So it is quite clear then that the analytic solution converges faster.
- The analytic solution is more precise, i.e. the same number is obtained for every calculation for a fixed computational cost (determined by the step size), while the numerical solution will always give a different value for a fixed computational cost (determined by the number of Monte Carlo simulations).

We noted in the third point that the variance of the error in the MC estimate decreases with t. An immediate question would be whether this variance decreases to zero eventually, i.e. if $t \to \infty \implies \text{Error} \to 0$. Note that the computational time t is determined by the number of Monte Carlo samples we generate in our calculation. This means that increasing the number of samples reduces the error of our MPR estimate. It has been shown that Monte Carlo converges with order $\mathcal{O}(\sqrt{N})$, where N is the number of samples. By the law of large numbers, our sample average (represented by each red dot) converges to the expected value, i.e. the true MPR under the plan, and by the central limit theorem, we can express the error between our approximation and the true solution as a Normal random variable with mean 0 and variance $\mathcal{O}(1/\sqrt{N})$. Therefore if we were to calculate the MPR for an infinite number of samples, we would get the true MPR value. See [16] for more details on the treatment of the Monte Carlo method and its convergence.

In practice, however, we can be completely satisfied with an accuracy of 5-6 decimal places in our approximate solution. However, the number of samples (and hence the computational cost) required to achieve this level of accuracy is relatively large, so we would be interested in making our implementations as efficient as possible (see Section 3.3) and in variance reduction techniques which would allow us to get the same order of accuracy but with a lower computational cost (see Section 3.1).

Now although it might seem like the analytic solution is better, we would argue that the numerical solution is actually more useful and advantageous in the context of estimating the MPR for the following reasons:

- **The variance is much more easily obtainable.** In fact, it is just one Python command away. On the other hand, we would need to derive the variance of the analytic solution, requiring much more effort than for the derivation of the expected value. The variance is important because we are more interested in having some kind of confidence interval for the mean value than we are in a single point estimate, and any potential stakeholder in a financial product will want to consider the likelihood of receiving an overall return that is much different from the estimated MPR.

We use the sample variance of the output as our estimator for the true variance of the underlying model, and similar to how the sample mean converges to the true expected value (by the law of large numbers), we can also obtain an accurate estimate of the underlying variance with a sufficiently large number of MC samples.

- **The numerical (Monte Carlo) method is robust and can easily accommodate other stochastic models.** The derivation of the analytic solution involved the use of transition probabilities which are quite easy to work with for the GBM. However, one would encounter some difficulties very quickly when dealing with more complicated stochastic models.
- **The numerical method is easier to implement.** The *P3-analytic.py* file gives the Python implementation of P3's analytic solution. It has fewer lines of code than the numerical approximation of the nested integrals in the analytic solution, and no prior derivation is required.

I have included a documentation of the shortcomings that I came across in [1] which I go through with some detail in Section 5.2. For now, we will only mention that the implementation of Plan P3's analytic solution in that article is not robust - it did not allow for varying partition sizes for the Riemann sums. So, a more flexible implementation was created here for the purpose of the above analysis.

3 Developments to the MPR Framework

We will now look at improvements on the method of estimating the MPR of any given savings plan. As mentioned, we will be limiting our focus to P3, P4 and P5 in our analyses. We will discuss the improvements we have made under the following headings:

1. **Variance reduction** in the resulting MC output
2. Assuming a **stochastic volatility** in the underlying model
3. **Efficiency** in the implementation

3.1 Variance Reduction

3.1.1 Motivation

When estimating the MPR, we are trying to calculate the expected value given by

$$\text{MPR} = \mathbb{E}[h(X(T))], \tag{3.1}$$

where h is the payoff function and $X(T)$ is a random variable representing the path of the Index between now (time $t = 0$) and the maturity date T. Note that in this case, we are referring to the entire trajectory over time $[0, T]$, and we do not restrict the definition of $X(T)$ to the final time value. This is necessary because some payoff functions are path-dependent and hence require us to look at different points on the trajectory.

The expectation (3.1) can be approximated by the Monte Carlo estimate

$$\hat{P} = \frac{1}{N} \sum_{i=1}^{N} h\left(X_T^{(i)}\right) \approx \text{MPR}, \tag{3.2}$$

where $X_T^{(i)}$ is an approximation of the i^{th} trajectory. In theory, we are trying to obtain the expected value of a random variable (the percentage return) given by the underlying SDE. However, additional errors are introduced when we use computational approximations. In particular, the overall error in our approximation has the following form: [6]

$$\begin{aligned}\hat{P} - \mathbb{E}[h(X(T))] &= \hat{P} - \mathbb{E}\big[h(X(T)) - h(X_T) + h(X_T)\big] \\ &= \hat{P} - \mathbb{E}\big[h(X_T)\big] + \mathbb{E}\big[h(X_T) - h(X(T))\big].\end{aligned} \tag{3.3}$$

Note that X_T can be obtained either via discretisation methods (Euler-Maruyama or Milstein) or directly if the analytic solution is known, as with the GBM. The expression (3.3) breaks down the error into two terms. The sampling error $\hat{P} - \mathbb{E}[h(X_T))]$ is concerned with how well we can approximate an expected value from a finite number of samples, and this generally decreases if we take more sample paths. The discretisation error $\mathbb{E}[h(X_T) - h(X(T))]$ arises when we approximate the solution of the underlying SDE with a difference equation (we do not have this problem when we compute our approximations using the analytic solution under the GBM model). This will generally decrease if we reduce the step size.

Increasing the sample size and reducing the step size will improve the precision and accuracy of our MPR estimate, but given the added computational cost, we are interested in techniques that will reduce the variance of our MPR estimate for a given computational cost.

3.1.2 Antithetic Variates

Our first variance reduction technique may be motivated as follows. Let X_1 and X_2 be random variables known to have the same expectation and variance, i.e.

$$\mathbb{E}[X_1] = \mathbb{E}[X_2] = \mathbb{E}\left[\frac{X_1 + X_2}{2}\right], \tag{3.4}$$

$$\mathbb{V}[X_1] = \mathbb{V}[X_2], \tag{3.5}$$

and suppose that we would like to estimate $\mathbb{E}[X_1]$.
Then

$$\begin{aligned}\mathbb{V}\left[\frac{X_1 + X_2}{2}\right] &= \frac{1}{4}\Big(\mathbb{V}[X_1] + \mathbb{V}[X_2] + 2\mathrm{Cov}[X_1, X_2]\Big) \\ &= \frac{\mathbb{V}[X_1]}{2} + \frac{\mathrm{Cov}[X_1, X_2]}{2}\end{aligned} \tag{3.6}$$

and

$$\mathrm{Cov}[X_1, X_2] < \mathbb{V}[X_1] \implies \mathbb{V}\left[\frac{X_1 + X_2}{2}\right] < \mathbb{V}[X_1] \tag{3.7}$$

By (3.4) and (3.7), we have an unbiased estimator for $\mathbb{E}[X_1]$ with a variance smaller than $\mathbb{V}[X_1]$ given by

$$Y = \frac{X_1 + X_2}{2}. \tag{3.8}$$

In this case, X_2 is referred to as the *antithetic variate* of X_1. The same result should hold for the function of a random variable, i.e. for $X_1 \equiv h(X)$.

In the context of our problem, each approximation X_T is generated using random samples Z_j from a Standard Normal distribution. For each $Z_j \sim N(0,1)$, the corresponding antithetic variate is $-Z_j \sim N(0,1)$. This is quite clear given that

$$\mathbb{E}\left[\frac{Z_j + (-Z_j)}{2}\right] = 0 = \mathbb{E}[Z_j],$$

and

$$\mathrm{Cov}[Z_j, -Z_j] = -\mathbb{V}[Z_j] < \mathbb{V}[Z_j].$$

Then for every path $X_T^{(i,+)}$ generated from a set of Normal random samples $\{Z_j\}_{j=1}^M$, another path $X_T^{(i,-)}$ is generated by transforming the same samples $\{-Z_j\}_{j=1}^M$. The i^{th} trajectory and its corresponding payoff are

$$X_T^{(i)} = \frac{X_T^{(i,+)} + X_T^{(i,-)}}{2} \qquad \text{and} \qquad h\big(X_T^{(i)}\big) = \frac{h\big(X_T^{(i,+)}\big) + h\big(X_T^{(i,-)}\big)}{2},$$

respectively. This choice of antithetic variate is also computationally inexpensive since it only requires a simple transformation of the sample.

More details on the use of antithetic variates for variance reduction in Monte Carlo methods can be found in [10], and certainly more analysis of the theory of variance reduction can be done here, but since we are more concerned with the variance reduction in real simulation and because each plan has a unique payoff structure, it would be much easier to just observe the variance reduction between the original and antithetic variate implementations.

To estimate the level of variance reduction in our AV implementations, we ran the original and antithetic variate implementations 100 times. Each time, we calculated the variance of both outputs and compared them. Figure 2 shows the mean factor of variance reduction as well as the mean increase in computation time, e.g. on average, the variance of our estimate for P5 is 4.119 times smaller and the calculation is 1.04 times faster when using antithetic variates. We have also included the number of paths simulated, the computation time (in mm:ss format) for the 100 runs under each plan and the number of Gaussian samples Z_j that are generated for each trajectory. Keep in mind that for the antithetic variate implementation, we ran half as many simulations as the original since the antithetic variate path is itself another sample trajectory, i.e. two paths are generated in each simulation under the AV implementation. Hence, we generated the exact same number of paths under both implementations. This would allow us to make valid comparisons.

We can see that the AV implementation is slightly faster, and the variance has been reduced

Plan	Paths Simulated	Mean Variance Reduction	Time (Original)	Time (AV)	Mean Increase in Computation Time	Z_j Samples Used Per Path
P3	5,000,000	3.14	2:08	1:41	1.26	3
P4	500,000	10.27	5:47	5:40	1.02	30
P5	50,000	4.119	10:50	10:23	1.04	304

Figure 2: Results of variance reduction using antithetic variates

significantly under each plan. We also found that a variance reduction was achieved 100% of the time. It is quite obvious then that antithetic variates will improve our estimation of the MPR, although the extent to which it will do this is not certain (see Section 5.1.2). Even though the computational cost is greater, we can see that using antithetic variates is more effective than just using 'brute force' Monte Carlo to generate the same number of samples.

The AV implementation is slightly faster than the original implementation because, as we mentioned, the antithetic variate paths are also sample trajectories (with the same expectation), but they were generated via a simple transformation of other existing samples. This is less costly than using a pseudo-random number generator to create another sample. So under the AV implementation, half of the samples are obtained quicker than normal.

The significantly reduced variance under the AV implementation can be explained by (3.7). Under 'brute force' Monte Carlo, all samples are independently generated. Using antithetic variates, we strategically sample half of the samples so that they have a negative correlation with the other half.

Finally, we would like to mention that the original and AV implementations were ran using vectorisation (see Section 3.3.2). Otherwise, the calculations would have taken much longer.

3.1.3 Multilevel Monte Carlo

Our second variance reduction technique was proposed by Giles[8, 9] as an improvement to the approach of discretising with a numerical method and applying standard Monte Carlo. His *Multilevel Monte Carlo* method, combined with a higher order numerical method, offers a speed up of $O(\epsilon^{-1})$.

Overview

Consider Monte Carlo path simulations with different time steps $h_l = 2^{-l}T$, $l = 0, 1, ..., L$. On the coarsest level, $l = 0$, the simulations use only one time step, while on the finest level, $l = L$, they use 2^L time steps. For any given path/trajectory $X(t)$, with payoff $h(X(t))$, let $\hat{P}_l$ denote its approximation using a numerical discretisation with time step h_l. Because of the linearity of the expectation operator, we can see that

$$\mathbb{E}[\hat{P}_L] = \mathbb{E}[\hat{P}_0] + \sum_{l=1}^{L} \mathbb{E}[\hat{P}_l - \hat{P}_{l-1}]. \tag{3.9}$$

It should be obvious that the right hand side is a telescoping sum. The expression (3.9) breaks down this expectation on the finest level (which is the quantity that we are interested in calculating) into the expectation on the coarsest level plus a sum of 'corrections' at finer levels. The idea behind the multilevel method is to independently estimate each of the expectations on the right hand side in a way that minimises the overall variance for a given computational cost.

Let $\hat{Y}_0$ be an estimator for $\mathbb{E}[\hat{P}_0]$ using N_0 samples, and let $\hat{Y}_l$ for $l > 0$ be an estimator for $\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}]$ using N_l samples. The simplest estimator is a mean of N_l independent samples, giving an overall estimator

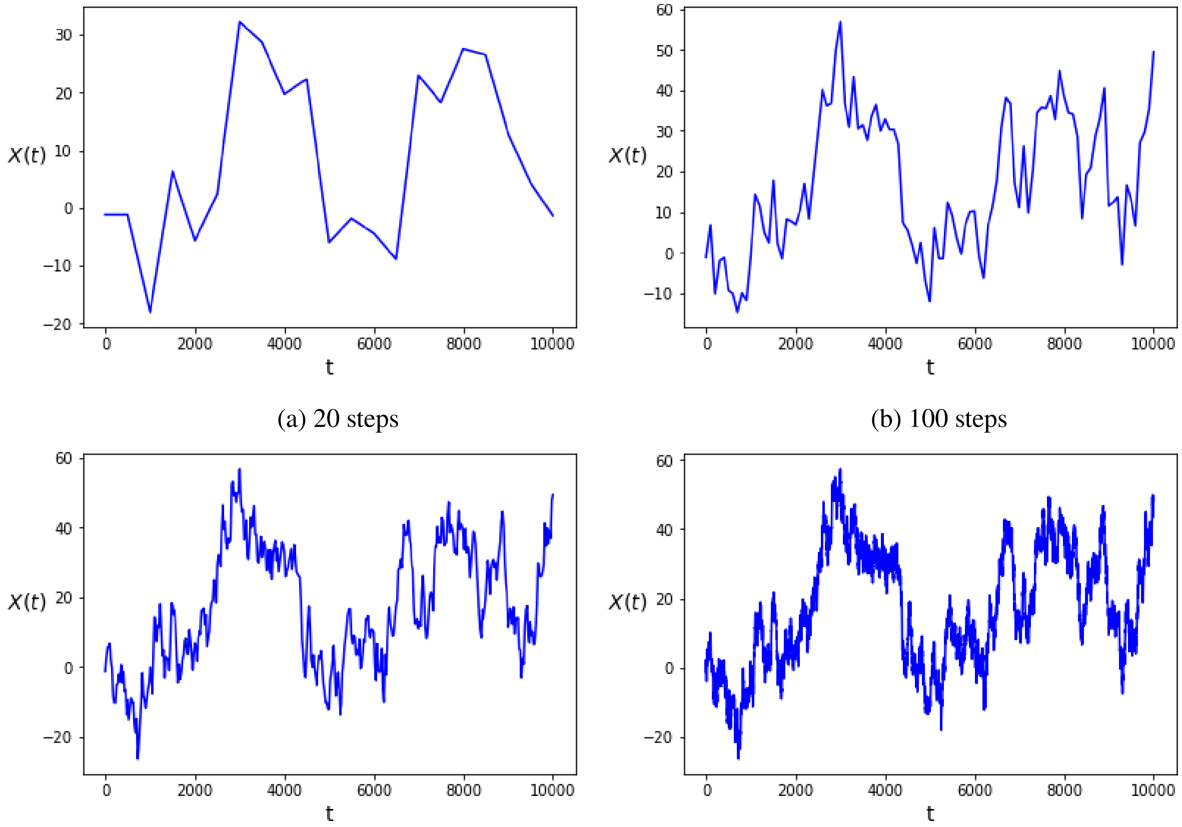
$$\hat{Y} = \sum_{l=0}^{L} \hat{Y}_l = \sum_{l=0}^{L} \left[\frac{1}{N_l} \sum_{i=1}^{N_l} \left(\hat{P}_l^{(i)} - \hat{P}_{l-1}^{(i)} \right) \right] \approx \mathbb{E}[\hat{P}_L], \tag{3.10}$$

where $\hat{P}_{-1} \equiv 0$. The key point here is that $\hat{P}_l^{(i)} - \hat{P}_{l-1}^{(i)}$ consists of two discrete approximations with different time steps but the same Brownian path. Figure 3 gives an intuitive picture of how we calculate our estimator for $\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}]$ for $l > 0$. We discretise the same Brownian path at different Δt levels, then we apply the payoff function to the paths at different levels and calculate the difference. This difference can be understood as a correction that is added at every extra level l. On the other hand, in the standard Monte Carlo method, we approximate the payoff with the following estimator:

$$\hat{Y} = \frac{1}{M} \sum_{j=1}^{M} \hat{P}_L^{(j)} \approx \mathbb{E}[\hat{P}_L], \tag{3.11}$$

where $M \gg N_L$ is the number of independent samples. The advantage of MLMC method over the standard MC is that most of the sample paths are calculated on the coarser levels with relatively large step sizes, and only relatively few samples are calculated at the smoother levels with high resolution.

Given $\epsilon > 0$, we may achieve accuracy ϵ by combining Euler-Maruyama and standard Monte Carlo at an overall cost that scales like ϵ^{-3}[6]. Giles shows in [8, 9] that the same level of accuracy can be achieved using Euler-Maruyama in a multilevel Monte Carlo setting at an overall cost that scales like $\epsilon^{-2}(\log\epsilon)^2$. He then finds in [7] that by using a higher strong order method, such as Milstein, it is possible to reduce the MLMC cost to $O(\epsilon^{-2})$. This improvement in computational complexity count of up to $O(\epsilon^{-1})$ would be a useful tool to include into our method of MPR estimation.



(a) 20 steps

(b) 100 steps

(c) 400 steps

(d) 10,000 steps

Figure 3: Brownian path approximated at different levels of discretisation

Implementation

Under the MLMC method, an initial calculation is first done at levels $0, 1, \ldots, L$ with initial number of samples N_0. The resulting variance from the estimate is then used to determine the optimal number of samples $\hat{N}_l$ at each level l in order to achieve a given accuracy ϵ specified beforehand. The algorithm is repeated until extra samples are no longer required under all levels, because our sample variance has 'converged' at all levels. The values for the overall estimators for the MPR and variance are then calculated. Note that the general formula for determining the optimal number of samples $\hat{N}_l$ is given by

$$\hat{N}_l = \left\lceil 2\epsilon^{-2}\sqrt{\frac{V_l}{2^l}}\left(\sum_{l=0}^{L}\sqrt{V_l 2^l}\right)\right\rceil \tag{3.12}$$

However, due to time constraints, our work with MLMC in the context of MPR estimation is quite limited. We did however create an implementation (based on Giles' *MLMC driver routine* code in [8]) for P3, assuming a stochastic (CIR) volatility model (see Section 3.2), to show that the MLMC method can, with some work, be adapted into the MPR framework. We chose the stochastic volatility model since it is for this that we require a numerical method to approximate solutions of $X(t)$, and the plans under the GBM already have an exact solution. We also chose to use the higher order Milstein method (see Equation (3.22) in Section 3.2.3) in our calculations below, although our implementation in the *P3-cir-mlmc-l.py* file allows a choice between the Euler-Maruyama and Milstein methods.

We note that in Giles' code, the Brownian paths at the coarsest level consist of one step (from time 0 to time T). This formulation becomes a problem for path-dependent plans which require a specified minimum number of points to sample from, which is the case for all three plans. To deal with this, we have set the coarsest level under the P3 implementation to have the minimum number of paths required. In particular, at level l, we have $L = 5(2^l)$ steps. So at level 0, we have 5 steps, corresponding to the closing price of the Index at the end of each year.

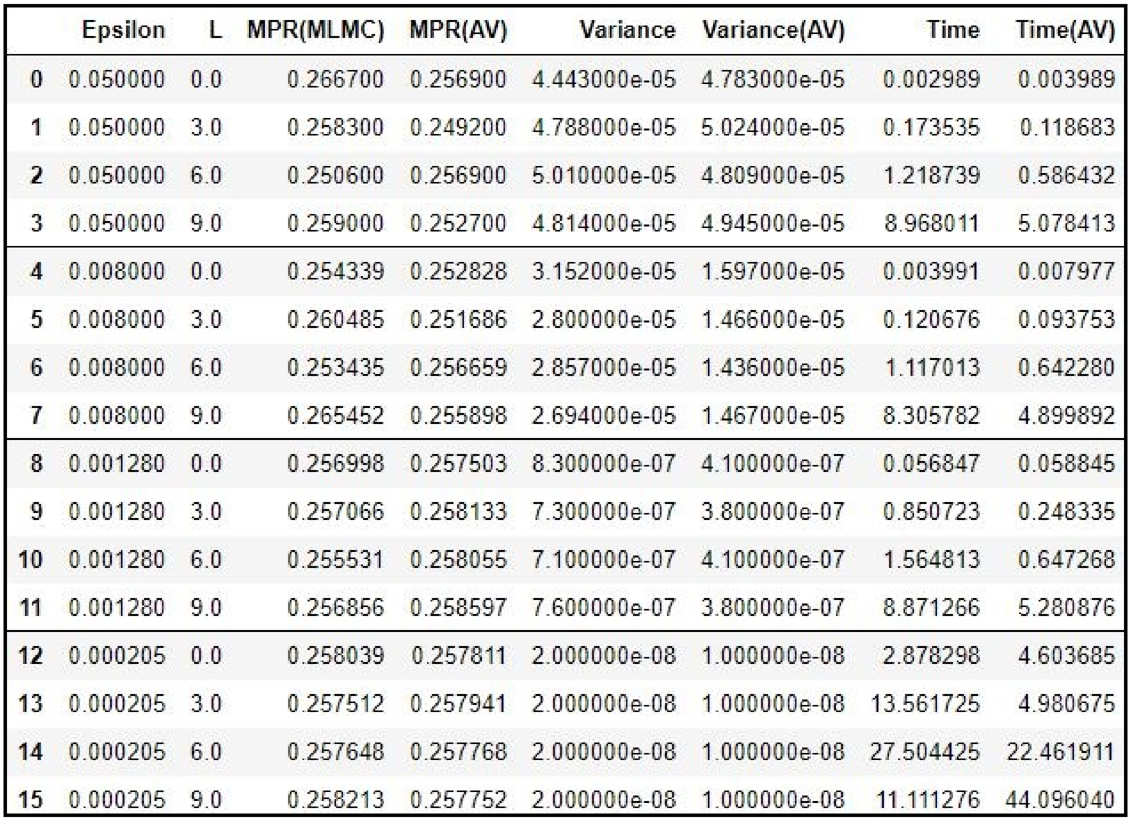
It would be very hard to objectively estimate the reduction of variance under the multilevel method given the unique way that it calculates the MPR. We cannot make a simple adjustment as we did with the antithetic variate implementations in order to make valid comparisons. However, we would like to refer the reader to [6] to see Higham's analysis of the improved accuracy under MLMC.

We note however that it is relatively easy to combine the antithetic variate technique with the multilevel method. This is because the algorithm that calculates the estimator $\hat{Y}_l$ is itself a self-contained function and can be run independently of the algorithm. This means that we would only need to modify this function to include antithetic variates.

Similar to our analysis of antithetic variates in Section 3.1.2, we have designed our level l estimator to have half of its samples generated in the usual manner and the other half as their antithetic variates. Figure 4 shows the results of running our MLMC implementations with and without antithetic variates. We used the initial number of samples $N_0 = 500$, fixed all the parameters of our model, and varied the accuracy requirement ϵ and total number of levels L. The model parameters were set so that the volatility remains relatively stable, only with very small perturbations, i.e. $\gamma = 0.01$, $\mu_2 = 1$, $\sigma_0 = \lambda$ (see Section 3.2.4 for more details on the role of the parameters). The MPR result using MLMC should be similar to the MPR result from the P3 analytic solution (but definitely not the same due to a slightly varying volatility under the MLMC), i.e. MPR = 0.256451.

We make the following observations regarding the output:

1. Higher accuracy requirements (lower values of ϵ) result in a lower variance of the estimate.
2. It seems that decreasing ϵ reduces the effect of adding extra levels. Increasing the number of



	Epsilon	L	MPR(MLMC)	MPR(AV)	Variance	Variance(AV)	Time	Time(AV)
0	0.050000	0.0	0.266700	0.256900	4.443000e-05	4.783000e-05	0.002989	0.003989
1	0.050000	3.0	0.258300	0.249200	4.788000e-05	5.024000e-05	0.173535	0.118683
2	0.050000	6.0	0.250600	0.256900	5.010000e-05	4.809000e-05	1.218739	0.586432
3	0.050000	9.0	0.259000	0.252700	4.814000e-05	4.945000e-05	8.968011	5.078413
4	0.008000	0.0	0.254339	0.252828	3.152000e-05	1.597000e-05	0.003991	0.007977
5	0.008000	3.0	0.260485	0.251686	2.800000e-05	1.466000e-05	0.120676	0.093753
6	0.008000	6.0	0.253435	0.256659	2.857000e-05	1.436000e-05	1.117013	0.642280
7	0.008000	9.0	0.265452	0.255898	2.694000e-05	1.467000e-05	8.305782	4.899892
8	0.001280	0.0	0.256998	0.257503	8.300000e-07	4.100000e-07	0.056847	0.058845
9	0.001280	3.0	0.257066	0.258133	7.300000e-07	3.800000e-07	0.850723	0.248335
10	0.001280	6.0	0.255531	0.258055	7.100000e-07	4.100000e-07	1.564813	0.647268
11	0.001280	9.0	0.256856	0.258597	7.600000e-07	3.800000e-07	8.871266	5.280876
12	0.000205	0.0	0.258039	0.257811	2.000000e-08	1.000000e-08	2.878298	4.603685
13	0.000205	3.0	0.257512	0.257941	2.000000e-08	1.000000e-08	13.561725	4.980675
14	0.000205	6.0	0.257648	0.257768	2.000000e-08	1.000000e-08	27.504425	22.461911
15	0.000205	9.0	0.258213	0.257752	2.000000e-08	1.000000e-08	11.111276	44.096040

Figure 4: Multilevel Monte Carlo with and without antithetic variates

levels under each ϵ has less of an effect on the MPR estimate. For example, the differences between MPR estimates for $\epsilon = 0.05$ are bigger than those between MPR estimates for $\epsilon = 0.000205$.

Upon inspection, we can see that the MPR values for $\epsilon = 0.05$ are almost the same, up to 2 decimal places, while the MPR values for $\epsilon = 0.000205$ are the same for up to 3 decimal places. Analysis of the variance of the four 'MPR(MLMC)' estimates and the four 'MPR(AV)' estimates under each value of ϵ confirm this.

3. Increasing ϵ and increasing L both have the same effect of increasing the computation time. This result is as expected.

   Recall that the MLMC method determines the optimal number of samples $\hat{N}_l$, where N_l is a function of ϵ^{-2} (3.12). [8, 9]. So decreasing ϵ by a factor of 10 will increase $\hat{N}_l$ by a factor of 100 for $l = 0, 1, ..., L$.

   Recall also that at higher levels the Brownian paths are approximated with smaller step sizes (in particular, with step size $\Delta t = 5(2^l)$). Increasing L by 1 will mean that the algorithm will have to do at least N_0 extra sets of calculations for the initial sampling, and because the calculations are being done at higher resolution, this means that the minimum marginal cost from increasing L, i.e. the extra cost for each new level added, also increases.

4. Overall, the effect of using antithetic variates is similar to what we saw in Section 3.1.2. That is, the variance is reduced, and the corresponding calculation time is smaller. The difference is that the reduction in computation time is greater, and the variance reduction is not as large. Recall that the optimal number of samples $\hat{N}_l$ is determined by the variance of the estimates. If they are smaller due to antithetic variates, then we would require fewer extra samples to calculate, hence reduction in computation time is more apparent.

   We note however that there does not seem to be a variance reduction for $L = 0$ where it takes the AV implementation longer to calculate the samples, even though the variance is reduced for all ϵ except for $\epsilon = 0.05$. Our suspicion is that this has to do with the way that the MLMC method recursively calculates $\hat{N}_l$. The final $\hat{N}_0$ value might be sensitive to a small number of initial samples N_0.

5. The last result, i.e. $(\epsilon, L) = (0.000205, 9)$, is suspicious. It seems that using a high L with low ϵ is causing the implementation to work improperly. However, the 'MPR(MLMC)' and 'MPR(AV)' values do not seem suspicious.

3.1.4 Conclusion

We are still a long way from being able to properly incorporate the MLMC method into our MPR framework. Although we have created a working implementation for P3 (and given more time, we could do so for P4 and P5), there are still some questionable results that we would like to delve deeper into and analyse. Nevertheless, the MLMC shows great potential, and a substantial amount of research is still being done since this is a relatively new technique.

3.2 Stochastic Volatility

3.2.1 Motivation

Recall from Section 2.2 that the GBM has the SDE

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t), \quad t \geq 0. \tag{2.2}$$

This model assumes that the diffusion coefficient σ is constant. This makes the mathematics much easier to handle and we have the well-known analytic solution (2.3). However, as we mentioned in passing in the Introduction (Section 1), this assumption of constant market volatility does not hold true, and even if it did, it would only be for a very short period. In general, volatility in the markets is not constant. [13]

There are also times when volatility increases significantly over a short period and then slowly stabilises over a longer period. This is usually the case when important events or financial news is made public, and with the speed of modern telecommunications, the effect on the stock prices is almost immediate but quite exaggerated as investors must quickly react but cannot always do so rationally. Hence, we tend to see the volatility reverting to a more stable level over time.

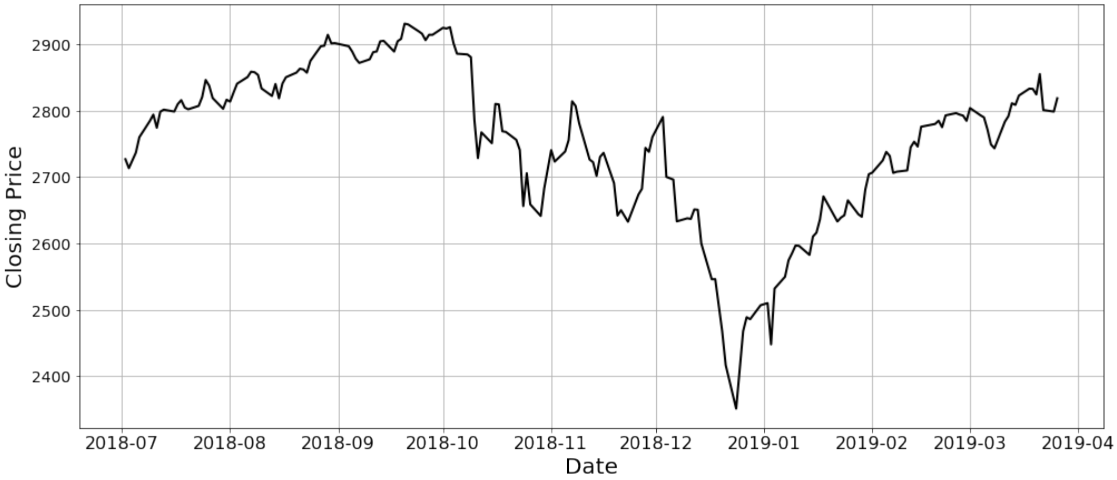


Figure 5: S&P 500 Index: June 2018 - March 2019

Figure 5 shows the historical price data for the S&P 500 Index between 1st June 2018 and 15th March 2019. If we were to divide the data into 3 sections, we can see different levels of volatility in each period. In the first section, the volatility is relatively stable (between June and October 2018). We then see a large decline in the level of the index followed by large volatility in the second section (between October 2018 and January 2019). This was driven mainly by fears of an economic slowdown in the market. Finally, we see that the volatility stabilises to a similar level as in the first section (between January and March 2019).

If we were to zoom in on the historical data and began looking at the hourly price data, we would see more of these shifts in volatility, each of which can usually be attributed to a series of news reports or release of key economic data.

Therefore, it would be more appropriate to include a stochastic volatility element in our model. In particular, we would like a mean-reverting model to reflect the fact that volatility tends to return to a set level (especially after periods of very high volatility). It is with this in mind that we have chosen the Cox-Ingersoll-Ross (CIR) process.

3.2.2 The CIR Model

Recall that the CIR model, also known as the mean reverting square root process[1], is given by the linear SDE

$$dX(t) = \mu(\lambda - X(t))dt + \sigma\sqrt{X(t)}dW(t)), \quad t \geq 0. \tag{2.4}$$

We are going to assume that the Index follows the stochastic volatility model

$$dX(t) = \mu_1 X(t)dt + \sigma(t)X(t)\big[\rho dW_1(t) + (1-\rho)dW_2(t)\big] \tag{3.13}$$

$$d\sigma(t) = \mu_2\big(\lambda - \sigma(t)\big)dt + \gamma\sqrt{\sigma(t)}dW_2(t), \tag{3.14}$$

where $X(t)$ is the Index level, $\sigma(t)$ is the volatility of the process, $W_1(t)$ and $W_2(t)$ are independent noise (Wiener) processes, and μ_1, μ_2, γ, λ and ρ are constants, with $\mu_2, \gamma \geq 0$, $\lambda > 0$ and $\rho \in [0, 1]$.

The two main differences between the GBM model and our new model are that the diffusion $\sigma(t)$ is no longer constant and that $X(t)$ is now influenced by a linear combination of two independent Wiener processes. The parameter ρ allows us to set a correlation between the volatility perturbing the volatility and the asset (see Section 3.2.4). The term $\big[\rho dW_1(t) + (1-\rho)dW_2(t)\big]$ is not a Wiener process given that

$$\begin{aligned}\mathbb{V}\big[\rho dW_1(t) + (1-\rho)dW_2(t)\big] &= \rho^2\mathbb{V}\big[W_1\big] + (1-\rho)^2\mathbb{V}\big[W_2\big] \\ &= \big[2\rho(\rho-1)+1\big]dt \\ &\neq dt, \end{aligned} \tag{3.15}$$

for $\rho \in (0, 1)$, but note that it can easily be re-scaled to be a Wiener process by using a multiplying factor $\big(2\rho(\rho-1)+1\big)^{-1/2}$. The result (3.15) will be relevant for our interpretation of the effect of varying ρ on the numerical distribution of $X(T)$ (see Section 3.2.4). Rather than attempting to derive an analytic solution for this SDE, we instead make use of the Euler-Maruyama (EM) method to obtain approximate numerical solutions of the trajectories.

3.2.3 Numerical Discretisation Techniques

We will define the two techniques that we have used in this project to calculate approximate solutions of the SDEs.

Euler-Maruyama Method

It is usual to write a scalar, autonomous SDE in differential equation form as

$$dX(t) = f(X(t))dt + g(X(t))dW(t), \quad t \geq 0, \tag{3.16}$$

with scalar functions f and g and initial condition $X(0) = X_0$. However, (3.16) can also be written in integral form as

$$X(t) = X_0 + \int_0^t f(X(s))ds + \int_0^t g(X(s))dW(s), \quad t \geq 0. \tag{3.17}$$

To apply a numerical method to (3.16), we first discretise the interval of interest, say $[0, T]$, where T is the maturity date. We let the step size $\Delta t = T/L$ for some positive integer L and $\tau_j = j\Delta t$. Our numerical approximation to $X(\tau_j)$ will be denoted X_j. Then the EM method takes the form

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})\Delta W_j, \quad j = 1, 2, ..., L, \tag{3.18}$$

where $\Delta W_j = W(\tau_j) - W(\tau_{j-1})$, and we generate the increments ΔW_j using a normal random sample with variance $\tau_j - \tau_{j-1}$. To help understand where (3.18) comes from, notice from the integral form (3.17) that

$$X(\tau_j) = X(\tau_{j-1}) + \int_{\tau_{j-1}}^{\tau_j} f(X(s))ds + \int_{\tau_{j-1}}^{\tau_j} g(X(s))dW(s). \tag{3.19}$$

Each of the three terms on the right-hand side of (3.18) approximates the corresponding term on the right-hand side of (3.19). We can then consider $X(t)$ as the random variable that arises when we take the limit $\Delta t \to 0$ for our numerical approximation. Note also that in the deterministic case, i.e. when $g \equiv 0$ and X_0 constant, (3.18) reduces to Euler's method. So the EM method can be thought of as the generalisation for SDEs of Euler's method for ODEs.

The specific form of our EM method applied simultaneously to (3.13) and (3.14) is

$$X_j = X_{j-1} + \mu_1 X_{j-1}\Delta t + \sigma_{j-1} X_{j-1}\big[\rho\Delta W_{1,j} + (1-\rho)\Delta W_{2,j}\big], \tag{3.20}$$

$$\sigma_j = \sigma_{j-1} + \mu_2\big(\lambda - \sigma_{j-1}\big)\Delta t + \gamma\sqrt{|\sigma_{j-1}|}\Delta W_{2,j}, \quad j = 1, 2, ..., L, \tag{3.21}$$

where $\Delta W_{i,j} = (W_i(\tau_j) - W_i(\tau_{j-1})), \quad i = 1, 2.$

Milstein's Higher Order Method[2]

We note that the Euler-Maruyama method has strong order of convergence $\gamma = 0.5$, whereas the underlying deterministic Euler method converges with classical order 1.[2] However, it is possible to raise the strong order of the method to 1 by adding a correction to the stochastic increment, giving us Milstein's method. This correction comes from modifying the Taylor expansion in the case of Itô calculus. The form of the Milstein Method applied to (3.16) is given by

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})\Delta W_j + \tfrac{1}{2} g(X_{j-1}) g'(X_{j-1})\big((\Delta W_j)^2 - \Delta t\big), \tag{3.22}$$

where $\Delta W_j = W(\tau_j) - W(\tau_{j-1})$.

3.2.4 Varying the Model Parameters

We have already mentioned that in estimating the MPR, we are interested in the sensitivity of this value to different market conditions. We do this by varying the parameters in our underlying model and observing the changes to the MPR estimate.

For example, in our GBM model (2.2), we would like to know what value of the drift coefficient μ will give us an MPR equal to the risk-free rate (at which point one would instead invest in the risk-free asset). If the μ drift has to be very low before this happens, then the investor might decide to invest in this savings plan. In terms of the market, this means that it would take a strong bearish (downward) trend in the price level of the Index to make the savings plan unappealing. Similarly, we would like to know how the MPR changes for different values of the diffusion coefficient σ, and if our 'break even' value is quite large/small and is therefore unlikely to happen, then the plan might be worth investing in.

Our underlying model given by (3.13) and (3.14) is more complicated, so the exact effect of each parameter on the model might not be immediately obvious, so we will now look at these. Note that even though we have created implementations of the CIR volatility models for the three plans, we will limit our analysis to the effects of the parameters on the final value of the process and not on the MPR.

We will be looking at numerical distributions of the final values of our approximation, denoted by X_T, and we will also consider final values of the volatility σ_T where relevant, when varying one parameter and fixing all other parameters. For comparison, we have tried to include the numerical distribution of an equivalent GBM model, where $\mu = \mu_1$, and we set λ and our initial value σ_0 in (3.14) equal to the value of σ in (2.2). We will then discuss the results and implications of each one for our estimation of the MPR under varying market conditions.

We will mention a few things before moving on to the analysis:

- Setting $\mu_2 = \gamma = 0$ and $\rho = 0$ or $\rho = 1$ reduces to the GBM model (see Figure 6).
- To help the reader, we have included the expressions of the Euler-Maruyama form for (3.13) and (3.14) under each subheading.
- We will be comparing the relative differences in shape and position between the density plots, so the scale values are of no interest to us.
- The final values differ from one plot to the next because the parameters that are fixed were changed each time to maintain the necessary condition $2\mu_2\lambda \geq \gamma^2$ to avoid zero values in the CIR model with probability 1 [3].

Varying ρ (with deterministic volatility)

$$X_j = X_{j-1} + \mu_1 X_{j-1}\Delta t + \sigma_{j-1}X_{j-1}\left[\rho\Delta W_{1,j} + (1-\rho)\Delta W_{2,j}\right], \tag{3.13}$$

$$\sigma_j = \sigma_{j-1} + \mu_2\left(\lambda - \sigma_{j-1}\right)\Delta t + \gamma\sqrt{|\sigma_{j-1}|}\Delta W_{2,j} \tag{3.14}$$

where $\Delta W_{i,j} = (W_i(\tau_j) - W_i(\tau_{j-1}))$, $i = 1, 2$.

Since ρ appears only in X_j, we are only interested in the changes to X_T. We have also set $\mu_2 = \gamma = 0$ so that the volatility remains constant.

At a glance, we can see that ρ is like a 'correlation' parameter that determines the extent to which the asset and volatility processes are influenced by the same random effects. The rationale behind including the W_2 term into (3.13) is that we expect events affecting the volatility of the Index to also directly affect the price, though not necessarily with the same magnitude. We allow ρ to vary between 0 and 1, i.e. $\rho \in [0, 1]$.

Figure 6 shows the density plot of X_T for varying values of ρ. As we have mentioned, the model given by (3.13) and (3.14) reduces to the GBM model when the volatility is deterministic and $\rho = 0$ or $\rho = 1$, which is why we can see the same distribution as the GBM model for these two values of ρ. When we look at the numerical distributions for $\rho = 0.25$ and $\rho = 0.75$, it seems that all the final values in the simulation are being shifted to the right, and again for $\rho = 0.5$, with a marked reduction in the standard deviation of the final values. However, further analysis shows that the maximum values (values on the end of the right tail of the density plots) are being shifted to the left.

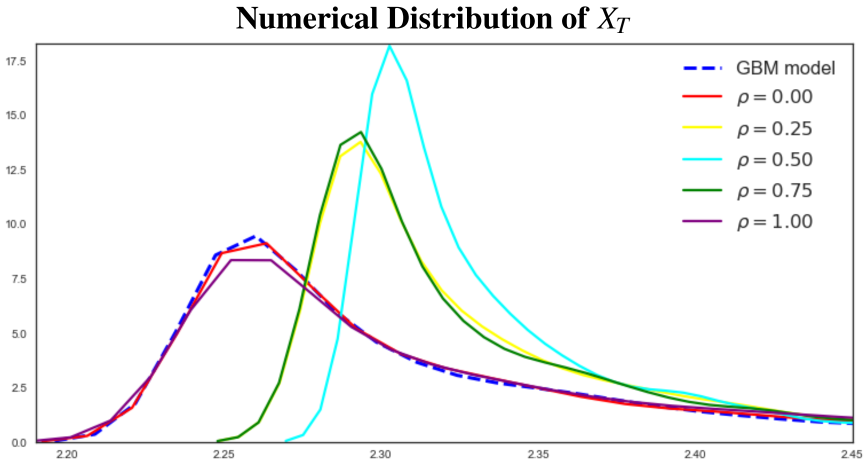


Figure 6: Varying ρ

So, what we are actually seeing is the result of a decrease in the standard deviation of the final values as we get closer to $\rho = 0.5$. This can be easily verified in theory:

$$W_1 \sim N(0, \delta t) \text{ and } W_2 \sim N(0, \delta t) \implies 0.5W_1 + 0.5W_2 \sim N(0, 0.5\delta t). \tag{3.23}$$

We note that because the variance of the final values is smaller, this does not mean our estimation of the MPR becomes more precise as $\rho \to 0.5$. Rather, it simply changes the model to one with a weaker noise process than our original W_t in the GBM model.

Varying γ

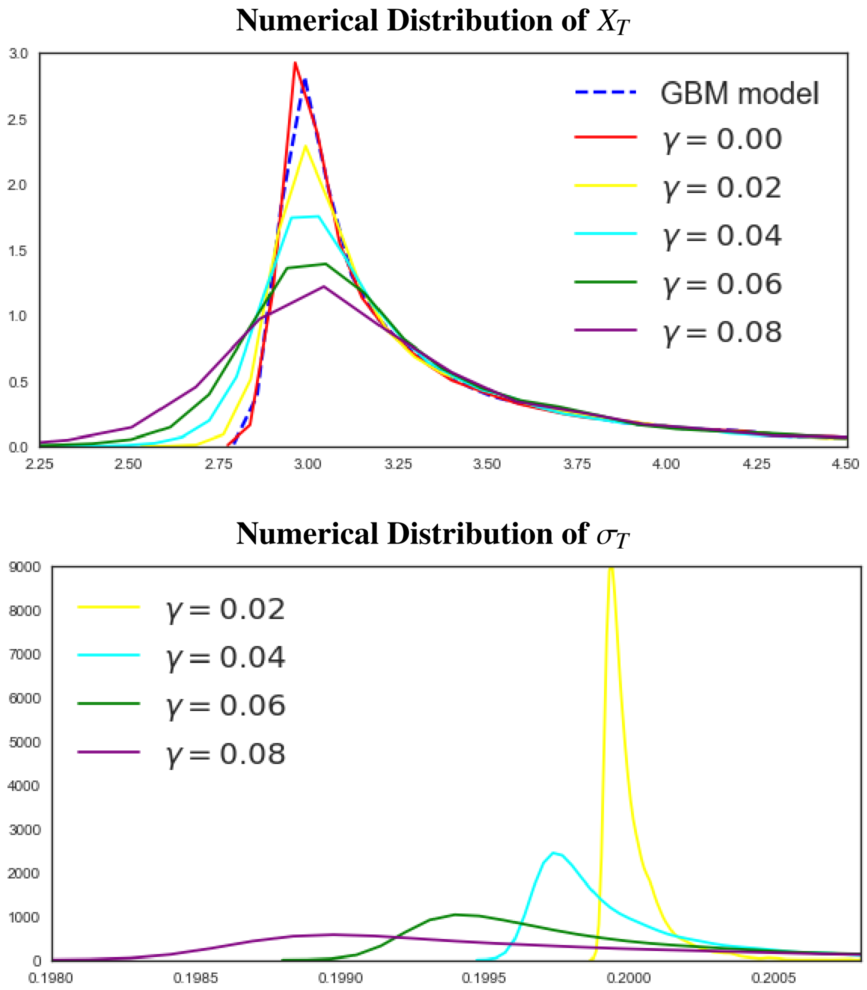
$$X_j = X_{j-1} + \mu_1 X_{j-1}\Delta t + \sigma_{j-1}X_{j-1}\left[\rho\Delta W_{1,j} + (1-\rho)\Delta W_{2,j}\right], \tag{3.13}$$

$$\sigma_j = \sigma_{j-1} + \mu_2(\lambda - \sigma_{j-1})\Delta t + \gamma\sqrt{|\sigma_{j-1}|}\Delta W_{2,j} \tag{3.14}$$

where $\Delta W_{i,j} = (W_i(\tau_j) - W_i(\tau_{j-1}))$, $i = 1, 2$.

The parameter γ is the diffusion coefficient in our volatility model, and given that it appears in our expression for σ_j, we must now consider the numerical distribution of both X_T and σ_T for varying values of γ. We would expect the effects of varying γ to be more evident in the density plot of σ_T given that γ is, in essence, the 'noise of the noise' of X_T, so its effects on X_T are implicit.

Figure 7 shows the density plot of X_T and σ_T for varying values of γ. Looking at the density plot of σ_T (with mean $\lambda = 0.2$), we can see that the range of values is very narrow for small γ. Note that if


Numerical Distribution of X_T

Numerical Distribution of σ_T

Figure 7: Varying γ

$\gamma = 0$ and $\lambda = \sigma_0 = 0.2$ in (3.14), then we would have a vertical line (since the volatility would remain constant). As we increase γ, the distribution of σ_T ‘flattens out’. This makes sense given that we are increasing the noise coefficient of the volatility model.

Looking then at the density plot of X_T, we see that the position of the peak remains relatively unchanged. The $\gamma = 0$ plot reduces to the GBM model. Then as we increase γ, we can observe an increase in the standard deviation of X_T. Our understanding of this is result is that the changes affect only the volatility, and not the drift, of X_T, and the effect is to increase the magnitude of the noise, not the direction, of the process. We expect the volatility to drive the price up and down with probability 0.5 in both directions. This is why the peak stays in the same position. Note that we did not use negative values of γ, since the noise terms can be either positive or negative, with mean 0, so the effect of $-\gamma$ should be the same as that of γ.

In practical terms, using a greater value of γ would mean that we expect a greater likelihood of high-volatility events to happen during the period.

Varying μ_1 (with deterministic volatility)

$$X_j = X_{j-1} + \mu_1 X_{j-1}\Delta t + \sigma_{j-1}X_{j-1}\left[\rho\Delta W_{1,j} + (1-\rho)\Delta W_{2,j}\right], \tag{3.13}$$

$$\sigma_j = \sigma_{j-1} + \mu_2(\lambda - \sigma_{j-1})\Delta t + \gamma\sqrt{|\sigma_{j-1}|}\Delta W_{2,j} \tag{3.14}$$

where $\Delta W_{i,j} = (W_i(\tau_j) - W_i(\tau_{j-1}))$, $i = 1, 2$.

We expect the effects of μ_1 in the CIR volatility model to be the same as that of μ in the GBM model, which is to shift the distribution to the right for increasing μ_1 and to the left for decreasing μ_1. Figure 8 shows the density plot of X_T for varying values of μ_1. Again, because μ_1 only appears in X_j, we are not interested in the density plot of σ_T.

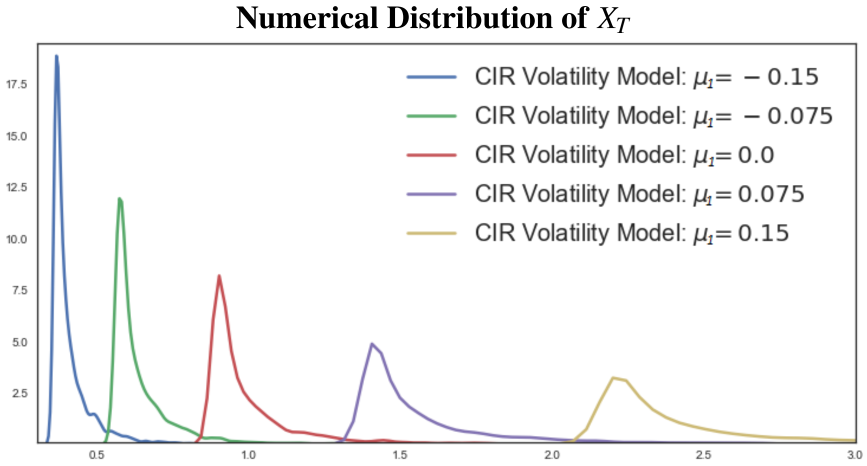


Figure 8: Varying μ_1

As expected, the density plot is shifted right for increasing μ_1. However, notice that the peaks are smaller for increasing μ_1, meaning that the volatility of X_T is also increasing. This is simply because of the scales that we are using. We can intuitively think of the drift term μ_1 as the rate of return applied to the asset over a very small period. The increase/decrease applied is compounding, not additive. For example, a 10% return on €100 would give €10 income, while the same rate of return on €10,000 would give €1,000; the values are different even though the rate of return remains the same. Hence, even though μ_1 remains constant over the period, we see a greater spread in the possible values of X_T.

In practical terms, increasing μ_1 means we expect a more positive outlook for the Index level over the period, and vice versa. The increased standard deviation result that we obtained for X_T should not cause a problem since investors would tend to look at changes in the value of their portfolios in terms of their percentage returns.

Varying λ

$$X_j = X_{j-1} + \mu_1 X_{j-1}\Delta t + \sigma_{j-1}X_{j-1}\left[\rho\Delta W_{1,j} + (1-\rho)\Delta W_{2,j}\right], \tag{3.13}$$

$$\sigma_j = \sigma_{j-1} + \mu_2\big(\lambda - \sigma_{j-1}\big)\Delta t + \gamma\sqrt{|\sigma_{j-1}|}\Delta W_{2,j} \tag{3.14}$$

where $\Delta W_{i,j} = (W_i(\tau_j) - W_i(\tau_{j-1}))$, $i = 1, 2$.

The parameter λ is the long term mean level of the volatility model. All future trajectories of σ_j will evolve around this level in the long run. Figure 9 shows the density plot of X_T and σ_T for varying values of λ. Here, we set σ in the GBM model (2.2) equal to λ in the CIR volatility model (3.14).

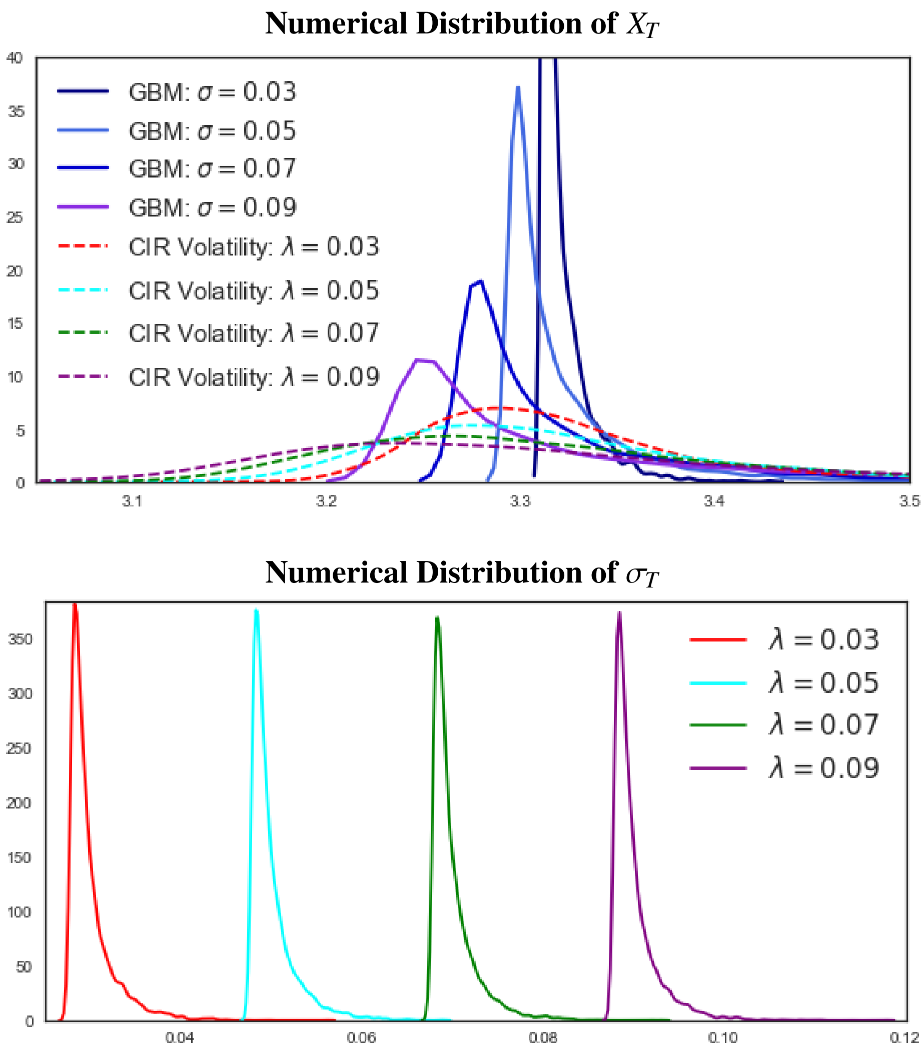
Looking first at the density plot of X_T, we observe two things about both models: as we increase σ and λ, we see that the standard deviation of the values increases and that the range of values seems to shift a little to the left. The first observation is quite obvious given that we are varying the diffusion coefficient, which is the main factor determining the overall volatility of the model. The second observation makes sense if we consider the expected value of the analytic solution for the GBM model. Recall that the GBM model (2.2) has the solution

$$X(t) = X_0\exp\big((\mu - 0.5\sigma^2)t + \sigma W(t)\big). \tag{2.3}$$

Note that $\log\Big(\frac{X(t)}{X_0}\Big) \sim N\big((\mu - 0.5\sigma^2)t, \sigma t\big)$. This means that increasing σ will also reduce the mean. We can expect a similar effect when we vary λ since the SDEs for X_T under the two models have the same form.

Comparing the distribution of X_T between the two models for $\lambda = \sigma$, we found that the difference in mean remains relatively the same, which means that varying λ and σ by the same amount has the same effect in shifting the mean of X_T under the GBM model and the CIR volatility model.

Finally, it seems the standard deviation of X_T is greater for our stochastic volatility model than for the constant volatility (GBM) model. We can see that for each plot under the GBM model, the


Numerical Distribution of X_T

Numerical Distribution of σ_T

Figure 9: Varying λ

corresponding CIR volatility plot is significantly ‘flatter’.

This parameter is very important in our interpretation of the dynamics of market prices. As we mentioned before (Section 3.2.1), high-volatility events tend to be followed by a stabilisation to a lower volatility level. This could mean a ‘reversion’ to some long-term volatility level. In fact, we can even see this in our plot of the S&P 500 Index in Figure 5 where the Index seems to have high volatility in the middle section but eventually returns to a similar volatility level as it did before entering into the period of high volatility.

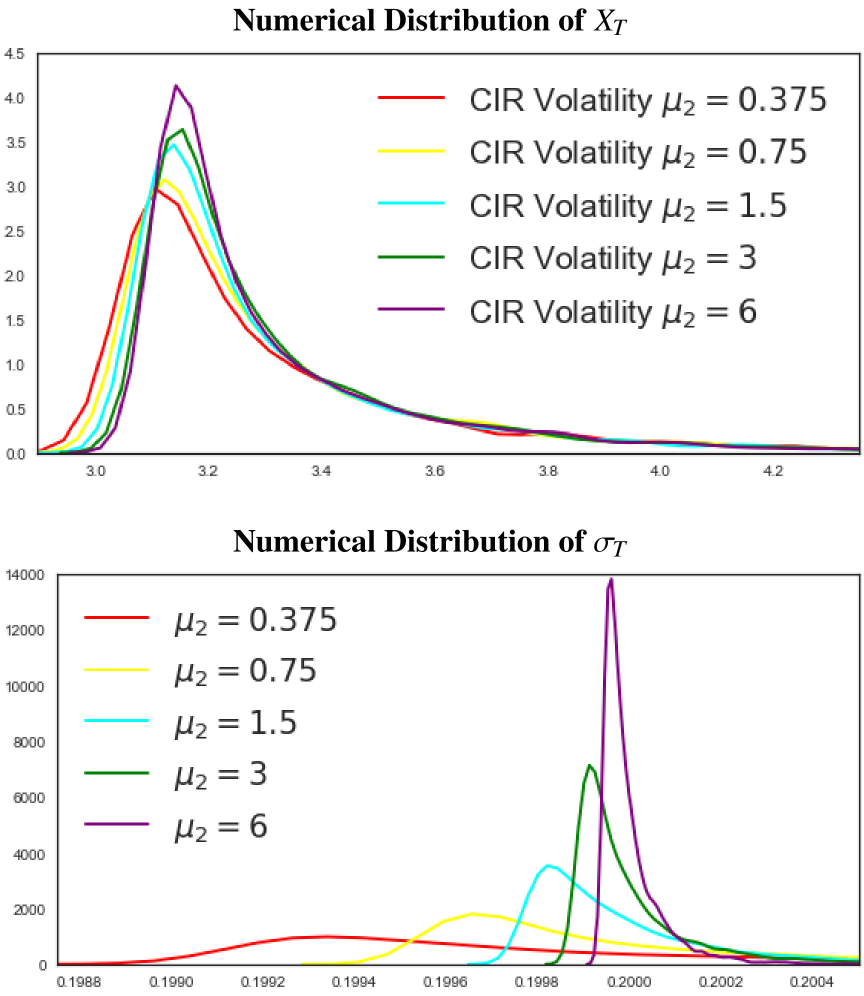
Varying μ_2

$$X_j = X_{j-1} + \mu_1 X_{j-1}\Delta t + \sigma_{j-1}X_{j-1}\left[\rho\Delta W_{1,j} + (1-\rho)\Delta W_{2,j}\right], \tag{3.13}$$

$$\sigma_j = \sigma_{j-1} + \mu_2(\lambda - \sigma_{j-1})\Delta t + \gamma\sqrt{|\sigma_{j-1}|}\Delta W_{2,j} \tag{3.14}$$

where $\Delta W_{i,j} = (W_i(\tau_j) - W_i(\tau_{j-1}))$, $i = 1, 2$.

The parameter μ_2 will determine the 'strength' of the mean reversion of the volatility. For a high value of μ_2, we expect the volatility to be closer to the mean volatility level λ throughout the period, while on the other hand, we expect the volatility to be more spread out for lower values of μ_2. Figure 10 shows the density plot for X_T and σ_T for varying values of μ_2.



Numerical Distribution of X_T

Numerical Distribution of σ_T

Figure 10: Varying μ_2

The results are as expected. Looking at the density plot for σ_T, we can see that the distribution of σ_T becomes narrower for increasing μ_2, signifying a reduction in the standard deviation of σ_T. Just like with γ, the effect of varying μ_2 on the distribution of X_T is implicit, so the effects are not as pronounced as they are for σ_T. Note also that increasing μ_2 has a similar effect as decreasing γ, i.e. the peaks are sharper but remain relatively in the same position, implying a reduction in the magnitude of the volatility (but not the direction).

In terms of the market price dynamics, μ_2 can be understood as the rate at which the market can return to stability after experiencing a shift in volatility from the long term mean level. Choosing a high value of μ_2 would mean that one is assuming a strongly efficient market (under the efficient market hypothesis[14, 15]).

3.2.5 Conclusion

Keeping in mind the effects of each parameter on the price dynamics of the Index, one can now move on to estimating the MPR of each of the three plans under the assumption of a CIR volatility model. And unlike the GBM model, we can simulate a huge variety of different market conditions, many of which have been empirically verified and agreed upon by general consensus, e.g. non-constant and mean-reverting price volatility.

3.3 Efficient Implementation

3.3.1 Motivation

We have already mentioned that Monte Carlo methods can be quite computationally expensive especially when the sample paths are generated via discretisation methods. In fact, speeding up the computation time of the implementations was motivated by the CIR Volatility models where, for a large enough number of samples that would provide a meaningful result, the code would take at least 30 minutes to run.

In a 2001 paper, Higham mentions the use of vectorisation to make his implementations run quickly, “less than 10 minutes on a modern desktop machine..."[2], and it is from this that we have designed ‘vectorised’ versions of all the implementations (for antithetic variates and CIR Volatility).

3.3.2 Vectorisation

Vectorisation is a form of parallel programming called single instruction, multiple data (SIMD). It is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time. In the context of our problem, this means that rather than using `for` loops and generating each path at each iteration of the loop, we instead generate all our paths at the same time using vectors/arrays. This can be done quite easily using the well-known *numpy* library for Python.

When comparing the improvement in the computation time due to vectorisation, we found that the result was positive for all the implementations, although we are more interested in the results for the CIR volatility models since these are the implementations that were ‘slow’ to begin with. The results of the analysis are given by Figure 11.

Plan	Computation Time Reduction Factor
P3	82.245
P4	6.57
P5	1.336
P3 (CIR Volatility)	32.99
P4 (CIR Volatility)	32.37
P5 (CIR Volatility)	31.84

Figure 11: Results of using vectorisation

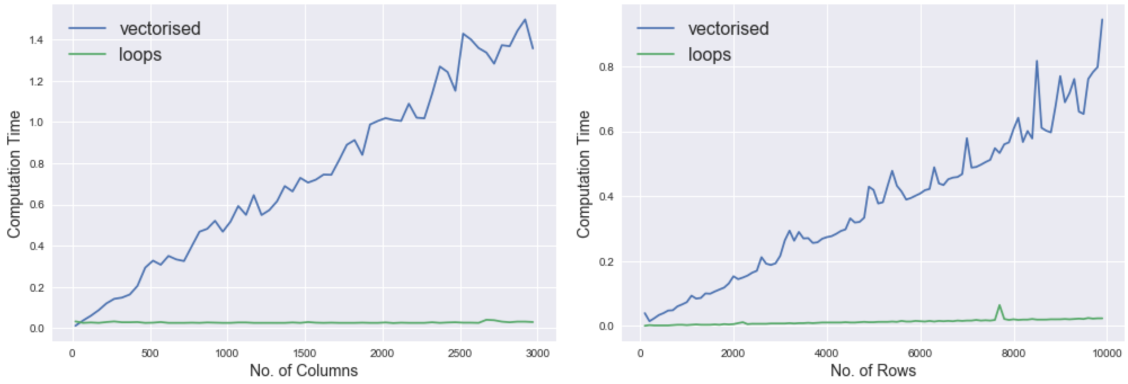
Little remains to be said about the massive improvement in the computation time of the code. Where any one of the CIR Volatility implementations would originally take 60 minutes to generate a meaningful result, we now only have to wait less than 2 minutes.

However, before moving on, we must bring the reader’s attention to the drawbacks and limitations of using vectorisation:

- **Speed at the cost of memory allocation:** The increased computation speed achieved from vectorisation is achieved mainly by employing more of the computer’s memory capacity at a time. This becomes a problem when we try to generate too many paths in one run of the function, especially when the paths are generated via discretisation. For example, if we wished to generate one million paths, with each path containing 300 steps, this would involve creating a *numpy* array with one million rows and 300 columns, i.e. 300 million values, not to mention the calculations involved in applying the payoff function and calculating the return. Computers with relatively small RAM sizes will likely experience slow performance or will crash due to the computer swapping out data to the hard drive.

- **Extra layers of vectorisation are not as effective:** Looking at the table above, it can be seen that vectorisation is most effective for P3, less for P4, and much less so for P5. We found that this is because there was already a layer of vectorisation in the original implementations. Recall that the total number of Standard Normal samples generated for each trajectory for P3, P4 and P5 are 3, 30 and 304, respectively. These samples were already being obtained via vectorisation, i.e. generating the samples at the same time using the *numpy.random.normal* function. We applied the vectorisation method to our implementations, not knowing this.

  Thus, vectorisation via two-dimensional arrays is not twice as effective as one-dimensional vectorisation, and the second layer becomes less effective as the dimensions of the array increase. Figure 12 shows the plot of computation time when generating a given number of normal random variables. Plot 12a gives the computation time for a varying number of columns (given a fixed number of rows), while Plot 12b gives the computation time for a varying number of rows (given a fixed number of columns). The 'vectorised' plots use two-dimensional arrays to generate the samples, while the 'loops' plots use a combination of a `for` loop and a one-dimensional array.



(a) Varying number of columns

(b) Varying number of rows

Figure 12: Comparison of computation time between vectorisation and `for` loops

3.3.3 Conclusion

Vectorisation is a powerful tool but must be used carefully. Based on this analysis, we reworked all our implementations so that a mixture of `for` loops and arrays are used in order to maximise the efficiency obtainable from vectorisation.

4 Estimating the MPR Under Varying Market Conditions

In Section 1, we provided the context for this project. In Section 2, we gave a description of the savings plans under consideration, provided a summary of our foundational article [1] and made some comparisons between the analytic and numerical solutions for P3. In Section 3, we discussed our improvements to the MPR framework in [1], particularly in the areas of variance reduction in the resulting MC output, assuming a stochastic volatility model and improving efficiency in the implementation. We note that this was done in [1], assuming a GBM model and varying the constants μ and σ. Therefore, we will limit our analysis here to the CIR volatility model.

Recall that in Section 1, we gave a general outline of how we would go about **estimating the MPR of a savings plan under different market conditions**:

1. *We make an assumption about the stochastic process that the underlying asset will follow in the future.*
2. *We choose specific values for the parameters of the model to reflect the market conditions which we assume will hold true in the future.*
3. *We design an algorithm that simulates the underlying model a given number of times, each time implementing the payoff structure over a certain period as specified in the product description and calculating a percentage return.*
4. *We calculate the average payoff, giving us the MPR.*

We will assume the CIR volatility model in Section 3.2.2, given by

$$\begin{aligned} dX(t) &= \mu_1 X(t)dt + \sigma(t)X(t)\big[\rho dW_1(t) + (1-\rho)dW_2(t)\big] \\ d\sigma(t) &= \mu_2\big(\lambda - \sigma(t)\big)dt + \gamma\sqrt{\sigma(t)}dW_2(t), \end{aligned}$$

where $X(t)$ is the Index level, $\sigma(t)$ is the volatility of the process, $W_1(t)$ and $W_2(t)$ are independent noise (Wiener) processes, and μ_1, μ_2, γ and ρ are constants, with $\mu_2, \gamma \geq 0$, $\lambda > 0$ and $\rho \in [0, 1]$.

We will now begin looking at how ‘appealing’ each of the three plans is under different market conditions relative to each other and to the risk-free rate, which we will assume to be constant at 2.875% for the whole period (same as in [1]). For comparison, we will also include the yearly return based on the plans, since P3 has a 5-year duration and P4 and P5 mature after 6 years.

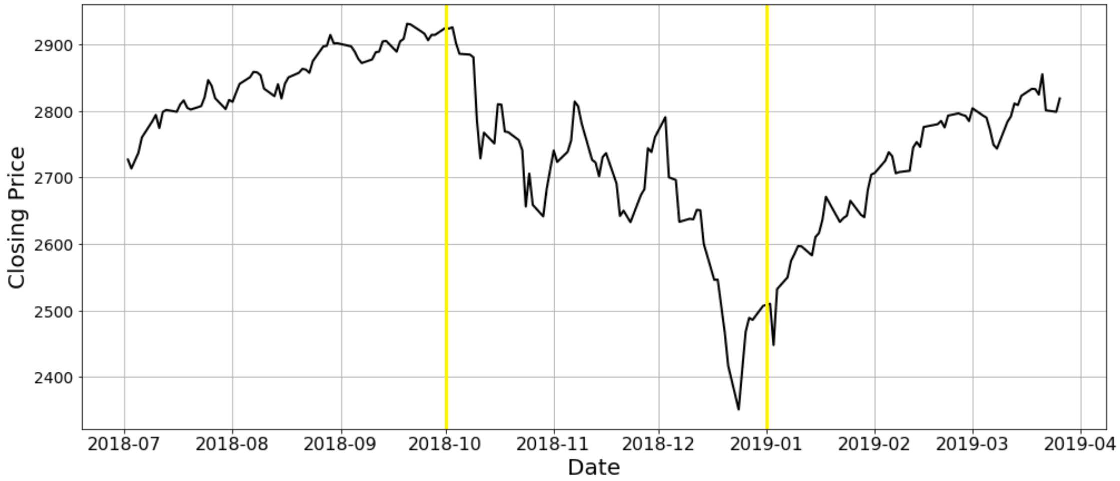
Since there are many ways that we could pick a set of market conditions, we have decided to motivate our choice by our previous plot of the S&P 500, given below. As we mentioned, the dynamics in the daily price data can be roughly divided into three sections. We will set our market conditions to be similar to that of the each section. Therefore, a description of our four market conditions is as follows:

1. A moderate bullish (upward) trend, with moderate volatility and few perturbations in the volatility (as in the first section). We expect any 'news' events to affect the volatility and Index level equally, and that the market is 'rational', so any perturbations in the volatility are 'accommodated' fairly quickly.

2. A strong bearish (downward) trend (but with the same volatility as (1). We also expect the perturbations affecting the volatility to affect the price to a greater extent than in (1). Due to a more 'fearful' market, volatility does not stabilise as quickly as in (1).

3. Same conditions as (2) but with high volatility and numerous 'strong' perturbations in the volatility.

4. Much more stable conditions, where the trend is more bullish than it was in (1) and where the volatility is slightly higher than in (1) but much lower than in (3).



S&P 500 Index: June 2018 - March 2019

Since P5 requires an initial Index level X_0, we have chosen values relative to the starting points in each section on the plot. We have also set our initial volatility σ_0 equal to λ. Our set of chosen parameters, as defined previously, is given by Figure 13.

We have three calculations under each of the four market conditions for each plan, so 12 results in total. Each MPR value was generated with a total of one million samples, with the paths approximated at 10 steps of discretisation per day. Since this kind of calculation would be too much in one run of the function, we chose to run the function with a `for` loop a set number of times, allowing us to essentially control the extent of our vectorisation. Figure 14 below shows the results of these calculations:

Market Condition	X_0	σ_0	μ_1	μ_2	ρ	λ	γ
1	4600	0.03	0.03	2.0	0.5	0.03	0.04
2	4950	0.03	-0.10	0.5	0.1	0.03	0.04
3	4950	0.15	-0.10	0.5	0.1	0.15	0.16
4	4200	0.04	0.06	1.0	0.5	0.04	0.05

Figure 13: Parameters reflecting each market condition

Market Condition	Plan	MPR	Standard Deviation	95% Confidence Interval	Total Computation Time (mm:ss)	Yearly Return
1	P3	0.34856	0.01577	[0.31765, 0.37948]	30:26	0.06163
	P4	0.32241	0	[0.32241, 0.32241]	39:02	0.04768
	P5	0.42317	0.00362	[0.41607, 0.43027]	38:18	0.06058
2	P3	0	0	[0.0, 0.0]	31:40	0
	P4	0.03209	0.00950	[0.01348,0.05071]	40:26	0.00528
	P5	-0.37726	0.01457	[-0.40581, -0.34870]	36:06	-0.07590
3	P3	0.01068	0.04186	[-0.07138, 0.09273]	28:14	0.00213
	P4	0.07180	0.05300	[-0.03207, 0.17568]	36:10	0.01162
	P5	-0.36279	0.10773	[-0.57394, -0.15164]	40:30	-0.07236
4	P3	0.35	0	[0.35, 0.35]	29:44	0.06186
	P4	0.2241	0	[0.32241,0.32241]	38:17	0.03428
	P5	0.32914	0.01286	[0.30394, 0.35435]	34:39	0.04856

Figure 14: Estimating the MPR under various market conditions

Recall that our corresponding risk-free rate of return is 0.02875. Many of the results were quite expected given our previous analysis regarding the parameters, and we will just mention a few points reconciling the results with the market interpretation.

- In the midst of bad market conditions (2 and 3), the risk-free asset would be the best choice, whereas any of the 3 plans would be better during better market conditions (1 and 4).

- Plan P3 has the highest yearly return among the three plans for the better conditions, but P4 performs better of the three. This result makes sense given the payoff structure of the plans. P3 has a higher maximum potential return (35%) versus P4 (~32.24%), but payments under P4 are made earlier, whereas P3's final payment is contingent on it never going below a certain value, which would certainly happen during a bear market.

- Unlike P3 and P4, P5 has a negative return potential, so investors would stay away from P5 if they expect unfavourable market conditions to prevail in the future.

- For plans like P3 and P4 whose returns do not explicitly depend on the initial Index level, the key factor in deciding to invest would be timing, i.e. investing right before a bull trend. On the other hand, for plans like P5, one must consider timing as well as position, since, under the plan, an investor will lose money if the plan matures with the Index level below 4,250 points.

- Note that even though P3 and P4 do not allow for negative returns, their respective confidence intervals indicate that it is because we have assumed a normal distribution in the estimate.

5 Discussion

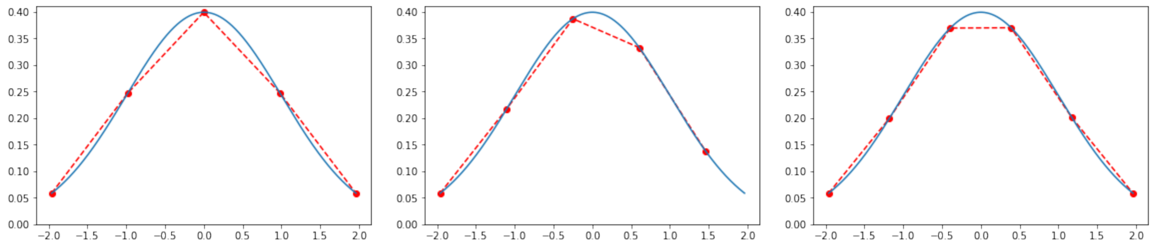
Here we document some problems encountered during the course of the project as well as some of our thoughts and findings that are worth mentioning.

5.1 Problems Encountered

5.1.1 Comparing the Convergence of the Analytic and Numerical Solution for P3

The initial result for the convergence of the analytic solution was oscillating, which prompted a further investigation. We then found that the error was in the implementation of the Riemann sums. In particular, it was because we used the *numpy.arange* function.

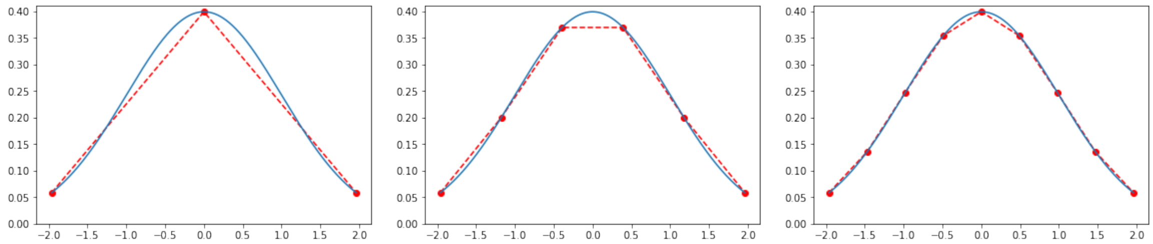
To use the *arange* function, we specify the endpoints of the sequence and the size of the steps between intermediate points. However, if the step size does not divide completely into the length of the interval, then the right endpoint of the sequence will actually be some other point along the interval. Figure 15 illustrates this problem. If we decrease the step size from $\delta t = 0.979$ (Figure 15a) to $\delta t = 0.783$ (Figure 15c), we can correctly increase the number of points on the sequence by 1. However, we run into problems when calculating the approximate integral using a step size $\delta t \in (0.783, 0.979)$. Figure 15b shows the points from Figure 15a being shifted to the left to reflect the decreasing step size. The problem lies in the fact that a new point will not be generated until we reach step size $\delta t = 0.783$, as in Figure 15c. This problem repeats as we move from $\delta t = 0.783$ to smaller step sizes. As a result, we get an oscillating trend in the convergence of the approximate integral.



(a) $\delta t = 0.979$ (b) $\delta t = 0.855$ (c) $\delta t = 0.783$

Figure 15: Using *numpy.arange* with decreasing step sizes

The issue was fixed when we replaced *arange* with the *numpy.linspace* function. This function is different in that instead of specifying the step size, we specify the number of points that we want between the endpoints. This ensures that the endpoints on the curve are fixed in place as we increase the number of intermediate points. Figure 16 illustrates how the *linspace* function solves the issue.



(a) 2 steps (b) 5 steps (c) 8 steps

Figure 16: Using *numpy.linspace* with increasing number of steps

5.1.2 Variance Reduction with Antithetic Variates for P4

Initially, the results were as expected for Plans P3 and P5. However, the result for P4 was quite mixed. Although the variance reduction was positive on average, there was a small proportion out of the 100 simulations that had no variance reduction.

To investigate further, a new implementation of P4 was created which generated many more samples over the same 6-year period, but there was still no consistent reduction of variance.

Next, we considered that it might have to do with the rule set of P4. Recall that the description of the payoff structure for P4 is as follows: *“A 6-year plan; the Index is measured at the start of the Plan, and then on a yearly basis thereafter. If the average closing level of the Index for the five business days up to and including the anniversary date is higher than 90% of the initial level, a fixed income of 5% gross will be paid for that year. If the anniversary Index level is equal to or below 90% of the Initial Index Level, no income will be paid. However, should the Index meet the required level on any future anniversary, any previously missed income payments will be added back and paid out."*

The plan description has the rule that any previously missed payments would be repaid when the required level is met on any future anniversary. Our initial suspicion was that this rule essentially created a non-uniform weighting, with greater weight of importance placed on later years, i.e. if the requirement is met at the end of Year 6, then the maximum achievable return (excluding the risk-free investment gained on previously paid income) is gained, regardless of where the price level was throughout the whole period. But once again, the problem was still unresolved. The following table shows the result of this test, which includes the number of times (out of 100) that the variance was not reduced:

Plan	Mean Variance Reduction Factor	Count of No Variance Reduction (out of 100)
P3	3.14	0
P4 (Original)	1.934	7
P4 (New)	1.653	13
P5	4.119	0

Finally, we discovered that the problem was mainly due to the drift coefficient being too high ($\mu = 0.1423$). However, this is also due to the nature of the payoff structure. Since there is a maximum achievable return under the plan, there is a point at which higher levels of the Index will not give a greater return. So, if we set the drift too high, then we will end up with simulations that give the maximum achievable return under the plan. Hence, antithetic variates are of no use because the final payoff reaches a deterministic value for a certain proportion of the simulations. This explains why there was still some variance reduction on average, and only some of the simulations had no reduction in variance. After setting the drift to a lower value ($\mu = -0.03$), we finally achieved our final result in Figure 2 (Section 3.1.2).

This result is very telling. We can conclude that the extent of the variance reduction using antithetic variates can depend on the nature of the payoff function for the plan and how it is applied to the trajectories. For example, if income payments are fixed (as in P4), then this places an upper bound on the possible income under the plan, which reduces the amount of variance reduction. Despite this potential limitation, anithetic variates are still beneficial due to the lower computational cost by creating half of the samples as antithetic variates.

5.2 Findings Regarding the Foundational Article[1]

5.2.1 Typos

There were quite a few significant typographical errors, though none that would drastically affect reproducibility of the results given, but time was lost in having to recognise and resolve some of these. In the derivation of the analytic solution for P3, some equations were typed up incorrectly, though the mistakes were not carried through to the final result. There was also a term, $\bar{x}_3$ that was understood in the context but was not directly defined. In one place, the equation reference was incorrect.

There were also some typos in the code. It would seem, however, after comparison with our corrected version, that they were just transcription errors (since the code would not work in the first place if implemented as written in the article).

Lastly, some parameters significant to the plans were not correctly initialised in the implementations, essentially changing the rule set of the plan, e.g. using a critical loss value of 4,500 instead of 4,250 as specified by Plan P5.

5.2.2 Vague and Contradictory Plan Descriptions

The descriptions of the plans, though quite wordy, were still vague at some points, and the rule set for Plan P3 was self-contradicting. It was necessary to look at the code for the implementation in order to understand what exactly was meant by the authors. We have attempted to be more precise in this project report.

5.2.3 Point Estimates of the MPR

All calculations for the MPR were given as point estimates, with no mention of the standard deviation or variance of the result. Given that the MPR is the expected value of a random variable (the payoff), it is important to calculate some kind of confidence interval for the MPR. This is especially useful when comparing two different models/implementations which yield the same MPR. It is more difficult to make investment decisions when only point estimates of the MPR are given.

5.3 Learnings

This project has been quite helpful with improving my proficiency in programming. A significant amount of time was spent translating code for the implementations from R[1] and Matlab[2, 8] to Python. This was originally done to develop a better grasp of the application of the method of MPR estimation. Once this was finished, I began creating my own implementations, including antithetic variates, CIR volatility and vectorisation. I also made extensive use of the *pandas*, *matplotlib* and *numpy* libraries, as well as Python-specific commands such as nested list comprehensions.

I have also learned to use numerical methods (Euler-Maruyama and Milstein) to implement stochastic simulations of different SDEs (GBM and CIR) and to appropriately implement vastly different payoff functions.

Finally, I was also able to learn about the fundamental theory behind Multilevel Monte Carlo (how it works, its use of the weak and strong convergence of a numerical method, the requirement of the Lipschitz assumption in both MC and MLMC, and the computational cost) as well as understanding how to apply it to a stochastic model.

5.4 Further Developments

There is still much left to be done in looking at the theory behind estimating the MPR under various market conditions. However, it is important to note that much of the work done in this paper can be extended to many other derivatives beyond stock market linked savings plans, and we can also make the assumption of different SDE models for other underlying instruments such as bonds, commodities, currencies and interest rates. This is because the fundamental idea is quite general: observing the behaviour of a stochastic process and applying a payoff function to its trajectory.

For derivative contracts, such as swaps, where there is more than one underlying asset, one could look at a two-dimensional SDE model and extending it further to the general multivariate case.

Some further analysis could be done for antithetic variates to find other possible characteristics of a payoff function that could affect the extent of the variance reduction.

More attention can definitely be given to the Multilevel Monte Carlo method and how it applies to the problem of MPR estimation. As we have mentioned in Section 3.1.3, our progress with this has been limited due to time constraints. However, given more time, we could look at implementing MLMC with antithetic variates for P4 and P5. We could also conduct further analysis of how the implementation works as we vary ϵ and and of the irregularities that we came across in our results.

Finally, we note that the Lipschitz assumption[11, 12] is a very important condition for the Monte Carlo estimate of the payoff. We have not mentioned anything about whether these functions satisfy the Lipschitz assumption (at a glance, we can say that they are not because of the discontinuous jumps, e.g. getting either a fixed return or none at all at the end of Plan P3). However, we have not encountered any obvious problems that could be attributed to the violation of the Lipschitz condition. Perhaps more attention could be given to this issue and why it seems to be working in the context of MPR estimation.

6 Appendices

6.1 Definitions

6.1.1 Wiener Process

The **Wiener process** (also called **Brownian motion**) $\{W_t\}_{t\geq 0}$ is a continuous-time stochastic process characterised by the following properties:

1. $W_0 = 0$ almost surely.
2. W has independent increments, i.e. for every $t > 0$, the future increments $W_{t+u} - W_t$, $u \geq 0$, are independent of the past values W_s, $s < t$.
3. W has Gaussian increments, i.e. $W_{t+u} - W_t \sim N(0, u)$
4. W has continuous paths, i.e. W_t is continuous in t with probability 1.

6.1.2 Filtered Probability Space

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. If $\{\mathcal{F}_t\}_{t\geq 0}$ is a filtration, then $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ is called a **filtered probability space**.

6.1.3 Complete Probability Space

A probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is said to be a **complete probability space** if for all $B \in \mathcal{F}$ with $\mathbb{P}(B) = 0$ and all $A \subset B$ one has $A \in \mathcal{F}$.

6.1.4 Filtration

Let $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ be a filtered probability space. For every $t \geq 0$, let $\mathcal{F}_t$ be a sub σ-algebra of $\mathcal{F}$. Then $\{\mathcal{F}_t\}_{t\geq 0}$ is called a **filtration** if $\mathcal{F}_s \subseteq \mathcal{F}_t \ \forall \ s \leq t$. So filtrations are families of σ-algebras that are ordered non-decreasingly.

6.2 Derivations

6.2.1 P1 Analytic Solution[1]

Recall that the description of Plan P1 is as follows: *“A 3-year plan delivering a 13% return is received if the Index is higher at the end of the plan and a return of 0% if it is lower."*

We assume that the Index follows a GBM, given by the linear SDE

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t), \quad t \geq 0,$$

with initial condition $X(s) = X$, giving the solution

$$X(s+t) = X\exp\Big((\mu - 0.5\sigma^2)t + \sigma\sqrt{t}Z\Big), \quad t \geq 0. \tag{6.1}$$

By (6.1), we have

$$X(3) = X_0 \exp\Big(3(\mu - 0.5\sigma^2) + \sigma W(3)\Big) = X_0 \exp\Big(3(\mu - 0.5\sigma^2) + \sigma\sqrt{3}Z\Big),$$

where $Z \sim N(0,1)$, since $W(3) \sim N(0,3) = \sqrt{3}Z$ and $X(0) = X_0$. We can see that $X(3) > X_0$ if and only if

$$\begin{aligned} &\exp\Big(3(\mu - 0.5\sigma^2) + \sigma\sqrt{3}Z, \Big) > 1 \\ \iff &Z > \frac{\sqrt{3}(\mu - 0.5\sigma^2)}{\sigma} =: z_1. \end{aligned}$$

Hence

$$\mathbb{P}[X(3) > X_0] = \mathbb{P}[Z > z_1)] = 1 - N(z_1),$$

where $N(x)$ is the cumulative probability function of the Standard Normal distribution, i.e.

$$N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2} dy, \quad x \in (-\infty, \infty).$$

Recall that under P1, the plan holder will receive a 13% return if $X(3) > X_0$ but nothing if $X(3) \leq X_0$. Hence, the MPR for P1 is

$$\text{MPR} = 0.13 \cdot \mathbb{P}[X(3) > X_0] + 0 \cdot \mathbb{P}[X(3) \leq X_0)] = 0.13(1 - N(z_1)). \qquad \square$$

6.2.2 P2 Analytic Solution[1]

Recall that the description of Plan P1 is as follows: *“A 5-year plan delivering a 40% return if the Index is increased by more than 40% at the end of the plan and a return of 0.5% if the Index is not increased by more than 0.5%. If at the end of the plan the Index is increased by a percentage between 0.5% and 40%, a return of the same percentage is received."*

Again, We assume that the Index follows a GBM, given by the linear SDE

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t), \quad t \geq 0,$$

with initial condition $X(s) = X$, giving the solution

$$X(s+t) = X\exp\Big((\mu - 0.5\sigma^2)t + \sigma\sqrt{t}Z\Big), \quad t \geq 0. \tag{6.2}$$

By (6.2), the percentage change of the Index in 5 years is $\hat{Z} - 1$, where

$$\hat{Z} := \exp\Big(5(\mu - 0.5\sigma^2) + \sigma\sqrt{5}Z\Big),$$

where Z appears as $W(5) \sim N(0,5) = \sqrt{5}Z$. Hence, the MPR for P2 is

$$\text{MPR} = 0.4 \cdot \mathbb{P}[\hat{Z} - 1 > 0.4] + \mathbb{E}\Big[(\hat{Z} - 1)I_{\{0.05 \leq \hat{Z} - 1 \leq 0.4\}}\Big] + 0.05 \cdot \mathbb{P}[\hat{Z} - 1 < 0.05], \tag{6.3}$$

where I_A is the indicator function of set A, i.e. $I_A(x) = 1$ if $x \in A$ and 0 otherwise. Similar to what we did for P1, we can show

$$\mathbb{P}[\hat{Z} - 1 > 0.4] = 1 - N(z_2), \quad \mathbb{P}[\hat{Z} - 1 < 0.05] = N(z_3), \tag{6.4}$$

where

$$z_2 = \frac{\log(1.4) - 5(\mu - 0.5\sigma^2)}{\sigma\sqrt{5}}, \quad z_3 = \frac{\log(1.05) - 5(\mu - 0.5\sigma^2)}{\sigma\sqrt{5}}. \tag{6.5}$$

We will now compute the second term in (6.3):

$$\begin{aligned}\mathbb{E}\Big[(\hat{Z} - 1)I_{\{0.05 \leq \hat{Z} - 1 \leq 0.4\}}\Big] &= \mathbb{E}\Big[\hat{Z}I_{\{1.05 \leq \hat{Z} \leq 1.4\}}\Big] - \mathbb{P}[1.05 \leq \hat{Z} \leq 1.4] \\ &= \mathbb{E}\Big[\hat{Z}I_{\{1.05 \leq \hat{Z} \leq 1.4\}}\Big] - N(z_2) + N(z_3).\end{aligned}$$

Now

$$\begin{aligned}\mathbb{E}\big[\hat{Z}I_{\{1.05 \leq \hat{Z} \leq 1.4\}}\big] &= \int_{z_3}^{z_2} \frac{1}{2\pi}\exp\Big(5(\mu - 0.5\sigma^2) + \sigma\sqrt{5}z - 0.5z^2\Big)dz \\ &= \int_{z_3}^{z_2} \frac{1}{2\pi}\exp\Big(5\mu - 0.5(z - \sigma\sqrt{5})^2\Big)dz \\ &= e^{5\mu}\int_{z_3 - \sigma\sqrt{5}}^{z_2 - \sigma\sqrt{5}} \frac{1}{2\pi}\exp\Big(-\frac{1}{2}y^2\Big)dy \\ &= e^{5\mu}\big[N(z_2 - \sigma\sqrt{5}) - N(z_3 - \sigma\sqrt{5})\big].\end{aligned}$$

Hence

$$\mathbb{E}\Big[(\hat{Z}-1)I_{\{0.05\leq\hat{Z}-1\leq0.4\}}\Big] = e^{5\mu}[N(z_2-\sqrt{5}\sigma)-N(z_3-\sqrt{5}\sigma)]+N(z_2)+N(z_3). \tag{6.6}$$

Substituting (6.4) and (6.6) into (6.3), we get our final expression for the MPR:

$$\text{MPR} = 0.4-1.4N(z_2)+1.05N(z_3)+e^{5\mu}\big[N(z_2-\sqrt{5}\sigma)-N(z_3-\sqrt{5}\sigma)\big],$$

where z_2 and z_3 are defined by (6.5).

6.2.3 P3 Analytic Solution[1]

Recall that the description of Plan P3 is as follows: *“A 5-year plan delivering a 35% return at the end of the plan if the Index was greater than its initial level at the end of Year 3, 4 and 5, but if the Index is lower at the end of any of these three years, a return of 0% is received."*

Once again, we assume that the Index follows a GBM, given by the linear SDE

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t), \quad t \geq 0,$$

with initial condition $X(s) = X$, giving the solution

$$X(s+t) = X\exp\Big((\mu-0.5\sigma^2)t+\sigma\sqrt{t}Z\Big), \quad t \geq 0. \tag{6.7}$$

Then by (6.7) we have that

$$\begin{aligned}\mathbb{P}\big[X(s+t)\leq Y|X(s)=X\big] &= \mathbb{P}\big[X\exp\big((\mu-0.5\sigma^2)t+\sigma\sqrt{t}Z\big)\leq Y|X(s)=X\big]\\ &= \mathbb{P}\left[Z\leq\frac{\log(Y/x)-(\mu-0.5\sigma)t}{\sigma\sqrt{t}}\right]\\ &= N(z_4), \quad Y>0, \end{aligned} \tag{6.8}$$

where

$$z_4 = \frac{\log(Y/x)-(\mu-0.5\sigma)t}{\sigma\sqrt{t}}.$$

From this, we can derive the transition probability density of the solution.

From (6.8), we know that

$$\mathbb{P}[X(s+t) \leq Y | X(s) = X] = \int_{-\infty}^{z_4} \frac{1}{\sqrt{2\pi}} \exp(-\tfrac{1}{2}s^2) ds.$$

Using a change of variables,

$$\hat{s} = X\exp\big((\mu - 0.5\sigma^2)t + \sigma\sqrt{t}s\big) \implies s = \frac{\log(\hat{s}/x) - (\mu - 0.5\sigma^2)t}{\sigma\sqrt{t}},$$

and

$$ds = \frac{1}{\hat{s}\sigma\sqrt{t}} d\hat{s}.$$

Then

$$\mathbb{P}[X(s+t) \leq Y | X(s) = X] = \int_{-\infty}^{Y} \frac{1}{\sqrt{2\pi t}\sigma\hat{s}} \exp\left(-\frac{1}{2}\left(\frac{\log(\hat{s}/x) - (\mu - 0.5\sigma^2)t}{\sigma\sqrt{t}}\right)^2\right) ds.$$

Hence, the transition probability density is

$$p(X; Y, t) = \frac{1}{\sqrt{2\pi t}\sigma Y} \exp\left(-\frac{[\log(Y/x) - (\mu - 0.5\sigma^2)t]^2}{2t\sigma^2}\right).$$

By the Markov Property of the GBM, we have

$$\begin{aligned} & \mathbb{P}\big[X(3) \geq X_0, \;\; X(4) \geq X_0, \;\; X(5) \geq X_0\big] \\ = \;& \mathbb{P}\big[X(3) \geq X_0\big] \cdot \mathbb{P}\big[X(4) \geq X_0 | X(3) \geq X_0\big] \cdot \mathbb{P}\big[X(5) \geq X_0 | X(4) \geq X_0\big] \\ = \;& \int_{X_0}^{\infty} p(X_0; X_3, 3)\left(\int_{X_0}^{\infty} p(X_3; X_4, 1)\left[\int_{X_0}^{\infty} p(X_4; X_5, 1) dX_5 \right] dX_4 \right) dX_3. \end{aligned} \tag{6.9}$$

Now, we will obtain an expression for the innermost integral $\int_{X_0}^{\infty} p(X_4; X_5, 1)dX_5$ in terms of X_4 only:

$$\int_{X_0}^{\infty} p(X_4; X_5, 1) dX_5 = \int_{X_0}^{\infty} \frac{1}{\sqrt{2\pi}\sigma X_5} \exp\left(-\frac{[\log(X_5/X_4) - (\mu - 0.5\sigma^2)]^2}{2\sigma^2}\right) dX_5.$$

Again, using a change of variables,

$$\hat{x} = \frac{\log(X_5/X_4) - (\mu - 0.5\sigma^2)}{\sigma} \implies dX_5 = \sigma X_5 d\hat{x}.$$

Then

$$\int_{X_0}^{\infty} p(X_4; X_5, 1) dX_5 = 1 - \int_{-\infty}^{d(X_4,1)} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\hat{x}^2 \right) d\hat{x}$$
$$= 1 - N\big(d(X_4, 1)\big),$$

where

$$d(x, t) = \frac{\log({}^{X_0}/_{x}) - (\mu - 0.5\sigma^2)t}{\sigma\sqrt{t}}. \tag{6.10}$$

Hence,

$$\int_{X_0}^{\infty} p(X_3; X_4, 1) \left[\int_{X_0}^{\infty} p(X_4; X_5, 1) dX_5 \right] dX_4$$
$$= \int_{X_0}^{\infty} \frac{1}{\sqrt{2\pi}\sigma X_4} \exp\left(-\frac{[\log({}^{X_4}/_{X_3}) - (\mu - 0.5\sigma^2)]^2}{2\sigma^2} \right) [1 - N(d(X_4, 1))] dX_4.$$

Setting

$$\frac{\log({}^{X_4}/_{X_3}) - \mu - 0.5\sigma^2}{\sigma} =: \bar{x}_4,$$

and noting

$$d(X_4, 1) = \frac{\log({}^{X_0}/_{X_3}) - [\sigma \bar{x}_4 + 2(\mu - 0.5\sigma^2)]}{\sigma} = d(X_3, 1) - \bar{x}_4 - \frac{(\mu - 0.5\sigma^2)}{\sigma},$$

we get

$$\int_{X_0}^{\infty} p(X_3; X_4, 1) \left[\int_{X_0}^{\infty} p(X_4; X_5, 1) dX_5 \right] dX_4$$
$$= \int_{d(X_3,1)}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-0.5\bar{x}_4^2} [1 - N(d(X_4, 1))] d\bar{x}_4$$
$$= 1 - N(d(X_3, 1)) - \int_{d(X_3,1)}^{\infty} \frac{1}{\sqrt{2\pi}} e^{0.5\bar{x}_4^2} N\left(d(X_3, 1) - \bar{x}_4 - \frac{\mu - 0.5\sigma^2}{\sigma} \right) d\bar{x}_4$$
$$= 1 - I(d(X_3, 1)), \tag{6.11}$$

where

$$I(x) := N(x) + \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-0.5\bar{x}_4^2} N\left(x - \bar{x}_4 - \frac{\mu - 0.5\sigma^2}{\sigma} \right) d\bar{x}_4, \quad x \in \mathbb{R}. \tag{6.12}$$

Substituting this into (6.9), we get

$$\mathbb{P}[X(3) \geq X_0, \;\; X(4) \geq X_0, \;\; X(5) \geq X_0] = \int_{X_0}^{\infty} p(X_0; X_3, 3)[1 - I(d(X_3, 1))] dX_3.$$

In the same way (6.11) was proved, we can then show

$$\begin{aligned}&\mathbb{P}[X(3) \geq X_0, \quad X(4) \geq X_0, \quad X(5) \geq X_0] \\ &= 1 - N(d(X_0, 3)) - \int_{d(X_0,3)}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-0.5\bar{x}_3^2} I(-\sqrt{3}\bar{x}_3 - 4(\mu - 0.5\sigma^2)/\sigma) d\bar{x}_3.\end{aligned}$$

Recall that the holder of Plan P3 will receive a 35% income only if $X(3) \geq X_0, X(4) \geq X_0$ and $X(5) \geq X_0$, otherwise they receive no income at the end of the plan. Hence, we obtain the following closed-form expression for the P3 MPR:

$$\text{MPR} = 0.35\left(1 - N(d(X_0, 3)) - \int_{d(X_0,3)}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-0.5\bar{x}_3^2} I(-\sqrt{3}\bar{x}_3 - 4(\mu - 0.5\sigma^2)/\sigma) d\bar{x}_3\right),$$

where $d(x, t)$ and $I(x)$ are defined by (6.10) and (6.12), respectively. □

6.3 Implementations

All implementations were written in Python and are publicly available on Github, here:

```
https://github.com/nvolfango/Final_Year_Project-Estimating_the_MPR
```

Acknowledgement

I would like to thank my supervisor, Dr. Cónall Kelly, for giving me guidance and direction over the course of the project.

References

[1] Feng, Ling and Huang, Zhigang and Mao, Xuerong (2015) *Mean Percentage of Returns for Stock Market Linked Savings Accounts* Applied Mathematics and Computation. ISSN 0096-3003 (In Press) , http://dx.doi.org/10.1016/j.amc.2015.09.049.

[2] Higham, Desmond J., *An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations*, SIAM Review, Vol. 43, No. 3 (2001), pp. 525–546.

[3] S. Dereich, A. Neuenkirch, L. Szpruch, *An Euler-Type Method for the Strong Approximation of the Cox-Ingersoll-Ross Process* (2000)

[4] Chan, K.C., Karolyi, G.A., Longstaff, F.A. and Sanders, A.B., *An Empirical Comparison of Alternative Models of the Short-Term Interest Rate*, The Journal of Finance, 47(3) (1992), 1209-1227.

[5] Nowman, K.B., *Gaussian Estimation of Single-Factor Continuous Time Models of the Term Structure of Interest Rates*, The Journal of Finance 52(4) (1997), 1695-1706.

[6] Higham, Desmond J., *An Introduction to Multilevel Monte Carlo for Option Valuation*, May 2015,

[7] Giles, M.B., *Improved Multilevel Monte Carlo Convergence uisng the Milstein Scheme*, in Monte Carlo and Quasi-Monte Carlo Methods, Springer, 2007, pp.343-358

[8] Giles, M.B., *Multilevel Monte Carlo Methods*, Acta Numerica (2018), Cambridge University Press, 2018

[9] Giles, M.B. *Multilevel Monte Carlo Path Simulation* Operations Research, Vol. 56, No.3, 2008, pp. 607-617

[10] Sigman, Karl, *Introduction to Variance Reduction Methods*, Columbia University, Department of Industrial Engineering and Operations Research, 2010.

[11] D. F. Anderson, D. J. Higham, and Y. Sun, *Complexity of multilevel Monte Carlo tau-leaping*, SIAM Journal on Numerical Analysis, 52 (2015), pp. 3106-3127.

[12] —— ,*Numerical approximations of stochastic differential equations with non-globally Lipschitz continuous coeffcients*, Memoirs of the American Mathematical Society, 236 (2014), p. in press.

[13] Duque, Joao & Paxson, Dean, *Empirical Evidence on Volatility Estimators* 2019.

[14] Samuelson, P. A. (1965), *Proof That Properly Anticipated Prices Fluctuate Randomly*, Industrial Management Review 6(2), 41-49.

[15] Lo, Andrew. (2007). *Efficient Markets Hypothesis*.

[16] P. Glasserman, Monte Carlo Methods in Financial Engineering, Springer, Berlin, 2004.