Monte Carlo and the American Put

Antoine Tambue (antonio@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by Professor David Taylor and Nadia Uys
Programme in Advanced Mathematics of Finance
University of the Witwatersrand

June 8, 2007

Abstract

Pricing a derivative security entails calculating the expected discounted value of its payoff. This reduces, in principle, to a problem of numerical integration; but in practice this calculation is often difficult for high-dimensional pricing problems. High-dimensionality arises in pricing options on multiple underlying assets and in pricing options in models that capture many sources of risk, such as stochastic volatility, interest rates and exchange rates. In this essay we are going to present Monte Carlo Methods, whose distinct advantage is that its convergence rate is typically independent of the number of state variables, and its applications to pricing American put options.

Contents

Introduction

The risks attached to any operation of stock exchange have led to imagine systems allowing to limit them. So, have developed the market of options, where the buyer acquires not the value itself, but the right to buy or to sell. For example a European call or put option allows the holder to exercise the option only at the option expiration date. An American option is different to the European option in the sense that the holder can exercise the option at any time between the initial and expiration date.

To have the right to buy or to sell the underlying asset at a fixed price, the buyer immediately pays to the seller the initial option price called premium. Premium for European option is generally easy to evaluate (see [1]). The fact that American options can be exercised at any time before the option's expiry date causes great difficulty in the pricing methodology. Although there are many techniques for pricing American options on a single underlying asset including lattices, PDE methods, finite difference methods, variational inequalities, and integral equation methods, when these techniques are generalised to handle multiple state variables, they require work which is exponential in the number of state variables (see [2]). This requirement renders these methods ineffective for more than about three or four state variables, then pricing high-dimensional options is further complicated for American versions.

Until 1993, it was generally believed that the Monte Carlo methods, the most attractive method to solve problems in high dimensions, were unsuitable for pricing American put options. This essay focuses, on Monte Carlo theory and its applications for pricing American put options. It is divided into three chapters.

Chapter one contains the mathematical tools in probability theory, stochastic processes and asset pricing such as probability space, Brownian motion, Itô formula and derivative security.

Chapter two presents Monte Carlo theory. In the first part we present the principles of Monte Carlo methods covering the description of the method, Monte Carlo integration, limits and convergence of Monte Carlo methods. In addition we prove the theorem of approximation of the variance by the empirical variance of the sample in Monte Carlo methods. Afterward we apply Monte Carlo methods to European call option (American call option on a stock that pays no dividends). We implement some Python simulations to ensure the convergence of the method.

Chapter three is the most interesting one. After formulating the American put option problem within Monte Carlo framework, we present Tilley's Bundling Algorithm (1993); The Grant, Vora and Weeks Methods (1996); and State Space Partitioning Algorithm (1995) for pricing American put. Once again, Python simulations ensure the convergence of some of them.

1. Preliminaries

In this chapter we are going to define some concepts and important results that we will use later in this essay. These results are essentially from probability theory, stochastic proccesses and asset pricing.

1.1 Some Results in Probability Theory and Stochastic processes

In this section our goal is to collect some definitions and results from probability theory. Further details and proofs for these results can be found in [3],[1],[4] and [5]. Throughout this section, Ω is the set of "outcomes" and we deal with continuous random variables.

Definition 1.1. $[\sigma - \textbf{\textit{algebra}}]$
Let Ω be a non-empty set. A σ- algebra $\mathbb{F}$ on Ω is a family of subsets on Ω such that:

- *the empty set $\emptyset$ belongs to $\mathbb{F}$*
- *if A belongs to $\mathbb{F}$, then so does the complement A^C*
- *if $A_1, A_2, ...$ is a sequence of sets in $\mathbb{F}$, then their union $\bigcup_i A_i$ also belongs to $\mathbb{F}$*

Definition 1.2. [***Probability measure***]
Let $\mathbb{F}$ be a σ- algebra on Ω. A probability measure P is a function
$P : \mathbb{F} \longrightarrow [0, 1]$ *such that:*

- $P(\Omega) = 1$.
- *if $A_1, A_2, ...$ are pairwise disjoint sets*[1] *of $\mathbb{F}$ belonging to $\mathbb{F}$, then $P(\bigcup_i Ai) = \sum_i P(A_i)$.*

The triple (Ω,$\mathbb{F}$,P) is called a probability space.
Examples
- *The smallest σ- algebra is the collection $\mathbb{F} = \{\emptyset, \Omega\}$.*
- *If A is any subset of Ω then $\mathbb{F} = \{\emptyset, A, A^C, \Omega\}$ is a σ- algebra.*

In all this section (Ω,$\mathbb{F}$,P) is a probability space.

Definition 1.3. *Given any family U of subsets of Ω there is a smallest σ-algebra denoted by $\sigma(U)$ containing U. $\sigma(U)$ is called the σ-algebra generated by U.*

[1] $A_i \bigcap A_j = \emptyset$ for $i \neq j$

Remark 1.1. $\sigma(U) = \bigcap \{H, H\ \sigma - algebra\ \ of\ \ \Omega, U \subset H\}$.
If U is the collection of all open subsets of a topological space Ω (e.g. $\Omega = \mathbb{R}^n$), then $\mathbb{B} = \sigma(U)$ is called the Borel σ-algebra on Ω and the elements $B \in \mathbb{B}$ are called Borel sets.

Definition 1.4. [***Measurability***]
A function $X : \Omega \longrightarrow \mathbb{R}^n$ is $\mathbb{F}$- measurable if

$$X^{-1}(U) := \{\omega \in \Omega, X(\omega) \in U\} \in \mathbb{F}\ \ for\ \ all\ \ Borel\ \ sets\ U \subset \mathbb{R}^n \tag{1.1}$$

A random variable X is a $\mathbb{F}$-measurable function $X : \Omega \longrightarrow \mathbb{R}^n$.

Remark 1.2. *Every random variable induces a probability measure μ_X on $\mathbb{R}^n$ defined by*

$$\mu_X(B) = P(X^{-1}(B)) \tag{1.2}$$

μ_X is called the distribution of X.

Remark 1.3. *If there exist a positive function f such that $\mu_X(D) = \int_D f(x)dx$ for all subsets $D \subseteq \mathbb{R}^d$ then f is called the density function of μ_X.*

Definition 1.5. [***Expectation***]
Let X be a random variable such that $\int_\Omega |X(\omega)|dP(\omega) < \infty$ a.s [2]. The number

$$\mathbb{E}(X) := \int_\Omega X(\omega)dP(\omega) = \int_{\mathbb{R}^n} xd\mu_X(x) \tag{1.3}$$

is called the expectation of X.
More generally, if $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is Borel measurable and $\int_\Omega |f(X(\omega))|dP(\omega) < \infty$ a.s then

$$\mathbb{E}(f(X)) := \int_\Omega f(X(\omega))dP(\omega) = \int_{\mathbb{R}^n} f(x)d\mu_X(x) \tag{1.4}$$

is called the expectation of $f(X)$.

Definition 1.6. [***Independent random variables***]
n random variables $X_1, X_2, ..., X_n$ are independent if for all measurable positive functions $f_1, f_2, ..., f_n$ we have:

$$\mathbb{E}(f_1(X_1) \cdots f_n(X_n)) = \mathbb{E}(f_1(X_1)) \cdots \mathbb{E}(f_n(X_n)) \tag{1.5}$$

Definition 1.7. [***Conditional expectation***]
Let X be a random variable such that $\int_\Omega |X(\omega)|dP(\omega) < \infty$ a.s, and $\mathbb{G}$ a sub-σ- algebra of $\mathbb{F}$. The conditional expectation of X relative to the σ- algebra $\mathbb{G}$ is a random variable denoted by $\mathbb{E}(X \mid \mathbb{G})$ with the following properties

- $\mathbb{E}(X \mid \mathbb{G})$ *is $\mathbb{G}$-measurable.*
- $\int_G \mathbb{E}(X \mid \mathbb{G})dP = \int_G XdP$ *for all $G \in \mathbb{G}$.*

[2] a.s means almost surely

Remark 1.4. *According to the previous definition, we can notice that:*

- $\mathbb{E}(\mathbb{E}(X \mid \mathbb{G})) = \mathbb{E}(X)$;
- *if X is* $\mathbb{G}$*-measurable then* $\mathbb{E}(X \mid \mathbb{G}) = X$,
- *if X is independent of* $\mathbb{G}$ *then* $\mathbb{E}(X \mid \mathbb{G})) = \mathbb{E}(X)$ *a.s.;*
- $\mathbb{E}(X \mid \mathbb{G})$ *is the mean value of X over* $\mathbb{G}$*; it is also called the projection of X upon* $\mathbb{G}$.

Definition 1.8. [***Filtration***]
A filtration is an increasing sequence of sub-σ*-algebras* $(F_n)_{n\geq 0}$ *of* $\mathbb{F}$.[3] *We say that the sequence* $(X_n)_{n\geqslant 0}$ *of random variables is adapted to the filtration* $(F_n)_{n\geq 0}$ *if* X_n *is* F_n*-measurable for every* $n \geq 0$.
The tuple (Ω,$\mathbb{F}, (F_n)_{n\geq 0}$,*P) is called a filtered probability space.*

Remark 1.5. *We shall usually assume in our applications that* $F_0 = \{\emptyset, \Omega\}$ *and very often we shall assume that the final* σ*-algebra is generated by the whole sequence i.e* $F_\infty = \sigma(\bigcup_{n\geqslant 0} Fn)$.

Definition 1.9. [***Martingale***]
Let (Ω, $\mathbb{F}, (F_n)_{n\geq 0}$,*P) be a filtered probability space . A sequence of random variables* $(X_n)_{n\geq 0}$ *on (*Ω,$\mathbb{F}$,*P) is a martingale relative to the filtration* $F = (F_n)_{n\geq 0}$ *if:*

- $(X_n)_{n\geq 0}$ *is adapted to* F.
- *For each* $n \geq 0$, $\int_\Omega X_n dP < \infty$.
- *For each* $n \geq 0$, $\mathbb{E}(X_{n+1} \mid F_n) = X_n$.

Definition 1.10. [***Stochastic process***]
A collection $\{X_t, t \geqslant 0\}$ *of random variables defined on (*Ω,$\mathbb{F}$,*P) is called a stochastic process.*

Remark 1.6. *Let* $\{X_t, t \geqslant 0\}$ *be a stochastic process defined on (*Ω,$\mathbb{F}$,*P).*

- *For each* ω *the mapping* $t \longrightarrow X(t, \omega)$ *is the corresponding sample path.*
- *For each* t *the mapping* $\omega \longrightarrow X(t, \omega)$ *is* $\mathbb{F}$*-measurable.*

Definition 1.11. [***Brownian motion***]
A stochastic process $\{W_t, t \geqslant 0\}$ *is a one dimensional Brownian motion started at* 0 *if:*

- $W_0 = 0$ *a.s.*
- $W_t - W_s$ *is a normal random variable with mean* 0 *and variance* $t - s$ *whenever* $s < t$
- $W_t - W_s$ *is independent of the* σ*-algebra generated by* $\{W_r, r \leq s\}$

[3] i.e. $F_0 \subset F_1 \subset F_2 ... \subset F_n \subset ... \subset \mathbb{F}$

- *the map $t \longrightarrow W_t(\omega)$ is continuous as a function of t a.s.*

Theorem 1.1. [***The 1-dimensional Itô formula***]
Let X be a stochastic process, suppose that X has a stochastic differential

$$dX_t = Fdt + GdW_t$$

or an integral form

$$X_t = X_0 + \int_0^t Fds + \int_0^t GdW_t$$

for $F \in \mathbb{L}^1(0,T)$[4] $and\ G\ \in \mathbb{L}^2(0,T)$[5], $T > 0$. Also assume that $u : \mathbb{R} \times [0,T] \longrightarrow \mathbb{R}$ is continuous and

$$\frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}\ and\ \frac{\partial^2 u}{\partial x^2}$$

exist and are continuous. Set $Y_t := u(X_t, t)$, then Y has the stochastic differential representation

$$\begin{aligned} dY_t &= \frac{\partial u}{\partial t}(X_t)dt + \frac{\partial u}{\partial x}(X_t)dX_t + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(X_t)dX_t^2 \qquad (1.6)\\ &= \left(\frac{\partial u}{\partial t}(X_t) + \frac{\partial u}{\partial x}(X_t)F + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(X_t)G^2\right)dt + \frac{\partial u}{\partial x}(X_t)GdW_t, \qquad (1.7)\end{aligned}$$

where W_t is a Brownian motion and dX_t^2 is calculated by using
$dW_t \cdot dW_t = dt,\ dW_t \cdot dt = 0,\ dt \cdot dt = 0$. The formula (1.6) is called Itô formula.

For a proof see [6] or [7].

Remark 1.7. [***Change of variables formula***]
Suppose W_t is a Brownian motion and $u : \mathbb{R} \longrightarrow \mathbb{R}$ is a C^2-function [6], if we set $X_t = W_t$ and $Y_t = u(W_t)$, by applying Itô formula we have:

$$dY_t = du(W_t) = \frac{1}{2}\frac{d^2u}{dx^2}(W_t)dt + \frac{du}{dx}(W_t)dW_t. \qquad (1.8)$$

By integrating over the interval $[0,t]$ we have:

$$u(W_t) - u(W_0) = \int_0^t u'(W_s)dW_s + \frac{1}{2}\int_0^t u''(W_s)ds. \qquad (1.9)$$

The formula (1.9) is called change of variables formula.

Remark 1.8. *The formula (1.9) is similar to the fundamental theorem of calculus*

$$f(t) - f(0) = \int_0^t f'(s)ds.$$

[4] This is the set of class of functions f defined on $[0,T]$ such that $\int_0^T \mid f(x) \mid dx < \infty$.
[5] This is the set of class of functions f defined on $[0,T]$ such that $\int_0^T f^2(x)dx < \infty$.
[6] means u and its first two derivatives are continuous

1.2 Asset Pricing

In this section our goal is to collect some definitions related to asset pricing. More details can be found in [8].

Definition 1.12. [***Derivative security***]
A derivative security is a financial contract whose value at its expiry date T is completely determined by the prices of an underlying asset in a fixed range of times within the interval $[0, T]$.

The most common derivatives are options. There are two main types of options, call options and put options.
A call option is a financial contract between two parties, the buyer and the seller, where the buyer of the option has the right, but not the obligation to buy an agreed quantity of a particular commodity or financial instrument from the seller of the option at a certain time (the expiration date) for a certain price (the strike price).
A European call option allows the holder to exercise the option (i.e to buy) only at the option expiration date. An American call option allows the holder to exercise the option at any time during the life of the option.
A put option (sometimes simply called a "put") is a financial contract between two parties, the buyer and the writer (seller) of the option, which allows the owner the right but not the obligation to sell a commodity or financial instrument to the writer (seller) of the option at a certain time (the expiration date) for a certain price (the strike price). The seller has the obligation to purchase the underlying asset at that strike price, if the buyer exercises the option.
Similar to a call option, a European put option allows the holder to exercise the put option only at the option expiration date. On the other hand an American put option allows its owner to exercise the option at any time during the life of the option.
There exist many other types of options, often with place names such as Asian, Bermudan, Russian and Parisian options.
Let us denote by S the price of the underlying asset at the exercise time T and K the strike price. We define the payoff of the call option as

$$g(S, K) = [S(T) - K]^+ := \max(S(T) - K, 0), \tag{1.10}$$

and the payoff of the put option as

$$g(S, K) = [K - S(T)]^+ := \max(K - S(T), 0), \tag{1.11}$$

It is important to note that the payoff above belongs to the holder of the option and not to the writer of the option.

Definition 1.13. [***Arbitrage opportunity***]
An arbitrage opportunity is a market condition that allows an investor to make a riskless profit without initial investment.
Mathematically, an arbitrage opportunity is a portfolio (x, y) with the initial value $V_0^{(x,y)} = 0$ and the final value $V_T^{(x,y)} \geqslant 0$, together with $V_T^{(x,y)}(\omega) > 0$ for at least one $\omega \in \Omega$, where Ω is the set of possible scenarios.

2. The Theory of Monte Carlo

In applied sciences, we need to solve complex problems. Since most of these problems cannot be evaluated analytically, numerical solutions are required. But many numerical techniques fail in higher dimensions. For example, numerical integration schemes typically approximate the value of an integral by partitioning the integration region into a set of discrete volumes with associated weights (quadrature methods for example). A sum of weighted function values form an estimate of the integral, but the magnitude error increases with the dimension of integration domain.
The goal of this chapter is to present Monte Carlo method which is an attractive method to solve problems in high dimensions.
Most results in this chapter are due to [9], [10] and [11].

2.1 Principles of Monte Carlo

Monte Carlo methods are based on the analogy between probability and volume. The mathematics of measure formalizes the intuitive notion of probability, associating an event with a set of outcomes and defining the probability of the event to be its volume or measure relative to that of a universe of possible outcomes. Monte Carlo uses this identity in reverse, calculating the volume of a set by interpreting the volume as a probability. In the simplest case like Monte Carlo integration, this means sampling randomly from a universe of possible outcomes and taking the fraction of random draws that fall in a given set as an estimate of the set's volume. The law of large numbers (Theorem 2.1) ensures that this estimate converges to the correct value as the number of draws increases, and the central limit theorem (Theorem 2.2) provides information about the likely magnitude of the error in the estimate after a finite number of draws.

2.1.1 Description of the Monte Carlo Methods

To use the Monte Carlo methods, we must first put the quantity which we wish to calculate in the form of an expected value.

At the end of this stage, we have to evaluate a quantity $\mathbb{E}(X)$ where X is a random variable. In order to estmate $\mathbb{E}(X)$, we need to know how to simulate a random variable having the same distribution as X.

Mathematically, this means that we assume that we have a sequence of independent random variables $(X_i, 1 \leq i \leq N)$ all following the distribution of X. Computationally, we reduce the simulation of an arbitrary distribution to that of a sequence of independent random variables following a uniform distribution [1] on the interval $[0, 1]$. We shall see later that many distributions

[1] This type of random sequence is often provided in programming languages: random.random in python,rand in C

can be generated by using uniform distribution. The last step is to approximate $\mathbb{E}(X)$ by

$$\mathbb{E}(X) \approx \frac{X_1 + X_2 + ... + X_N}{N}, \tag{2.1}$$

which is very easy to compute.

2.1.2 Examples of Monte Carlo integration

Example 2.1. *Consider the integral*

$$I = \int_{[0,1]^d} f(u_1, u_2, ..., u_d) du_1 du_2 ... du_d. \tag{2.2}$$

We set

$$X = f(U_1, U_2, ..., U_d),$$

where $U_1, U_2, ..., U_d$ are independent random variables distributed unformly on the interval $[0, 1]$. We have:

$$\begin{aligned} E(X) &= E(f(U_1, U_2, ..., U_d)) \\ &= \int_{[0,1]^d} f(u_1, u_2, ..., u_d)\ du_1 du_2 .... du_d. \end{aligned}$$

Suppose we have a mechanism for drawing $(U_i, i \geq 1)$ a sequence of independent random variables uniformly distributed over $[0, 1]^d$ [2], and set

$$X_1 = f(U_1, U_2, ..., U_d), X_2 = f(U_{d+1}, U_{d+2}, ..., U_{2d}), X_3 = f(U_{2d+1}, U_{2d+2}, ..., U_{3d}), ...$$
$$..., X_N = f(U_{(N-1)d+1}, U_{(N-1)d+2}, ..., U_{Nd})$$

Then the sequence $(X_i, i \geq 1)$ is a sequence of random variables under the uniform distribution, and a good approximation of I is given by

$$\frac{X_1 + X_2 + ... + X_N}{N}.$$

Remark 2.1. *This method is very easy to program. It is also notable that this method does not depend on the regularity of f, which can be simply measurable.*

Example 2.2. *Often we want to evaluate an integral in $\mathbb{R}^d$ of the form*

$$J = \int_{\mathbb{R}^d} g(x) f(x) dx = \int_{\mathbb{R}^d} g(x_1, x_2, ..., x_d) f(x_1, x_2, ..., x_d) dx_1 dx_2 .... dx_d \tag{2.3}$$

where f is a positive function such that $\int_{\mathbb{R}^d} f(x)dx = 1$.

[2] obtained by successive evaluation of a random function

If X is a random variable with values in $\mathbb{R}^d$ and density function f, then $J = \mathbb{E}(g(X))$ and we can therefore approximate J by:

$$J \approx \frac{1}{n}\sum_{i=1}^{n} g(X_i), \tag{2.4}$$

where $(X_i, i \geq 1)$ is sampled from the density f.

If we assume that we know how to simulate distributions, it remains to answer two questions:

- How and when does Monte Carlo methods converge?
- What can we say about the precision of the approximation?

2.1.3 Limits and Convergence

The answers to the previous questions are given by two important theorems in stochastic calculus, the strong law of large numbers, which allows us to justify the convergence of the method and the central limit theorem, which gives the rate of convergence.

Theorem 2.1. [***Strong law of large numbers***]
Let $X_1, X_2, ..., X_n, ...$ be a sequence of independent, identically distributed, integrable random variables, where all of them have the same expected value μ and same variance σ^2. Then

$$P\left(\lim_{n\to\infty} \frac{X_1 + X_2 + ... + X_n}{n} = \mu\right) = 1 \tag{2.5}$$

For the proof see [5]

For this method to be useful, there must be a way of evaluating the error:

$$\varepsilon_n = \mathbb{E}(X) - \frac{1}{n}(X_1 + X_2 + ... + X_n). \tag{2.6}$$

Theorem 2.2. [***Central Limit Theorem***]
Let $X_1, X_2, X_3, ..., X_i, ...$ be a sequence of independent, identically distributed random variables such that $\mathbb{E}(X_i) = \mu < \infty$ and $V(X_i) = \sigma^2 < \infty$ for $i = 1, 2, ...$
For all $-\infty < a < b < +\infty$, we have:

$$\lim_{n\to\infty} P(\frac{\sigma}{\sqrt{n}}a \leq \varepsilon_n \leq \frac{\sigma}{\sqrt{n}}b) = \frac{1}{\sqrt{2\pi}}\int_a^b e^{-\frac{x^2}{2}}dx \tag{2.7}$$

This means that

$$\frac{\sqrt{n}}{\sigma}\varepsilon_n \qquad converges\, in\, probability\, to\, a\, reduced\, centred\, Gaussian\ \ distribution\, G. \tag{2.8}$$

For the proof see [5]

Remark 2.2. *In practical applications, we forget the passage to the limit and we replace ε_n by a centred Gaussian distribution with variance $\frac{\sigma^2}{n}$.*

We can also note that the central limit theorem never allows us to bound the error, since the support of the Gaussian is equal to whole $\mathbb{R}$. We often describe the error of the Monte Carlo method by giving the standard deviation of ε_n, that is $\frac{\sigma}{\sqrt{n}}$, or by giving a 95% confidence interval [3]*. As*

$$P(\mid G \mid \leq 1.96) \approx 0.95$$

we are led to a confidence interval of the form:

$$\left[\mu - 1.96\frac{\sigma}{\sqrt{n}}, \mu + 1.96\frac{\sigma}{\sqrt{n}}\right].$$

Remark 2.3. *We must also note that the rate of convergence in Monte Carlo methods is $\frac{1}{\sqrt{n}}$ which is not very good is one dimension since trapezoidal rule in numerical integration for example has at least $\frac{1}{n^2}$, like rate of convergence for twice continuously differentiable functions. The advantage in Monte Carlo methods is that the convergence rate $\frac{1}{\sqrt{n}}$ holds for all dimensions.*

Remark 2.4. *Remark 2.2 shows that it is important to know the range of the variance σ^2 of the random variable since the error is proportional to σ. This variance would typically be unknown in a setting in which the expectation is unknown. It is often approximated by the empirical variance of the sample.*

Theorem 2.3. [***Empirical variance of the sample***]
Let X a random variable such that $\mathbb{E}(X) = \mu < \infty$. Let X_i be independent realizations under the distribution of X such that μ can be approximated by $\overline{m}_n$ given by

$$\overline{m}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Then for n large enough the variance σ^2 can be approximated by $\overline{\sigma}^2$ defined as :

$$\overline{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n} (X_i - \overline{m}_n)^2$$

This number is called empirical variance of the n samples.
Proof. Set $\triangle X_i = X_i - \overline{m}_n$ and let us calculate the random variable $\sum_{i=1}^{n} (\triangle X_i)^2$. We can

[3] i.e the result is found with 95% chance in the given interval and 5% chance outside .Obviously, this value may be replaced by another value close to 1

estimate it by its average value $\mathbb{E}\left[\sum_{i=1}^{n}(X_i-\overline{m}_n)^2\right]$ for n large.
By using the fact that (X_i) are independent we have:

$$\begin{aligned}
\sum_{i=1}^{n}(X_i-\overline{m}_n)^2 &\approx \mathbb{E}\left[\sum_{i=1}^{n}(\triangle X_i)^2\right] \\
&= \sum_{i=1}^{n}\mathbb{E}\left[(X_i-\overline{m}_n)^2\right] \\
&= \sum_{i=1}^{n}\mathbb{E}\left[X_i^2-\frac{2}{n}X_i\sum_{j=1}^{n}X_j+\left(\frac{1}{n}\right)^2\left(\sum_{j=1}^{n}X_i\right)^2\right] \\
&= n\,\mathbb{E}\left[X^2\right]-\frac{2}{n}\mathbb{E}\left[\left(\sum_{j=1}^{n}X_j\right)^2\right]+\frac{1}{n}\mathbb{E}\left[\left(\sum_{j=1}^{n}X_j\right)^2\right] \\
&= n\,\mathbb{E}\left[X^2\right]-\frac{1}{n}\mathbb{E}\left[\sum_{j=1}^{n}X_i^2+\sum_{i,j=1,\,i\neq j}^{n}X_iX_j\right] \\
&= n\,\mathbb{E}\left[X^2\right]-\frac{1}{n}\left(n\mathbb{E}\left[X^2\right]+n(n-1)\mathbb{E}\left[X\right]^2\right) \\
&= (n-1)\left(\mathbb{E}\left[X^2\right]-\mathbb{E}\left[X\right]^2\right) \\
&= (n-1)\sigma^2
\end{aligned}$$

Then

$$\sigma^2 \approx \frac{1}{n-1}\sum_{i=1}^{n}(X_i-\overline{m}_n)^2\,.$$

■

Remark 2.5. *We have seen that the rate of convergence of the Monte Carlo method is of the order* $\frac{\sigma}{\sqrt{n}}$*. Then if the variance is small then so is the error. There are numerous techniques* [4] *to improve the rate of convergence of this method, which try to reduce the value of the variance* σ^2*. The general idea is to give another representation, in the form of an expected value*

$$\mathbb{E}(X)=\mathbb{E}(Y)$$

where Y *is a random variable with variance smaller than the variance of* X*. Many techniques are proposed for this aim. See [9] or [10] for more informations.*

2.2 Simulation of Random Variables

In the previous section, we have seen that one of the main steps in Monte Carlo method is to simulate a sequence of independent random variables following the same distribution as a random

[4] called reduction of variance techniques

variable X. Most of the languages available in modern computers have a random function, already programmed, whose output is a pseudorandom number[5] between 0 and 1, or a random number in a fixed interval uniformly distributed, this function is called rand() in C, random.random () in Python and random() in Turbo Pascal. In some cases these functions available in the software are not entirely satisfactory. We shall see how to construct random number generators in these cases. For more information see [9] or [10].

2.2.1 Simulation of a Uniform Distribution on [0,m-1] and [0,1]

The simplest and most commonly used method is the method of linear congruencies. We generate a sequence $(x_n)_{n\geq 0}$ of integers between 0 and $m-1$ in the following way:

$$\begin{cases} x_0 \in \{0, 1, ..., m-1\} \ (initial\ value) \\ x_{n+1} = (ax_n + b)\, modulo\ m \end{cases} \tag{2.9}$$

a, b, m being integers which must be chosen carefully if we want the characteristic statistics of the sequence to be satisfactory. The scientist Sedgewick in [12], recommends the following choice:

$$\begin{cases} a = 31415821 \\ b = 1 \\ m = 10^8. \end{cases} \tag{2.10}$$

This method allows us to simulate pseudorandom integers between 0 and $m-1$. To obtain a random real number between 0 and 1 we divide the random integer by m.

Remark 2.6. *The Sedgewick period ($m = 10^8$) can sometimes appear insufficient. We can obtain random number generators with arbitrarily long period by increasing m.*

Remark 2.7. *The other simulations of random variables can be simulated starting from a sequence of uniform random variables over [0,1]. We shall show in the next subsection, the case of Gaussian and exponential distribution. For more information about the other cases see [9] or [10].*

2.2.2 Simulation of Gaussian Variables

A classical method to simulate Gaussian random variables uses the observation that if (U_1, U_2) are two independent uniform random variables over [0,1], then

$$\left(\sqrt{-2\log(U_1)}\,\cos(2\pi U_2), \sqrt{-2\,\log(U_1)}\,\sin(2\pi U_2)\right)$$

is a pair of independent random variables following reduced centred Gaussian distributions [6].
To simulate Gaussians of average μ and variance σ^2, we can just set $X = \mu + \sigma g$, where g is a reduced, centred Gaussian random variable.

[5] As said Donald E. Knuth, Every random number generator will fail in at least one application, it is for this reason that computer-generated random numbers are referred to as pseudo-random numbers. They simply cannot be random.

[6] i.e. with average zero and variance 1

2.2.3 Simulation of an Exponential Distribution

We know that a random variable X which follows an exponential distribution with parameter μ has density function $\mathbb{I}_{\{x\geq 0\}}\mu e^{-\mu x}$.

We can simulate X by noting that, if U follows a uniform distribution on [0,1], $\frac{log(U)}{\mu}$ follows an exponential distribution with parameter μ. See [9] or [10].

2.3 First application: European Options

In this section we are going to use Monte Carlo methods to calculate the expected present value of the payoff [7] of a call option on a stock.

Let $S(t)$ denote the price of the stock at time t. Consider a call option giving the holder the right to buy the stock at a fixed price K at a fixed time T in the future; the current time is $t = 0$. As we have mentioned in the previous chapter, the payoff to the option holder at time T is

$$g(S, K) = [S(T) - K]^+ = \max(S(T) - K, 0)$$

The fair price of a European contingent claim is equal to the discounted expected value of its payoff (under the risk-neutral measure Q) at expiry, see [13]. Thus to get the present value of this payoff we multiply the payoff by a discount factor e^{-rT}, with r a constant interest rate and take the expected value (i.e. the present value of the option is $\mathbb{E}_Q\left[e^{-rT}\left[S(T) - K\right]^+\right]$).
The evolution of the stock price is given by the following Black-Scholes model

$$\frac{dS(t)}{S(t)} = rdt + \sigma dW(t) \tag{2.11}$$

where W is a standard Brownian motion and σ the volatility of the stock price, the rate of return has being taken to be the same as the interest rate.

Theorem 2.4. *The solution of the stochastic differential equation (2.11) at expiry date T is given by*

$$S(T) = S(0)\exp\left(\left[r - \frac{1}{2}\sigma^2\right]T + \sigma W(T)\right) \tag{2.12}$$

$$\sim S(0)\exp\left(\left[r - \frac{1}{2}\sigma^2\right]T + \sigma\sqrt{T}N\right) \tag{2.13}$$

where S(0) is the current price of the stock and $N := N(0, 1)$ is a standard normal random variable.

Proof. By integrating both sides of equation (2.11) we have:

$$\int_0^t \left[\frac{dS(s)}{S(s)}\right] ds = rt + \sigma W(t) \quad (W_0 = 0) \tag{2.14}$$

[7] which is the option price

To evaluate the integral on the left hand side we use the Itô formula for the function

$$u(t,x) = \log x \quad , x > 0$$

and obtain

$$\begin{aligned} d(\log(S(t)) &= \frac{1}{S(t)} dS(t) + \frac{1}{2}\left(\frac{-1}{S(t)^2}\right)\left(\sigma^2 S(t)^2 dt\right) \\ &= \frac{dS(t)}{S(t)} - \frac{1}{2}\sigma^2 dt. \end{aligned}$$

Hence

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

and from (2.14) we have

$$\log\left(\frac{S(t)}{S(0)}\right) = (r - \frac{1}{2}\sigma^2)t + \sigma W(t)$$

or

$$S(t) = S(0)\exp\left(\left[r - \frac{1}{2}\sigma^2\right]t + \sigma W(t)\right) \tag{2.15}$$

We know that

$$W(t) = W(t) - W(0) \sim N(0,t) = \sqrt{t}N$$

thus

$$S(t) \sim S(0)\ \exp\left(\left[r - \frac{1}{2}\sigma^2\right]t + \sigma\sqrt{t}N\right).$$

We just take $t = T$ to conclude our proof. ■

According to (2.13) the logarithm of the stock price is normally distributed, and the stock price itself has a lognormal distribution.
The expectation $\mathbb{E}_Q\left[e^{-rT}\left[S(T) - K\right]^+\right]$, is an integral with respect to the lognormal density of $S(T)$. By using the normal distribution we have:

$$\mathbb{E}_Q\left[e^{-rT}\left[S(T) - K\right]^+\right] = \frac{1}{\sqrt{2\pi T}}\int_{\mathbb{R}}\left[S(0)\ e^{(\sigma y - \sigma^2\frac{T}{2})} - e^{-rT}K\right]^+ e^{-\frac{y^2}{2T}}dy.$$

After some calculus (see [1] or [14] for details), this reduces to

$$S(0)\Phi(g(S(0),T)) - Ke^{-rT}\Phi(h(S(0),T),$$

where

$$\Phi(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} e^{-\frac{y^2}{2}}dy;$$

$$g(x,T) = \frac{\log(\frac{x}{K}) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}} \quad h(x,T) = g(x,T) - \sigma\sqrt{T}.$$

In light of the availability of this formula, there is no need to use Monte Carlo to compute $\mathbb{E}_Q\left[e^{-rT}\left[S(T)-K\right]^+\right]$.
From (2.13), we see that to draw samples of the terminal stock price $S(T)$ it suffices to have a mechanism for drawing samples from the standard normal distribution. Given a mechanism for drawing $(N_i)_{i=1}^n$, a sequence of independent standard normal random variables we can easily estimate $\mathbb{E}_Q\left[e^{-rT}\left[S(T)-K\right]^+\right]$ by

$$\frac{1}{n}\sum_{i=1}^{n}\exp\left(-rT\right)\left[S_T^i-K\right]^+ \tag{2.16}$$

where $S_T^i = S(0)\exp\left(\left[r-\frac{1}{2}\sigma^2\right]T+\sigma\sqrt{T}N_i\right), i=1,...,n$ (see Python code in appendix)

Remark 2.8. *It was shown in [15] that the price of an American call option written on a stock that pays no dividends is equivalent to the price of the corresponding European call option. Then the previous example can also be used to evaluate the price of an American call options written on a stock that pays no dividends.*

Remark 2.9. *Figures 2.1 and 2.2 show the convergence of Monte Carlo methods for evaluating a European Call option (American Call option on a stock that pays no dividends).*

Figure 2.3 shows that when the number of steps is large the true error in Monte Carlo method approaches $\frac{\overline{\sigma}}{\sqrt{n}}$ *where* $\overline{\sigma}^2$ *is the empirical variance of the sample given by* $\overline{\sigma}^2=\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i-\overline{m}_n\right)^2$.

Figure 2.1: Convergence of the Monte Carlo methods for evaluating a European Call Option (American Call option on a stock that pays no dividends), the true price here is 27.66.

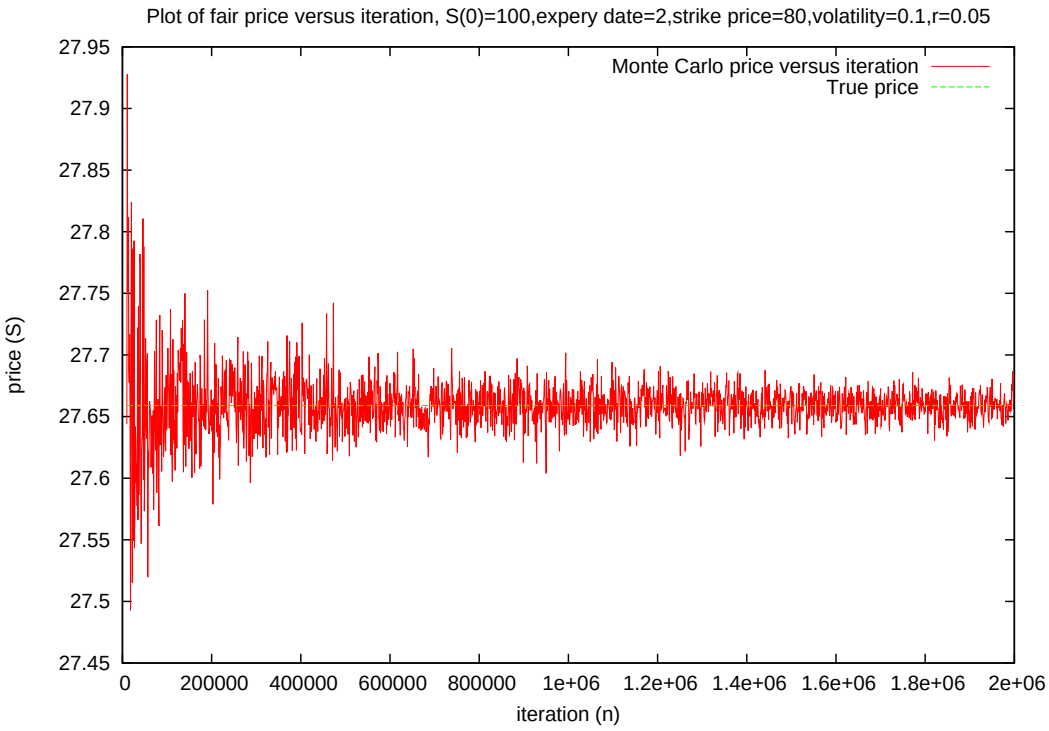

Figure 2.2: Convergence of the Monte Carlo method for evaluating a European Call Option(American Call option on a stock that pays no dividends) for a large number of step. The true price here is 27.66.

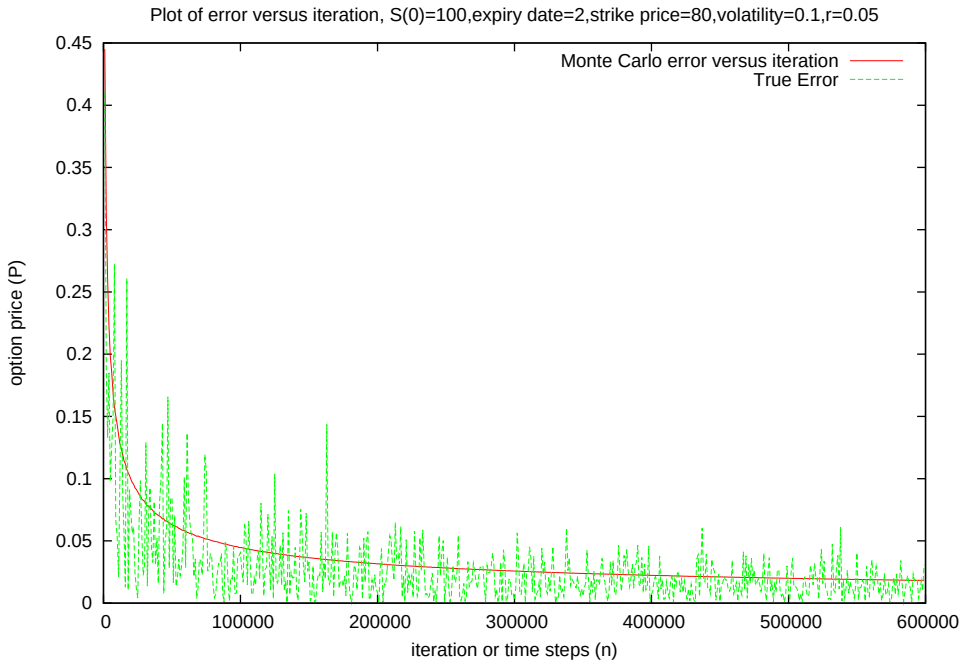

Figure 2.3: Comparison true error and Monte Carlo error.

3. American Put Pricing and Various Monte Carlo Techniques

American options contain the simple feature that they can be exercised at any time before the option's expiry date. However, this feature causes great difficulty in the pricing methodology. Until 1993, it was generally believed that the Monte Carlo methods, the most attractive method to solve problems in high dimensions, were unsuitable for pricing American put options. The goal of this chapter is to present some Monte Carlo techniques used to solve this problem. Most results in this chapter are due to [10],[11]and[2].

3.1 American put pricing problem within the Monte Carlo framework

In this section, our aim is to formulate the American put pricing within the Monte Carlo framework for utilisation in the next section.

3.1.1 Problem Formulation

The proper valuation of American puts is one of the important unsolved problems. Recall that a European put payoff is $(K - S(T))^+$ at time T, while an American put allows to exercise early. If we exercise an American put at time $t < T$ [1], we receive $(K - S(t))^+$. Then during the period $[t, T]$ we receive interest, and the amount we have at time is $T\ (K - S(t))^+ e^{r(T-t)}$. At initial date this amount is equivalent to $(K - S(t))^+ e^{-rt}$.
We want to find a rule, known as the exercise policy, to decide when to exercise the put, and the corresponding value for that policy. Since we cannot look into the future, we are in fact looking for a stopping time τ that maximizes

$$\mathbb{E}\left(e^{-r\tau}\left(K - S(\tau)\right)^+\right). \tag{3.1}$$

There is no good theoretical solution to finding the stopping time τ.

Definition 3.1. [***Mathematical definition of stopping time***]
Let (Ω,$\mathbb{F}$, $(F_t)_{t\geq 0}$,P) a filtered probability space. A random variable $\tau : \Omega \longrightarrow [0, T]$, $T > 0$ *is called a stopping time if*

$$\{\omega, \tau(\omega) \leq t\} \in F_t \ for\ all\ t \geq 0.$$

Remark 3.1. *It is important to note that if $\tau(\omega) = t_0$ (constant) for all ω, then τ is trivially a stopping time, because*

$$\{\omega, \tau(\omega) \leq t\} = \begin{cases} \Omega\ if\ t_0 < t \\ \emptyset\ if\ t_0 > t \end{cases}$$

[1] Early exercise

In general, American option pricing problems can be formulated by specifying a process of discounted payoff $U(t)$ from exercise at time t, and a class of admissible stopping times Γ with values in $[0,T]$. Under the assumption of no arbitrage opportunity, look for the option price is equivalent to find the optimal expected discounted payoff

$$\sup_{\tau\in\Gamma} \mathbb{E}\left[U(\tau)\right] \tag{3.2}$$

(see [1]).

Remark 3.2. *This previous result allows to evaluate the option price at time* 0. *To evaluate the option price at any time* t_0, *formula (3.2) is used where* Γ *is the class of stopping times with values in* $[t_0,T]$. *In our study without loss of generality, the option price will be calculated at time* 0.

The formulation (3.2) includes the classical American Put as a special case. Consider our initial American put option with constant bank account interest rate r which allows the holder the right to sell the stock of a single underlying asset S at a fixed price K and during the period $[0,T]$. The evolution of S is modeled by the Black-Scholes model (2.11). The option price at time 0 is given by

$$P(0,S) = \sup_{\tau\in\Gamma} \mathbb{E}\left[e^{-r\tau}\left(K - S(\tau)\right)^+\right] \tag{3.3}$$

where Γ is the set of stopping times (with respect to S) taking values in $[0,T]$,see [1].

The positive probability of early exercise implies that there exists a time dependent function, $b(t)$ called early exercise boundary, which can be used to determine if early exercise is optimal or not. If the stock price coincides with $b(t)$ at any time $t \in [0,T]$, it is optimal to exercise the option. The early boundary is defined by a continuous set of stock prices and partitions the time-stock price domain into a continuation region where the value of the option P is greater than its intrinsic value $(K-S)^+$ and satisfies the Black-Scholes PDE

$$\frac{\partial P}{\partial t} + rS\frac{\partial P}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} - rP = 0 \tag{3.4}$$

(see [14] or [2]) and a stopping region, where $P = (K-S)^+$ and exercise is optimal.
The supremum in (3.3) is achieved by an optimal stopping time τ^* such that

$$\tau^* = \inf\left\{t \geqslant 0 : S(t) \leqslant b(t)\right\} \tag{3.5}$$

Remark 3.3. *Equation(3.3) is already the formulation of American put pricing within the Monte Carlo framework since this price is written as the supremum of the expected discounted payoff over the set of stopping times, but unfortunately the early exercise boundary is unknown a priori and as a consequence the set of stopping time is unknown a priori. In the following lines we are going to see how to calculate this supremum.*

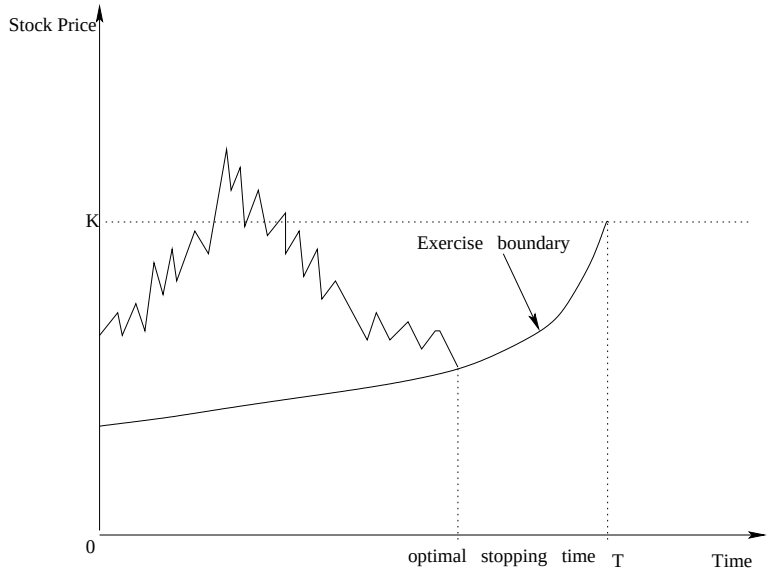

Figure 3.1: Exercise boundary for American Put with payoff $(K-S)^+$ .The option is exercised at τ, the first time the underlying asset reaches the boundary.

3.1.2 Dynamic Programming Formulation

In discussing simulation methods for pricing American options, we restrict our study to options that can be exercised only at a fixed set of exercise opportunities $0 = t_0 < t_1 ....., < t_n = T$, where the life of the option, $[0, T]$ is discretised into n equal intervals of size $\frac{T}{n}$, according to Remark 3.1 all t_i are stopping times since they are constant. Most of the time we will use the notation $S_i = S(t_i)$ and this notation would be generalised to any quantity at time t_i.
It is important to recall that the evolution of the stock price is given by Equation (2.15). To generate an independent sample, we use the usual way as for European options in the previous chapter. Given S_{i-1}, one can obtain S_i from S_{i-1} by

$$S_i = S_{i-1} \exp\left\{\left[r - \frac{1}{2}\sigma^2\right](t_i - t_{i-1}) + \sigma\sqrt{t_i - t_{i-1}}N_i\right\} \tag{3.6}$$

where $i = 1, ..., n$, $N_i \sim N(0, 1)$, this creates one path of the stock price.

A common approach to evaluating stopping times is to use Bellman's principle where the price of the American put option is approximated as a discrete dynamic programming problem which is solved using backward recursion. The difficulty here is that the Monte Carlo technique is forward recursive.
In dynamic programming, the value of the option at expiry date $t_n = T$ is the payoff function g such that $g(T, S) = (K - S(T))^+$ and at time t_i, $i = n-1, n-2, ..., 0$ we have two choices: exercise the option and then have immediately the intrinsic value $g(t_i, S_i) = (K - S_i)^+$ or hold the option and hope to have the value $P_{i+1}(S) = P(t_{i+1}, S_{i+1})$ whose value at time t_i is $H_{i+1}(S) = d_i \mathbb{E}[P_{i+1} \mid S = S_i]$ where $d_i = \exp\{-r(t_{i+1} - t_i)\}$ is the standardised discount factor between t_i to t_{i+1} (see [8] evaluation of option price, discrete case, Theorem 9.10). It is

also important to specify that the total discount factor D_i between $t_0 = 0$ and t_i is

$$D_i = \prod_{j=0}^{i} d_j = \exp\{-r(t_i - t_0)\} = \exp\{-rt_i\}. \tag{3.7}$$

The discrete dynamic programming problem for American put option is then reduced to:

$$P_n(S) = (K - S(T))^+ \tag{3.8}$$
$$P_{i-1}(S) = \max\{g(t_{i-1}, S_{i-1}), H_i(S)\} \ \ i = n, ..., 1 \tag{3.9}$$

where the continuation value

$$H_i(S) = d_{i-1}\mathbb{E}[P_i(S_i) \mid S = S_{i-1}] \ \ i = 1, ..., n$$

is the value of the option if it is unexercised at t_{i-1} and

$$g(t_i, S_i) = (K - S(t_i))^+ \ \ i = 0, 1, ..., n$$

is the optimal stopping value.

Estimating the continuation value is not straightforward. The Monte Carlo methods discussed below use different approaches to estimate this value and it can be used either to approximate the early exercise boundary, or to backward propagate the option value by approximating the optimal exercise time.

3.2 Monte Carlo techniques for solving American put pricing

In this section our aim is to present three of various techniques used since 1993 for solving American put pricing.

3.2.1 The Early Exercise Boundary and Optimal Stopping for Monte Carlo Methods

Most of Monte Carlo methods do not explicitly calculate the early boundary during the evaluation of option price. In these methods, exercise occurs when the continuation value drops below the intrinsic value. The optimal stopping time $\tau^*(m)$ is then chosen as the earliest time step this occurs,

$$\tau^*(m) = \min\left\{t_i \leq t_n \mid \widehat{H_i}(S_i^{(m)}) \leq g(t_i, S_i^{(m)})\right\} \tag{3.10}$$

where $\widehat{H_i}(S_i^{(m)})$ is the estimation of the continuation value $H_i(S_i^{(m)})$ using Monte Carlo. With values of all stopping times, the Monte Carlo estimation $\widehat{P}(0)$ of the initial option price is thus

$$\widehat{P}(0) = \frac{1}{N}\sum_{m=1}^{N} D_{\tau^*(m)}\left(K - S_{\tau^*(m)}^{(m)}\right)^+ \tag{3.11}$$

where $S^{(m)}_{\tau^*(m)} = S^{(m)}_i$ with $t_i = \tau^*(m)$.

3.2.2 Tilley's Bundling Algorithm

Tilley's bundling algorithm was the first Monte Carlo technique to estimate the American put option price in 1993. To estimate an American put price $P(0)$ on a single underlying asset, Tilley uses the dynamic programming algorithm (3.8) and (3.19) and describes a "bundling procedure ". He suggests to simulate N paths of asset prices denoted $S^{(m)}_i$, for $i = 1, ...n$ and $m = 1, ..., N$ in the usual way. At each time step $t_i,\ i = n-1, ..., 1$ Tilley orders the stock prices $S^{(m)}_i$, $m = 1, ..., N$. Once in descending order, the stock prices are divided into distinct bundles each containing r_1 stock prices. If there are a such distinct bundles, $A_j : j = 1, ..., a$ then $N = a \times r_1$. An average continuation value, $\widehat{H_i}(A_j)$, is associated with each distinct bundle, at each time step t_i. This average is calculated by taking the average of the discounted option values one time step ahead for each stock price in the bundle, i.e.

$$\widehat{H_i}(A_j) = \frac{1}{r_1} \sum_{i=1}^{N} d_i \, \widehat{P}_{i+1} \mathbb{I}_{\left\{S^{(m)}_i \in A_j\right\}}. \tag{3.12}$$

With these continuation values, the decisions for each individual sample path in each bundle are made. The associated average is compared with the intrinsic value at each stock price, and the option price is calculated by

$$\widehat{P}_i\left(S^{(m)}_i\right) = \begin{cases} g\left(t_i, S^{(m)}_i\right) \ \ if \ \ g\left(t_i, S^{(m)}_i\right) \geqslant \widehat{H_i}(A_j) \\ d_i \, \widehat{P}_{i+1} \ if \ \ g\left(t_i, S^{(m)}_i\right) < \widehat{H_i}(A_j) \end{cases} \tag{3.13}$$

for each $S^{(m)}_i \in A_j$, and where $m = 1, ..., N$.

Here, using the average continuation value for each bundle to make exercise decisions creates a transition zone instead of an exact exercise boundary, each transition zone being a subset of the ordered stock prices. There may be contradicting exercise decisions between stock paths in neighbouring bundles. To be more explicit, consider two stock prices $S^{(m_1)}_i < S^{(m_2)}_i$, each in a separate bundle. Due to the average nature of the exercise decision, exercise may be calculated to be optimal at stock price $S^{(m_2)}_i$ but not at $S^{(m_1)}_i$, even though the intrinsic value at $S^{(m_1)}_i$ is obviously greater than the intrinsic value $S^{(m_2)}_i$. This is because the intrinsic value is being compared with an average calculated continuation value. To ameliorate the transition zone, Tilley introduces in [16] a sharp boundary. The idea for sharp boundary is to pick the first stock price below which decisions to exercise dominate decisions not to exercise. The sharp boundaries can then be used to make exercise decisions and to calculate the option estimates, $\widehat{P}_i : i = n-1, ..., 1$. The final approximation of $\widehat{P}(0)$ is calculated by taking the discounted arithmetic mean of the approximated option values one time step in the future, i.e.

$$\widehat{P}(0) = \frac{1}{N} \sum_{m=1}^{N} d_0 \, \widehat{P_1}\left(S^{(m)}_1\right). \tag{3.14}$$

Steps of Tilley's Algorithm

To be more pratical, we can summarize Tilley's Algorithm in eight steps at each time step. The option values at expiry being equal to the intrinsic value, $\widehat{P}_n\left(S_n^{(m)}\right) = g\left(t_n, S_n^{(m)}\right)$, at each time step $t_i : i = n-1, ..., 1$ we need to:

(i) Reord the stock prices $S_i^{(m)} : m = 1, ...N$, from highest to lowest.

(ii) Compute the intrinsic value for each stock price, $g\left(t_i, S_i^{(m)}\right) : m = 1, ...N$.

(iii) Partition the N ordered stock prices into a distinct bundles, $A_j : j = 1, ..., a$ containing r_1 paths in each bundles, and define the bundling parameter, α such that $a = N^{\alpha}$, $r_1 = N^{1-\alpha}$ with obviously $N = a\, r_1$

(iv) Evaluate the continuation value for each bundle, $\widehat{H_i}(A_j) : \ j = 1, ..., a$ by using (3.12). The continuation value for each stock price path is then the continuation value of the bundle it belongs to.

(v) Evaluate a temporary indicator variable, $y_i()$, for each stock price, by using

$$y_i\left(S_i^{(m)}\right) = \begin{cases} 1 \ \ if \ \ g\left(t_i, S_i^{(m)}\right) \geqslant \widehat{H_i}\left(S_i^{(m)}\right) & Exercise \\ 0 \ \ if \ \ g\left(t_i, S_i^{(m)}\right) < \widehat{H_i}\left(S_i^{(m)}\right) & Continue \end{cases} \tag{3.15}$$

for $m = 1, ...N$. Exercise decisions are then given by sequences of zeros and ones generated by this indicator variable. The transition zone begins with the first 'Exercise' indication and ends with the last 'Continue' indication,e.g.

$$\begin{array}{lc} & \downarrow Transition\ zone \downarrow \\ \overline{y_i}\left(S_i^{(m)}\right) & 0.....\|0\,0\,0\,0\,0 \quad 1\|0\,0\,1\,1\,1\,1\|0\,0\,0\,1\,1\,1\|0 \quad 1\,1\,1\,1\,1\|1...1 \\ & \quad\ A_k \qquad\ A_{k+1} \qquad\ A_{k+2} \qquad\ A_{k+3} \end{array}$$

(vi) Look if the transition zone contains a number of false indicators. If so eliminate them to improve the convergence. In the previous example, the first '1' which marks the beginning of the transition zone is a false exercise indicator, the final '0' is also a false continue indicator; the first string of '1' which is longer than any string of '0' is effectively the true early exercise boundary indicator. To calculate $\widehat{b}_i$, Tilley chooses the unique value of m, denoted k^* which corresponds to the position of the first temporary indicator value of '1' in this particular sequence e.g.

$$\begin{array}{lc} & Sharp\ Boundary \qquad k^* \\ & \qquad\qquad\qquad\quad \downarrow \\ \overline{y_i}\left(S_i^{(m)}\right) & 0\,0\,0\,0\,1\,0\,0 \quad 1 \quad 1\,1\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,1 \end{array}$$

Since this method is both time consuming and a little arbitrary, Tilley suggested in [16] another method which consists of moving all the zeros to the left and the ones to the right, k^* is then the number of zeros in the temporary indicator variable.

$$k^* = \sum_{m=1}^{N} \mathbb{I}_{\{y_i(S_i^m)=0\}} \tag{3.16}$$

The stock price which corresponds to $m = k^*$ $(i.e\ S_i^{k^*})$, is an approximation $\widehat{b}_i$ of the early exercise boundary b_i at time t_i.

(vii) Decide which sharp boundary method to use. Then, define a new exercise-or-continue indicator variable, $\overline{y_i}(.)$ by

$$\overline{y_i}\left(S_i^{(m)}\right) = \begin{cases} 1 \text{ for } m \geq k^* \\ 0 \text{ for } m < k^* \end{cases} \tag{3.17}$$

by using the previous step.

$$\begin{array}{lc} Sharp\ Boundary & k^* \\ & \downarrow \\ \overline{y_i}\left(S_i^{(m)}\right) & 0\,0\,0\,0\,0\,0 \quad 1 \quad 1\,1\,1\,1\,1\,1\,1\,1\,1 \end{array}$$

(viii) Take the approximate option value $\widehat{P}_i(.)$ at t_i by using the formula

$$\widehat{P}_i\left(S_i^{(m)}\right) = \begin{cases} g\left(t_i, S_i^{(m)}\right) \ if\ \overline{y_i}\left(S_i^{(m)}\right) = 1 \\ d_i\,\widehat{P}_{i+1}\ if\ \overline{y_i}\left(S_i^{(m)}\right) = 0 \end{cases}$$

and use formula (3.14) to calculate $\widehat{P}(0)$.

Naive Tilley's Method and Bias of Tilley's Method

In 1993 Tilley implemented a naive method for solving American put option pricing. This method replaces the optimal stopping time for each sample path $\tau^*(m), m = 1..., N$, with a time $\tau_*(m)$ such that this time maximises the discounted exercise value for each stock price path. This is no longer a stopping time, since it uses the information inherent in the entire sample path $\left\{S_i^{(m)} : i = 1, ..., n, m = 1, ..., N\right\}$ in order to determine its value. The naive estimate of the price at time 0 is then given by

$$\widehat{P}^N_{naive}(0) = \frac{1}{N}\sum_{m=1}^{N} e^{-r\cdot\tau_*(m)}\left(K - S^{(m)}_{\tau_*(m)}\right)^+.$$

This exercise strategy results is a systematic upward bias in the premium estimate since the exercise time is chosen to yield the maximum payoff per path and no other strategy can yield

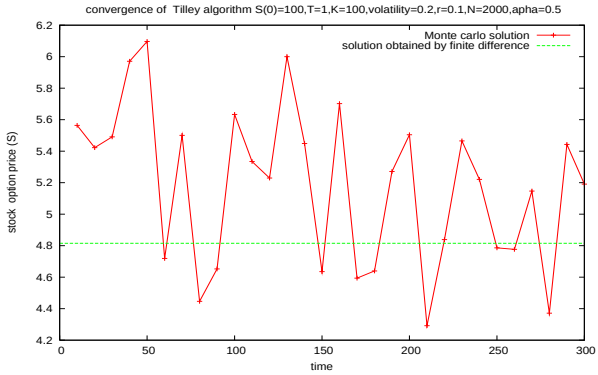

Figure 3.2: Convergence of Tilley's Algorithm, the finite difference solution 4.815 has being obtained in [2], for large N and n convergence will be better

a higher premium estimate. If we introduce a hypothetical estimate $\widehat{P}^N_{Exact}(0)$, for the price of the American Put option making the correct early exercise decisions then we have obviously the inequalities:

$$\widehat{P}^N_{Exact}(0) \leq \widehat{P}^N_{naive}(0), \qquad \widehat{P}^N_{Tilley}(0) \leq \widehat{P}^N_{naive}(0).$$

It is thus unclear how the bias of $\widehat{P}^N_{Tilley}(0)$ compares with $\widehat{P}^N_{Exact}(0)$. It is shown in [16] that Tilley's method contains two sources of bias: one upward due to the fact that it uses at each time step complete information, up to expiry for computing the continuation value, and one downward results from using a imprecise rule to select the sharp boundary.

Convergence of Tilley's Algorithm

- The estimation of American put pricing given by Tilley's algorithm converges as the number of time steps n tends towards infinity for a fixed bundling parameter α and a fixed large number of sample stock price paths N(see Figure 3.2). In general it is unclear whether this estimation has an upward or downward bias.
- Inversely, this estimation converges as the large number of sample stock price paths N tends towards infinity for a fixed large number of time steps n and fixed bundling parameter α (see[2]).
- The bundling parameter α has a large effect on the convergence, a small α (which implies a small number of bundles) results in an over-averaging of the continuation value and a large α reduces the accuracy of the mathematical expectation, because too few samples are used. It is shown in [2] that empirically, an $\alpha \approx 0.5$ yields the most stable results.

3.2.3 The Grant, Vora and Weeks Method (GVW)

This algorithm was created by Grant, Vora and Weeks in 1996. One advantage here is that it can be easily applied to higher dimensions. However, this method reduces to a search of the specific

state at each time step where early exercise is optimal.

The Early Exercise Boundary at Each Time Step

This method uses Monte Carlo simulation to approximate explicitly the early exercise boundary. The discrete approximation of the early exercise boundary $\left\{\widehat{b_i}\right\}_{i=0,\ldots,n-1}$ is calculated by comparing intrinsic values with approximate continuation values at each time step, the exercise boundary at expiry date T is equal to the strike price $b_n = K$. At each time step an approximation of the early exercise boundary value is found by searching the stock price which makes the intrinsic value equal to the continuation value i.e.

$$\left\{\widehat{b_i} = S | \, (K - S)^+ = \widehat{H_i}(S)\right\} \tag{3.18}$$

which is equivalent to solve the equation $f(S) = 0$ where

$$f(S) = g(t_i, S) - \widehat{H_i}(S).$$

It is important to note that each continuation value $\widehat{H_i}(S)$ is calculated using Monte Carlo methods and the already calculated portion of the early exercise boundary $\left\{\widehat{b_i}\right\}_{i=j+1,\ldots,n}$.
To estimate $H_i(S)$ by Monte Carlo, we need to generate N' stock price sample paths $\left\{S_j^{(m)}\right\}_{j=i,\ldots,n}$ for each $m = 1, \ldots..N'$ conditional on $S_i^{(m)} = S$ for all m,

$$S_j^{(m)} = S \exp\left\{\left(r - \frac{\sigma^2}{2}\right)(t_j - t_i) + \sum_{k=i+1}^{j} \sqrt{t_k - t_{k-1}} N_k^{(m)}\right\}$$

where $j = i+1, \ldots., n$, $N_k^{(m)} \sim N(0,1)$ and the same random numbers are used for each successive value of $S_j^{(m)}$.
For each value of m a corresponding optimal exercise time is

$$\tau(m) = \min\left\{t_{i+1} \leq t_j \leqslant t_n \mid S_i^{(m)} \leqslant \widehat{b_j}\right\}$$

where $\left\{\widehat{b_j}\right\}_{j=i+1,\ldots,n}$ are the early exercise boundary estimates already calculated in the previous steps. Thus,

$$f(S) = g(t_i, S) - \frac{1}{N'} \sum_{m=0}^{N'} \exp\left\{-r(\tau(m) - t_i)\right\} g\left(\tau(m), S_{\tau(m)}^{(m)}\right) \tag{3.19}$$

g being the payoff function.
Equation (3.19) does realy simplify the process. The previous Monte Carlo estimation of $\widehat{H_i}(S)$ affects the monotonicity of the early exercise boundary (see Figure 3.3). The technique suggested by the authors is the following: The globally admissible early exercise domain being $[0, K]$, at each time step t_i they suggest to:

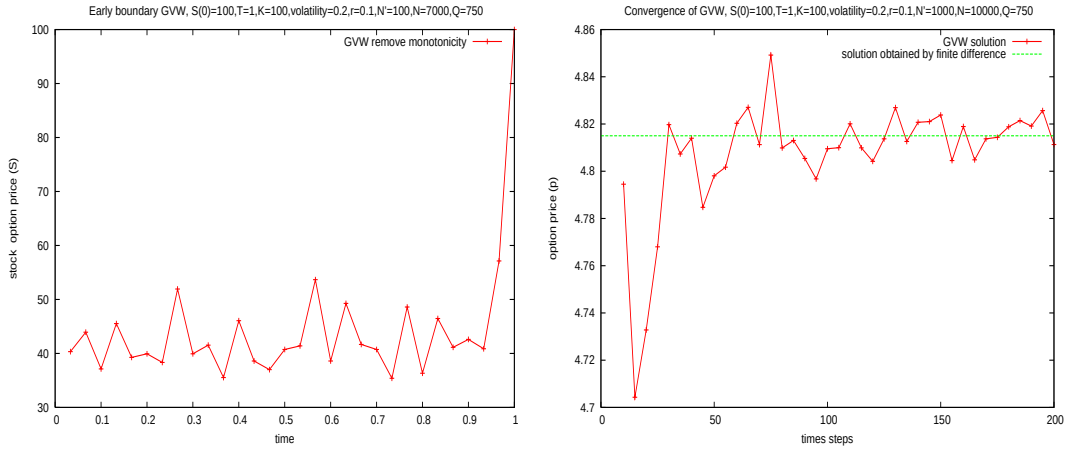

Figure 3.3: On the left we have the early exercise boundary obtained with GVW Algorithm and on the right the convergence of GVW Algorithm, the finite difference solution 4.815 has being obtained in [2]

- Divide $[0, K]$ into Q possible values, $\beta_q : q = 1, ...Q$.
- Evaluate $f(\beta_q)$ using formula (3.19) for each β_q.
- At some value q^* where $f(\beta_{q^*})$ changes sign, the root of f is in the interval $(\beta_{q^*-1}, \beta_{q^*}]$. Thus take the linearly interpolated from $(\beta_{q^*-1}, \beta_{q^*}]$ for approximation of the early boundary value $\widehat{b_i}$.

Early exercise boundary being known, N independent stock price paths can be generated by standard Monte Carlo. The optimal stopping times for these new paths are evaluated with the early exercise boundary $\left\{\widehat{b_i}\right\}_{i=1,...,n}$. These optimal stopping times, $\tau^*(m)$ for all $m = 1, 2, ..., N$ are defined by

$$\tau^*(m) = \min\left\{t_i \leqslant t_n \mid S_i^{(m)} \leqslant \widehat{b_i}\right\}$$

for $i = 1, ..., n$.
The Monte Carlo approximation for the initial price $\widehat{P}(0)$ is then calculated by using the formula (3.11).

Convergence of the GVW Algorithm

- The estimation of American put pricing by GVW algorithm converges as the number of time steps n tends towards infinity for a fixed large number of sample stock price paths N(see Figure 3.3)

- Inversely, this estimation converges as large number of sample stock price paths N tends towards infinity for a fixed large number of time steps n. The number of stock prices N'

used to calculate the early exercise boundary does not necessarily have a large effect (see [2]).

3.2.4 State Space Partitioning Algorithm (SSAP)

This Algorithm was created in 1995 by Barraquand and Martineau and uses a similar dymanic programming technique like Tilley's method, but partitions the stock price domain (state space) into a tractable number of states instead of organising the stock prices into bundles at each time step. Monte Carlo is used to estimate the continuation value associated with each state by calculating the conditional probabilities of moving between states, and dymanic programming is used to estimate the initial price $\widehat{P}(0)$.

Partitioning the State Space

At time step $t_i, i = 0, ..., n$, the state price is partitioned into a_i disjoint states,$A_i^j : j = 1, ..., a_i$. The authors use a real valued function to partition a region of interest bounded by an upper and lower value, $S_{\min}$ and $S_{\max}$. At $t_0 = 0$, this region of interest is the initial stock price S_0. Thus $A_0^1 = \{S_0\}$ and $a_0 = 0$. For all future times $t_i, i = 1, ..., 0$, the partitioning is chosen using

$$A_i^j = (\alpha_i \exp \beta_i(j-2), \alpha_i \exp \beta_i(j-1)] \tag{3.20}$$

where $j = 2, ..., a_i - 1$ and $\alpha_i,\ \beta_i$ are determined by restricting the region of interest to the set of values where simulated stock price paths are most likely to occur. This region is $[S_{\min}, S_{\max}]$ with

$$P(S_{t_i} < S_{\min}) \approx P(S_{t_i} > S_{\max}) \approx 0.01 \tag{3.21}$$

The probability that a sample stock price lies outside the region of interest is non-zero, consequently the first and last states are defined to make allowance for these events: $A_i^1 = [0, S_{\min}]$ and $A_i^{a_i} = [S_{\max}, \infty)$. By using formula (3.20) and the fact that $A_i^j,\ j = 2, ..., a_i - 1$ are included in $[S_{\min}, S_{\max}]$ at each time t_i and reach the extremities we have:

$$\begin{aligned} \alpha_i &= S_{\min} \quad (lower\ bound\ of\ A_i^2 \\ S_{\min} \exp\{\beta_i(a_i - 2)\} &= S_{\max} \quad (upper\ bound\ of\ A_i^{a_i - 1} \end{aligned}$$

Values for $S_{\min}$ and $S_{\max}$ are chosen using the inverse cumulative normal function Φ^{-1} on the log price $S_{t_i} \sim S(0) \exp\left(\left[r - \frac{1}{2}\sigma^2\right] t_i + \sigma\sqrt{t_i}N\right)$ which is distributed normally with a mean $\mu_i = \log(S_0) + (r - \frac{\sigma^2}{2})t_i$ and a standard derivation $\sigma_i = \sigma\sqrt{t_i}$. Then we have

$$P(S_{t_i} < S_{\min}) \approx 0.01 \iff P\left(\log\left(S_{t_i}\right) < \log\left(S_{\min}\right)\right) \approx 0.01 \tag{3.22}$$

$$\iff \Phi\left(\frac{\log\left(S_{\min}\right) - \mu_i}{\sigma_i}\right) \approx 0.01 \tag{3.23}$$

$$\iff \frac{\log\left(S_{\min}\right) - \mu_i}{\sigma_i} = \Phi^{-1}(0.01) \tag{3.24}$$

$$\iff \alpha_i = S_{\min} = \exp\left\{\mu_i + \Phi^{-1}(0.01)\sigma_i\right\} \tag{3.25}$$

where $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{y^2}{2}} dy$.
On a similar manner we find the following expression of β_i

$$\beta_i = \frac{\mu_i + \Phi^{-1}(0.99)\sigma_i - \log(\alpha_i)}{a_i - 2} \tag{3.26}$$

The remaining partitions are evaluated using Equation (3.20). At each time step, a fixed number ($a_i = a$, for all $i > 0$) of partitions is decided a priori. This number can be varied to optimise the convergence rate of the estimation.

Calculation of the Average Intrinsic Value for each State

We need to evaluate an average intrinsic value $g\left(A_i^j\right) : i = 1, ..., n,\ j = 1, ..., a_i$. We generate N sample stock price paths, $\left\{S_i^{(m)}\right\}\ i = 1, ..., n,\ \ m = 1, ..., N$. The average intrinsic values are calculated as the arithmetic mean of the intrinsic values of sample stock price paths which lie in the particular state, thus

$$g\left(A_i^j\right) = \frac{1}{N_i^j} \sum_{m=1}^{N} \left(K - S_i^{(m)}\right)^+ \mathbb{I}_{\left\{S_i^{(m)} | S_i^{(m)} \in A_i^j\right\}} \tag{3.27}$$

where $i = 1, ..., n,\ j = 1, ..., a_i$ and $N_i^j = \sum_{m=1}^{N} \mathbb{I}_{\left\{S_i^{(m)} | S_i^{(m)} \in A_i^j\right\}}$ is the number of sample stock price paths that belong to each state A_i^j.

Calculation of the Transition Probabilities

To calculate transition probabilities, SSAP uses the dynamic programming algorithm at each of the states A_i^j. The continuation value associated with each state is evaluated as the discounted expected value of the option one time step in the future, conditional on the originating stock price being in the state. To calculate this value, we need the transition probabilities $q_i^{j,k}$. These represent the probability of moving from state A_i^j at time t_i, to state A_{i+1}^k, at time t_{i+1}, we require then the solution of the equation

$$q_i^{j,k} = P\left(S_{i+1} \in A_{i+1}^k \mid S_i \in A_i^j\right) \quad i = 0, ..., n-1,\ \ j = 1, ..., a_i,\ \ k = 1, ..., a_{i+1}$$

These probabilities can be estimated by using the same sample stock price paths used to estimate the intrinsic values of each state. If $\widehat{q}_i^{j,k}$ are these estimations, we thus have

$$\widehat{q}_i^{j,k} = \frac{N_i^{j,k}}{N_i^j}$$

where $N_i^{j,k}$ is the number of stock price paths that originate in state A_i^j and subsequently pass through state A_{i+1}^k.

Estimation of the Option Price

After evaluation of average intrinsic values and transition probabilities, the continuation values, $\widehat{H}_i(A_i^j)$ are calculated for each state at each time step by

$$\widehat{H}_i(A_i^j) = \exp\{-r(t_{i+1} - t_i)\} \sum_{k=1}^{a_i+1} \widehat{q}_i^{j,k} \widehat{P}_{i+1}\left(A_{i+1}^k\right) \quad i = 0, ..., n-1, \;\; j = 1, ..., a_i,$$

The dynamic programming problem becomes thus

$$\begin{aligned} \widehat{P}_n\left(A_n^j\right) &= \widehat{g}_n\left(A_n^j\right) \quad j = 1, ..., a_n \\ \widehat{P}_{i-1}\left(A_{i-1}^j\right) &= \max\left\{\widehat{g}_{i-1}\left(A_{i-1}^j\right), \widehat{H}_{i-1}(A_{i-1}^j)\right\} \quad i = n, ..., 1, \;\; j = 1, ..., a_i \end{aligned}$$

and the final estimation of the option price at time 0 is set by

$$\widehat{P}(0) = \widehat{P}\left(A_0^1\right). \tag{3.28}$$

Bias of the SSAP

Like Tilley's, SSAP display a combination of an upward and downward bias. The upward bias is the consequence of using at each time step future imformation. Early exercise decisions based on an estimated early exercise boundary cause the downward bias. Although neither bias can be estimated, the fact that they counteract ensures a more accurate estimation. We can minimize the upward bias by using one set of sample stock price paths to estimate the functional form of the continuation values, and another independent set of sample stock price paths to estimate the option price.

Convergence of the SSAP

- The estimation of American put pricing given by SSAP algorithm converges as the number of time steps n tends towards infinity for a fixed number of states a and a fixed large number of sample stock price paths N (see[2]).
- Inversely this estimation converges as the large number of sample stock price paths N tends towards infinity for a fixed large number of time steps n and a fixed number of states a (see[2]).
- The number of states a has a significant impact on both the quality of the estimated option value and the computational time of the algorithm. It is shown in [2] that empirically a value $a = 2n$, gives an optimal balance between precision and computational time.

Conclusion and perspectives

In this essay, we have presented Monte Carlo methods and its applications in finance, mainly for pricing American put options. To achieve this goal, we have given in the first part some mathematical tools in probability theory, stochastic processes and asset pricing which allow us in the two last parts to study Monte Carlo principles and its applications to pricing European call and American options.

We have considered only the case of one stock price with constant interest rate and volatility, for which we have shown in the second part the convergence of Monte Carlo methods for pricing European call options (American call option that pay no dividend), and presented in the last part three of various Monte Carlo techniques (Tilley's Algorithm, Grant, Vora and Weeks Method, State Space Partitioning Algorithm) used since 1993 for pricing American put options.

Our future perspectives is to apply previous Monte Carlo techniques in high dimensional pricing (e.g. cases including stochastic volatility and interest rate) where Monte Carlo methods are more attractive.

Appendix A. Code

The aim of this part is to present Python codes using for similation.

A.1 Code for computing the fair price for European call options (American Call options on a stock that pays no dividends)

```
from __future__ import division
from visual.graph import*
from math import *
from scipy import*
from scipy.stats import *
from scipy.integrate import*
from random import*
import Numeric
import Gnuplot
gp=Gnuplot.Gnuplot(persist=1)
gp('set data style line')
gp.xlabel('iteration (n)')
gp.ylabel('price (S)')
S0=100
T=2
sigma=0.1
K=80
r=0.05
a=(log(S0/K)+(r+0.5*sigma**2)*T)/(sigma*sqrt(T))
b=a-sigma*sqrt(T)
price=S0*(1-0.5*erfc(a/sqrt(2)))-exp(-r*T)*K*(1-0.5*erfc(b/sqrt(2)))
list2=[]                        # price =true price
list3=[]
for i in arange(10000,200000,5000):
    list1=[]
    for j in arange(1,i+1):
       t=gauss(0,1)

       c=exp(-r*T)*max(s-K,0)
       list1.append(c)
    som=0
    for k in list1:
        som=som+k
```

```
    list2.append([i,som/i])
    list3.append([i,price])
data1=Gnuplot.PlotItems.Data(list2,with="lines",title= " Monte Carlo
 price versus iteration")
data2=Gnuplot.PlotItems.Data(list3,with="lines",title= "True price" )
gp.title("Plot of fair price versus iteration, S(0)="+str(S0)+",
expery date="+str(T)+",strike price="+str(K)+",volatility="+str(sigma)+"
,r="+str(r)+"")
gp.plot(data1,data2)
gp.hardcopy(filename='essay1.eps',terminal='postscript',eps=True, color=True)
```

A.2 Code for computing the error in the Monte Carlo Method

```
from __future__ import division
from math import *
from scipy import*
from random import*
import Numeric
import Gnuplot
gp=Gnuplot.Gnuplot(persist=1)
gp('set data style line')
gp.xlabel('iteration (n)')
gp.ylabel('price (S)')
S0=100
T=2
sigma=0.1
K=80
r=0.05
a=(log(S0/K)+(r+0.5*sigma**2)*T)/(sigma*sqrt(T))
b=a-sigma*sqrt(T)
price=S0*(1-0.5*erfc(a/sqrt(2)))-exp(-r*T)*K*(1-0.5*erfc(b/sqrt(2)))
error2=[]
error3=[]
for i in arange(1000,600000,1000):
    list1=[]
    for j in arange(1,i+1):
       t=gauss(0,1)
       s=S0*exp((r-0.5*sigma**2)*T+sigma*sqrt(T)*t)
       c=exp(-r*T)*max(s-K,0)
       list1.append(c)
    som=0
```

```
    for k in list1:
        som=som+k
    er=0
    for x in list1:
        v=som/i
        er=er+(x-v)**2
    error2.append([i,sqrt(er/(i*(i-1)))])
    error3.append([i,abs(price-som/i)])
data1=Gnuplot.PlotItems.Data(error2,with="lines",title=
" Monte Carlo error versus iteration")
data2=Gnuplot.PlotItems.Data(error3,with="lines",title= "True Error" )
gp.title("Plot of error versus iteration, S(0)="+str(S0)+",expery
date="+str(T)+",strike price="+str(K)+",volatility="+str(sigma)+",r="+str(r)+"")
gp.plot(data1,data2)
gp.hardcopy(filename='essay2.eps',terminal='postscript',eps=True, color=True)
```

A.3 Code for computing Tilley's Algorithm

```
 from visual.graph import*
from math import *
from scipy import*
from random import*
import Numeric
import Gnuplot
gp=Gnuplot.Gnuplot(persist=1)
gp('set data style line')
gp.xlabel('time')
gp.ylabel('stock  option price (S)')
r=0.1
K=100
S0=100
exb=[K]
T=1
vola=0.2
apha=0.5
price=[]
sol1=[]
sol2=[[0,4.815]]
for N in arange(5000,5001,1):
    a= int(N**apha)
    r1=int(N**(1-apha))
    N0=a*r1
    for n in range(10,601,20):
```

```
step=T/float(n)
M=zeros([n,N0])
N1=zeros([n,N0])
#N1=zeros([n,N0])
for i in range(0,n):
    for j in range(0,N0):
        N1[i][j]=gauss(0,1)
for k in range(0,n):
    for y in range(0,N0):
        g1=0
        o=0
        while (o<=k):
            g1=sqrt(step)*N1[o][y]+g1
            o=o+1
        s1=S0*exp((r-0.5*vola**2)*(k+1)*step+vola*g1)
        M[k][y]=s1
price=[]
for s in M[n-1,]:
     price.append(max(K-s,0))
for i in range(n-2,-1,-1):
    intr=[]
    A=[]
    M[i,].sort()
    for k in range(0,N0):
        A.append(M[i,][N0-1-k])
    for u in A:
        intr.append(max(K-u,0))
    c=0
    list1=[]
    for t in range(1,N0+1):
        c=exp(-r*step)*price[t-1]+c
        if t%r1==0:
            list1.append(c/r1)
            c=0
    con=[]
    for e in list1:
        l=1
        while l<=r1:
            con.append(e)
            l=l+1
    tem=[]
    for k in range(0,N0):
        if intr[k]>=con[k]:
            tem.append(1)
        else:
```

```
                tem.append(0)
            tem.sort()
            k1=tem.count(0)
            nprice=[]
            for f in range(0,N0):
                if tem[f]==1:
                    nprice.append(intr[f])
                else:
                    w=exp(-r*step)*price[f]
                    nprice.append(w)
            price=nprice
        z=0
        for o in price:
            z=o+z
        a1=(z/N0)*exp(-r*step)
        a2=(z/N)*exp(-r*step)
        sol1.append([n,a2])
        sol2.append([n,4.815])

print a2
gp.title(" convergence of  Tilley algorithm S(0)="+str(S0)+",T="+str(T)+",K="+str(K)+
data1=Gnuplot.PlotItems.Data(sol1,with="linespoints",title= "Monte carlo solution")
data2=Gnuplot.PlotItems.Data(sol2,with="lines",title= "solution obtained by finite di
gp.plot(data1,data2)
gp.hardcopy(filename='Tilley.eps',terminal='postscript',eps=True, color=True)
```

A.4 Code for computing Grant,Vora and Weeks Methods

```
 from __future__ import division
from math import *
from scipy import*
import Gnuplot
from random import*
import Numeric
import Gnuplot
gp=Gnuplot.Gnuplot(persist=1)
gp('set data style line')
gp.xlabel('time')
gp.ylabel('stock  option price (S)')
Q=750
K=100
```

```
S0=100
exb=[K]
T=1
N0=10000
vola=0.2
c=[]
r=0.1
a1=0
b=0
sol1=[]
sol2=[[0,4.815]]
ex=[[T,K]]
for n in range(20,21,4):
    step=T/float(n)
    for i in arange(T-step,0,-step):   #fix time ti in inverse order
        #for m in arange(2,10,1): #fix a sample paths N'
        m=100
        c=zeros([len(exb)+1,m])
        s=K/Q
        for j in range(0,m):
            c[0][j]=s
        N=zeros([len(exb),m])    # array of random gauss numbers
        for u in range(0,len(exb)):
             for b in range(0,m):
                N[u][b]=gauss(0,1)
        for j in range(1,len(exb)+1): # fix line different for the first
            for y in range(0,m):
                g1=0
                for o in range(0,j):
                    g1=sqrt(step)*N[o][y]+g1
                s1=s*exp((r-0.5*vola**2)*j*step+vola*g1)
                c[j][y]=s1
        list1=[]
        for t in range(0,m):
            k=0
            exb1=exb
            exb1.reverse()
            while(c[k+1][t]>exb1[k])and(k<len(exb1)-1):
                k=k+1
            if k==0:
                if c[1][t]<=exb1[k]:
                    list1.append(1)
                else:
                    list1.append(len(exb1))
            else:
```

```
            list1.append(k)
        g=0
        for y in range(0,m):
            g=exp(-r*step*list1[y])*max(K-c[list1[y]][y],0)+g
        a1=max(K-c[0][0],0)-g/m
        a2=sign(a1)
        l=len(exb1)
        while (a2==sign(a1)):#and(s<exb1[l-1]):
            s= s+K/Q
            for j in range(0,m):
                c[0][j]=s
            N=zeros([len(exb1),m])    # array of random gauss numbers
            for u in range(0,len(exb1)):
                for b in range(0,m):
                    N[u][b]=gauss(0,1)
            for j in range(1,len(exb1)+1): # fix line different for the first
                for y in range(0,m):
                    g1=0
                    for o in range(0,j):
                        g1=sqrt(step)*N[o][y]+g1
                    s1=s*exp((r-0.5*vola**2)*j*step+vola*g1)
                    c[j][y]=s1
            list1=[]
            for t in range(0,m):
                k=0
                while(c[k+1][t]>exb1[k])and(k<len(exb1)-1):
                    k=k+1
                if k==0:
                    if c[1][t]<=exb1[k]:
                        list1.append(1)
                    else:
                        list1.append(len(exb1))
                else:
                    list1.append(k)
            g=0
            for y in range(0,m):
                g=exp(-r*step*list1[y])*max(K-c[list1[y-1]][y],0)+g
            a1=max(K-c[0][0],0)-g/m
        b=s-K/(2*Q)
        exb.append(b)
        ex.append([i,b])
    x=len(exb)
    exb.reverse()
    M=zeros([x,N0])
    list1=[]
```

```
    N=zeros([x,N0])
    for i in range(0,x):
        for j in range(0,N0):
            N[i][j]=gauss(0,1)

    for j in range(0,x):
        for y in range(0,N0):
            g1=0
            o=0
            if j==0:
                g1=sqrt(step)*N[0][y]
            else:
                while (o<=j):
                    g1=sqrt(step)*N[o][y]+g1
                    o=o+1
            s1=S0*exp((r-0.5*vola**2)*(j+1)*step+vola*g1)
            M[j][y]=s1
    for t in range(0,N0):
        k=0
        while(M[k][t]>exb[k])and (k<x-1):
            k=k+1
        if k==0:
            if M[0][t]<=exb[k]:
                list1.append(1)
            else:
                list1.append(len(exb))
        else:
            list1.append(k)
    g=0
    for y in range(0,N0):
        g=exp(-r*step*list1[y])*(max(K-M[list1[y]-1][y],0))+g
    a1=g/N0
    sol1.append([n,a1])
    sol2.append([n,4.815])
print a1
gp.title(" Early boundary GVW,S(0)="+str(S0)+",T="+str(T)+",K="+str(K)+",volatility="
N'="+str(m)+",N="+str(N0)+",Q="+str(Q)+"")
data1=Gnuplot.PlotItems.Data(sol1,with="linespoints",title= "GVW solution")
data2=Gnuplot.PlotItems.Data(sol2,with="lines",title= "solution obtained by finite di
gp.plot(data1,data2)
gp.hardcopy(filename='gv.eps',terminal='postscript',eps=True, color=True)
```

Acknowledgements

I would like to thank first God, who created the whole Universe and who makes all things possible.

I would like to express my deepest gratitude to my supervisors Prof David Taylor and Nadia Uys, without their guidance and support this work would not have been possible.

I am greatly indebted to the African Institute for Mathematical Sciences (AIMS) for giving me this opportunity to be here. I especially thank Prof Niel Turok for his foresight in founding this Institute, Prof Fritz Hahne our director and all the lecturers and tutors.

I would like also to say a big thank you to all my colleague at AIMS; their co-operation and sense of friendliness will ever remain fresh in my memory.

I would like also to say a big thank to Clementine Feugueu for her love and my son Tambue Kamga Serges Lions for his patience to live far from his father during ten months.

I wish to thank all my lecturers at University of Dschang and University of Yaounde I in Cameroon for their training and continuous support. Especially Dr Njifenjou Abdou, Dr Kengne Emmanuel, Prof Marcel Dossa, Prof Tayou Simo Jacques, Prof Norbert Noutchegueme, Prof Nguetsing Gabriel and Prof Wamon François.

Finally, I wish to thank my family for their support. Especially my Mother Ngamgo Madeleine and my sister Mrs Fonkwoua Pauline.

Bibliography

[1] Richard Bass. The basics of financial mathematics. Department of Mathematics University of Connecticut, Spring 2003.

[2] S. Randell. *Numerical Techniques for the American Put*. Msc dissertation, University of Witwatersrand, 2006.

[3] Marek Capinski and Ekkehard Kopp. Measure integral and probability. Springer verlag, October 2003.

[4] Richard Bass. Probability theory. Department of Mathematics University of Connecticut, 2001.

[5] Lawrence C. Evans. An introduction to stochastic differential equations. version 1.2.

[6] Bernt Oksendal. Stochastic differential equation: An introduction with application. Springer, sixth Edition.

[7] Richard Bass. Stochastic calculus,with applications to finance pde and potential theory. www.math.uconn.edu/ bass/lecture.html,Department of Mathematics University of Connecticut, Fall 2001.

[8] Alex Roux. Mathematical methods in modern finance. African Institute for Mathematical Sciences, 2007.

[9] B. Lapeyre, R. Pardoux, and R. Sentis. Introduction to monte carlo methods for transport and diffusion equation. Oxford University of Press.

[10] P. Glasserman. Monte carlo methods in financial engineering. Springer, 2004.

[11] P. Boyle, M. Broadie, and P. Glasserman. Monte carlo methods for security pricing. *Journal of Economics and Dynamic Control,.*, 21, 1997.

[12] Sedgewick. Algorithms. Addison-Wesley,Reading,MA., 1987.

[13] Elliott and P.E. Kopp. Mathematics of financial markets. Springer, 1999.

[14] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy, and Dynamic Control.*, 81(3),637-654, 1973.

[15] R.C. Merton. Theory of rational option pricing. *Bell Journal of Economics and Mangement Science.*, 4,141-183, 1973.

[16] Tilley and J.A. valuing american options in a path simulation model. *Transactions of the Society of Actuaries.*, 45,499-549, 1993.

[17] I.M. Sobol. The monte carlo method. The University of Chicago Press, 1967.

[18] Tilley and J.A. valuing american options in a path simulation model and discussion. *Transactions of the Society of Actuaries*, 45,93-104, 1993.