Loughborough University

18MAP301 Mathematical Finance Research Project

Simulating Sample Paths of Stochastic Processes Arising in Financial Engineering

John Morgan Blake
B830102

supervised by

Dr. Dmitri Tseluiko

Abstract

This project develops a sequence of increasingly involved methods for simulating paths of stochastic processes that arise in financial mathematics, including those used to model stock prices, and exact methods and the methods based on discretisation of stochastic differential equations. First we consider simulation of Brownian and geometric Brownian motions in one and multiple dimensions, and apply these models to the Monte Carlo pricing of exotic options. We also consider Gaussian models, as well as square root diffusions, for modelling interest rates. We finish by considering processes with jumps and option pricing when the underlying follows this process. As well as providing an overview of the relevant theory, plenty of MATLAB code for implementing the various models is included.

12th September, 2019

Contents

0 Introduction

This project aims to build upon the foundations lain by first courses in stochastic calculus and derivatives pricing. We will explore the theory behind—and the applications of—simulating realisations of various stochastic processes. These processes will include Brownian motion, geometric Brownian motion, more general Ornstein-Uhlenbeck processes, square root diffusions, and finally jump processes. We will also introduce Monte Carlo simulation, and propose methods to price exotic options using this technique.

In quantitative finance, we are often trying to develop a framework under which we can model various phenomena (e.g. stock prices, option prices, interest rates, default probability, etc). This project touches upon the development of such models, but is principally concerned with the problem of extracting data under the assumption that the model is completely valid (though, in practice, it is rare that a model is entirely correct).

The intended audience for this project is anyone with an interest in financial modelling, and has taken at least an introductory course to stochastic calculus. Some technical details about options, such as how payoffs are calculated, are explained at the appropriate point; for more information on option pricing, one cannot go wrong by consulting [2], and [21]. When discussing Brownian motion and geometric Brownian motion, [10] was a very thorough reference. On the topic of simulation, [6] was the primary text used, and this project is structured to follow a similar progression to the book. Moreover, when discussing variance reduction in Monte Carlo simulations, [12] gave a very good overview of the range of methods available. In this project, only variance reduction by the use of antithetic variates was considered, but there are many more that could have been included. For the final chapter–on jump processes–[17] was invaluable in providing background as to the processes that were simulated.

We begin with a brief overview of the preliminary understanding required for this project, and a lot of this will be referred back to later. There is little prose in the preliminaries, as most of it should already be known to the reader: probability distributions, stochastic calculus, and basic option theory.

Chapter 1 starts off by introducing two methods of discretising a stochastic differential equation (SDE) over a given time interval, and then applies this to generate simulations of Brownian motion using the random walk, and Brownian bridge, constructions. We also cover the method of principal components construction, which provides a smooth approximation to a sample path of Brownian motion. We first do these with sample paths of standard Brownian motion, before generalising to all Brownian motion.

Chapter 2 continues where the previous chapter left off, and we describe the simple way in which to transform sample paths of Brownian motion to geometric Brownian motion. We also generalise this and consider multi-dimensional sample paths of Brownian motion,

where the displacements in each dimension are not independent. We end this chapter by examining an alternative to geometric Brownian motion in modelling stock prices: the CEV model.

Chapter 3 is the densest chapter in this project, and builds upon the methods presented in chapters 1 and 2. We begin by covering the pricing of an exotic option over a single (1-dimensional) sample path, and then introduce Monte Carlo methods to estimate the actual option price by generating large numbers of sample paths. We then extend this methodology to price options that are dependent on more than one asset (which are not necessarily independent). In this chapter, we price Asian options, lookback options, spread options, and barrier options. We also apply antithetic variates to achieve some variance reduction in the Monte Carlo simulations.

Chapter 4 is slightly different to the other chapters in that we are no longer looking at simulating sample paths of a stock price process, but are simulating sample paths for stochastic interest rates. We cover multiple interest rate models in this chapter: the popular Vasicek model, the Ho-Lee model, and the square-root diffusion Cox-Ingersoll-Ross (CIR) model.

Chapter 5 ends the project with a foray into jump processes, which add a discrete "jump" term to the usual stochastic differential equation in order to capture this discontinuity in the stochastic process. We examine the way in which these jumps can be distributed, and also the distribution of the size of these jumps.

Throughout the project, figures generated using MATLAB have been included to illustrate the results, and the code that can be used to recreate these have been included in the appendix.

0.1 Preliminaries

In this preliminary section, we give some background to the project, and give some results and definitions that will be built upon in the following chapters.

Definition 0.1.1 (Normal Distribution)**.** *The normal distribution with mean μ and variance σ^2 has density*

$$\phi_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty \tag{1}$$

and cumulative distribution function

$$\Phi_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x} e^{-\frac{(y-\mu)^2}{2\sigma^2}} dy. \tag{2}$$

We use the notation $X \sim N(\mu, \sigma^2)$ to abbreviate the statement that the random variable X is normally distributed with mean μ and variance σ^2. If $Z \sim N(0,1)$, we say that the random variable Z is the standard normal distribution.

If $Z \sim N(0,1)$, (i.e. Z has a standard normal distribution), then $\mu + \sigma Z \sim N(\mu, \sigma^2)$. In this way, we can sample from any normal distribution by sampling only from the standard normal distribution and applying the given linear map.

Definition 0.1.2 (Multi-dimensional Normal Distribution)**.** *A d-dimensional normal distribution is described by a d-vector $\boldsymbol{\mu}$ and a $d \times d$ covariance matrix $\boldsymbol{\Sigma}$; we abbreviate this to $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (n.b. by the definition of covariance, $\boldsymbol{\Sigma}$ must be symmetric and positive semidefinite). The normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ has density*

$$\phi_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(x) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{(-\frac{1}{2}(x-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}))\right\}, \quad x \in \mathbb{R}^d, \tag{3}$$

where $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$.

Definition 0.1.3 (Chi-Square Distribution)**.** *The (central) chi-square distribution with k degrees of freedom is defined as the distribution of the sum of the squares of k independent standard normal random variables. We denote chi-square random variables by $Q \sim \chi^2_k$. The probability density function of Q is*

$$f(x;k) = \frac{x^{\frac{k}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{k}{2}} \Gamma(\frac{k}{2})}$$

for $x > 0$ and 0 otherwise, where $\Gamma(\cdot)$ is the gamma function, defined by the integral

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx, \quad \mathrm{Re}(z) > 0.$$

Note that the sum of independent chi-square variables is also itself a chi-square variable.

There also exists the non-central chi-square distribution with k degrees of freedom, which is distributed according to the sum $S = \sum_{i=0}^{k} X_i^2$ *where* $X_i \sim N(\mu_i, 1)$. *We say that* $S \sim \chi_k(\lambda)$, *where the non-centrality parameter* λ *is given by* $\lambda = \sum_{i=1}^{k} \mu_i^2$.

Definition 0.1.4 (Poisson Distribution)**.** *We say that a discrete random variable* X *follows a Poisson distribution (i.e.* $X \sim Po(\lambda)$*) if it has probability mass function*

$$\mathbb{P}(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}.$$

The Poisson distribution expresses the probability of a given number of events occurring in a fixed interval, under the assumption that the events occur at a constant rate λ, *and independently of the time elapsed since the last event.*

Definition 0.1.5 (Exponential Distribution)**.** *We say that a random variable* X *with probability density function*

$$f(t) = \begin{cases} \lambda e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

for some constant $\lambda > 0$ *follows an exponential distribution.*
Hence, the X *has cumulative distribution function*

$$F(t) = \mathbb{P}(X \leq t) = \int_0^t \lambda e^{-\lambda u} du = 1 - e^{-\lambda u}, \quad t \geq 0.$$

It also follows that the mean is $\mathbb{E}[X] = 1/\lambda$ *and the variance is* $\mathrm{Var}(X) = 1/\lambda^2$. *The moments of the exponential distribution are given as* $\mathbb{E}[X^n] = n!/\lambda^n$.

Definition 0.1.6 (Stochastic Process)**.** *Let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space and* S, T *sets. The family of* S*-valued random variables* $(X_t)_{t \in T}$ *is called a stochastic process with index set* T.

Definition 0.1.7 (Filtration)**.** *Let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space and* $(\mathcal{F}_s)_{s \geq 0}$ *a family of sub-*σ*-fields of* $\mathcal{F}$. *The collection* $(\mathcal{F}_s)_{s \geq 0}$ *is called a filtration whenever* $\mathcal{F}_s \subset \mathcal{F}_t$ *for* $s \leq t$. *A measurable space endowed with a filtration is called a filtered space.*

Definition 0.1.8 (Adapted Process)**.** *Let* $(\Omega, \mathcal{F})$ *be a filtered space,* $(\mathcal{F}_s)_{s \geq 0}$ *a given filtration and* $X = (X_t)_{t \geq 0}$ *a stochastic process. The process* X *is called* $(\mathcal{F}_s)_{s \geq 0}$*-adapted whenever* X_t *is* $\mathcal{F}_t$*-measurable for every* $t \geq 0$.

Theorem 0.1.9 (Strong Law of Large Numbers)**.** *Let* $(B_t)_{t \geq 0}$ *be standard Brownian motion. Then* $\lim_{t \to \infty} \frac{B_t}{t} = 0 \quad \mathbb{P} - a.s.$

Definition 0.1.10 (Markov Process)**.** *Let* $(X_t)_{t \geq 0}$ *be an adapted process on a given filtered probability space* $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t \geq 0}, \mathbb{P})$. *The process* $(X_t)_{t \geq 0}$ *is called a Markov process with respect to* $(\mathcal{F}_t)_{t \geq 0}$ *whenever*

$$\mathbb{E}_{\mathbb{P}}[f(X_t) \mid \mathcal{F}_s] = \mathbb{E}_{\mathbb{P}}[f(X_t) \mid \sigma(X_s)], \quad 0 \leq s \leq t$$

for all bounded measurable functions f.

Definition 0.1.11 (Brownian Motion)**.** *A d-dimensional Brownian motion $B = (B_t)_{t\geq 0}$ is a stochastic process, indexed by $[0,\infty)$, taking values in $\mathbb{R}^d$ such that:*

$B(0,\omega) = 0$, *for $\mathbb{P}$-almost all ω;*
$B(t_n) - B(t_{n-1}), \ldots, B(t_1) - B(t_0)$ *are independent for $n \geq 1$, $0 = t_0 < t_1 < \cdots < t_n < \infty$;*
$B(t) - B(s) \sim B(t+h) - B(s+h)$ *for all $0 \leq s < t$, $h \geq -s$;*
$B(t) - B(s) \sim N(0, t-s)^{\bigotimes}$, $\quad N(0,t)(dx) = \frac{1}{\sqrt{2\pi t}} e^{-x^2/2t} dx$.

Definition 0.1.12 (Wiener Measure)**.** *Let $0 = t_0 < t_1 < \cdots < t_n$ be arbitrary time-points and $A_j \in \mathcal{B}(\mathbb{R})$ arbitrary sets, $j = 1, \ldots, n$, $n = 1, 2, \ldots$. The unique probability measure $\mathbb{P}^x$ on $(\Omega, \mathbb{B}(\Omega))$ whose finite dimensional distributions are given by*

$$\mathbb{P}^x(B_0 = x) = 1$$

and

$$\mathbb{P}^x(B_{t_1} \in A_1, \ldots, B_{t_n} \in A_n) = \int_{\mathbb{R}} \prod_{i=1}^{n} 1_{A_i}(x_i) p_{t_i - t_{i-1}}(x_i - x_{i-1}) dx_i, \quad x_0 = x,\ n \in \mathbb{N}$$

is called the Wiener measure starting at x.

Theorem 0.1.13 (Central Limit Theorem)**.** *Let $X_1, X_2, \ldots$ be independent random variables, each with the same expected value μ and variance $\sigma^2 < \infty$. Then $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ satisfies:*

$$\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \xrightarrow{d} N(0,1).$$

Definition 0.1.14 (Martingale)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a filtered space with given filtration $(\mathcal{F}_t)_{t\geq 0}$. The random process $(X_t)_{t\geq 0}$ is an $(\mathcal{F}_t)_{t\geq 0}$-martingale whenever:*

$(X_t)_{t\geq 0}$ *is $(\mathcal{F}_t)_{t\geq 0}$-adapted;*
$\mathbb{E}_{\mathbb{P}}[|X_t|] < \infty$ *for all $t \geq 0$;*
$\mathbb{E}_{\mathbb{P}}[X_t|\mathcal{F}] = X_s$ *for each $s \leq t$;*

If we change the final condition and instead require $\mathbb{E}_{\mathbb{P}}[X_t|\mathcal{F}] \leq X_s$ or $\mathbb{E}_{\mathbb{P}}[X_t|\mathcal{F}] \geq X_s$, then $(X_t)_{t\geq 0}$ is called a supermartingale, or a submartingale, respectively.

Definition 0.1.15 ($M^2(S,T)$)**.** *We denote by $M^2(S,T)$ the class of stochastic processes $f : \mathbb{R}^+ \times \Omega \to \mathbb{R}$ such that:*

$f \in \mathcal{B}(\mathbb{R}^+) \times \mathcal{B}(\Omega)/\mathcal{B}(\mathbb{R})$;
$f(t,\omega)$ *is $\mathcal{F}_t^{\mathcal{B}}$-adapted;*
$\mathbb{E}\Big[\int_S^T f(t,\omega)^2 dt\Big] < \infty$.

Definition 0.1.16 (L_t^1)**.** *Let $t \geq 0$. A stochastic process $(X_t)_{t\geq 0}$ adapted to $(\mathcal{F}_t^X)_{t\geq 0}$ such that $\int_0^t |X_s| ds < \infty$ $\mathbb{P}$-a.s. is said to be of class L_t^1.*

Definition 0.1.17 (Itô Process)**.** *A stochastic process $(X_t)_{t\geq 0}$ is called an Itô process if: (1) $t \mapsto X_t$ is continuous $\mathbb{P}$-a.s. and (2) there exist stochastic processes $(b_s)_{s\in[0,t]} \in L_t^1$ and $(\sigma_s)_{s\in[0,t]} \in M_t^2$, for all $t \geq 0$, such that $dX_t = b_t dt + \sigma_t dB_t$. We refer to b_t as the drift term and σ_t as the diffusion term.*

Theorem 0.1.18 (Special Itô Formula)**.** *Let $(B_t)_{t\geq 0}$ be standard Brownian motion. Moreover, let $h : \mathbb{R}^+ \times \mathbb{R} \mapsto \mathbb{R}$ be C^1 with respect to its first variable, and C^2 with respect to its second variable, such that $\sigma_t \frac{\partial h}{\partial x}(t, B_t) \in M_{[0,t]}^2$ for all $t \geq 0$. Define $X_t := h(t, B_t)$. Then $(X_t)_{t\geq 0}$ is an Itô process and we have:*

$$X_t - X_0 = \int_0^t \left(\frac{\partial h}{\partial s}(s, B_s) + \frac{1}{2} \frac{\partial^2 h}{\partial x^2}(s, B_s) \right) ds + \int_0^t \frac{\partial h}{\partial x}(s, B_s) dB_s,$$

or more concisely:

$$dX_t = \left(\frac{\partial h}{\partial t}(t, B_t) + \frac{1}{2} \frac{\partial^2 h}{\partial x^2}(t, B_t) \right) dt + \frac{\partial h}{\partial x}(t, B_t) dB_t.$$

Theorem 0.1.19 (General Itô Formula)**.** *Let $(X_t)_{t\geq 0}$ be an Itô process with drift term b_t and diffusion term σ_t. Moreover, let $h : \mathbb{R}^+ \times \mathbb{R} \mapsto \mathbb{R}$ be C^1 with respect to its first variable, and C^2 with respect to its second variable, such that $\sigma_t \frac{\partial h}{\partial x} \in M_{[0,t]}^2$ for all $t \geq 0$. Define $Y_t := h(t, X_t)$. Then $(Y_t)_{t\geq 0}$ is an Itô process and we have:*

$$Y_t - Y_0 = \int_0^t \left(\frac{\partial h}{\partial s}(s, X_s) + b_s \frac{\partial h}{\partial x}(s, X_s) + \frac{1}{2} \sigma_s^2 \frac{\partial^2 h}{\partial x^2}(s, X_s) \right) ds + \int_0^t \sigma_s \frac{\partial h}{\partial x}(s, X_s) dB_s,$$

or more concisely:

$$dY_t = \left(\frac{\partial h}{\partial t}(t, X_t) + b_t \frac{\partial h}{\partial x}(t, X_t) + \frac{1}{2} \sigma_t^2 \frac{\partial^2 h}{\partial x^2}(t, X_t) \right) dt + \sigma_t \frac{\partial h}{\partial x}(t, X_t) dB_t.$$

Theorem 0.1.20 (Cameron-Martin)**.** *Let $m \in \mathbb{R}$, and define the stochastic process:*

$$Z_t^m = e^{mB_t - \frac{1}{2}m^2 t}, \quad t \geq 0.$$

Also, on the space $(\Omega, \mathcal{F}_T^B)$, define the probability measure

$$\mathbb{Q}_T^m(A) = \mathbb{E}_{\mathbb{P}}[Z_T^m 1_A],$$

where $\mathbb{P}$ is the Wiener measure. Then under the probability measure $\mathbb{Q}_T^m$, the random process

$$B_t^m := B_t - mt, \quad 0 \leq t \leq T,$$

is a Brownian motion.

Theorem 0.1.21 (Girsanov). *Let $(X_t)_{t\geq 0}$ be an Itô process of the form $dX_t = a_t dt + dB_t$, $t \in [0,T]$, with a given $T > 0$. Put*

$$M_t := e^{-\int_0^t a_s dBs - \frac{1}{2}\int_0^t a_s^2 ds}, \quad t \in [0,T],$$

and suppose Novikov's condition:

$$\mathbb{E}[e^{\frac{1}{2}\int_0^T a_s^2 ds}] < \infty$$

holds. Define the probability measure $\mathbb{Q}_T(A) = \mathbb{E}[M_T 1_A]$ on $(\Omega, \mathcal{F}_T^B)$. Then $(X_t)_{t\geq 0}$ is standard Brownian motion under $\mathbb{Q}_T$

Remark 0.1.22 (Expectation of Geometric Brownian Motion). *If B_t is standard Brownian motion, then*

$$\mathbb{E}[e^{uB_t}] = e^{\frac{1}{2}u^2 t}, \quad u \in \mathbb{R}. \tag{4}$$

Moreover, if $X_t \sim GBM(\mu, \sigma^2)$ (i.e. $X_t = x_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma B_t}$) then

$$\mathbb{E}[X_t] = x_0 e^{\mu t}. \tag{5}$$

Definition 0.1.23 (Ornstein-Uhlenbeck Process). *The Ornstein-Uhlenbeck process $X(t)$ is defined by the SDE*

$$dX(t) = -\theta X(t)dt + \sigma dB(t)$$

where $\theta > 0$ and $\sigma > 0$. Explicitly, this process is given as

$$X(t) = e^{-\frac{\alpha t}{2}} B_{e^{\alpha t}}$$

with $\alpha > 0$.

Definition 0.1.24. *Let μ and ν be two measures on the measurable space $(X, \mathcal{A})$ and let $\mathcal{N}_\mu := A \in \mathcal{A} \mid \mu(A) = 0$ be the set of all μ-null sets, and define $\mathcal{N}_\nu$ similarly. Then, the measure ν is called absolutely continuous with respect to μ if $\mathcal{N}_\nu \supset \mathcal{N}_\mu$. This is denoted by $\nu \ll \mu$.*

Definition 0.1.25 (Equivalent Measure). *These two measures are called equivalent if $\nu \ll \mu$ and $\mu \ll \nu$. Equivalently, two measures are equivalent if $\mathcal{N}_\mu = \mathcal{N}_\nu$.*

Definition 0.1.26 (Risk-Neutral Measure). *A risk-neutral measure is a measure $\tilde{\mathbb{P}}$ that is equivalent to $\mathbb{P}$ under which the discounted stock price process $D(t)S(t)$ is a martingale.*

Theorem 0.1.27 (Risk-Neutral Pricing Formula). *Let $X(T)$ be a $\mathcal{F}_T$-measurable random variable that represents the payoff of a security, and let $\tilde{\mathbb{P}} = \tilde{\mathbb{P}}_T$ be the risk-neutral measure. The arbitrage-free price at time t of the security is then given by*

$$V(t) = \tilde{\mathbb{E}}\left[\exp\left(-\int_t^T R(s)ds\right)X(T) \mid \mathcal{F}_t\right].$$

Recall that the payoff of a European option is given as a function of only the strike price, K, and the price of the underlying asset at the maturity date T. The payoff of the European call and put options are given by:

$$C_t = e^{rt}\mathbb{E}^*[(S_T^* - K^*)^+|\mathcal{F}_t], \quad \mathbb{P}^*\text{-a.s.}$$

or with the notation $C_t^* = e^{-rt}C_t$,

$$C_t^* = \mathbb{E}^*[(S_T^* - K^*)^+|\mathcal{F}_t], \quad \mathbb{P}^*\text{-a.s.}$$

Under the assumptions of Black-Scholes-Merton, we have a tractable solution for the pricing of European options:

Theorem 0.1.28 (Black-Scholes Formula)**.** *Suppose the share price of a stock with volatility σ is S_0 at time $t = 0$. Then with maturity date T, the expression*

$$C_0 = S_0\Phi\Big(\frac{\log(S_0/K) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}\Big) - Ke^{-rT}\Phi\Big(\frac{\log(S_0/K) + (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}\Big)$$

gives for the buyer the fair price for a European call option at strike price K given the risk-free interest rate r.

We also have the put-call parity formula:

Theorem 0.1.29 (Put-Call Parity)**.** *Let C_t and P_t denote respectively the prices of the European call and put options at time $t \in [0, T]$ with maturity date T, S_t be the stock price at time t, and K the strike price. Then the put-call parity formula holds:*

$$C_t - P_t = S_t - Ke^{-r(T-t)}, \quad t \in [0, T].$$

1 Constructing Brownian Motion

Before we continue with the simulation of sample paths of a given process, we must introduce a method for finding approximate numerical solutions to the SDE that governs the process. In deterministic calculus, the Euler method does exactly this for ordinary differential equations, and it turns out that this generalises well to an Itô calculus setting [11, p.305]. In the following, we will use the notation $X_j = X(t_j)$, $\Delta t_j = t_j - t_{j-1}$, and $\Delta B_j = B_j - B_{j-1} = B(t_j) - B(t_{j-1})$.

Theorem 1.0.1 (Euler-Maruyama Approximation)**.** *Let $(X(t))_{t\in[0,T]}$ be an Itô process satisfying the scalar stochastic differential equation*

$$dX(t) = f(t, X(t))dt + g(t, X(t))dB(t) \tag{6}$$

with initial value $X(0) = x$. Now let $0 = t_0 < t_1 < \cdots < t_n = T$ be a set of time points partitioning the interval $[0, T]$. Then, the Euler-Maruyama approximation is a continuous time stochastic process $(Y(t))_{t\in[0,T]}$ satisfying the iteration

$$Y_i = Y_{i-1} + f(t_{i-1}, Y_{i-1})\Delta t_i + g(t_{i-1}, Y_{i-1})\Delta B_i \tag{7}$$

for $i = 1, \ldots, n$ with initial value $Y_0 = x$.

We will be using this method of approximating solutions to SDEs throughout the project. It will be useful to remember that $(B_i - B_{i-1}) \sim N(0, \sqrt{t_i - t_{i-1}})$. It is possible to show that the Euler-Maruyama method converges with (strong) order of convergence $\gamma = 1/2$ for sufficiently well behaved functions f, g [7, p.537]. Note that if $g \equiv 0$, then $X_j - X_{j-1} = f(X_{j-1})\Delta t$, and $\mathbb{E}[X_j - X_{j-1}] \leq \max_{t\in[0,T]}\{f(t)\}\Delta t$ and the method reduces to Euler's method for ordinary differential equations (which has $\gamma = 1$ in the deterministic case). Hence, this method is subject to discretisation error, and it is a natural question to ask whether one can reduce the discretisation error (equivalently, whether one can improve the strong rate of convergence). It turns out that considering also the second order terms in the Itô-Taylor expansion raises the strong rate of convergence to $\gamma = 1$ in the stochastic case.

Theorem 1.0.2 (Milstein Approximation)**.** *Let $(X(t))_{t\in[0,T]}$ be an Itô process satisfying the scalar stochastic differential equation*

$$dX(t) = f(t, X(t))dt + g(t, X(t))dB(t) \tag{8}$$

with initial value $X(0) = x$. Now let $0 = t_0 < t_1 < \cdots < t_n = T$ be a set of time points partitioning the interval $[0, T]$. Then, the Milstein approximation is a continuous time stochastic process $(Y(t))_{t\in[0,T]}$ satisfying the iteration

$$Y_j = Y_{j-1} + \Delta t f(Y_{j-1}) + g(X_{j-1})\Delta B_j + \tfrac{1}{2}g(X_{j-1})g'(X_{j-1})((\Delta B_j)^2 - \Delta t) \tag{9}$$

for $j = 1, \ldots, n$ with initial value $Y_0 = x$.

The Milstein approximation gives a smaller error than the Euler-Maruyama at each step, but has the disadvantage of being much more computationally expensive by requiring the first derivative of g. The feasibility of the Milstein approximation in this project is not questioned, but could become apparent in industry, where computational efficiency is much more important.

1.1 Random Walk Construction

We take a sequence of points $\{t_k\}_{k=1}^n$ such that $0 < t_1 < \cdots < t_n$. By independence and the Gaussian distribution of Brownian increments, we can model a discrete version of Brownian motion by drawing independent random variables $\{Z_{k=1}^n\}$ from the standard Gaussian distribution (i.e. $Z_k \sim N(0,1)$). We then set $t_0 = 0$ and $B(0) = 0$. Successive values are then generated by the recursion

$$B(t_{i+1}) = B(t_i) + \sqrt{t_{i+1} - t_i} Z_{i+1} \tag{10}$$

for $i = 0, \ldots, n-1$, which follows from Theorem 1.0.1.

If instead $X \sim BM(\mu, \sigma^2)$ with constant μ, σ, and given $X(0)$, we can use the fact that $((X(t) - \mu t)/\sigma) \sim BM(0,1)$ to rewrite the above recursion as

$$X(t_{i+1}) = X(t_i) + (t_{i+1} - t_i)\mu + \sqrt{t_{i+1} - t_i}\sigma Z_{i+1} \tag{11}$$

for $i = 0, \ldots, n-1$.

We can extend this to non-constant time-dependent $\mu(t)$, $\sigma(t)$:

$$X(t_{i+1}) = X(t_i) + \int_{t_i}^{t_{i+1}} \mu(s)ds + \sqrt{\int_{t_i}^{t_{i+1}} \sigma^2(s)ds} Z_{i+1} \tag{12}$$

for $i = 0, \ldots, n-1$.

All of these recurrences have provided simulated values $\{X(t_i)\}_{i=1}^n$ that have a joint distribution coinciding with the joint distribution of the corresponding Brownian motion at $\{t_i\}_{i=1}^n$, and for this reason are called 'exact methods'.

Definition 1.1.1. *We call a method exact if the joint distribution of the simulated values coincides with the joint distribution of the corresponding Brownian motion at the sampled points.*

Example 1.1.2. *The methods presented in (10)–(12) are all exact, as the joint distribution of $(B(t_1), \ldots, B(t_n))$ or $(X(t_1), \ldots, X(t_n))$ coincide with the joint distribution of the corresponding Brownian motion on $(t_1, \ldots, t_n)$.*

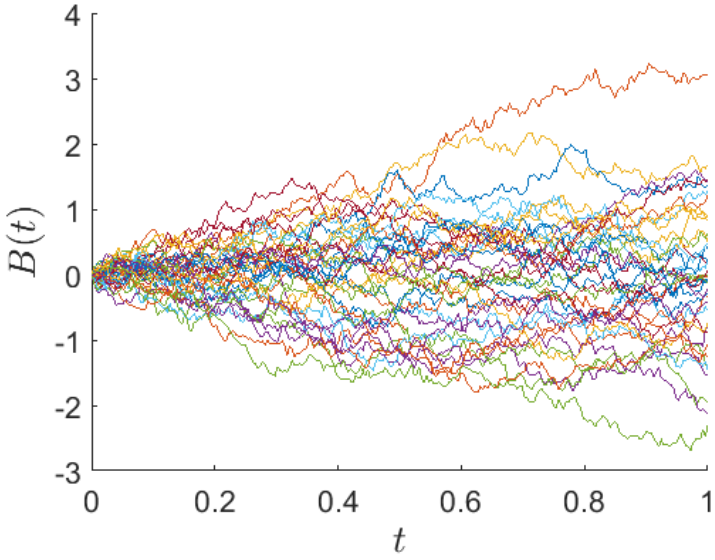

Figure 1: Forty sample paths of standard Brownian motion under the random walk construction with 252 timesteps between $t = 0$ and $t = 1$.

1.1.1 Cholesky Factorisation

We now expand on another method of simulating a vector $(B(t_1), \ldots, B(t_n)) \sim N(0, \boldsymbol{C})$, where $\boldsymbol{C}_{ij} = \min(t_i, t_j)$ defines the covariance matrix. We may simulate this vector by AZ, where $Z = (Z_1, \ldots, \mathbb{Z}_n)^T \sim N(0, \boldsymbol{I})$ and $\boldsymbol{A}$ is the lower triangular matrix satisfying $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{C}$. When the covariance matrix $\boldsymbol{C}$ is as above,

$$A = \begin{pmatrix} \sqrt{t_1} & 0 & \ldots & 0 \\ \sqrt{t_1} & \sqrt{t_2 - t_1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{t_1} & \sqrt{t_2 - t_1} & \ldots & 0 \end{pmatrix} \tag{13}$$

which can be verified by direct computation.

Note that evaluating $\boldsymbol{A}\boldsymbol{Z}$ in this fashion has complexity $O(n^2)$, but the random walk construction (10) does exactly this in $O(n)$. This is due to the implicit exploitation of the fact that all of the non-zero entries in each column are identical.

If $X \sim BM(\mu, \sigma^2)$, the mean vector of $(X(t_1), \ldots, X(t_n))$ has i^{th} component μt and covariance matrix $\sigma^2 \boldsymbol{C}$. The Cholesky factor is $\sigma \boldsymbol{A}$. Again, this factorisation is efficiently implemented by the random walk construction (11).

1.2 Brownian Bridge Construction

Another method of simulating Brownian motion relies again on independence of increments. In the random walk construction, we generated the vector $(B(t_1), \ldots, B(t_n))$ from left to right; we now generate a final value for $B(t_n)$, then $B(t_{\lfloor n/2 \rfloor})$ conditioned on $B(t_n)$, and continue in this manner, filling in intermediate values. If we take the rightmost endpoint of the interval to be a power of two, this is reminiscent of Lévy's original construction of Brownian motion [15, p.28–29]. We can continue the process of further refining the construction as long as we like, and as the dyadic rationals on a closed interval are dense on that interval, we have that in the limit (as the length of the timesteps go to zero), this process gives us Brownian motion.

One situation in which we may choose the Brownian Bridge construction over the random walk would be if we had some fixed time in which we had to perform computations (e.g. high-frequency options trading), and didn't know the amount of time each computation would take. Then, we could implement a Brownian Bridge construction with 2^k timepoints (for some small k), cache the result, then increment k by one and repeat the process, thus increasing the resolution of the sample path by a factor of 2. When the total allowed time ran out, the last cached sample path would be returned as the best approximation. This caching is impossible using the random walk simulation, as the step size between timepoints must be specified prior to runtime, whilst the Brownian Bridge method can continue adding resolution to the sample path indefinitely.

1.3 Principal Components Construction

Using the representation

$$\begin{pmatrix} B(t_1) \\ B(t_2) \\ \vdots \\ B(t_n) \end{pmatrix} = \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix} Z_1 + \cdots + \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{nn} \end{pmatrix} Z_n \tag{14}$$

where $(Z_1, Z_2, \ldots, Z_n) \sim N(0, \boldsymbol{I})$, let $a_i = (a_{1i}, a_{2i}, \ldots, a_{ni})^T$. Then let $\boldsymbol{A}$ be the $n \times n$ matrix with columns $a_1, a_2, \ldots, a_n$. This is a valid construction of the discrete Brownian path if $\boldsymbol{A}\boldsymbol{A}^T$ is the covariance matrix, $\boldsymbol{C}$, of $B = (B(t_1), B(t_2), \ldots, B(t_n))^T$. The expected approximation error is given by $\mathbb{E}[||B - \sum_{i=1}^{k} a_i Z_i||^2]$, where $||x||^2 = x^T x$, when only the first k terms in (14) are considered. For each fixed k, this is minimised by the use of principal components; specifically, when $a_i = \sqrt{\lambda_i}\vec{v_i}$, $i = 1, \ldots, n$ with $\lambda_1 > \lambda_2 > \cdots > \lambda_n > 0$ the eigenvalues of $\boldsymbol{C}$, with corresponding normalised eigenvectors $\vec{v_i}$.

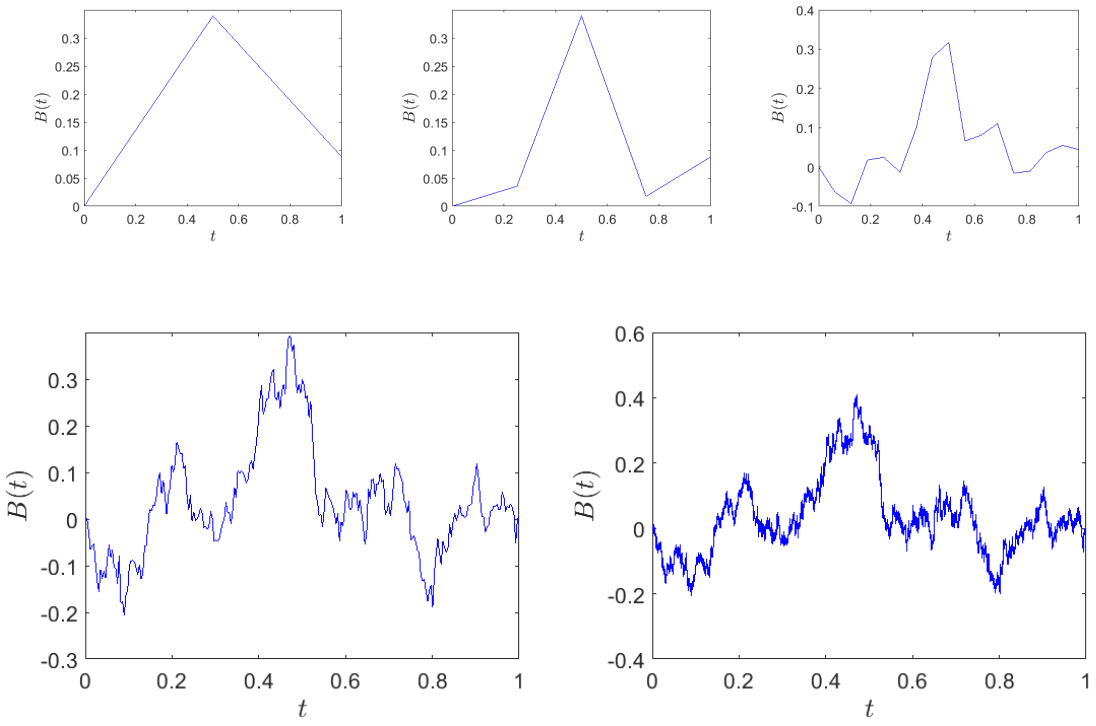

Figure 2: One sample path of Brownian motion, simulated using the Brownian Bridge construction, with 2^k time-steps, where $k = 1, 2, 4, 8, 16$ when read from left to right, top row then bottom row. We have fixed $\mu = 1$, $\sigma = 0.5$.

For an n-step path with equal spacing, $t_{i+1} - t_i = \Delta t$, [1] showed that

$$v_i(j) = \frac{2}{\sqrt{2n+1}} \sin\left(\frac{2i-1}{2n+1} j\pi\right), \quad j = 1, \ldots, n. \tag{15}$$

and

$$\lambda_i = \frac{\Delta t}{4} \sin^{-2}\left(\frac{2i-1}{2n+1}\frac{\pi}{2}\right), \quad i = 1, \ldots, n. \tag{16}$$

The intuition behind principal components construction is that the largest eigenvalues of $\boldsymbol{C}$ contribute the most to the overall shape of the Brownian motion, and so we can discard all but the first few eigenvalue/eigenvector pairs and get a smooth approximation to the overall shape of Brownian motion.

This method is not exact, but gives a good idea as to the overall shape of the sample path.

One can use the deconstruction of a sample path into its principal components as a technique to statistically determine whether that path is likely to be one of Brownian motion. The expected size of the largest eigenvalue of a random correlation matrix can be calculated, and if the largest eigenvalue of the correlation matrix between a set of asset returns is significantly larger than expected, one can say with some degree of certainty

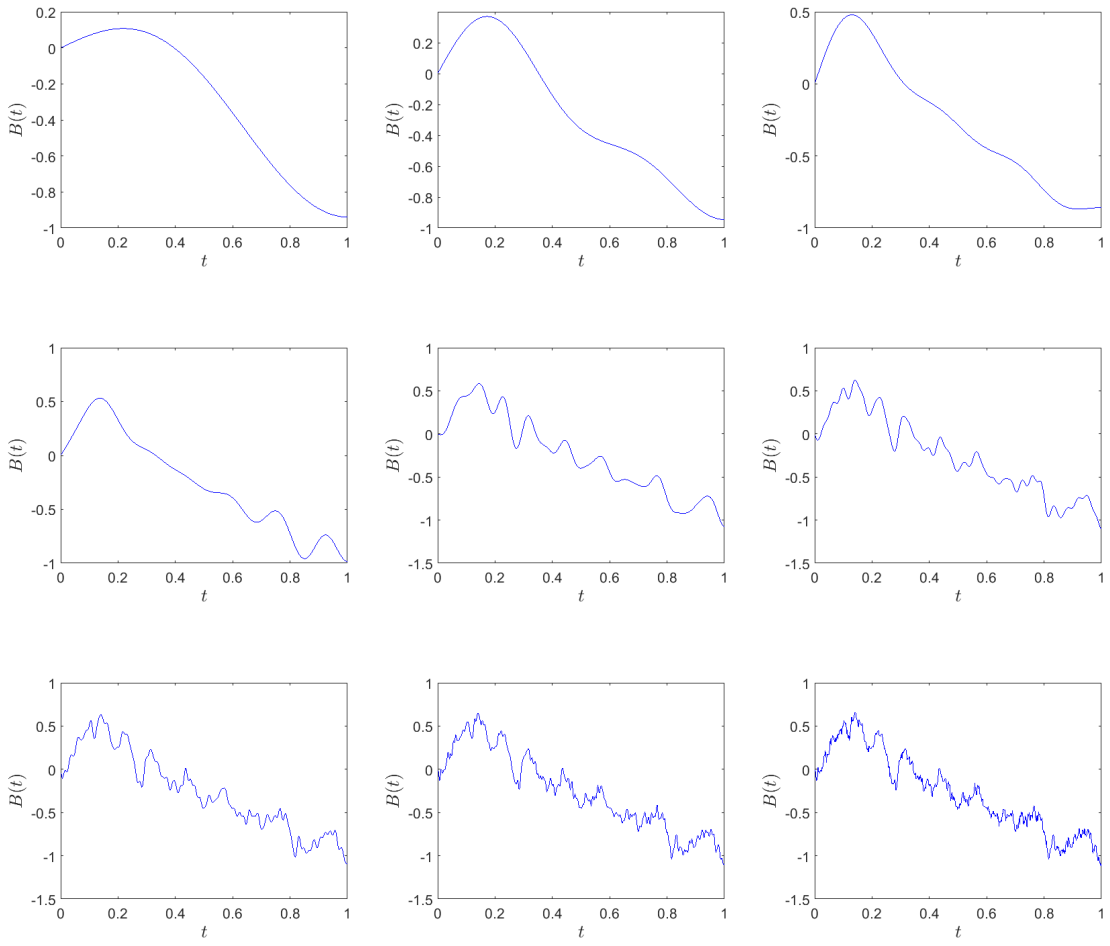

Figure 3: Simulations of a sample path of standard Brownian motion using the principal components construction, with the number of components considered given by 2^k, $k = 1, \ldots, 9$ respectively, when read from left to right, top to bottom.

that the asset returns are not explained by "random noise". Bouchaud and Potters applied this technique to stock returns from the S & P 500 and found that the largest eigenvalue of the correlation matrix between these returns was about 30 times greater than expected [3].

2 Geometric Brownian Motion

2.1 One Dimension

A stochastic process $S(t)$ is a geometric Brownian motion if $\log S(t)$ is Brownian motion with initial value $\log S(0)$, i.e. geometric Brownian motion is exponentiated Brownian motion. Hence, simulation of geometric Brownian motion is simply simulating exponentiated Brownian motion. Geometric Brownian motion is one of the simplest models of the price of a financial asset that is used in practice. Other methods for financial asset pricing exist, but geometric Brownian motion is used most commonly as it does not allow for negative prices, and leads to tractable equations for the pricing of derivatives[2, p.63–64].

For geometric Brownian motion, the relative changes $(S(t_{i+1} - S(t_i))/S(t_i)$ are independent for $t_1 < t_2 < \cdots < t_n$. We can compare this with independence of increments of Brownian motion, where each of the $B(t_{i+1}) - B(t_i)$ are independent.

Suppose B is standard Brownian motion and X satisfies:

$$dX(t) = \mu dt + \sigma dB(t),$$

so that $X \sim BM(\mu, \sigma^2)$. Set $S(t) = S(0)e^{X(t)} \equiv f(X(t))$. Then by the Itô formula, it follows that:

$$\begin{aligned} dS(t) &= f'(X(t))dX(t) + \tfrac{1}{2}\sigma^2 f''(X(t))dt \\ &= S(0)e^{X(t)}\big[\mu dt + \sigma dB(t)\big] + \tfrac{1}{2}\sigma^2 S(0)e^{X(t)}dt \\ &= S(t)\big[\mu + \tfrac{1}{2}\sigma^2\big]dt + S(t)\sigma dB(t) \end{aligned}$$

Notation 2.1.1. *We use the notation $X \sim GBM(\mu, \sigma^2)$ to indicate that S is a process of the type $\frac{dS(t)}{S(t)} = \mu dt + \sigma dB(t)$. We refer to μ in this stochastic differential equation as the drift parameter, though it is not the drift parameter of either $S(t)$ or $\log S(t)$. Similarly, we refer to the σ given in the SDE as the volatility parameter (N.B. the diffusion coefficient of $S(t)$ is $\sigma^2 S^2(t)$).*

If $S \sim GBM(\mu, \sigma^2)$ with initial value $S(0)$, then:

$$S(t) = S(0)\exp\big([\mu - \tfrac{1}{2}\sigma^2]t + \sigma B(t)\big), \tag{17}$$

or more generally, if $u < t$, then:

$$S(t) = S(u)\exp\big([\mu - \tfrac{1}{2}\sigma^2](t-u) + \sigma(B(t) - B(u))\big). \tag{18}$$

This suggests the following method for simulating values of S at $0 = t_0 < t_1 < \cdots < t_n$:

$$S(t_{i+1}) = S(t_i)e^{[\mu - \frac{1}{2}\sigma^2](t_{i+1}-t_i)+\sigma\sqrt{t_{i+1}-t_i}Z_{i+1}}, \quad i = 0, 1, \ldots, n-1, \tag{19}$$

where each of the Z_i are independent random variables drawn from the standard normal distribution. Note that this recursion is equivalent to exponentiating both sides of the recursion 10 used to generate Brownian motion under the random walk construction, and replacing μ by $\mu - \frac{1}{2}\sigma^2$.

This method is exact, on the index set, as the $(S(t_1), \ldots, S(t_n))$ produced has the joint distribution of $S \sim GBM(\mu, \sigma^2)$ at $(t_1, \ldots, t_n)$, though there is discretization error on the intervals between these time-points. We can adapt this method to incorporate time-dependent parameters by exponentiating both sides of (12).

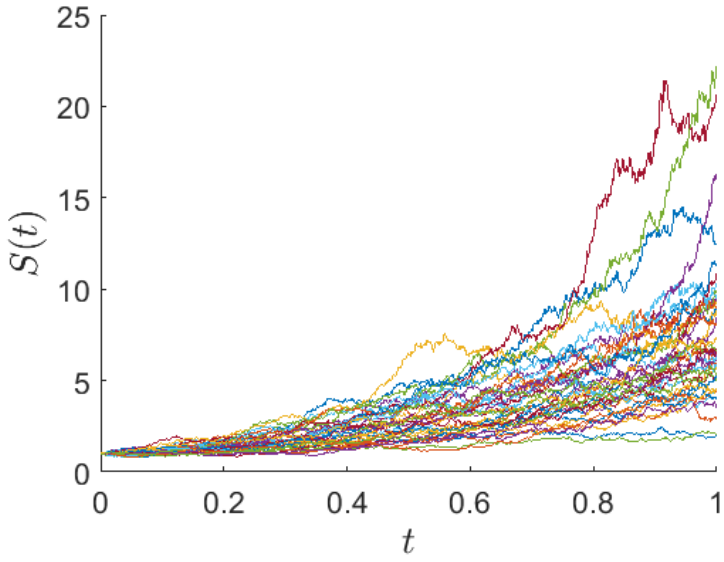

Figure 4: Forty sample paths of geometric Brownian motion under the random walk construction (19). $S(0) = 0$, $\mu = 2$, $\sigma = 0.5$, and each path is composed of 1000 timesteps. Hence, the expectation of the geometric Brownian motion at $t = 1$ in this case is actually $e^2 \approx 7.39$, by (5). This seems to agree with the distribution of values taken by the sample paths at $t = 1$.

From (18), it follows that if $S \sim GBM(\mu, \sigma^2)$, then the marginal distribution of $S(t)$ is that of the exponential of a Gaussian random variable; we call this a lognormal distribution.

Definition 2.1.2 (Lognormal Distribution)**.** *We write $Y \sim LN(\mu, \sigma^2)$ if the random variable Y has the distribution of* $\exp(\mu + \sigma Z)$*, with $Z \sim N(0, 1)$. Hence, the distribution*

of Y is given by

$$\begin{aligned}\mathbb{P}(Y \leq y) &= \mathbb{P}(Z \leq [\log(y) - \mu]/\sigma) \\ &= \Phi\left(\frac{\log(y) - \mu}{\sigma}\right)\end{aligned} \tag{20}$$

and the density of Y as

$$\begin{aligned}f_Y(y) &= \frac{\mathrm{d}}{\mathrm{d}x}\Phi\left(\frac{\log(y) - \mu}{\sigma}\right) \\ &= \frac{1}{y\sigma}\phi\left(\frac{\log(y) - \mu}{\sigma}\right)\end{aligned} \tag{21}$$

where Φ and ϕ are, respectively, the cumulative distribution function and probability density function of the standard normal distribution, $N(0, 1)$.

2.2 Multiple Dimensions

2.2.1 Construction

Consider a system of SDEs of the form

$$\frac{dS_i(t)}{S_i(t)} = \mu_i dt + \sigma_i dX_i(t), \quad i = 1, \ldots, d \tag{22}$$

where each X_i is one-dimensional standard Brownian motion and $\rho_{i,j} = \mathrm{Corr}(X_i(t), X_j(t))$. Define the $d \times d$ matrix $\boldsymbol{\Sigma}$ by $(\boldsymbol{\Sigma})_{i,j} = \sigma_i \sigma_j \rho_{i,j}$. Then $(\sigma_1 X_1, \ldots, \sigma_d X_d) \sim BM(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and we write $\boldsymbol{S} = (S_1, \ldots, S_d) \sim GBM(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)$. We abuse notation slightly and refer to $\boldsymbol{\mu}$ as the drift vector of S, and $\boldsymbol{\Sigma}$ as the covariance matrix of S.

Remark 2.2.1. *It is important to note that the drift vector of S is actually given by:*

$$(\mu_1 S_1(t), \ldots, \mu_d S_d(t))$$

and the covariances are given by

$$\mathrm{Cov}(S_i(t), S_j(t)) = S_i(0) S_j(0) e^{(\mu_i + \mu_j)t}(e^{\rho_{i,j}\sigma_i\sigma_j} - 1).$$

This follows from

$$S_i(t) = S_i(0) e^{(\mu - \frac{1}{2}\sigma_i^2)t + \sigma_i X_i(t)}, \quad i = 1, \ldots, d.$$

Let $\boldsymbol{A}$ be a $d \times d$ matrix such that $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{\Sigma}$ (i.e. $\boldsymbol{A}$ is a Cholesky factor of $\boldsymbol{\Sigma}$). Then $BM(0, \boldsymbol{\Sigma})$ can be represented as $\boldsymbol{A}B(t)$ with B standard Brownian motion. Applying this

to $(\sigma_1 X_1, \ldots, \sigma_d X_d)$ and rewriting the systems of SDEs (22) gives

$$\frac{dS_i(t)}{S_i(t)} = \mu_i dt + \sum_{j=1}^{d} a_i dB_i(t), \quad i = 1, \ldots, d \tag{23}$$

where a_i is the i^{th} row of A. More explicitly:

$$\frac{dS_i(t)}{S_i(t)} = \mu_i dt + \sum_{j=1}^{d} A_{i,j} dB_j(t) \tag{24}$$

This gives us an algorithm for simulating $GBM(\mu, \mathbf{\Sigma})$ at a set of timepoints $0 = t_0 < t_1 < \cdots < t_n$:

$$S_i(t_{k+1}) = S_i(t_k) \exp\left\{ (\mu_i - \tfrac{1}{2}\sigma_i^2)(t_{k+1} - t_k) + \sqrt{t_{k+1} - t_k} \sum_{j=1}^{d} A_{i,j} Z_{k+1,j} \right\}, \tag{25}$$

for each $k = 0, \ldots, n-1$ and $Z_k = (Z_{k,1}, \ldots, Z_{k,d}) \sim N(0, I)$ with $Z_1, \ldots, Z_n$ independent.

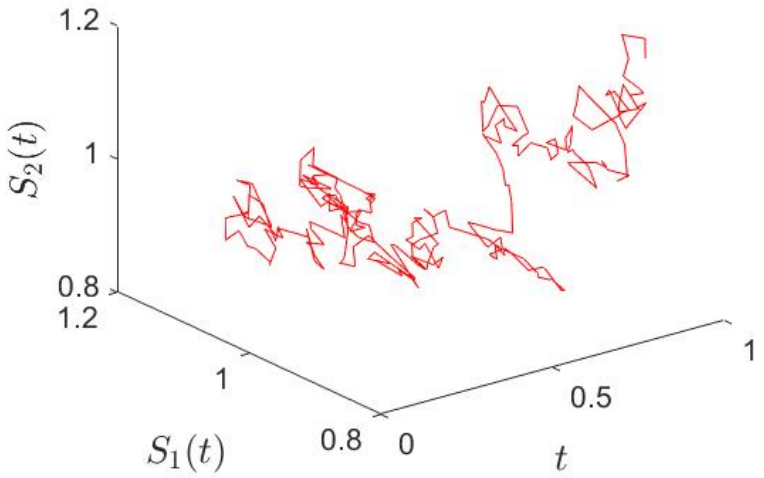

Figure 5: One realisation of 2-dimensional geometric Brownian motion under the random walk construction (25). $S(0) = (1, 1)$, $\mu = (0.3, 0.3)$, $\sigma = (0.3, 0.3)$, $\rho_{12} = 0.8$.

2.3 CEV Process

To conclude this section, we will briefly introduce the Constant Elasticity of Variance (CEV) process. The CEV model was proposed by Cox in 1975 to capture the property of non-constant volatility. The CEV process actually encompasses a class of processes, which are often used as alternatives to the simpler lognormal model, which assumes constant

volatility. The CEV process is described by the SDE

$$dS(t) = \mu S(t)dt + \sigma S(t)^{\beta/2} dB(t). \tag{26}$$

Dividing by $S(t)$ yields

$$\frac{dS(t)}{S(t)} = \mu dt + \sigma S(t)^{(\beta-2)/2} dB(t) \tag{27}$$

Note that in the case $\beta = 2$, the CEV process reduces to geometric Brownian motion. In fact, the dependence of the instantaneous volatility, $\sigma S(t)^{(\beta-2)/2}$, is what sets this apart from geometric Brownian motion; regression has shown that values of $\beta < 2$ fit stock prices better than geometric Brownian motion does [16]. Moreover, the $\beta < 2$ case implies a negative correlation between asset prices and their volatility.

In this project, we have used Milstein's method to simulate realisations of the CEV process. One can also simulate the CEV process by sampling from its transition density, as we will do with the CIR model in section 4.5.

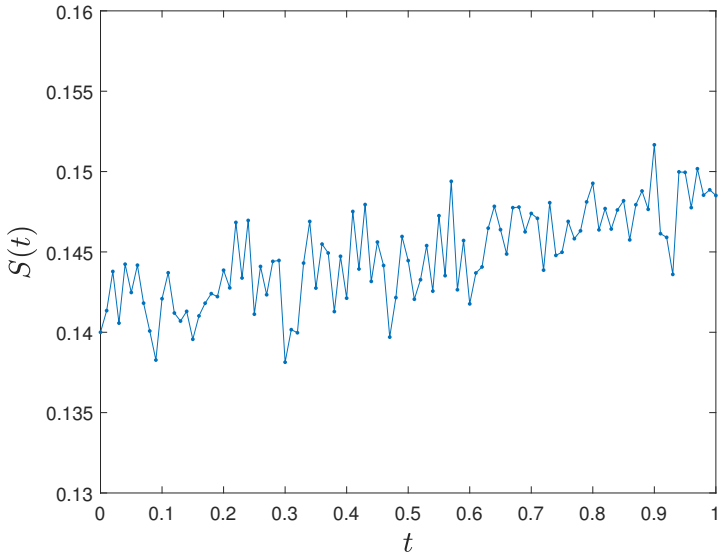

Figure 6: One realisation of the CEV process using the Milstein approximation. $\beta = 3/2$, $S(0) = 0.14$, $\mu = 5\%$, $\sigma = 1\%$.

3 Pricing Path Dependent Options Using Monte Carlo

Suppose that you are faced with the problem of finding the area enclosed by a closed curve in the plane. In general, this problem is nigh impossible to solve exactly. However, framing this problem in a different manner suggests a very simple way of achieving arbitrarily good approximations to the area: let the plane curve be drawn entirely on a square piece of paper with known area. We can then approximate the required area by "throwing darts" (i.e. taking independent draws from $U([0,1]^2)$, without loss of generality assuming the paper to be homomorphic to the unit square), and considering the ratio of "darts" that hit the interior of the curve to the total number of "darts" thrown. We can further generalise this by considering the scenario in which a d-dimensional form is suspended in a d-dimensional hypercube (i.e. the d-dimensional analogue of a square), where we are taking independent draws from $U([0,1]^d)$, and we take the ratio to find the d-dimensional volume of the shape. We have now lain the foundations of what we will call Monte Carlo simulation, where we extract information about an object by applying a procedure to it a large number of times and then examine the distribution of outcomes. One advantage of Monte Carlo simulations is that the rate of convergence of the error term to zero is independent of the dimensionality of the problem. Going back to the example of estimating the d-dimensional volume of the shape embedded in the d-hypercube, the error due to using N draws from the appropriate uniform distribution has order $O(N^{-1/2})$.

At its core, Monte Carlo simulation is an application of the Law of Large Numbers to evaluate the expectation of some function at a given point, $\mathbb{E}(f(S_T))$ [9, p.191]. By the Law of Large Numbers, if $(Y_n)_{n\geq 1}$ is a sequence of independent and identically distributed random variables, $\lim_{n\to\infty} \frac{1}{N}\sum_{j=1}^{N} Y_j = \mathbb{E}(Y_j)$. Moreover, by the Central Limit Theorem, this estimate follows the distribution $\mathbb{E}(Y_1) + \sqrt{\mathrm{Var}(Y_1)/n}N(0,1)$.

So far, the constructions we have provided have been given little in the way of practical context. In this chapter, we will present Monte Carlo-style methods for pricing certain types of exotic options.

We now turn our attention to path-dependent (or *exotic*) options: while European options have payoffs depending simply on $S(T)$, the value of the underlying asset at a fixed exercise date T, path-dependent options have payoffs that depend on the path that the value of the underlying takes. Many of these path-dependent options do not have a closed-form pricing formula, and so Monte Carlo methods have to be employed. This scheme takes the following form:

1. Generate sample paths of the underlying asset;
2. Evaluate the discounted payoff on each path;
3. Average the discounted payoffs.

There are many reasons as to why practitioners use these exotic options, and these reasons can be different for each type of option. For example, the Asian option, which will be discussed shortly, has a payoff that depends on the average of the past prices of a stock price. In this way, it can be understood that the Asian option decreases the losses that would incur if one were short a call option for a stock which suddenly had a price spike at expiry of the option. If the option were European, the option payoff would depend only on that uncharacteristically high stock price, whereas in the case of the Asian option, the payoff would take into account the previous prices could give you a "fairer" loss on your option.

The question now becomes that of which method we should use to price these exotic options. In practice, plenty of complicated pricing models exist, but for now we will stick to the "standard" assumption that price processes follow some sort of geometric Brownian motion. We will then endeavour to apply the Monte Carlo pricing scheme to simulations of this process. Throughout the project, we have chosen realistic values for the drift and diffusion terms of the geometric Brownian motion, but in practice, these would be calculated by regressions on previous stock price data, and other fundamental factors.

One of the advantages of Monte Carlo pricing is that the actual mathematics used is quite simple, especially when compared to the incredibly complicated formulae that are sometimes used to accomplish the same thing. Another is that correlations between assets can be easily incorporated, and the constructions for this have in fact already been introduced in section 2.2.1; this could be compared to models which do not so easily include correlations between assets, such as those modelling CDOs which many blame for the 2008 financial crisis. Finally, from a more practical perspective, one can get arbitrarily good approximations to the price by simply running more and more trials.

However, Monte Carlo pricing does not come without its flaws. Chief among them is the slow rate of convergence in low dimensions when compared to other methods of approximation, such as finite difference methods. However, this perceived inefficiency disappears when the dimensionality of the problem is increased, as the rate of convergence of Monte Carlo pricing is independent of dimension, whereas finite difference methods are not. Another issue with Monte Carlo pricing is the difficulty that one has with applying the technique to pricing American options: when an American option is priced for $t = 0$, the option must be priced for every point in (S, t)-space in order to check for whether the option should be exercised early. Monte Carlo pricing only prices the option now, and at the current value (i.e. one point in (S, t)-space). This subject is attacked in [6, pp.421–480], but will not be covered in this project.

MATLAB code to price the following options is given in A.4.

3.1 Variance Reduction Techniques

As one might expect, Monte Carlo simulation can be very computationally expensive, especially when one requires a tight confidence interval on an estimate. It is therefore imperative that the Monte Carlo estimate converges as rapidly as possible. There are many so-called variance reduction techniques, but in this project we will consider only the technique of antithetic variates.

3.1.1 Antithetic Variates

Before starting the simulation of stochastic paths for asset pricing in multiple dimensions, we consider a method to reduce the variance of the Monte Carlo estimator. It turns out that we can introduce some dependence within the sample set to reduce the variance of the estimator.

Definition 3.1.1. *A pair of real-valued random variables (Y, Y^*) is called an antithetic pair if Y and Y^* have the same distribution and are negatively correlated.*

Theorem 3.1.2 (Antithetic Estimator)**.** *Let N be even and let $(Y_1, Y_1^*), \dots, (Y_{N/2}, Y_{N/2}^*)$ be independent antithetic pairs of random variables, where each Y_k and Y_k^* is distributed as Y. The antithetic estimator*

$$\hat{\ell}^{(a)} = \frac{1}{N} \sum_{k=1}^{N/2} \{Y_k + Y_k^*\} \tag{28}$$

is an unbiased estimator of $\ell = \mathbb{E}[Y]$, with variance

$$\begin{aligned} \mathrm{Var}(\hat{\ell}^{(a)}) &= \tfrac{N/2}{N^2}(\mathrm{Var}(Y) + \mathrm{Var}(Y^*) + 2\,\mathrm{Cov}(Y, Y^*)) \\ &= (\mathrm{Var}(Y) + \mathrm{Cov}(Y, Y^*))/N \\ &= \tfrac{\mathrm{Var}(Y)}{N}(1 + \rho_{Y,Y^*}) \end{aligned}$$

where ρ_{Y,Y^*} is the correlation of Y and Y^*.

Note that (28) is the sample mean of the $N/2$ independent random variables $\{(Y_k + Y_k^*)/2\}$. The variance of the Crude Monte Carlo estimator $\hat{\ell} = \frac{1}{N}\sum_{k=1}^{N} Y_i$ is $Var(Y)/N$, and so this theorem says that using antithetic variables leads to a reduction in variance of the estimator by a factor of $(1 + \rho_{Y,Y^*})$.

Remark 3.1.3. *We now consider the application of antithetic variance reduction to the normal random variable case. Suppose that $Y = H(Z)$ where $Z \sim N(0, I)$. Then, we can write $Y = h(U)$, with $h(u) = H(\Phi^{-1}(u_1), \Phi^{-1}(u_2), \dots)$. Taking $U^* = 1 - U$ gives $Z^* = (\Phi^{-1}(U_1^*), \Phi^{-1}(U_2^*), \dots) = -Z$ so that $Y^* = H(-Z)$ forms an antithetic pair provided that Y and Y^* are negatively correlated, which is the case if H is monotone in each of its components.*

3.2 Asian Options: Arithmetic Average with Discrete Monitoring

An Asian option is an option priced on a time average of the underlying asset. Asian calls and puts have payoffs $(\bar{S} - K)^+$ and $(K - \bar{S})^+$ respectively, where the strike price K is a constant and

$$\bar{S} = \frac{1}{n} \sum_{i=1}^{n} S(t_i) \tag{29}$$

is the average price of the underlying asset over the discrete set of monitoring dates $t_1, \ldots, t_n$. In general, there do not exist exact formulae for the prices of these options.

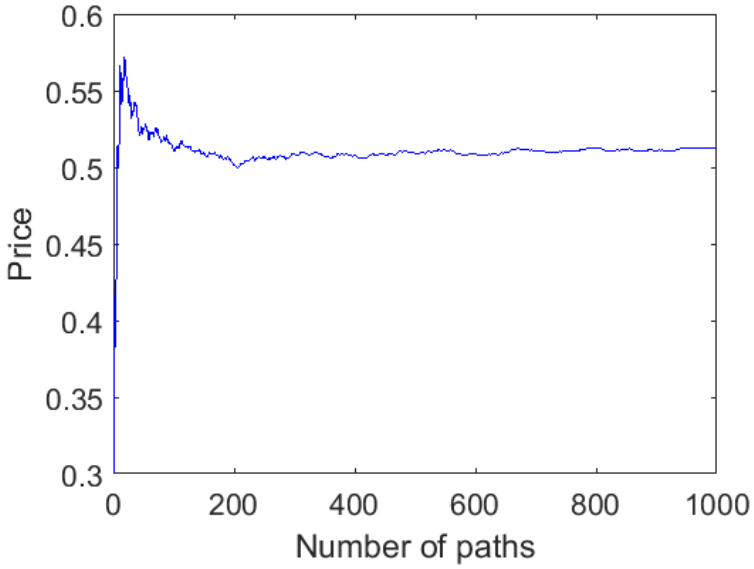

Figure 7: The convergence of the Monte Carlo pricing method on an Asian call option, with strike price $K = 0.5$, $\mu = 0.5$, $\sigma = 0.25$ and 252 timesteps. The convergence is of order $O(\sqrt{n})$, where n is the number of sample paths considered.

3.3 Lookback Options

Lookback options are options on the extremal price of the underlying asset attained over the life of the option. Let T denote the expiration of the option and $[0, T]$ the lookback period. We denote the maximum and minimum values of the asset price over $[0, t]$, with $t \leq T$, by $M_t = \max_{0 \leq \xi \leq t} S_\xi$ and $m_t = \min_{0 \leq \xi \leq t} S_\xi$.

The floating lookback call, so called because the strike price 'floats' depending on the path of the option, gives the holder the right to buy the underlying asset at time T for the lowest realised price over the lookback period (i.e. for m_t). Similarly, the floating lookback put gives the holder the right to sell the underlying asset for the highest realised price (i.e. for M_t).

Hence, floating lookback calls and puts expiring at T have payoffs

$$(S(T) - m_t)^+ \quad \text{and} \quad (M_t - S(T))^+$$

respectively.

Note that $m_T \leq S_T \leq M_T$ so the option is always in-the-money (here, we are taking the at-the-money case when the payoff is zero to be essentially in-the-money), so the holder will always exercise the option. Take $\tau = T - t$ and define the functions:

$$\delta_\pm(\tau, s) = \frac{1}{\sigma\sqrt{\tau}}\left[\log s + (r \pm \tfrac{1}{2}\sigma^2)\tau\right].$$

Then the lookback put option has value process $V(t) = v(t, x, y)$, where $x = S(t)$, $y = Y(t) = \max_{0\leq\xi\leq t} S(\xi)$. We have an explicit formula [17, p.314–320] for v:

$$\begin{aligned} v(t,x,y) = (1 + \frac{\sigma^2}{2r})x\Phi(\delta_+(\tau, \frac{x}{y})) + e^{-r\tau}y\Phi(-\delta_-(\tau, \frac{x}{y})) \\ - \frac{\sigma^2}{2r}e^{-r\tau}(\frac{y}{x})^{2r/\sigma^2}x\Phi(-\delta_-(\tau, \frac{x}{y})) - x \end{aligned} \tag{30}$$

where $\Phi(\cdot)$ is the c.d.f. of the standard normal distribution.

Figure 8 shows the systematic underpricing of a continuously monitored lookback option when it is instead only discretely monitored. As the time between monitoring points becomes smaller (or equivalently, in our case, the number of monitoring points increases), the discretely monitored lookback option price tends to the continuously monitored option price, as we would expect. In practice, continuous monitoring of a stock price is impossible, so the "exact" lookback option pricing formula (30) is useful only to establish an upper bound for the actual price.

Now, we use the tools built up in the chapter on simulating geometric Brownian motion in order to price more complicated exotic options. All of the options considered are of the European style: they can only be exercised at expiry.

3.4 Spread Options

We now consider the call option on the spread of the prices of two assets. Denote the price processes of the two underlying assets by $S_1(t)$ and $S_2(t)$, and suppose the option has expiry T and strike price K. Then, the payoff for this call spread option is given by

$$\big([S_1(T) - S_2(T)] - K\big)^+ \tag{31}$$

Some MATLAB code to illustrate the application of Monte Carlo pricing to this op-

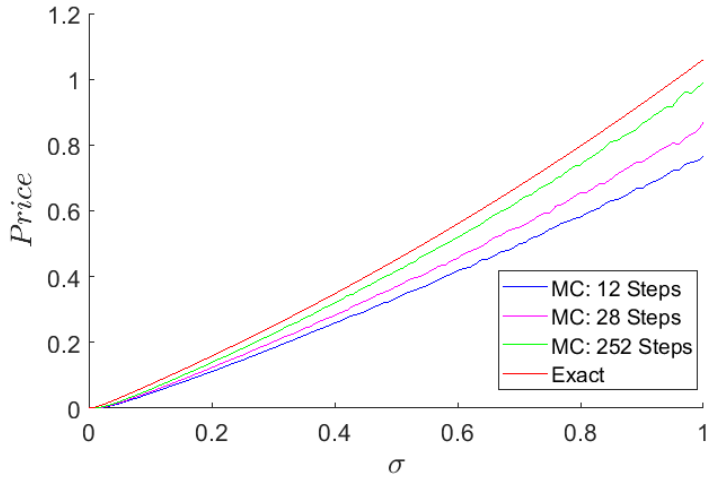

Figure 8: Various pricing methods of a lookback option with $\mu = 0.05$, $r = 0.02$, $S(0) = 1$, and varying σ. The Monte Carlo pricings each take the average of $10,000$ simulations. The figure shows the undervaluation of a continuously-monitored lookback option when it is not continuously monitored, due to the almost-sure event of a higher maximum being attained outside the discrete set of monitored points than within the set. Note that as the number of points in the monitored set increases, the Monte Carlo price curve approaches the exact price curve.

tion can be found in A.4.6. For the implementation, we used the simulation of multi-dimensional geometric Brownian motion, as described in (25). In reality, it is highly unlikely that any two stock price processes have zero covariance, so this program allows the covariance of the two asset to be considered as well. The typical Monte Carlo procedure then follows, and the option is priced.

3.5 Barrier Option

We consider a discretely-monitored two-asset down-and-in barrier put option. As before, let the price processes of the underlying assets be $S_1(t)$ and $S_2(t)$, and suppose the option has expiry T, strike K, and barrier b. Moreover, suppose that the assets are monitored at a set of timepoints $\{0 = t_0, t_1, \ldots, t_n = T\}$. Then, the payoff of the two-asset down-and-in barrier put is given by

$$\mathbb{1}_{\left\{\min\limits_{i=1,\ldots,n} S_2(t_i) < b\right\}} (K - S_1(T))^+ \tag{32}$$

Intuitively, this represents a payoff that is zero until asset S_2 hits the barrier b, then has the same payoff as a European put option on S_1.

Some MATLAB code to illustrate the Monte Carlo pricing of this option can be found in A.4.7. The methodology for pricing this option on two assets was very similar to that used in the pricing of the spread option.

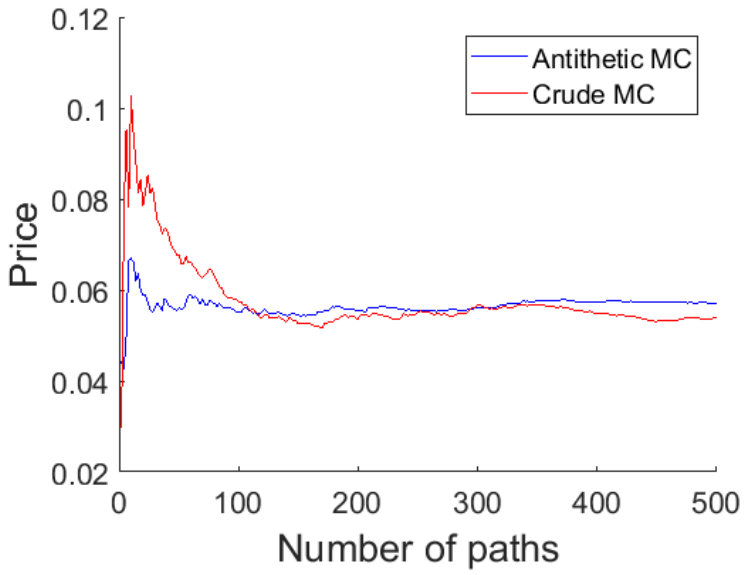

Figure 9: A graph showing the convergence of the crude and antithetic Monte Carlo pricing of a lookback call option. Note that the antithetic pricing algorithm seems to exhibit faster convergence.

4 Short Rate Models

4.1 Gaussian Short Rate Models

Up until this point, we have only considered a continuously compounded constant interest rate. Now, we generalise this model to a stochastic setting. An investment earning interest at a rate $r(u)$ at time u grows from a value of 1 at time 0 to a value of

$$\beta(t) = \exp\left(\int_0^t r(u)du\right)$$

at time t. This quantity is stochastic, but can still be treated as the numeraire for risk-neutral pricing.

Hence, the price at time 0 of a derivative security that pays X at time T is the expectation of $X/\beta(T)$:

$$\mathbb{E}\left[\exp\left(-\int_0^T r(u)du\right)X\right]. \tag{33}$$

In particular, the time-0 discounted price of a bond paying 1 at T is

$$B(0,T) = \mathbb{E}\left[\exp\left(-\int_0^T r(u)du\right)\right]. \tag{34}$$

We now consider different models for the behaviour of the stochastic interest rate $r(t)$:

the Vasicek model[20], and the Ho-Lee model[8]. We do not yet allow σ to be stochastic, but rather leave it constant (later, we will generalise further to time-varying $\sigma(t)$).

4.2 Vasicek Model

Vasicek proposed modelling the short rate by an Ornstein-Uhlenbeck process:

$$dr(t) = \alpha(b - r(t))dt + \sigma dB(t) \tag{35}$$

Here, $B(t)$ is standard Brownian motion, and $\alpha, b, \sigma > 0$ are constants. Note that the drift term is positive if $r(t) < b$, and negative if $r(t) > b$, so $r(t)$ is pulled towards b (a property known as mean-reversion). In this sense, b can be thought of as a long-run interest rate level, and α as the speed at which $r(t)$ is pulled towards b.

4.2.1 Simulation

The general solution of the SDE (35) is given by (compare this with the solution general solution (41)):

$$r(t) = e^{-\alpha t}r(0) + \alpha \int_0^t e^{-\alpha(t-s)}b(s)ds + \sigma \int_0^t e^{-\alpha(t-s)}dB(s). \tag{36}$$

Similarly, for any $0 < u < t$:

$$r(t) = e^{-\alpha(t-u)}r(u) + \alpha \int_u^t e^{-\alpha(t-s)}b(s)ds + \sigma \int_u^t e^{-\alpha(t-s)}dB(s). \tag{37}$$

Hence, given some $r(u)$,

$$r(t) \sim N\left(e^{-\alpha(t-u)}r(u) + \alpha \int_u^t e^{-\alpha(t-s)}b(s)ds,\ \frac{\sigma^2}{2\alpha}(1 - e^{-2\alpha(t-u)})\right).$$

To simulate r at times $0 = t_0 < t_1 < \cdots < t_n$, we may set

$$r(t_{i+1}) = e^{-\alpha(t_{i+1}-t_i)}r(t_i) + \mu(t_i, t_{i+1}) + \sigma_r(t_i, t_{i+1})Z_{i+1} \tag{38}$$

where

$$\mu(u,t) = \alpha \int_0^t e^{-\alpha(t-s)}b(s)ds, \quad \sigma_r^2(u,t) = \frac{\sigma^2}{2\alpha}(1 - e^{-2\alpha(t-u)})$$

and $Z_1, \ldots, Z_n$ are independent random variables drawn from $N(0,1)$.

Remark 4.2.1. *By comparison with the random walk construction of Brownian motion, it is clear that this simulation of the Vasicek model is exact only on the set of timepoints* $\{0 = t_0 < t_1 < \cdots < t_n\}$.

Using this simulation, we can find a Monte Carlo estimate for the time-0 discounted price of a bond paying 1 at time T. We repeatedly generate sample paths of the short rate, and approximate the integral in (34) by the trapezium rule. We calculate the discounted value on this sample path as in the equation, and then repeat this whole sequence a large number of times. Figure 10 shows the distribution of the prices for a large number of simulations of a particular bond. Below is the MATLAB output, alongside figure 10, of the code in A.5.1. The regression performed by the *distfit* function fit the simulations to the normal distribution with parameters $\mu = 0.903134$ and $\sigma = 0.00166463$. The intervals displayed alongside these values are their respective 95% confidence intervals.

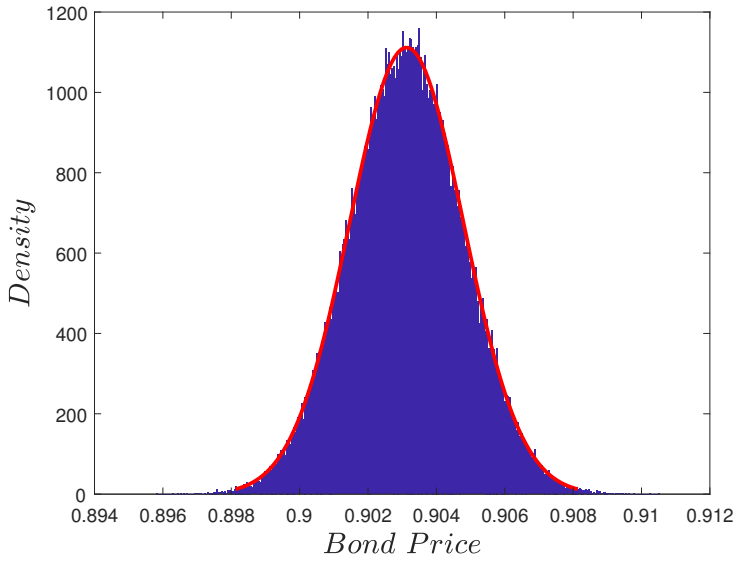

Figure 10: The time-0 discounted Monte Carlo pricing of a bond paying 1 at expiry, under the Vasicek model. The time interval is $[0, 1]$, there are 1000 points sampled for each path, the long-run interest rate is $b = 0.1$, the "speed" is $\alpha = 10$, the variance is $\sigma = 0.02$, and $r(0) = 0.12$. There were 10^5 trials in the Monte Carlo estimation.

```
Normal distribution
 mu =    0.903134    [0.903124, 0.903145]
sigma = 0.00166463   [0.00165737, 0.00167196]
```

4.3 Ho-Lee Model in Continuous Time

In 1986, Ho and Lee proposed modelling the short rate by a different SDE:

$$dr(t) = g(t)dt + \sigma dB(t) \tag{39}$$

where g is a deterministic function of t. This model generates a symmetric distribution for future rates. Hence, negative rates are possible, as in the Vasicek model. However, unlike the Vasicek model, there is no mean reversion. This is one reason that the Vasicek model

is often preferred to the Ho-Lee model, as there is significant evidence for mean reversion in past stock market data, and the Vasicek model can capture this.

Simulation of the Ho-Lee model will not be discussed in this project, but can be performed in similar fashion to simulation of the Vasicek model.

4.4 The General Gaussian Markov Process

Note that both of the models that we have considered define Gaussian processes. In fact, both the Vasicek and the Ho-Lee models are special cases of the more general Gaussian Markov process:

$$dr(t) = [g(t) + h(t)r(t)]dt + \sigma(t)dB(t) \tag{40}$$

where g, h, σ are all deterministic functions of t.

Remark 4.4.1. *The equation (40) reduces to the Vasicek SDE (35) on setting $g(t) = \alpha b$, $h(t) = -\alpha$, and fixing $\sigma(t)$ to be constant.*
Similarly, (40) reduces to the Ho-Lee SDE (39) on setting $h(t) = 0$ and holding $\sigma(t)$ constant.

The SDE (40) has solution

$$r(t) = e^{H(t)}r(0) + \int_0^t e^{H(t)-H(s)}g(s)ds + \int_0^t e^{H(t)-H(s)}\sigma(s)dB(s) \tag{41}$$

with $H(t) = \int_0^t h(s)ds$.

One could choose functions for g, h, σ and then discretise the SDE, and produce simulations for this process in similar fashion as we have done with the Vasicek model.

4.5 Square-Root Diffusions: CIR Model

Square-root diffusions take the form

$$dr(t) = \alpha(b - r(t))dt + \sigma\sqrt{r(t)}dB(t) \tag{42}$$

We restrict ourselves to the case where $\alpha, b > 0$. Note that if $r(0) > 0$ then $r(t) \geq 0$, and if $2\alpha b \geq \sigma^2$ then $r(t) > 0$.

Cox, Ingersoll, and Ross (1985) proposed what became known as the CIR model: the square-root diffusion as a model for the short rate. As in the Vasicek model, $r(t)$ is pulled towards b at a speed governed by α. The major difference is that the diffusion term $\sigma\sqrt{r(t)} \to 0$, from above, as $r(t) \to 0$. Often the existence of negative interest rates is incompatible with other market models, so the CIR model can be employed in those circumstances where negative interest rates are disallowed.

Applying Theorem 1.0.1 to (42) yields the following recursion for simulation of $r(t)$ at a set of time points $0 = t_0 < t_1 < \cdots < t_n = T$ with fixed $r(0) = r_0$:

$$r(t_{i+1}) = r(t_i) + \alpha(b - r(t))[t_{i+1} - t_i] + \sqrt{r(t_i)^+}\sqrt{t_{i+1} - t_i}Z_{i+1} \tag{43}$$

for $i = 0, 1, \ldots, n-1$, where each of the $Z_1, \ldots, Z_n$ are drawn independently from the standard normal distribution.

However, this introduces some significant error if $r(t)$ does ever become negative. For this reason, we can consider the much more involved method of sampling directly from the transition density [6, p122]: given some $r(u)$ with $u < t$, $r(t)$ has a scaled non-central chi-squared distribution. In particular,

$$r(u) \sim \frac{\sigma^2(1 - e^{-\alpha(t-u)})}{4\alpha}\chi'^2_d\left(\frac{4\alpha e^{-\alpha(t-u)}}{\sigma^2(1 - e^{\alpha(t-u)})}r(u)\right), \quad d = \frac{4b\alpha}{\sigma^2}. \tag{44}$$

Recall that $\chi'^2_\nu(\lambda) = (Z + \sqrt{\lambda})^2 + \chi^2_{\nu-1}$ where $Z \sim N(0, 1)$ and so we can easily implement the simulation of the CIR model using the above method, which can be found in A.5.2.

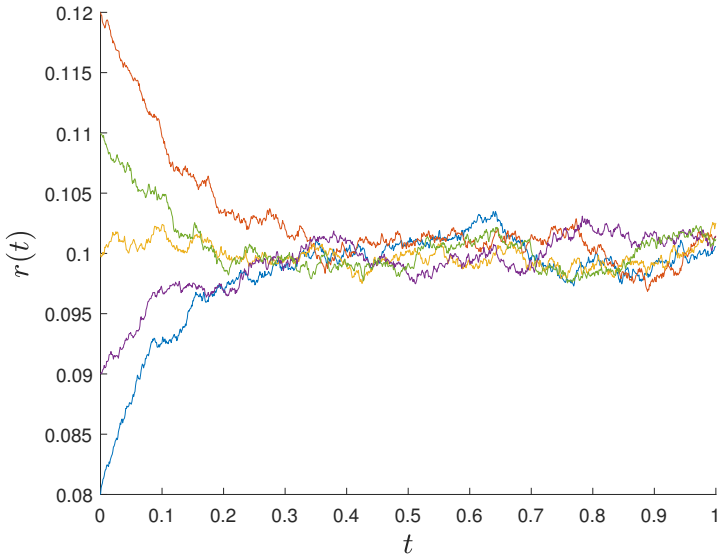

Figure 11: Five sample paths of the CIR process with the same parameters as the Vasicek simulations in figure 10: the time interval is $[0, 1]$, there are 1000 points sampled for each path, the long-run interest rate is $b = 0.1$, the "speed" is $\alpha = 10$, the variance is $\sigma = 0.02$. We have including one sample path for each of $r(0) = 0.08, 0.09, 0.1, 0.11, 0.12$. The simulation was produced using the method described by (44).

5 Processes with Jumps

A typical assumption in option pricing models in that the sample paths of the underlying asset are continuous. Empirically, stock returns tend to feature fat tails, which is inconsistent with the typical Black-Scholes assumption. In this section, we cast off this assumption and consider processes with discontinuities, then use these discontinuous sample paths in our Monte Carlo pricing.

The convention we use is that the jump processes, at their discontinuities, are continuous from the right, and the limits from the left exist. This condition is known as RCLL (Right Continuous with Left Limits).

Intuitively, we can think of jumps in the value of the underlying as idiosyncratic shocks that affect a particular company but not the whole market [14].

Merton's jump-diffusion model SDE is given as

$$\frac{dS(t)}{S(t-)} = \mu dt + \sigma dB(t) + dJ(t) \tag{45}$$

where J is a process independent of B, with piecewise constant sample paths. In particular, $J(t) = \sum_{j=1}^{N(t)}(Y_j - 1)$ where the $Y_1, Y_2, \ldots$ are random variables and $N(t)$ is a counting process: there exist random arrival times $0 < \tau_1 < \tau_2 < \ldots$ and $N(t) = \sup\{n \mid \tau_n \leq t\}$ counts the number of arrivals in $[0, t]$. $dJ(t)$ is the size of the jump at time t, and has size $(Y_j - 1)$ if $t = \tau_j$, or 0 otherwise.

Remark 5.0.1. *The convention is that the process is RCLL, so we write:*

$$S(t) = \lim_{u \downarrow t} S(u), \qquad S(t-) = \lim_{u \uparrow t} S(u)$$

Hence, we can rewrite the jump diffusion SDE (45) as

$$dS(t) = \mu S(t-)dt + \sigma S(t-)dB(t) + S(t-)dJ(t) \tag{46}$$

As we would expect, this process has the Markov property, as the increment $dS(t)$ in S at t depends on the value of S just before t. This jump is

$$S(\tau_j) - S(\tau_j-) = S(\tau_j-)[J(\tau_j) - J(\tau_j-)] = S(\tau_j-)(Y_j - 1),$$

so it follows that

$$S(\tau_j) = S(\tau_j-)Y_j.$$

We now make some assumptions for the distributions of the jump times and the sizes of each jump, and concentrate for the meantime only on the jump term in the jump-diffusion SDE.

5.1 Simple Poisson Process

One logical assumption would be that the number of jumps in a given time interval follows a Poisson distribution. In reality, this may not be the case as the "shocks" might not occur independently of one another at a constant rate (e.g. shocks in stock prices may be more concentrated towards periods in which there are earnings announcements etc.). Assuming $N(t)$ follows the Poisson distribution, we have:

$$\mathbb{P}(N(t) - N(s) = k) = \frac{\lambda^k (t-s)^k}{k!} e^{-\lambda(t-s)}. \tag{47}$$

By considering the exponential power series, it follows that

$$\sum_{k=1}^{\infty} \mathbb{P}(N(t) - N(s) = k) = 1$$

as expected. Moreover, $\mathbb{E}[N(t) - N(s)] = \lambda(t-s)$. Similarly, $\text{Var}(N(t) - N(s)) = \lambda(t-s)$.

The times between jumps (i.e. the interarrival times), $\tau_1, \tau_2, \ldots$, then follow an exponential distribution.

This leads to the idea of the simple Poisson process, which is simply a finite set of jumps, with interarrival times following an exponential distribution, and each jump being a unit in the positive direction.

5.1.1 Simulation

As might be expected, the simulation here is quite simple, especially with MATLAB's built in **exprand** function, which draws from the exponential distribution. First, we fix the end of the time interval, T, over which we will be simulating, as well as the parameter of the exponential distribution. Then, we draw from the exponential distribution repeatedly to generate the interarrival times, until the sum of these values exceeds T (we discard the final value that brings this total over T). Finally, for each of the indexed time points, the value of the process at this point is simply the number of jumps up to this point (recall the RCLL convention).

5.2 Compound Poisson Process

We can now make the size of the jumps non-constant. Let $N(t)$ be a Poisson process with intensity λ, and let $Y_1, Y_2, \ldots$ be a sequence of identical and independently distributed

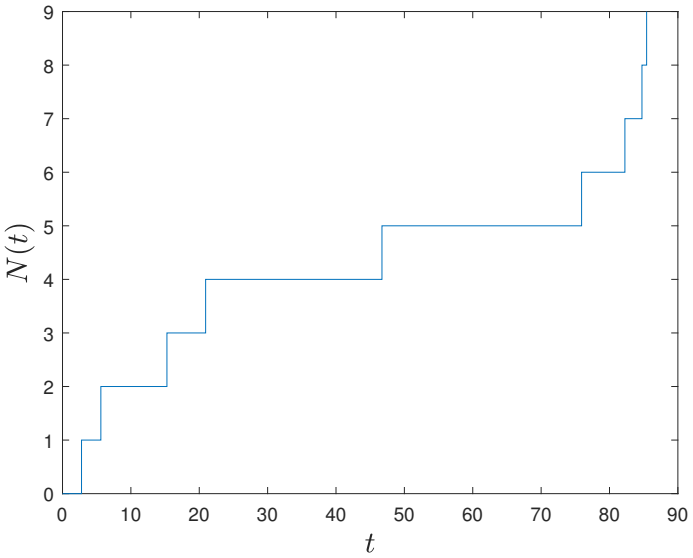

Figure 12: One realisation of the Pure Poisson Process, where each jump has size 1. The interval is $[0, 90]$, and the parameter of the exponential distribution is 10.

random variables with mean $\beta = \mathbb{E}[Y_i]$. Now define the compound Poisson process

$$Q(t) = \sum_{i=1}^{N(t)} Y_i, \quad t \geq 0. \tag{48}$$

It then follows that

$$\begin{aligned}
\mathbb{E}[Q(t)] &= \sum_{k=0}^{\infty} \mathbb{E}\left[\sum_{i=1}^{k} Y_i | N(t) = k\right] \mathbb{P}(N(t) = k) \\
&= \sum_{k=0}^{\infty} \beta k \frac{(\lambda t)^k}{k!} e^{-\lambda t} \\
&= \beta \lambda t e^{-\lambda t} \sum_{k=1}^{\infty} \frac{(\lambda t)^{k-1}}{(k-1)!} \\
&= \beta \lambda t
\end{aligned}$$

This is consistent with our intuition: there are on average λt jumps in $[0, t]$, the average jump size is β, and all of these random variables are independent.

It is convenient to model the size of the jumps, Y_i, by a lognormal distribution [14]:

$$Y_i \sim LN(a, b^2).$$

Then, the product of jumps is also lognormal:

$$\prod_{j=1}^{n} Y_j \sim LN(an, b^2 n).$$

From this assumption, we can easily simulate the compound Poisson process through generating exponential and lognormal random variables.

5.2.1 Simulation

We proceed exactly as we did with the simulation of the simple Poisson process, but now for each jump time, we draw from some lognormal distribution with given parameters, and let this be the size of the respective jump.

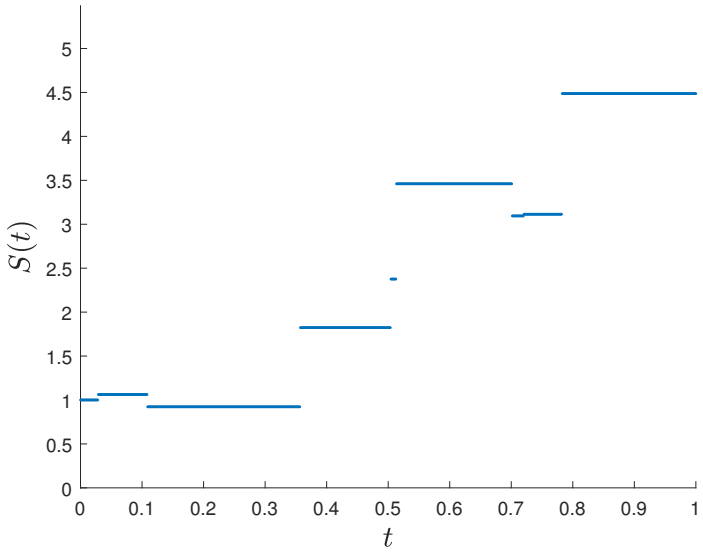

Figure 13: One realisation of a Compound Poisson Process over the interval $[0, 1]$, with $N(0) = 1$. For the jump times, the exponential distribution has rate $\lambda = 0.1$. For the jump sizes, the lognormal distribution has parameters $\mu_{jump} = 0.1$ and $\sigma_{jump} = 0.2$.

5.3 Merton's Jump-Diffusion Model

The final stage in the building of Merton's jump-diffusion model is to include the geometric Brownian motion component. The model we have constructed is then consistent with the SDE (45).

5.3.1 Simulation

As the jump terms are multiplicative, we can create the simulation by simply generating the appropriate sample paths of geometric Brownian motion and of the jump process, and then taking their pointwise product. This is executed by the MATLAB code in A.6.3.

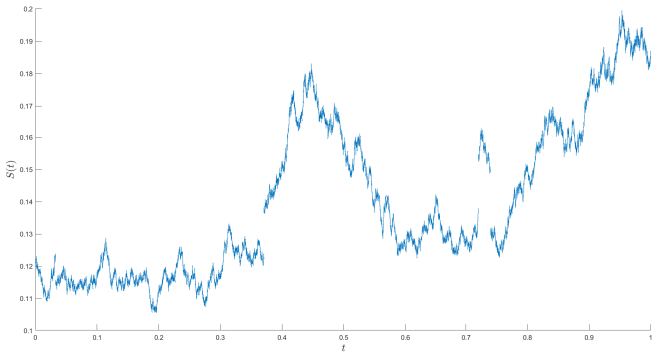

Figure 14: One realisation of the Merton Jump-Diffusion Model over the interval $[0, 1]$, with 10^6 timepoints, $T = 1$, $S(0) = 0.12$. For the jump times, the exponential distribution has rate $\lambda = 0.15$. For the jump sizes, the lognormal distribution has parameters $\mu_{jump} = 0.05$ and $\sigma_{jump} = 0.08$, and the geometric Brownian motion has parameters $\mu = 0.1$, $\sigma = 0.4$.

We can now recreate our Monte Carlo lookback option pricing, but under the assumptions of the Merton jump-diffusion model. The MATLAB code used to do this can be found in A.6.4. As can be seen in figure 15, the density plot exhibits a fatter tail than might be expected under the typical geometric Brownian motion model. This fat tail is more in line with empirical data. The creation of the plot shows that we can not only extract data such as the expected return, but infer the distribution of the returns, which would be useful if we were calculating metrics such as the Value at Risk (VaR).

Also note how the option price increases exponentially in figure 16 as the sigma of the lognormal distribution describing the jump size increases (so the option price would increase linearly as the standard deviation of the lognormal distribution were increased linearly). It is instructive to compare this to the changes in option price described by figure 8.

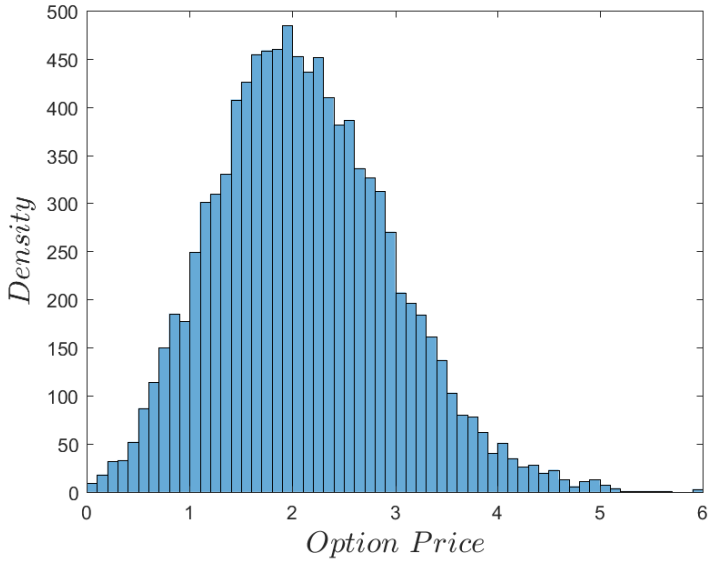

Figure 15: A density plot of the option prices in each simulation (in this case, prices and payoffs are the same) of the jump-diffusion model. Note the fat right-hand tail.

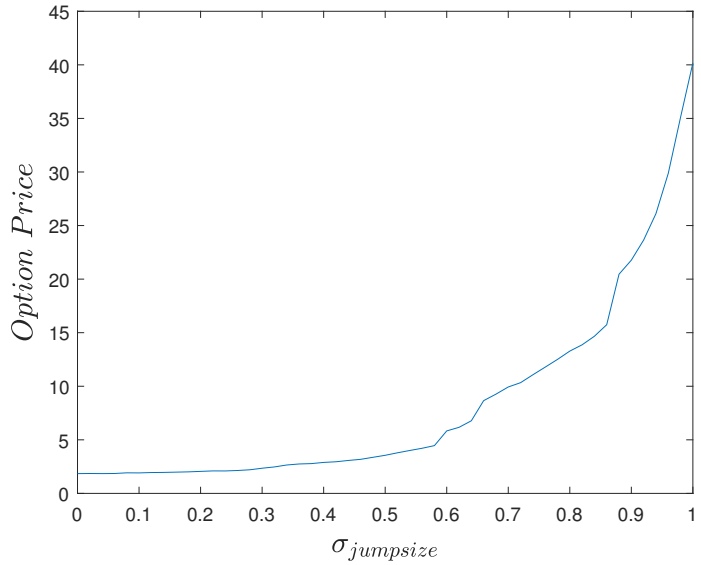

Figure 16: A plot to show the changes in the Monte Carlo option pricing as the sigma of the lognormal distribution describing the jump sizes varies. In this case, the time interval is $[0, 1]$, there are 100 timepoints, the time between jumps follows the exponential distribution with parameter $\lambda = 0.15$, the jump sizes are lognormally distributed with parameter $\mu_{jumps} = 0.4$, the geometric Brownian motion has parameters $\mu = 0.1$ and $\sigma = 0.4$, and $S(0) = 1$.

6 Conclusion

Over the duration of this project, we have successfully built up a versatile framework for simulating stochastic processes, including some that are often used in the modelling of stock prices and interest rates. In addition to this, Monte Carlo methods have been applied to extract data from various processes that we constructed. This includes the pricing of exotic options on an asset with its price process following a particular model. It would be possible to extend the work already done to other forms of exotic options; this could be done simly by changing the payoff functions in some of the MATLAB code that has been written.

Moreover, it would not be much work to extend what has already been covered by this project to simulate returns generated by financial instruments such as index funds (or any instrument where the price process is a deterministic function of the underlying stochastic price processes), given the processes followed by the underlying assets.

Another natural extension to the work presented in this project would be to introduce more variance reduction techniques (e.g. control variates, importance sampling, and Latin hypercube sampling) and implement these in the Monte Carlo option pricing algorithms that have been built. The theory behind variance reduction is covered by [12] in prodigious detail. This is also covered in [6].

In practice, it would be necessary to find the nature and parameters of the process before simulation. This would be performed through a robust regression and backtesting scheme, and is beyond the scope of this project; two relevant books on this would be [18] and [5]. In a similar vein, it is important to remember that all of the theory developed in this project has been been under the assumption that a given stochastic process is a perfect model for the actual process being considered. It is all too easy to gravitate to one of these models without the proper justification, and to forget that, just because a trend has been followed up until a certain point in time, there is no guarantee that this particular trend will continue.

A MATLAB Code

A.1 Brownian Motion

A.1.1 Random Walk Construction of Brownian Motion

```
close all
clear variables

T=1; %the length of the time interval
N=1000; %the number of timesteps within the interval
dt=T/N; %the length of each timestep
mu=0; sigma=1; %these parameters can be changed for general Brownian motion.

dB=zeros(1,N); %the change in Brownian motion at each timestep
B=zeros(1,N); %the value Brownian motion takes at each timestep

%generating the values at the first time step
dB(1)=mu*dt+sigma*sqrt(dt)*randn;
B(1)=dB(1);

for i=2:N
    dB(i)=mu*dt+sigma*sqrt(dt)*randn;
    B(i)=B(i-1)+dB(i); %the recurrence relation
end

figure
plot(0:dt:T, [0,B], 'b');
set(gca,'FontSize',16);
xlabel('$t$','FontSize',20,'interpreter','latex');
ylabel('$B(t)$','FontSize',20,'interpreter','latex');
```

A.1.2 Brownian Bridge Construction of Brownian Motion

```
close all
clear variables

mu=1; sigma=0.5;
T=1; m=4;
N=2^m; dt=T/N;
t=N^(-1):N^(-1):T;
B=zeros(1,N+1);
h=N; j_max=1;

```

```
B(h+1)=sqrt(T)*normrnd(mu*t(m),t(m)); %generating the rightmost point

for i=1:N+1
    t(i)=(i-1)*dt; %creating the vector of time points
end

for k=1:m
    i_min=h/2; i=i_min+1;
    l=1; r=h+1;
    for j=1:j_max
        a=((t(r)-t(i))*B(l)+(t(i)-t(l))*B(r))/(t(r)-t(l));
        b=sqrt((t(i)-t(l))*(t(r)-t(i))/(t(r)-t(l)));
        B(i)=a+b*randn;
        i=i+h; l=l+h; r=r+h;
    end
    j_max=2*j_max;
    h=i_min;
end

figure
%multiplying by sigma scales BM to the desired distribution
plot(t, sigma*B, 'b');
set(gca,'FontSize',16);
xlabel('$t$','FontSize',20,'interpreter','latex');
ylabel('$B(t)$','FontSize',20,'interpreter','latex');
```

A.1.3 Principal Component Construction of Brownian Motion

```
close all
clear variables

rng(1); %fixing the rng seed.
T=1; N=512;
comp=min([128,N-1]); %the number of principal components considered
t=N^(-1):N^(-1):T;
C=zeros(N,N);
for i=1:N
    for j=1:N
        C(i,j)=min([i,j])/N; %creating the covariance matrix
    end
end
%returning a diagonal matrix D of eigenvalues and matrix V of eigenvectors
[V,D]=eig(C);
B=zeros(N,1);
for i=1:comp
    B=B+sqrt(D(N+1-i,N+1-i))*V(:,N+1-i)*randn;
end
```

```
20
21
22 figure
23 plot(t,B,'b');
24 set(gca,'FontSize',16);
25 xlabel('$t$','FontSize',20,'interpreter','latex');
26 ylabel('$B(t)$','FontSize',20,'interpreter','latex');
```

A.2 Geometric Brownian Motion

A.2.1 Random Walk Construction of Geometric Brownian Motion

```
1  close all
2  clear variables
3
4  T=1;     %the right endpoint of the time interval
5  N=1000; %the number of timesteps within the interval
6  dt=T/N; %the length of each timestep
7  mu=2; sigma=0.5;     %change parameters to model different GBM
8  initial=1;           %the initial value taken by GBM
9  B=zeros(1,N);        %the value GBM takes at each timestep
10
11 %generating the values at the first time step
12 B(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
13
14 for i=2:N
15     %the recurrence relation
16     B(i)=B(i-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
17 end
18
19 figure
20 plot(0:dt:T, [initial,B], 'b');
21 set(gca,'FontSize',16);
22 xlabel('$t$','FontSize',20,'interpreter','latex');
23 ylabel('$S(t)$','FontSize',20,'interpreter','latex');
```

A.2.2 Multidimensional Geometric Brownian Motion

```
1  close all
2  clear variables
3
4  T=1; N=252;              %dividing the time interval into N timepoints
5  dt=T/N;
```

```
6
7  d=2;                                %the number of dimensions
8  mu=0.3*ones(1,d);                      %the drift vector
9  sigma=0.3*ones(1,d);                   %the vector of variances
10 Corr=0.8*ones(d)+0.2*eye(d);     %leading diagonal must be 1's
11
12 Cov=zeros(d);                        %the covariance matrix
13 for covrow=1:d
14     for covcol=1:d
15         Cov(covrow,covcol)=sigma(covrow)*sigma(covcol)*Corr(covrow,covcol);
16     end
17 end
18
19 %we now find the Cholesky factor A such that AA'=Cov.
20 %The Matlab implementation of chol(Cov) returns A such that A'A=Cov,
21 %so we need to consider the transpose (i.e. the lower triangular factor).
22 A=chol(Cov).';             %the Cholesky factorisation
23
24 S=zeros(d,N);              %this matrix will hold each path in a row
25
26 %the following implements the procedure described on p104, Glasserman
27 for i=1:d           %we create a path for each of the d assets
28     randsum=0;
29     for j=1:d
30         randsum=randsum+A(i,j)*randn;
31     end
32     S(i,1)=exp((mu(i)-0.5*A(i,i))*dt+sqrt(dt)*randsum);
33     for k=2:N
34         randsum=0;
35         for j=1:d
36             randsum=randsum+A(i,j)*randn;
37         end
38         S(i,k)=S(i,k-1)*exp((mu(i)-0.5*A(i,i))*dt+sqrt(dt)*randsum);
39     end
40 end
41
42 figure
43 plot3([0:dt:T], [1,S(1,:)], [1,S(2,:)], 'color', 'red');
44
45 hold off
46 set(gca,'FontSize',16);
47 xlabel('$t$','FontSize',20,'interpreter','latex');
48 ylabel('$S_1(t)$','FontSize',20,'interpreter','latex');
49 zlabel('$S_2(t)$','FontSize',20,'interpreter','latex');
```

A.3 CEV Process

A.3.1 Simulation of the CEV Process by Milstein Approximation

```
%% A simulation of the CEV Model using the Milstein approximation

close all
clear variables

%the rightmost end of the time interval, and the number of subintervals
T=1; N=100;
mu=0.05; sigma=0.005;
Y0=0.14; % the intial value of the process at t=0
beta = 3/2; %the exponent of the price in the stochastic term of the SDE
%%
dt=T/N;
timePoints = 0:dt:T;
Y=[Y0 zeros(1,N)]; % the Milstein approximation to the process
B=randn(1,N+1);

for i=1:N
    Y(i+1) = Y(i) + mu*Y(i)*dt + sigma*Y(i)^(beta/2)*(B(i+1)-B(i)) ...
    + (beta/4)*sigma^2*Y(i)^(beta-1)*((B(i+1)-B(i))^2-dt);
end
%%
figure
plot(timePoints,Y,'r.-')
axis([0 1 0.13 0.16])
xlabel('$t$','Interpreter','latex','fontsize',16)
ylabel('$S(t)$','Interpreter','latex','fontsize',16)
```

A.4 Option Pricing

A.4.1 Asian Call Pricing

```
close all
clear variables

%this program provides a Monte Carlo estimate for the payoff of an Asian
%call option.

T=1;    %the right endpoint of the time interval
N=252; %the number of timesteps within the interval
dt=T/N; %the length of each timestep

```

```
mu=0.05; sigma=0.25;     %change parameters to model different assets
initial=1;               %the initial value taken by GBM
strike=0.5;              %the strike price of the option
r=0.02;                  %the risk-free rate

B=zeros(1,N);      %the value GBM takes at each timestep

M=1000; %the greatest number of paths for the Monte Carlo pricing

prices=zeros(1,M);

cumsum=0;       %the running total of the means of the sample paths.

for j=1:M
    %generating the values at the first time step
    B(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
    for i=2:N
        %the recurrence relation
        B(i)=B(i-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
    end

    %we now map the entries in B to their risk-neutral prices by
    %discounting at each timestep and putting these into another array
    rnB=zeros(1,N);
    for k=1:N
        rnB(k)=B(k)*exp(-r*k*T/N);
    end

    %now adding the payoff for this path to the total
    cumsum = cumsum + max(mean([initial,rnB]) - strike, 0);
    prices(j) = cumsum / j;
end


figure;
x=1:M;
y1=prices;

disp(x);
%disp(y1);

plot(x, y1, 'b-')
set(gca,'FontSize',16)
xlabel('Number of paths','FontSize',20);
ylabel('Price','FontSize',20);
```

A.4.2 Crude Lookback Call Pricing

```
%close all
%clear variables

%this program provides a Monte Carlo estimate for the payoff of a lookback
%call option.

T=1;     %the right endpoint of the time interval
N=1000; %the number of timesteps within the interval
dt=T/N; %the length of each timestep

mu=0.05; sigma=0.1; %change parameters to model different assets
initial=1;          %the initial value taken by GBM
numpaths = 10000; %the number of paths sampled in the Monte Carlo method
r=0.02; %the risk-free rate

B=zeros(1,N);       %the value GBM takes at each timestep
cumsum=0; %the running total of the means of the sample paths.


for j=1:numpaths
    %generating the values at the first time step
    B(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);

    for i=2:N
        %the recurrence relation
        B(i)=B(i-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
    end

    %now adding the payoff of the call option for the path to the total
    %the payoff of the lookback call is max(max(B)-B(N),0)
    cumsum = cumsum + (max([initial,B]) - B(N));
end

payoff = cumsum / numpaths;
price = payoff * exp(-r*T);
disp(payoff);
```

A.4.3 Antithetic Lookback Pricing

```
close all
clear variables

T=1; N=252;
dt=T/N;
trials=10^6; %must be even

```

```
mu=0.05; sigma=0.1;
initial=1;
r=0.02;

B1=zeros(1,N);
B2=zeros(1,N);
cumsum1=0;
cumsum2=0;

for i=1:(trials/2)
    Y1=randn(1,N);
    Y2=-Y1;
    B1(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*Y1(1));
    B2(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*Y2(1));
    for j=2:N
        B1(j)=B1(j-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*Y1(j));
        B2(j)=B2(j-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*Y2(j));
    end
    cumsum1=cumsum1+(max([initial,B1])-B1(N));
    cumsum2=cumsum2+(max([initial,B2])-B2(N));
end
payoff=(cumsum1+cumsum2)/trials;
price=payoff*exp(-r*T);
disp(payoff)
```

A.4.4 Exact Lookback Pricing

```
close all
clear variables

%this program provides an exact pricing for a lookback call option.
T=1;
initial=1;              %the price of the asset at t=0
mu=0.05; sigma=0.1;     %change parameters to model different assets
r=0.02;                 %the risk-free rate
t=0;                    %the time that we want to price the option for

tau=T-t;        %the time until expiry of the option
%we now define two functions to help give the exact pricing formula
deltaP = @(Tau,s) (log(s)+(r+0.5*sigma^2)*Tau)/(sigma*sqrt(Tau));
deltaM = @(Tau,s) (log(s)+(r-0.5*sigma^2)*Tau)/(sigma*sqrt(Tau));

%this can be found on [p319, Shreve]
%t is the time we are pricing at;
%x is the stock price at the given t;
%y is the maximum of the stock price process up to, and including, time t.
%this means that 0<x<=y and x=y at t=0.
```

```
term1 = @(t,x,y) (1+sigma^2/(2*r))*x*normcdf(deltaP(tau,x/y));
term2 = @(t,x,y) exp(-r*tau)*y*normcdf(-deltaM(tau,x/y));
term3 = @(t,x,y) (sigma^2/(2*r))*exp(-r*tau)*(y/x)^(2*r/sigma^2) ...
                    *x*normcdf(-deltaM(tau,y/x));

v = @(t,x,y) term1(t,x,y)+term2(t,x,y)-term3(t,x,y)-x;
price=v(0,initial,initial);

disp(price);
```

A.4.5 Comparison Between Antithetic and Crude Lookback Pricing

```
close all
clear variables

T=1;    %the right endpoint of the time interval
N=252; %the number of timesteps within the interval
dt=T/N; %the length of each timestep

mu=0.05; sigma=0.1;     %change parameters to model different assets
initial=1;              %the initial value taken by GBM
r=0.02;                 %the risk-free rate

B1=zeros(1,N);     %the value GBM takes at each timestep
B2=zeros(1,N);
B=zeros(1,N);

M=500; %max number of trials. Must be even

antiprices=zeros(1,M/2);
crudeprices=zeros(1,M);

anticumsum=0;      %the running total of the means of the sample paths.
crudecumsum=0;

%the antithetic lookback option pricing
for j=1:M/2
    Z=randn(1,N);
    %generating the values at the first time step
    B1(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*Z(1));
    B2(1)=initial*exp((mu-0.5*sigma^2)*dt-sigma*sqrt(dt)*Z(1));
    for i=2:N
        %the recurrence relation
        B1(i)=B1(i-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*Z(i));
        B2(i)=B2(i-1)*exp((mu-0.5*sigma^2)*dt-sigma*sqrt(dt)*Z(i));
    end
    payoff1=max([initial,B1]-B1(N));
```

```
    payoff2=max([initial,B2]-B2(N));
    avpayoff=(payoff1+payoff2)/2;
    %now adding the payoff for this path to the total
    anticumsum = anticumsum + avpayoff;
    antiprices(j) = anticumsum / j;
end
antiprices=antiprices*exp(-r*T);

%crude lookback option pricing
for p=1:M
    B(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
    for q=2:N
        %the recurrence relation
        B(q)=B(q-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
    end
    payoff=max(B-B(N)); %error here
    crudecumsum=crudecumsum + payoff;
    crudeprices(p)=crudecumsum/p;
end
crudeprices=crudeprices*exp(-r*T);

%exact lookback pricing
tau=T;
deltaP = @(Tau,s) (log(s)+(r+0.5*sigma^2)*Tau)/(sigma*sqrt(Tau));
deltaM = @(Tau,s) (log(s)+(r-0.5*sigma^2)*Tau)/(sigma*sqrt(Tau));
term1 = @(t,x,y) (1+sigma^2/(2*r))*x*normcdf(deltaP(tau,x/y));
term2 = @(t,x,y) exp(-r*tau)*y*normcdf(-deltaM(tau,x/y));
term3 = @(t,x,y) (sigma^2/(2*r))*exp(-r*tau)*(y/x)^(2*r/sigma^2) ...
                    *x*normcdf(-deltaM(tau,y/x));
v = @(t,x,y) term1(t,x,y)+term2(t,x,y)-term3(t,x,y)-x;
exactprice=v(0,initial,initial);

figure;
hold on
x=2:2:M;
y1=antiprices;
y2=crudeprices(2:2:M);

plot(x, y1, 'b-')
plot(x, y2, 'r-')

set(gca,'FontSize',16)
xlabel('Number of paths','FontSize',20);
ylabel('Price','FontSize',20);
hold off
legend('Antithetic MC','Crude MC')
```

A.4.6 Spread Option Pricing

```
close all
clear variables

%this program prices a call option on the spread between two assets
%n.b. multiGBMfn takes parameters (d,initial,mu,sigma,Corr,T,N)

%note that spread options are priced only on the value of the assets at
%t=T, so we need only consider the one-step paths.

r=0.05;                          %the risk-free rate
strike=0.1;                      %the strike price of the option
trials=10^6;                     %the number of Monte Carlo trials

d=2;                             %the number of assets
initial=[1,1];                   %the prices of the assets at t=0
mu=0.2*ones(1,d);                %the drift vector of the assets
sigma=0.1*ones(1,d);             %the variance of each of the assets
Corr=0.8*ones(d)+0.2*eye(d);     %the assets' correlation matrix
T=1;                             %length of time interval
N=1;                             %number of timesteps

cumsum=0;

for m=1:trials
    paths = multiGBMfn(d,initial,mu,sigma,Corr,T,N);

    %the spread option is an option on the value of the asset at expiry of
    %the option, so we only need those as opposed to the entire path:
    finalpoints=paths(:,N);
    payoff=max((finalpoints(1,1)-finalpoints(2,1))-strike,0);
    cumsum=cumsum+payoff;
end
%discounting the average terminal payoffs
price=exp(-r*T)*cumsum/trials;
disp(price);
```

A.4.7 Barrier Option Pricing

```
close all
clear variables

%this program prices a two-asset down-and-in barrier put option
%i.e. a put on S_1 that knocks in when S_2 drops below a barrier at b.
```

```
6
7  %n.b. multiGBMfn takes parameters (d,initial,mu,sigma,Corr,T,N)
8
9  r=0.05;                              %the risk-free rate
10 b=1;                                 %the barrier
11 strike=2;                            %the strike price of the option
12 trials=10^3;                         %the number of Monte Carlo trials
13
14 d=2;                                %the number of assets
15 initial=[1,1];                      %the prices of the assets at t=0
16 mu=0.2*ones(1,d);                   %the drift vector of the assets
17 sigma=0.1*ones(1,d);                %the variance of each of the assets
18 Corr=0.8*ones(d)+0.2*eye(d);        %the assets' correlation matrix
19 T=1;                                %length of time interval
20 N=252;                              %number of timesteps
21
22 cumsum=0;
23
24 for m=1:trials
25     paths = multiGBMfn(d,initial,mu,sigma,Corr,T,N);
26     if min(paths(2,:))<b
27         payoff=max(strike-paths(1,N),0);
28         cumsum=cumsum+payoff;
29     end
30 end
31 %discounting the average terminal payoffs
32 price=exp(-r*T)*cumsum/trials;
33 disp(price);
```

A.5 Short Rate Models

A.5.1 Vasicek Model Simulation

```
1  function sim = VasicekSim(T,N,b,alpha,sigma,r0)
2      %This creates a simulation of the Vasicek model with given parameters.
3      %
4
5      dt=T/N;
6      timePoints=linspace(0,T,N);
7
8      r=zeros(1,N); %the interest rate at the given time points
9      r(1)=r0;
10     Z=randn(1,N-1); %the array Z will hold the random draws from N(0,1)
11
12     for i=2:N
13         r(i)=exp(-alpha*dt)*r(i-1)+b*(1-exp(-alpha*dt))...
```

```
            +sigma*sqrt((1-exp(-2*alpha*dt))/(2*alpha))*Z(i-1);
    end
    sim = r;
end
```

```
close all
clear variables

%T is the endpoint of the time interval, N is the number of timepoints
T=1; N=1000;
%b is the long-run interest rate and alpha is the speed r(t) approaches b
sigma=0.02; %variance
b=.1; alpha=10;
r0=0.12; %the interest rate at t=0
trials=10^5;

cumSum = 0;
priceSet = [];
for i=1:trials
    sample=VasicekSim(T,N,b,alpha,sigma,r0);
    approxIntegral=(sum(sample)-sample(1))*T/N;
    samplePrice=exp(-approxIntegral);
    cumSum = cumSum + samplePrice;
    priceSet = [priceSet samplePrice];
end

priceSet = priceSet.';

fitdist(priceSet,'Normal')
figure
histfit(priceSet)
xlabel('$Bond ~ Price$','FontSize',16,'Interpreter','latex')
ylabel('$Density$','FontSize',16,'Interpreter','latex')
```

A.5.2 CIR Model Simulation

```
%% A program to simulate interest rates under the Cox-Ingersoll-Ross model
% We sample from the exact transition law of the process.
% We must keep 2*alpha*b >= sigma^2.
% An explanation can be found on pp.121-122, Glasserman.

close all
clear variables

T=1; N=1001; %T is the terminal time, N number of timepoints
```

```
%b is the long-run interest rate,alpha is the speed r approaches b
b=0.1; alpha=10;
sigma=0.02; %variance
r0=0.12; %the interest rate at t=0

timePoints=linspace(0,T,N);
dt=timePoints(2)-timePoints(1);

r=zeros(1,N); %the interest rate at the given time points
r(1)=r0;

% lambda is the noncentrality paramenter and d=4*b*alpha/sigma^2 is the
% degree of freedom of the chi-squared distribution we draw from
lambda = zeros(1,N-1);
d=4*b*alpha/sigma^2;

Chi=zeros(1,N-1); %this is the set of chi-squared random variables

for i=2:N
    lambda(i-1) = r(i-1)*4*alpha*exp(-alpha*dt) ...
                    /(sigma^2*(1-exp(-alpha*dt)));
    Chi(i-1) = (randn+sqrt(lambda(i-1)))^2 +chi2rnd(d-1);
    r(i) = sigma^2*(1-exp(-alpha*dt))/(4*alpha)*Chi(i-1);
end

figure
plot(timePoints,r)
xlabel('$t$','Interpreter','latex','fontsize',16)
ylabel('$r(t)$','Interpreter','latex','fontsize',16)
```

A.6 Jump Processes

A.6.1 Simulating a Fundamental Pure Poisson Process

```
close all
clear variables

T=100;
%mu is the mean of the exponential distribution
%that the interarrival times follow
mu=10;
interarrivalTimes=[]; %this will be populated with the interarrival times
while sum(interarrivalTimes)<=T
    interarrivalTimes = [interarrivalTimes exprnd(mu)];
end
interarrivalTimes(end)=[];
```

```
13
14 jumpTimes = cumsum(interarrivalTimes); %the arrival times
15
16 [m,n]=size(jumpTimes); %n is the number of jumps
17 jumpProcess = 0:n;
18
19 figure
20 stairs([0 jumpTimes], jumpProcess)
21 xlabel('$t$', 'Interpreter','latex', 'FontSize',16)
22 ylabel('$N(t)$', 'Interpreter','latex', 'FontSize',16)
```

A.6.2 Simulating a Simple Jump Model

```
1  % A program to simulate Merton Jump Diffusions
2  % We assume the time between jumps follows an exponential distribution
3  % We assume the relative size of each jump is lognormally distributed
4
5  close all
6  clear variables
7
8  T=100;
9  %the mean of the exponential distribution
10 %that the interarrival times follow
11 timesMu=10;
12 %the parameters describing the lognormal distribution of the jump sizes
13 jumpSizeMu=0.1; jumpSizeSigma=0.2;
14
15 interarrivalTimes=[]; %this will be populated with the interarrival times
16 while sum(interarrivalTimes)<=T
17     interarrivalTimes = [interarrivalTimes exprnd(timesMu)];
18 end
19
20 %the last entry in interarrivalTimes is greater than T so we remove it
21 interarrivalTimes(end)=[];
22
23 jumpTimes = cumsum(interarrivalTimes); %the arrival times
24
25 n=size(jumpTimes,2); %n is the number of jumps
26
27 %we generate the jump sizes
28 jumpSizes = lognrnd(jumpSizeMu,jumpSizeSigma,1,n);
29 %jumpFactor(i) is the factor by which we multiply the GBM sim by if
30 %jumpTime(i)<t<jumpTime(i+1)
31 jumpFactor = cumprod(jumpSizes);
32
33 X=linspace(0,T,1000);
34 n=size(X,2);
```

```
latestJump=zeros(1,n);

for i = 1:n
    latestJump(i) = max([0, jumpTimes(jumpTimes <= X(i))]);
end

jumpIndex=zeros(1,n);
%jumpIndex will hold the number of jumps up to the given time indexes
for j = 1:n
    jumpIndex(j) = max([0,find(jumpTimes == latestJump(j))]);
end
jumpFactor=[1 jumpFactor];

figure
scatter(X,jumpFactor(jumpIndex+1),2,'filled')
xlim([0, T])
ylim([0,max(jumpFactor)])
xlabel('$t$', 'Interpreter','latex', 'FontSize',16)
ylabel('$S(t)$', 'Interpreter','latex', 'FontSize',16)
```

A.6.3 Simulating Merton's Jump Diffusion Model

```
function sim = PureJump(T,N,timesMu,jumpSizeMu,jumpSizeSigma)

dt=T/N;

interarrivalTimes=[]; %this will be populated with the interarrival times
while sum(interarrivalTimes)<=T
    interarrivalTimes = [interarrivalTimes exprnd(timesMu)];
end
%the last entry in interarrivalTimes is greater than T so we remove it
interarrivalTimes(end)=[];

jumpTimes = cumsum(interarrivalTimes); %the arrival times

n=size(jumpTimes,2); %n is the number of jumps

%we generate the jump sizes
jumpSizes = lognrnd(jumpSizeMu,jumpSizeSigma,1,n);
%jumpFactor(i) is the factor by which we multiply the GBM sim by if
%jumpTime(i)<t<jumpTime(i+1)
jumpFactor = cumprod(jumpSizes);

X=0:dt:T;
n=size(X,2);
latestJump=zeros(1,n);

```

```
for i = 1:n
    latestJump(i) = max([0, jumpTimes(jumpTimes <= X(i))]);
end

jumpIndex=zeros(1,n);
%jumpIndex will hold the number of jumps up to the given time indexes
for j = 1:n
    jumpIndex(j) = max([0,find(jumpTimes == latestJump(j))]);
end
jumpFactor=[1 jumpFactor];

sim = jumpFactor(jumpIndex+1);
end
```

```
function sim = GBM(T,N,mu,sigma,initial)
dt = T/N;
B=zeros(1,N);       %the value GBM takes at each timestep
%generating the values at the first time step
B(1)=initial*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);

for i=2:N
    %the recurrence relation
    B(i)=B(i-1)*exp((mu-0.5*sigma^2)*dt+sigma*sqrt(dt)*randn);
end

sim = [initial,B];
end
```

```
close all
clear variables
T=1;
N=999999;
timesMu=0.15;
jumpSizeMu=0.05;
jumpSizeSigma=0.08;
mu=0.1;
sigma=0.4;
initial=0.12;

process = PureJump(T,N,timesMu,jumpSizeMu,jumpSizeSigma) ...
          .*GBM(T,N,mu,sigma,initial);
dt=T/N;
X=0:dt:T;
figure
scatter(X,process,1,'filled')
xlabel('$t$', 'Interpreter','latex', 'FontSize',16)
```

```
ylabel('$S(t)$', 'Interpreter','latex', 'FontSize',16)
```

A.6.4 Lookback Option Pricing Under Merton's Model

```
close all
clear variables
T=1;
N=99;
timesMu=0.15;
jumpSizeMu=0.3;
jumpSizeSigma=0.08;
mu=0.1;
sigma=0.4;
initial=1;

payoffSet = [];
trials=10^4;

for i=1:trials
    sample = PureJump(T,N,timesMu,jumpSizeMu,jumpSizeSigma)...
                .*GBM(T,N,mu,sigma,initial);
    samplePayoff = max(max(sample),0);
    payoffSet=[payoffSet samplePayoff];
end

disp(mean(payoffSet))
disp(var(payoffSet))

figure
histogram(log(payoffSet))
xlabel('$Option ~ Price$','FontSize',16,'Interpreter','latex')
ylabel('$Density$','FontSize',16,'Interpreter','latex')
```

```
close all
clear variables
T=1;
N=99;
timesMu=0.15;
jumpSizeMu=0.4;
mu=0.1;
sigma=0.4;
initial=1;

prices = [];
payoffSet = [];
```

```
trials=10^3;

X=0:0.02:1;

for jumpSizeSigma = X
    for i=1:trials
        sample = PureJump(T,N,timesMu,jumpSizeMu,jumpSizeSigma)...
                    .*GBM(T,N,mu,sigma,initial);
        samplePayoff = max(max(sample)-sample(end),0);
        payoffSet = [payoffSet samplePayoff];
    end
    prices = [prices mean(payoffSet)];
end


figure
plot(X,prices)
xlabel('$\sigma_{jump size}$','FontSize',16,'Interpreter','latex')
ylabel('$Option ~ Price$','FontSize',16,'Interpreter','latex')
```

References

[1] ÅKESSON, F., and LEHOCZKY, J.P. *Discrete Eigenfunction Expansion of Multi-Dimensional Brownian Motion and the Ornstein-Uhlenbeck Process*. Department of Statistics, Carnegie-Mellon University, Pittsburgh, 1998.

[2] BAZ, J., and CHACKO, G. *Financial Derivatives: Pricing, Applications, and Mathematics*. Cambridge: Cambridge University Press, 2005.

[3] BOUCHAUD, J.-P. and POTTERS, M. *Theory of Financial Risk and Derivative Pricing*(2nd ed.) Cambridge: Cambridge University Press, 2003.

[4] COX, J.C., INGERSOLL, J.E., and ROSS, S.A *A Theory of the Term Structure of Interest Rates*. Econometrica, 1985, 53(2), pp.385–407.

[5] GAMERMAN, D. and LOPES, H.F. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*(2nd ed.) Boca Raton: CRC Press, 2006.

[6] GLASSERMAN, P. *Monte Carlo Methods in Financial Engineering*. Berlin: Springer-Verlag, 2003.

[7] HIGHAM, D.J. *An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations*. SIAM Review, 2001, 43(3), pp.525–546.

[8] HO, T.S.Y., and LEE S.-B. *Term Structure Movements and Pricing Interest Rate Contingent Claims*. The Journal of Finance, 1986, 41(5), pp.1011–1029.

[9] JOSHI, M.S. *The Concepts and Practice of Mathematical Finance*(2nd ed.) Cambridge: Cambridge University Press, 2008

[10] KARATZAS, I., and SHREVE, S. *Brownian Motion and Stochastic Calculus*(2nd ed.) Berlin: Springer-Verlag, 1991.

[11] KLOEDEN, P.E., and PLATEN, E. *Numerical Solution of Stochastic Differential Equations*. Berlin: Springer-Verlag, 1992.

[12] KROESE, D.P., TAIMRE, T., and BOTEV, Z.I. *Handbook of Monte Carlo Methods*. Hoboken: Wiley, 2011.

[13] LŐRINCZI, J. *Lecture Notes to 18MAP204 Stochastic Calculus and Theory of Pricing*. 2019.

[14] MERTON, R.C. *Option Pricing when Underlying Stock Returns are Discontinuous*. Journal of Financial Economics, 1976, 3(1–2), pp.125–144.

[15] SCHILLING, R.L. and PARTZSCH, L. *Brownian Motion: An Introduction to Stochastic Processes*. Berlin: De Gruyter, 2012.

[16] SCHRODER, M. *Computing the Constant Elasticity of Variance Option Pricing Formula*. The Journal of Finance, 1989, 44(1), pp.211–219.

[17] SHREVE, S. *Stochastic Calculus for Finance II*. Berlin: Springer-Verlag, 2004.

[18] SHUMWAY, R.H. and STOFFER D.S. *Time Series Analysis and Its Applications*(4th ed.) Berlin: Springer-Verlag, 2017.

[19] SUNDARAM, R. and DAS, S. *Derivatives: Principles and Practice*. New York: McGraw-Hill/Irwin, 2011.

[20] VASICEK, O. *An Equilibrium Characterisation of the Term Structure*. Journal of Financial Economics, 1977, 5(2), pp.177–188.

[21] WILMOTT, P. *Paul Wilmott Introduces Quantitative Finance*(2nd ed.) Hoboken: Wiley, 2007.