Monte Carlo and Binomial Tree Methods in Pricing of Options

Matthew Pollard*

4th June 2005

Abstract

This report serves as an introduction to option pricing by Monte Carlo and binomial tree methods. The report treats both the theoretical and applied aspects in pricing options using the two methods and presents the $R-$based *POP* (Pollard Option Pricing) function package that performs the pricing. A comparison between results given by both methods and from Black-Scholes formula is made. The Cox-Ross-Rubinstein binomial model is shown to be empirically more efficient than Monte Carlo simulation in pricing European options.

*Faculty of Science, Department of Mathematics, The Australian National University, Australia.

Contents

1 Introduction **3**

2 Options Basics **4**

3 Pricing theory and the Black-Scholes model **6**
3.1 The Black-Scholes model . . . 6
3.2 Brownian motion price model . . . 7
3.3 Extensions . . . 9
3.3.1 Stochastic Volatility . . . 9

4 Numerical Methods **10**
4.1 Monte Carlo Simulation . . . 10
4.1.1 Simulation of the price path . . . 13
4.1.2 Convergence Properties . . . 13
4.2 Binomial Tree Approximation . . . 14
4.2.1 Convergence properties . . . 16
4.2.2 Alternative Binomial Tree models . . . 16

5 Results **18**

6 Conclusion **18**

A Appendix **19**
A.1 Constructing Monte Carlo confidence intervals . . . 19

B Software Implementation **22**
B.1 Examples . . . 22

C Program Code **28**
C.1 General functions . . . 28
C.1.1 BSeuro . . . 28
C.1.2 gen.brown . . . 29
C.2 Monte Carlo Simulation . . . 29
C.2.1 MCsim . . . 29
C.2.2 MCstocsim . . . 31
C.2.3 ntrial . . . 32
C.3 Binomial Tree Approximation . . . 33
C.3.1 Binsim . . . 33
C.3.2 BinsimTree . . . 34
C.3.3 TreePlot . . . 35
C.3.4 BinsimOpt . . . 36
C.3.5 BinsimError . . . 36

1 Introduction

Since the original 1973 publication, the Black-Scholes model and it's pricing formula has risen to become one of the most celebrated results in mathematical finance. The Black-Scholes formula for pricing European put and call options is used by thousands of options traders each day and is the subject of libraries of academic literature. However, the model underlying the formula is restrictive and makes several well-documented unrealistic assumptions [5]. One such assumption is constant market volatility. A highly active research area over the last decade has been in addressing Black-Scholes' shortcomings and proposing less restrictive alternative models. Unfortunately, while the Black-Scholes model has a closed-form price formula, most general models do not. Instead, numeric methods such as Monte Carlo simulation and Binomial tree methods are heavily relied upon to price options.

Options are derivative securities. That is, they are securities that are derived from underlying assets. They have been traded for centuries: the earliest documented trade is from Ancient Greece; but it was only in the late 20th century that they gained economic importance. They have become especially popular since 1973, when the Chicago Board Options Exchange was established and trading became organised. Options come in two fundamental types: call and put. A *call option* is a contract that gives the holder the right (but not the obligation) to buy a fixed amount of an asset at a specified time in future for an already agreed price, the *strike price,* from the seller (or *writer* of the option). The opposite option is a *put option.* By buying it, the holder receives the right to sell a fixed amount of asset to writer for the strike price. Here the writer is obliged to buy the asset while the holder may decide on selling or not.

The Black-Scholes formula is the solution of a diffusion partial differential equation (PDE). The price of a financial asset is modeled by a Brownian process with constant volatility. The PDE for European financial options is mathematically tractable. However, solving the associated PDE for other option types, such as American or Asian options, is analytically much harder and more "exotic" options generally yield intractable PDEs.

In cases where a closed-form solution for price cannot be found, numerical methods must be employed. One approach is numerically solving the PDE, such as using finite difference methods. A second, and more interesting, approach works by simulating possible pathways for the price of an asset repeatedly and estimating the expected option price. Both Monte Carlo and binomial tree methods use this approach.

Monte Carlo simulation works by directly simulating the price path of an asset, and for each simulation calculates the realised payoff of the option. The expected value is then estimated by the sample mean of payoffs and is justified by the Law of Large Numbers. Binomial tree simulation, instead models price paths in discrete time by binomial branching. At each node of the tree, the price can increase by factor m with probability p and decrease by factor n with probability *1-p*. This approach is justified by the Central Limit Theorem.

This report serves as an introduction to option pricing by both Monte Carlo and binomial tree methods. The report treats both the theoretical and applied aspects in pricing options using the two methods and presents the $R-$based *POP* (Pollard Option Pricing) function package that performs the pricing. A comparison between results given by both methods and from Black-Scholes formula is made. The Cox-Ross-Rubinstein binomial model is shown to be empirically

more efficient than Monte Carlo simulation in pricing European options.

The report is organised as follows. Section two serves as a general introduction to financial options and introduces the fundamental discounted expectation pricing formula. Section three introduces asset price theory and the modelling of asset prices as geometric Brownian motion processes. Also, the Black-Scholes model and formula is presented.

Sections three and four deal with the theory and implementation of Monte Carlo and binomial tree methods respectively. Section 5 presents the numeric results of the *POP* package in pricing various options and compares them to Black-Scholes formula. Section B of the appendix provides a tutorial in using *POP* to price various options and perform error analysis and section C provides the code for all functions for *POP.*

The following notation will be used throughout the report:

S_t	asset price at time t
S_0	initial asset price
T	expiration date of option contract
K	strike price of option contract
r	annual riskless interest rate
σ	volitility of asset price
μ	drift in mean of price
P_T	option value at maturity
$f(\cdot)$	exercise payoff function
$(x)^+$	$\max(x, 0)$

2 Options Basics

The problem of pricing options is one of the central problems in finance. All asset derivatives can be decomposed down to two financial instruments: forward contracts and options. The problem of pricing forward contracts is relatively simple, however until Black-Scholes was developed, pricing options was effectively guess-work. Now nearly all options traders use variants of Black-Scholes original formula, and the options market has since seen explosive growth.

Options come in various contract rules. The two most widely traded are *European* and *American.* The holder of an European option may only exercise it at a fixed pre-determined future date, the *maturity date* T. An American option, whereas, may be exercised at any time before and up to the agreed contract length. These two contract forms are commonly referred to as *plain vanilla* put and calls. The explicit form of traded option contracts are frequently very different from the two described. These are classified as *exotic options*, and *Asian option* is the most well known of these.

A *European* call option on one share of stock offers the buyer of the option the right to by this share at time $t = T$ for strike price $K \geq 0$ which is fixed at time $t = 0$. If the share price S_t at $t = T$ exceeds the strike price, then the holder can exercise the option and buy the share at

price $K < S_T$. He can then immediately sell the share at the market price S_T and make a gain of $S_t - K$. If, however, we have $S_T < K$, then the option holder will not exercise the option since he can buy the stock at a lower price than K. Thus the gain from holding the option is zero. Therefore, the gain V from holding a European option is given by

$$V = (S_T - K)^+$$

where $(x)^+$denotes $\max(S_T - K, 0)$. The payoff functions for several other option types is shown below,

$$\begin{aligned}
\text{European put} &: (K - S_T)^+ \\
\text{American call} &: (S_\tau - K)^+,\ \tau \in [0, T] \\
\text{Asian call} &: \left(S_T - T^{-1}\int_0^T S_t\ dt\right)^+ \\
\text{Fixed-strike average call} &: \left(T^{-1}\int_0^T S_t\ dt - K\right)^+ .
\end{aligned}$$

In general, the value of an option, P_t, at $t = 0$ is is the *discounted expectation* of the payoff function f at maturity:

$$P_T = E\{e^{-rT} f(S_t)\}$$

The e^{-rT} term is the future discounting term that takes account of the time-value of money: a fixed amount of money is worth more at $t = 0$ than $t = T$ by factor e^{rT} due to compounding interest.

Options serve the economic function of hedging against unfavourable price fluctuations of the underlying asset. A conservative stock holder may protect him- or herself against future possible loss in asset value by selling some possible profits from a large increase in the stock price. That is, if the asset price drops, the sold option will not be exercised and the stock holder has made a gain from the revenue from sale. If asset price increases beyond the strike price, the stock holder now may be obliged to sell the asset at below the current market price, thus has lost some possible profits.

Of course, options are also traded by speculators who hope for an over-proportional increase in the option value compared to the price of the underlying asset. Options are attractive for speculators as they are much cheaper than the underlying asset itself and so with little capital it is possible to make relatively large gains. But history has shown that option speculators also suffer big losses.

One of the largest wide-spread losses occurred at the beginning of the 17th century in the Netherlands. Then, during the "tulip mania", many tulip growers wanted some insurance against fluctuations in prices. They bought contracts giving them the right to sell tulips for a fixed price to the sellers of these contracts if the market price of tulips would decline. This was a type of put option. The sellers of these contracts were hoping for a price increase so that the tulip growers would not make use of the contracts and they could get away with the gain out of the contract

deal. But then, in 1637, the Dutch tulip market spectacularly crashed overnight and the option sellers were not able to keep their part of the agreement. The consequence was a serious economic crisis in the Netherlands and option-type contracts received a bad reputation in Europe, which lasted for some time. A more recent example concerns the fate of the company Robert Merton and Myron Scholes, of Black-Scholes fame, set up in the 1990s called Long Term Capital Management (LTCM). By 1997, the fund had returned US $6 billion to its investors, however in 1998, sudden shifts in market volatility caused collapse of the fund and a total loss of US $14 billion.

3 Pricing theory and the Black-Scholes model

This section presents the Black-Scholes model and formula. It discusses in detail the fundamental assumption that asset prices follow a geometric Brownian motion process.

3.1 The Black-Scholes model

The original Black-Scholes model was presented in the paper titled "The Pricing of Options and Corporate Liabilities" in 1973, and quickly became the standard model for pricing options. This model was designed, however, with several simplifications.

The key assumptions are that:

- The price of the underlying instrument is a geometric Brownian motion, in particular with constant drift and volatility.
- It is possible to short sell the underlying stock.
- There are no riskless arbitrage opportunities.
- Trading in the stock is continuous.
- There are no transaction costs or taxes.
- All securities are perfectly divisible (e.g. it is possible to buy 1/100th of a share).
- The risk free interest rate is constant, and the same for all maturity dates.

Under this model, a closed-form solution for P exists - the famous Black-Scholes formula.

The Black-Scholes formula *The price P_C of a European call option with strike price $K > 0$ and maturity at $t = T$ is given by*

$$P_C = S_0 \cdot \Phi\{d_1\} - K \cdot e^{-rT}\Phi\{d_2\}$$

and the price P_P of a European put option is given by

$$P_P = K \cdot e^{-rT}\Phi\{-d_2\} - S_0\Phi\{-d_1\}$$

where

$$d_1 = \frac{\ln\left(\frac{S_0}{K}\right) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1(t) - \sigma\sqrt{T}$$

and where Φ *is the cumulative normal distribution function.*

See [6] for rigorous proof using Green's functions.

3.2 Brownian motion price model

At the heart of the Black-Scholes model is the assumption that the price of the underlying asset, S_t, follows a geometric Brownian motion process. Brownian motion, denoted $W(t)$, is a stochastic process similar to a random walk, however it has several very interesting and mathematically deep features. It is defined by a continuous-time stochastic process where

$$\begin{aligned} W(0) &= 0 \\ W(t) - W(s) &\sim \mathcal{N}(0, t-s) \text{ for any } t > s \geq 0 \\ W(t + \Gamma t) & \quad \text{ is independent of } W(t) \end{aligned}$$

While the process is a continuous function at every point, it is no-where differentiable.[1] Also, for any $\varepsilon > 0$, $W(t) : \ t \in [0, \varepsilon]$ crosses the line $W(t) = 0$ an uncountably infinite number of times [7]. Einstein first proposed the existence of such a process, and the idea was applied to modelling the diffusion of gas particles.

In Black-Scholes model, the asset price S_t is the solution of a *stochastic differential equation* (SDE) given by

$$dS_t = \mu S_t dt + \sigma S_t dW_t. \tag{1}$$

where W_t is a Brownian motion process and dW_t is a stochastic differential. The solution, S_t, is given by

$$S(t) = S_0 \exp\left\{\left[\mu - \frac{\sigma^2}{2}\right] t + \sigma X(t)\right\} \tag{2}$$

where

$$X(t) = \underbrace{W(t) - W(0)}_{\sim\mathcal{N}(0,t)}$$

An example of a simulated asset price path generated by (2) is shown in figure 2.

Proof of (2) by first principles is difficult, however we can verify it is a solution to (1). To do this, we will use Itô's lemma. This is the stochastic integral equivalent of the fundamental theorem of calculus.

[1] Before the mathematician Wierstrass, most mathematicians believed that all continuous functions were differentiable at every point.

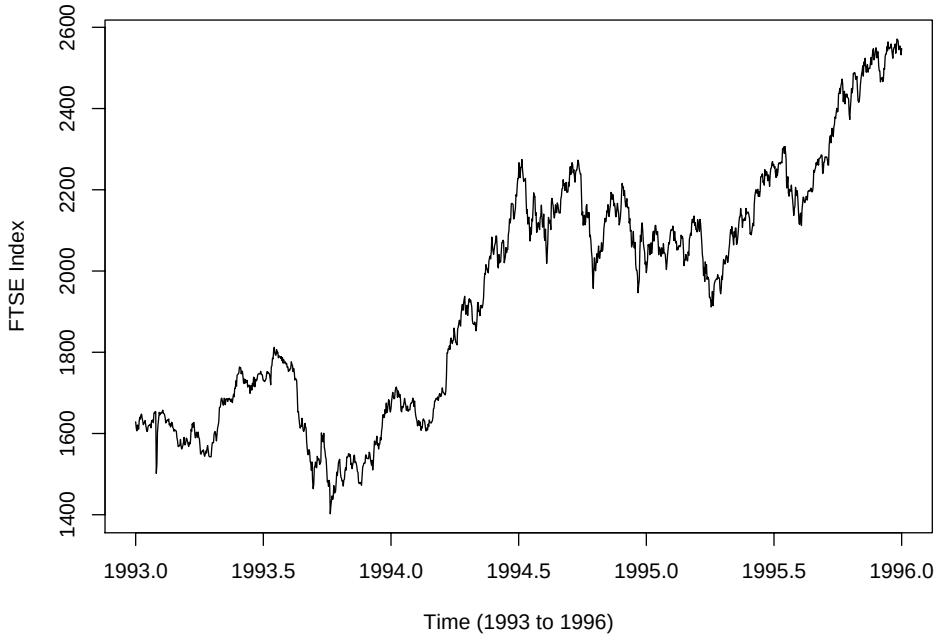


Figure 1: *The daily FTSE index from 1993 to 1996*

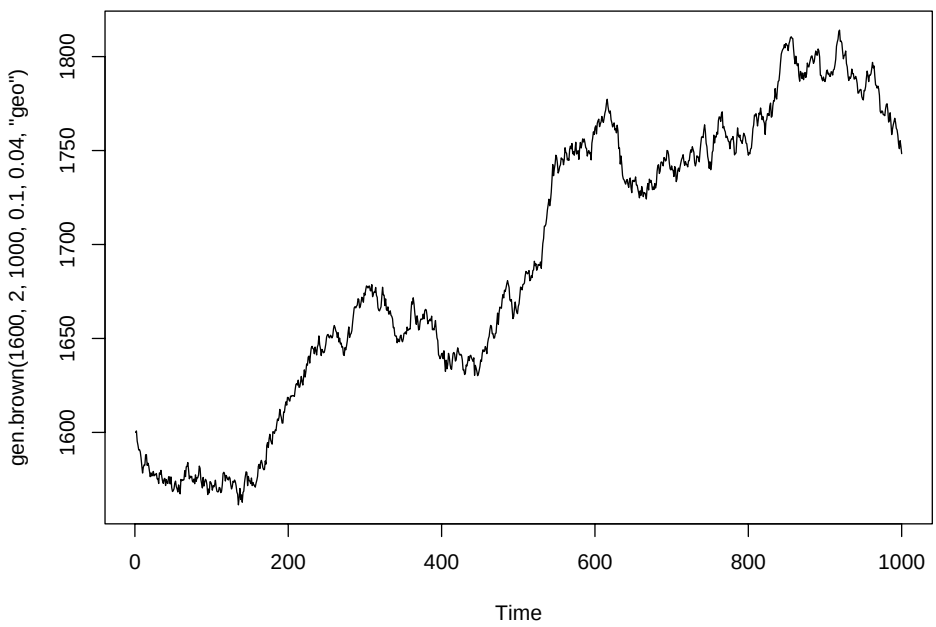


Figure 2: *Simulated asset price path,* S_t*, with interest rate* $r = 0.05$ *and volatility* $\sigma = 0.1$

Proposition 1 (Itô's lemma) *Let W_t be a Brownian motion process and x_t an Itô process with $dx_t = a(x,t)dt + b(x,t)dW_t$. Let $y_t = F(x,t)$. Then*

$$dy_t = \left(a\frac{\partial F}{\partial x} + \frac{\partial F}{\partial t} + \frac{b^2}{2}\frac{\partial^2 F}{\partial x^2} \right) dt + b\frac{\partial F}{\partial x} dW_t.$$

To show that $S(t) = S_0 \exp\left\{(\mu - \sigma^2/2)t + \sigma X(t)\right\}$ is a solution of our SDE, let $a = 0$ and $b = 1$. This gives

$$dx_t = dW_t$$

Let $F(X,t) = S_t$. Then by Itô, we have

$$dS_t = \left(\frac{\partial S_t}{\partial t} + \frac{1}{2}\frac{\partial^2 S_t}{\partial W_t^2} \right) dt + \frac{\partial S_t}{\partial W_t} dW_t$$

Note that $\frac{\partial S_t}{\partial t} = (\mu - \sigma^2/2)S_t$, $\frac{\partial S_t}{\partial W_t} = \frac{\partial \sigma\{W(t) - W(0)\}}{\partial W_t} S_t = \sigma S_t$ and $\frac{\partial^2 S_t}{\partial W_t^2} = \sigma^2 S_t$, so

$$\begin{aligned} dS_t &= \left(\left[\mu - \frac{\sigma^2}{2} \right] S_t + \frac{1}{2}\sigma^2 S_t \right) dt + \sigma S_t dW_t \\ &= \mu S_t dt + \sigma S_t dW_t \end{aligned}$$

hence S_t is a solution to (1).

3.3 Extensions

The original Black-Scholes model makes several limiting assumptions. The two most important of these are:

- No dividends are payed during the option contract;
- Volatility is constant up to maturity;

The first limitation was quickly removed by Black in 1976 (the model is known as Black-76), and subsequently improved by Garman and Kohlhagen in 1983, allowing both discrete ("lumpy") and continuous dividend flows. The second limitation has proven to be a much more formidable obstacle. This is discussed in the subsequent section.

3.3.1 Stochastic Volatility

A well-known limitation of the Black-Scholes model is that it assumes market volatility is constant over the option contract , that is,

$$\sigma(t) \equiv \sigma$$

In real-world market data, even in relatively short periods, volatility is seen to change and is often correlated with total volume of assets traded (in particular, what is termed *volatility clustering*) [5].

Several stochastic models have been proposed to model underlying volatility

- ARCH/GARCH time series models;
- Arithmetic random walk processes;
- Geometric random walk processes.

The arithmetic random walk model assumes that the volatility at time t, σ_t is the arithmetic sum of previous discrete volatility measurements $(\sigma_1, ..., \sigma_{t-1})$ and a error term:

$$\begin{aligned} \sigma_t &= \sigma_{t-1} + \varepsilon_{t-1}, \quad \varepsilon_i \sim N(0, \theta^2) \\ \Rightarrow \sigma_t &= \overline{\sigma} + \sum_{j=1}^{t-1} \varepsilon_i \end{aligned}$$

where $\overline{\sigma}$ is global mean volatility. The geometric random walk model instead assumes that

$$\begin{aligned} \sigma_t &= \sigma_{t-1}(1 + \varepsilon_{t-1}), \quad \varepsilon_i \sim N(0, \theta^2) \\ \Rightarrow \sigma_t &= \prod_{j=1}^{t-1} \overline{\sigma}(1 + \varepsilon_j). \end{aligned}$$

Figure 3 shows a comparison between constant and stochastic volatility price paths using the arithmetic random walk mode. The stochastic volatility plot is characterised by periods of low and high fluctuations.

While the Black-Scholes PDE has a closed-form solution, most of the PDEs associated with stochastic volatility models are mathematically intractable. That is, there is no closed form solution. In such cases, numerical techniques, such as Monte Carlo and binomial trees, must be used.

4 Numerical Methods

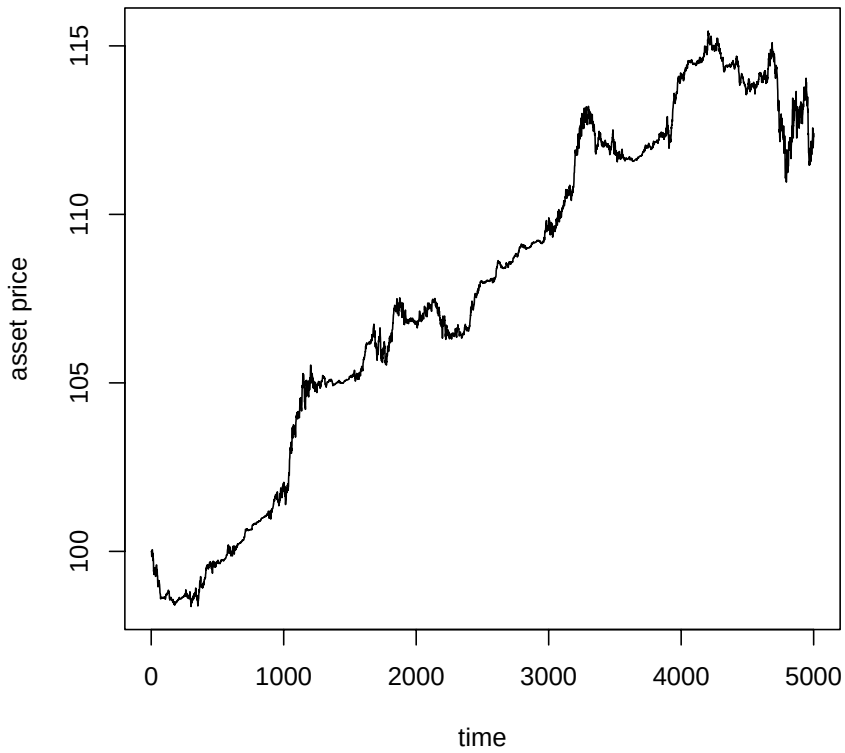
This section presents Monte Carlo and Binomial Tree methods applied to option pricing.

4.1 Monte Carlo Simulation

Monte Carlo simulation works by directly simulating the price path of an asset, and for each simulation, calculates the realised payoff of the option. The price is then estimated by the sample mean of payoffs. Monte Carlo simulation is a particularly popular method in pricing options due to its easy implementation. Furthermore, every arbitrary exotic option can be approximated by the method.

The basis for Monte-Carlo simulation is the strong law of large numbers (SLLN). This law states that for a finite sequence of random variables $X_1, X_2, ..., X_n$ independently and identically distributed with mean μ and variance $\sigma^2 < \infty$, the sample average

$$\bar{X}_n = (X_1 + ... + X_n)/n$$



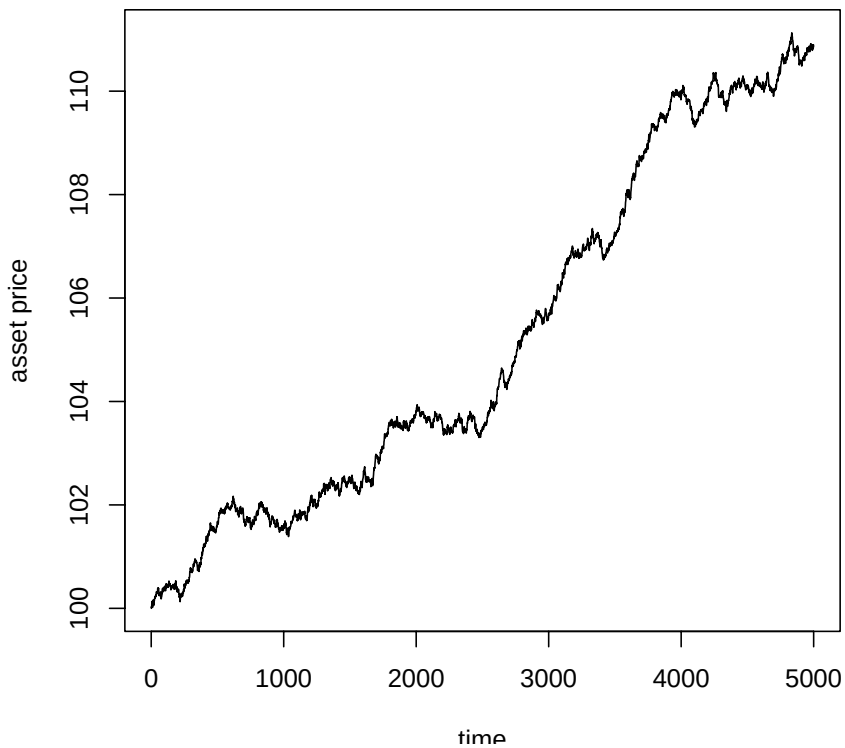


Figure 3: *Top:* Simulated S_t with stochastic volatility (arithmetic random walk), $r = 0.1$, $\bar{\sigma} = 0.02$. *Bottom:* Simulated S_t with constant volatility, $r = 0.1$, $\sigma = 0.02$.

converges almost surely to μ as $n \to \infty$. That is,

$$P(\lim_{n\to\infty} \bar{X}_n = \mu) = 1$$

The so-called "crude" Monte Carlo estimator of μ is simply $\bar{X}_n$. This is an unbiased estimator since

$$\begin{aligned} E(\bar{X}_n) &= E(X_1 + ... + X_n)/n \\ &= n\mu/n \\ &= \mu, \end{aligned}$$

and the variance of the estimator is

$$\begin{aligned} Var(\bar{X}_n) &= Var(X_1 + ... + X_n)/n^2 \\ &= \{Var(X_1) + ... + Var(X_n)\}/n^2 \text{ by independence} \\ &= n\sigma^2/n^2 \\ &= \sigma^2/n \end{aligned}$$

Recall that computing an option price is nothing else than computing the discounted expectation of the payoff $V = f(S_t)$, i.e.

$$P = E_Q\{e^{-rT} f(S_t)\}$$

This suggests the following MC estimator for P

$$\widehat{P}_n = e^{-rT} \frac{\sum_{j=1}^n f(S_t^{(j)})}{n} \tag{3}$$

where $S_t^{(j)}$ are simulated price paths of the underlying asset. By direct application of the SLLN, we have

$$\begin{array}{cccc} & \frac{\sum_{j=1}^n V_j}{n} & \to & E_Q(V) \quad \text{as } n \to \infty \\ \Rightarrow & \widehat{P}_n & \to & P \quad \text{as } n \to \infty \end{array}$$

The algorithm for this method given below.

Monte Carlo algorithm

1. for $k = 1, 2, ..., n$

   (a) generate price path $S_t^{(j)} = \{S_0, S_{\Delta t}, ..., S_{T-\Delta t}, S_T\}^{(j)}$

   (b) compute payoff $V_j = f(S_t^{(j)})$

2. let $\widehat{P}_n = e^{-rT}\bar{V} = e^{-rT}\frac{\sum_{j=1}^n V_j}{n}$

The function **MCsim** caries out this algorithm. The difficult aspect to Monte Carlo simulation is in generating S_t. The following subsection presents an algorithm for doing this.

4.1.1 Simulation of the price path

Recall that the price path S_t is modelled by a geometric Brownian motion process satisfying (1). For path-independent options, such as the European, the terminal price S_T is only used in calculating the payoff V. From (2) the terminal price can be simulated by

$$S(T) = S_0 \exp\left\{ \left[\mu - \frac{\sigma^2}{2}\right] T + \sigma \underbrace{X(T)}_{\sim\mathcal{N}(0,T)} \right\},$$

and thus calculation of (3) with n simulations simply involves generating n random numbers drawn from $\mathcal{N}(0,T)$.

For path-dependent options, such as Asian, the entire price path S_t from $t = 0$ to $t = T$ must be simulated. As such a path is given by infinitely many (i.e. uncountably many) values, it can only be simulated approximately. To do this, the following procedure [2] is used.

Price path algorithm

1. Divide the interval $[0,T]$ into $N \gg 1$ equidistant parts.
2. Generate N independent random numbers from $\mathcal{N}(0,1)$.
3. Simulate an approximate Brownian motion path $W(t)$ on $[0,T]$ given by:

$$W(0) = 0,$$
$$W(j \cdot \frac{T}{N}) = W((j-1) \cdot \frac{T}{N}) + \sqrt{\frac{T}{N}} \cdot Y_j, \quad j = 1, ..., N,$$
$$W(t) = W((j-1)\frac{T}{N}) + (t - (j-1)\frac{T}{N}) \cdot \frac{N}{T} \cdot [W(j\frac{T}{N}) - W((j-1)\frac{T}{N})] \quad \text{for } t \in [(j-1)\tfrac{T}{N}, j\tfrac{T}{N}]$$

4. Use this to generate an approximate path of the price process S_t :

$$S_t = S_0 e^{(r - \frac{1}{2}\sigma^2)t} \cdot e^{\sigma W(t)}, \quad t \in [0,T].$$

Korn (1999) shows that the approximate $W(t)$ generated by this procedure converge in probability to a true Brownian motion process as $N \to \infty$. Figure 4 shows several price paths generated by the above method. The function **gen.brown** of the *POP* package implements generation of price paths by this method.

4.1.2 Convergence Properties

We've seen that by the law of large numbers, $\widehat{P}_n \to P$ as $n \to \infty$. Suppose we wish to determine a lower bound N for n to ensure the relative error between $\widehat{P}_n$ and $\widehat{P}$ is no more than than ε for $n > N$ and for a certain confidence level α. That is,

$$Pr\left(\varepsilon \geq \left|\frac{\bar{P}_n - P}{P}\right|\right) > \alpha \text{ for } n > N$$

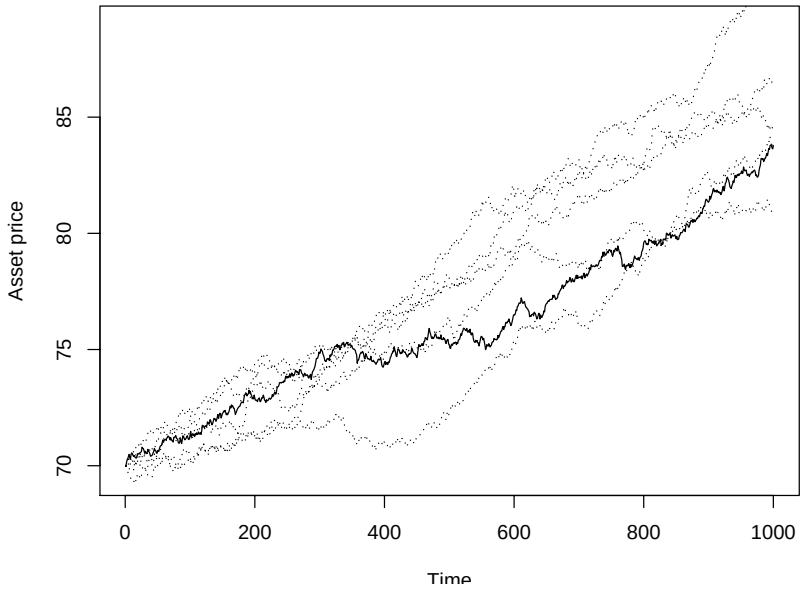


Figure 4: Several simulated price paths with market parameters $r = 0.1,\ \sigma = 0.1,\ T = 1,\ S_0 = 70$.

Set $\varepsilon = \left|\frac{\bar{P}_N - P}{P}\right|$. By constructing a confidence interval for $\bar{P}_n$ and application of the central limit theorem, the following bound may be derived

$$N \doteq \frac{\sigma^2 \xi^2}{P^2 \varepsilon^2}.$$

where $\sigma^2 = Var(V)$ and $\xi = F^{-1}\left(\frac{1+\alpha}{2}\right)$. Proof of this is given in the appendix. This is a useful result in determining how large n should be for a given problem. The function **ntrial** calculates this bound.

4.2 Binomial Tree Approximation

Binomial tree simulation models price paths in discrete time by binomial branching. Binomial tree methods typically converge much faster than Monte-Carlo simulation, however are less popular, not as intuitive and relatively harder to implement.

Suppose we model the price path process S_t in discrete time by

$$S^{(n)}(t),\ \ t \in \{0, \frac{T}{n}, \frac{2T}{n} ..., T\}.$$

At each t, the price path splits into two paths, and can either increase by the factor u with probability p or decrease by factor d with probability $1 - p$. Such a stock price model is called a Cox-Ross-Rubinstein model (see[1]). The idea behind the method is to set the terminal prices

$$S^{(n)}(T) = \{S_0 u^0 d^n, S_0 u^1 d^{n-1}, ..., S_0 u^n d^0\},$$

set the terminal payoffs as

$$V^{(n)}(T) = \{f(S_0u^0d^n), f(S_0u^1d^{n-1}), ..., f(S_0u^nd^0)\}$$

and place these on the outermost branches of the tree. To work out $V(T - \frac{T}{n})$ at each node, for every $V(T)$ node work backwards by setting the payoff as $pV_{upper}(T) + (1-p)V_{lower}(T)$ where V_{upper} is the payoff given by the top branch connecting to the node, V_{lower} is the payoff given by the bottom branch connecting to the node. Repeat inductively for $V(T - \frac{2T}{n}), ..., V(T - \frac{nT}{n}) = V(0)$. Then set

$$\widehat{P}^{(n)} = e^{-rT}V(0).$$

Cox, Ross and Rubinstein show that for this binomial method to converge, u and d and p must be set according to

$$u = e^{\sigma\sqrt{T}}, \;\; u = e^{-\sigma\sqrt{T}}, \;\; p = \frac{e^{r\Delta t} - d}{u - d}.$$

A binomial tree with $V(t)$ drawn for each branching is shown in figure 5. The algorithm for this method is given below.

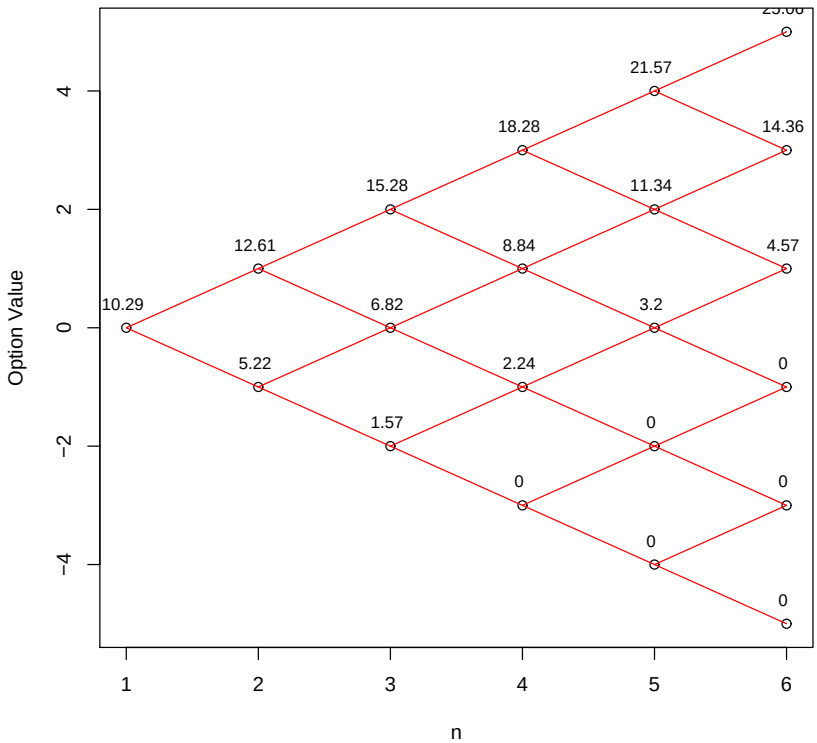


Figure 5: *A $N = 5$ branch binomial tree for European option with $r = 0.1,\ \sigma = 0.1,\ T = 1,\ S_0 = 100$. Estimated option value is the number at the left-most node.*

Binomial Tree Algorithm

1. let $\Delta t = T/n$, $u = e^{\sigma\sqrt{T}}$, $d = e^{-\sigma\sqrt{T}}$, $p = \frac{e^{r\Delta t}-d}{u-d}$, $df = e^{-r\Delta t}$

2. generate terminal prices $S^{(n)}(T) = \{S_0u^0d^n, S_0u^1d^{n-1}, ..., S_0u^nd^0\}$ and payoffs $V^{(n)}(T) = \{f(S_0u^0d^n), f(S_0u^1d^{n-1}), ..., f(S_0u^nd^0)\}$
3. for $j = n-1, ..., 0$, solve the recursion
$V^{(n)}(i \cdot \frac{T}{n}, S^{(n)}(i)) = \left[pV^{(n)}\{(i+1) \cdot \frac{T}{n}, uS^{(n)}(i)\} + (1-p)V^{(n)}\{(i+1) \cdot \frac{T}{n}, dS^{(n)}(i)\right] e^{-r\frac{T}{n}}$
4. let $\widehat{P}_n = V^{(n)}(0)$.

The function **Binsim** implements binomial tree pricing.

4.2.1 Convergence properties

Showing convergence of binomial tree methods is difficult. By application of Donsker's Theorem, Korn ([2]) shows that given the choices of u, d and p, for European options

$$\widehat{P}_n \longrightarrow E(e^{-rT}f(S_t) \text{ as } n \to \infty$$

Proving convergence for different types of options must be done on an individual basis.

Generally, binomial tree methods display faster rates of convergence than Monte Carlo methods. However, irregular and slow convergence can be observed for particular option parameters. To demonstrate this, nine plots of binomial tree estimates of price against increasing n were calculated using a European option with various $S_0,\ K,\ r$ and σ. The results are shown in figure 6. The two first plots in the upper row are typical convergence patterns for a given problem: either the solution converges with n to the exact result from one side (above or below), or it oscillates over the exact result. However, for high r or low σ, increasing n may not give better estimates, as the top right and bottom left plots show.

4.2.2 Alternative Binomial Tree models

The Cox-Ross-Rubinstein model is the simplest in the family of binomial tree models but performs well. Several alternative models have been proposed and display better convergence properties. These include

- Trinomial tree methods
  - Uses 3 branches at each stem instead of 2. Results in faster convergence.
- Rogers and Stapleton Pathwise approach
  - Uses so-called infinite binomial trees, where each path of $S_t$is modelled by a step function that is only allowed to attain values in a given discrete set.
  - Avoids the "saw-tooth" convergence plots as seen in figure 6.
- Hybrid Monte Carlo/binomial tree methods
  - Involves simulation of a sufficiently large number of asset price paths in the binomial method, and then use the arithmetic mean over the corresponding $V^{(n)}$ as an approximation for the option price.

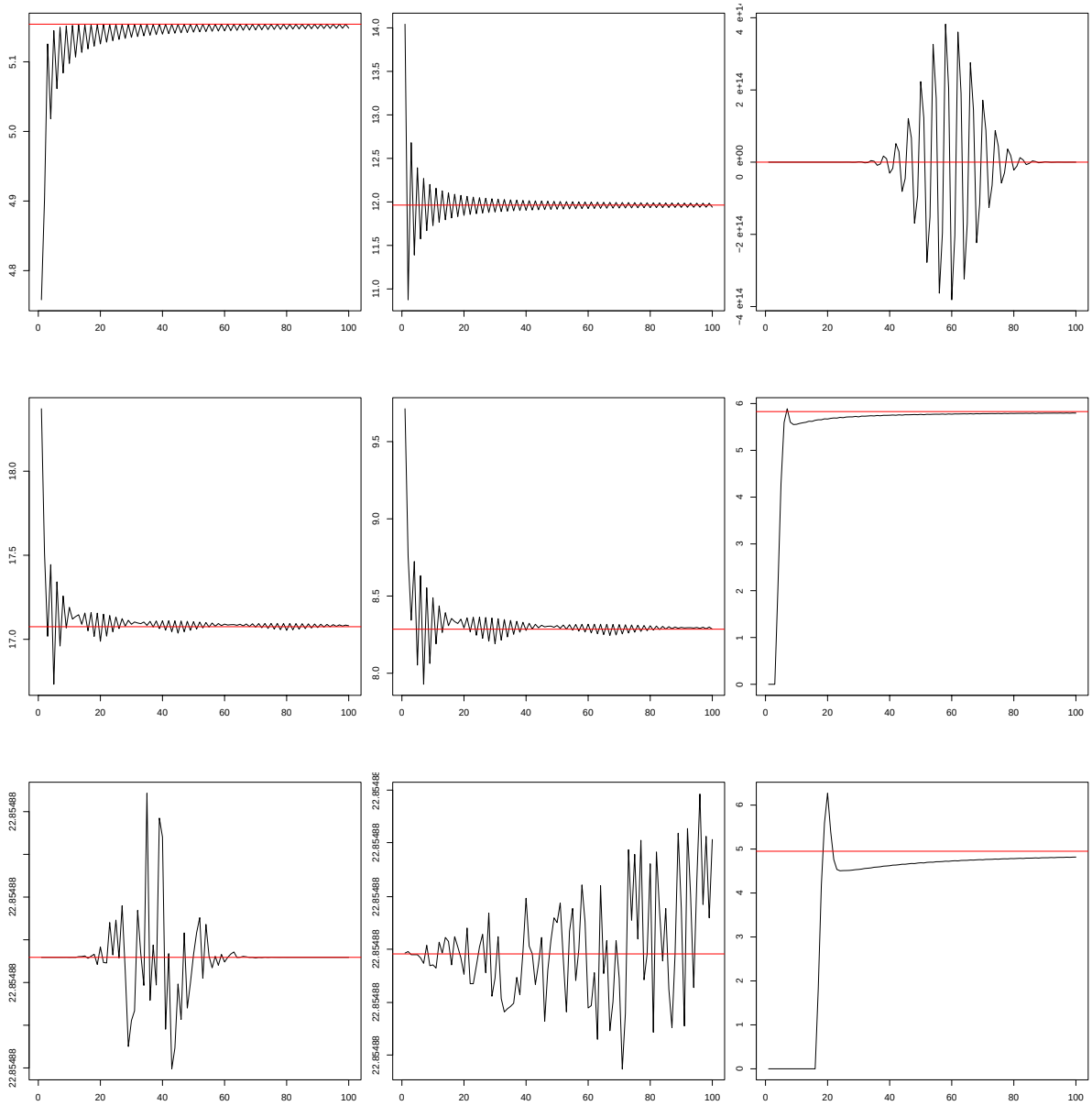


Figure 6: BT price estimates for various European options with increasing n. The horizontal line is the exact result. From top to bottom, left to right, the option types in format (S, K, T, r, σ) are: (50,50,1,0.1,0.1); (50,50,1,0.1,0.5); (50,50,1,0.5,0.03); (50,40,1,0.1,0.5); (50,60,1,0.1,0.5); (50,60,1,0.3,0.1); (50,30,1,0.1,0.01); (50,30,1,0.1,0.05); (50,75,1,0.5,0.1).

5 Results

The table below gives estimated prices for European call and put options for various K and n. For the Monte Carlo estimates, n is the number of simulations. For binomial tree estimates, n is the number of branches. Note that $n = 10000$ Monte Carlo simulations takes approximately the same time as a binomial tree simulation with $n = 1000$: both are about 6 seconds. The following option parameters were used: $T = 1$, $r = 0.1$, $\sigma = 0.1$.

European	MC			BT			Black-Scholes
Call	$n = 100$	$n = 1000$	$n = 10000$	$n = 10$	$n = 100$	$n = 1000$	
$K = 90$	18.5686	18.6402	18.6343	18.6338	18.6306	18.6308	18.6308
$K = 110$	3.9276	4.088	4.2274	4.1210	4.2017	4.2143	4.2167
Put							
$K = 90$	0.05739	0.0906	0.0560	0.0692	0.0660	0.0662	0.0662
$K = 110$	4.0562	3.7480	3.7644	3.6531	3.7338	3.7464	3.7488
Asian (call)							
$K = 90$	5.2806	5.6053	5.3459				
Average (call)							
$K = 90$	14.3049	13.9523	13.62967				

We can see that for European call and put options, the binomial tree approximations converge to the exact Black-Scholes formula result faster and have smaller errors.

6 Conclusion

This report serves as an introduction to option pricing by Monte Carlo and binomial tree methods. It treats in detail both the theory and the software implementation aspects of either method.

The main results include: showing empirically that the Cox-Ross-Rubinstein binomial tree model outperforms Monte Carlo methods in pricing European options; investigating the convergence properties of binomial tree estimates and implementing both numeric methods into a comprehensive software package, *POP* (Pollard Option Pricing).

Invariable, I did not have time to do all that I wanted to in this project. Possible future work and extensions to this project include:

- Proving the derivation of the Black-Scholes formula by Ito's lemma;
- Adding Asian and fixed-strike average option types to the binomial tree function Binsim;
- Further investigating stochastic volatility models and implementing this in the binomial tree method;
- Looking at the "market-time" approach to stochastic volatility.

A Appendix

A.1 Constructing Monte Carlo confidence intervals

Proposition (Number of Monte Carlo samples) Let X be an unknown distribution with mean $E\{X\}$ and variance σ^2, and $f(X)$ with mean $\mu = E\{f(X)\}$. Suppose we estimate μ by

$$\bar{X}_n = \frac{1}{n}\sum_{j=1}^{n} f(x_j), \quad x_1, ..., x_n \sim \text{i.i.d } X,$$

the Monte Carlo estimator. Then for a relative precision level $\varepsilon = \left|\frac{\bar{X}_n - \mu}{\mu}\right|$ and confidence level α, and where $\xi = F^{-1}\left(\frac{1+\alpha}{2}\right)$, the required number of samples is

$$n \doteq \frac{\sigma^2 \xi^2}{\mu^2 \varepsilon^2}.$$

Proof We wish to construct a confidence interval for $\bar{X}_n$ as a function of a significance level α, a relative precision level $\varepsilon > 0$ and n. Let ε be a bound for the percentage difference between $\bar{X}_n$ and μ, i.e.,

$$\left|\frac{\bar{X}_n - \mu}{\mu}\right| \leq \varepsilon \tag{4}$$

Let α denote the the probability of our estimate being within ε

$$P\left\{\left|\frac{\bar{X}_n - \mu}{\mu}\right| \leq \varepsilon\right\} = \alpha \tag{5}$$

Using $\bar{X}_n = \frac{1}{n}\sum_{j=1}^{n} f(x_j)$, this can be written as

$$\begin{aligned} \alpha &= P\left\{-\varepsilon \leq \frac{\frac{1}{n}\sum_{j=1}^{n} f(x_j) - \mu}{\mu} \leq \varepsilon\right\} \\ &= P\left\{-\varepsilon \leq \frac{1}{\mu\sqrt{n}} \frac{\sum_{j=1}^{n}\{f(x_j) - \mu\}}{\sqrt{n}} \leq \varepsilon\right\} \end{aligned}$$

We will use the central limit theorem to approximate this probability.

Central Limit Theorem *Let $f_1, f_2, ..., f_n$ be a sequence of i.i.d. random variables with distribution F whose mean is μ and variance is $\sigma^2 < \infty$. Then*

$$\lim_{n\to\infty} \frac{1}{\sqrt{n}} \sum_{j=1}^{n}\{f_j - \mu\} \sim N(0, \sigma^2).$$

Let Z_{σ^2} be a random variable distributed by $N(0, \sigma^2)$. Then asymptotically we have

$$\frac{1}{\mu\sqrt{n}} \frac{\sum_{j=1}^{n}\{f(x_j) - \mu\}}{\sqrt{n}} \to \frac{Z_{\sigma^2}}{\mu\sqrt{n}} \sim N\left(0, \frac{\sigma^2}{\mu^2 n}\right)$$

and an approximate confidence interval is given by

$$\alpha \doteq P\left\{-\varepsilon \leq \frac{Z_{\sigma^2}}{\mu\sqrt{n}} \leq \varepsilon\right\} \tag{6}$$

Now consider some number ξ that solves $\alpha = P(-\xi < Z_1 < \xi)$ where $Z_1 \sim N(0,1)$ with distribution function $F(x)$. By symmetry of the normal distribution, $\alpha = F(\xi) - F(-\xi)$ implying

$$\xi = F^{-1}\left(\frac{1+\alpha}{2}\right)$$

where F^{-1} is the standard normal quantile (inverse) distribution function. Since $P(-\xi < Z_1 < \xi) = P(-\sigma\xi < Z_{\sigma^2} < \sigma\xi)$, we have

$$\alpha = P\left\{\frac{-\sigma\xi}{\mu\sqrt{n}} \leq \frac{Z_{\sigma^2}}{\mu\sqrt{n}} \leq \frac{\sigma\xi}{\mu\sqrt{n}}\right\} \tag{7}$$

Comparing (6) and (7), we see that $\varepsilon \doteq \frac{\sigma\xi}{\mu\sqrt{n}}$ and solving for n gives

$$n \doteq \frac{\sigma^2\xi^2}{\mu^2\varepsilon^2}$$

■

Typically, we do not know σ^2 or μ, however these may be estimated by $S^2 = \sum_{j=1}^{n}\left(x_j - n^{-1}\sum_{j=1}^{n} x\right)^2$ and $\bar{X}$ respectively. Thus an estimate for n is

$$\widehat{n} = \frac{S^2\xi^2}{\bar{X}^2\varepsilon^2} \tag{8}$$

Example Suppose that we wish to estimate the price of an option with properties

$$\begin{aligned} S &= 100 \\ K &= 100 \\ T &= 1 \\ r &= 0.1 \\ \sigma\ (vol.) &= 0.1 \end{aligned}$$

A trial Monte Carlo simulation with $n = 1000$ gives us that

$$\begin{aligned} \bar{X} &= 10.31 \\ S^2 &= 53.37 \end{aligned}$$

also, $\xi = F^{-1}(0.975) = 1.96$. Plugging this into (8) gives

$$\begin{aligned} \widehat{n} &= \frac{53.37 \times 1.96^2}{10.31^2 \times 0.01^2} \\ &= 1.03 \times 10^6 \end{aligned}$$

While this seems like a tremendous number of simulations, the function **MCsim** took 3.03 seconds on a standard ANU Library Dell P4 to price the option using this n. The function **ntrial** in the programming appendix evaluates $\widehat{n}$.

B Software Implementation

This section demonstrates use of the *R*-based *POP*[2] (*Pollard Option Pricing*) function package, written for the purposes of this project. *POP* is a collection of efficient and reliable functions that can be used to:

- Perform Monte Carlo (MC) simulation to price European, Asian and fixed-strike average options with both put and call types.
- Perform binomial tree (BT) approximation to price European and American options (put/call).
- Plot binomial tree model solutions.
- Generate asset price paths by geometric Brownian motion with constant and stochastic volatility.
- Provide error analysis for both MC and BT pricing results.
- Give Black-Scholes formula option prices for European options.

A summary of each function is given below.

		Operation
	BSeuro	European option pricing (call/put) by Black-Scholes formula
	gen.brown	Generates geometric Brownian motion and asset-price paths
	MCsim	Performs Monte Carlo simulation
	MCstocsim	Performs Monte Carlo simulation using stochastic volatility
Functions	**ntrial**	Gives number of trials for given error tolerance and confidence
	Binsim	Performs Binomial Tree approximation
	BinsimTree	Similar to **Binsim**, gives *matrix* form of tree.
	TreePlot	Plots binomial tree from **BinsimTree** matrix output
	BinsimOpt	Better (∼10 times) estimation than **Binsim**, ∼2 times slower.
	BinsimError	Plots binomial tree estimates for increasing n, plots exact result

The following examples demonstrate use of the package.

B.1 Examples

Pricing an European option by Monte Carlo simulation Suppose we want to price a call option with following parameters by Monte Carlo simulation:

type	S_0	K	T	r	σ	n
call	100	100	1	0.1	0.1	10^6

Using the function **MCsim** with $n = 10^6$, the relevant function call in *R* is

```
> MCsim(S=100,K=100,T=1,r=0.1,sigma=0.1,n=10^6,type="ce")
```

[2]The code for all functions is given in the appendix. Requires a version of *R* greater or equal to 1.8.0.

```
Estimate 10.31083
BS exact 10.30815
variance 81.98794
error 0.00267912
```

The price estimate is \$10.31083, and the exact price using Black-Scholes' formula is \$10.30815. The simulation took 3.1 seconds on a 2.0ghz Pentium 4. The error in difference is 0.00267912 and the sample variance of price distribution is 81.987.

Now suppose we wish to price the same option, now as a put type. The function call is

```
> MCsim(100,100,1,0.1,0.1,1000000,"pe")

$Param
[,1]
type "ce"
S "100"
K "100"
Time "1"
r "0.1"
sigma "0.1"
n "100"

Estimate 0.792979617
BS exact 0.791892729
variance 5.508835839
error 0.001086888
```

The estimate is $\widehat{P}$ =\$0.79298 and there is a 0.00109 difference between Black-Scholes' price and this estimate. The "variance" entry is the sample variance of distribution of payoff V. Recall that $Var(\bar{X}_n) = Var(X)/n$: here $\bar{X}_n = \widehat{P}$ and X is V. The variance of $\widehat{P}$ is thus given by

$$Var(\widehat{P}) = \frac{5.508}{10^6} = 5.508 \times 10^{-6}.$$

Performing MC error analysis Suppose we want an MC estimate within 5% of the true result with a confidence level $\alpha = 0.95$. An initial MC simulation gives an approximate price of $P_T = \$20$ and the variance of distribution V (as given in the output of **MCsim)** is 60. The function **ntrial** is used and the call is

```
> ntrial(0.05,0.95,20,60)
[1] 13829
```

Thus n should be set at approximately 13800 or more.

Now suppose we want a confidence interval for the MC estimate of P_T to infer the estimate accuracy, as given in example 1. The appropriate call is

```
> MCsim(100,100,1,0.1,0.1,1000000,"ce",boot=T)$boot

$"Bootstrap Conf Int."
    alpha      abc        stan
[1,] 0.025 9.895368*   9.892928
[2,] 0.050 9.957220   9.956335
[3,] 0.100 10.028862 10.029439
[4,] 0.160 10.085729 10.087208
[5,] 0.840 10.485874 10.487417
[6,] 0.900 10.544539 10.545186
[7,] 0.950 10.619109 10.618290
[8,] 0.975 10.684088* 10.681697
```

For a two sided non-parametric confidence interval with $\alpha = 0.95$, the lower bound is given by 9.8953 (denoted by *: abc column, "alpha"=0.025) and the upper bound is given by 10.6841 (*: abc column, "alpha"=0.975). Hence, with probability of 0.95, the true price P_T lies in the interval

$$0.98953 < P_T < 10.6841$$

Note that the right column ("stan") is for comparison purposes and are standard (parametric) confidence intervals.

Pricing an European option by BT approximation Consider an European option where

type	S_0	K	T	r	σ	n
call	50	60	1	0.1	0.5	1000

The appropriate call is given by

```
> Binsim(50,60,1,0.1,0.5,1000,"ce")

$Param
[,1]
type "ce"
S "50"
K "60"
Time "1"
r "0.1"
sigma "0.5"
n "1000"

$Result
```

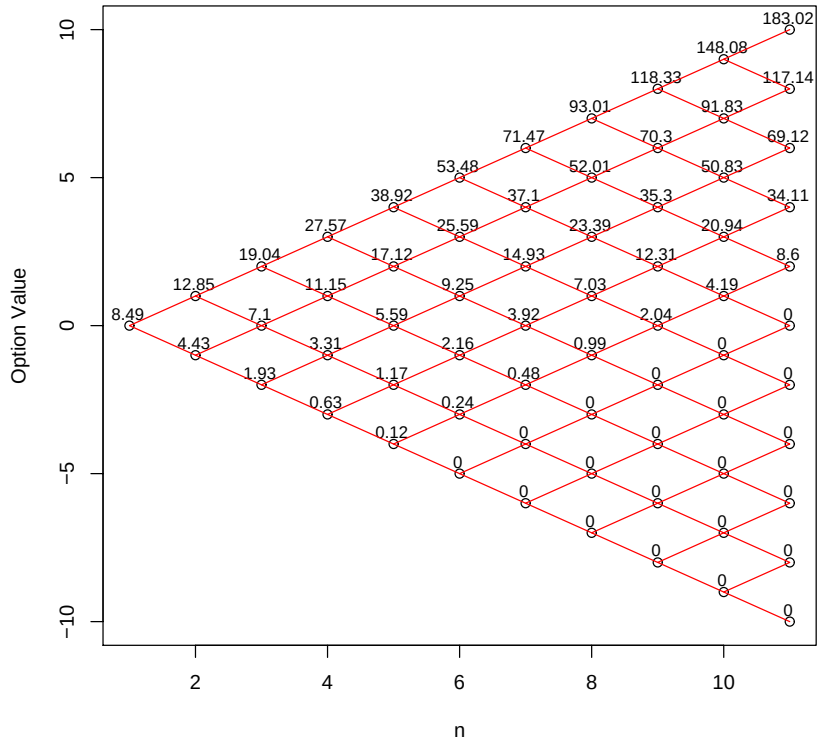


Figure 7: Binomial tree with $n = 10$ branches, $S = 50,\ K = 50,\ T = 1,\ r = 0.1,\ \sigma = 0.5$.

```
[,1]
Option Value 8.2859286161
BS Exact 8.2852208045
Error 0.0007078116
```

The estimated option price is 8.2859 and differs from the Black-Scholes' result by 7.1×10^{-4}. It took 2.5 seconds to perform on a 2.0ghz P4.

The binomial tree used in estimating the above price can be drawn using the function **BinsimTree**. Let $n = 10$, so that the tree is readable. The appropriate call is

```
>Binsim(50,60,1,0.1,0.5,10,"ce")
```

The output is shown in figure 7.

Performing BT error analysis To produce a convergence plot with increasing n, use the function **BinsimError**. The appropriate call, using the previous option parameters and $n = 100$, is

```
>BinsimError(50,60,1,0.1,0.1,100,''ce'')
```

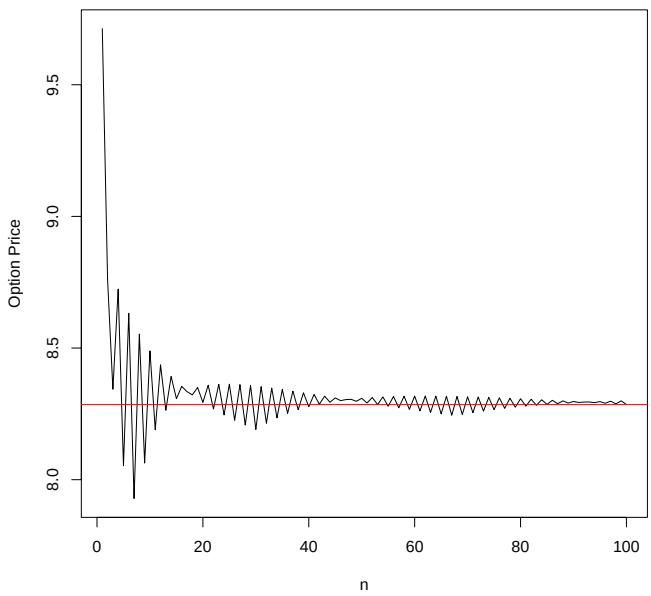


Figure 8: Convergence plot for $1 < n < 100$ branches, $S = 50,\ K = 50,\ T = 1,\ r = 0.1,\ \sigma = 0.5$.

The output is shown in figure 8. We see the estimate converges to the exact result by periodic pulsation and that for $n > 80$, the rate of convergence quickly drops off. Thus with the selection of $n = 1000$ in the previous example, we can be fairly confident that the result is accurate. Of course in this example, we *know* what the exact result is by Black-Scholes formula and can measure the error directly. However, for Asian and fixed-strike average options, we do not know the exact result and looking at convergence rates is a method of inferring the error.

Pricing a European option by MC with stochastic volatility[3] Consider an European option where

type	S_0	K	T	r	$\bar{\sigma}$	n
call	100	100	1	0.1	0.1	10^4

and where volatility is stochastic with mean $\bar{\sigma} = 0.1$. Suppose volatility is modelled by a geometric random walk process, where

$$\sigma_t = \prod_{j=1}^{t-1} \overline{\sigma}(1 + \varepsilon_j)\ \ \varepsilon_j \sim \text{i.i.d}\ \mathcal{N}(0,1)$$

The appropriate call is given by

```
> MCstocsim(100,100,1,0.1,0.1,10000,"ce","geostoc")

[,1]
Estimate 10.895550
```

[3]Stochastic volatility implementation is experimental.

```
BS exact 10.308151
variance 120.187439
```

We see that $\widehat{P} = 10.89$ which is greater than the Black-Scholes result, where volatility is assumed constant.

C Program Code

This section presents all the *R*-based functions written for the report. The two main functions are **MCsim** and **Binsim** which implement Monte Carlo and binomial tree methods respectively.

MCsim values European, Asian, and fixed-strike average call and put options. It calls functions **gen.brown** and **BSeuro**. **gen.brown** is the core of the function and generates asset price-paths used in the simulation.

Binsim values European and American call and put options. It calls **BSeuro**. The function **BinsimOpt** uses local averaging to improve estimates, **BinsimTree** performs the same task as **Binsim** but stores each branch price in a large matrix that is used for plotting the binomial tree. **TreePlot** is a small graphical function that plots the matrix output from **BinsimTree**.

	Argument	**Description**
	S	initial asset price S_0
	BSeuro	strike price
	T	period until maturity in years
	r	annual riskless interest rate
Global arguments	*sigma*	percentage annual volatility
	N	number of MC simulations or binomial branches
	n	length of price pathway for path-dependent options
	type	Type of option: string *"ce"/"pe"* for call/put European, *"cas"/"pas"* for call/put Asian, "*cav"/"pav"* for call/put average and *"ca"/"pa"* for call/put American

C.1 General functions

C.1.1 BSeuro

Uses the Black-Scholes formula to calculate call and put European option value.

```
BSeuro<-function(price,strike,period,r,sigma,type=c("ce","pe")){

if (type!="ce" && type!="pe")
   stop("type misspecified: ce|pe")

d1<-(log(price/strike)+(r+1/2*sigma^2)*(period))/(sigma*sqrt(period))
d2<-(d1-sigma*sqrt(period))

if (type=="ce")
   value<-(price*pnorm(d1,mean=0,sd=1)-strike*exp(-r*(period))*pnorm(d2,mean=0,sd=1))
if (type=="pe")
  value<-(-price*pnorm(-d1,mean=0,sd=1)+strike*exp(-r*(period))*pnorm(-d2,mean=0,sd=1))

result<-list(rbind("period"=period,"current price"=price,"strike"=strike,
```

```
+"interest rate"=r,"volitility"=sigma,"type"=type),"Option Value"=value)

return(result)
}
```

C.1.2 gen.brown

Generates approximate geometric Brownian motion of length n. Type="geo" gives a whole price path, type="jump" gives terminal prices, type="sumstoc" gives arithmetic random walk price paths, and type="geostoc" gives geometric random walk price paths.

```
gen.brown<-function(S,T,n,r,sigma,type="geo",var=1,anti=T){

if (type=="geo")
  return(cumprod(c(S,(1+r*T/n+sigma*sqrt(T/n)*rnorm(n-1)))))

if (type=="jump"){
   if (anti==T){
      a<-rnorm(round(n/2))
      return(S*exp((r-sigma^2/2)*T+sigma**sqrt(T)*c(a,-a)))}
   else return(S*exp((r-sigma^2/2)*T+sigma*sqrt(T)*rnorm(n)))
   }

if (type=="sumstoc"){
   stvol<-cumsum(rnorm(n-1,sd=sqrt(var)))
   stvol<-(stvol-mean(stvol)+sigma)
   return(cumprod(c(S,(1+r*T/n+stvol*sqrt(T/n)*rnorm(n-1)))))
   }

if (type=="geostoc"){
   stvol<-cumprod(c(sigma,(1+var*sqrt(T/n)*rnorm(n-2))))
   return(cumprod(c(S,(1+r*T/n+stvol*sqrt(T/n)*rnorm(n-1)))))}
   }
}
```

C.2 Monte Carlo Simulation

C.2.1 MCsim

Performs Monte Carlo simulation. Boot="TRUE" outputs bootstrap confidence intervals.

```
MCsim<-function(S,K,T,r,sigma,N,type=c("ce","cas","cav","pe","pas","pav"),boot="FALSE"){

z=NA
if (type == "ce" || type == "cas" || type == "cav")
```

```
    z = +1
if (type == "pe" || type == "pas" || type == "pav")
    z = -1
if (is.na(z))
    stop("type misspecified: ce|cav|cas|pe|pav|pas")

if (type == "ce" || type == "pe"){
    sim<-gen.brown(S,T,N,r,sigma,type="jump",anti=T)
    R<-exp(-T*r)*pmax(z*(sim-K),0)}

else {
    n<-round(N/3)
    sim<-matrix(nrow=N,ncol=n)
    for (i in 1:N)
        sim[i,]<-(gen.brown(S,T,n,r,sigma,type="geo"))
    R<-1:N

    if (type == "cas" || type == "pas"){
        for (i in 1:N)
        R[i]<-exp(-T*r)*max(z*(sim[i,n]-mean(sim[i,])),0)
        }

    if (type == "cav" || type == "pav"){
        for (i in 1:N)
        R[i]<-exp(-T*r)*max(z*(mean(sim[i,])-K),0)
        }
    }

plot(density(R))

if (type == "ce" || type == "pe"){
    BS<-BSeuro(S,K,T,r,sigma,type)$O
    summary<-rbind(
    "Estimate"=mean(R),
    "BS exact"=BS,
    "variance"=var(R),
    "bias"=(mean(R)-BS))}

else {
    summary<-rbind(
    "Estimate"=mean(R),
    "variance"=var(R))}
```

```
    if (boot=="TRUE"){
        sampmean<-function(p,x) {sum(p*x)/sum(p)}
        if (length(R)<=2000)
            conf<-abcnon[4](R,sampmean)$limit
        else
            conf<-abcnon(R[1:2000],sampmean)$limit
        summary<-list(summary,"Bootstrap Conf Int."=conf)
        }

    return(summary)
}
```

C.2.2 MCstocsim

Performs Monte Carlo simulation for the two "sumstoc" (arithmetic r.w.) and "geostoc" (geometric r.w.) stochastic volatility models.

```
    MCstocsim<-function(S,K,T,r,sigma,N,type,stoctype="sumstoc",boot="FALSE"){

        z=NA
        if (type == "ce" || type == "cas" || type == "cav")
                z = +1
            if (type == "pe" || type == "pas" || type == "pav")
                z = -1
        if (is.na(z))
            stop("type misspecified: ce|cav|cas|pe|pav|pas")

        n<-round(N/3)
        sim<-matrix(nrow=N,ncol=n)
        for (i in 1:N)
            sim[i,]<-(gen.brown(S,T,n,r,sigma,type=stoctype))
        R<-1:N

        if (type == "ce" || type == "pe"){
            for (i in 1:N)
                R[i]<-exp(-T*r)*max(z*(sim[i,n]-K),0)
            }

        if (type == "cas" || type == "pas"){
            for (i in 1:N)
                R[i]<-exp(-T*r)*max(z*(sim[i,n]-mean(sim[i,])),0)
```

[4] **abcnon** part of the R *bootstrap* library. Generates non-parametric bootstrap confidence intervals for the sample mean.

```
        }

    if (type == "cav" || type == "pav"){
        for (i in 1:N)
            R[i]<-exp(-T*r)*max(z*(mean(sim[i,])-K),0)
        }

    plot(density(R))
    if (type == "ce" || type == "pe"){
        BS<-BSeuro(S,K,T,r,sigma,type)$O
        summary<-rbind(
        "Estimate"=mean(R),
        "BS exact"=BS,
        "variance"=var(R),
        "bias"=(mean(R)-BS))}
    else {
        summary<-rbind(
        "Estimate"=mean(R),
        "variance"=var(R))}

    if (boot=="TRUE"){
        sampmean<-function(p,x) {sum(p*x)/sum(p)}
        if (length(R)<=2000)
            conf<-abcnon(R,sampmean)$limit

        else
            conf<-abcnon(R[1:2000],sampmean)$limit
            summary<-list(summary,"Bootstrap Conf Int."=conf)
        }
    return(summary)
}
```

C.2.3 ntrial

Gives a lower bound for the number of trials required to obtain accuracy *epsilon* for confidence *alpha* and estimated option value *price*. *variance* is the variance of V (payoff) distribution, and is given in MCsim.

```
ntrial<-function(epsilon,alpha,price,variance){
    zi<-qnorm((1+alpha)/2)
    return(round((variance^2*zi^2/(price^2*epsilon^2))))
}
```

C.3 Binomial Tree Approximation

C.3.1 Binsim

Performs binomial tree approximation using the CRR model. Type is either "ce"/"ca"/"pe"/"pa" (European and American call and put).

```
Binsim<-function(S,K,T,r,sigma,n,type){

    z=NA
    if (type == "ce" || type == "ca")
        z = +1
      if (type == "pe" || type == "pa")
        z = -1
    if (is.na(z))
      stop("type misspecified: ce|ca|pe|pa")

    dt = T/n
    u = exp(sigma * sqrt(dt))
    d = 1/u
    p = (exp(r * dt) - d)/(u - d)
    Df = exp(-r * dt)
    OptionValue = z * (S * u^(0:n) * d^(n:0) - K)
    OptionValue = (abs(OptionValue) + OptionValue)/2

    if (type == "ce" || type == "pe") {
      for (j in seq(from = n - 1, to = 0, by = -1))
        for (i in 0:j)
          OptionValue[i+1] = (p * OptionValue[i + 2] + (1 - p) * OptionValue[i+1]) * Df
      }

    if (type == "ca" || type == "pa") {
      for (j in seq(from = n - 1, to = 0, by = -1))
        for (i in 0:j)
          OptionValue[i+1] = max((z * (S * u^i * d^(abs(i - j)) - K)),
          + (p * OptionValue[i + 2]+(1 - p) * OptionValue[i + 1]) * Df)
      }

    BS<-BSeuro(S,K,T,r,sigma,type=type)$O

    result<-list("Param"=rbind(
      "type" = type,
      "S" = S,
      "K" = K,
```

```
        "Time" = T,
        "r" = r,
        "sigma" = sigma,
        "n" = n),"Result"=rbind(
        "Option Value"=OptionValue[1],
        "BS Exact"=BS,
        "Bias"=(OptionValue[1]-BS)))

    return(result)
}
```

C.3.2 BinsimTree

Plots binomial tree representation. The algorithm here is the same as in Binsim, but more memory demanding as it stores all $V(t)$ in a large (n by n) matrix. Argument "cex" controls size of text in plot. Calls TreePlot to do the graphics.

```
BinsimTree<-function(S,K,T,r,sigma,n,type=c("ca","ce"),cex=0.8,matrix=F){

    z=NA
    if (type == "ce")
        z = +1
    if (type == "pe")
        z = -1
    if (is.na(z))
    stop("type misspecified: ce|ca")
    dt = T/n
    u = exp(sigma * sqrt(dt))
    d = 1/u
    p = (exp(r * dt) - d)/(u - d)
    Df = exp(-r * dt)
    OptionValue = z * (S * u^(0:n) * d^(n:0) - K)
    Tree=OptionValue = (abs(OptionValue) + OptionValue)/2
    offset = 1

    for (j in (n - 1):0) {
    Tree <- c(Tree, rep(0, times = n - j))
        for (i in 0:j) {
            OptionValue[i + offset] = (p * OptionValue[i + 1 + offset]
            + (1 - p) * OptionValue[i + offset]) * Df
            Tree = c(Tree, OptionValue[i + offset])
            }
    }
```

```
    Tree = matrix(rev(Tree), byrow = FALSE, ncol = n + 1)
    TreePlot(Tree,cex=cex)

    result<-list("param"=rbind(
        "type" = type,
        "S" = S,
        "K" = K,
        "Time" = T,
        "r" = r,
        "sigma" = sigma,
        "n" = n),"result"=rbind(
        "Option Value"=Tree[1,1],
        "BS Exact"=BSeuro(S,K,T,r,sigma,type=type)$O))

    if (matrix==T)
        result<-c(result,list("Tree"=Tree))

    return(result)
}
```

C.3.3 TreePlot

Draws a binomial tree given a n by n matrix ("Tree"). Argument dx and dy control positioning of test, cex controls text size and digits gives total number of decimal places.

```
TreePlot<-function(Tree, dx = -0.025, dy = 0.4, cex = 0.8, digits = 2, ...){

    Tree = round(Tree, digits = digits)
    depth = ncol(Tree)
    plot(x = c(1, depth), y = c(-depth + 1, depth - 1), type = "n",
        col = 0, ...,xlab="n",ylab="Option Value")
    points(x = 1, y = 0)
    text(1 + dx, 0 + dy, deparse(Tree[1, 1]), cex = cex)

    for (i in 1:(depth - 1)) {
        y = seq(from = -i, by = 2, length = i + 1)
        x = rep(i, times = length(y)) + 1
        points(x, y, col = 1)
            for (j in 1:length(x)) text(x[j] + dx, y[j] + dy, deparse(Tree[length(x) +
            1 - j, i + 1]), cex = cex)
        y = (-i):i
        x = rep(c(i + 1, i), times = 2 * i)[1:length(y)]
        lines(x, y, col = 2)
```

```
        }

    invisible()
}
```

C.3.4 BinsimOpt

Uses the Black-Scholes formula to calculate call and put European option value.

```
BinsimOpt<-function(S,K,T,r,sigma,n,type=c("ca","ce"),p=2){

    R<-1:p
    for (i in 0:p)
        R[i]<-Binsim(S,K,T,r,sigma,n+i,type)$Result[1]

    result<-mean(R)
    BS<-BSeuro(S,K,T,r,sigma,type=type)$O

    summary<-list("Param"=rbind(
        "type" = type,
        "S" = S,
        "K" = K,
        "Time" = T,
        "r" = r,
        "sigma" = sigma,
        "n" = n),"Result"=rbind(
        "Option Value"=result,
        "BS Exact"=BS,
        "Bias"=(mean(R)-BS)))

    return(summary)
}
```

C.3.5 BinsimError

Plots estimated price using BT method against increasing n. Used to generate plots for BT convergence analysis section.

```
BinsimError<-function(S,K,T,r,sigma,n,type){

    result<-1:n
    for (i in 1:n)
        result[i]<-Binsim(S,K,T,r,sigma,i,type)$Result[1,1]
```

```
    plot(result,type="l",xlab="n",ylab="Estimate")
    abline(h=BSeuro(S,T,K,r,sigma,type)$O,col=2)
}
```

References

[1] Cox, J., Ross, A. & Rubinstein, M., 1979, *Option pricing: a simplified approach*, Journal of Financial Economics, Vol 7, pp. 229-263.

[2] Korn, R., 2001, *Option Pricing and Portfolio Optimisation*, American Mathematical Society, Chapter 3 and 4, pp. 79-134, 153-201.

[3] Shafer, G. & Vovk, V., 2001, *Probability and Finance*, Wiley Series in Statistics, Wiley & Sons.

[4] Black, F. & Scholes, M., 1973, *The Pricing of Options and Corporate Liabilities*, Journal of Political Economy, Vol 81, pp. 637-654.

[5] Abken, P. & Nandi. S, 1999, *Options and Volatility*, Federal Reserve Bank of Atlanta, Economic Review 95.

[6] Silverman, D., 1999, *Solution to the Black-Scholes Equation Using the Green's Function of the Diffusion Equation*, University of California.

[7] Chang, J., 1999, *Brownian Motion*, Baskin School of Engineering, http://www.cse.ucsc.edu/˜chang/203/bm.pdf.

[8] Graveline, J. 2000, *Path-Dependent Asian Options*, University of Waterloo, http://www.stanford.edu/˜jjgravel/ research/working_papers/pathdependent.pdf.