On the pricing of barrier options

Dissertation for MSc Financial Mathematics

David Tor Bonde — 1469023

Advisor: Professor István Gyöngy

School of Mathematics
University of Edinburgh

August 2015

Colophon

This document was created using the LaTeX 2ε typesetting system, distributed through TeX Live. References were managed by BibTeX and all figures were created using Ti*k*Z and PGFPLOTS. The text is set in Adobe Utopia and mathematics is set in Fourier, both available from the `fourier`-package. The source code is set in Bera Mono. All references and citations are clickable internal links, as made possible by the `hyperref`-package.

All code was implemented in the MATLAB language and can be found on `https://github.com/torbonde/dissertation`. Simulations were run on a medio 2014 MacBook Pro with a 2.6 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 RAM in the MATLAB 2015b prerelease. This version of MATLAB was chosen in particular, due to the massive speed-up provided by the new JIT compiler.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Acknowledgements

I would like to acknowledge my advisor Professor István Gyöngy, who has been of some help with this project. I would furthermore like to thank my fellow students, and in particular Connor O'Neill, Darren Morrison and Christos Karamitsos, for many valuable discussions - some even on the topics of our dissertations.

I am forever grateful for the opportunities The Traveling Scholarship for Mathematicians has given me, without which the past year would not have been a reality. Finally, I owe it all to my family; my fiancé Janni and my son Magne. Thank you for taking this adventure with me. Many more will follow, I promise.

University of Edinburgh
Own Work Declaration

This sheet must be filled in (each box ticked to show that the condition has been met), signed and dated, and included with all assessments - work will not be marked unless this is done.

Name: ……………………………………..

Matriculation Number: …………………

Programme: ………………………………………………………………….

Dissertation Title:…………………………………………………………………………….

………………………………………………………………………………………………..

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate ☐
- Referenced and put in inverted commas all quoted text (from books, web, etc) ☐
- Given the sources of all pictures, data etc. that are not my own ☐
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present ☐
- Not sought or used the help of any external professional academic agencies for the work ☐
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources) ☐
- Complied with any other plagiarism criteria specified in the Course handbook ☐

I understand that any false claim for this work will be penalised in accordance with the University regulations ☐

☐ Should it be selected, I agree to allow my dissertation to be used by the School of Geosciences as an example of an exemplary dissertation.

Signature ……………………………………………**Date** …………………………………

Please note: If you need further guidance on plagiarism, you can

1. Consult your programme handbook
2. Speak to your programme director
3. Check out: http://www.ed.ac.uk/schools-departments/academic-services/students/postgraduate-taught/discipline/plagiarism

Contents

1

INTRODUCTION

The purpose of this chapter is to introduce the topic of discussion in this project. In Section 1.1 the scene is set by describing the model and in Section 1.2 some background information about barrier options is given, including some known pricing formulas.

1.1 The model

Let $(\Omega, \mathbf{P}, \mathscr{F}, \mathbb{F} = (\mathscr{F}_t)_{t\geq 0})$ be a filtered probability space, where $\mathscr{F}_t = \sigma(W_s : 0 \leq s \leq t)$ is the filtration generated by the Wiener process $W = (W_t)_{t\geq 0}$. The model under consideration in this paper will be the standard one-dimensional Black-Scholes model without dividends. That is, the market is assumed to consist of a riskless asset $B = (B_t)_{t\geq 0}$ given by

$$\mathrm{d}B_t = rB_t\,\mathrm{d}t, \quad B_0 = 1,$$

with the solution $B = \mathrm{e}^{rt}$ and a single risky asset $S = (S_t)_{t\geq 0}$ given by

$$\mathrm{d}S_t = \mu S_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t. \tag{1.1}$$

Here the risk-free rate of return $r \geq 0$, the drift $\mu \in \mathbb{R}$, the volatility $\sigma > 0$ and the spot price $S_0 \geq 0$ are fixed constants. For the sake of simplicity assume further that $\mathbf{P}$ is the risk-neutral measure, such that $\mu = r$. Then the solution to (1.1) is $S_t = S_0\mathrm{e}^{(r-\frac{1}{2}\sigma^2)t+\sigma W_t}$ and $\mathbf{E}S_t = S_0\mathrm{e}^{rt}$, i.e. the stock price is expected to grow at the risk-free rate.

The price of a European type option with payoff $h\big((S_t)_{t\in[0,T]}\big)$ at time t is given by

$$\mathrm{e}^{-r(T-t)}\mathbf{E}\big(h\big((S_t)_{t\in[0,T]}\big) \,\big|\, \mathscr{F}_t\big).$$

Note that the payoff may depend on the entire path of S. This kind of option is known as a path-dependent.

1.2 Barrier options

Barrier options are a type of path dependent option, where the payoff is conditioned on the movement of the underlying during the life time of the option. The two types of barrier options that will be considered in this paper are single barrier options and double barrier options. Each of these types can be further classified as either a knock-out or a knock-in option. This classification determines how the payoff is conditioned on the path of the underlying.

Type	Barrier	Payoff
Up-and-out	$B > S_0$	$h(S_T)\mathbb{1}_{\{S_t<B,\text{ for all } t\in[0,T)\}}$
Up-and-in	$B > S_0$	$h(S_T)\mathbb{1}_{\{S_t\geq B,\text{ for some } t\in[0,T)\}}$
Down-and-out	$B < S_0$	$h(S_T)\mathbb{1}_{\{S_t>B,\text{ for all } t\in[0,T)\}}$
Down-and-in	$B < S_0$	$h(S_T)\mathbb{1}_{\{S_t\leq B,\text{ for some } t\in[0,T)\}}$

Table 1.1: Payoffs for single barrier options, where $h(x)$ is the payoff. E.g. $h(x) = (x-K)^+$ for call options, and $h(x) = (K-x)^+$ for put options.

In the case of a single barrier knock-out option with a barrier $B > 0$, the option is nullified if the underlying reaches the barrier B. Likewise, for a knock-in option, the option only comes into existence if the underlying reaches the level B. In each case, there are two interesting possibilities; $B > S_0$ and $B < S_0$. Obviously, the uninteresting case when $B = S_0$ yields either a worthless option or a European type option, depending on whether the option is a knock-out or a knock-in type option.

When $B > S_0$, the knock-out and knock-in options are for simplicity called up-and-out and up-and-in options, respectively. For $B < S_0$ they are called down-and-out and down-and-in options. The payoffs of these different types of barrier options are described in Table 1.1.

Where single barrier options have only one barrier, double barrier options have, a bit unsurprisingly perhaps, two barriers; one lower and one upper, $L < S_0 < U$. As before, these can be classified into knock-out options and knock-in options. A double barrier option is hence either knocked out or knocked in, if the underlying reaches one of the barriers before maturity. The payoffs of double barrier options are listed in Table 1.2.

Type	Payoff
Knock-out	$h(S_T)\mathbb{1}_{\{L<S_t<U,\text{ for all } t\in[0,T)\}}$
Knock-in	$h(S_T)\mathbb{1}_{\{S_t\leq L\text{ or } S_t\geq U,\text{ for some } t\in[0,T)\}}$

Table 1.2: Payoffs for double barrier options, where $h(x)$ is the payoff. E.g. $h(x) = (x-K)^+$ for call options, and $h(x) = (K-x)^+$ for put options.

All barrier options can be further classified by on how they are monitored; discretely or continuously. Both are traded on the exchanges, although discretely monitored barrier options are more common. There are several ways of pricing such options. In some cases, discretely monitored options are more easy to deal with, while for others continously monitored options a more easy; see Chapters 4 and 3, respectively. In the same way, some methods deal more easily with single barrier options, whereas some are more well suited for double barrier options; see again Chapters 4 and 3. These features and more will be discussed later in this paper.

1.2.1 Option pricing formulas

As previously stated, this paper will focus on different methods for pricing barrier optioins. Before going in depth with these, it is of course worth mentioning some well known analytical formulas for pricing options.

Below are formulas for continuously monitored barrier call options in a number of cases. Formulas for put options will not be stated, since the pricing of these are outside the scope of this paper. The formulas for the single barrier options are taken from Hull (2009), and the formula for the double barrier option is from Kunitomo and Ikeda (1992). Note that in each case the knock-in and knock-out options are related through the in-out parity

$$c = c_{\mathrm{i}} + c_{\mathrm{o}}, \tag{1.2}$$

where c_{i} is the price of an in call option, and c_{o} is the price of the corresponding out call option. This parity holds of course for all other payoffs h as well, since it relies only on

$$\mathbf{E}h(S_T) = \mathbf{E}[h(S_T);\text{barrier breached}] + \mathbf{E}[h(S_T);\text{barrier not breached}].$$

The present value of a down-and-in call is given by

$$c_{\mathrm{di}} = S_0\left(\frac{B}{S_0}\right)^{2\lambda}\Phi(x) - K\mathrm{e}^{-rT}\left(\frac{B}{S_0}\right)^{2\lambda-2}\Phi(x-\sigma\sqrt{T})$$

where K is the strike price, B is the barrier and

$$\lambda = \frac{r}{\sigma^2}+\frac{1}{2} \qquad \text{and} \qquad x = \frac{\log\big(B^2/(S_0K)\big)}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T},$$

when $K \geq B$. The down-and-out call price is given through the in-out parity (1.2). In the case when $K \leq B$ the down-and-out call price is given by

$$c_{\mathrm{do}} = S_0\Phi(y_1) - K\mathrm{e}^{-rT}\Phi(y_1-\sigma\sqrt{T}) - S_0\left(\frac{B}{S_0}\right)^{2\lambda}\Phi(y_2) + K\mathrm{e}^{-rT}\left(\frac{B}{S_0}\right)^{2\lambda-2}\Phi(y_2-\sigma\sqrt{T}),$$

where

$$y_1 = \frac{\log(S_0/B)}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T} \qquad \text{and} \qquad y_2 = \frac{\log(B/S_0)}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}$$

and the down-and-in call price is given through (1.2). In each of the above two cases one can of course also consider the up-options. However, notice that when $K \geq B$ the up-and-out call is worthless, and hence the up-and-in call has the same value as a regular call option. In the case when $K \leq B$ the up-and-in price is

$$\begin{aligned} c_{\mathrm{ui}} = S_0\Phi(y_1) - K\mathrm{e}^{-rT}\Phi(y_1-\sigma\sqrt{T}) - S_0\left(\frac{B}{S_0}\right)^{2\lambda}&(\Phi(-x)-\Phi(-y_2)) \\ + K\mathrm{e}^{-rT}\left(\frac{B}{S_0}\right)^{2\lambda-2}&(\Phi(-x+\sigma\sqrt{T})-\Phi(-y_2+\sigma\sqrt{T})) \end{aligned}$$

and the corresponding out-option is given by (1.2).

The price of the double barrier knock-out call option is given by

$$\begin{aligned} c_{\mathrm{ko}} = S_0 \sum_{n=-\infty}^{\infty}&\left(\left(\frac{U^n}{L^n}\right)^{2\lambda}(\Phi(d_{1,n})-\Phi(d_{2,n})) - \left(\frac{L^{n+1}}{S_0U^n}\right)^{2\lambda}(\Phi(d_{3,n})-\Phi(d_{4,n}))\right) \\ &- K\mathrm{e}^{-rT}\sum_{n=-\infty}^{\infty}\left(\left(\frac{U^n}{L^n}\right)^{2\lambda-2}(\Phi(d_{1,n}-\sigma\sqrt{T})-\Phi(d_{2,n}-\sigma\sqrt{T}))\right. \\ &\qquad\left. - \left(\frac{L^{n+1}}{S_0U^n}\right)^{2\lambda-2}(\Phi(d_{3,n}-\sigma\sqrt{T})-\Phi(d_{4,n}-\sigma\sqrt{T}))\right), \end{aligned}$$

where

$$d_{1,n} = \frac{\log(S_0 U^{2n}/(KL^{2n}))}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}, \qquad d_{2,n} = \frac{\log(S_0 U^{2n}/(KL^{2n}))}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}$$
$$d_{3,n} = \frac{\log(L^{2n+2}/(S_0 K U^{2n}))}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}, \quad d_{4,n} = \frac{\log(L^{2n+2}/(S_0 U^{2n+1}))}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}.$$

Even though c_{ko} is given as an infinite sum, in practice only quite few terms are necessary.

Remark. The formulas in this section gives the value of continuously monitored barrier options. Formulas for discrete barrier options does not exist, however, one may use the continuity correction introduced by Broadie et al. (1997) to fit these to the discrete case. •

2

FOURIER APPROACH

In this chapter the focus is on a direct approach presented by Borovkov and Novikov (2002). Suppose one wishes to calculate the expectations $P_1 := \mathbf{E}(e^X - K)^+$ and $P_2 := \mathbf{E}[(e^X - K)^+; Z \le B]$ for some random variables X and Z, and real numbers K and B, where for any set $A \subset \Omega$, $\mathbf{E}[X; A] := \mathbf{E}[X \mathbb{1}_A]$. In order to use the following method, the moment generating functions φ and ψ of X and (X, Z) are required to be known;

$$\varphi(u) = \mathbf{E}e^{uX} \qquad \text{and} \qquad \psi(u, v) = \mathbf{E}e^{uX + vZ}.$$

Section 2.1 presents analytical expressions for φ and ψ under the model described in Section 1.1.

The following theorem was presented in Borovkov and Novikov (2002).

Theorem 2.1 *Let $K > 0$ be any positive, real number, and let $a > 0$ be such that*

$$\varphi(1 + a) < \infty. \tag{2.1}$$

The expectation P_1 is given by

$$P_1 = \frac{1}{2\pi} \int_{-\infty}^{\infty} \varphi(1 + a - is) g(s) \, ds$$

where $g(s) = \frac{e^{isb - ab}}{(a - is)(1 + a - is)}$ and $b = \log K$.

Let now B be any real number. The expectation P_2 is then given by $P_2 = P_1 W(B)$, where W is the distribution function determined by the characteristic function

$$w(v) = \frac{1}{2\pi P_1} \int_{-\infty}^{\infty} \psi(1 + a - is, iv) g(s) \, ds. \tag{2.2}$$

Proof. In order to prove the first part of the theorem, observe first that for any real numbers a and x

$$(e^x - K)^+ = e^{(1+a)x} G(x) \quad \text{with} \quad G(x) = e^{-ax} H(x) \quad \text{and} \quad H(x) = (1 - e^{b-x})^+. \tag{2.3}$$

Next, see that for any complex s with $\operatorname{Im} s \ge 0$

$$\int_{-\infty}^{\infty} e^{isx} H'(x) \, dx = \frac{e^b}{is - 1} \left[e^{(is-1)x} \right]_b^{\infty} = \frac{e^{isb}}{1 - is}, \qquad H'(x) = \begin{cases} 0 & x < b \\ e^{b-x} & \text{else} \end{cases}$$

where H' is the derivative of H. Thus, taking now s to be real and $a > 0$,

$$\int_{-\infty}^{\infty} e^{i(s+ia)x} H'(x) \, dx = \frac{e^{i(s+ia)b}}{1 - i(s + ia)} = \frac{e^{isb - ab}}{1 + a - is}.$$

However, integrating the left-hand side by parts, one also notices that

$$\begin{aligned}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(s+\mathrm{i}a)x} H'(x) &= \left[\mathrm{e}^{\mathrm{i}(s+\mathrm{i}a)x} H(x)\right]_{-\infty}^{\infty} - \mathrm{i}(s+\mathrm{i}a)\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(s+\mathrm{i}a)x} H(x)\,\mathrm{d}x \\ &= \left[\mathrm{e}^{\mathrm{i}(s+\mathrm{i}a)x}(1-\mathrm{e}^{b-x})\right]_{b}^{\infty} - \mathrm{i}(s+\mathrm{i}a)\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}sx} G(x)\,\mathrm{d}x\end{aligned}$$

wherein the first summand vanishes (since $a > 0$) and the integral is the Fourier transform of G, hence

$$g(s) = \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}sx} G(x)\,\mathrm{d}x = \frac{\mathrm{e}^{\mathrm{i}sb-ab}}{(a-\mathrm{i}s)(1+a-\mathrm{i}s)}$$

Obviously g is integrable on $\mathbb{R}$, and so G can be represented as the inverse Fourier transform of g, giving finally an expression for P_1;

$$P_1 = \mathbf{E}\left[\mathrm{e}^{(1+a)X} G(X)\right] = \frac{1}{2\pi}\mathbf{E}\int_{-\infty}^{\infty} \mathrm{e}^{(1+a-\mathrm{i}s)X} g(s)\,\mathrm{d}s = \frac{1}{2\pi}\int_{-\infty}^{\infty} \varphi(1+a-\mathrm{i}s) g(s)\,\mathrm{d}s.$$

Note that the last equality follows by Fubini's theorem, which can be applied due to the assumption (2.1) giving absolute integrability.

For the second part of the theorem see that

$$\begin{aligned}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{(\mathrm{e}^x-K)^+}{P_1} p_{X,Z}(x,z)\,\mathrm{d}x\mathrm{d}z &= \frac{1}{P_1}\int_{-\infty}^{\infty} (\mathrm{e}^x-K)^+ p_X(x) \int_{-\infty}^{\infty} p_{Z|X=x}(z)\,\mathrm{d}z\,\mathrm{d}x \\ &= \frac{1}{P_1}\int_{-\infty}^{\infty} (\mathrm{e}^x-K)^+ p_X(x)\,\mathrm{d}x = 1,\end{aligned}$$

where $p_{X,Z}$, p_X and $p_{Z|X}$ are densities for (X,Z), X and $Z|X$ under $\mathbf{P}$, respectively. Hence $x \mapsto (\mathrm{e}^x - K)^+/P_1$ is a density with respect to the original distribution of (X,Z), and

$$P_2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\mathrm{e}^x-K)^+ \mathbb{1}_{\{z\le B\}} p_{X,Z}(x,z)\,\mathrm{d}x\,\mathrm{d}z = P_1 \mathbf{P}^*(Z \le B)$$

where $\mathbf{P}^*$ is the probability measure with this density. Letting p_Z^* be the density for Z under $\mathbf{P}^*$, see finally that

$$\begin{aligned}w(v) = \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}vz} p_Z^*(z)\,\mathrm{d}z &= \frac{1}{P_1}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}vz}(\mathrm{e}^x-K)^+ p_{X,Z}(x,z)\,\mathrm{d}x\,\mathrm{d}z \\ &= \frac{1}{2\pi P_1}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}vz} p_{X,Z}(x,z)\int_{-\infty}^{\infty} \mathrm{e}^{(1+a-\mathrm{i}s)x} g(s)\,\mathrm{d}s\,\mathrm{d}x\,\mathrm{d}z \\ &= \frac{1}{2\pi P_1}\int_{-\infty}^{\infty} \psi(1+a-\mathrm{i}s, \mathrm{i}v) g(s)\,\mathrm{d}s,\end{aligned}$$

which is the characteristic function of Z under $\mathbf{P}^*$, and so $W(B) = \mathbf{P}^*(Z \le B)$. The third equality comes from (2.3) and that G is the Fourier transform of g; $G(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}sx} g(x)\,\mathrm{d}x$. □

The parameter a in the proof can according to Borovkov and Novikov (2002) be chosen arbitrarily. In the implementation given in Appendix A this is chosen to be 9.

Remark. The method introduced in the above theorem and its proof can easily be extended to the case when there are two barriers. Suppose for instance that one wishes to calculate the expectation $P_3 = \mathbf{E}[(\mathrm{e}^X - K)^+; Y \le B_-, Z \le B_+]$, and that the moment generating function χ of (X,Y,Z) is known. By virtually the same arguments as in the second part of the proof, this expectation is given by $P_3 = P_1 \bar{W}(B_-, B_+)$ where $\bar{W}$ is the distribution function with the characteristic function

$$\bar{w}(u,v) = \frac{1}{2\pi P_1}\int_{-\infty}^{\infty} \chi(1+a-\mathrm{i}s, \mathrm{i}u, \mathrm{i}v) g(s)\,\mathrm{d}s. \tag{2.4}$$

•

Under the current model, and for the applications to vanilla European, single barrier up-and-out and double barrier knock-out call options, one obvious way of defining the random variables X, Y and Z is by

$$X = X_T, \quad Y = -\min_{t\in[0,T]} X_t, \quad Z = \max_{t\in[0,T]} X_t$$

where $X_t = \big(r - \frac{1}{2}\sigma^2\big)t + \sigma W_t$ as before. Suppose the strike price of the option is K', the barrier for the up-and-out option is B' and the lower and upper barriers for the knock-out option are L and U. Then, taking $K := K'/S_0$, $B := \log(B'/S_0)$, $B_- := \log(S_0/L)$ and $B_+ := \log(U/S_0)$, the price of the European call option is $c = \mathrm{e}^{-rT}S_0P_1$, the price of the up-and-out call option is $c_{\mathrm{uo}} = \mathrm{e}^{-rT}S_0P_2$ and the proce of the knock-out call option is $c_{\mathrm{ko}} = \mathrm{e}^{-rT}S_0P_3$.

2.1 Moment generating functions

In order to apply Theorem 2.1 and Remark 2, it is necessary to first arrive at expressions for the moment generating functions φ, ψ and χ. As a first step towards obtaining these, define $\theta := \frac{r}{\sigma} - \frac{\sigma}{2}$ and $W_t^* := W_t + \theta t$, and observe that by the Girsanov theorem $(W_t^*)_{t\in[0,T]}$ is a Wiener martingale with respect to the history of W, under some probability measure $\mathbf{P}^*$. Note that this is *not* the same probability measure as in the proof of Theorem 2.1. Letting now $m_T = \min_{t\in[0,T]} W_t^*$ and $M_T = \max_{t\in[0,T]} W_t^*$ one can see that

$$\begin{aligned}\varphi(u) &= \mathrm{e}^{-\frac{1}{2}\theta^2 T}\bar{\varphi}(u\sigma+\theta),\\ \psi(u,v) &= \mathrm{e}^{-\frac{1}{2}\theta^2 T}\bar{\psi}(u\sigma+\theta, v\sigma),\\ \chi(u,v,w) &= \mathrm{e}^{-\frac{1}{2}\theta^2 T}\bar{\chi}(u\sigma+\theta, v\sigma, w\sigma),\end{aligned} \tag{2.5}$$

where $\bar{\varphi}$, $\bar{\psi}$ and $\bar{\chi}$ are the moment generating functions of W_T^*, (W_T^*, M_T) and $(W_T^*, -m_T, M_T)$ under $\mathbf{P}^*$, respectively.

It can be seen quite easily that $\bar{\varphi}(u) = \mathrm{e}^{\frac{1}{2}Tu^2}$ and hence

$$\varphi(u) = \mathrm{e}^{-\frac{1}{2}\theta^2 T}\mathrm{e}^{\frac{1}{2}T(u\sigma+\theta)^2} = \mathrm{e}^{\left(r-\frac{1}{2}\sigma^2\right)u+\frac{1}{2}T\sigma^2u^2}.$$

Of course, this could have been seen directly, by noting that $X_t \sim N\big(r - \frac{1}{2}\sigma^2, t\sigma^2\big)$ and using the well known formula for the moment generating function of a normally distributed random variable.

When it comes to the calculation of ψ and χ, however, the representation in (2.5) is really rather useful. The density function of the pair (W_T^*, M_T) is given by

$$p(x,z) = \begin{cases}\sqrt{\frac{2}{\pi}}\frac{2z-x}{T^{3/2}}\mathrm{e}^{-\frac{(2z-x)^2}{2T}} & \text{when } x \le y, y \ge 0\\ 0 & \text{else}\end{cases},$$

and can be found in Karatzas and Shreve (1991). From this one can arrive at the following expression for $\bar{\psi}$;

$$\bar{\psi}(u,v) = 2\Phi\big(\sqrt{T}(u+v)\big)\mathrm{e}^{\frac{1}{2}T(u+v)^2}\frac{u+v}{2u+v} + \Phi\Big(\frac{1}{2}\sqrt{T}v\Big)\mathrm{e}^{\frac{1}{8}v^2}\left(\frac{2u\mathrm{e}^{\frac{1}{4}Tu(2u+v)}}{2u+v} - 1\right).$$

Doing so is a matter of will power more than a matter of brain power, although a little of both is advisible. In lack of either, one can use instead a computer algebra system, which is what was done here.

As might perhaps be expected, obtaining $\bar{\chi}$ is less trivial, and even the density of $(W_T^*, -m_T, M_T)$ is not easy to come by. It can be found, however, from the following expression given in Geman and Yor

(1996);

$$\begin{aligned}
&\mathbf{P}\big(W_T^* \in \mathrm{d}x, y \le m_T \le M_T \le z\big)\\
&\quad = \frac{1}{\sqrt{2\pi T}} \sum_{k=-\infty}^{\infty} \left[\exp\Big(-\frac{1}{2T}(x-2k(z-y))^2\Big) - \exp\Big(-\frac{1}{2T}((x-2z)+2k(z-a))^2\Big)\right] \mathrm{d}x.
\end{aligned}$$

Differentiating with respect to y and z, and substituting $-y$ for y, one obtains the density $\bar{p}$ of the triple $(W_T^*, -m_T, M_T)$

$$\begin{aligned}
\bar{p}(x,y,z) = \frac{1}{\sqrt{2\pi T}} \sum_{k=-\infty}^{\infty} \frac{4k}{2T} \Bigg[& k(T-(2k(z+y)+x)^2)\exp\Big(-\frac{1}{2T}(x+2k(z+y))^2\Big)\\
&+ (k-1)(((x-2z)+2k(z+y))^2 - T)\exp\Big(-\frac{1}{2T}((x-2z)+2k(z+y))^2\Big)\Bigg].
\end{aligned}$$

Finally $\bar{\chi}$ is given by the three-dimensional integral

$$\bar{\chi}(u,v,w) = \int_0^\infty \int_{-\infty}^0 \int_y^z \mathrm{e}^{ux+vy+wz}\, \bar{p}(x,y,z)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z. \tag{2.6}$$

2.2 Remarks

The method discussed in this chapter is of a fairly general kind. In fact, all it requires is the knowledge of certain moment generating functions. This requirement does not put any immediate restrictions on the model, although Borovkov and Novikov (2002) does require the underlying process to be of Lévy type. It will appear, however, that this restriction is merely necessary due to the example in the same paper, in which they apply a result that requires the process to have independent increments.

This generality does in theory open up for applications to a wide range of models, apart from the beloved Black-Scholes model. One of the reasons for the well-likedness of this model is the ease with which one can calculate so many things; e.g. barrier option prices, as was seen in Section 1.2.1. However, even in this relatively simple model, the expressions for the moment generating function needed for pricing barrier options given in Section 2.1 are not very nice to work with.

There is another important point to make, when it comes to the application of the second part of Theorem 2.1 and Remark 2. When pricing a single barrier option, one needs to invert the characteristic function w in (2.2), in order to evaluate the distribution function W. To do so, Shephard (1991) suggest the following formula

$$W(x) = \frac{1}{2} - \frac{1}{2\pi}\int_0^\infty \underset{u}{\Delta}\left[\frac{1}{\mathrm{i}u} w(u)\mathrm{e}^{-\mathrm{i}ux}\right]\mathrm{d}u \tag{2.7}$$

where the operator Δ is given by $\Delta_u f(u) = f(u) + f(-u)$. For double barrier options the situation is similar, but more complicated, since the characteristic function which needs to be inverted is the two-dimensional $\bar{w}$ in (2.4). This can be done by the recursive relation (Shephard, 1991, Section 4)

$$2\bar{W}(x,y) - W_1(x) - W_2(y) + \frac{1}{2} = -\frac{1}{\pi^2}\int_0^\infty \int_0^\infty \underset{v}{\Delta}\,\mathrm{Re}\left[\frac{1}{uv}\bar{w}(u,v)\mathrm{e}^{-\mathrm{i}ux-\mathrm{i}vy}\right]\mathrm{d}u\,\mathrm{d}v, \tag{2.8}$$

where W_1 is the distribution function with characteristic function $\bar{w}(\cdot,0)$ and W_1 is the distribution function with characteristic function $\bar{w}(0,\cdot)$. Note that W_1 is in fact W as given by (2.7), and that W_2 can be expressed in a similar fashion.

Evaluating (2.7) requires estimating a two-dimensional integral where the integrand is described in terms of the standard normal distribution function Φ. Performing the two-dimensional quadrature was manageable, however one problem was introduced when evaluating Φ at complex numbers, which MATLAB does not support natively. This problem was sought solved using the `erfc` function of the `Faddeeva` package[1], which unfortunately produces not-a-number and infinite values for some inputs. For this reason, the method has not been succesfully applied.

Obviously, evaluating $\bar{W}$ is more complicated than evaluating W, since it requires evaluating two integrals of the type (2.7) along with the double integral in (2.8). Due to (2.4), $\bar{w}$ is itself an integral and its integrand involves $\bar{\chi}$. By (2.6) $\bar{\chi}$ is a three-dimensional integral with its integrand given by an infinite series. Putting all of this together, evaluating $\bar{W}$ requires calculating a six-dimensional integral of an infinite series over an infinite domain. Numerical evaluation of multi-dimensional integrals are not generally handled well by standard quadrature schemes, and it is often beneficial to estimate the integral using Monte Carlo methods (Heath, 2001). This has not been investigated further.

Despite the problems described above, the method has a certain beauty in its own right. Borovkov and Novikov (2002) gives an example wherein they price a discretely monitored lookback option under two different models. One model is the same as in this paper, and the other is the variance-gamma model where the risky asset is described by

$$S_t = S_0 \mathrm{e}^{(r-\frac{1}{2}\sigma^2)t+\sigma W_{\gamma_t}},$$

where γ is a gamma process independent of W with $\mathbf{E}\gamma_t = t$ and $\mathbf{Var}(\gamma_t) = t$. This is only possible because of how generic the method is.

A function for pricing a European call option under the current model has been implemented in `bn_eu_call.m`, using the methods described in this chapter. In this case the method performs reasonably well.

[1] `http://ab-initio.mit.edu/Faddeeva`

3

FINITE DIFFERENCES

This chapter will focus on finite differences, which is a method for numerically solving partial differential equations. Two partial differential equations will be considered for each of the case of single and double barrier options. First, a result on the existence of a unique solution to the partial differential equation (3.1), and a result on the relationship between this solution and the stochastic differential equation (3.3).

Let in the following $\mathrm{D}_t u(t,x) := \frac{\partial}{\partial t}u(t,x)$, $\mathrm{D}_x u(t,x) := \frac{\partial}{\partial x}u(t,x)$ and $\mathrm{D}_{xx} := \frac{\partial^2}{\partial x^2}u(t,x)$ for $u \in C^{1,2}$ and let the elliptic operator L be given by

$$\mathrm{L}u(t,x) := b(t,x)\mathrm{D}_x u(t,x) + \frac{1}{2}a(t,x)\mathrm{D}_{xx}u(t,x) + c(t,x)u(t,x)$$

with $a(t,x) = \sigma^2(t,x)$ on $\bar{H}$, where $H = [0,T)\times Q$, Q is an open interval in $\mathbb{R}$, and σ, b and c are Lipschitz functions. Consider the following partial differential equation

$$\mathrm{D}_t u(t,x) + \mathrm{L}u(t,x) = f(t,x) \qquad \text{for } (t,x) \in H \tag{3.1a}$$

$$u(t,x) = g(t,x) \qquad \text{for } (t,x) \in B \tag{3.1b}$$

$$u(T,x) = h(x) \qquad \text{for } x \in Q \tag{3.1c}$$

for some functions f, g and h, where $B = [0,T)\times\partial Q$. The following result is taken from Friedman (2006).

Theorem 3.1 *Assume that a is bounded away from zero; i.e. there is a fixed $\varepsilon > 0$ such that $a(t,x) \geq \varepsilon$ for all $(t,x) \in H$, and assume furthermore that a and b are uniformly Lipschitz continuous on $\bar{H}$ and that c and f are uniformly Hölder continuous on $\bar{H}$. Let now g and h be functions such that g is continuous on $\bar{B}$, h is continuous on $\bar{Q}$ and*

$$g(T,x) = h(x) \quad \textit{for all } x \in \partial Q. \tag{3.2}$$

Then there exist a unique solution $u \in C^{1,2}([0,T)\times Q)$ to the partial differential equation (3.1).

Let now $t \in [0,T]$, $x \in \mathbb{R}$ and consider the stochastic differential equation

$$\mathrm{d}S_s = b(s,S_s)\,\mathrm{d}s + \sigma(s,S_s)\,\mathrm{d}W_s, \qquad s \in [t,T], \qquad S_t = x. \tag{3.3}$$

This has a unique solution, which in the following will be denoted by $S^{t,x} = (S_s^{t,x})_{s\in[t,T]}$. Below, this superscript might sometimes be dropped when the process is started at $t = 0$ and the initial value is not important. As the following theorem will show, the partial differential equation (3.1) is very closely related to $S^{t,x}$.

Theorem 3.2 *Assume that the conditions on a, b, c, f, g and h from Theorem 3.1 hold. Let* $\tau^{t,x} := \inf\{s \geq t : S^{t,x}_s \notin Q\}$ *be the first exit time of the process* $S^{t,x}$ *from the interval Q and let furthermore* $Z^{t,x}_s := \mathrm{e}^{\int_t^s c(q,S^{t,x}_q)\,\mathrm{d}q}$. *Then*

$$u(t,x) = \mathbf{E}\big[Z^{t,x}_{\tau^{t,x}} g(\tau^{t,x}, S^{t,x}_{\tau^{t,x}}) \mathbb{1}_{\tau^{t,x}<T}\big] + \mathbf{E}\big[Z^{t,x}_T h(S^{t,x}_T)\mathbb{1}_{\tau^{t,x}\geq T}\big] - \mathbf{E}\Big[\int_t^{\tau^{t,x}\wedge T} Z^{t,x}_s f(s,S^{t,x}_s)\,\mathrm{d}s\Big]$$

is the unique solution to (3.1).

Proof. Let $u \in C^{1,2}([0,T)\times Q)$ be the unique solution to (3.1) and take $s \in [t,T]$ and observe that $\mathrm{d}Z^{t,x}_s = c(s,S^{t,x}_s)Z^{t,x}_s\,\mathrm{d}s$. Then by Itô's formula

$$\begin{aligned}
\mathrm{d}\big(Z^{t,x}_s u(s,S^{t,x}_s)\big) &= Z^{t,x}_s \mathrm{d}u(s,S^{t,x}_s) + u(s,S^{t,x}_s)\mathrm{d}Z^{t,x}_s + \mathrm{d}Z^{t,x}_s\,\mathrm{d}u(s,S^{t,x}_s)\\
&= Z^{t,x}_s\big(\mathrm{D}_t u(s,S^{t,x}_s) + b(s,S^{t,x}_s)\mathrm{D}_x u(s,S^{t,x}_s) + \frac{1}{2}\sigma^2(s,S^{t,x}_s)\mathrm{D}_{xx}u(s,S^{t,x}_s)\big)\,\mathrm{d}s\\
&\quad + Z^{t,x}_s\sigma(s,S^{t,x}_s)\mathrm{D}_x u(s,S^{t,x}_s)\,\mathrm{d}W_s + Z^{t,x}_s c(s,S^{t,x}_s)u(s,S^{t,x}_s)\,\mathrm{d}s\\
&= Z^{t,x}_s(\mathrm{D}_t + L)u(s,S^{t,x}_s)\,\mathrm{d}s + Z^{t,x}_s\sigma(s,S^{t,x}_s)\mathrm{D}_x u(s,S^{t,x}_s)\,\mathrm{d}W_s,
\end{aligned}$$

which can be written equivalently in its integral form

$$\begin{aligned}
Z^{t,x}_s u(s,S^{t,x}_s) &= u(t,x) + \int_t^s Z^{t,x}_q(\mathrm{D}_t+L)u(q,S^{t,x}_q)\,\mathrm{d}q + \int_t^s Z^{t,x}_q\sigma(q,S^{t,x}_q)\mathrm{D}_x u(q,S^{t,x}_q)\,\mathrm{d}W_q\\
&= u(t,x) + \int_t^s Z^{t,x}_q f(q,S^{t,x}_q)\,\mathrm{d}q + m_s,
\end{aligned} \tag{3.4}$$

where $m_s := \int_t^s Z^{t,x}_q\sigma(q,S^{t,x}_q)\mathrm{D}_x u(q,S^{t,x}_q)\,\mathrm{d}W_q$ for $s\in[t,T]$ and the last equality follows from (3.1a). Let now $\rho^{t,x} := \tau^{t,x}\wedge T$ and notice that the stopped process $(m_{s\wedge\rho^{t,x}})_{s\in[t,T]}$ is a martingale, and hence $\mathbf{E}m_{s\wedge\rho^{t,x}} = 0$ for any $s\in[t,T]$. Replacing s with $\rho^{t,x}$ in (3.4) and taking expectations, one gets

$$\begin{aligned}
u(t,x) &= \mathbf{E}[Z^{t,x}_{\rho^{t,x}} u(\rho^{t,x}, S^{t,x}_{\rho^{t,x}})] - \mathbf{E}\Big[\int_t^{\rho^{t,x}} Z^{t,x}_s f(s,S^{t,x}_s)\,\mathrm{d}s\Big]\\
&= \mathbf{E}[Z^{t,x}_{\rho^{t,x}} u(\rho^{t,x}, S^{t,x}_{\rho^{t,x}})\mathbb{1}_{\tau^{t,x}<T}] + \mathbf{E}[Z^{t,x}_{\rho^{t,x}} u(\rho^{t,x}, S^{t,x}_{\rho^{t,x}})\mathbb{1}_{\tau^{t,x}\geq T}] - \mathbf{E}\Big[\int_t^{\rho^{t,x}} Z^{t,x}_s f(s,S^{t,x}_s)\,\mathrm{d}s\Big]\\
&= \mathbf{E}\big[Z^{t,x}_{\tau^{t,x}} g(\tau^{t,x}, S^{t,x}_{\tau^{t,x}})\mathbb{1}_{\tau^{t,x}<T}\big] + \mathbf{E}\big[Z^{t,x}_T h(S^{t,x}_t)\mathbb{1}_{\tau^{t,x}\geq T}\big] - \mathbf{E}\Big[\int_t^{\tau^{t,x}\wedge T} Z^{t,x}_s f(s,S^{t,x}_s)\,\mathrm{d}s\Big].
\end{aligned}$$

The third equality follows by (3.1b) and (3.1c). □

Under the classic Black-Scholes model (1.1) considered in this paper, $b(t,x) = rx$, $a(t,x) = \sigma^2x^2$ and $c \equiv -r$, where σ is a constant, and L is hence given by

$$\mathrm{L}u(t,x) = rx\mathrm{D}_x u(t,x) + \frac{1}{2}\sigma^2x^2\mathrm{D}_{xx}u(t,x) - ru(t,x). \tag{3.5}$$

When $Q = (L,U)$ with $0 < L < U$, $f \equiv 0$ and $g \equiv 0$, u is the price of the double barrier knock-out option with payoff $h(S_T)$ and (3.1) becomes

$$\mathrm{D}_t u(t,x) + \mathrm{L}u(t,x) = 0 \qquad \text{for } (t,x)\in[0,T)\times(L,U) \tag{3.6a}$$

$$u(t,L) = u(t,U) = 0 \qquad \text{for } t\in[0,T) \tag{3.6b}$$

$$u(T,x) = h(x) \qquad \text{for } x\in[L,U] \tag{3.6c}$$

Note that if h is continuous on $[L,U]$ with $h(L) = h(U) = 0$, then all the assumptions of Theorem 3.1 hold, hence (3.6) has a unique solution and Theorem 3.2 reduces to the following result.

Theorem 3.3 *Let* L *be given as in* (3.5) *and let h be any continuous function on* $[L,U]$ *such that* $h(L) = h(U) = 0$. *Let* $\tau^{t,x} = \inf\{s \geq t : S_s^{t,x} \notin (L,U)\}$ *be the first exit time of the process* $S^{t,x}$ *from* (L,U). *Then*

$$u(t,x) = e^{-r(T-t)}\mathbf{E}\big[h(S_T^{t,x})\mathbb{1}_{\tau^{t,x}>T}\big]$$

is the unique solution to (3.6).

If $h(x) = (x-K)^+$ with $K \in (L,U)$ then the continuity condition (3.2) of Theorem 3.1 fails due to $h(U) = U - K \neq 0$. It can be shown that the result still holds in this case, however it has not been possible to find a reference, since the consulted texts on financial mathematics seems to neglect such details. For an intuitive explanation of why the result must still hold, one should realize that the underlying process (1.1) will only reach the problematic point (T,U) with probability zero.

Proving this is outside the scope of this paper, although it will be shown, at least experimentally, that the result of the finite differences scheme presented below converges to the solution of the partial differential equation. Alternatively, consider the case when h is replaced by the function

$$h^\varepsilon(x) := \begin{cases} \frac{U-x}{\varepsilon}(U-\varepsilon-K) & \text{for } x \in [U-\varepsilon, U] \\ (x-K)^+ & \text{else} \end{cases}$$

for some $\varepsilon > 0$. Then clearly all conditions of Theorem 3.1 are satisfied, and by decreasing ε the solution u^ε to the partial differential equation (3.6) can approximate the price u arbitrarily well, since $\mathbf{P}(S_T \in [U-\varepsilon, U]) \to 0$ as $\varepsilon \to 0$. Hence, since u^ε can be approximated arbitrarily well using finite differences, so can the price u.

Note that when L is given as in (3.5) the partial differential equation (3.6) might degenerate if the lower boundary L becomes too small, due to the x and x^2 coefficients on D_x and D_{xx}, respectively. In practice, this will in particular cause trouble if these coefficients fall below machine precision. To account for this, one might apply the log transform which will give rise to a new and somewhat nicer partial differential equation with constant coefficients: Let u be the unique solution to (3.6) and define $v(t,y) := u(t,\mathrm{e}^y)$, or equivalently $u(t,x) = v(t,\log x)$ for $x > 0$. Then clearly

$$\begin{aligned} \mathrm{D}_t u(t,x) &= \mathrm{D}_t v(t,x), \\ \mathrm{D}_x u(t,x) &= \frac{1}{x}\mathrm{D}_y v(t,\log x), \\ \mathrm{D}_{xx} u(t,x) &= -\frac{1}{x^2}\mathrm{D}_y v(t,\log x) + \frac{1}{x^2}\mathrm{D}_{yy} v(t,\log x) \end{aligned}$$

and so v must solve the partial differential equation

$$\begin{aligned} \mathrm{D}_t v(t,y) + \tfrac{1}{2}\sigma^2\mathrm{D}_{yy}v(t,y) + \theta\mathrm{D}_y v(t,y) - r\,v(t,y) &= 0 && \text{for } (t,y) \in [0,T) \times (\log L, \log U) \\ v(t,\log L) = v(t,\log U) &= 0 && \text{for } t \in [0,T) \\ v(T,y) &= h(\mathrm{e}^y) && \text{for } y \in (\log L, \log U) \end{aligned} \tag{3.7}$$

where $\theta = r - \frac{1}{2}\sigma^2$. Both of these methods have been implemented for comparison. The code can be found in Appendix A.

3.1 The single barrier case

In this section the case of a single barrier option will be considered. Denote by $u(t,x)$ the price of a up-and-out barrier option on S with barrier B and payoff $h(x) = (x-K)^+$ for some $K < B$, where $S_t = x$. That is

$$u(t,x) = \mathrm{e}^{-rT}\mathbf{E}\big[h(S_T^{t,x})\mathbb{1}_{\tau^{t,x}>T}\big] \quad \text{where} \quad \tau^{t,x} := \inf\{s \geq t : S_s^{t,x} \geq B\}. \tag{3.8}$$

Then by pde theory, and by Shreve (2004), u is the unique solution to the Black-Scholes partial differential equation

$$\begin{aligned} \mathrm{D}_t u(t,x) + \mathrm{L}u(t,x) &= 0 && \text{for } (t,x) \in [0,T) \times (0,B) \\ u(t,0) = u(t,B) &= 0 && \text{for } t \in [0,T) \\ u(T,x) &= h(x) && \text{for } x \in [0,B] \end{aligned} \tag{3.9}$$

where L is given as in (3.5). Note that the lower boundary being 0 makes sense, since if the process S is started at 0 it will remain 0, and $h(0) = 0$.

As an alternative, consider the log transformed partial differential equation. Naturally, one can not apply the log transform to (3.9), due to the lower boundary being 0. If one were to naïvely apply the log transform, the domain of the resulting partial differential equation would be unbounded. In practice the domain would then need to be truncated. Consider therefore instead directly the truncated partial differential equation

$$\begin{aligned} \mathrm{D}_t v_R(t,y) + \tfrac{1}{2}\sigma^2 \mathrm{D}_{yy} v_R(t,y) + \theta \mathrm{D}_y v_R(t,y) - r v_R(t,y) &= 0 && \text{for } (t,y) \in [0,T) \times (-R, \log B) \\ v_R(t,-R) = v_R(t,\log B) &= 0 && \text{for } t \in [0,T) \\ v_R(T,y) &= h(\mathrm{e}^y) && \text{for } y \in (-R, \log B) \end{aligned}$$

with $R > 0$. This corresponds to (3.7) with $U = B$ and $L = \mathrm{e}^{-R}$ and hence $v_R(t,\log x)$ is the price of a double barrier knock-out option with these barriers;

$$v_R(t,\log x) = \mathrm{e}^{-r(T-t)} \mathbf{E}\big[h(S_T^{t,x}) \mathbb{1}_{\tau_R^{t,x} > T}\big] \quad \text{where} \quad \tau_R^{t,x} := \inf\{s \geq t : S_s^{t,x} \notin (\mathrm{e}^{-R}, B)\}. \tag{3.10}$$

Truncating the domain naturally introduces an extra source of error, which depends on R. The question is then how big this truncation error is. To estimate this, let first $\rho_R^{t,x} := \inf\{s \geq t : \log x + (r - \frac{1}{2}\sigma^2)(s-t) + \sigma W_{(s-t)} \leq -R\}$ so that $\tau_R^{t,x} = \tau^{t,x} \wedge \rho_R^{t,x}$ and in particular

$$\mathbb{1}_{\tau_R^{t,x} > T} = \mathbb{1}_{\tau^{t,x} > T} \mathbb{1}_{\rho_R^{t,x} > T}.$$

In the following it will be assumed for simplicity that $t = 0$, i.e. that the price of interest is $u(0, S_0)$. To avoid clutter the superscripts will be dropped. Then by (3.8), (3.10) and the Cauchy-Schwartz inequality the error can be bounded from above by

$$\begin{aligned} \varepsilon_R &= |v_R(0,\log S_0) - u(0,S_0)| \leq \mathrm{e}^{-rT} \mathbf{E}\Big[\big|h(S_T)\mathbb{1}_{\tau > T}(\mathbb{1}_{\rho_R > T} - 1)\big|\Big] \\ &= \mathrm{e}^{-rT} \mathbf{E}\Big[\big|h(S_T)\mathbb{1}_{\tau > T}\big| \mathbb{1}_{\rho_R \leq T}\Big] \leq \mathrm{e}^{-rT} \sqrt{\mathbf{E}\big[h^2(S_T)\mathbb{1}_{\tau > T}\big]} \sqrt{\mathbf{P}(\rho_R \leq T)} \\ &\leq C_1 \sqrt{\mathbf{P}(\rho_R \leq T)}, \end{aligned}$$

for some constant C_1, where the last inequality follows since h is bounded on (e^{-R}, B). Now let $\alpha, \beta \in (-R, \log B)$ with $\alpha < \beta$, and assume that $\log S_0 \in (\alpha, \beta)$. Then there is a constant $v \in (0,1)$, independent of S_0 such that $|\log S_0| \leq vR$ and hence $-|R + \log S_0| \geq -R(v+1)$. Now one can calculate, see for instance Gyöngy (2015) or Krylov (2002),

$$\mathbf{P}(\rho_R \leq T) = \mathbf{P}\left(\inf_{t\in[0,T]} \left\{\left(\frac{r}{\sigma} - \frac{\sigma}{2}\right)t + W_t\right\} \leq \frac{-(R + \log S_0)}{\sigma}\right) \leq C_2 \mathrm{e}^{-C_3(R+\log S_0)^2/\sigma^2} \leq C_2 \mathrm{e}^{-C_4 R^2},$$

where C_2 and C_3 are constants depending only on $|r - \frac{1}{2}\sigma^2|$, σ^2 and T, and $C_4 = C_3(v+1)^2/\sigma^2$. Combining these calculations, one gets the upper bound on the truncation error $\varepsilon_R \leq C_1 \sqrt{C_2} \mathrm{e}^{-\frac{1}{2}C_4 R^2}$.

3.2 Finite differences

Theorem 3.3 verifies that one can find the price of a double barrier option by solving the partial differential equation (3.6). Doing this explicitly is not generally possible. Rather, this section will focus on solving the partial differential equation numerically, by applying a finite differences scheme.

A finite differences scheme works by replacing differentials by finite differences. For instance, $\frac{\mathrm{d}}{\mathrm{d}x}\varphi(x)$ might be replaced by $(\varphi(x+h)-\varphi(x))/h$, where $h > 0$ is the grid size. This is a forward difference estimate and converges to $\frac{\mathrm{d}}{\mathrm{d}x}\varphi(x)$ as $h \to 0$. Instead of this one could have chosen the backward difference $(\varphi(x)-\varphi(x-h))/h$ or the centered difference $(\varphi(x+h)-\varphi(x-h))/(2h)$.

Let in the following M and N be positive integers, and let $h = (U-L)/M$ and $\tau = T/N$. Consider the following difference operators

$$\begin{aligned} \delta_{x,h}\varphi(t,x) &:= \frac{\varphi(t,x+h)-\varphi(t,x)}{h}, \quad & \delta_{x,-h}\varphi(t,x) &:= \frac{\varphi(t,x)-\varphi(t,x-h)}{h}, \\ \delta_{t,\tau}(t,x) &:= \frac{\varphi(t+\tau,x)-\varphi(t,x)}{\tau}, \quad & \delta_{t,-\tau}(t,x) &:= \frac{\varphi(t,x)-\varphi(t-\tau)}{\tau}. \end{aligned} \tag{3.11}$$

Here $\delta_{x,h}$ is a forward operator and $\delta_{x,-h}$ is a backward operator. From the perspective of the partial differential equation (3.6), $\delta_{t,\tau}$ is in fact also a backward operator and $\delta_{t,-\tau}$ is a forward operator. The first two of these may be used to estimate $\mathrm{D}_x\varphi(t,x)$, while the last two may be used for estimating $\mathrm{D}_t\varphi(t,x)$. When estimating $\mathrm{D}_{xx}\varphi(t,x)$ it is sensible to use a centered operator, combining the forward and backward operators;

$$\delta_{x,h}\delta_{x,-h}\varphi(t,x) = \frac{\varphi(t,x+h)-2\varphi(t,x)+\varphi(t,x-h)}{h^2}. \tag{3.12}$$

When one wishes to use finite differences on $b\frac{\mathrm{d}}{\mathrm{d}x}\varphi(x)$ for some constant b, the upwind scheme suggests using the forward operator for $b > 0$, and the backwards scheme for $b < 0$. This method is used in the implementation solving the log transformed partial differential equation (3.7). In the implementation solving (3.6) the following centered difference operator is used:

$$\frac{1}{2}(\delta_{x,h}+\delta_{x,-h})\varphi(t,x) = \frac{\varphi(t,x+h)-\varphi(t,x-h)}{2h}. \tag{3.13}$$

For estimating the time derivative, two operators were given in (3.11), both of which have their advantages and disadvantages. Suppose L^h is an approximation of L by some finite differences method, and let $t_i = i\tau$ for $i = 0,\dots,N$ and $x_j = L + jh$ for $j = 0,\dots,M$. For the sake of brevity let also $u_{i,j} = u(t_i,x_j)$. Then, to solve (3.6) numerically using finite differences, observe first that $u_{N,j} = h(x_j)$ for all $j = 0,\dots,M$ and $u_{i,0} = u_{i,M} = 0$ for all $i = 0,\dots,N-1$. Next, see that by replacing D_t by the forward operator $\delta_{t,-\tau}$, to estimate the solution u one must solve

$$\frac{u_{i+1,j}-u_{i,j}}{\tau} + \mathrm{L}^h u_{i+1,j} = 0 \tag{3.14}$$

for all $i = 0,\dots,N-1$ and $j = 1,\dots,M-1$, which can be done recursively in i using

$$u_{i,j} = u_{i+1,j} + \tau \mathrm{L}^h u_{i+1,j}.$$

This can be done very efficiently which gives this method, called explicit finite differences, a clear advantage. However, it can be shown that it places certain restrictions on the relationship between τ and h for the scheme to converge (Heath, 2001). For this reason, more complicated schemes are often used.

Another approach is called the implicit finite differences scheme, in which one uses the backward operator $\delta_{t,\tau}$. This scheme has the advantage of not putting any restrictions on τ and h, and guarantees convergence. However, instead of (3.14) one gets for $i = 0,\dots,N-1$ and $j = 1,\dots,M-1$

$$\frac{u_{i+1,j} - u_{i,j}}{\tau} + \mathrm{L}^h u_{i,j} = 0,$$

which leads to a system of $M-1$ equations which must be solved at each step i. Suppose for instance that D_x and D_{xx} are replaced by the centered operators in (3.13) and (3.12), respectively, then the equation to solve is

$$\frac{u_{i+1,j} - u_{i,j}}{\tau} + r x_j \frac{u_{i,j+1} - u_{i,j-1}}{2h} + \frac{1}{2}\sigma^2 x_j^2 \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} - r u_{i,j} = 0,$$

or equivalently

$$a_j u_{i,j-1} + b_j u_{i,j} + c_j u_{i,j+1} = u_{i+1,j}$$

where $a_1 = 0$, $c_{M-1} = 0$ and in all other cases

$$a_j = \frac{\tau}{2h} r x_j - \frac{1}{2}\frac{\tau}{h^2}\sigma^2 x_j^2, \quad b_j = 1 + \frac{\tau}{h^2}\sigma^2 x_j^2 + \tau r \quad \text{and} \quad c_j = -\frac{\tau}{2h} r x_j - \frac{1}{2}\frac{\tau}{h^2}\sigma^2 x_j^2.$$

In the implementation this is solved by stating the problem in its matrix form

$$\begin{bmatrix} b_1 & c_1 & & \\ a_2 & \ddots & & \\ & & \ddots & c_{M-2} \\ & & a_{M-1} & b_{M-1} \end{bmatrix} \begin{bmatrix} u_{i,1} \\ \vdots \\ \vdots \\ u_{i,M-1} \end{bmatrix} = \begin{bmatrix} u_{i+1,1} \\ \vdots \\ \vdots \\ u_{i+1,M-1} \end{bmatrix}$$

and using the MATLAB backslash-operator. To obtain higher precision using any finite differences scheme M and N must be increased, which in the implicit case gives more and larger systems of the type above to solve. This is far less efficient than the explicit scheme.

3.3 Remarks

The methods introduced in this chapter are fairly general in their nature. Due to Theorem 3.1 and Theorem 3.2 finite differences schemes can be applied to a very large class of models. Moreover, the same methods can be applied to cases where the barriers are not constant, but functions of time.

One case where this approach does not perform well is when the option is to be monitorred at discrete times only. In this case the barriers are discontinuous and the solution $u(t,x)$ jumps at the monitoring times. Since the schemes presented in this chapter assumes continuity of the solution, they may lead to unwanted features such as spikes in the approximation in this setting. A finite difference scheme allowing such jumps is proposed in Duffy (2013).

4

Sequential Monte Carlo

This chapter will introduce two estimators for each of the discretely monitored and the continuously monitored barrier options; a standard Monte Carlo estimator and a sequential Monte Carlo estimator. The sequential Monte Carlo estimator was introduced in Shevchenko and Del Moral (2014), in which the option prices are expressed in terms of so-called Feynman-Kac expectations.

4.1 Feynman-Kac expectations

Consider for simplicity first the discretely monitored double barrier knock-out option with barriers L and U, and suppose it is to be monitored at times $0 = t_0 < t_1 < \cdots < t_N = T$, where $N \in \mathbb{N}$ and $t_i := iT/N$. For the sake of ease, let $S_i = S_{t_i}$ and let furthermore $\tilde{\tau}$ be the first time t_i such that $S_i \notin (L, U)$, i.e. $\tilde{\tau} := \inf\{t_i : S_{t_i} \notin (L, U)\}$, and let h denote the payoff function. Then the price of the option is given by

$$d_{\text{ko}} = \mathrm{e}^{-rT}\mathbf{E}\big[h(S_T)\mathbb{1}_{\tilde{\tau}>t_N}\big] = \mathrm{e}^{-rT}\mathbf{E}\Big[h(S_T)\prod_{n=0}^{N-1}\mathbb{1}_{(L,U)}(S_{n+1})\Big]. \tag{4.1}$$

The price of the continuously monited double barrier option can be written in a similar form. To see this let for $s, t \in [0, T]$ with $s < t$ $f(S_s, S_t)$ be the probability of not hitting the barrier between times s and t, conditioned on S being within the barriers at the end-points; i.e.

$$f(S_s, S_t) := \mathbf{P}\big(S_q \in (L, U) \text{ for all } q \in (s, t) \,\big|\, S_s, S_t \in (L, U)\big). \tag{4.2}$$

Taking $s \in (0, T)$ and using the tower property and the definition of f,

$$\begin{aligned}
\mathbf{E}\big[h(S_T); S_t \in (L, U) \text{ for all } t \in (0, T]\big] &= \mathbf{E}\big[h(S_T); S_s, S_T \in (L, U), S_t \in (L, U) \text{ for all } t \in (0, s) \cup (s, T)\big] \\
&= \mathbf{E}\big[h(S_T)f(S_s, S_T); S_s, S_T \in (L, U), S_t \in (L, U) \text{ for all } t \in (0, s)\big] \\
&= \mathbf{E}\big[h(S_T)f(S_0, S_s)f(S_s, S_T); S_s, S_T \in (L, U)\big] \\
&= \mathbf{E}\big[h(S_T)\big(\mathbb{1}_{(L,U)}(S_s)f(S_0, S_s)\big)\big(\mathbb{1}_{(L,U)}(S_T)f(S_s, S_T)\big)\big],
\end{aligned}$$

where $\tau := \inf\{t \geq 0 : S_t \notin (L, U)\}$. Repeating this steps, one obtains the following formula for the price of the continuously monitored double barrier knock-out option

$$c_{\text{ko}} = \mathrm{e}^{-rT}\mathbf{E}\big[h(S_T)\mathbb{1}_{\tau>T}\big] = \mathrm{e}^{-rT}\mathbf{E}\Big[h(S_T)\prod_{n=0}^{N-1}\big(\mathbb{1}_{(L,U)}(S_{n+1})f(S_n, S_{n+1})\big)\Big]. \tag{4.3}$$

Let in the following $X_n \coloneqq (S_n, S_{n+1})$, which forms a Markov chain due to the Markov property of S (Øksendal, 2003). By defining $G_n(X_n) \coloneqq \mathbb{1}_{(L,U)}(S_{n+1}) f(S_n, S_{n+1})$ and $H(X_n) \coloneqq h(S_n)$, one can write (4.3) as a telescoping product

$$\begin{aligned} c_{\text{ko}} &= \mathrm{e}^{-rT}\mathbf{E}\Big[H(X_N)\prod_{n=0}^{N-1} G_n(X_n)\Big] \\ &= \mathrm{e}^{-rT}\frac{\mathbf{E}\big[H(X_N)\prod_{n=0}^{N-1} G_n(X_n)\big]}{\mathbf{E}\big[\prod_{n=0}^{N-1} G_n(X_n)\big]}\cdots\frac{\mathbf{E}\big[G_1(X_1)G_0(X_0)\big]}{\mathbf{E}\big[G_0(X_0)\big]}\mathbf{E}\big[G_0(X_0)\big] \\ &= \mathrm{e}^{-rT}\eta_N(H)\prod_{n=0}^{N-1}\eta_n(G_n) \end{aligned} \tag{4.4}$$

where $\eta_n(F) \coloneqq \mathbf{E}\big[F(X_n)\prod_{\ell=0}^{n-1} G_\ell(X_\ell)\big]/\mathbf{E}\big[\prod_{\ell=0}^{n-1} G_\ell(X_\ell)\big]$ under the convention $\prod_{\ell\in\emptyset} G_\ell(X_\ell) = 1$. In the setting of particle methods the expectation in (4.4) is commonly referred to as a Feynman-Kac expectation, the functions G_n are called potentials and the random variables X_n are called particles. Thus, introducing X_n does not only help to avoid clutter, but also makes it easier to connect the formulas to a more general theory. For an introduction to particle methods and their financial applications the reader is referred to Carmona et al. (2012).

Now define the random variables $\xi_n \coloneqq \big[\prod_{\ell=0}^{n-1} G_\ell(X_\ell)\big]/\mathbf{E}\big[\prod_{\ell=0}^{n-1} G_\ell(X_\ell)\big]$ and see that $\xi_n \geq 0$ and $\mathbf{E}\xi_n = 1$. Then one can define a series of measures $\mathbf{Q}_n$ by $\mathrm{d}\mathbf{Q}_n = \xi_n \mathrm{d}\mathbf{P}$ and so $\eta_n(F) = \mathbf{E}[F(X_n)\xi_n] = \mathbf{E}_{\mathbf{Q}_n}F(X_n)$. Hence, by sampling M values of X_n, $X_n^{(1)},\dots,X_n^{(M)}$, according to $\mathbf{Q}_n$, one gets the estimator $\frac{1}{M}\sum_{m=1}^{M} F(X_n^{(m)})$ of $\eta_n(F)$. This is essentially was is done in the algorithm introduced below. However, one can not sample these directly under $\mathbf{Q}_n$. Instead the particles $X_n^{(1)},\dots,X_n^{(M)}$ are sampled from an approximation to the density of X_n under $\mathbf{Q}_n$, using an importance sampling technique called sequential importance resampling. This technique plays an important role in some sequential Monte Carlo methods, such as the bootstrap filter (Doucet et al., 2001).

Remark. In the formulas for d_{ko} and c_{ko}, one could of course instead of the interval (L,U) have taken $(-\infty,B)$ or (B,∞), $B > 0$, getting formulas for up-and-out or down-and-out options, respectively. •

Remark. These formulas make it easy to deal with the case when the barriers, interest rate and volatility are only piece-wise constant, and when the time steps are not equal in size. However, this setting will not be discussed further in this paper. •

4.2 Monte Carlo estimators

This section will present the crude Monte Carlo estimator and the sequential Monte Carlo estimator for the option prices d_{ko} and c_{ko} given in (4.1) and (4.3). Both methods will rely on sampling paths of S, $\hat{S} = (\hat{S}_0,\dots,\hat{S}_N)$. This can be done either using

$$\hat{S}_n = \hat{S}_{n-1}\mathrm{e}^{(r-\frac{1}{2}\sigma^2)\Delta t+\sigma\Delta W_n} \tag{4.5}$$

where $\Delta t = T/N$ and $\Delta W_n = W_{t_{n+1}} - W_{t_n} \sim N(0,\Delta t)$, or by using a discritization scheme on the stochastic differential equation (1.1) such as the Euler-Maruyama scheme,

$$\hat{S}_n = \hat{S}_{n-1} + r\hat{S}_{n-1}\Delta t + \sigma\hat{S}_{n-1}\Delta W_n. \tag{4.6}$$

The Euler-Maruyama scheme can be applied to more general models than the one under consideration here, however, this comes at the cost of an extra error term. In the implementation (4.5) will be used instead, since in fact $\hat{S}_n \overset{d}{=} S_{t_n}^{t_{n-1},\hat{S}_{n-1}}$ where $\overset{d}{=}$ means equal in distribution.

Figure 4.1: Number of barrier hits out of 200 sampled trajectories of the process S, with $S_0 = 50$, $r = 0.05$, $\sigma = 0.2$ over a period of length $T = 1$, in four different cases. The barriers are $50 \pm B$ for $B = 5, 10, 15, 20$.

4.2.1 Crude Monte Carlo

The unbiased crude Monte Carlo estimators $\hat{d}_{\text{ko}}^{\text{MC}}$ and $\hat{c}_{\text{ko}}^{\text{MC}}$ are obtained directly from (4.1) and (4.3) by sampling M paths $S^{(m)} = (S_1^{(m)}, \dots, S_N^{(m)})$, $m = 1, \dots, M$ and approximating the expectations by averages;

$$\hat{d}_{\text{ko}}^{\text{MC}} = \mathrm{e}^{-rT} \frac{1}{M} \sum_{m=1}^{M} \left(h\big(S_N^{(m)}\big) \prod_{n=0}^{N-1} \mathbb{1}_{(L,U)}\big(S_{n+1}^{(m)}\big) \right),$$

$$\hat{c}_{\text{ko}}^{\text{MC}} = \mathrm{e}^{-rT} \frac{1}{M} \sum_{m=1}^{M} \left(h\big(S_N^{(m)}\big) \prod_{n=0}^{N-1} \Big(\mathbb{1}_{(L,U)}\big(S_{n+1}^{(m)}\big) f\big(S_n^{(m)}, S_{n+1}^{(m)}\big) \Big) \right).$$

These estimators are very simple to implement. However, it is easy to realize that in order to get a good estimate, one will need to take a very high number of sample paths. This is due to the fact that if a barrier is breached for a sampled trajectory, the payoff will be zero for that trajectory. As can be seen in Figure 4.1, the number of trajectories for which the payoff is zero increases as the barriers get closer to each other.

When designing a Monte Carlo estimator it is not uncommon to want to change the measure, in order to get more samples in a certain area, e.g. for estimating $\mathbf{E}(S_T - K)^+$ when $K \gg S_0$. This technique is known as importance sampling, and it is in fact a variant of this method called sequential importance resampling which will be the key step in the sequential Monte Carlo algorithm.

4.2.2 Sequential Monte Carlo

In this section the sequential Monte Carlo algorithm developed in Shevchenko and Del Moral (2014) and the resulting estimators are presented. It will be useful to consider particles $X_n^{(m)}$ with $X_0^{(m)} = (S_0^{(m)}, S_1^{(m)})$, which can be sampled using (4.5). As mentioned above, the crucial step in the algorithm is the sequential importance resampling step, which itself consists of two steps: the acceptance-rejection step, and the resampling step.

In the acceptance-rejection step, each particle $X_n^{(m)}$ is either accepted or rejected, with probability determined by $G_n(X_n^{(m)})$. In particular, a particle $X_n^{(m)}$ will be accepted with probability $G_n(X_n^{(m)})$, and rejected with probability $1 - G_n(X_n^{(m)})$. This means that particles $X_n^{(m)} = (S_n^{(m)}, S_{n+1}^{(m)})$ with a high probability of having crossed the boundary over in the period t_n to t_{n+1} are more likely to be rejected. Note that if $S_{n+1}^{(m)} \notin (L, U)$ then $G_n(X_n^{(m)}) = 0$, and so the particle will be rejected.

In the resampling step the rejected particles are resampled from the discrete distribution with density function $\sum_{m=1}^{M} w_n^{(m)} \delta_{X_n^{(m)}}$ where the weights $w_n^{(m)}$ are given by

$$w_n^{(m)} = \frac{G_n(X_n^{(m)})}{\sum_{k=1}^{M} G_n(X_n^{(k)})}.$$

These newly sampled particles will be denoted by $\hat{X}_n^{(m)} = (\hat{S}_n^{(m)}, \hat{S}_{n+1}^{(m)})$ with $\hat{X}_n^{(m)} = X_n^{(m)}$ when $X_n^{(m)}$ was not rejected. Note again that if $S_{n+1}^{(m)} \notin (L, U)$, then $X_n^{(m)}$ can not be picked from the discrete distribution, to replace some other particle.

Following these steps, $S_{n+2}^{(m)}$ is sampled using (4.5) and one takes $X_{n+1}^{(m)} = (\hat{S}_{n+1}^{(m)}, S_{n+2}^{(m)})$. The sequential importance resampling is illustrated in Figure 4.2 and the full sequential Monte Carlo algorithm

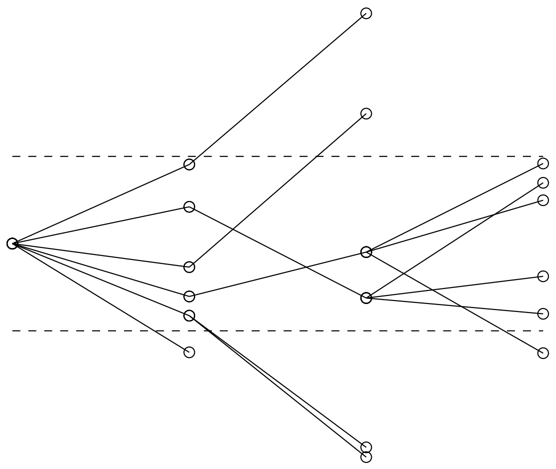
Figure 4.2: The evolution of six particles showing the sequential importance resampling step in action.

is presented in Algorithm 4.1. By taking $\tilde{G}_n(X_n) = \mathbb{1}_{(L,U)}(S_{n+1})$ (4.1) can be written as a Feynman-Kac expectation

$$d_{\text{ko}} = \mathrm{e}^{-rT}\mathbf{E}\Big[H(X_N)\prod_{n=0}^{N-1}\tilde{G}_n(X_n)\Big]$$

and the estimators $\hat{d}_{\text{ko}}^{\text{SMC}}$ and $\hat{c}_{\text{ko}}^{\text{SMC}}$ are given by

$$\begin{aligned}
\hat{d}_{\text{ko}}^{\text{SMC}} &= \mathrm{e}^{-rT}\prod_{n=0}^{N-1}\left(\frac{1}{M}\sum_{m=1}^{M}\tilde{G}_n(X_n^{(m)})\right)\frac{1}{M}\sum_{m=1}^{M}H(X_N^{(m)}) \\
\hat{c}_{\text{ko}}^{\text{SMC}} &= \mathrm{e}^{-rT}\prod_{n=0}^{N-1}\left(\frac{1}{M}\sum_{m=1}^{M}G_n(X_n^{(m)})\right)\frac{1}{M}\sum_{m=1}^{M}H(X_N^{(m)}).
\end{aligned}$$

Note that line 10 in Algorithm 4.1 implicitly assumes that not all of the simulated paths cross the barriers at the same time. In this case, all of the w_m's will be zero, in which case it is reasonable to take the estimator to be zero as well.

In line 13 one needs to sample from the probability distribution mentioned above. This can be done easily by simulating a random variable $Y \sim U[0,1]$ and taking $\hat{X}_n^{(m)} = X_n^{(k)}$, where k is the smallest number $1 \le k \le M$ such that $Y \le \sum_{\ell=1}^{k} w_\ell$. Alternatively (Shevchenko and Del Moral, 2014) proposes a method for simulating several of such random variables efficiently. This is the method used in the code provided in Appendix A.

Remark. Algorithm 4.1 is generic enough to contain both types of barrier options, in both the discretely and the continuously monitored cases. For example, in the case of a discretely monitored up-and-out call option with barrier B, the algorithm can be adapted by taking $L = -\infty$, $U = B$ and $G_n = \tilde{G}_n$. The algorithm can even be adapted to the case of a European-type option, by taking $G_n \equiv 1$. In this case however, the algorithm just reduces to the standard Monte Carlo method. •

Algorithm 4.1 Sequential Monte Carlo algorithm

```
1:  function SMC($M, N$)
2:      $\hat{S}_0^{(m)} \leftarrow S_0$ for $m \leftarrow 1, \dots, M$.
3:      for $n \leftarrow 0, \dots, N-1$ do
4:          for $m \leftarrow 1, \dots, M$ do
5:              Sample $S_{n+1}^{(m)}$ using (4.5)
6:              $X_n^{(m)} \leftarrow (\hat{S}_n^{(m)}, S_{n+1}^{(m)})$
7:              Sample $U_n^{(m)} \sim U[0,1]$
8:              $G_n^{(m)} \leftarrow G_n(X_n^{(m)})$
9:          end for
10:         $w_n^{(m)} \leftarrow \frac{G_n^{(m)}}{\sum_{k=1}^M G_n^{(k)}}$ for $m \leftarrow 1, \dots, M$
11:         for $m \leftarrow 1, \dots, M$ do
12:             if $G_n^{(m)} < U_n^{(m)}$ then
13:                 Sample $\hat{X}_n^{(m)}$ from the discrete distribution with density $\sum_{k=1}^M w_n^{(k)} \delta_{X_n^{(k)}}$
14:             else
15:                 $\hat{X}_n^{(m)} \leftarrow X_n^{(m)}$
16:             end if
17:         end for
18:         $E_n \leftarrow \frac{1}{M} \sum_{m=1}^M G_n^{(m)}$
19:     end for
20:     $\hat{h} \leftarrow \frac{1}{M} \sum_{m=1}^M H(\hat{X}_N^{(m)})$
21:     return $e^{-rT} \hat{h} \prod_{n=1}^N E_n$
22: end function
```

Unbiasedness of the SMC estimator

The goal of this section is to give a sketch of the proof that the sequential Monte Carlo estimator is unbiased. First, define for $n \geq 0$ and $M \geq 1$ the estimators

$$\eta_n^M(H) := \frac{1}{M} \sum_{m=1}^M H(X_n^{(m)}), \quad \gamma_n^M(1) = \prod_{\ell=0}^{n-1} \eta_\ell^M(G_\ell) \quad \text{and} \quad \gamma_n^M(H) = \gamma_n^M(1) \eta_n^M(H).$$

Hence one can write $\hat{c} = e^{-rT} \gamma_N^M(H)$, and so the goal is to show that $\mathbf{E}\gamma_N^M(H) = \gamma_N(H)$ where

$$\gamma_n(H) = \mathbf{E}\Big[H(X_n) \prod_{\ell=0}^{n-1} G_\ell(X_\ell)\Big],$$

which will be shown by an induction argument. By convention

$$\gamma_0^M(H) = \gamma_0^M(1)\eta_0^M(H) = \eta_0^M(H)$$

and so $\mathbf{E}\gamma_0^M(H) = \mathbf{E}\eta_0^M(H)$. Hurthermore $\eta_0(H) = \mathbf{E}H(X_0)$ and $\mathbf{E}\eta_0^M(H) = \mathbf{E}H(X_0)$, hence

$$\mathbf{E}\gamma_0^M(H) = \mathbf{E}\eta_0^M(H) = \eta_0(H) = \gamma_0(H).$$

Assume now that $\mathbf{E}\gamma_{n-1}^M(H) = \gamma_{n-1}(H)$ for some $n \geq 1$. Notice that due to the resampling step in the algorithm,

$$\begin{aligned} \mathbf{E}\big(\eta_n^M(H) \mid (X_0^{(m)}, \dots, X_{n-1}^{(m)})_{m=1,\dots,M}\big) &= \mathbf{E}\big(H(X_n) \mid (X_0^{(m)}, \dots, X_{n-1}^{(m)})_{m=1,\dots,M}\big) \\ &= \sum_{m=1}^M w_n^{(m)} \mathbf{E}\big(H(X_n) \mid X_{n-1} = X_{n-1}^{(m)}\big) \end{aligned}$$

where the weights are given by

$$w_n^{(m)} := \frac{G_{n-1}(X_{n-1}^{(m)})}{\sum_{k=1}^M G_{n-1}(X_{n-1}^{(k)})}.$$

These will be 0 in case of a barrier breach. Observing that $w_n^{(m)} = \frac{1}{M}\frac{1}{\eta_{n-1}(G_{n-1})}G_{n-1}(X_{n-1}^{(m)})$ and since $\mathbf{E}\big(\gamma_n^M(1) \mid (X_0^{(m)},\dots,X_{n-1}^{(m)})_{m=1,\dots,M}\big) = \gamma_n^M(1)$,

$$\begin{aligned}
\mathbf{E}\big(\gamma_n^M(H) \mid (X_0^{(m)},\dots,X_{n-1}^{(m)})_{m=1,\dots,M}\big) &= \gamma_n^M(1)\sum_{m=1}^M w_n^{(m)}\mathbf{E}\big(H(X_n) \mid X_{n-1} = X_{n-1}^{(m)}\big) \\
&= \left(\prod_{\ell=0}^{n-1}\eta_\ell^M(G_\ell)\right)\frac{1}{M}\frac{1}{\eta_{n-1}(G_{n-1})}\sum_{m=1}^M G_{n-1}\big(X_{n-1}^{(m)}\big)\mathbf{E}\big(H(X_n) \mid X_{n-1} = X_{n-1}^{(m)}\big) \\
&= \left(\prod_{\ell=0}^{n-2}\eta_\ell^M(G_\ell)\right)\frac{1}{M}\sum_{m=1}^M G_{n-1}\big(X_{n-1}^{(m)}\big)\mathbf{E}\big(H(X_n) \mid X_{n-1} = X_{n-1}^{(m)}\big) \\
&= \gamma_{n-1}^M(1)\eta_{n-1}(Q_n(H)) = \gamma_{n-1}^M(Q_n(H)),
\end{aligned}$$

where $Q_n(H)(x) = G_{n-1}(x)\mathbf{E}\big(H(X_n) \mid X_{n-1} = x\big)$. From the induction hypothesis it follows that

$$\mathbf{E}\gamma_n^M(H) = \mathbf{E}\big[\mathbf{E}\big(\gamma_n^M(H) \mid (X_0^{(m)},\dots,X_{n-1}^{(m)})_{m=1,\dots,M}\big)\big] = \mathbf{E}\gamma_{n-1}^M(Q_n(H)) = \gamma_{n-1}(Q_n(H)).$$

Now all that is left to complete the proof is to show that $\gamma_n(H) = \gamma_{n-1}(Q_n(H))$, which follows quite easily;

$$\begin{aligned}
\gamma_n(H) &= \mathbf{E}\left[H(X_n)\prod_{\ell=0}^{n-1}G_\ell(X_\ell)\right] = \mathbf{E}\left[\mathbf{E}\left(H(X_n)\prod_{\ell=0}^{n-1}G_\ell(X_\ell) \,\middle|\, (X_0,\dots,X_{n-1})\right)\right] \\
&= \mathbf{E}\left[\mathbf{E}\big(H(X_n) \mid (X_0,\dots,X_{n-1})\big)\prod_{\ell=0}^{n-1}G_\ell(X_\ell)\right] \\
&= \mathbf{E}\left[G_{n-1}(X_{n-1})\mathbf{E}\big(H(X_n) \mid X_{n-1}\big)\prod_{\ell=0}^{n-2}G_\ell(X_\ell)\right] \\
&= \mathbf{E}\left[Q_n(H)(X_{n-1})\prod_{\ell=0}^{n-2}G_\ell(X_\ell)\right] = \gamma_{n-1}(Q_n(H)).
\end{aligned}$$

4.3 Boundary crossing probabilities

In order to be able to use Algorithm 4.1 it is of course necessary to have a way of calculating the probability of crossing the boundary between to points in time, as stated in (4.2). So consider s and t with $0 \le s < t \le T$ and assume that $S_s = x_1$ and $S_t = x_2$ are both within the boundaries. In the case with only a single (upper or lower) barrier B, this is given by

$$f(x_1,x_2) = 1 - \exp\left(-2\frac{\log(x_1/B)\log(x_2/B)}{\sigma^2(t-s)}\right).$$

When there are two boundaries L and U, the expression becomes more complicated;

$$f(x_1,x_2) = 1 - \sum_{k=1}^{\infty}\big(R(y,qk-a) + R(y,-qk+b) - R(y,qk) - R(y,-qk)\big)$$

where $y = \log(x_2/x_1)$, $q = \log(U/L)$, $a = \log(x_1/L)$, $b = \log(U/x_1)$ and $R(y,z) = \exp\left(-2\frac{z(z-y)}{\sigma^2(t-s)}\right)$. These expressions can be found in Shevchenko and Del Moral (2014).

4.4 Remarks

The method described in this chapter is very flexible and can be applied to a great many cases. For instance, by dividing the interval $[0, T]$ into several pieces, the interest rate, volatility and boundaries can be taken to be only piecewise constant. By using the boundary crossing probabilities given in Kunitomo and Ikeda (1992), it is even possible to price options with curved boundaries.

Some of this flexibility is provided by the restatement of the price in terms of Feynman-Kac expectations. In this way, it is even possible to have stochastic interest rate, by including the interest rate in the potential G. For more information see Carmona et al. (2012).

The sequential Monte Carlo method is especially efficient for discretely monitored options, when the boundary crossing probabilities does not have to be calculated at each step, which adds an extra layer of complexity. Under other models than the current, the boundary crossing probabilities might not have analytical expressions. However, since they are not needed in the discrete case, the method is easily adapted to a broader range of models, in which case one can sample paths using for instance the Euler-Maruyama scheme (4.6).

5
Numerical results

In this chapter some numerical results obtained from the implementations in Appendix A are presented. The underlying is in all cases assumed to follow (1.1) with $\sigma = 0.2$ and $r = 0.05$. When the option considered has only one barrier, this is taken to be $B = 60$. When there are two barriers, these are $L = 40$ and $U = 60$. In both cases the strike price is $K = 50$ and the time to maturity $T = 1$.

5.1 Finite differences

Figure 5.1 shows the relative errors of a number of single barrier option prices, evaluated using finite differences on the two different partial differential equations given in Chapter 3. These are plotted against computation time, measured in seconds. Each of the points corresponds to a combination of M and N, both taking values in $\{50, 100, 500, 1500, 2000, 3000\}$. Here M is the number of steps in the space dimension, and N is the number of steps in the time dimension. Points which are connected corresponds to a fixed value of N, and varying values of M. The plots show quite clearly the trade off between accuracy and efficiency.

Furthermore, one can see that in most cases, the biggest gain in accuracy comes by increasing M rather than N. This is not true for the Black-Scholes equation when the spot price is is close to 0, where the biggest gain comes from increasing M, i.e. refining the space dimension, rather than the time dimension. Notice that in the case where $S_0 = 10$, the actual price is of the order of 10^{-15}, which might explain the very high relative errors for this case, giving for instance round-off errors a much greater impact.

In this example, the finite differences method on the Black-Scholes equation generally performs less than a factor of ten better than the log transformed equation, in terms of relative error. It is also worth noting that when $S_0 = 55$, the log transformed equation does not actually give the smallest errors at the largest computation times.

Similarly to Figure 5.1, Figure 5.2 shows the relative errors of some double barrier options, evaluated using finite differences. Again, each point corresponds to a pair of values M, N, each of which takes values in $\{50, 100, 500, 1500, 2000, 3000\}$, with M determining the space-discretization and N the time-discretization. Connected points corresponds to the same value of N and varying values of M.

In the case of the Black-Scholes equation, the errors generally decreases as a function of computation time. For the log transformed equation, however, there is no such trend. In particular, one can see that the errors do not decrease strictly as the computation time grows, and in fact, one can notice that for each N, the method generally performs better before reaching the maximum computation time. Furthermore, the solution of the log transformed equation is generally more accurate than the

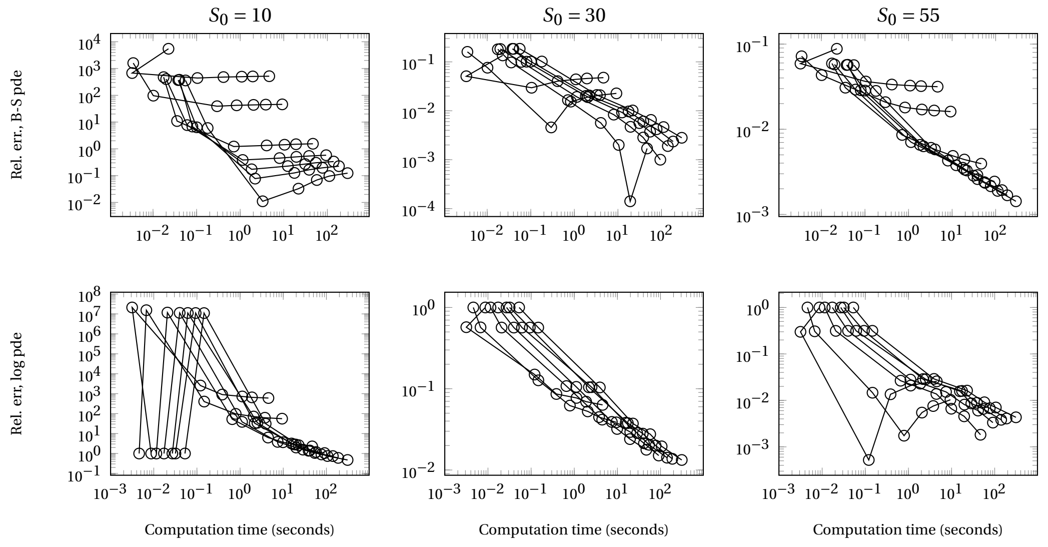

Figure 5.1: Relative errors of single barrier call option prices for three different spot prices.

solution of the Black-Scholes equation, up to around a factor of 10. This is contrary to the situation above.

These plots shows that both methods do converge to the actual value, in both cases. However, they also show that there is no silver bullet, even when considering just finite differences methods. The optimal method and how the paramters should be chosen very much depends on the problem at hand.

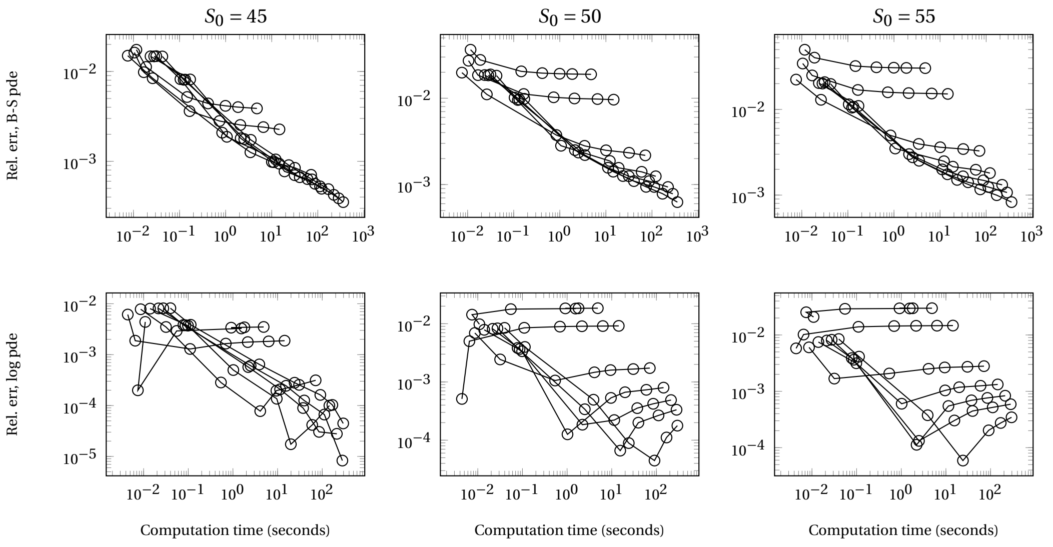

Figure 5.2: Relative errors of double barrier call option prices for three different spot prices.

Figures 5.3, 5.4, 5.5 and 5.6 more directly show the convergence of these schemes to the actual

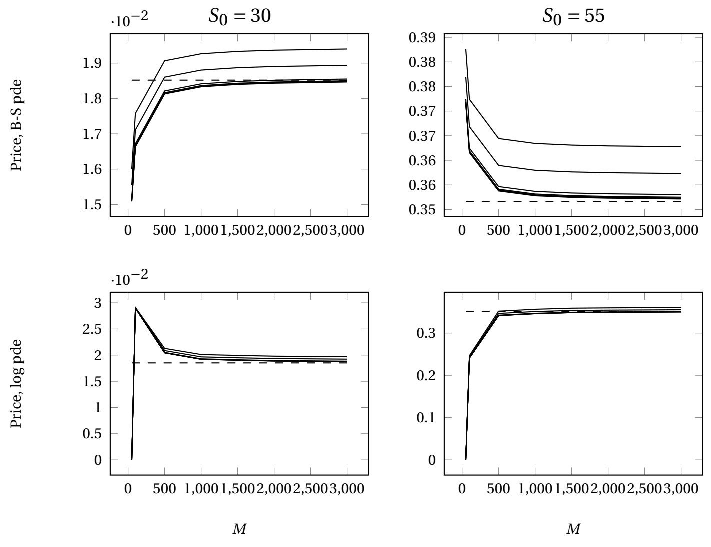

Figure 5.3: Single barrier prices calculated by finite differences for both partial differential equations. Each line corresponds to a fixed value of N. The dashed line is the actual price.

prices. By looking at these graphs, it will appear that the finite differences method applied to the log transformed Black-Scholes partial differential equation has faster convergence, than when applied to the classic Black-Scholes equation. To make this more precise, let $u_{M,N}(t,x)$ be a finite differences approximation to u at a point $(t,x) \in [0,T) \times (L,U)$ for some interval (L,U), where the space step-size is $(U-L)/M$ and the time step-size is T/N. If there are constants C_N and and q_N for a fixed M such that

$$|u_{M,N}(t,x) - u(t,x)| \leq C_N N^{-q_N} \text{ for all } N, \tag{5.1}$$

then $u_{M,N}$ has order of convergence q_N in N. Note that, despite the subscript, q_N and C_N might depend on M, but not N.

Assuming that the inequality in (5.1) holds with approximate equality, then by taking logarithms

$$\log|u_{M,N}(t,x) - u(t,x)| \approx \log C_N - q_N \log N \text{ for all } N.$$

Hence the order of convergence can be approximated by simple linear regression. Table 5.1 shows approximated orders of convergence in M and N for both single and double barrier options. The orders of convergence in N was estimated for $M = 3000$, and likewise the orders in M were estimated for $N = 3000$. The table

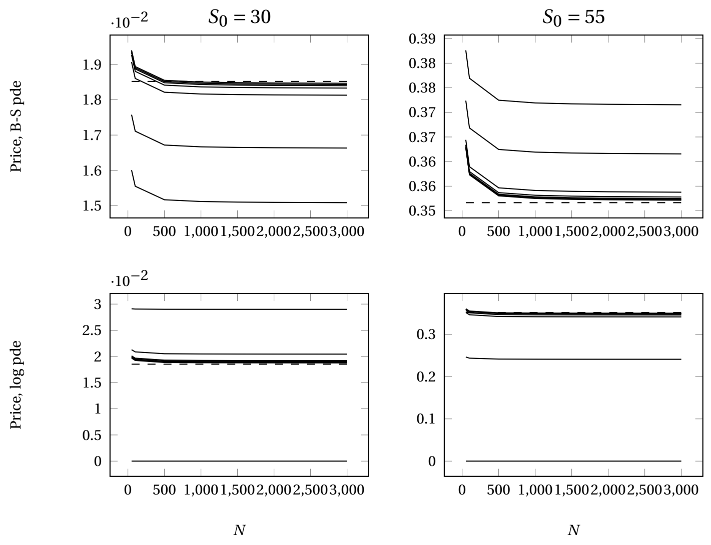

Figure 5.4: Single barrier prices calculated by finite differences for both partial differential equations. Each line corresponds to a fixed value of M. The dashed line is the actual price.

		Single barrier			Double barrier		
Convergence in	PDE	$S_0 = 10$	$S_0 = 30$	$S_0 = 55$	$S_0 = 45$	$S_0 = 50$	$S_0 = 55$
M	B-S	1.8978	1.0165	0.9096	0.9238	0.8397	0.7938
N	B-S	1.9742	0.8039	0.7733	0.6045	0.8475	0.8922
M	log	2.0587	1.0851	1.3171	1.2229	1.1941	0.9602
N	log	1.7194	0.3836	0.4196	1.3521	1.1087	1.0743

Table 5.1: Orders of convergence of the finite difference-solutions to the Black-Scholes (B-S) equation and the log transformed Black-Scholes equation.

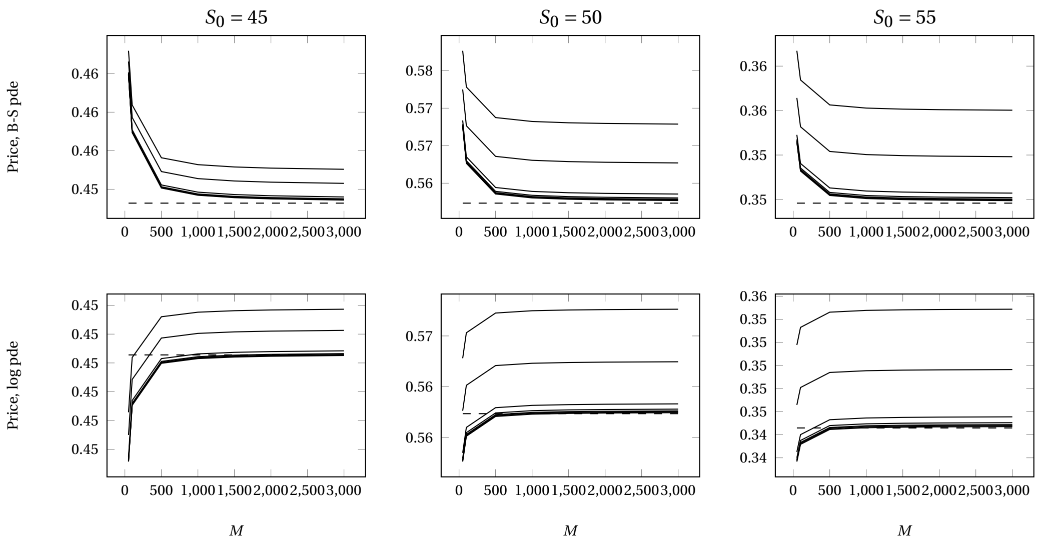

Figure 5.5: Double barrier prices calculated by finite differences for both partial differential equations. Each line corresponds to a fixed value of N. The dashed line is the actual price.

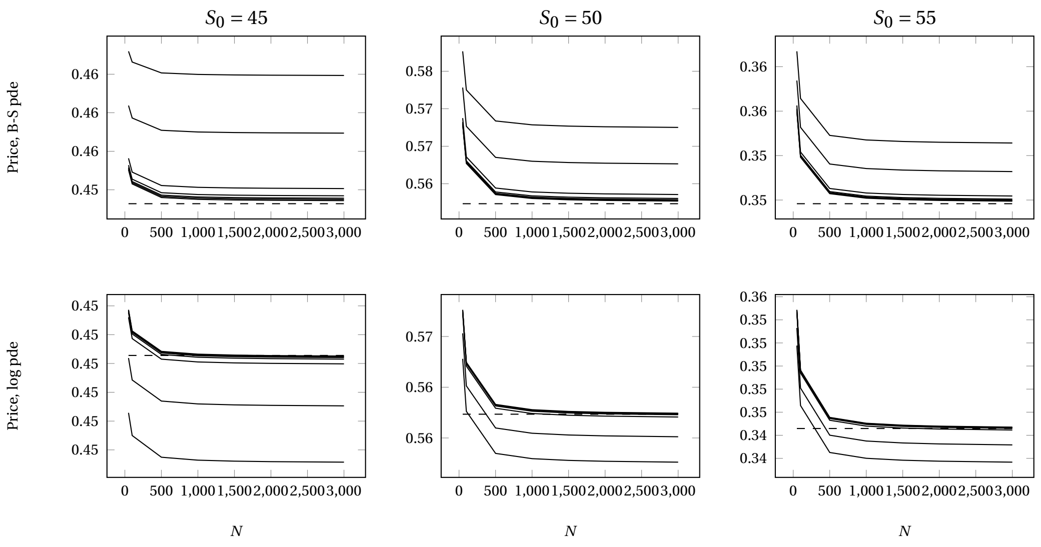

Figure 5.6: Double barrier prices calculated by finite differences for both partial differential equations. Each line corresponds to a fixed value of M. The dashed line is the actual price.

A

Code

In this appendix, all the implemented functions are listed, in order to make this document as self-contained as possible. These are also available at `https://github.com/torbonde/dissertation`.

Should there be any interest in running any of the functions, this code can be obtained by running the following command in a terminal

```
git clone https://github.com/torbonde/dissertation.git
```

Note that this assumes `git` is installed. Alternatively, one can download and extract the zip-file at `https://github.com/torbonde/dissertation/archive/master.zip`. Please also note that the file `bn_uo_call.m` requires the `Faddeeva` package. This is not included but can be found at `http://ab-initio.mit.edu/Faddeeva`.

The code is divided in to two sections. Section A.1 contains six functions for pricing up-and-out call options, five functions for pricing knock-out call options and one function for pricing European call options, all based on the methods described in this paper. Section A.2 contains the helper functions, in which most of the actual work is done.

A.1 Pricing functions

uo_call.m

```
function c = uo_call(model, option)
% UO_CALL prices an up-and-out call option using the analytical formula.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           B : Barrier

if option.K >= option.B
    c = 0;
    return;
end

% Up−and−in price
lambda = model.r/model.sigma^2 + 1/2;
y1 = log(model.S0/option.B)/(model.sigma∗sqrt(option.T)) ...
    + lambda∗model.sigma∗sqrt(option.T);
y2 = log(option.B/model.S0)/(model.sigma∗sqrt(option.T)) ...
    + lambda∗model.sigma∗sqrt(option.T);
x = log(option.B^2/(model.S0∗option.K))/(model.sigma∗sqrt(option.T)) ...
    + lambda∗model.sigma∗sqrt(option.T);
ci = model.S0∗normcdf(y1) − option.K ...
        ∗exp(−model.r∗option.T)∗normcdf(y1 − model.sigma∗sqrt(option.T)) ...
    − model.S0∗(option.B/model.S0)^(2∗lambda)∗(normcdf(−x) − normcdf(−y2)) ...
    + option.K∗exp(−model.r∗option.T)∗(option.B/model.S0)^(2∗lambda−2) ...
        ∗(normcdf(−x+model.sigma∗sqrt(option.T)) ...
            − normcdf(−y2+model.sigma∗sqrt(option.T)));

% European call price
d1 = log(model.S0/option.K)/(model.sigma∗sqrt(option.T)) ...
    + lambda∗model.sigma∗sqrt(option.T);
d2 = d1 − model.sigma∗sqrt(option.T);
ce = model.S0∗normcdf(d1) − option.K∗exp(−model.r∗option.T)∗normcdf(d2);

% Up−and−out price from in−out parity
c = ce − ci;
```

ko_call.m

```
function c = ko_call(model, option, N)
% KO_CALL prices a knock−out call option using the analytical formula.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk−free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           L : Lower barrier
%           U : Upper barrier
%       N : Point at which the infinite sum is cut off.
c1 = 2∗model.r/model.sigma^2 + 1;
d1 = @(n) (log(model.S0∗option.U.^(2∗n)./(option.K∗option.L.^(2∗n))) ...
    + (model.r + model.sigma^2/2)∗option.T)/(model.sigma∗sqrt(option.T));
d2 = @(n) (log(model.S0∗option.U.^(2∗n−1)./(option.L.^(2∗n))) ...
    + (model.r + model.sigma^2/2)∗option.T)/(model.sigma∗sqrt(option.T));
d3 = @(n) (log(option.L.^(2∗n+2)./(option.K∗model.S0∗option.U.^(2∗n))) ...
    + (model.r + model.sigma^2/2)∗option.T)/(model.sigma∗sqrt(option.T));
```

```
d4 = @(n) (log(option.L.^(2*n+2)./(model.S0*option.U.^(2*n+1))) ....
    + (model.r + model.sigma^2/2)*option.T)/(model.sigma*sqrt(option.T));
p = @(n) model.S0*((option.U.^n./option.L.^n).^c1 ...
            .*(normcdf(d1(n)) - normcdf(d2(n))) ...
        - (option.L.^(n+1)./(option.U.^n*model.S0)).^c1 ...
            .*(normcdf(d3(n)) - normcdf(d4(n)))) ...
    - option.K*exp(-model.r*option.T)*((option.U.^n./option.L.^n).^(c1-2) ...
        .*(normcdf(d1(n) - model.sigma*sqrt(option.T)) ...
            - normcdf(d2(n) - model.sigma*sqrt(option.T))) ...
    - (option.L.^(n+1)./(option.U.^n*model.S0)).^(c1-2) ...
        .*(normcdf(d3(n) - model.sigma*sqrt(option.T)) ...
            - normcdf(d4(n) - model.sigma*sqrt(option.T))));
c = sum(p(-N:N));
```

bn_eu_call.m

```
function c = bn_eu_call(model, option)
% BN_EU_CALL prices a European call option using the Borovkov-Novikov method.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
phi = @(u) terminal_mgf(model, option, u);
c = exp(-model.r*option.T)*model.S0*bn_p1(option.K/model.S0, phi);
```

bn_uo_call.m

```
function c = bn_uo_call(model, option)
% BN_UO_CALL prices an up-and-out call option using the Borovkov-Novikov method.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           B : Barrier
phi = @(u) terminal_mgf(model, option, u);
psi = @(u, v) terminal_max_mgf(model, option, u, v);
B = log(option.U/model.S0);
c = exp(-model.r*option.T)*model.S0*bn_p2(option.K, B, phi, psi);
```

bs_pde_uo_call.m

```matlab
function c = bs_pde_uo_call(model, option, M, N)
% BS_PDE_UO_CALL prices an up-and-out call option using finite differences on
% the Black-Scholes PDE.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           B : Barrier
%       M : Number of pieces the domain will be divided into.
%       N : Number of pieces [0,T] will be divided into.
option.h = @(x) max(x-option.K,0);
option.U = option.B;
option.L = 0;
[c, ~] = pde_solve(model, option, M, N);
```

bs_pde_ko_call.m

```matlab
function c = bs_pde_ko_call(model, option, M, N)
% BS_PDE_KO_CALL prices a knock-out call option using finite differences on
% the Black-Scholes PDE.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           L : Lower barrier
%           U : Upper barrier
%       M : Number of pieces the domain will be divided into.
%       N : Number of pieces [0,T] will be divided into.
option.h = @(x) max(x-option.K,0);
[c, ~] = pde_solve(model, option, M, N);
```

log_pde_uo_call.m

```matlab
function c = log_pde_uo_call(model, option, M, N, R)
% LOG_PDE_UO_CALL prices an up-and-out call option using finite differences on
% the log transformed Black-Scholes PDE.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           B : Barrier
%       M : Number of pieces the domain will be divided into.
%       N : Number of pieces [0,T] will be divided into.
%       R : Truncation limit
opt.T = option.T;
opt.h = @(x) max(exp(x)-option.K,0);
opt.L = -R;
opt.U = log(option.B);

model.x0 = log(model.S0);
[c, ~] = log_pde_solve(model, opt, M, N);
```

log_pde_ko_call.m

```
function c = log_pde_ko_call(model, option, M, N)
% LOG_PDE_KO_CALL prices a knock-out call option using finite differences on
% the log transformed Black-Scholes PDE.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           L : Lower barrier
%           U : Upper barrier
%       M : Number of pieces the domain will be divided into.
%       N : Number of pieces [0,T] will be divided into.
opt.T = option.T;
opt.h = @(x) max(exp(x)-option.K,0);
opt.L = log(option.L);
opt.U = log(option.U);

model.x0 = log(model.S0);
[c, ~] = log_pde_solve(model, opt, M, N);
```

mc_uo_call.m

```
function c = mc_uo_call(model, option, M, N)
% MC_UO_CALL prices an up-and-out call option using crude Monte Carlo.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           B : Barrier
%       M : Number of paths
%       N : Number of time-steps
option.h = @(x) max(x-option.K,0);
option.U = option.B;
c = mc(model, option, M, N);
```

mc_ko_call.m

```
function c = mc_ko_call(model, option, M, N)
% MC_KO_CALL prices a knock-out call option using crude Monte Carlo.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           L : Lower barrier
%           U : Upper barrier
%       M : Number of paths
%       N : Number of time-steps
option.h = @(x) max(x-option.K,0);
c = mc(model, option, M, N);
```

smc_uo_call.m

```
function c = smc_uo_call(model, option, M, N)
% SMC_UO_CALL prices an up-and-out call option using sequential Monte Carlo.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           B : Barrier
%       M : Number of paths
%       N : Number of time-steps
option.h = @(x) max(x-option.K,0);
```

```matlab
option.U = option.B;
c = smc(model, option, M, N);
```

smc_ko_call.m

```matlab
function c = smc_ko_call(model, option, M, N)
% SMC_KO_CALL prices a knock-out call option using sequential Monte Carlo.
%
%   Required are:
%       model : struct with fields
%           sigma : Volatility
%           r     : Risk-free interest
%           S0    : Initial value
%       option : struct with fields
%           T : Time to maturity
%           K : Strike price
%           L : Lower barrier
%           U : Upper barrier
%       M : Number of paths
%       N : Number of time-steps
option.h = @(x) max(x-option.K, 0);
c = smc(model, option, M, N);
```

A.2 Helper functions

bn_g.m

```matlab
function g = bn_g(a, b, s)
% BN_P2 evaluates function g in the Borovkov-Novikov theorem.
%
%   g = BN_G(a, b, s) returns the value of the function g in the
%   Borovkov-Novokov theorem evaluated at s.
%       a : real number
%       b : real number
%       s : real number
g = exp(1i*s*b-a*b)./((a-1i*s).*(1+a-1i*s));
```

bn_p1.m

```matlab
function P1 = bn_p1(K, phi)
% BN_P1 evaluates the expectation P1 in the Borovkov-Novikov theorem.
%
%   P1 = BN_P2(K, phi) returns the value of the expectation
%       P1 = E[max(exp(X)-K,0)]
%   with
%       K   : real number
%       phi : mgf of X
a=9;
g = @(s) bn_g(a, log(K), s);
% `real` just to satisfy Matlab
P1 = real(1/(2*pi)*integral(@(s) phi(1+a-1i*s).*g(s), -Inf, Inf));
```

bn_p2.m

```
function P2 = bn_p2(K, B, phi, psi)
% BN_P2 evaluates the expectation P2 in the Borovkov-Novikov theorem.
%
%   P2 = BN_P2(K, B, phi, psi) returns the value of the expectation
%       P2 = E[max(exp(X)-K,0) ; Z <= B]
%   with
%       K   : real number
%       B   : real number
%       phi : mgf of X
%       psi : joint mgf of (X,Z)
a=9;
g = @(s) bn_g(a, log(K), s);
P1 = bn_p1(K, phi);
integrand = @(x, u, v) 1./(1i*u).*psi(1+a-1i*v,1i*u).*g(v).*exp(-1i*u*x);
W = @(x) 1/2 - 1/(4*pi^2*P1)...
    *integral2(@(u, v) integrand(x, u, v), -Inf, Inf, -Inf, Inf);
P2 = P1*W(B);
```

terminal_mgf.m

```
function phi = terminal_mgf(mod, opt, u)
% TERMINAL_MGF evaluates the mgf of the terminal of a Wiener process with drift.
%
%   phi = TERMINAL_MGF(mod, opt, u) gives the mgf of the terminal value of
%   a Wiener process with drift evaluated at u.
%
%   The struct mod must contain the fields
%       r      : Risk-free interest
%       sigma  : Volatility
%
%   The struct opt must contain the fields
%       T      : Time to maturity
phi = exp((mod.r - 1/2*mod.sigma^2)*u + 1/2*opt.T*mod.sigma^2*u.^2);
```

terminal_max_mgf.m

```
function psi = terminal_max_mgf(mod, opt, u, v)
% TERMINAL_MAX_MGF evaluates the joint mgf of the terminal and max value of
% a Wiener process with drift.
%
%   psi = TERMINAL_MAX_MGF(mod, opt, u, v) gives the joint mgf of the
%   terminal and maximum value of a Wiener process with drift evaluated at (u,v).
%
%   The struct mod must contain the fields
%       r      : Risk-free interest
%       sigma  : Volatility
%
%   The struct opt must contain the fields
%       T      : Time to maturity
normcdf_ = @(x) 0.5*(Faddeeva_erfc(-x/sqrt(2)));
psibar = @(u, v) 2*normcdf_(sqrt(opt.T)*(u+v)).*exp(1/2*opt.T*(u+v).^2).*(u+v)
    ./(2*u+v) ...
    + normcdf_(1/2*sqrt(opt.T)*v).*exp(1/8*opt.T*v.^2).*(2*u.*exp(1/4*opt.T*u.*(2*
        u+v))./(2*u+v) - 1);
theta = mod.r/mod.sigma - mod.sigma/2;
psi = exp(-1/2*theta^2*opt.T)*psibar(u*mod.sigma + theta, v*mod.sigma);
```

pde_solve.m

```
function [u0, U] = pde_solve(mod, opt, M, N)
% PDE_SOLVE solves the Black-Scholes PDE numerically on a bounded domain.
%
%   [u0, U] = PDE_SOLVE(mod, opt, M, N) returns u0 = u(0,S0), where u
%   solves the Black-Scholes partial differential equation with boundary
%   conditions u(t,L) = u(t,U) = 0 for 0 <= t < T and u(T,x) = h(x) for
%   L <= x <= U, and V the full numerical solution to this partial differential
%   equation. V is calculated by using a finite differences scheme with [0,T]
%   divided into N pieces and the domain [L,U] divided into M pieces. P is
%   calculated by linear interpolation.
%
%   The struct mod must contain the fields
%       S0     : Initial value
%       r      : Risk-free interest
%       sigma  : Volatility
%
%   The struct opt must contain the fields
%       T      : Time to maturity
%       h      : Function of terminal value, describing payoff at maturity
%       L      : Lower boundary
%       U      : Upper boundary

% Time step
dt = opt.T/N;
% Space step
dx = (opt.U - opt.L)/M;

% Initialize solution matrix
xs = opt.L:dx:opt.U;
U = zeros(M+1,N+1);
U(2:end-1, end) = opt.h(xs(2:end-1));

% Create matrix from matrix representation of system of linear equations.
a = @(x) dt*(mod.r*x/(2*dx) - mod.sigma^2*x.^2/(2*dx^2));
b = @(x) 1 + mod.r*dt + mod.sigma^2*x.^2*dt/dx^2;
c = @(x) -dt*(mod.r*x/(2*dx) + mod.sigma^2*x.^2/(2*dx^2));
D = zeros(M-1);
D(2:M:end) = a(xs(3:M));
D(1:M:end) = b(xs(2:M));
```

```
D(M:M:end) = c(xs(2:M-1));

for n = N:-1:1
    % Solve system of linear equations
    U(2:end-1, n) = D\U(2:end-1, n+1);
end

% Interpolate
u0 = interp1(xs, U(:,1), mod.S0);
```

log_pde_solve.m

```
function [v0, V] = log_pde_solve(mod, opt, M, N)
% LOG_PDE_SOLVE solves the log transformed Black-Scholes PDE numerically on a
% bounded domain.
%
%   [v0, V] = LOG_PDE_SOLVE(mod, opt, M, N) returns v0 = v(0,x0), where v
%   solves the log transformed Black-Scholes partial differential equation with
%   boundary conditions v(t,L) = v(t,U) = 0 for 0 <= t < T and v(T,x) = h(x) for
%   L <= x <= U, and V the full numerical solution to this partial differential
%   equation. V is calculated by using a finite differences scheme with [0,T]
%   divided into N pieces and the domain [L,U] divided into M pieces. P is
%   calculated by linear interpolation.
%
%   The struct mod must contain the fields
%       x0     : Initial value
%       r      : Risk-free interest
%       sigma  : Volatility
%
%   The struct opt must contain the fields
%       T      : Time to maturity
%       h      : Function of terminal value
%       L      : Lower boundary of domain
%       U      : Upper boundary of domain

% Time step
dt = opt.T/N;
% Space step
dx = (opt.U - opt.L)/M;

% Initialize solution matrix
xs = opt.L:dx:opt.U;
V = zeros(M+1,N+1);
V(2:end-1, end) = opt.h(xs(2:end-1));

% Create matrix from matrix representation of system of linear equations.
theta = mod.r - 1/2*mod.sigma^2;
a = dt*(theta - abs(theta))/(2*dx) - mod.sigma^2*dt/(2*dx^2);
b = 1 + mod.sigma^2*dt/dx^2 + abs(theta)*dt/dx + mod.r*dt;
c = -dt*(theta + abs(theta))/(2*dx) - mod.sigma^2*dt/(2*dx^2);
```

```
D = zeros(M-1);
D(2:M:end) = a;
D(1:M:end) = b;
D(M:M:end) = c;

for n = N:-1:1
    % Solve system of linear equations
    V(2:end-1, n) = D\V(2:end-1, n+1);
end

% Interpolate
v0 = interp1(xs, V(:, 1), mod.x0);
```

mc.m

```
function P = mc(mod, opt, M, N)
% MC runs the crude Monte Carlo algorithm.
%
%   P = MC(mod, opt, M, N) returns the time 0-price P of a barrier option with
%   payoff function h, estimated using the crude Monte Carlo estimator under the
%   Black-Scholes model with parameters specified by the struct mod.
%   The specifics of the option is  described by the struct opt. M trajectories
%   are simulated at N intermediate time-steps to estimate P.
%
%   The struct mod must contain the fields
%       S0     : Spot price
%       r      : Risk-free interest
%       sigma  : Volatility
%
%   The struct opt must contain the fields
%       T      : Time to maturity
%       h      : Function of terminal value, describing payoff at maturity.
%   and at least one of
%       L      : Lower boundary
%       U      : Upper boundary

dt = opt.T/N;

% Check boundaries and define indicator function and probabilities of not
% hitting the boundary.
if isfield(opt, 'L') && isfield(opt, 'U')
    ind = @(s) opt.L < s & s < opt.U;

    cutoff = 3;
    f = @(x,y) 1 - double_barrier_exit_prob(x, y, opt.L, opt.U, mod.sigma, dt,
        cutoff);
elseif isfield(opt, 'L')
    ind = @(s) opt.L < s;
    f = @(x,y) 1 - single_barrier_exit_prob(x, y, opt.L, mod.sigma, dt);
elseif isfield(opt, 'U')
```

```
    ind = @(s) s < opt.U;
    f = @(x,y) 1 − single_barrier_exit_prob(x, y, opt.U, mod.sigma, dt);
else
    error('No boundary specified in model.');
end

% Initialize variables
dW = randn(M,N);
G = zeros(M,N);
Snm1 = repmat(mod.S0, M, 1);
for n = 1:N
    % Sample next S values
    Sn = Snm1.∗exp((mod.r − mod.sigma^2/2)∗dt + mod.sigma∗sqrt(dt)∗dW(:,n));
    G(:,n) = ind(Sn).∗f(Snm1,Sn);
    Snm1 = Sn;
end
P = exp(−mod.r∗opt.T)∗sum(prod([opt.h(Sn) G],2))/M;
```

smc.m

```
function P = smc(mod, opt, M, N)
% SMC runs the sequential Monte Carlo algorithm.
%
%   P = SMC(mod, opt, M, N) returns the time 0−price P of a barrier option with
%   payoff function h, estimated using the sequential Monte Carlo estimator
%   under the Black−Scholes model with parameters specified by the struct mod.
%   The specifics of the option is described by the struct opt. M trajectories
%   are simulated at N intermediate time−steps to estimate P.
%
%   The struct mod must contain the fields
%       S0     : Spot price
%       r      : Risk−free interest
%       sigma  : Volatility
%
%   The struct opt must contain the fields
%       T      : Time to maturity
%       h      : Function of terminal value, describing payoff at maturity.
%   and at least one of
%       L      : Lower boundary
%       U      : Upper boundary

dt = opt.T/N;

% Check boundaries and define indicator function and probabilities of not
% hitting the boundary.
if isfield(opt, 'L') && isfield(opt, 'U')
    ind = @(s) opt.L < s & s < opt.U;

    cutoff = 3;
    f = @(x,y) 1 - double_barrier_exit_prob(x, y, opt.L, opt.U, mod.sigma, dt,
        cutoff);
elseif isfield(opt, 'L')
    ind = @(s) opt.L < s;
    f = @(x,y) 1 - single_barrier_exit_prob(x, y, opt.L, mod.sigma, dt);
elseif isfield(opt, 'U')
    ind = @(s) s < opt.U;
    f = @(x,y) 1 - single_barrier_exit_prob(x, y, opt.U, mod.sigma, dt);
else
    error('No boundary specified in model.');
end

% Initialize variables
dW = randn(M,N);
G = zeros(M,N);
E = zeros(N,1);
Snm1 = repmat(mod.S0, M, 1);

for n = 1:N
    % Sample next S values
    Sn = Snm1.*exp((mod.r - mod.sigma^2/2)*dt + mod.sigma*sqrt(dt)*dW(:,n));
    G(:,n) = ind(Sn).*f(Snm1, Sn);

    % If all trajectories crossed a barrier, the estimator is null
    if ~any(G(:,n))
        return
    end

    % Acceptance/rejection
    U = rand(M,1);
    reject = G(:,n) < U;

    % Resampling
    w = G(:,n)/sum(G(:,n));
    num_rejects = sum(reject);
    Sn(reject, :) = sample_weighted(num_rejects, Sn, w);

    % Estimate expectation
    E(n) = sum(G(:,n))/M;

    % Store resampled Sn for next step.
    Snm1 = Sn;
end
payoff = sum(opt.h(Sn))/M;
P = exp(-mod.r*opt.T)*payoff*prod(E);
```

single_barrier_exit_prob.m

```
function p = single_barrier_exit_prob(x, y, N, sigma, dt)
% SINGLE_BARRIER_EXIT_PROB calculates the probability of a GBM reaching N.
```

```
%
%   p = SINGLE_BARRIER_EXIT_PROB(x, y, N, sigma, dt) gives the probability
%   for a GBM S of hitting a barrier N over a time period of length dt,
%   conditional on S taking the values x and y at the end-points. The
%   parameter sigma specifies the volatility of S.
p = exp(-2*log(x/N).*log(y/N)/(sigma^2*dt));
```

double_barrier_exit_prob.m

```
function p = double_barrier_exit_prob(x, xprime, L, U, sigma, dt, cutoff)
% DOUBLE_BARRIER_EXIT_PROB calculates the probability of a GBM reaching L or U.
%
%   p = DOUBLE_BARRIER_EXIT_PROB(x, xprime, L, U, sigma, dt, cutoff) gives
%   the probability for a GBM S of hitting one of the barriers L and U over
%   a time period of length dt, conditional on S taking the values x and
%   xprime at the end-points. The parameter sigma specifies the volatility
%   of S. The probability is in theory given as an infinite series, and the
%   parameter cutoff specifies how many terms of this series to calculate.
y = log(xprime./x);
q = log(U/L);
a = log(x/L);
b = log(U./x);
R = @(y,z) exp(-2*z.*(z-y)/(sigma^2*dt));
p = 0;
for k = 1:cutoff
    p = p + R(y, q*k - a) + R(y, -q*k + b) - R(y, q*k) - R(y, -q*k);
end
```

sample_weighted.m

```
function y = sample_weighted(n, x, w)
% SAMPLE_WEIGHTED samples from a discrete distribution, taking values x.
%
%   y = SAMPLE_WEIGHTED(n, x, w) samples n random variables from the
%   distribution taking the value x(j) with probability w(j), for j =
%   1..length(x). Both x and w are assumed to be vectors and have the same
%   length.
if length(x) ~= length(w)
    error('Length mismatch between parameters ''x'' and ''w''.');
end
E = exprnd(1,n+1,1);
T = cumsum(E);
V = T./T(end);
k = 1; r = 1;
y = zeros(n,1);
W = cumsum(w);
while r <= n
    while V(r) < W(k) && r <= n
        y(r) = x(k);
        r = r + 1;
    end
    k = k + 1;
end
```

BIBLIOGRAPHY

Borovkov, K. and A. Novikov (2002). On a new approach to calculating expectations for option pricing. *Journal of Applied Probability 39*(4), 889–895.

Boyle, P. P. (1977). Options: A monte carlo approach. *Journal of financial economics 4*(3), 323–338.

Broadie, M., P. Glasserman, S. Kou, et al. (1997). A continuity correction for discrete barrier options. *Mathematical Finance 7*(4), 325–349.

Carmona, R., P. Del Moral, P. Hu, and N. Oudjane (2012). An introduction to particle methods with financial applications. In *Numerical methods in finance*, pp. 3–49. Springer.

Doucet, A., N. de Freitas, and N. Gordon (2001). *Sequential Monte Carlo Methods in Practice.* Springer Science & Business Media.

Duffy, D. J. (2013). *Finite Difference methods in financial engineering: a Partial Differential Equation approach.* John Wiley & Sons.

Friedman, A. (2006). *Stochastic Differential Equations and Applications.* Dover Publications.

Geman, H. and M. Yor (1996). Pricing and hedging double-barrier options: A probabilistic approach. *Mathematical finance 6*(4), 365–378.

Glasserman, P. (2003). *Monte Carlo methods in financial engineering*, Volume 53. Springer Science & Business Media.

Gordon, N. J., D. J. Salmond, and A. F. Smith (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, Volume 140, pp. 107–113. IET.

Gyöngy, I. (2015). On finite difference approximations in option pricing. Edinburgh University.

Heath, M. T. (2001). *Scientific Computing: An Introductory Survery.* McGraw-Hill.

Hull, J. C. (2009). *Options, Futures and Other Derivatives.* Pearson Education.

Karatzas, I. and S. E. Shreve (1991). Brownian motion and stochastic calculus. *Graduate Texts in Mathematics 113.*

Krylov, N. V. (2002). *Introduction to the theory of random processes.* American Mathematical Soc.

Kunitomo, N. and M. Ikeda (1992). Pricing options with curved boundaries1. *Mathematical finance 2*(4), 275–298.

Øksendal, B. (2003). *Stochastic differential equations.* Springer.

Shephard, N. G. (1991). From characteristic function to distribution function: a simple framework for the theory. *Econometric theory 7*(04), 519–529.

Shevchenko, P. V. and P. Del Moral (2014). Valuation of barrier options using sequential Monte Carlo. *Available at SSRN 2529539*.

Shreve, S. E. (2004). *Stochastic calculus for finance II: Continuous-time models*, Volume 11. Springer Science & Business Media.