Pricing options with the SABR Model

Geeske Vlaming

June 30, 2008

Universiteit Utrecht

SAEN OPTIONS

Acknowledgements

This research could not have been possible without the guidance and support of many people from a variety of institutions. In particular, I would like to express my gratitude towards the undermentioned.
To me, the most important and inspiring person during this research project was Michel Vellekoop, from the University of Twente. As my main supervisor, he initially suggested this interesting area of research, and gave excellent and sustained guidance and support during the course of this project. Together with my other two supervisors, Bastiaan de Geeter of Saen Options B.V. and Karma Dajani of the University of Utrecht, they formed a team with great expertise, providing me with adequate additional insights in mathematics and option modeling and giving me useful comments on the draft versions of this thesis.
In addition, I would like to express my appreciation to Saen Options B.V. and its employees, for the opportunity afforded me to do research on this subject under supervision of Michel Vellekoop, for providing me access to the required data, and for the pleasant and constructive working atmosphere. My co-intern Hanafi Auzar I thank for all the discussions we had on a wide variety of subjects.
Finally, it is my closest friend and partner Steven Fernandes, whom I want to thank for his endless support and encouragement, and especially for structuring the start-up of this research.

Abstract

This thesis discusses the pricing of stock options using the recently developed SABR model. The SABR model assumes the volatility to be stochastic, and its main advantage compared to other stochastic volatility models is that there exists a direct formula which approximates prices of European options. The main question of this thesis is:

- Is the SABR model a good model to use when pricing European and American options?

To answer this question, three different subquestions are dealt with, namely:

- In which situations is the approximating direct formula good enough to price European options?
- What is a good numerical method to price European options under the SABR model?
- What is good method to price American options under the SABR model?

To price European options under the SABR model, we design two numerical methods. The first one uses Monte Carlo simulations, and the second one is based on the Vellekoop and Nieuwenhuis tree method. By comparing the results of these two methods and the approximating direct formula, for different chosen values of the parameters, it is shown that the direct formula approximates prices of European options well for most values of the parameters, but not for all. Furthermore, it is demonstrated that the two numerical methods give similar results for the price of European options, but that the tree method is much faster. This is why we use this tree method, when applying the SABR model to real market data of European options.
To price American options, once more two numerical methods are designed. The first one is based on the Least-Squares Monte Carlo method, and the second one is the same tree method used to price European options under the SABR model, with some slight modifications. By comparing the results of these two methods for the different chosen values of the parameters, we see that the two numerical methods give similar results for the prices of American options, but that the tree method is again much faster. This is why we use this tree method again, when applying the SABR model to real market data of American options.
From the results of the fitting of the SABR model to real market data, it follows that the approximating direct formula can be used to price real European options. Furthermore, the results show that the SABR model is indeed a good model to use when pricing European and American options. It is even demonstrated that the SABR model prices American options with short maturity times more accurately than the commonly used Heston model.

Contents

Chapter 1

Introduction

This thesis discusses the pricing stock options using the SABR model. Before the SABR model itself is introduced, first a small introduction about options is given, as well as some basic mathematical definitions, and a short historical description of the development of models to price options.

1.1 Introduction to options

An option is a financial product one can buy (or sell). A buyer of a call option buys a contract, which gives the owner the right to buy some asset at a certain price and at a certain time or during a certain time period. A put option is equal to a call option except for the fact that it gives the right of selling instead of buying this asset. This asset is called "the underlying", and normally this is a stock or collection of stocks, as it will be in this thesis, but it can be anything with value. The price for which the contract gives the right to buy or sell the underlying is called the strike price, and the time at which or until which the contract gives this right is called maturity (time) [12].
The two most commonly known kinds of options are European and American options. A European option can only be exercised at maturity, while an American option can be exercised at any time between the time of buying and maturity [12].
Very important is to notice that the owner of an option does not have to exercise the contract, so the pay-off of the option will never be negative, and it can of course be positive. That is why options have a positive value, and the question arises how to determine this value.

1.2 Mathematical background

To be able to understand how to derive the value of an option, some basic mathematical theories and definitions must be known. In this section most of these will be given, but the reader is assumed to have some basic knowledge of probability theory.
For every random variable X one can speak of its expected value $\mathbb{E}[X]$. To be able to derive this value, it has to be known at which probability space $(\Omega, \mathcal{F}, \mathbb{P})$ this variable is defined. Ω is the set of all possible values of X, $\mathcal{F}$ is a σ-algebra of subsets of Ω and $\mathbb{P}$ is the probability measure. $\mathcal{F}$ is a σ-algebra if:

- the empty set belongs to $\mathcal{F}$
- whenever a set A belongs to $\mathcal{F}$, its complement A^c also belongs to $\mathcal{F}$, and
- whenever a sequence of sets A_1, $A_2, ...$ belongs to $\mathcal{F}$, their union $\cup_{n=1}^{\infty} A_n$ also belongs to $\mathcal{F}$. [18]

A probability measure $\mathbb{P}$ is a function that gives every set $A \in \mathcal{F}$ a probability, so $\mathbb{P}(A) \in [0, 1]$. It is required that:

- $\mathbb{P}(\Omega) = 1$, and
- whenever A_1, $A_2, ...$ is a sequence of disjoint sets in $\mathcal{F}$, i.e. $A_i \cap A_j = \emptyset$ for $i \neq j$, then

$$\mathbb{P}(\cup_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} \mathbb{P}(A_n)$$

Assume that X is a random variable that is known at time T. We define $\mathcal{F}_t$ to be all information that is known by time t. At time $t = 0$, there is nothing known yet, so $\mathcal{F}_0 = \{\emptyset, \Omega\}$, and at time T, all information is known, so $\mathcal{F}_T =$ the σ-algebra of all subsets of Ω. Furthermore at time $s \leq t$ there is not more information known then at time t, so $\mathcal{F}_s \subset \mathcal{F}_t$. A collection of σ-algebra's $\mathcal{F}_i$ is called a filtration if $\mathcal{F}_i \subset \mathcal{F}_j$ for $i \leq j$, so the collection of σ-algebra's $\mathcal{F}_t$, $0 \leq t \leq T$, is a filtration [18].
X can also be defined as the collection of random variables $\{X_t; 0 \leq t \leq T\}$, where X_t is $\mathcal{F}_t$-measurable, which means that the information known at time t determines the value of X_t. A collection of random variables such that X_t is $\mathcal{F}_t$-measurable, for each $t \in [0, T]$ is called an adapted stochastic stochastic process to the filtration $\mathcal{F}$ [18]. In the context of this thesis, X_t can be seen as the price of an asset, and $\mathcal{F}_t$ as all market values of the asset between time zero and time t.
A conditional expectation is an expectation of a random variable given some information, which can be denoted by $\mathbb{E}[X_t|\mathcal{G}]$, where $\mathcal{G} \subset \mathcal{F}_t$ and $\mathcal{G}$ also is a σ-algebra [18]. $\mathbb{E}[X_t|\mathcal{F}_0] = \mathbb{E}[X_t]$, since at $t = 0$ no information is known, and $\mathbb{E}[X_t|\mathcal{F}_t] = X_t$, since X_t is $\mathcal{F}_t$-measurable, so X_t is known. This makes it possible to define another important concept of financial mathematics, namely a martingale.
Using X_t as defined above, we say that X_t is a martingale if $\mathbb{E}[X_t|\mathcal{F}_s] = X_s$ for all $0 \leq s \leq t \leq T$, which means that X_t has no tendency to rise or fall [18].

In a normal market there exists a risk-free interest rate $r(t)$, which means that putting 1 euro on a bank account now, will grow to $e^{\int_0^t r(s)ds}$ in t years from now. From this it follows that, if the expected value of an investment at time $t = 1$ is x, the value of this investment now is $e^{-\int_0^1 r(s)ds}x$. The factor $e^{-\int_{t_1}^{t_2} r(s)ds}$ is called the discount factor from time t_2 to t_1, and is denoted by $D(t_1, t_2)$.
An important assumption, when deriving the price of an option, is that it is not possible to make a risk-free profit. The possibility of making a risk-free profit is often referred to as an arbitrage opportunity. The mathematical definition of an arbitrage opportunity is a value process X_t satisfying $X_0 = 0$, and also satisfying, for some time $T > 0$

$$\mathbb{P}\{X_T \geq 0\} = 1, \quad \mathbb{P}\{X_T > 0\} > 0$$

This value process X_t is a portfolio consisting of assets and options [18]. Intuitively, this assumption can be understood by the fact, that if there would exist an arbitrage opportunity, everyone would directly want to own this portfolio, which would drive up the price of the options and assets which one should buy in this portfolio, and drive down the price of the options and assets which one should sell to have this portfolio. The arbitrage opportunity would disappear instantly.
There exist no arbitrage opportunities if and only if there exists a risk neutral probability measure $\mathbb{Q}$ [17]. And another important lemma is that, if X_t is the value of a portfolio, under a risk-neutral probability measure, the discounted portfolio value $D(0, t)X_t$ is a martingale [18]. Together this means that, under the assumption of no arbitrage, for every value process X_t, there exists a risk-neutral probability measure $\mathbb{Q}$, such that the value of X_t should be equal to the discounted expected value of X_T, under $\mathbb{Q}$, given all information up to time t. In shorter notation this means that, for every value process X_t, there exists a risk-neutral probability measure $\mathbb{Q}$, such that

$$X_t = D(t, T)\mathbb{E}_{\mathbb{Q}}[X_T|\mathcal{F}_t]$$

Another definition which should be known when discussing option pricing, is of a Brownian motion. A Brownian motion W_t is a stochastic process which has a normal distribution $\mathcal{N}(0, t)$, $W_0 = 0$ and each increment $W_t - W_s$ is independent of all information before time s, when $s \leq t$. Which means that $W_t - W_s$ is normally distributed with mean 0 and variance $t - s$ [18]. A stochastic process based on a Brownian motion can be defined by the following stochastic differential equation(SDE):

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t,$$

where $a(X_t, t)$ and $b(X_t, t)$ are adapted processes to the filtration of the Brownian motion W_t.
An important formula, which is often used in financial mathematics, is the Itô formula. If the stochastic differential equation of X_t is as given above, then the SDE for $f(X_t, t)$ is given by the Itô formula, which is:

$$df(x,t) = (\frac{\partial f}{\partial t} + a(x,t)\frac{\partial f}{\partial x} + \frac{1}{2}b(x,t)^2\frac{\partial^2 f}{\partial x^2})dt + b(x,t)\frac{\partial f}{\partial x}dW_t, \tag{1.1}$$

The final theorem given in this section is the martingale representation theorem (for one dimension) [18]: Let W_t, $0 \leq t \leq T$, be a Brownian motion on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and let $\mathcal{F}_t$, $0 \leq t \leq T$, be the filtration generated by this Brownian motion. Let M_t, $0 \leq t \leq T$, be a martingale with respect to this filtration, so for every t, M_t is $\mathcal{F}_t$-measurable and for $0 \leq s \leq t \leq T$, $\mathbb{E}[M_t|\mathcal{F}_s] = M_s$. Then there exists an adapted process C_u, $0 \leq u \leq T$, such that

$$M_t = M_0 + \int_0^t C_u dW_u, \quad 0 \leq t \leq T,$$

or in stochastic differential equation form:

$$dM_t = C_t dW_t$$

1.3 History of option pricing models

The price of one option is also a value process on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is $\mathbb{R}^+$, $\mathcal{F}$ is the filtration consisting of σ-algebra's $\mathcal{F}_t$, which contains all information given by the market until time t. Since option-pricing theory is based on the assumption of no arbitrage possibilities, it is known that for every value process, there exists a risk-neutral measure such that its value at time t is equal to its discounted expected value at time T, given all information up to time t. This means that the value of a European option is given by:

$$V(t) = \mathbb{E}_{\mathbb{Q}}[D(t,T)V(T)|\mathcal{F}(t)], \tag{1.2}$$

where $V(t)$ is the price of the option at time t, $V(T) = \max(c * (S_T - K), 0)$, where $c = 1$ means it is a call option, $c = -1$ means it is a put option and S_t is the value of the underlying at time t, $\mathbb{Q}$ is the risk-neutral measure, $D(t,T)$ is the discount factor between time t and T, T is the maturity time and $\mathcal{F}(t)$ is the σ-algebra which models all the information known up to time t [18].
According to the lemma given above about $\mathbb{Q}$, the risk-neutral measure, we also know that $\mathbb{E}_{\mathbb{Q}}[D(0,t)S_t] = S_0$, in other words, $D(0,t)S_t$ is a martingale under $\mathbb{Q}$. Using the martingale representation theorem, under the assumptions that the model is based on one Brownian motion and that the interest rate is a deterministic function of time, it now follows that there exists an adapted process $\widehat{C}_t$, such that :

$$dD(0,t)S_t = \widehat{C}_t dW_t, \quad D(0,0)S_0 = s$$

Using the Itô formula with $X_t = D(0,t)S_t$ and $f(X_t,t) = e^{\int_0^t r(s)ds}X_t$, the stochastic differential equation of the underlying is given by:

$$dS_t = (\frac{d}{dt}e^{\int_0^t r(s)ds}X_t + 0 + 0)dt + \widehat{C}_t e^{\int_0^t r(s)ds}dW_t \Rightarrow dS_t = r(t)S_t dt + C_t dW_t, \quad S_0 = s,$$

where C_t is again an adapted process.

In 1973, Black and Scholes [3] assumed "ideal conditions" in the market for the stock and for the option. That is, they assumed that the interest rate is constant, that C_t is equal to the volatility (σ) times S_t, and that the underlying does not pay dividend. So they assumed the following model for the underlying, further called the Black-Scholes model:

$$dS_t = rS_t dt + \sigma S_t dW_t, \quad S(0) = s, \tag{1.3}$$

for fixed constants $r > 0$ and $\sigma > 0$.
Under these assumptions, they proved that there exists a direct formula for the price of a European call option, namely:

$$V_{call}(t) = S_t \mathcal{N}(d_1) - e^{-r(T-t)} K \mathcal{N}(d_2) \tag{1.4}$$

with

$$d_{1,2} = \frac{\ln(\frac{S_t}{K}) + (r \pm \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}},$$

where $V_{call}(t)$ is the price of the call option at time t, K is the strike price of the option and $\mathcal{N}$ is the standard normal (cumulative) distribution function [3]. By the put-call parity it follows that :

$$V_{put}(t) = V_{call}(t) + e^{-r(T-t)}K - S_t$$

This put-call parity can be explained by the no-arbitrage principle. A portfolio consisting of one call option and $e^{-r(T-t)}K$ cash in the bank at time t, will give a pay-off at maturity time of $\max(S_T - K, 0) + K = \max(S_T, K)$. A portfolio of one put option plus one underlying at time t, will give a pay-off at maturity time of $\max(K - S_T, 0) + S_T = \max(K, S_T)$. This shows that both portfolio's have the same value at maturity and should therefore have the same value at time t, so $V_{call}(t) + e^{-r(T-t)}K = V_{put}(t) + S_t$.
European options are often priced using the Black-Scholes formula, but the problem with doing so, is that in practice options with different strikes K or different maturies T, require different volatilities. The Black-Scholes formula is a bijective function from volatility to the price of the European option once other parameters have been fixed. This can be seen from the fact that, at the time of valuating the option, the volatility is the only unknown, and the price is an increasing function of volatility. It follows now that from every price of a European option, one can derive the Black-Scholes volatility, also called the implied volatility, σ_B. And the other way around, the implied volatility gives the price of the option. So, in other words, the problem with using the Black-Scholes model is that the implied volatility σ_B nearly always varies with different strikes or maturity times, so the assumption of constant volatility is too strong.
The function for the implied volatility in terms of the strike price, with T given, is often referred to as the market skew or smile. The word skew is usually used for the slope of this function, while the word smile usually is reserved for its curvature. Handling these market smiles and skews correctly is critical for hedging (reducing risk using trading strategies), and the Black-Scholes model does not do that.
A major step in handling smiles and skews was the development of local volatility models. It was posed that in the Black-Scholes model C_t was too restricted, and that instead one should only assume that C is Markovian [5, 6], which means that if the value of C_s is known, with $s \leq t$, than the value of C_t is independent of all information before time s. Rewriting C_t as $\sigma_{loc}(t, S(t))S_t$ yields the local volatility model, with σ_{loc} a deterministic function.
Dupire [6] argued that one should use calibration to market prices of liquid, which means often traded, European options to obtain $\sigma_{loc}(t, S_t)$. This means that one should start with a given local volatility function, and evaluate the price of the option using equation (1.2) to obtain the theoretical option price. After that, one should vary the local volatility function until the theoretical prices match the market prices for each strike K and maturity T. Commonly the local volatilities are taken to be piecewise constant in time between two successive exercise dates [9]. Once $\sigma_{loc}(t, S_t)$ has been obtained, the model correctly reproduces the market prices of European call and put options for all strikes K and maturity dates T. The local volatility model thus provides a method of pricing and hedging options in the presence of market smiles and skews.
Unfortunately, the local volatility model predicts the wrong dynamics of the implied volatility curve, which leads to inaccurate and often unstable hedges. Local volatility models predict that the market smile/skew moves in the opposite direction as the price of the underlying asset, which is opposite to typical market behavior [9]. It was even seen that the original Black-Scholes model yields more accurate hedges than the local volatility model, even though the local volatility model is self-consistent across strikes and the Black-Scholes model is inconsistent. To resolve this problem the SABR model was formulated.

1.4 The SABR model

Since the local volatility model fails to give accurate hedges, the conclusion is that we cannot use a Markovian model based on a single Brownian motion [9]. To develop a new model, there are now three options. The first one is to make the model non-Markovian. The second one is to keep it Markovian, but base the model on other stochastic processes than Brownian motion. And the third one is to develop a two factor model. In the derivation of the SABR model, Hagen et al. [9] chose the third option, since the properties of a Markovian model based on Brownian motions still seemed to be reasonable for stock prices. Furthermore, they used the suggestion that the volatility is not constant, but is itself a random function of time, and they used the fact that the Constant Elasticity of Variance (CEV) model gives a better approximation of the prices of European options than the Black-Scholes model, since the implied volatility curve given by this model does depend on the strike price, so is not constant over strikes [12]. In the definition of the CEV model, the risk-neutral process of the underlying S_t is given by:

$$dS_t = rS_t dt + \sigma S_t^{\beta} dW_t,$$

where r is the risk-free interest rate, W is a Brownian motion, σ is the volatility and β is a positive constant [12].

By choosing the simplest reasonable processes using the mentioned suggestions, the "stochastic-$\alpha\beta\rho$ model" was derived, which has now become known as the SABR model. In this model, the forward price F_t and the volatility α_t are given by:

$$\begin{aligned} dF_t &= \alpha_t F_t^{\beta}\ dW_t^1, \quad F_0 = f \\ d\alpha_t &= \nu\alpha_t\ dW_t^2, \quad \alpha_0 = \alpha, \end{aligned} \tag{1.5}$$

under the risk neutral measure ($\mathbb{Q}$), where the elasticity coefficient $\beta \in [0,1]$ and the volatility of the volatility $\nu > 0$ are assumed to be known constants, and where the Brownian motions W^1 and W^2 are correlated (under $\mathbb{Q}$) with

$$d\left\langle W_t^1, W_t^2 \right\rangle = \rho dt,$$

for some $\rho \in [-1,1]$.
Special cases of the SABR model are the cases where $\beta = 0$ or $\beta = 1$. When $\beta = 0$, the movement of the forward price becomes independent of the price itself. Since a Brownian motion W_t has a normal distribution with mean zero and variance t, it follows that in this case F_t has a normal distribution with stochastic variance. When $\beta = 1$, it follows that F_t has a lognormal distribution. This can be seen using the Itô formula (1.1), with $f(x,t) = \ln F_t$:

$$\frac{\partial f}{\partial t} = 0, \quad a(x,t)\frac{\partial f}{\partial x} = 0, \quad \frac{1}{2}b(x,t)^2\frac{\partial^2 f}{\partial x^2} = \frac{1}{2}\alpha_t^2 F_t^2 \frac{-1}{F_t^2}, \quad b(x,t)\frac{\partial f}{\partial x} = \alpha_t F_t \frac{1}{F_t}$$

So

$$d\ln F_t = \frac{-1}{2}\alpha_t^2 dt + \alpha_t dW_t^1,$$

which means that $\ln F_t$ has a normal distribution, so F_t has a lognormal distribution.
The forward price is the expected value of the underlying asset at maturity, under the risk neutral measure [12]. When the underlying asset pays no dividend, this means that:

$$F_t = S_t e^{\int_t^T r(s)ds} = S_t / D(t,T).$$

When the underlying asset pays dividend, this formula for the forward price, and the pricing of options in general, becomes more complicated. This is because the owner of the option does not receive anything of the dividend, but the price of the asset drops at the time of dividend with the amount equal to the dividend paid [12].
When pricing options, there can be three different models for dividends. The first one is called dividend yield. The second one is a discrete proportional dividend. And the third kind is a discrete fixed cash dividend.
Dividend yield assumes a continuously paid amount of dividend of $100q\%$ of the underlying asset. This can be the case when the option is on an index, because an index consists of a huge amount of stocks from different companies, which all pay dividend at different times, so assuming a continuously paid dividend gives a good approximation. In this case, the formula for the forward price becomes as follows:

$$F_t = S_t e^{\int_t^T (r(s)-q)ds}$$

Discrete proportional dividend is dividend that is only paid once in a while, thus not continuously, and this dividend payment will be a fixed and initially known proportion of the price of the underlying. The formula for the forward price becomes:

$$F_t = e^{\int_t^T r(s)ds} S_t (1-q)^{n(T)-n(t)},$$

where q is the proportion of S_t that will be paid as dividend, and the function $n(t)$ is the number of times dividend is paid between time zero and time t.
Discrete fixed cash dividend is actually what it says. It is discretely paid dividend, of which the value is initially known, so the value does not depend on the value of the underlying. In this case, the formula for the forward price becomes:

$$F_t = e^{\int_t^T r(s)ds} S_t - \sum_{i=1}^{j} c_i e^{\int_{t_i}^T r(s)ds} 1_{t_i > t},$$

where j is the number of dividends paid between time zero and maturity, c_i is the value of the i'th dividend, and t_i is the time at which the i'th dividend will be paid.

Since it is more convenient to work with stock prices instead of forward prices, the Itô formula (1.1) can be applied to the stochastic differential equation of the forward price (1.5), using the different formulas for the forward price, to derive more practical ways to write the model by Hagan et al.. First the SDE of S_t is derived in case it pays no dividend.
In this case $X_t = F_t$, $f(F_t,t) = D(t,T)F_t$, $a(F_t,t) = 0$ and $b(F_t,t) = \alpha_t F_t^{\beta}$, in the Itô formula. The only parts of the Itô formula left to calculate are the partial derivatives of the function f. These are derived as follows:

$$D(t,T) = e^{-\int_t^T r(s)ds}, \quad \frac{\partial f}{\partial t} = \frac{\partial D(t,T)}{\partial t}F_t = r(t)S_t$$

$$\frac{\partial f}{\partial F_t} = D(t,T), \quad \frac{\partial^2 f}{\partial F_t^2} = 0$$

Putting now all this information in the Itô formula, the SABR model in terms of S_t, in case there are no dividends paid, is derived:

$$dS_t = (r(t)S_t + 0 + 0)dt + \alpha_t F_t^{\beta} D(t,T)dW_t^1, \quad S_0 = s \Rightarrow$$

$$dS_t = r(t)S_t dt + \alpha_t D(t,T)^{1-\beta} S_t^{\beta} dW_t^1, \quad S_0 = s \tag{1.6}$$

$$d\alpha_t = \nu\alpha_t \; dW_t^2, \quad \alpha_0 = \alpha,$$

where the parameters and the correlation between the Brownian motions are the same as in equations (1.5).
In the cases of dividend payments of the underlying, X_t, $a(X_t,t)$ and $b(X_t,t)$ are defined equally, but the function $f(X_t,t)$ changes, so the partial derivatives also change.
In the case of dividend yield,

$$f(F_t,t) = e^{-\int_t^T (r(s)-q)ds}F_t,$$

and thus:

$$\frac{\partial f}{\partial t} = (r(t)-q)e^{-\int_t^T (r(s)-q)ds}F_t = (r(t)-q)S_t,$$

$$\frac{\partial f}{\partial F_t} = e^{-\int_t^T (r(s)-q)ds}, \quad \frac{\partial^2}{\partial F_t^2} = 0$$

This leads to the following SDE for S_t:

$$dS_t = (r(t)-q)S_t dt + \alpha_t e^{-(1-\beta)\int_t^T (r(s)-q)ds} S_t^{\beta} dW_t,$$

If S_t is assumed to pay discrete proportional dividends,

$$f(F_t,t) = F_t e^{-\int_t^T r(s)ds}(1-q)^{n(t)-n(T)}.$$

In this formula, the function $n(t)$ depends on t, but is constant in the intervals $[t_i, t_{i+1})$, where t_i is the time at which dividend i is paid, $i = 0, ..., j$, $j = n(T)$ is the total number of dividends paid, $t_0 = 0$ and $t_{j+1} = T$. Furthermore, S_t is discontinuous on all dividend time points, but is continuous in all intervals (t_i, t_{i+1}), so we get that for all $t \in (t_i, t_{i+1})$, with $i = 0, ..., j$:

$$\frac{\partial f}{\partial t} = F_t r(t) e^{-\int_t^T r(s)ds}(1-q)^{i-j} = r(t)S_t$$

$$\frac{\partial f}{\partial F_t} = e^{-\int_t^T r(s)ds}(1-q)^{i-j}, \quad \frac{\partial^2 f}{\partial F_t^2} = 0$$

So for all $t \in (t_i, t_{i+1})$:

$$dS_t = r(t)S_t dt + \alpha_t e^{-(1-\beta)\int_t^T r(s)ds}(1-q)^{(1-\beta)(i-j)} S_t^{\beta} dW_t$$

Together with the fact that $S_{t_i} = (1-q)S_{t_i^-}$, where t_i^- means just before the dividend i is paid, so $n(t_i^-) = i-1$, now the whole process of S_t is given.
If S_t is assumed to pay fixed cash dividends,

$$f(F_t,t) = (F_t + \sum_{i=1}^{j} c_i e^{\int_{t_i}^T r(s)ds} 1_{t_i > t}) e^{-\int_t^T r(s)ds},$$

The derivation of the SDE of S_t goes analogously to the discrete proportional dividend case. The function $\sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i>t}$ depends on t, but is constant again in the intervals $[t_i, t_{i+1})$. S_t is discontinuous on all dividend time points, but is continuous in all intervals (t_i, t_{i+1}). So we get that for all $t \in (t_i, t_{i+1})$, with $i = 0, ..., j$:

$$\frac{\partial f}{\partial t} = (F_t + \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i>t}) r(t) e^{-\int_t^T r(s)ds} = r(t)S_t$$

$$\frac{\partial f}{\partial F_t} = e^{-\int_t^T r(s)ds}, \quad \frac{\partial^2 f}{\partial F_t^2} = 0$$

So for all $t \in (t_i, t_{i+1})$:

$$dS_t = r(t)S_t dt + \alpha_t e^{-\int_t^T r(s)ds} \max(e^{\int_t^T r(s)ds} S_t - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i>t}, 0)^\beta dW_t,$$

where we take the maximum with zero, since the value of the underlying can not become negative, even if the paid dividend is higher than the value of the underlying.
Together with the fact that $S_{t_i} = S_{t_i^-} - c_i$, now the whole process of S_t is given.

The stochastic differential equation of S_t in the SABR model is now given for all the different dividend cases. Unfortunately, it has to be noted that the value of the maturity time and the values of the dividends paid between time t and maturity appeared in these SDE's. There can be numerous options with different maturity times on the same underlying, so this dependency on the maturity time of one option can not be right. Despite this flaw in the model, the model that is investigated in this thesis will not be changed, since this is the SABR model stated by Hagan et al., which has a main virtue, namely there exists an approximating direct formula for the price of a European option under this model.
Many other stochastic volatility models have been proposed, of which the Heston model is the most popular [21]. This model is described, under $\mathbb{Q}$, by the following two stochastic differential equations:

$$\begin{aligned} dS_t &= r(t)S_t dt + \sqrt{\sigma_t} S_t dW_t^1 \\ d\sigma_t &= \kappa(\theta - \sigma_t)dt + \omega\sqrt{\sigma_t} dW_t^2, \end{aligned} \tag{1.7}$$

where the Brownian motions are again correlated [21].
A clear difference between the Heston and the SABR model is that the drift term of the volatility in the Heston model, causes the volatility to stay around θ, while in the SABR model ν (the volatility of the volatility) is not bounded, so the volatility can fluctuate much more. Furthermore, if we assume that $\beta = 1$ in the SABR model, and the underlying does not pay fixed cash dividends, it appears that $\alpha_t = \sqrt{\sigma_t}$ to get the same SDE for the underlying in both models. Using the Itô formula (1.1), where $x = \sigma_t$, $a(\sigma_t, t) = \kappa(\theta - \sigma_t)$ and $b(\sigma_t, t) = \omega\sqrt{\sigma_t}$, the SDE of α_t in the Heston model can now be derived:

$$f(x,t) = \sqrt{\sigma_t}, \quad \frac{\partial f}{\partial t} = 0,$$

$$\frac{\partial f}{\partial \sigma_t} = \frac{1}{2\sqrt{\sigma_t}}, \quad \frac{\partial^2 f}{\partial \sigma_t^2} = \frac{-1}{4\sigma_t\sqrt{\sigma_t}}$$

This leads to the following SDE for α_t in the Heston model:

$$d\alpha_t = (\kappa(\theta - \sigma_t)\frac{1}{2\sqrt{\sigma_t}} + \frac{1}{2}\omega^2\sigma_t\frac{-1}{4\sigma_t\sqrt{\sigma_t}})dt + \omega\sqrt{\sigma_t}\frac{1}{2\sqrt{\sigma_t}}dW_t^2 \Rightarrow$$

$$d\alpha_t = \frac{1}{2\alpha_t}(\kappa(\theta - \alpha_t^2) - \frac{1}{4}\omega^2)dt + \frac{\omega}{2}dW_t^2 \tag{1.8}$$

Another important difference that now appears is that the diffusion function of α_t is in the Heston model equal to $\frac{\omega}{2}$, while it is in the SABR model defined by $\nu\alpha_t$. This means that in the Heston model α_t is assumed to have a normal distribution, while in the SABR model α_t is assumed to have a lognormal distribution. Although this can only be stated in the case that $\beta = 1$, this is an important difference to notice.

The Heston model is often used, but there does not exist an approximating direct formula for the price of a European option under the Heston model. That is why the SABR model has a big advantage compared to the Heston model, and it is worth investigating this model.

1.5 Research questions

The main question of this thesis is:

- Is the SABR model a good model to use when pricing European and American options?

To answer this question, different subquestions should be answered.
In the second chapter, two of them are discussed, namely:

- In which situations is the approximating direct formula good enough to price European options?
- What is a good numerical method to price European options under the SABR model?

To answer these questions, first the approximating direct formula is explained, after which two numerical methods are designed, which generate the prices of European options under the SABR model. One of these methods is based on Monte Carlo simulations and the other one is based on trees. For different chosen values of the parameters, results of the three different methods are derived, after which the two subquestions will be answered.
The third chapter deals with American options and an answer to the following subquestion is given:

- What is good method to price American options under the SABR model?

The answer to this question is found in the same way as in chapter two, except that there does not exist an approximating direct formula to price American options.
In the fourth chapter it is explained how the parameters should be fitted to real market data, and results of applying the SABR model to real market data are given. The results are also compared with results of the Heston model, based on a masters thesis by Ter Horst [20], and the main question is answered.
In the fifth and final chapter of this thesis, final conclusions together with some discussion and recommendations for further investigation are presented.

Chapter 2

Deriving prices for European options under the SABR model

In this chapter, two numerical methods are designed to approximate prices of European options under the SABR model. One is based on Monte Carlo simulations and the other is based on trees. In section 2.4 results of the three methods are given for chosen parameter values. From these results it can be seen whether the methods give the same prices for the same parameter values, and by comparing the results of the numerical methods with the prices given by the approximating direct formula, it can be seen whether this formula approximates the prices under the SABR model closely. Furthermore, it is shown which numerical method is the best to use for the pricing of ‘real’ European options under the SABR model.

2.1 The approximating direct formula

In the theoretical introduction the Black-Scholes formula (1.4) has already been discussed. Hyland et al. [13] proved that the assumption of constant interest rate in the Black-Scholes model can be relaxed, while a similar formula still holds, namely:

$$V_{call}(t) = S_t\mathcal{N}(d_1) - e^{-\int_t^T r(s)ds}K\mathcal{N}(d_2) \equiv V_{put}(t) + (S_t - e^{-\int_t^T r(s)ds}K) \tag{2.1}$$

with

$$d_{1,2} = \frac{\ln(\frac{S_t}{K}) + \int_t^T r(s)ds \pm \frac{1}{2}\sigma^2(T-t)}{\sigma\sqrt{T-t}}$$

since $\ln(xy) = \ln(x) + \ln(y)$, this formula can be written in an easier form by using the forward value F [2]:

$$V_{call}(t) = e^{-\int_t^T r(s)ds}(F_t\mathcal{N}(d_1) - K\mathcal{N}(d_2)) \equiv V_{put}(t) + e^{-\int_t^T r(s)ds}(F_t - K) \tag{2.2}$$

with

$$d_{1,2} = \frac{\ln(\frac{F_t}{K}) \pm \frac{1}{2}\sigma^2(T-t)}{\sigma\sqrt{T-t}}$$

This formula is called Black’s formula. The formula is still a bijective function from volatility to the price of the option, which means that from every price of a European option, thus also for prices given by the SABR model, one can derive σ_B, the (Black-)implied volatility.
Hagan et al. derived, with perturbation techniques and using formula (2.2), in the paper “Managing smile risk’ [9], an approximating direct formula for this implied volatility under the SABR model, namely:

$$\begin{aligned}\sigma_B(f,K) \approx & \frac{\alpha}{(fK)^{(1-\beta)/2}\{1+\frac{(1-\beta)^2}{24}\ln^2(\frac{f}{K})+\frac{(1-\beta)^4}{1920}\ln^4\frac{f}{K}\}}(\frac{z}{x(z)}) \\ & \{1+[\frac{(1-\beta)^2}{24}\frac{\alpha^2}{(fK)^{1-\beta}} + \frac{1}{4}\frac{\rho\beta\nu\alpha}{(fK)^{(1-\beta)/2}} + \frac{2-3\rho^2}{24}\nu^2](T-t)\},\end{aligned} \tag{2.3}$$

where f is the forward price, and where z and $x(z)$ are defined by:

$$z = \frac{\nu}{\alpha}(fK)^{(1-\beta)/2}\ln(\frac{f}{K})$$

$$x(z) = \ln(\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho})$$

This means that an approximating direct formula for the price of a European option is given by the Black-Scholes formula (2.2), where σ is replaced by the σ_B defined above.
Since the formula is given in terms of the forward price, the same formula can be used in case of the underlying paying no dividend as well as the underlying paying dividend. The only difference between these cases is the formula used to derive the price of the forward. These different formulas are described in section 1.4.
Not only is this method a very quick way to find the price of a European option, but by using this approximating direct formula for implied volatility, also, good starting values for the parameters can be derived, which are needed to fit the parameters of the SABR model to real market data. How these starting values can be derived from this formula is described in section 4.1.

2.2 The Monte Carlo method

The Monte Carlo method is a method in which different paths for the price of the underlying asset are simulated under $\mathbb{Q}$, and the value of the option is set equal to the mean of the payoff each path generates, discounted to time zero [12]. It is a very time consuming method, but it is fairly easy to apply, even for options for which the price depends on the entire path of the underlying asset, and/or for options that involve more than one stochastic process.
An example of using the Monte Carlo method is given here by applying the method to the Black-Scholes model with time-dependent interest rate, for which the differential equation is given by:

$$dS_t = r(t)S_t dt + \sigma S_t dW_t, \quad S_0 = s$$

To simulate a path of S under this model, the SDE can be approximated by the discrete time evolution

$$S_{t+\Delta t} = S_t + r(t)S_t\Delta t + \sigma S_t Z(t)\sqrt{\Delta t} \quad S(0) = s,$$

with $Z(t) \sim N(0,1)$, which is the standard normal distribution, $Z(i)$ and $Z(j)$ are independent if $i \neq j$ and $n\Delta t = T$, so n is the total number of time steps. This approximation can be given since $W_t - W_s \sim N(0, t-s)$, so $W_{t+\Delta t} - W_t \sim \sqrt{\Delta t}N(0,1)$.
Applying the Monte Carlo method to this model yields sampling n random values of a standard normal distribution, one for each period of time $(i\Delta t, (i+1)\Delta t]$ to get one value of S_T. A call option gives a payoff of $\max(S_T - K, 0) = (S_T - K)^+$ and a put option gives a payoff of $(K - S_T)^+$ at the end of the life time of a European option. Sample in total m different paths for S to get m possible payoffs of the option and discount each payoff to time zero. The last step of the Monte Carlo method is to take the mean of these m discounted payoffs to get an approximation of the option price.
An approximation of a confidence interval can also be calculated. Set μ_m the mean and σ_m the estimated standard deviation of these m option payoffs. If a 95% confidence interval for the option price V is wanted, then an approximation of this interval is given by:

$$\mu_m - \mathcal{N}^{-1}(1-0.05/2)\frac{\sigma_m}{\sqrt{m}} < V < \mu_m + \mathcal{N}^{-1}(1-0.05/2)\frac{\sigma_m}{\sqrt{m}}, \tag{2.4}$$

with $\mathcal{N}^{-1}$ the inverse of the standard normal cumulative distribution function.
From this formula it can be seen that quadrupling the number of paths simulated approximately doubles the accuracy.

Application to the SABR model

The Monte Carlo method can also be used to derive prices of European options under the SABR model. This goes in an analogous manner as for the Black-Scholes model. To simulate a path of S, the life of the option can be divided into n short time intervals Δt, and the equations of the SABR model without dividends, 1.6, can be approximated by

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k D(k\Delta t, T)^{1-\beta} S_k^{\beta}\epsilon_1(k\Delta t)\sqrt{\Delta t}, \quad S_0 = s$$

$$\alpha_{k+1} = \alpha_k + \nu\alpha_k\ (\rho\epsilon_1(k\Delta t) + \epsilon_2(k\Delta t)\sqrt{1-\rho^2})\sqrt{\Delta t}, \quad \alpha_0 = \alpha \qquad ,$$

$$k = 0, 1, ..., n-1$$

where S_i and α_i are the values of S and α at time $i\Delta t$, the total number of time steps is n, and the expression $(\rho\epsilon_1(k\Delta t)+\epsilon_2(k\Delta t)\sqrt{1-\rho^2})$ appears, because the two Brownian motions in the SABR model are correlated with a factor ρ. In this notation $\epsilon_1(k\Delta t)$ and $\epsilon_2(k\Delta t)$ are two independent random samples from a standard normal distribution, both independent of each other and independent with respect to different values of k. Of course $(\rho\epsilon_1(k\Delta t)+\epsilon_2(k\Delta t)\sqrt{1-\rho^2})$ has correlation ρ with $\epsilon_1(k\Delta t)$ and it is also a sample of the standard normal distribution

In case the underlying pays dividend, the only change is the SDE of the underlying, and thus the approximation of this SDE.

In case the underlying is assumed to pay dividend yield, the SDE of the underlying can be approximated by:

$$S_{k+1} = S_k + (r(k\Delta t) - q)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T}(r(s)-q)ds} S_k^{\beta}\epsilon_1(k\Delta t)\sqrt{\Delta t}, \quad S(0) = s$$

The changes that should be made to the approximation of the path of the underlying, in case the underlying pays discrete dividends, are somewhat more complicated.

Firstly, the last time points before the dividend payments should be calculated. This means that for dividend payment i, the maximum value of $m_i \in \mathbb{N}$ should be found such that $\frac{m_i T}{n} < t_i$, where t_i is the time at which dividend i is payed, and n is the total number of time steps between zero and T. After that, the discrete time approximation of the stochastic differential equation for S_t is split into different parts, and in this approximation it is assumed that the dividend time points are given by $(m_i+1)\Delta t$, for $i = 1, ..., j$

In case the underlying is assumed to pay discrete proportional dividends, the path of the underlying can be approximated by:

For $0 \leq k \leq m_1 - 1$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds}(1-q)^{(1-\beta)(-j)} S_k^{\beta}\epsilon_1(k\Delta t)\sqrt{\Delta t},$$

For $i = 1, ..., j$:

$$S_{m_i+1} = (1-q)S_{(m_i+1)^-} =$$

$$(1-q)(S_{m_i} + r(m_i\Delta t)S_{m_i}\Delta t + \alpha_{m_i} e^{-(1-\beta)\int_{m_i\Delta t}^{T} r(s)ds}(1-q)^{(1-\beta)(i-1-j)} S_{m_i}^{\beta}\epsilon_1(m_i\Delta t)\sqrt{\Delta t}),$$

where $(m_i+1)^-$ means just before dividend i is paid.

For $m_i + 1 \leq k \leq m_{i+1} - 1, \quad i = 1, ..., j$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds}(1-q)^{(1-\beta)(i-j)} S_k^{\beta}\epsilon_1(k\Delta t)\sqrt{\Delta t}$$

Finally for $m_j + 1 \leq k < n$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds} S_k^{\beta}\epsilon_1(k\Delta t)\sqrt{\Delta t}$$

In case the underlying is assumed to pay fixed cash dividends, the path of the underlying can be approximated by:

For $0 \leq k \leq m_1 - 1$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-\int_{k\Delta t}^{T} r(s)ds}$$

$$\max(e^{\int_{k\Delta t}^{T} r(s)ds} S_k - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i > k\Delta t}, 0)^{\beta}\epsilon_1(k\Delta t)\sqrt{\Delta t},$$

For $i = 1, ..., j$:

$$S_{m_i+1} = \max(0, S_{(m_i+1)^-} - c_i) = \max(0, S_{m_i} + r(m_i\Delta t)S_{m_i}\Delta t + \alpha_{m_i} e^{-\int_{m_i\Delta t}^{T} r(s)ds}$$

$$\max(e^{\int_{m_i\Delta t}^{T} r(s)ds} S_{m_i} - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i > m_i\Delta t}, 0)^{\beta} \epsilon_1(m_i\Delta t)\sqrt{\Delta t} - c_i),$$

where $(m_i + 1)^-$ means just before dividend i is paid.
For $m_i + 1 \leq k \leq m_{i+1} - 1, \quad i = 1, ..., j$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-\int_{k\Delta t}^{T} r(s)ds} \max(e^{\int_{k\Delta t}^{T} r(s)ds} S_k - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i > k\Delta t}, 0)^{\beta} \epsilon_1(k\Delta t)\sqrt{\Delta t}$$

Finally for $m_j + 1 \leq k < n$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds} S_k^{\beta} \epsilon_1(k\Delta t)\sqrt{\Delta t}$$

To get an approximation of the price of a European option under the SABR model with the Monte Carlo method, one should now follow almost the same steps as in the example. The only difference is that there should be sampled $2n$ instead of n values of a standard normal distribution to get one value of S_T.

2.3 Simulating with trees

After the publication of an important paper by Cox, Ross and Rubinstein [4], tree-methods became widely used to price all different kinds of options. The principles behind binomial and trinomial trees, and the numerical procedures involved, can be found in several publications [12]. Tree methods are very easy to apply to one-factor models, so although the SABR model is a two-factor model, deriving such a method to price European options under the SABR model seems to be a logical step. The application to stochastic volatility driven models however, is far more complicated, not in the least sense because the computational time has to be kept within bounds, while normally the tree that is built is non-recombining, which means that different nodes at one time point do not have successor nodes in common. If branches of the tree do not recombine, then the number of nodes in the tree grows exponentially with the number of time steps taken. Different articles have been written about this subject, among which articles by Florescu and Viens [7, 8], Leisen [15], Kargin [14], Vellekoop and Nieuwenhuis [21] and Hilliard and Schwartz [11]. Florescu and Viens show how to estimate the distribution of the volatility component. They use this to construct a binomial tree and they sample, using the Monte Carlo method, from this tree to obtain a smaller recombining tree. Leisen constructs a sequence of discrete-time models, that converges to stochastic volatility models in continuous time. Kargin uses modern methods of adaptive interpolation. Vellekoop and Nieuwenhuis define a fixed grid at each time point and they use linear (or cubic) interpolation to obtain the expected value at each grid point. Finally, Hilliard and Schwartz give a method to transform the stochastic differential equations such that they have unit volatility. This last method seemed to be the simplest approach which was applicable to the SABR model, that is why the first attempt to find a tree-method for the SABR model was based on this article.

2.3.1 The Hilliard and Schwartz tree-method

The principle idea of the article written by Hilliard and Schwartz [11] is that they transform both processes in such a way that the new processes have unit volatility. This gives a solution to the problems that exist when trying to apply a tree-method to a stochastic volatility model, since from a model with unit volatility a recombining tree can easily be built.

The model that is discussed by Hilliard and Schwartz [11] is the following:

$$\begin{aligned} dS_t &= m_t^S dt + \sqrt{V_t} S_t^{\beta} dW_t^1 \\ dV_t &= m_t^V dt + bV_t dW_t^2, \end{aligned} \tag{2.5}$$

under the risk-neutral measure, where $d\left\langle W_t^1, W_t^2 \right\rangle = \rho \; dt$, $m_t^S = rS_t$, $\beta \in [0,1]$ is the elasticity coefficient and m_t^V is the drift of V_t.
By applying the Itô formula (1.1), it can be seen that the transformation $Y = \log(V)/b$ gives a process with unit volatility:

$$dY_t = (\frac{m_t^V}{bV_t} - \frac{1}{2}b)dt + dW_t^2$$

For the transformation of S_t to constant volatility, a two-step transformation is used. First a function $H(S,V)$ is defined, with a diffusion given by:

$$dH_t = H_t^S S_t^{\beta} \sqrt{V_t} dW_t^1 + H_t^V bV_t dW_t^2 + m_t^h dt$$

where H_t^S and H_t^V denote the partial derivatives and m_t^h is the drift of H_t which depends on m_t^S, m_t^V and second-order partial derivatives.
A second transformation of H to Q is used to give a diffusion of the form:

$$dQ_t = m_t^q dt + dW_t^3$$

Explicit formulas for m_t^h and m_t^q are given in the article by Hilliard and Schwartz [11].
The binomial tree method can be applied to (Y,Q), and the corresponding values of V and S are given by the inverse transforms:

$$V = e^{\nu Y}$$
$$H = \frac{2\rho - (1-\rho^2)e^{-\frac{1}{2}bQ} + e^{\frac{1}{2}bQ}}{b}$$
$$S = \begin{cases} (\sqrt{V}(1-\beta)H)^{\beta-1} & \text{if} \quad \beta \neq 1 \\ e^{H\sqrt{V}} & \text{if} \quad \beta = 1 \end{cases}$$

A description of the construction of the tree with, explicit formula's for the jump probabilities, can be found in the article by Hilliard and Schwartz [11].

Application to the SABR model

To apply the method of Hilliard and Schwartz [11], to the SABR model (without dividends), the SABR model has to be rewritten in the form of equations (2.5), which means that $\alpha_t D(t,T)^{1-\beta} = \sqrt{V_t}$ (see equation 1.6). With the Itô formula (1.1) applied to $f(\alpha_t, t) = V_t = \alpha_t^2 D(t,T)^{2(1-\beta)}$, where $a(\alpha_t, t) = 0$, $b(\alpha_t, t) = \nu\alpha_t$, and the partial derivatives of f are given by:

$$\frac{\partial f}{\partial t} = \alpha_t^2 \frac{d}{dt}(e^{-2(1-\beta)\int_t^T r(s)ds}) = \alpha_t^2 2(1-\beta)r(t)D(t,T)^{2(1-\beta)} = 2(1-\beta)r(t)V_t$$
$$\frac{\partial f}{\partial \alpha_t} = 2\alpha_t D(t,T)^{2(1-\beta)} = 2\frac{V_t}{\alpha_t}, \quad \frac{\partial^2 f}{\partial \alpha_t^2} = 2D(t,T)^{2(1-\beta)} = 2\frac{V_t}{\alpha_t^2},$$

the following form of the SABR model is derived:

$$\begin{aligned} dS_t &= r(t)S_t dt + S^{\beta}\sqrt{V_t} dW_t^1 \\ dV_t &= (2r(t)(1-\beta)V_t + 0 + \tfrac{1}{2}\nu^2\alpha_t^2 2\tfrac{V_t}{\alpha_t^2})dt + \nu\alpha_t 2\tfrac{V_t}{\alpha_t} dW_t^2 = (2r(t)(1-\beta) + \nu^2)V_t dt + 2\nu V_t dW_t^2 \end{aligned} \tag{2.6}$$

Now the binomial tree method of Hilliard and Schwartz can be applied to the SABR model, with $m_t^V = (2r(t)(1-\beta) + \nu^2)V_t$ and $b = 2\nu$. The main problem that came at light after programming this method, is that the joint probabilities can be negative. After fixing $\beta = 1$, restricting S and restricting m_t^q in the interval $[\frac{-1}{\sqrt{(\Delta t)}}, \frac{1}{\sqrt{(\Delta t)}}]$, such that the joint probabilities were in $[0,1]$, the numbers of the example in the article by Hilliard and Schwartz were obtained, but this was already with too many restrictions. Furthermore, the option prices given by the tree when T or ν were high or $\beta \neq 1$ were incorrect. These arguments together lead to the conclusion that this tree-method was not the method that should be used to price options under the SABR model.

2.3.2 The Vellekoop and Nieuwenhuis tree-method

In the paper by Vellekoop and Nieuwenhuis [21], the problem of exponentially growing points in the tree is solved by working with a fixed grid at each time point, which changes every time step, with a finer mesh for ΔS and $\Delta\alpha$ than the usual square root of Δt. When a value is needed in a point which is not on a grid point, which will usually be the case, interpolation is used.
In detail and with formulas, the method can be described as follows:
Define the following discrete time stochastic processes for a general stochastic volatility model:

$$\begin{aligned} V_{k+1}^n &= V_k^n + f_V(S_k^n, V_k^n)\Delta t^n + g_V(S_k^n, V_k^n)Y_{k+1}^{n,2}\sqrt{\Delta t^n}, \quad V_0^n = V_0 \\ S_{k+1}^n &= S_k^n + f_S(S_k^n, V_k^n)\Delta t^n + g_S(S_k^n, V_k^n)Y_{k+1}^{n,1}\sqrt{\Delta t^n}, S_0^n = S_0, \end{aligned} \tag{2.7}$$

where $k = 0, 1, ..., n-1$, $\Delta t^n = T/n$, V_i and S_i are the stochastic processes V and S at time $i\Delta t$, f_V and f_S are the drift terms of V respectively S and g_V and g_S are the diffusion terms of V respectively S. [21].
The variables $(Y_k^{n,1}, Y_k^{n,2})$ are i.i.d. distributed in k, with the following probabilities, under the risk neutral pricing measure $\mathbb{Q}^n$:

$$\begin{aligned} \mathbb{Q}^n(Y_k^{n,1} = +1, Y_k^{n,2} = +1) &= \frac{1}{4}(1+\rho) \\ \mathbb{Q}^n(Y_k^{n,1} = -1, Y_k^{n,2} = +1) &= \frac{1}{4}(1-\rho) \\ \mathbb{Q}^n(Y_k^{n,1} = +1, Y_k^{n,2} = -1) &= \frac{1}{4}(1-\rho) \\ \mathbb{Q}^n(Y_k^{n,1} = -1, Y_k^{n,2} = -1) &= \frac{1}{4}(1+\rho), \end{aligned}$$

which is the most natural discrete time stochastic process to approximate equations (2.7) in continuous time, but it defines a non-recombining tree [21].
Define

$$s_k^{n,\max} = \max\left(s : \mathbb{Q}^n(S_k^n = s) > 0\right)$$

and $s_k^{n,\min}$, $v_k^{n,\max}$ and $v_k^{n,\min}$ analogously. Define

$$\begin{aligned} \Delta s_k^n &= (s_k^{n,\max} - s_k^{n,\min})/(m_s - 1) \\ \Delta v_k^n &= (v_k^{n,\max} - v_k^{n,\min})/(m_v - 1), \end{aligned}$$

for m_s, $m_v \in \mathbb{N}^+$, which describe how fine the taken mesh is. With these mesh sizes, the set of grid points at each time point k is defined an can be written as follows:

$$G_k^n = \{(v_k^{n,\min} + i\Delta v_k^n, s_k^{n,\min} + j\Delta s_k^n) | i = 0, ..., m_v - 1, \; j = 0, ..., m_s - 1\}$$

If f is a function from the grid points to $\mathbb{R}$, than the piecewise bilinear interpolating function corresponding to this function on the grid can be denoted by $\mathcal{L}^{G_k^n}[f] : \mathbb{R}^2 \to \mathbb{R}$ [21], so

$$\begin{aligned} \mathcal{L}^{G_k^n}[f](x,y) = c_{00}^{G_k^n}(x,y) f(x_0^{G_k^n}(x), y_0^{G_k^n}(y)) + c_{10}^{G_k^n}(x,y) f(x_1^{G_k^n}(x), y_0^{G_k^n}(y)) + \\ c_{01}^{G_k^n}(x,y) f(x_0^{G_k^n}(x), y_1^{G_k^n}(y)) + c_{11}^{G_k^n}(x,y) f(x_1^{G_k^n}(x), y_1^{G_k^n}(y)), \end{aligned}$$

where

$$(x_0^{G_k^n}(x), y_0^{G_k^n}(y)) \in G_k^n, \quad (x_1^{G_k^n}(x), y_1^{G_k^n}(y)) \in G_k^n, \quad x_0^{G_k^n}(x) \le x \le x_1^{G_k^n}(x), \quad y_0^{G_k^n}(y) \le y \le y_1^{G_k^n}(y),$$

$$\tilde{x} = \frac{x - x_0^{G_k^n}(x)}{x_1^{G_k^n}(x) - x_0^{G_k^n}(x)}, \quad \tilde{y} = \frac{y - y_0^{G_k^n}(y)}{y_1^{G_k^n}(y) - y_0^{G_k^n}(y)}$$

$$c_{00}^{G_k^n}(x,y) = (1-\tilde{x})(1-\tilde{y}), \quad c_{10}^{G_k^n}(x,y) = \tilde{x}(1-\tilde{y}), \quad c_{01}^{G_k^n}(x,y) = (1-\tilde{x})\tilde{y}, \quad c_{11}^{G_k^n}(x,y) = \tilde{x}\tilde{y}.$$

Now a new process $(\tilde{V}^n, \tilde{S}^n)$ can be defined, for which all possible values are in G_k^n:

$$(\tilde{V}_{k+1}^n, \tilde{S}_{k+1}^n) = (x_{y_{k+1}^{n,3}}^{G_{k+1}^n}(v^n(Y_{k+1}^{n,1}, \tilde{V}_k^n, \tilde{S}_k^n)), y_{Y_{k+1}^{n,4}}^{G_{k+1}^n}(s^n(Y_{k+1}^{n,2}, \tilde{V}_k^n, \tilde{S}_k^n))),$$

where

$$Y_{k+1}^{n,3},\ Y_{k+1}^{n,4} \in \{0,1\}, \quad v^n(Y_{k+1}^{n,1}, \tilde{V}_k^n, \tilde{S}_k^n) = \tilde{V}_k^n + f_V(\tilde{S}_k^n, \tilde{V}_k^n)\Delta t + g_V(\tilde{S}_k^n, \tilde{V}_k^n)Y_{k+1}^{n,1}\sqrt{\Delta t},$$

$$s^n(Y_{k+1}^{n,2}, \tilde{V}_k^n, \tilde{S}_k^n) = \tilde{S}_k^n + f_S(\tilde{S}_k^n, \tilde{V}_k^n)\Delta t + g_S(\tilde{S}_k^n, \tilde{V}_k^n)Y_{k+1}^{n,2}\sqrt{\Delta t},$$

and the conditional distribution of $(Y^{n,3}, Y^{n,4})$ is given by:

$$\mathbb{Q}^n(Y_{k+1}^{n,3} = i, Y_{k+1}^{n,4} = j | (Y_{k+1}^{n,1}, Y_{k+1}^{n,2}, \tilde{V}_k^n, \tilde{S}_k^n)) =$$

$$c_{ij}^{G_{k+1}^n}(v^n(Y_{k+1}^{n,1}, \tilde{V}_k^n, \tilde{S}_k^n), s^n(Y_{k+1}^{n,2}, \tilde{V}_k^n, \tilde{S}_k^n)) \qquad i,j \in \{0,1\}$$

Since every current value (V_k^n, S_k^n) has four successor nodes, and each of these successor nodes is split in this method in four new points, there are in total sixteen new possible values for $(\tilde{V}_{k+1}^n, \tilde{S}_{k+1}^n)$, based on the current value $(\tilde{V}_k^n, \tilde{S}_k^n)$. This can also be seen from the fact that $Y_{k+1}^{n,1}, Y_{k+1}^{n,2}, Y_{k+1}^{n,3}$ and $Y_{k+1}^{n,4}$ are the only variables one has to know, when knowing the current value $(\tilde{V}_k^n, \tilde{S}_k^n)$, to derive the new value $(\tilde{V}_{k+1}^n, \tilde{S}_{k+1}^n)$. These Y's can all have two different values, so in total there are 2^4 possible new values. By the following mathematical definition for conditional probabilities:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)},$$

the probabilities for each of the sixteen successor nodes are given by:

$$\mathbb{Q}^n(Y_{k+1}^{n,1} = i_1, Y_{k+1}^{n,2} = i_2, Y_{k+1}^{n,3} = j_1, Y_{k+1}^{n,4} = j_2 | (\tilde{V}_k^n, \tilde{S}_k^n)) = c_{j_1 j_2}^{G_{k+1}^n}(v^n(i_1, \tilde{V}_k^n, \tilde{S}_k^n), s^n(i_2, \tilde{V}_k^n, \tilde{S}_k^n))\frac{1 + i_1 i_2 \rho}{4}$$

where $i_1, i_2 \in \{-1, 1\}$ and $j_1, j_2 \in \{0, 1\}$.

The article by Vellekoop and Nieuwenhuis [21], is written for the Heston model and uses the logarithm of the stock price process, because this gives an easier diffusion function. For the SABR model, using the logarithm does not simplify the diffusion function, so in this thesis the method described is applied to the SABR model in its original form.

Application to the SABR model

In this section it is explained how this method can be applied to the SABR model. The following discrete time stochastic processes, to approximate the SABR model when the underlying does not pay dividend, were defined:

$$\alpha_{k+1}^n = \alpha_k^n + \nu \alpha_k^n Y_{k+1}^{n,1}\sqrt{\Delta t}, \quad \alpha_0^n = \alpha_0$$

$$S_{k+1}^n = S_k^n + r(k\Delta t)S_k^n \Delta t + \alpha_k^n(k\Delta t, T)^{1-\beta}(S_k^n)^\beta Y_{k+1}^{n,2}\sqrt{\Delta t}, \quad S_0^n = S_0$$

$$k = 0, 1, ..., n-1$$

In case the underlying pays dividend, the only change is again the SDE of the underlying, and thus the approximation of this SDE.
In case the underlying is assumed to pay dividend yield, the SDE of the underlying can be approximated by:

$$S_{k+1} = S_k + (r(k\Delta t) - q)S_k \Delta t + \alpha(k)e^{-(1-\beta)\int_{k\Delta t}^T (r(s)-q)ds} S_k^\beta Y_{k+1}^{n,2}\sqrt{\Delta t}, \quad S(0) = s$$

The changes that should be made to the approximation of the path of the underlying in case the underlying pays discrete dividends, are as described in section 2.2, but are repeated here.
Firstly, the last time points before the dividend payments should be calculated. This means that for dividend payment i, the maximum value of $m_i \in \mathbb{N}$ should be found such that $\frac{m_i T}{n} < t_i$, where t_i is the time at which dividend i is payed, and n is the total number of time points between zero and T. After that, the discrete time approximation of the stochastic differential equation for S_t is split into different parts and in this approximation it is assumed that the dividend time points are given by $(m_i + 1)\Delta t$, for $i = 1, ..., j$.

In case the underlying is assumed to pay discrete proportional dividends, the path of the underlying can be approximated by:
For $0 \leq k \leq m_1 - 1$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds}(1-q)^{(1-\beta)(-j)} S_k^{\beta} Y_{k+1}^{n,2}\sqrt{\Delta t},$$

For $i = 1, ..., j$:

$$S_{m_i+1} = (1-q)S_{(m_i+1)^-} = (1-q)(S_{m_i} + r(m_i)S_{m_i}\Delta t + \alpha_{m_i} e^{-(1-\beta)\int_{m_i\Delta t}^{T} r(s)ds}(1-q)^{(1-\beta)(i-1-j)} S_{m_i}^{\beta} Y_{m_i+1}^{n,2}\sqrt{\Delta t}),$$

where $(m_i + 1)^-$ means just before dividend i is paid.
For $m_i + 1 \leq k \leq m_{i+1} - 1, \quad i = 1, ..., j$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds}(1-q)^{(1-\beta)(i-j)} S_k^{\beta} Y_{k+1}^{n,2}\sqrt{\Delta t}$$

Finally for $m_j + 1 \leq k < n$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds} S_k^{\beta} Y_{k+1}^{n,2}\sqrt{\Delta t}$$

In case the underlying is assumed to pay fixed cash dividends, the path of the underlying can be approximated by:
For $0 \leq k \leq m_1 - 1$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-\int_{k\Delta t}^{T} r(s)ds} \max(e^{\int_{k\Delta t}^{T} r(s)ds} S_k - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i > k\Delta t}, 0)^{\beta} Y_{k+1}^{n,2}\sqrt{\Delta t},$$

For $i = 1, ..., j$:

$$S_{m_i+1} = \max(0, S_{(m_i+1)^-} - c_i) = \max(0, S_{m_i} + r(m_i\Delta t)S_{m_i}\Delta t + \alpha_{m_i} e^{-\int_{m_i\Delta t}^{T} r(s)ds}$$

$$\max(e^{\int_{m_i\Delta t}^{T} r(s)ds} S_{m_i} - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i > m_i\Delta t}, 0)^{\beta} Y_{m_i+1}^{n,2}\sqrt{\Delta t} - c_i),$$

where $(m_i + 1)^-$ means just before dividend i is paid.
For $m_i + 1 \leq k \leq m_{i+1} - 1, \quad i = 1, ..., j$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-\int_{k\Delta t}^{T} r(s)ds} \max(e^{\int_{k\Delta t}^{T} r(s)ds} S_k - \sum_{i=1}^{j} c_i e^{\int_{t_i}^{T} r(s)ds} 1_{t_i > k\Delta t}, 0)^{\beta} Y_{k+1}^{n,2}\sqrt{\Delta t}$$

Finally for $m_j + 1 \leq k < n$:

$$S_{k+1} = S_k + r(k\Delta t)S_k\Delta t + \alpha_k e^{-(1-\beta)\int_{k\Delta t}^{T} r(s)ds} S_k^{\beta} Y_{k+1}^{n,2}\sqrt{\Delta t}$$

The method was firstly applied exactly as written in the paper by Vellekoop and Nieuwenhuis [21], but this appeared to converge too slow for the SABR model, or in other words, it took too long before a good approximation came out of this method. That is why some changes had to be made.
Using cubic interpolation takes more time, but it converges faster [21]. It seems that for the SABR model the differences with prices given by the Monte Carlo method are smaller in the same calculation time. So using cubic interpolation on a less fine grid is the first step of making the method faster.
Furthermore, it appeared that $S_k^{n,\max}$ and $\alpha_k^{n,\max}$ grew exponentially for every time step. This implied that S_k^n and α_k^n should be bounded. Also, it is not very efficient to take the same mesh size of the grid far from the expected value as close to this value. Since making a continuously changing mesh for each value further away from the expected value is time consuming, the total span should be split into pieces and a different mesh size for each of these pieces should be chosen.
The problem is to find a way to make choices based on good grounds for the bounds of α_k^n and S_k^n, and for bounds of the different intervals the total span of the grid should be split into, for each $k \in 0, 1, ..., n$.

If known marginal distribution functions exist for α_k^n and S_k^n, this can be done. For S_k^n one can then talk about the probability that S_k^n will be in a certain interval, and one can calculate the maximal error of the prices when setting a bound on S_k^n. This can be seen as follows: If one wants the maximal pricing error by setting a bound on S_k^n to be 0.002 euro's, the value of y for which the following holds should be calculated: $\int_y^\infty s\phi_k(s)ds < 0.002$, where ϕ_k is the distribution function of S_k^n. For a bound on α_k^n, the maximal error in cents can not be talked of, but one can define, for example, that a maximum probability mass from the bound to infinity of less then 0.01% is wanted, which means that the value of y for which the following holds: $\int_y^\infty \psi_k(a)da < 0.0001$ has to be calculated, where ψ_k is the distribution function of α_k^n.
To define the intervals, percentages can be chosen, from which the intervals can be derived. For example 40% to 60%, which means, in case of the underlying, the interval $[y_1, y_2]$ with $\int_0^{y_1} \phi_k(s)ds = 0.4$ and $\int_0^{y_2} \phi_k(s)ds = 0.6$.

The marginal distribution functions of S and α
Hagan, Lesniewski and Woodward wrote a paper about the marginal probability distribution of F_t in the SABR model [10], but this is still a preprint article. The probability distribution function they derive, in terms of the forward price f, is:

$$\begin{gathered}\tfrac{\partial}{\partial f}Q(f_T \leq f) = P_f(f) = \tfrac{1}{\sqrt{2\pi T}}\tfrac{1}{\alpha K^\beta I^{3/2}}e^{-\frac{D^2}{2T\nu^2}}(1 + \tfrac{\alpha\beta f^{\beta-1}D}{2\nu\sqrt{1-\rho^2 I}} \\ +T\nu^2[-\tfrac{1}{8} + \tfrac{7}{16}\tfrac{1-D\coth(D)}{D}\tfrac{\alpha\beta f^{\beta-1}}{\nu\sqrt{1-\rho^2 I}} + \tfrac{3}{8}\tfrac{1-\rho^2}{I}\tfrac{\sinh(D)}{D}(1+\tfrac{\alpha\beta f^{\beta-1}D}{2\nu\sqrt{1-\rho^2 I}}) + \tfrac{\alpha\beta f^{\beta-1}(\zeta-\rho)}{4\nu\sqrt{1-\rho^2 I}}]),\end{gathered} \tag{2.8}$$

where

$$\zeta = \frac{\nu}{\alpha}\frac{f^{1-\beta} - K^{1-\beta}}{1-\beta}, \quad \text{for} \quad 0 \leq \beta < 1$$
$$\zeta = \frac{\nu}{\alpha}\ln(\frac{f}{K}), \quad \text{for} \quad \beta = 1$$
$$I = \sqrt{\zeta^2 - 2\rho\zeta + 1}$$
$$D = \ln(\frac{I + \zeta - \rho}{1 - \rho})$$

The curious phenomenon of this formula is that the strike price K appears as a parameter in it. This should of course not be the case, since the same underlying will not act differently at the same time for options with different strikes. Furthermore, when you take $\nu = 0$, $\beta = 1$, the SDE's of the SABR model in terms of forward prices, equation (1.5), become equal to:

$$dF_t = \alpha_t F_t dW_t, \quad d\alpha_t = 0,$$

so α_t is constant and equal to $\alpha = \alpha_0$.
Since it follows now, by applying the Itô formula (1.1) to $\ln(F_t)$, that

$$d\ln(F_t) = (0 + 0 + \frac{1}{2}\alpha^2 F_t^2 \frac{-1}{F_t^2})dt + \alpha F_t \frac{1}{F_t}dW_t = -\frac{\alpha^2}{2}dt + \alpha dW_t,$$

and W_t is standard normally distributed, it can be seen that F_t has a log-normal distribution.
When $\nu \to 0$ and $\beta \to 1$ the marginal distribution function of F_T should therefore be:

$$P_f(f) = \frac{1}{f\alpha\sqrt{2\pi T}}e^{-\frac{(\ln(f)+(\alpha\sqrt{T})/2)^2}{2\alpha^2 T}},$$

which is the distribution function of the Black-Scholes model in terms of the forward price. But the distribution function given by Hagen et al. [10], given these parameters is:

$$P_f(f) = \frac{1}{K\alpha\sqrt{2\pi T}}e^{-\frac{\ln(\frac{f}{K})^2}{2\alpha^2 T}}(1 + \frac{\ln(\frac{f}{K})}{2\sqrt{1-\rho^2}}) \tag{2.9}$$

This follows by the following facts:

$$\zeta \to 0, \quad I \to 1, \quad \zeta/\nu = \ln(\frac{f}{K})/\alpha,$$

with first order Taylor expansion on $\sqrt{\zeta^2 - 2\rho\zeta + 1}$ and $\ln(1+x)$ it follows that:

$$\frac{D}{\zeta} \to \frac{\ln(1+\zeta)}{\zeta} \to 1$$

So the distribution function given by Hagan et al. [10] is not correct, and another way of finding this distribution function should be thought of.
Since the price of a European option is given by equation (1.2), and the pay-off of a call option at maturity is $(S_T - K)^+$, it follows that

$$V_{call}(0) = D(0,T) \int_K^\infty (s-K)\phi(s)ds,$$

where $\phi(s)$ is the distribution function of $S_T = F_T$. By taking the first derivative with respect to the strike price K it follows that

$$\frac{\partial V_{call}(0)}{\partial K} = D(0,T)(-K\phi(K) - \int_K^\infty \phi(s)ds + K\phi(K)) = -D(0,T)\int_K^\infty \phi(s)ds$$

and from that the following can be derived:

$$\frac{\partial^2 V_{call}(0)}{\partial K^2} = D(0,T)\phi(K)$$

The idea now, is that, since there is an approximating direct formula for the price of a call option, we can take the second derivative to K of this formula and derive an approximation of the distribution function of S_T. The approximating direct formula for the price of a European call option is:

$$V_{call}(0) = D(0,T)(F_0\mathcal{N}(d_1) - K\mathcal{N}(d_2))$$

with

$$d_{1,2} = \frac{\ln(\frac{F_0}{K}) \pm \frac{1}{2}\sigma_B^2 T}{\sigma_B\sqrt{T}},$$

where σ_B is given by equation 2.3.
The calculation of the first and second partial derivative of σ_B with respect to K is done by Matlab. The rest of the derivation goes as follows:

$$\mathcal{N}'(x) = \frac{1}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}, \quad \frac{\partial d_{1,2}}{\partial \sigma_B} = \frac{-\ln(\frac{F_0}{K})}{\sigma_B^2\sqrt{T}} \pm \frac{\sqrt{T}}{2}, \quad \frac{\partial d_{1,2}}{\partial K} = \frac{-1}{K\sqrt{T}\sigma_B} + \frac{\partial d_1}{\partial \sigma_B}\frac{\partial \sigma_B}{\partial K}$$

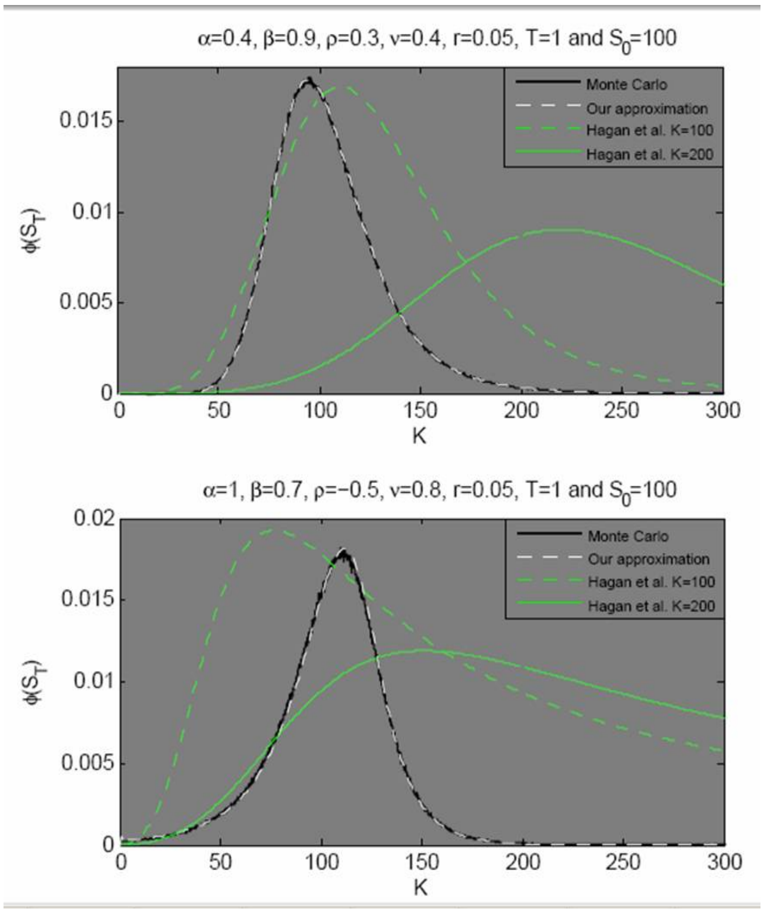
$$\frac{1}{D(0,T)}\frac{\partial V_{call}}{\partial K} = F_0\mathcal{N}'(d_1)\frac{\partial d_1}{\partial K} - \mathcal{N}(d_2) - K\mathcal{N}'(d_2)\frac{\partial d_2}{\partial K}$$

$$\mathcal{N}''(x) = \frac{-x}{\sqrt{2\pi}}e^{\frac{-x^2}{2}}, \quad \frac{\partial^2 d_{1,2}}{\partial K^2} = \frac{1}{K^2\sqrt{T}\sigma_B} + \frac{2}{K\sqrt{T}\sigma_B^2}\frac{\partial \sigma_B}{\partial K} + \frac{2\ln(\frac{F_0}{K})}{\sigma_B^3\sqrt{T}}(\frac{\partial \sigma_B}{\partial K})^2 + \frac{\partial d_{1,2}}{\partial \sigma B}\frac{\partial^2 \sigma_B}{\partial K^2}$$

$$\phi(K) = F_0(\mathcal{N}''(d_1)(\frac{\partial d_1}{\partial K})^2 + \mathcal{N}'(d_1)\frac{\partial^2 d_1}{\partial K^2}) - 2\mathcal{N}'(d_2)\frac{\partial d_2}{\partial K} - K(\mathcal{N}''(d_2)(\frac{\partial d_2}{\partial K})^2 + \mathcal{N}'(d_2)\frac{\partial^2 d_2}{\partial K^2})$$

To check whether this distribution function is a good approximation of the real distribution, the graph of this function is plotted for different parameters together with a scaled histogram of 4 million paths simulated by the Monte Carlo method. These graphs, together with the chosen parameters and plots of the distribution function described in the article by Hagan et al. [10], can be found in figure (2.1). Note that the distribution functions are in terms of the forward price, so they do not change in the case the underlying pays any kind of dividend.

Figure 2.1: Examples of the distribution of the underlying

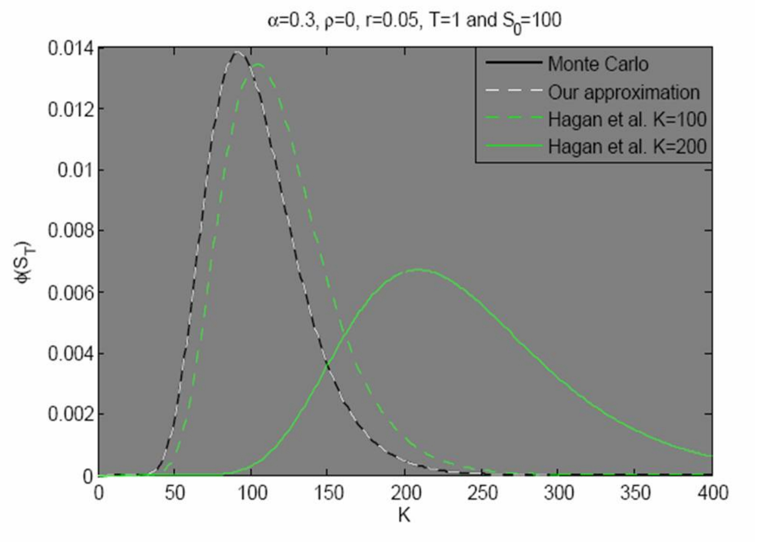


It can be seen from these graphs that, as was already expected, the formula given by Hagan et al. is incorrect, and the formula derived by taking the second derivative to K is very close to the distribution given by the histogram. It can even hardly be seen, since it almost falls together with the distribution given by Monte Carlo simulations.
To give a last argument that the distribution formula of the paper written by Hagan et al. [10], is incorrect, and the formula derived in this thesis is a good approximation of the distribution function, the distribution function for the Black-Scholes model is plotted in figure (2.2) together with the mentioned other two distribution functions with $\beta \to 1$ and $\nu \to 0$. Once more it is seen that the formula derived in this thesis, for $\beta \to 1$ and $\nu \to 0$, is equal to the Black-Scholes distribution function and the distribution function derived by Hagan et al. [10] is not equal to this distribution function.
Since the Stochastic Differential equation of α_t does not depend on the underlying, and the correlation between the Brownian motions is already taken into account in the distribution function of S_t, it can be seen that a good approximation of the distribution of α_t is the log-normal distribution.

Defining the grid

First the upper bounds $S_k^{n,\text{bound}}$ and $\alpha_k^{n,\text{bound}}$ have to be calculated. This can be done exactly as written before, since there are now approximating direct formulas for the distribution functions. Notice that, for the definition of the distribution functions of S_k^n and α_k^n, we should take $k\Delta t$ as the time to maturity, and not T. Since the value of F_0 also depends on the value of the maturity time, this implies that the value of F_0 is different for every value of k. The maximal pricing error that can be obtained by fixing an upper bound on S_k^n is chosen to be 0.002 euro's and for α_k^n the maximum probability mass from the bound to infinity is set 0.01%.

Figure 2.2: SABR distributions with $\beta \to 1$ and $\nu \to 0$



After this, the discrete time stochastic processes have to be redefined:

$$\alpha_{k+1}^{n} = \min(\alpha_k^n + \nu \alpha_k^n Y_{k+1}^{n,1} \sqrt{\Delta t}, \alpha_{k+1}^{n,\text{bound}}), \quad \alpha_0^n = \alpha_0,$$

and the same for S_{k+1}^n, with $S_{k+1}^{n,\text{bound}}$ instead of $\alpha_{k+1}^{n,\text{bound}}$. Now the new values for $S_k^{n,\max}$, $S_k^{n,\min}$, $\alpha_k^{n,\max}$ and $\alpha_k^{n,\min}$ can be derived as before.
The following step is to define the boundaries of the different intervals, together with the different mesh sizes in these intervals. For α_k^n the following boundaries were chosen, where $g_k(x) = \min(\max(\psi_k^{-1}(x), \alpha_k^{n,\min}), \alpha_k^{n,\max})$, and $\psi_k^{-1}(x)$ gives the value y for which $P(\alpha_k^n \leq y) = x$:

$$\alpha_k^{n,\min}, \quad g_k(0.02), \quad g_k(0.1), \quad g_k(0.25), \quad g_k(0.4), \quad g_k(0.6), \quad g_k(0.75), \quad g_k(0.9), \quad g_k(0.98), \quad \alpha_k^{n,\max}$$

The boundaries for S_k^n were chosen in the same way, except that the last interval is split into 3 different intervals, namely:

$$[h_k(0.98), h_k(0.99)], \quad [h_k(0.99), h_k(0.998)], \quad [h_k(0.998), S_k^{n,\max}],$$

where $h_k(x) = \min(\max(\phi_k^{-1}(x), S_k^{n,\min}), S_k^{n,\max})$, and $\phi_k^{-1}(x)$ gives the value y for which $P(S_k^n \leq y) = x$. Since calculating these boundaries and intervals is very time-consuming, while this is meant to decrease the calculation time and only a good guess of these values is wanted, we calculate these values only at maturity T and at $T/2$, and make an approximation based on these values for all other time points. For the upper bounds, an approximation is wanted which is not smaller than the value which would be derived by direct calculation. As was mentioned before, $S_k^{n,\max}$ and $\alpha_k^{n,\max}$ grow exponentially for every time step, when there are no upper bounds and the underlying does not pay discrete dividends. The following reasoning is therefore only valid in the case that the underlying does not pay discrete dividends.
A function that increases exponentially is convex, which means that when one draws a line between two points on the graph of the function, this line will totally lie above the graph of the function. That is why the values for the upper bounds of α and S at all time points between $T/2$ and T are chosen to be on the lines (seen as functions of time) from $\alpha_{n/2}^{\text{bound}}$ to α_n^{bound} and $S_{n/2}^{\text{bound}}$ to S_n^{bound}. For all time points before $T/2$, the upper bounds are chosen to be equal to $\alpha_{n/2}^{\text{bound}}$ and $S_{n/2}^{\text{bound}}$.
The boundaries of the intervals are also assumed to increase/decrease exponentially, that is why the intervals are approximated in the same way as the upper bounds between $T/2$ and T, i.e. using straight lines. So, if $k < n/2$, $g_k(x)$ is approximated by $\alpha_0 + \frac{k}{n/2}(g_{n/2}(x) - \alpha_0)$, and if $k > n/2$, $g_k(x)$ is approximated by $g_{n/2}(x) + \frac{k-n/2}{n/2}(g_n(x) - g_{n/2}(x))$. The same approximation is used for the intervals of S_k^n.

In case the underlying is assumed to pay discrete dividends, these should be taken into account when defining the upper bounds and boundaries of the intervals between the calculated points. The values for the volatility α_t can be derived with the reasoning as above, but the values for the underlying need some changes. For reasons of readability it is chosen to write "case 1" when the underlying is assumed to pay discrete proportional dividends, and "case 2" when the underlying is assumed to pay discrete fixed cash dividends. For the upper bounds of S_k^n, it is still chosen to have one value for all time points between zero and $T/2$.
Since there are $n(T/2)$ dividend payments up until time $T/2$, and a low value is wanted which is certainly higher than the highest value that would be obtained by calculating the bounds for every time point, the value for the upper bound between zero and $T/2$ is chosen to be $(1-q)^{-n(T/2)}S_{n/2}^{\text{bound}}$ in case 1, and $S_{n/2}^{\text{bound}} + \sum_{i=1}^{n(T/2)} c_i$ in case 2. Since the number of dividend payments between $T/2$ and T is equal to $n(T)-n(T/2) = j_1$, first the value $A = (1-q)^{-j_1}S_n^{\text{bound}}$ in case 1, or $A = S_n^{\text{bound}} + \sum_{i=n(T/2)+1}^{n(T)} c_i$ in case 2, is calculated. It is known that there are $j - j_1$ dividend payments between zero and $T/2$, so m_{j-j_1+1} is the last time point before the first dividend payment after $T/2$. For the purpose of readability, the following rescaling is now used: $m_i = m_{j-j_1+i} - n/2$, where $i = 1, ..., j_1$. From this, the following points are defined to give the values of S_k^{bound}, for $k = n/2, ..., n$, which is further called the vector l.

$$\Delta = \frac{A - S_{n/2}^{\text{bound}}}{n/2}$$

$$l(1:m_1) = [S_{n/2}^{\text{bound}} : \Delta : S_{n/2}^{\text{bound}} + \Delta(m_1 - 1)]$$

For $1 \leq i \leq j_1 - 1$
case 1:

$$l(m_i + 1 : m_{i+1}) = [(l(m_i) + \Delta)(1-q) : \Delta : (l(m_i) + \Delta)(1-q) + \Delta(m_{i+1} - m_i - 1)]$$

case 2:

$$l(m_i + 1 : m_{i+1}) = [l(m_i) + \Delta - c_{j-j_1+i} : \Delta : l(m_i) - c_{j-j_1+i} + \Delta(m_{i+1} - m_i)]$$

and
case 1:

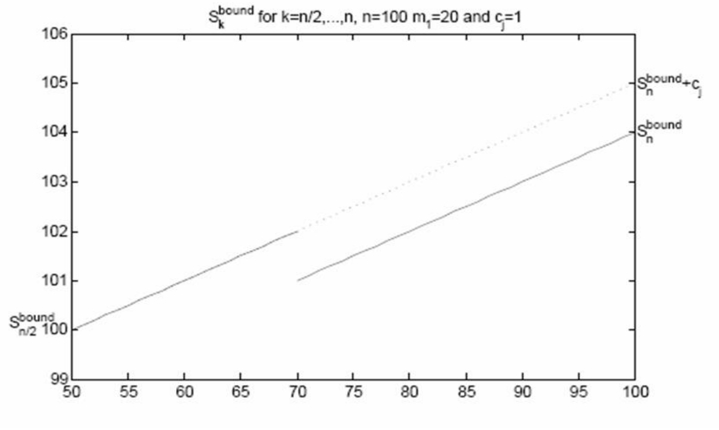
$$l(m_{j_1} + 1 : n/2) = [(l(m_{j_1}) + \Delta)(1-q) : \Delta : (l(m_{j_1}) + \Delta)(1-q) + \Delta(n/2 - m_{j_1} - 1)]$$

case 2:

$$l(m_{j_1} + 1 : n/2) = [l(m_{j_1}) + \Delta - c_j : \Delta : l(m_{j_1}) - c_j + \Delta(n/2 - m_{j_1})]$$

For case 2 we have exactly that $l(n/2) = S_n^{\text{bound}}$. Since $q(l(m_i) + \Delta)$ probably is not equal to qS_n^{bound}, for all $i \in 1, ..., j_1$, the value of $l(n/2)$ will not be exactly equal to S_n^{bound} in case 1, but it will be very close.
To make this procedure more clear, an example is drawn in figure (2.3). In this example there is only one dividend payment between $T/2$ and T.

Figure 2.3: Example of the vector for the bounds of S for $n/2 \leq k \leq n$



Defining the boundary values of the intervals used in the grid between zero and $T/2$ as well as between $T/2$ and T goes analogously. The mesh size is of course chosen to be finest in the intervals $[g_k(0.4), g_k(0.6)]$ and $[h_k(0.4), h_k(0.6)]$. For every interval $[g_k(x), g_k(y)]$ above $g_k(0.6)$ it is chosen that it has the same mesh size as its "opposite" interval $[g_k(1-y), g_k(1-x)]$, and the same again for intervals of S_k^n. Which means that not the number of points in both intervals are equal, but the distance between the grid points is equal on those two intervals. Finally the mesh size for each interval is defined by choosing a total number of grid points in the intervals with the same mesh size. These numbers can be chosen freely (as long as it are strictly positive natural numbers of course). Table 2.1 shows the final definition of the grid, at each point in time, that is used in this thesis.

Table 2.1: Defining the grid used at each point in time

intervals	number of grid points	mesh size
$[g_k(0.4), g_k(0.6)]$	8	$(g_k(0.6) - g_k(0.4))/8$
$[g_k(0.25), g_k(0.4)]$,$[g_k(0.6), g_k(0.75)]$	6	$(g_k(0.75) - g_k(0.6) + g_k(0.4) - g_k(0.25))/6$
$[g_k(0.1), g_k(0.25)]$,$[g_k(0.75), g_k(0.9)]$	4	$(g_k(0.9) - g_k(0.75) + g_k(0.25) - g_k(0.1))/4$
$[g_k(0.02), g_k(0.1)]$,$[g_k(0.9), g_k(0.98)]$	4	$(g_k(0.98) - g_k(0.9) + g_k(0.1) - g_k(0.02))/4$
$[\alpha_k^{n,\min}, g_k(0.02)]$,$[g_k(0.98), \alpha_k^{n,\max}]$	4	$(\alpha_k^{n,\max} - g_k(0.98) + g_k(0.02) - \alpha_k^{n,\min})/4$
$[h_k(0.4), h_k(0.6)]$	20	$(h_k(0.6) - h_k(0.4))/20$
$[h_k(0.25), h_k(0.4)]$,$[h_k(0.6), h_k(0.75)]$	15	$(h_k(0.75) - h_k(0.6) + h_k(0.4) - h_k(0.25))/15$
$[h_k(0.1), h_k(0.25)]$,$[h_k(0.75), h_k(0.9)]$	10	$(h_k(0.9) - h_k(0.75) + h_k(0.25) - h_k(0.1))/10$
$[h_k(0.02), h_k(0.1)]$,$[h_k(0.9), h_k(0.98)]$	10	$(h_k(0.98) - h_k(0.9) + h_k(0.1) - h_k(0.02))/10$
$[S_k^{n,\min}, h_k(0.02)]$,$[h_k(0.98), h_k(0.99)]$	10	$(h_k(0.99) - h_k(0.98) + h_k(0.02) - S_k^{n,\min})/10$
$[h_k(0.99), h_k(0.998)]$	10	$(h_k(0.998) - h_k(0.99))/10$
$[h_k(0.998), S_k^{n,\max}]$	10	$(S_k^{n,\max} - h_k(0.998))/10$

2.4 Comparing the different methods

To compare the methods and to see whether they give the same values, the methods are applied to the same theoretical situations. This means that different theoretical situations are described in which all different parameters are given. The two general situations for which results are given are:
1: European options on an underlying asset which does not pay dividend and the interest rate is constant.
2: European options on an underlying asset which does pay discrete fixed cash dividends and the interest rate depends on time.
The results for situation 1, with different values for the parameters, can be found in table 2.2. The results for situation 2, with different values for the parameters, can be found in table 2.4. To create the results of these tables, the tree method is applied exactly as described before, with the same number of grid points, and the Monte Carlo method is applied with 2 million paths. In both methods the time is divided into 100 steps.
In table 2.3 results are shown for the Monte Carlo method (MC50) when applied with 50 million paths to the first situation. Furthermore, this table shows results for the tree method and the Monte Carlo method applied as described above, with 200 time steps instead of 100 (Tree,n=200 and MC,n=200).
In the second situation, interest rate is time-dependent. In this case, certain interest levels are assumed at certain given time points, and the interest levels in between are derived by linear interpolation. The following data were assumed: $r(0) = 0.0445$, $r(0.25) = 0.0395$, $r(0.5) = 0.0505$, $r(0.75) = 0.0475$ and $r(1) = 0.0425$. Furthermore, the underlying is assumed to pay one dividend per year, with a value of 2.5 euro at $t = \frac{3}{4}$.

Table 2.2: Comparing three different methods for pricing European options under the SABR model with no dividends

The basis situation is $S_0 = 100 = K$, $r = 0.05$, $\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis		Direct	Tree	MC	95% conf.	Direct-Tree	Direct-MC	Tree-MC
-	call	12.4707	12.4804	12.4855	± 0.0291	-0.0097	-0.0148	-0.0051
	put	7.5936	7.6050	7.6038	± 0.0152	-0.0114	-0.0102	0.0012
$\alpha = 25$ &	call	12.1250	12.1249	12.1272	± 0.0237	0.0001	-0.0022	-0.0023
$\beta = 0$	put	7.2480	7.2482	7.2406	± 0.0166	-0.0002	0.0074	0.0076
$\rho = -1$	call	12.4859	12.4964	12.4819	± 0.0200	-0.0105	0.0040	0.0145
	put	7.6088	7.6201	7.6041	± 0.0188	-0.0113	0.0047	0.0160
$\nu = 0.9$	call	12.8807	12.9217	12.9505	± 0.0572	-0.0410	-0.0698	-0.0288
	put	8.0037	8.0459	8.0574	± 0.0174	-0.0422	-0.0537	-0.0115
$K = 120$	call	5.5296	5.5206	5.5034	± 0.0214	0.0090	0.0262	0.0172
	put	19.6771	19.6697	19.6617	± 0.0237	0.0074	0.0154	0.0080
$T = 2.5$	call	21.6784	21.7029	21.7143	± 0.0657	-0.0245	-0.0359	-0.0114
	put	9.9281	9.9605	9.9669	± 0.0207	-0.0324	-0.0388	-0.0064

Table 2.3: Some more results of situation 1

The basis situation is $S_0 = 100 = K$, $r = 0.05$,
$\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis		MC50	95% conf.	Tree,n=200	MC,n=200
-	call	12.4783	± 0.0058	12.4775	12.4906
	put	7.6034	± 0.0030	7.6016	7.5926
$\rho = -1$	call	12.4883	± 0.0040	12.4932	12.4803
	put	7.6131	± 0.0038	7.6144	7.6005

More results of the different methods can be found in appendix A.
The computation time for the given results in tables 2.2 and 2.4, on the computer used[1], is approximately 15 seconds per option for the tree method and approximately 70 seconds for the put and call together, so per single option 35 seconds, for the Monte Carlo method.
From these results above and in appendix A, it can be seen that the option prices given by the tree method are all within the confidence interval of the Monte Carlo method. Comparing table 2.3 to the results in table 2.2 leads to the conclusion that both numerical methods give prices with almost equal precision. Together this shows that both numerical methods can be used to price European options under the SABR model, but since the tree method is much faster this is the best numerical method to use. Table 2.3 also shows that taking 100 time steps seems to be a good choice, since not much precision is added when doubling the number of time steps.
The results in tables 2.2 and 2.4 and in appendix A show that the option prices given by the approximating direct formula for implied volatility are almost all within the approximated confidence interval of the Monte Carlo method. The tables show that the direct method for pricing European options should not be used when the strike price is high, because it then overprices the options, or when ν is high, because it then under prices the options. Furthermore, it seems that the direct formula does not price put options with long time to maturity very well. These last two observations can be explained by the fact that the direct formula is derived by expansion in $\nu^2 T$ [9]. This means that when $\nu^2 T$ has a high value, the approximating direct formula will be less accurate.

[1]The calculations were done on a computer with Linux as its operating system and specifications: 1 AMD Opteron processor 148 (2.2 GHz) 64 bit, Cache memory type L2, 1 Mb, 2 Gb DDR SDRAM - ECC 400 MHz, 2 Gb swap-space

Table 2.4: Pricing European options under the SABR model with fixed cash discrete dividend

The basis situation is $S_0 = 100 = K$, $\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis		Direct	Tree	MC	95% conf.	Direct-Tree	Direct-MC	Tree-MC
-	call	10.8545	10.8632	10.8602	$\pm$ 0.0274	-0.0087	-0.0057	0.0030
	put	8.8470	8.8559	8.8473	$\pm$ 0.0162	-0.0089	-0.0003	0.0086
$\alpha = 25$ &	call	10.6317	10.6308	10.6404	$\pm$ 0.0226	0.0009	-0.0087	-0.0096
$\beta = 0$	put	8.6242	8.6219	8.6131	$\pm$ 0.0180	0.0023	0.0111	0.0088
$\rho = -1$	call	10.5962	10.6028	10.5959	$\pm$ 0.0182	-0.0066	0.0003	0.0069
	put	8.5887	8.5942	8.5863	$\pm$ 0.0197	-0.0055	0.0024	0.0079
$\nu = 0.9$	call	11.3265	11.3490	11.3572	$\pm$ 0.0502	-0.0225	-0.0307	-0.0082
	put	9.3190	9.3414	9.3542	$\pm$ 0.0182	-0.0224	-0.0352	-0.0128
$K = 120$	call	4.6846	4.6719	4.6638	$\pm$ 0.0198	0.0127	0.0208	0.0081
	put	21.7923	21.7798	21.7702	$\pm$ 0.0245	0.0125	0.0221	0.0096

Although the tree method gives more accurate prices for European options, calculating a price with the approximating direct formula is much quicker, since it can be done within a tenth of a second. The application of both methods to real market data has to show if the approximating direct formula is really too inaccurate, or if the difference with the tree method is too small to compensate the huge amount of extra calculation time needed for the tree method.

Chapter 3

Deriving prices for American options under the SABR model

In this chapter, once more two numerical methods are derived to price American options under the SABR model. One is based on the Least-Square Monte Carlo method [16]. The other is the tree method as discussed before, with some slight modifications. In Section 4.3, results of these two numerical methods can be found, for different chosen parameters. From this it can be seen whether the methods give equal options the same values, and which method should be used when pricing American options under the SABR model.

3.1 Deriving a method using Monte Carlo simulations

The method to derive prices of American options using Monte Carlo simulations starts in the same way as the one for European options, namely simulating paths of S. Again the life of the option can be divided into n short time intervals Δt, and the equations of the SABR model can be approximated as before. The difference is now that the holder of the option can also choose to exercise the option at each moment in time between time zero and time T (of course the option can not be exercised more then once). This means for the approximation that at each time step it has to be evaluated if exercising at that moment gives a higher payoff than the expected discounted payoff of holding the option at least one more time step.
The payoff of exercising at time t is usually easy to determine, since this decision can only be made at time t itself. So the value of the stock at time t is known, and the payoff of exercising the option is equal to:

$$[S_t - K]^+ \quad \text{or} \quad [K - S_t]^+,$$

depending on whether it is a call option or a put option.
The expected discounted payoff of continuing however is far more difficult to calculate. Longstaff and Schwartz [16] provide a way to approximate this value when Monte Carlo simulation is used, namely the Least-Squares Monte Carlo (LSM) method. There are other methods based on Monte Carlo simulation, like the one proposed by Andersen [1], but the LSM method is easier to apply to models with multiple stochastic factors, and has a good trade-off between computational time and precision [19]. That is why the Least-Squares Monte Carlo method is used here to derive prices of American options under the SABR model.
We now describe the LSM method, based on the article by Longstaff and Schwartz [16].
After sampling m paths for S under the SABR model, there are m possible values for each S_{t_i}, with $t_i = i\Delta t$, and $i = 0, 1, ..., n$. First the option payoff for each path when exercising at $t_n = T$ is derived. After that, all paths for which the option is in-the-money[1] at time t_{n-1} are considered, which form a set called I_{n-1}. Now the key idea of Longstaff and Schwartz [16] comes into play. They assume an approximate relationship between the value of continuing and the value of the stock of:

$$V_{n-1} \approx \sum_{i=1}^{j} a_{n-1}^{i} g^{i}(S_{n-1}), \tag{3.1}$$

where V_{n-1} is the approximated value of continuing discounted back to the point t_{n-1}, S_{n-1} is the value of the stock at time t_{n-1}, a_{n-1}^{i} are constants, g^{i} is the i'th function of a chosen set of basis functions, like

[1]options which give a strictly positive pay-off when they are exercised at that specific time

Laguerre polynomials, Legendre polynomials or standard polynomials and j is the number basis functions in the chosen set. To find the constants a^i_{n-1}, the following is minimized:

$$\sum_{k\in I_{n-1}} (V^k_{n-1} - \sum_{i=1}^{j} a^i_{n-1} g^i(S^k_{n-1}))^2,$$

where V^k_{n-1} is the value of continuing with path k at time t_{n-1} discounted back to time t_{n-1}, and S^k_{n-1} is the stock price of path k at time t_{n-1}.
Now equation (3.1) gives the expected payoff when continuing at t_{n-1}, and these values are compared with the pay-off of exercising at time t_{n-1}. With this data the decision to exercise or not at t_{n-1} can be taken for each path. Of course the option will not be exercised at t_{n-1}, when it is out of the money at t_{n-1}.
After this step, all paths for which the option is in the money at time t_{n-2} are considered, and so on until time t_0. At every time step the formula $\sum_{k\in I_l}(V^k_l - \sum_{i=1}^{j} a^i_l g^i(S^k_l))^2$ is minimized, to derive all values of a^i_l, where I_l is the set of paths for which the option is in the money at time t_l. The value of continuing is again compared with the value of exercising at time t_l. At the end of this procedure, each path has one exercise time, so at most at only one time point t_l a positive value for the pay-off. These m pay-offs should all be discounted to time t_0 and averaged. This will give the value of the option with the LSM method [16].
A numerical example of this LSM method can be found in the article by Longstaff and Schwartz [16].

Application to the SABR model

The LSM method could be applied to the SABR model in the same way as described above. However, since each simulated path of the SABR model consists of not only information about the stock price, but also about the volatility, this information should also be used in the LSM method. So the basis functions should not be functions of one variable S, but it should be functions of the two variables S and α. It is chosen to use the simplest set of basis functions, namely normal polynomials, since using more complicated functions costs more computation time. Firstly, the results were checked when the highest degree of the polynomials is only 2. In that case, the following approximated relationship is assumed:

$$V_i \approx a^1 + a^2\alpha_i + a^3(\alpha_i)^2 + a^4 S_i + a^5(S_i)^2 + a^6\alpha_i S_i$$

Since the LSM method is a method in which the best strategy is approximated, it will always give prices that are too low, so higher prices means a higher degree of precision. Of course, taking a polynomial of higher degree also costs more time, but after some numerical tests it appeared that taking 400000 paths instead of 1 million and a polynomial of degree 4 instead of 2 made the computational time approximately equal and gave better approximations of the option prices. Some results of these tests can be found in table 3.1, where the method is applied to price put options on a stock which does not give dividend and the interest rate is constant. The word 'time' in the table stands for the approximated calculation time in seconds.

Table 3.1: Comparing the LSM method with polynomials of degree 2 and 4.

The basis situation is $S_0 = 100 = K$, $r = 0.05$,
$\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis	LSM degree 2	time	LSM degree 4	time
-	8.1277	170	8.1727	180
$\rho = -1$	7.9870	150	8.0116	160
T=2.5	11.5733	160	11.6023	170

Thus the LSM method is applied to the SABR model using polynomials in α and S of degree 4. The rest of the method was exactly applied as in the general LSM method.

3.2 Deriving a method using trees

Since transforming a tree method for pricing European options to a method for pricing American options is very simple, the following section is brief.

Again the Vellekoop and Nieuwenhuis tree method is used to price options. It is applied in the same way as described for European options, even in the case of dividends. The only difference is that now the value at each node of the tree is not the sum of the jump probabilities times the interpolated value of the accompanying successor node, which is how the value was calculated in the European case, but in this case it is the maximum of this value and the payoff of exercising at that point.

3.3 Comparing both methods

To compare the methods and see if they give the same values, the methods are again applied to the same theoretical options as for the European case. So in tabel 3.2 the results are given for American options on an underlying that does not pay dividend, and the interest rate is assumed to be constant. In this table, only prices of put options are stated, since it is never optimal to exercise a call option before maturity, when there is no dividend. This can be seen by the facts that the option will not be exercised if $(S_{t_k} - K) \leq 0$, so only the case where $\max(S_{t_k} - K, 0) = S_{t_k} - K$ has to be considered, and the expected price of the underlying asset at time t_{k+1} is $S_{t_k}/D(t_k, t_{k+1})$. The following (in)equalities show that the expected payoff at t_{k+1} discounted to time t_k is greater or equal to the payoff at time t_k, which is $S_{t_k} - K$:

$$D(t_k, t_{k+1})\mathbb{E}_{\mathbb{Q}}[\max(S_{t_{k+1}} - K, 0)|S_{t_k}] \geq D(t_k, t_{k+1})\mathbb{E}_{\mathbb{Q}}[S_{t_{k+1}} - K|S_{t_k}] = D(t_k, t_{k+1})(\mathbb{E}_{\mathbb{Q}}[S_{t_{k+1}}|S_{t_k}] - K) =$$

$$D(t_k, t_{k+1})(S_{t_k}/D(t_k, t_{k+1}) - K) = S_{t_k} - D(t_k, t_{k+1})K \geq S_{t_k} - K$$

Thus it is shown that it is never the best strategy to exercise a call option before maturity, in the case the underlying does not pay dividend.
In table 3.4 results are given for American options on an underlying that gives discrete non-proportional dividends, and the interest rate is not constant with values defined in section 2.4.
To create the results of both tables, the tree method is applied in the same way as described in section 2.4 and the LSM method is applied using 400.000 paths, and using polynomials of degree 4. In both methods, the time is divided into 100 steps. In table 3.3 again some results can be found for both methods applied to the first situation(no dividends), taking 200 time steps instead of 100.

Table 3.2: Comparing two different methods for pricing American put options under the SABR model with no dividends

The basis situation is $S_0 = 100 = K$, $r = 0.05$,
$\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis	Tree	LSM	95% conf.	Tree-LSM
-	8.1685	8.1727	± 0.0295	-0.0042
$\alpha = 25, \beta = 0$	7.7372	7.7349	± 0.0313	0.0023
$\rho = -1$	8.0216	8.0116	± 0.0340	0.0100
$\nu = 0.9$	8.7173	8.7314	± 0.0361	-0.0141
$K = 120$	21.6545	21.6576	± 0.0377	-0.0031
T=2.5	11.6098	11.6023	± 0.0423	0.0075

Table 3.3: Results for situation 1, using 200 time steps

The basis situation is $S_0 = 100 = K$,
$\alpha = 0.4$, $\beta = 0.7$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis	Tree,n=200	LSM,n=200
-	8.1640	8.1519
$\rho = -1$	8.0179	8.0168
$\nu = 0.9$	8.6995	8.7140
$T = 2.5$	11.6006	11.5984

Table 3.4: Pricing American options under the SABR model with non-proportional discrete dividend

The basis situation is $S_0 = 100 = K$,$\alpha = 0.4$,
$\beta = 0.7$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$

Change from basis		Tree	LSM	95% conf.	Tree-LSM
-	call	11.1266	11.1198	$\pm$ 0.0551	0.0068
	put	9.1906	9.1814	$\pm$ 0.0320	0.0092
$\beta = 0$ &	call	10.9444	10.9456	$\pm$ 0.0464	-0.0012
$\alpha = 25$	put	8.9215	8.9189	$\pm$ 0.0373	0.0026
$\rho = -1$	call	10.9716	10.9606	$\pm$ 0.0381	0.0110
	put	8.8469	8.8347	$\pm$ 0.0399	0.0122
$\nu = 0.9$	call	11.6396	11.6201	$\pm$ 0.0890	0.0195
	put	9.7233	9.7131	$\pm$ 0.0393	0.0090
$K = 120$	call	4.7127	4.7220	$\pm$ 0.0414	-0.0093
	put	22.5958	22.5901	$\pm$ 0.0472	0.0057

More results of the two methods can be found in appendix B.
The computation time for the given results, on the earlier specified computer, is again approximately 15 seconds per option for the tree method and approximately 175 seconds per option for the LSM method.
From these results it can be seen that the prices given by the tree method are all within the confidence interval given by the LSM method. Table 3.3 shows that taking 200 instead of 100 time steps does not add much to the precision (the differences are almost equal), while the calculation time is doubled. That is why the choice of taking 100 time steps again was made.
It can be concluded that both methods can be used to price American options under the SABR model, and since the tree method is much quicker, this method seems to be the most viable, and is used in the following chapter to fit the parameters to real market data.

Chapter 4

Fitting to real market data

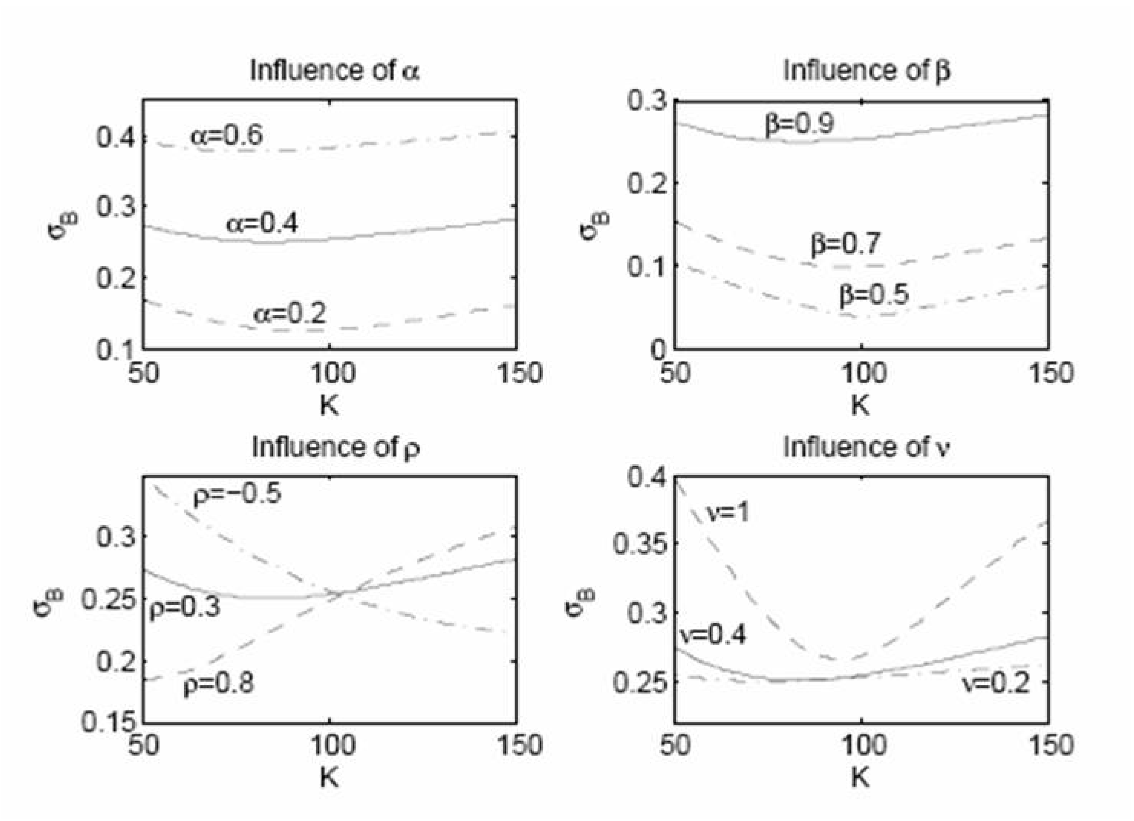
In this chapter, the approximating direct formula and the tree method are applied to real market data. The Monte Carlo and Least-Squares Monte Carlo method are not used anymore, since it was concluded in chapter 2 and 3 that the tree method is much quicker and just as good.
First it will be discussed how the parameters can be fitted to real market data, after which results will be shown.

4.1 Fitting the parameters

To fit the parameters, first a set of options has to be chosen for which the parameters have to be found. This set should consist of all European or all American options on the same underlying, with all market prices given at the same moment in time. Usually it is chosen to handle only sets of options with equal maturity times. The most ideal situation is that the parameters are consistent over different maturities, but this is mostly not the case. An explanation for this can be that parameters are chosen to be constant over time, while in practice they should be time-dependent. Thus the only difference between the different options in the chosen set is the strike price and whether they are call or put options.
The general idea of fitting the parameters is that the differences between the market and the model option prices are minimized. For the objective function, often the sum of squared errors, $\sum_i (U_i - V_i)^2$, is chosen [12]. Here U_i is the market value of the i'th option that is selected and V_i is the value for the i'th option given by the model. Other possibilities for the objective function are, for example, the sum of relative squared errors, $\sum_i (\frac{U_i - V_i}{U_i})^2$, and the sum of weighted squared errors, $\sum_i w_i(U_i - V_i)^2$, where w_i is the weight that is given to the i'th option.
Since for most options the value is known up to a couple of cents, almost independent of the total value of the option, in this thesis it was decided to minimize the normal sum of squared errors.

Since the approximating direct formula is of course a very quick method to price options, it will also be very quick to calibrate. Although it is shown in chapter 2 that this approximating direct formula can not be used to price options in all cases, it can be used to get good starting values for the calibration of the SABR model using the tree method. This is a very important feature of the SABR model, since having good starting values for the calibration is crucial for keeping the time needed for the calibration within bounds.
Before starting the calibration, it is important to know the influence of the different parameters on the strike versus implied volatility graph of the approximating direct formula. Starting values of the parameters can then be derived from a plot of the Black-Scholes implied volatilities given by the market data. In figure 4.1, the influence of the different parameters on the graph of σ_B as function of strike prices is shown. From these figures a statement by Hagan et al. can be confirmed, that the parameters α and β have the same effect on the volatility smile, which means that fitting β would clearly amount to "fitting the market noise" [9]. So the exact value of β is not that important and can be chosen from "aesthetic" considerations, like a consideration by Hagan et al. [9] that the stochastic lognormal model is the "most natural" representation, so β should be 1. Numerical evidence of this can be found in the results section.

Figure 4.1: Influence of each parameter, standard situation is $\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$, $S_0 = 100$, $r = 0.05$ and $T = 1$



An at-the-money (ATM) option is an option for which the strike price is exactly equal to the forward price, i.e. $F = K$. In this situation, the approximating direct formula for implied volatility becomes:

$$\sigma_B(f,f) = \frac{\alpha}{f^{(1-\beta)}}\{1 + [\frac{(1-\beta)^2}{24}\frac{\alpha^2}{f^{2-2\beta}} + \frac{1}{4}\frac{\rho\beta\nu\alpha}{f^{(1-\beta)}} + \frac{2-3\rho^2}{24}\nu^2](T-t) + \ldots\},$$

where f is again the value of the forward.
When the term $[...](T-t)$ is very small, it follows that $\alpha \approx f^{(1-\beta)}\sigma_B(f,f)$. Usually an option with strike price exactly equal to f does not exist, but from the market data one can derive a good guess of the Black-Scholes ATM volatility. When β is chosen to be 1, it now follows that a good starting value for α is this ATM volatility given by the market data. Starting values of ρ and ν can also be guessed by drawing the implied volatility curve given by the market data and comparing this with figure 4.1.
The approximating direct formula for implied volatility and the Black-Scholes formula are only derived to price European options. So in principle, the above reasoning does not give good starting values for American options. However, since American call options on an underlying which does not give dividend, or gives dividend yield lower than the interest rate, should never be exercised before maturity, it follows that this option can be seen as European. So the above starting values can still be used. For American options on an underlying that does give discrete dividends, starting values of ρ and ν can be derived in the same way, but the starting value of α given by the above reasoning will be not very good. The starting value of α should be chosen slightly higher, but how much higher depends on the situation. For example, if the underlying gives, between the time of buying and maturity, only one dividend near maturity, a starting value of α can be derived by calculating the ATM implied volatility from the price discounted to the dividend date and taking the dividend date as maturity, because the option will probably be exercised at this date.

For European options, the SABR model can be calibrated using the approximating direct formula and the tree method. For the tree method, as starting values can be used the calibrated parameters from the approximating direct formula, since these can be calculated very quickly. Before starting the calibration of the tree method, it has to be noticed that the grid used to price the options does not depend on the strike price. So for each set of parameters, the grid only has to be calculated once, and only the valuation part by working backward through the tree has to be done for each strike price.
For American options, calibrating the direct formula does of course only make sense when the set of options consists only of call options and the underlying does not pay dividend or pays a dividend yield lower than the interest rate. For all other options only the tree method can be used, with starting values given by the reasoning above on how to derive starting values.

4.2 Results

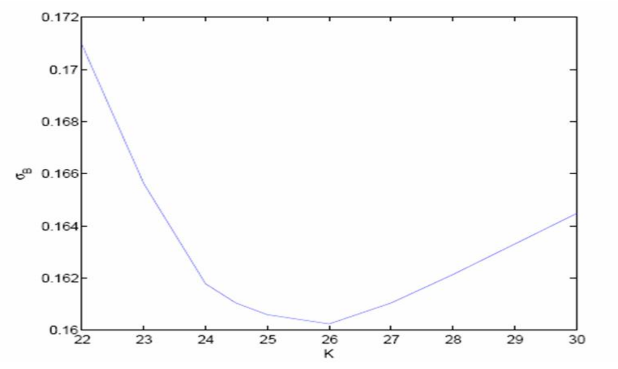
Most of the fitting procedure described in the previous section is made clear by applying it to Royal Dutch Shell American call and put options at the 7th of March 2007, with maturity of 72 days. In these 72 days one dividend is paid of 0.265 euro at day 63. The value of the underlying is 24.46 euro. The market data used can be found in table 4.1, where the data are based on the convention that a year is exactly 365 days.

Table 4.1: American option prices Shell, 7th of March 2007, $T = 72/365$

Strike(K)	22	23	24	24.5	25	26	27	28	30
Call price	2.6803	1.7979	1.0608	0.7720	0.5403	0.2345	0.0881	0.0288	0.0022
Put price	0.1214	0.2699	0.5793	0.8147	1.1052	1.8310	2.6987	3.6420	5.6068

The graph of the implied volatility curve given by the call option market prices can be found in figure 4.2.

Figure 4.2: Market smile of Shell call options



The ATM strike price is 24.38 ($24.46e^{\int_0^{72/365} r(s)ds} - 0.265e^{\int_{63/365}^{73/265} r(s)ds}$). Comparing the implied volatility graph 4.2 with the graphs in figure 4.1, good starting values of ρ and ν can be derived. A good starting value for ρ seems to be -0.2, since graph 4.2 looks very much like the graph in figure 4.1 of $\rho = 0.3$, but the lowest point lies at the right of the ATM point. By looking at the graphs of different values of ρ this means that ρ is negative. A good starting value for ν seems to be 1.2, since the difference in value of σ_B at a strike price 20% under the ATM value and at the lowest point is already 1%, which seems to be around equal to this difference in the graph of $\nu = 1$. Furthermore, the lowest point is at the right of the ATM point, which means by looking at the graphs of different values of ν that ν should be bigger than 1.
The ATM price of the call option is approximately 0.8392 (by linear interpolation). Discounting this price back from 72 days to 63 days gives a value of 0.8384. The ATM implied volatility, using $T = 63/365$, is 0.1766, which gives that a good starting value for α is $24.38^{(1-\beta)}0.1766$. When choosing $\beta = 0.5$, 0.7 and 1, it follows that good starting values for α, with these different values of β, are: $\alpha = 0.872$, 0.4604 and 0.1766.

To minimize the sum of squared errors, the MATLAB function "lsqnonlin[1]" is used. First the calibration is done with the number of time steps, n, equal to 20, and the calibrated parameters from this calibration are used as starting values for the calibration with $n = 50$. The results of the different calibrations with the three different sets of starting values can be found in table 4.2, where LSE stands for Least Square Error.

Table 4.2: Calibrating Shell put options with different sets of starting values

	α	β	ρ	ν	LSE
Starting values 1	0.8720	0.5	-0.2	1.2	
Calibration 1	0.8629	0.4932	-0.1528	1.1012	0.000056
Starting values 2	0.4604	0.7	-0.2	1.2	
Calibration 2	0.4546	0.6926	-0.1820	1.1851	0.000082
Starting values 3	0.1766	1.0	-0.2	1.2	
Calibration 3	0.1773	0.9890	-0.2382	1.1375	0.000066
Starting values 3, β fixed	0.1766	1.0	-0.2	1.2	
Calibration 3, β fixed	0.1714	1.0	-0.2412	1.1232	0.000064

From these results it can be seen that there is no significant difference in least square error between the four calibrations, while the calibrated value of β is around its starting value. It can even be seen that the least square error for β fixed at one or not fixed, but starting at one is approximately equal. This confirms the earlier statement that fitting β would amount to fitting "market noise" [9] and the choice is made to always fix β at 1.

In earlier research, Ter Horst [20] fitted the parameters of the Heston model (equation (1.7)) using the same set of put options of Shell as mentioned in table 4.1. In that research, the sum of relative squared errors, $\sum_i (\frac{U_i - V_i}{U_i})^2$, was minimized to fit the parameters. To be able to compare the results of the Heston model with the results of the SABR model, this objective function should also be minimized to fit the parameters of the SABR model. In table 4.3 again the market prices of the calibrated put options can be found, together with the values of the options given by the SABR model (with the fitted parameters) as well as the Heston model (with its fitted parameters), and both least square errors.

Table 4.3: Comparing the SABR model with the Heston model

Strike(K)	22	23	24	24.5	25	26	27	28	30	LSE
Market price	0.1214	0.2699	0.5793	0.8147	1.1052	1.8310	2.6987	3.6420	5.6068	
SABR prices	0.1183	0.2782	0.5762	0.8096	1.1022	1.8350	2.7042	3.6459	5.6060	0.0002
Heston prices	0.1200	0.2769	0.5785	0.8049	1.0973	1.8653	2.7910	3.7757	5.7718	0.0047

From this table it can be seen that the SABR model fits the market prices much better for the in-the-money options and slightly worse for the out-of-the-money options. Seen over all 9 options, the SABR model can fit the market prices much better. Since β is chosen to be one, the differences between the Heston and the SABR model is only in the differential equation for the volatility. In section 1.4, it was shown with the Itô formula that, in the case $\beta = 1$, α_t in the Heston model is given by equation (1.8). The better fitting of the SABR model to the market can therefore have two different explanations. Firstly, it can be caused by the fact that the drift term of the volatility in the Heston model, causes the volatility to stay around θ (it is mean-reversing), while in the SABR model ν (the volatility of the volatility) is not bounded, so the volatility can fluctuate much more. A volatility of 40% is not uncommon in the market, so when the starting volatility is 15%, it is not strange if ν is around 1, which indicates big fluctuations of the volatility. Secondly, the assumption of the volatility α_t to have a lognormal distribution can approximate the market behaviour closer than the assumption of α_t having a normal distribution.

[1]with settings TolX=0.0001, TolFun=0.000005, DiffMaxChange=0.5 and DiffMinChange=0.00001. TolX and TolFun mean that lsqnonlin is terminated when the change in variables is less that 0.0001 or when the change in the objective function value is less than 0.000005. DiffMaxChange and DiffMinChange are the maximum and minimum value with which the values of the variables change at each step of the iteration.

Besides fitting the American put options of Shell with maturity 72 days, other sets of options were fitted. The market data used can be found in appendix C. In table 4.4, the fitted parameters can be found for put options, call options and call plus put options with 5 different maturity times. The fitting is done exactly as described before, with starting values $\alpha = 0.1766, \rho = -0.2$ and $\nu = 1.2$, except that for the starting values of the calibration of the put plus call options the average of the calibrated values of the put and the call options is chosen. Options 1 have a maturity of 9 days and options 2 have a maturity of 44 days, both are American options without dividends. Options 3 are the earlier mentioned options with 1 dividend and maturity of 72 days, options 4 have a maturity of 653 days during which there are 7 dividends paid and options 5 have a maturity of 1017 during which 11 dividends are paid. In this table β is not mentioned, since it is taken to be one. Furthermore, LSE stands again for least squares error, max error is the absolute maximum pricing error between prices given by the SABR model with the fitted parameters and market prices, n low is the number of time points used for the first calibration and n final is the number of time points used in the final calibration. Since for calibrating put plus call options real good starting values are known, this calibration is done directly with the final number of time steps wanted.

Table 4.4: Results of calibration of Shell options

	α	ρ	ν	LSE	max error	n low	n final
put options 1	0.1969	-0.2950	2.4986	0.000044	0.0046	10	30
call options 1	0.2004	-0.3641	1.9560	0.000119	0.0075	10	30
put+call options 1	0.1986	-0.3295	2.2273	0.000166	0.0070	-	30
put options 2	0.1756	-0.2715	1.3832	0.000054	0.0056	20	50
call options 2	0.1747	-0.2405	1.6042	0.000071	0.0044	20	50
put+call options 2	0.1752	-0.2537	1.4940	0.000129	0.0056	-	50
put options 3	0.1714	-0.2412	1.1232	0.000064	0.0042	20	50
call options 3	0.1702	-0.2210	1.0088	0.000071	0.0047	20	50
put+call options 3	0.1708	-0.2292	1.0655	0.000347	0.0097	-	50
put options 4	0.1885	-0.4752	0.4114	0.000058	0.0036	40	200
call options 4	0.1919	-0.4927	0.3758	0.000024	0.0028	40	200
put+call options 4	0.1902	-0.4845	0.3937	0.002500	0.0153	-	200
put options 5	0.2448	-0.6592	0.3326	0.000289	0.0099	50	300
call options 5	0.2385	-0.5281	0.2306	0.000079	0.0064	50	300
put+call options 5	0.2425	-0.6453	0.2667	0.037000	0.0795	-	300

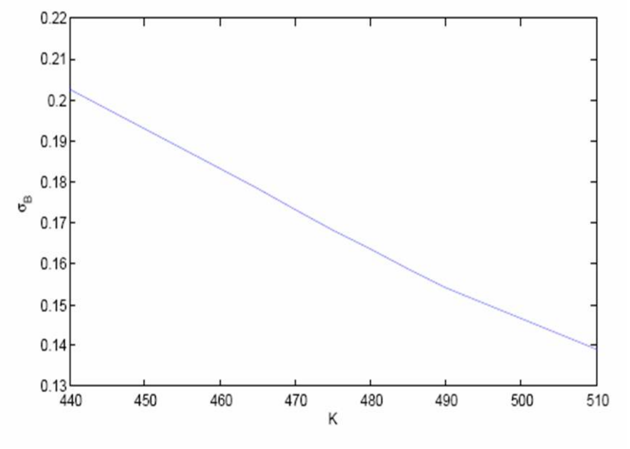
From these results it can be concluded that the fits of put and call options separately are very good (all prices are given within one cent of the market price). The fits of put plus call options together are reasonably well for most different maturity times. Only for the longest time to maturity, namely 1017 days, fitting the put and the call options together does not give very good results. This can be explained by the fact that for longer times to maturity the prices are higher, the prices are less certain and the options are not as liquid as the options with a short time to maturity. Together, this leads to the fact that the bid-ask spread for these options is much wider than for options with a short time to maturity, so the prices, given by the SABR model when the put and call options with maturity 1017 days are fitted together, may well all be within the bid ask spread of these options.

The parameter α is almost stable over maturities, which is what is wanted, since in an ideal situation the parameters are equal over different maturity times. The parameter ρ is not totally stable, but does not fluctuate a lot, which could implicate that ρ should be made time-dependent. The parameter ν decreases for longer maturities, which implies that this parameter depends on the time to maturity and that is not what is wanted. Looking at the results, an explanation of these values of ν can be that the mean-reversion assumption of the volatility used in the Heston model is a right assumption. This mean-reversion causes the volatility to stay around a certain value, so the chance of the value of the volatility to be far away from its starting value is not much larger for longer time spans, or in other words, the variance of the distribution of α_t does not grow (much) for higher values of t. In the SABR model the assumption is that the variance is $\nu^2 T$, so it does grow for higher values of maturity times and the only way to compensate this, is to have lower values of ν for higher time to maturities. Since the results are very good, it seems that, although there is this flaw in the SABR model, the model seems to be a good model to use when pricing American options.

The time needed to derive these results depends on the number of time steps and the number of options in the set that is calibrated. For example, for 20 time steps it costs, with matlab, on the used computer, approximately 20 seconds to price 9 options with a given set of parameters, while for 50 time steps it costs approximately 40 seconds. On average it took 6 iterations to calibrate with the low number of time steps and after that 2 iterations to calibrate with the final number of time steps. So the time needed to calibrate for example call options 3 was approximately $28 * 20 + 12 * 40$ seconds, which is approximately 17 minutes (with 3 parameters, n iterations means $4(n + 1)$ calculations of the prices of the options in the set).

The mentioned results are of American options with and without dividends. To be able to say something about the performance of the SABR model for European and American options, also European options have to be calibrated. For European options, once more the starting values can be derived by looking at the implied volatility curve of market data. These values can now be used to calibrate the SABR model using the approximating direct formula, and the results of this calibration can be used to calibrate the SABR model with the tree method. Since the parameters given by the calibration using the approximating direct formula will probably give already very good starting values for the tree method, the calibration using this method will be done directly with the number of time steps wanted. The data used are from options on the AEX index, again on the 7th of March 2007. Options with 5 different maturities are calibrated, namely: 45 days, 73 days, 101 days, 654 days and 1382 days. The number of dividends paid during the lifetime of the options is respectively 4, 15, 17, 65 and 134. The market data used can again be found in appendix C. From the market data of the call options with a maturity time of 73 days the implied volatility curve of figure 4.3 was derived.

Figure 4.3: Market implied voaltility curve of AEX call options 3



Using the same reasoning as for the Shell options it can be seen that good starting values of ρ and ν are -0.7 and 1.5. The at the money value is approximately 475.38. So using the option with strike 475 as the approximation of the at the money option, the starting value of $\alpha = 0.1685$ was derived. The calibrated parameters of the different sets of options, using the approximating direct formula, can be found in table 4.5. The calibrated parameters using the tree method[2] can be found in table 4.6. Again as starting values for calibrating put+call options, the average calibrated parameters of the put and the call options were taken.

[2]the settings used in “lsqnonlin” were TolX=0.0001, TolFun=0.00001, DiffMaxChange=0.5 and DiffMinChange=0.0001

Table 4.5: Results of calibration of AEX options using the approximating direct formula

	α	ρ	ν	LSE	max error
call options 1	0.1702	-0.7742	1.3911	0.0003	.0096
put options 1	0.1700	-0.7808	1.3801	0.0003	0.0097
put+call options 1	0.1701	-0.7775	1.3857	0.0013	0.0160
call options 2	0.1686	-0.7552	1.2118	0.0035	0.0337
put options 2	0.1683	-0.7692	1.1919	0.0035	0.0339
put+call options 2	0.1685	-0.7622	1.2015	0.0124	0.0494
call options 3	0.1676	-0.7334	1.0651	0.0245	0.0923
put options 3	0.1673	-0.7639	1.0268	0.0190	0.0827
put+call options 3	0.1675	-0.7485	1.0456	0.0639	0.1201
call options 4	0.1817	-0.7529	0.4405	0.0251	0.0892
put options 4	0.1811	-0.7775	0.4285	0.0245	0.2921
put+call options 4	0.1814	-0.7652	0.4345	0.2921	0.1976
call options 5	0.2035	-0.9879	0.2338	0.4476	0.4048
put options 5	0.2035	-0.9999	0.2310	0.4459	0.3993
put+call options 5	0.2035	-0.9939	0.2323	0.8978	0.4143

Table 4.6: Results of calibration of AEX options using the tree method

	α	ρ	ν	LSE	max error	n
call options 1	0.1699	-0.7755	1.4259	0.0003	0.0088	50
put options 1	0.1698	-0.7820	1.4133	0.0003	0.0093	50
put+call options 1	0.1698	-0.7785	1.4194	0.0013	0.0153	50
call options 2	0.1684	-0.7527	1.2619	0.0036	0.0304	76
put options 2	0.1681	-0.7641	1.2407	0.0036	0.0312	76
put+call options 2	0.1682	-0.7543	1.2593	0.0125	0.0493	76
call options 3	0.1676	-0.7368	1.1065	0.0256	0.0960	100
put options 3	0.1673	-0.7657	1.0621	0.0195	0.0847	100
put+call options 3	0.1674	-0.7509	1.0844	0.0654	0.1229	100
call options 4	0.1821	-0.7538	0.4588	0.0308	0.0971	200
put options 4	0.1812	-0.7647	0.4533	0.0308	0.1107	200
put+call options 4	0.1817	-0.7592	0.4562	0.2964	0.2014	200
call options 5	0.2032	-0.9880	0.2341	0.4531	0.4075	200
put options 5	0.2031	-0.9999	0.2309	0.4525	0.3996	200
put+call options 5	0.2032	0.9940	0.2324	0.9157	0.4220	200
call options 5	0.2031	-0.9878	0.2339	0.4515	0.4021	400
put options 5	0.2031	-0.9999	0.2308	0.4502	0.3953	400
put+call options 5	0.2031	-0.9938	0.2324	0.9056	0.4112	400

The values of the LSE in tables 4.5 and 4.6 are mostly higher than they were for the Shell options, but the reason for this is that the value of the underlying and the values of the options are much higher. In comparison to the American options with maturity 72 days, the values for the AEX options 1 until 5 are respectively approximately 10, 15, 20, 45 and 65 times higher. Keeping this in mind, it can be seen that the prices of European options can be fitted precisely. It seems that the approximating direct formula can be used well to price European options. Using the tree method for calibration leads to approximately the same values of the least square errors and the maximum errors. Since calibrating with the approximating direct formula can be done within seconds, calibrating with the tree method takes far too long and can only be used once in a while as a check for the calibration of the approximating direct formula. When using this tree method, it is seen from table 4.6 that the number of time points needed to price accurately is not high. The fitting of the options with 1382 days to maturity using 200 time steps instead of 400 is only slightly worse, so the number of time steps does not have to increase linearly with the increase of the time to maturity.
The values of the parameters, for different maturity times, display the same tendencies as noticed from the results of the American options, namely α is fairly stable over different time to maturities, ρ should probably be made time-dependent, and ν decreases for higher values of T.

Chapter 5

Conclusion and discussion

The main question of this thesis was:

- Is the SABR model a good model to use when pricing European and American options?

To answer this question, first it was shown in chapter 2 that the approximating direct formula does give accurate prices for most situations, but does not give the same European option prices as the numerical methods, when the strike price K is high and when $\nu^2 T$ has a high value, where ν is the volatility of the volatility and T is the time to maturity. Furthermore, two numerical methods were derived for pricing European options (chapter 2) as well as American options (chapter 3). By comparison of these methods it was shown that the numerical method based on the Vellekoop and Nieuwenhuis method [21] is relatively quick and accurate for pricing European as well as American options. That is why this numerical method for the pricing of options under the SABR model was applied to real market data.
To conclude if the SABR model gives accurate prices for real traded options, the right values of the parameters of the SABR model (α, β, ρ and ν) need to be found. For this fitting of the parameters it is crucial to have good starting values. It is shown in chapter 4 that the value of β does not really influence the accuracy of the fit, since the value of α adjusts to the value of β. Therefore β can be chosen in the interval $[0, 1]$ and in this thesis it is chosen to be 1. Furthermore, it is shown that it is easy to derive good starting values of ρ and ν from the implied volatility curve given by the market data. In section 4.2 results are given for the fitting of the SABR model to American options as well as European options. From these results several observations can be made:

1. The SABR model seems to have a better fit to market prices of American put options than the Heston model. This is probably caused by the fact that the volatility in the Heston model ($\sqrt{\sigma_t}$) is assumed to have a normal distribution, while in the SABR model it is assumed to have a lognormal distribution.

2. For American options on a single stock, the error between the prices given by the SABR model and the market prices, for a set of only call or only put options with the same time to maturity, is less than 1 cent, even for long time to maturity. (Which is very good, since options are always traded on whole cents.) Fitting put and call options together gives this same maximal error only for short time to maturities. For longer maturity times, this maximal error grows, due to higher option prices and larger insecurity, but it still gives a price within a normal range of its bid-ask spread.

3. For European options on an index, the error mostly is over 1 cent, but when the error is seen relative to the average option prices, it is mostly under 0.5%. Only for the time to maturity of almost 4 years, this error grows above 0.5%, but the prices are still in the expected bid ask spreads.

4. The SABR model can be fitted to European options just as well with the approximating direct formula as with the tree method, while this fitting takes only a couple of seconds instead of 10 minutes up to a couple of hours for the tree method.

5. The parameter α is almost stable over different maturity times, for American options as well as European options. Since the process of the underlying should not depend on maturity times of options, this stability of the parameter is what is wanted. The parameter ρ fluctuates a little over different maturity times, which can imply that this parameter should be made time-dependent, and the parameter ν decreases for higher maturities, which implies that the volatility α_t should probably have a mean-reversion term.

From this enumeration it follows that the SABR model is a good model to use when pricing options, and it seems even better than the Heston model.

Although the results seem to be very good, there are critical remarks to be made, which lead to recommendations for further investigations.
The SABR model as defined by Hagan et al. [9], depends on the maturity time of the option (see equation (1.6)). Since there are a lot of different options on the same underlying, the movement of the underlying should not depend on these maturity times. By taking $\beta = 1$, as was done when deriving the results, this dependency disappears for options without dividend, with dividend yield or with discrete proportional dividends, but for options on an underlying which gives discrete fixed cash dividends the dependency remains (see section 1.4). That is why research should be done on the following, almost equal, model:

$$dS_t = r(t)S_t dt + \alpha_t S_t^{\beta} dW_t^1, \quad S_0 = s$$

$$d\alpha_t = \nu\alpha_t \; dW_t^2, \quad \alpha_0 = \alpha$$

$$d\left\langle W_t^1, W_t^2 \right\rangle = \rho dt,$$

This model is equal to the SABR model if $\beta = 1$ and the underlying does not pay discrete fixed cash dividends, while it does not have the dependency on T and/or dividends in case $\beta \neq 1$ and/or the underlying does pay fixed cash dividends.
Since it was seen that the parameters ρ and ν are (slightly) instable of different maturity times, there should be investigated a model which incorporates mean reversion of α_t and time-depency of ρ. The model we propose to investigate is therefore a combination of the Heston and the SABR model, namely:

$$dS_t = r(t)S_t dt + \alpha_t S_t dW_t^1, \quad S_0 = s$$

$$d\alpha_t = \kappa(\theta - \alpha_t)dt + \nu\alpha_t \; dW_t^2, \quad \alpha_0 = \alpha$$

$$d\left\langle W_t^1, W_t^2 \right\rangle = \rho(t) dt$$

The conclusion concerning the better fitting of the SABR model compared to the Heston model, is now only based on the fitting of one set of American put options. Both models should be applied to more sets of options before a stronger conclusion can be drawn.
There is no proof that the original Vellekoop and Nieuwenhuis method [21], using linear interpolation, converges when applied to the SABR model (although numerical results seem to show this). This is due to the fact that, for the SABR model it has not been proved that the martingale problem, defined in the paper by Vellekoop and Nieuwenhuis [21], has a unique solution. Furthermore, it has not been proved that the original method converges to a unique solution when cubic interpolation is used instead of linear interpolation [21]. Further theoretical investigation on these subjects is therefore recommended.
The distribution function used in the definition of the grid, in the numerical tree method, is based on the approximating direct formula. Since this approximating direct formula seems not to be accurate for high values of $\nu^2 T$, the distribution function used in these cases can define the grid inaccurately. Further research on the distribution function is for this reason desirable.
It has not been determined whether the intervals, together with the number of grid points chosen in this numerical method, define an optimal grid. Further investigation on defining an optimal grid can improve the given results.
Additionally, research should be done to define the optimal number of time points at which the upper bounds and the bounds of the intervals should be calculated, and to define the best way to find these values at other time points, based on the values at the time points calculated.
Finally, although the absolute errors are important to know, the fitting of different sets of options can not always be compared very well by looking at the absolute least square errors. This is due to the fact that an error of 1 cent on an option value of 10 euro is normally better than an error of 1 cent on an option value of 1 euro. The precision that is needed depends on the bid-ask spread, so probably the objective function that should be minimized for the fitting of the parameters, should depend on this spread as well. Further research should be done to find an objective function for which, when applied to different sets of options, the answers can be compared more easily.

References

[1] Andersen, L. A simple approach to the pricing of Bermudan swaptions in the multifactor LIBOR market model. *Journal of Computational Finance 3* (2000), 1–32.

[2] Black, F. The pricing of commodity contracts. *The Journal of Financial Economics 3* (1976), 167–179.

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

[4] Cox, J., Ross, S., and Rubinstein, M. Option pricing: A simplified approach. *Journal of Financial Economics 7* (1979), 229–263.

[5] Derman, K., and Kani, I. Riding on a smile. *Risk* (1994), 32–39.

[6] Dupire, B. Pricing with a smile. *Risk* (1994), 18–20.

[7] Florescu, I., and Viens, F. A binomial tree approach to stochastic volatility driven model of the stock price. *Annals of the University of Craiova, Mathematics and Computer Science series 32* (2005), 126–142.

[8] Florescu, I., and Viens, F. Stochastic volatility: option pricing using a multinomial recombining tree. Preprint Department of Statistics, Purdue University, 2007.

[9] Hagan, P., Kumar, D., Lesniewski, A., and Woodward, D. Managing smile risk. *Wilmott magazine* (2002), 84–108.

[10] Hagan, P., Lesniewski, A., and Woodward, D. Probability distribution in the SABR model of stochastic volatility. Wilmott magazine, 2004.

[11] Hilliard, J., and Schwartz, A. Binomial option pricing under stochastic volatility and correlated state variables. *The Journal of Derivatives 4* (1996), 23–39.

[12] Hull, J. *Options, futures and other derivatives*, 6 ed. Pearson Prentice Hall, 2006.

[13] Hyland, K., McKee, S., and Waddell, C. Option pricing, Black-Scholes, and novel arbitrage possibilities. *IMA Journal of Mathematics Applied in Business & Industry 10* (1999), 177–186.

[14] Kargin, V. Lattice option pricing by multidimensional interpolation. *Mathematical Finance 15* (2005), 635–647.

[15] Leisen, D. Stock evolution under stochastic volatility: a discrete approach. *The Journal of Derivatives 8* (2000), 8–27.

[16] Longstaff, F., and Schwartz, E. Valuing American options by simulation: A simple least-squares approach. *The review of financial studies 14* (2001), 113–147.

[17] Pliska, S. *Introduction to Mathematical Finance.* Blackwell Publishing, 1997.

[18] Shreve, S. *Stochastic calculus for finance II.* Springer, 2004.

[19] Stenstoft, L. Assessing the least squares Monte-Carlo approach to American option valuation. *Review of derivatives research 7* (2004), 129–168.

[20] Ter Horst, J. American option pricing in the Heston model. Master's thesis, Delft Univeristy of Technology, 2007.

[21] Vellekoop, M., and Nieuwenhuis, J. A tree-based method to price American options in the Heston model. Preprint Financial Engineering Laboratory, University of Twente, 2007.

Appendix A

Results of methods for the pricing of European options

The table below (table A.1) is an extension of table 2.2. So here are shown prices of European options given by the three different methods used in this thesis to price European options under the SABR model, on an underlying that does not pay dividend. The standard situation for the table is again given by the following parameter values: $\alpha = 0.4, \quad \beta = 0.9, \quad \rho = 0.3, \quad \nu = 0.4, \quad S_0 = 100, \quad K = 100, \quad r = 0.05, \quad T = 1.$ The parameters under 'Situation' are the ones which differ from this situation.

Table A.1: Comparing three different methods for pricing European options under the SABR model with no dividends

Situation		Direct	Tree	MC	95% conf.	Direct-Tree	Direct-MC	Tree-MC
$K = 90$	call	18.1610	18.1699	18.1657	± 0.0362	-0.0089	-0.0047	0.0042
	put	3.7716	3.7822	3.7815	± 0.0105	-0.0106	-0.0099	0.0007
$K = 105$	call	10.2265	10.2319	10.2293	± 0.0272	-0.0054	-0.0028	0.0026
	put	10.1056	10.1126	10.1063	± 0.0175	-0.0070	-0.0007	0.0063
$K = 110$	call	8.3492	8.3496	8.3370	± 0.0252	-0.0004	0.0122	0.0126
	put	12.9844	12.9865	12.9803	± 0.0198	-0.0021	0.0041	0.0062
$\alpha = 0.2$	call	7.6911	7.6944	7.6920	± 0.0144	-0.0033	-0.0009	0.0024
	put	2.8140	2.8188	2.8168	± 0.0070	-0.0048	-0.0028	0.0020
$\alpha = 0.8$	call	22.2896	22.3020	22.2954	± 0.0672	-0.0124	-0.0058	0.0066
	put	17.4125	17.4265	17.4156	± 0.0285	-0.0140	-0.0031	0.0109
$\beta = 0.5$ &	call	12.2605	12.2670	12.2597	± 0.0262	-0.0065	0.0008	0.0073
$\alpha = 2.5$	put	7.3834	7.3916	7.4052	± 0.0157	-0.0082	-0.0218	-0.0136
$\beta = 0.7$ &	call	12.3640	12.3717	12.3777	± 0.0275	-0.0077	-0.0137	-0.0060
$\alpha = 1$	put	7.4869	7.4962	7.4985	± 0.0154	-0.0093	-0.0116	-0.0023
$\rho = 0$	call	12.5398	12.5378	12.5415	± 0.0269	0.0020	-0.0017	-0.0037
	put	7.6627	7.6623	7.6618	± 0.0162	0.0004	0.0009	0.0005
$\nu = 0.1$	call	12.3977	12.4026	12.3942	± 0.0259	-0.0049	0.0035	0.0084
	put	7.5207	7.5272	7.5203	± 0.0152	-0.0065	0.0004	0.0069
$T = 1/12$	call	3.1135	3.1152	3.1130	± 0.0065	-0.0017	0.0005	0.0022
	put	2.6977	2.6996	2.6979	± 0.0054	-0.0019	-0.0002	0.0017

Table A.2 is an extension of table 2.4. So here are shown prices of European options under the SABR model, on an underlying which gives a discrete dividend of 2.5 euro at $t = \frac{3}{4}$, and where the interest rate is time-dependent with assumed values $r(0) = 0.0445$, $r(0.25) = 0.0395$, $r(0.5) = 0.0505$, $r(0.75) = 0.0475$ and $r(1) = 0.0425$ and the values in between are derived by linear interpolation. Again the basis situation is given by: $S_0 = 100 = K$, $\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$.

Table A.2: Pricing European options under the SABR model with non-proportional discrete dividend

Situation		Direct	Tree	MC	95% conf.	Direct-Tree	Direct-MC	Tree-MC
$K = 90$	call	16.1087	16.1198	16.1361	± 0.0311	-0.0111	-0.0274	-0.0163
	put	4.5436	4.5549	4.5600	± 0.0115	-0.0113	-0.0164	-0.0051
$K = 105$	call	8.8285	8.8327	8.8244	± 0.0255	-0.0042	0.0041	0.0083
	put	11.5998	11.6042	11.5982	± 0.0185	-0.0044	0.0016	0.0060
$K = 110$	call	7.1559	7.1539	7.1433	± 0.0235	0.0020	0.0126	0.0106
	put	14.7060	14.7042	14.6887	± 0.0206	0.0018	0.0173	0.0155
$\alpha = 0.2$	call	5.9499	5.9544	5.9541	± 0.0130	-0.0045	-0.0042	0.0003
	put	3.9424	3.9470	3.9457	± 0.0082	-0.0046	-0.0033	0.0013
$\alpha = 0.8$	call	20.6718	20.6808	20.6591	± 0.0659	-0.0090	0.0127	0.0217
	put	18.6643	18.6734	18.6684	± 0.0292	-0.0091	-0.0041	0.0050
$\beta = 0.5$ &	call	10.6988	10.7047	10.6936	± 0.0247	-0.0059	0.0052	0.0111
$\alpha = 2.5$	put	8.6913	8.6974	8.6852	± 0.0168	-0.0061	0.0061	0.0122
$\beta = 0.7$ &	call	10.7753	10.7822	10.7938	± 0.0260	-0.0069	-0.0185	-0.0116
$\alpha = 1$	put	8.7678	8.7749	8.7711	± 0.0165	-0.0071	-0.0033	0.0038
$\rho = 0$	call	10.8550	10.8545	10.8544	± 0.0252	0.0005	0.0006	0.0001
	put	8.8475	8.8473	8.8438	± 0.0172	0.0002	0.0037	0.0035
$\nu = 0.1$	call	10.7367	10.7418	10.7320	± 0.0242	-0.0051	0.0047	0.0098
	put	8.7292	8.7345	8.7273	± 0.0163	-0.0053	0.0019	0.0072

Appendix B

Results of methods for the pricing of American options

The table below (table B.1) is an extension of table 3.2. So here are shown prices of American options given by the two different methods used in this thesis to price American options under the SABR model, on an underlying that does not pay dividend. The standard situation for the table is again given by the following parameter values: $\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$, $S_0 = 100$, $K = 100$, $r = 0.05$, $T = 1$. The parameters under 'Situation' are again the ones which differ from this situation.

Table B.1: Comparing two different methods for pricing American put options under the SABR model with no dividends

Situation	Tree	LSM	95% conf.	Tree-LSM
K=90	4.0086	4.0103	± 0.0212	-0.0017
K=105	10.9333	10.9310	± 0.0331	0.0023
K=110	14.1295	14.1100	± 0.0359	0.0195
$\alpha = 0.2$	3.3853	3.3765	± 0.0133	0.0088
$\alpha = 0.8$	17.9983	18.0168	± 0.0578	-0.0185
$\beta = 0.5, \alpha = 2.5$	7.9199	7.9242	± 0.0302	-0.0043
$\beta = 0.7, \alpha = 1$	8.0420	8.0356	± 0.0296	0.0064
$\rho = 0$	8.1876	8.1667	± 0.0311	0.0209
$\nu = 0.1$	8.0457	8.0256	± 0.0285	0.0201
$T = 1/12$	2.7306	2.7255	± 0.0104	0.0051

Table B.2 is an extension of table 3.4. So here are shown prices of American options under the SABR model, on an underlying which gives a discrete dividend of 2.5 euro at $t = \frac{3}{4}$, and where the interest rate is time-dependent with assumed values $r(0) = 0.0445$, $r(0.25) = 0.0395$, $r(0.5) = 0.0505$, $r(0.75) = 0.0475$ and $r(1) = 0.0425$ and the values in between are derived by linear interpolation. The basis situation situation is again given by: $S_0 = 100 = K$, $\alpha = 0.4$, $\beta = 0.9$, $\rho = 0.3$, $\nu = 0.4$ and $T = 1$.

Table B.2: Pricing American options under the SABR model with non-proportional discrete dividend

Situation		Tree	LSM	95% conf.	Tree-LSM
K=90	call	16.6697	16.6579	± 0.0612	0.0118
	put	4.7216	4.7165	± 0.0242	0.0051
K=105	call	9.0047	8.9852	± 0.0512	0.0195
	put	12.0412	12.0499	± 0.0385	-0.0087
K=110	call	7.2630	7.2593	± 0.0484	0.0037
	put	15.2530	15.2307	± 0.0426	0.0223
$\alpha = 0.2$	call	6.3675	6.3618	± 0.0263	0.0068
	put	4.2756	4.2623	± 0.0173	0.0133
$\alpha = 0.8$	call	20.8359	20.8495	± 0.1322	-0.0136
	put	19.0275	19.0359	± 0.0618	-0.0084
$\beta = 0.5\&$	call	10.9912	10.9910	± 0.0504	0.0002
$\alpha = 2.5$	put	9.0149	9.0093	± 0.0351	0.0056
$\beta = 0.7\&$	call	11.0574	11.0475	± 0.0526	0.0099
$\alpha = 1$	put	9.1010	9.0810	± 0.0345	0.0200
$\rho = 0$	call	11.1449	11.1429	± 0.0514	0.0200
	put	9.1634	9.1468	± 0.0359	0.0166
$\nu = 0.1$	call	11.0120	11.0081	± 0.0487	0.0039
	put	9.0490	9.0414	± 0.0336	0.0076

Appendix C

Market data of American and European options on March 7th, 2007

Table C.1 shows the market data for American options on Shell. The strike prices are given plus the market prices of the call and put options described in chapter 4, so the maturities for options 1 until 4 are: 9 days, 44 days, 72 days, 653 days and 1017 days.

Table C.1: Market values of different American options on Shell at the 7th of March 2007

Options 1										
Strikes	22	23	24	24.5	25	25.5	26			
Put prices	0.0040	0.0196	0.1292	0.3136	0.6328	1.0581	1.5401			
Call prices	2.4868	1.5035	0.6136	0.2974	0.1142	0.0345	0.0080			
Options 2										
Strikes	22	23	24	24.5	25	26	26.5	27		
Put prices	0.0541	0.1407	0.3713	0.5775	0.8546	1.6055	2.0562	2.5401		
Call prices	2.6190	1.7093	0.9406	0.6453	0.4177	0.1484	0.0816	0.0424		
Options 3										
Strikes	22	23	24	24.5	25	26	27	28	30	
Put prices	0.1214	0.2699	0.5793	0.8147	1.1052	1.8310	2.6987	3.6420	5.6068	
Call prices	2.6803	1.7979	1.0608	0.7720	0.5403	0.2345	0.0881	0.0288	0.0022	
Options 4										
Strikes	19	20	21	22	22.5	23	24	25	26	28
Put prices	0.6298	0.8259	1.06737	1.3691	1.5401	1.7272	2.1522	2.6387	3.1899	4.4794
Call prices	5.8614	5.0649	4.3284	3.6608	3.3500	3.0568	2.5230	2.0542	1.6950	1.0228
Options 5										
Strikes	16	18	20	24	25	28	30	32		
Put prices	0.8612	1.2701	1.8064	3.3297	3.8076	5.4711	6.7749	8.2461		
Call prices	8.6631	7.0221	5.6433	3.5319	3.1192	2.1030	1.5884	1.1937		

Table C.2 shows the market data for European options on the AEX. Again the strike prices are given plus the market prices of the call and put options described in chapter 4, so the maturities for options 1 until 5 are: 45 days, 73 days, 101 days, 654 days and 1382 days.

Table C.2: Market values of different European options on the AEX at the 7th of March 2007

Options 1									
Strikes	460	465	470	475	480	485	490	495	500
Put prices	5.0979	6.2662	7.6936	9.3802	11.3982	13.7596	16.5153	19.6417	23.1427
Call prices	24.8513	21.0434	17.4945	14.2049	11.2465	8.6316	6.4110	4.5611	3.0859
Options 2									
Strikes	440	460	465	470	475	480	485	490	510
Put prices	4.4716	8.7204	10.2202	11.9432	13.9178	16.1970	18.7675	21.6362	36.2704
Call prices	39.6143	24.0179	20.5563	17.3181	14.3313	11.6493	9.2584	7.1659	1.9548
Options 3									
Strikes	420	440	460	470	480	490	500	520	540
Put prices	3.3339	6.1397	10.8168	14.1502	18.3694	23.6049	29.9082	45.5899	64.0636
Call prices	58.4785	41.4996	26.3922	19.8333	14.1602	9.5034	5.9145	1.8115	0.5007
Options 4									
Strikes	400	420	440	460	480	500	520	540	560
Put prices	17.0251	21.7961	27.4023	34.1627	42.0055	51.2408	61.8353	73.8143	87.25097
Call prices	91.8630	78.0169	65.0060	53.1494	42.3751	32.9934	24.9707	18.3327	13.15226
Options 5									
Strikes	360	400	440	460	480	500	520	560	600
Put prices	22.6783	33.2694	45.9641	53.1953	61.0845	69.6909	79.0728	100.2465	124.5300
Call prices	126.9736	103.1623	81.4547	71.4850	62.1730	53.5782	45.7590	32.5303	22.4114

Appendix D

Matlab codes

In this appendix a selection of the matlab codes are shown used to derive the results of these thesis.
The first code is the one to derive results for the Monte Carlo method in case of one cash dividend and time-dependent interest rate. First the main program is stated, after which the codes of the subprograms can be found.

```
function y = meanMCEUrtdiv2(a,b,ro,v,S,K,T,n,m)
% the first answer is price of call and the second
% the price of the put option.
% Same for the values of length of confidence interval.
x = payoffrtdiv2(a,b,ro,v,S,K,T,n,m);
y1 = mean(x(1,:));
y2 = mean(x(2,:));
s1 = var(x(1,:));
s2 = var(x(2,:));
conf95 = [norminv(0.975)*sqrt(s1/m) norminv(0.975)*sqrt(s2/m)]
y = [y1 y2];

function x = payoffrtdiv2(a,b,ro,v,S,K,T,n,m)
alpha1= a;
St1 = S;
alpha2 = a;
St2 = S;
c1=2.5;
t1=3/4;
ro2 = sqrt(1-ro^2);
dt = T/n;
sqrtdt = sqrt(dt);
rdt=interest2((0:dt:T)).*dt;
q1=c1*exp(intergratingr2(t1,T));
for j=1:3*n/4-1;
    A1=randn(1,m/2);
    A2=-A1;
    B1=randn(1,m/2);
    B2=-B1;
    St1 = max(0,St1+rdt(j).*St1+alpha1.*exp(-(intergratingr2((j-1)*dt,T))).*
          max(St1.*exp(intergratingr2((j-1)*dt,T))-q1,0).^b.*A1.*sqrtdt);
    alpha1 = alpha1+v.*alpha1.*(ro.*A1+ro2.*B1).*sqrtdt;
    St2 = max(0,St2+rdt(j).*St2+alpha2.*exp(-(intergratingr2((j-1)*dt,T))).*
          max(St2.*exp(intergratingr2((j-1)*dt,T))-q1,0).^b.*A2.*sqrtdt);
    alpha2 = alpha2+v.*alpha2.*(ro.*A2+ro2.*B2).*sqrtdt;
end
j=3*n/4;
    A1=randn(1,m/2);
    A2=-A1;
    B1=randn(1,m/2);
    B2=-B1;
    St1 = max(0,St1+rdt(j).*St1+alpha1.*exp(-(intergratingr2((j-1)*dt,T))).*
          max(St1.*exp(intergratingr2((j-1)*dt,T))-q1,0).^b.*A1.*sqrtdt-c1);
    alpha1 = alpha1+v.*alpha1.*(ro.*A1+ro2.*B1).*sqrtdt;
    St2 = max(0,St2+rdt(j).*St2+alpha2.*exp(-(intergratingr2((j-1)*dt,T))).*
          max(St2.*exp(intergratingr2((j-1)*dt,T))-q1,0).^b.*A2.*sqrtdt-c1);
    alpha2 = alpha2+v.*alpha2.*(ro.*A2+ro2.*B2).*sqrtdt;
for j=3*n/4+1:n;
    A1=randn(1,m/2);
    A2=-A1;
    B1=randn(1,m/2);
    B2=-B1;
    St1 = max(0,St1+rdt(j).*St1+alpha1.*exp(-(1-b)*(intergratingr2((j-1)*dt,T))).*St1.^b.*A1.*sqrtdt);
    alpha1 = alpha1+v.*alpha1.*(ro.*A1+ro2.*B1).*sqrtdt;
    St2 = max(0,St2+rdt(j).*St2+alpha2.*exp(-(1-b)*(intergratingr2((j-1)*dt,T))).*St2.^b.*A2.*sqrtdt);
    alpha2 = alpha2+v.*alpha2.*(ro.*A2+ro2.*B2).*sqrtdt;
end
x1 = exp(-intergratingr2(0,T)).*max([St1 St2]-K,0);
x2 = exp(-intergratingr2(0,T)).*max(K-[St1 St2],0);
x = [x1 ; x2];

function y=interest2(s)
x=[0 1/4 1/2 3/4 1];
z=[0.0445 0.0395 0.0505 0.0475 0.0425];
y=interp1(x,z,s);

function y=intergratingr2(t1,t2)
y=quad(@(s) interest2(s),t1,t2);
```

The matlab code below is to price European options, with constant interest rate and no dividends involved, with the tree method described in this thesis. Again, first the main program is given, after which the subprograms can be found.

```
function y=VNEUnew(c,a,b,rho,nu,S,K,r,T,n)
% c=1 is call option, c=-1 is put option
dt=T/n;
rdt=r*dt;
sqrtdt=sqrt(dt);
disc=exp(-rdt);
amax=zeros(1,n+1);
amin=zeros(1,n+1);
amax(1)=a;
amin(1)=a;
smax=zeros(1,n+1);
smin=zeros(1,n+1);
smax(1)=S;
smin(1)=S;
rtT=exp(-r.*(1-b).*(T-(0:n-1).*dt));
f=exp(r*T)*S;
f1=exp(r*T/2)*S;
oldopts=optimset('fzero');
options=optimset(oldopts,'TolX',0.01);
bounds1=fzero(@(y) quad(@(k) diff2v(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.002,[f1*0.99 1e6*S-1],
        options);
bounds2=fzero(@(y) quad(@(k) diff2v(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.002,[f*0.99 1e6*S-1],
        options);
d2=(bounds2-bounds1)/(n/2);
bounds(1:n/2+1)=bounds1;
bounds(n/2+2:n+1)=bounds1+d2:d2:bounds2;
bounda1=logninv(0.9999,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
bounda2=logninv(0.9999,log(a)-1/2*nu^2*T,nu*sqrt(T));
d1=(bounda2-bounda1)/(n/2);
bounda(1:n/2+1)=bounda1;
bounda(n/2+2:n+1)=bounda1+d1:d1:bounda2;
for k=1:n
    amax(k+1)=min(amax(k)+nu*amax(k)*sqrtdt,bounda(k+1));
    amin(k+1)=max(amin(k)-nu*amin(k)*sqrtdt,0);
    smax(k+1)=min(smax(k)+rdt*smax(k)+amax(k)*rtT(k)*smax(k)^b*sqrtdt,bounds(k+1));
    smin(k+1)=max(smin(k)+rdt*smin(k)-amax(k)*rtT(k)*smin(k)^b*sqrtdt,0);
end
e=0.00001;
oldopts=optimset('fzero');
options=optimset(oldopts,'TolX',0.0001);
S2p002=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),e,y,0.0001)-0.02,[2*e f-e],options);
S2p01=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),e,y,0.0001)-0.1,[S2p002 1.2*f],options);
S2p025=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),e,y,0.0001)-0.25,[S2p01 1.2*f],options);
S2p04=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),e,y,0.0001)-0.4,[S2p025 1.3*f],options);
S2p06=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.4,[S2p04 1e6*S-1],
      options);
S2p075=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.25,[S2p06 1e6*S-1],
       options);
S2p09=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.1,[S2p075 1e6*S-1],
      options);
S2p098=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.02,[S2p09 1e6*S-1],
       options);
S2p099=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.01,[S2p098 1e6*S-1],
       options);
S2p0998=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f,k,T),y,1e6*S,0.0001)-0.002,[S2p099 1e6*S-1],
        options);
rs=exp(r/2);
```

```
S1p002=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),e,y,0.0001)-0.02,[2*e f-e],options);
S1p01=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),e,y,0.0001)-0.1,[S1p002 f-e],options);
S1p025=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),e,y,0.0001)-0.25,[S1p01 1.1*f],options);
S1p04=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),e,y,0.0001)-0.4,[S1p025 1.2*f],options);
S1p06=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.4,[S1p04 rs*S2p06],
      options);
S1p075=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.25,[S1p06 rs*S2p075],
       options);
S1p09=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.1,[S1p075 rs*S2p09],
      options);
S1p098=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.02,[S1p09 rs*S2p098],
       options);
S1p099=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.01,[S1p098 rs*S2p099],
       options);
S1p0998=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,f1,k,T/2),y,1e6*S,0.0001)-0.002,[S1p099 rs*S2p0998],
        options);
Sp002=max([S:(S1p002-S)/(n/2):S1p002 S1p002+(S2p002-S1p002)/(n/2):(S2p002-S1p002)/(n/2):S2p002],smin);
Sp01=max([S:(S1p01-S)/(n/2):S1p01 S1p01+(S2p01-S1p01)/(n/2):(S2p01-S1p01)/(n/2):S2p01],smin);
Sp025=max([S:(S1p025-S)/(n/2):S1p025 S1p025+(S2p025-S1p025)/(n/2):(S2p025-S1p025)/(n/2):S2p025],smin);
Sp04=max([S:(S1p04-S)/(n/2):S1p04 S1p04+(S2p04-S1p04)/(n/2):(S2p04-S1p04)/(n/2):S2p04],smin);
Sp06=min([S:(S1p06-S)/(n/2):S1p06 S1p06+(S2p06-S1p06)/(n/2):(S2p06-S1p06)/(n/2):S2p06],smax);
Sp075=min([S:(S1p075-S)/(n/2):S1p075 S1p075+(S2p075-S1p075)/(n/2):(S2p075-S1p075)/(n/2):S2p075],smax);
Sp09=min([S:(S1p09-S)/(n/2):S1p09 S1p09+(S2p09-S1p09)/(n/2):(S2p09-S1p09)/(n/2):S2p09],smax);
Sp098=min([S:(S1p098-S)/(n/2):S1p098 S1p098+(S2p098-S1p098)/(n/2):(S2p098-S1p098)/(n/2):S2p098],smax);
Sp099=min([S:(S1p099-S)/(n/2):S1p099 S1p099+(S2p099-S1p099)/(n/2):(S2p099-S1p099)/(n/2):S2p099],smax);
Sp0998=min([S:(S1p0998-S)/(n/2):S1p0998 S1p0998+(S2p0998-S1p0998)/(n/2):(S2p0998-S1p0998)/(n/2):S2p0998],
       smax);
ds7=max((smax-Sp0998)./10,e);
ds6=max((Sp0998-Sp099)./10,e);
ds5=max((Sp099-Sp098+Sp002-smin)./10,e);
ds4=max((Sp098-Sp09+Sp01-Sp002)./10,e);
ds3=max((Sp09-Sp075+Sp025-Sp01)./10,e);
ds2=max((Sp075-Sp06+Sp04-Sp025)./15,e);
ds1=(Sp06-Sp04)./20;
a1p002=logninv(0.02,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p01=logninv(0.1,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p025=logninv(0.25,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p04=logninv(0.4,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p06=logninv(0.6,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p075=logninv(0.75,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p09=logninv(0.9,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p098=logninv(0.98,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a2p002=logninv(0.02,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p01=logninv(0.1,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p025=logninv(0.25,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p04=logninv(0.4,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p06=logninv(0.6,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p075=logninv(0.75,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p09=logninv(0.9,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p098=logninv(0.98,log(a)-1/2*nu^2*T,nu*sqrt(T));
ap002=max([a:(a1p002-a)/(n/2):a1p002 a1p002+(a2p002-a1p002)/(n/2):(a2p002-a1p002)/(n/2):a2p002],amin);
ap01=max([a:(a1p01-a)/(n/2):a1p01 a1p01+(a2p01-a1p01)/(n/2):(a2p01-a1p01)/(n/2):a2p01],amin);
ap025=max([a:(a1p025-a)/(n/2):a1p025 a1p025+(a2p025-a1p025)/(n/2):(a2p025-a1p025)/(n/2):a2p025],amin);
ap04=max([a:(a1p04-a)/(n/2):a1p04 a1p04+(a2p04-a1p04)/(n/2):(a2p04-a1p04)/(n/2):a2p04],amin);
ap06=min([a:(a1p06-a)/(n/2):a1p06 a1p06+(a2p06-a1p06)/(n/2):(a2p06-a1p06)/(n/2):a2p06],amax);
ap075=min([a:(a1p075-a)/(n/2):a1p075 a1p075+(a2p075-a1p075)/(n/2):(a2p075-a1p075)/(n/2):a2p075],amax);
ap09=min([a:(a1p09-a)/(n/2):a1p09 a1p09+(a2p09-a1p09)/(n/2):(a2p09-a1p09)/(n/2):a2p09],amax);
ap098=min([a:(a1p098-a)/(n/2):a1p098 a1p098+(a2p098-a1p098)/(n/2):(a2p098-a1p098)/(n/2):a2p098],amax);
da5=max((amax-ap098+ap002-amin)./4,e);
da4=max((ap098-ap09+ap01-ap002)./4,e);
da3=max((ap09-ap075+ap025-ap01)./4,e);
```

```
da2=max((ap075-ap06+ap04-ap025)./6,e);
da1=(ap06-ap04)./8;
[X Y]=meshgrid([amin(n+1):da5(n+1):(ap002(n+1)-e) ap002(n+1):da4(n+1):(ap01(n+1)-e)
      ap01(n+1):da3(n+1):(ap025(n+1)-e) ap025(n+1):da2(n+1):(ap04(n+1)-e) ap04(n+1):da1(n+1):
      (ap06(n+1)-e) ap06(n+1):da2(n+1):(ap075(n+1)-e) ap075(n+1):da3(n+1):(ap09(n+1)-e)
      ap09(n+1):da4(n+1):(ap098(n+1)-e) ap098(n+1):da5(n+1):(amax(n+1)-e) amax(n+1)],
      [smin(n+1):ds5(n+1):(Sp002(n+1)-e) Sp002(n+1):ds4(n+1):(Sp01(n+1)-e) Sp01(n+1):ds3(n+1):
      (Sp025(n+1)-e) Sp025(n+1):ds2(n+1):(Sp04(n+1)-e) Sp04(n+1):ds1(n+1):(Sp06(n+1)-e)
      Sp06(n+1):ds2(n+1):(Sp075(n+1)-e) Sp075(n+1):ds3(n+1):(Sp09(n+1)-e) Sp09(n+1):ds4(n+1):
      (Sp098(n+1)-e) Sp098(n+1):ds5(n+1):(Sp099(n+1)-e) Sp099(n+1):ds6(n+1):(Sp0998(n+1)-e)
      Sp0998(n+1):ds7(n+1):smax(n+1)-e smax(n+1)]);
f=max(c.*(Y-K),0);
p1=(1+rho)/4;
p2=(1-rho)/4;
for l=n:-1:2
    X1=X;
    Y1=Y;
    [X Y]=meshgrid([amin(l):da5(l):(ap002(l)-e) ap002(l):da4(l):(ap01(l)-e) ap01(l):da3(l):(ap025(l)-e)
          ap025(l):da2(l):(ap04(l)-e) ap04(l):da1(l):(ap06(l)-e) ap06(l):da2(l):(ap075(l)-e)
          ap075(l):da3(l):(ap09(l)-e) ap09(l):da4(l):(ap098(l)-e) ap098(l):da5(l):(amax(l)-e) amax(l)],
          [smin(l):ds5(l):(Sp002(l)-e) Sp002(l):ds4(l):(Sp01(l)-e) Sp01(l):ds3(l):(Sp025(l)-e)
          Sp025(l):ds2(l):(Sp04(l)-e) Sp04(l):ds1(l):(Sp06(l)-e) Sp06(l):ds2(l):(Sp075(l)-e)
          Sp075(l):ds3(l):(Sp09(l)-e) Sp09(l):ds4(l):(Sp098(l)-e) Sp098(l):ds5(l):(Sp099(l)-e)
          Sp099(l):ds6(l):(Sp0998(l)-e) Sp0998(l):ds7(l):(smax(l)-e) smax(l)]);
    Z11=min(X+nu.*X.*sqrtdt,bounda(l+1));
    Z12=max(X-nu.*X.*sqrtdt,0);
    Z21=min(Y+rdt.*Y+X.*rtT(l).*Y.^b.*sqrtdt,bounds(l+1));
    Z22=max(min(Y+rdt.*Y-X.*rtT(l).*Y.^b.*sqrtdt,bounds(l+1)),0);
    f=disc.*(p1.*interp2(X1,Y1,f,Z11,Z21,'cubic')+p2.*interp2(X1,Y1,f,Z11,Z22,'cubic')+
      p2.*interp2(X1,Y1,f,Z12,Z21,'cubic')+p1.*interp2(X1,Y1,f,Z12,Z22,'cubic'));
end
Z1a=a+nu*sqrtdt*a;
Z2a=a-nu*sqrtdt*a;
Z1s=S+rdt*S+a*S^b*rtT(1)*sqrtdt;
Z2s=S+rdt*S-a*S^b*rtT(1)*sqrtdt;
y=disc*(p1*interp2(X,Y,f,Z1a,Z1s,'cubic')+p2*interp2(X,Y,f,Z1a,Z2s,'cubic')+
  p2*interp2(X,Y,f,Z2a,Z1s,'cubic')+p1*interp2(X,Y,f,Z2a,Z2s,'cubic'));

function y=diff2vput(alpha,beta,rho,v,f,K,tt)
n=length(K);
y=zeros(1,n);
for j=1:n
    k=K(j);
    sigma=sigma_imp(alpha,beta,rho,v,f,k,tt);
    d1=(log(f/k)+sigma^2*tt/2)/(sigma*sqrt(tt));
    d2=d1-sigma*sqrt(tt);
    nd1=1/sqrt(2*pi)*exp(-d1^2/2);
    nd2=1/sqrt(2*pi)*exp(-d2^2/2);
    z1= first derivative of sigma to K (too long to state the formula here)
    z2= second derivative of sigma to K ( too long to state the formula here)
    d1sb1=log(k/f)/(sigma^2*sqrt(tt))+sqrt(tt)/2;
    d2sb1=d1sb1-sqrt(tt);
    d1K1=-1/(k*sqrt(tt)*sigma)+d1sb1*z1;
    d2K1=-1/(k*sqrt(tt)*sigma)+d2sb1*z1;
    dsb2=2*(log(f/k))/(sigma^3*sqrt(tt));
    d1K2=1/(k^2*sigma*sqrt(tt))+1/(k*sigma^2*sqrt(tt))*z1+d1sb1*z2+z1*(-1/(k*sigma^2*sqrt(tt))+dsb2*z1);
    d2K2=d1K2-sqrt(tt)*z2;
    y(j)=max((-f*d1*nd1*d1K1^2+f*nd1*d1K2-2*nd2*d2K1+k*d2*nd2*d2K1^2-k*nd2*d2K2),0);
end
```

The program of diff2v is the same as diff2vput, except that $y(j)$ is multiplied by k.

```
function x=sigma_imp(a,b,ro,v,f,K,T)
z= (v/a).*(f.*K).^((1-b)/2).*log(f./K);
y1 = log((sqrt(1-(2*ro).*z+z.^2)+z-ro)./(1-ro));
y2=(1-b).*log(f./K);
y3=(f.*K).^(1-b);
x = a./(sqrt(y3).*(y2.^0+y2.^2./24+y2.^4./1920)).*(z./y1).*(1+(((1-b)^2/24*a^2)./y3+
 (ro*b*v*a)./(4.*sqrt(y3))+(2-3*ro^2)/24*v^2).*T);
```

The matlab code below is to price American options, on an underlying which pays cash dividend (can be any number of dividends), and where the interest rate is time-dependent, with the tree method described in this thesis. Again the subprograms can be found at the end.

```
dt=T/n;
rdt=interest((0:dt:T)).*dt;
sqrtdt=sqrt(dt);
amax=zeros(1,n+1);
amin=zeros(1,n+1);
amax(1)=a;
amin(1)=a;
smax=zeros(1,n+1);
smin=zeros(1,n+1);
smax(1)=S;
smin(1)=S;
mk=length(c1);
jl=ones(1,mk);
for i=1:mk
while jl(i)*dt<t1(i)
    jl(i)=jl(i)+1;
end
end
l=0;
j=jl;
for i=1:mk-1
    if  jl(i) == jl(i+1)
        c1(i-l)=c1(i-l)+c1(i-l+1);
        c1(i-l+1:mk-1-l)=c1(i-l+2:mk-l);
        t1(i+1-l:mk-1-l)=t1(i+2-l:mk-l);
        j(i+1-l:mk-1-l)=j(i+2-l:mk-l);
        l=l+1;
    end
end
c1=c1(1:mk-l);
j=j(1:mk-l);
lk=length(c1);
q1=zeros(1,lk);
for i=1:lk
    q1(i)=c1(i)*exp(intergratingr(t1(i),T));
end
m=1;
if t1(lk)<T/2
    m=lk+1;
else
while t1(m)<T/2
    m=m+1;
end
end
if t1(1)>T/2
    q2=0;
else
    q2=zeros(1,m-1);
    for i=1:m-1
    q2(i)=c1(i)*exp(intergratingr(t1(i),T/2));
    end
end
F=S*exp(intergratingr(0,T))-sum(q1);
F1=S*exp(intergratingr(0,T/2))-sum(q2);
e=0.00001;
oldopts=optimset('fzero');
options=optimset(oldopts,'TolX',0.01);
```

```
bounds1=fzero(@(y) quad(@(k) diff2v(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.002,[F1*0.99 1e6*S-1]
,options);
bounds2=fzero(@(y) quad(@(k) diff2v(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.002,[F*0.99 1e6*S-1]
,options);
d2=(bounds2+sum(c1(m:lk))-bounds1)/(n/2);
bounds(1:n/2+1)=bounds1+sum(c1(1:m-1));
if m>lk
    bounds(n/2+2:n+1)=bounds1+d2:d2:bounds2;
else
    bounds(n/2+2:j(m))=bounds1+d2:d2:bounds1+d2*(j(m)-n/2-1);
    for i=m:lk-1
    bounds(j(i)+1:j(i+1))=bounds1+d2*(j(i)-n/2)-sum(c1(m:i)):d2:bounds1+d2*(j(i+1)-n/2-1)-sum(c1(m:i));
    end
    for i=j(lk)+1:n+1
    bounds(j(lk)+1:n+1)=bounds1+d2*(j(lk)-n/2)-sum(c1(m:lk)):d2:boundsd2;
    end
end
bounda1=logninv(0.9999,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
bounda2=logninv(0.9999,log(a)-1/2*nu^2*T,nu*sqrt(T));
d1=(bounda2-bounda1)/(n/2);
bounda(1:n/2+1)=bounda1;
bounda(n/2+2:n+1)=bounda1+d1:d1:bounda2;
for k=1:j(1)-1
    amax(k+1)=min(amax(k)+nu*amax(k)*sqrtdt,bounda(k+1));
    amin(k+1)=max(amin(k)-nu*amin(k)*sqrtdt,0);
    smax(k+1)=min(smax(k)+rdt(k)*smax(k)+amax(k)*exp(-intergratingr((k-1)*dt,T))*
             (smax(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(1:lk)))^b*sqrtdt,bounds(k+1));
    smin(k+1)=max(smin(k)+rdt(k)*smin(k)-amax(k)*exp(-intergratingr((k-1)*dt,T))*
              max(smin(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(1:lk)),0)^b*sqrtdt,0);
end
for i=1:lk-1
    k=j(i);
    amax(k+1)=min(amax(k)+nu*amax(k)*sqrtdt,bounda(k+1));
    amin(k+1)=max(amin(k)-nu*amin(k)*sqrtdt,0);
    smax(k+1)=min(smax(k)+rdt(k)*smax(k)+amax(k)*exp(-intergratingr((k-1)*dt,T))*
              (smax(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(i:lk)))^b*sqrtdt-c1(i),bounds(k+1));
    smin(k+1)=max(smin(k)+rdt(k)*smin(k)-amax(k)*exp(-intergratingr((k-1)*dt,T))*
              max(smin(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(i:lk)),0)^b*sqrtdt-c1(i),0);
    for k=j(i)+1:j(i+1)-1
    amax(k+1)=min(amax(k)+nu*amax(k)*sqrtdt,bounda(k+1));
    amin(k+1)=max(amin(k)-nu*amin(k)*sqrtdt,0);
    smax(k+1)=min(smax(k)+rdt(k)*smax(k)+amax(k)*exp(-intergratingr((k-1)*dt,T))*
              (smax(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(i+1:lk)))^b*sqrtdt,bounds(k+1));
    smin(k+1)=max(smin(k)+rdt(k)*smin(k)-amax(k)*exp(-intergratingr((k-1)*dt,T))*
              max(smin(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(i+1:lk)),0)^b*sqrtdt,0);
    end
end
k=j(lk);
    amax(k+1)=min(amax(k)+nu*amax(k)*sqrtdt,bounda(k+1));
    amin(k+1)=max(amin(k)-nu*amin(k)*sqrtdt,0);
    smax(k+1)=min(smax(k)+rdt(k)*smax(k)+amax(k)*exp(-intergratingr((k-1)*dt,T))*
              (smax(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(lk)))^b*sqrtdt-c1(lk),bounds(k+1));
    smin(k+1)=max(smin(k)+rdt(k)*smin(k)-amax(k)*exp(-intergratingr((k-1)*dt,T))*
              max(smin(k)/exp(-intergratingr((k-1)*dt,T))-sum(q1(lk)),0)^b*sqrtdt-c1(lk),0);
for k=j(lk)+1:n
    amax(k+1)=min(amax(k)+nu*amax(k)*sqrtdt,bounda(k+1));
    amin(k+1)=max(amin(k)-nu*amin(k)*sqrtdt,0);
    smax(k+1)=min(smax(k)+rdt(k)*smax(k)+amax(k)*exp(-intergratingr((k-1)*dt,T))^(1-b)*
              smax(k)^b*sqrtdt,bounds(k+1));
    smin(k+1)=max(smin(k)+rdt(k)*smin(k)-amax(k)*exp(-intergratingr((k-1)*dt,T))^(1-b)*
              smin(k)^b*sqrtdt,0);
```

```
end
oldopts=optimset('fzero');
options=optimset(oldopts,'TolX',0.0001);
S2p002=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),e,y,0.0001)-0.02,[2*e F-e],options);
S2p01=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),e,y,0.0001)-0.1,[S2p002 1.2*F],options);
S2p025=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),e,y,0.0001)-0.25,[S2p01 1.2*F],options);
S2p04=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),e,y,0.0001)-0.4,[S2p025 1.3*F],options);
S2p06=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.4,[S2p04 1e6*S-1],options);
S2p075=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.25,[S2p06 1e6*S-1],options);
S2p09=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.1,[S2p075 1e6*S-1],options);
S2p098=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.02,[S2p09 1e6*S-1],options);
S2p099=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.01,[S2p098 1e6*S-1],options);
S2p0998=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F,k,T),y,1e6*S,0.0001)-0.002,[S2p099 1e6*S-1],options);
r=exp(intergratingr(T/2,T));
S1p002=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),e,y,0.0001)-0.02,[0.6*S2p002 F-e],options);
S1p01=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),e,y,0.0001)-0.1,[S1p002 F-e],options);
S1p025=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),e,y,0.0001)-0.25,[S1p01 1.1*F],options);
S1p04=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),e,y,0.0001)-0.4,[S1p025 1.2*F],options);
S1p06=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.4,
      [S1p04 r*(S2p06+sum(c1(m:lk)))],options);
S1p075=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.25,
       [S1p06 r*(S2p075+sum(c1(m:lk)))],options);
S1p09=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.1,
      [S1p075 r*(S2p09+sum(c1(m:lk)))],options);
S1p098=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.02,
       [S1p09 1.1*r*(S2p098+sum(c1(m:lk)))],options);
S1p099=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.01,
       [S1p098 1.1*r*(S2p099+sum(c1(m:lk)))],options);
S1p0998=fzero(@(y) quad(@(k) diff2vput(a,b,rho,nu,F1,k,T/2),y,1e6*S,0.0001)-0.002,
        [S1p099 2*r*(S2p0998+sum(c1(m:lk)))],options);
if m<2
    Sp002(1:n/2+1)=max(S:(S1p002-S)/(n/2):S1p002,smin(1:n/2+1));
    Sp01(1:n/2+1)=max(S:(S1p01-S)/(n/2):S1p01,smin(1:n/2+1));
    Sp025(1:n/2+1)=max(S:(S1p025-S)/(n/2):S1p025,smin(1:n/2+1));
    Sp04(1:n/2+1)=max(S:(S1p04-S)/(n/2):S1p04,smin(1:n/2+1));
    Sp06(1:n/2+1)=min(S:(S1p06-S)/(n/2):S1p06,smax(1:n/2+1));
    Sp075(1:n/2+1)=min(S:(S1p075-S)/(n/2):S1p075,smax(1:n/2+1));
    Sp09(1:n/2+1)=min(S:(S1p09-S)/(n/2):S1p09,smax(1:n/2+1));
    Sp098(1:n/2+1)=min(S:(S1p098-S)/(n/2):S1p098,smax(1:n/2+1));
    Sp099(1:n/2+1)=min(S:(S1p099-S)/(n/2):S1p099,smax(1:n/2+1));
    Sp0998(1:n/2+1)=min(S:(S1p0998-S)/(n/2):S1p0998,smax(1:n/2+1));
else
    d2002=(S1p002+sum(c1(1:m-1))-S)/(n/2);
    Sp002(1:j(1))=max(S:d2002:S+d2002*(j(1)-1),smin(1:j(1)));
    d201=(S1p01+sum(c1(1:m-1))-S)/(n/2);
    Sp01(1:j(1))=max(S:d201:S+d201*(j(1)-1),smin(1:j(1)));
    d2025=(S1p025+sum(c1(1:m-1))-S)/(n/2);
    Sp025(1:j(1))=max(S:d2025:S+d2025*(j(1)-1),smin(1:j(1)));
    d204=(S1p04+sum(c1(1:m-1))-S)/(n/2);
    Sp04(1:j(1))=max(S:d204:S+d204*(j(1)-1),smin(1:j(1)));
    d206=(S1p06+sum(c1(1:m-1))-S)/(n/2);
    Sp06(1:j(1))=min(S:d206:S+d206*(j(1)-1),smax(1:j(1)));
    d2075=(S1p075+sum(c1(1:m-1))-S)/(n/2);
    Sp075(1:j(1))=min(S:d2075:S+d2075*(j(1)-1),smax(1:j(1)));
    d209=(S1p09+sum(c1(1:m-1))-S)/(n/2);
    Sp09(1:j(1))=min(S:d209:S+d209*(j(1)-1),smax(1:j(1)));
    d2098=(S1p098+sum(c1(1:m-1))-S)/(n/2);
    Sp098(1:j(1))=min(S:d2098:S+d2098*(j(1)-1),smax(1:j(1)));
    d2099=(S1p099+sum(c1(1:m-1))-S)/(n/2);
    Sp099(1:j(1))=min(S:d2099:S+d2099*(j(1)-1),smax(1:j(1)));
    d20998=(S1p0998+sum(c1(1:m-1))-S)/(n/2);
    Sp0998(1:j(1))=min(S:d20998:S+d20998*(j(1)-1),smax(1:j(1)));
    for i=1:m-2
    Sp002(j(i)+1:j(i+1))=max(S+d2002*(j(i))-sum(c1(1:i)):d2002:S+d2002*(j(i+1)-1)-sum(c1(1:i)),
                         smin(j(i)+1:j(i+1)));
    Sp01(j(i)+1:j(i+1))=max(S+d201*(j(i))-sum(c1(1:i)):d201:S+d201*(j(i+1)-1)-sum(c1(1:i)),
                        smin(j(i)+1:j(i+1)));
    Sp025(j(i)+1:j(i+1))=max(S+d2025*(j(i))-sum(c1(1:i)):d2025:S+d2025*(j(i+1)-1)-sum(c1(1:i)),
                         smin(j(i)+1:j(i+1)));
    Sp04(j(i)+1:j(i+1))=max(S+d204*(j(i))-sum(c1(1:i)):d204:S+d204*(j(i+1)-1)-sum(c1(1:i)),
                        smin(j(i)+1:j(i+1)));
    Sp06(j(i)+1:j(i+1))=min(S+d206*(j(i))-sum(c1(1:i)):d206:S+d206*(j(i+1)-1)-sum(c1(1:i)),
                        smax(j(i)+1:j(i+1)));
    Sp075(j(i)+1:j(i+1))=min(S+d2075*(j(i))-sum(c1(1:i)):d2075:S+d2075*(j(i+1)-1)-sum(c1(1:i)),
                         smax(j(i)+1:j(i+1)));
    Sp09(j(i)+1:j(i+1))=min(S+d209*(j(i))-sum(c1(1:i)):d209:S+d209*(j(i+1)-1)-sum(c1(1:i)),
                        smax(j(i)+1:j(i+1)));
    Sp098(j(i)+1:j(i+1))=min(S+d2098*(j(i))-sum(c1(1:i)):d2098:S+d2098*(j(i+1)-1)-sum(c1(1:i)),
                         smax(j(i)+1:j(i+1)));
    Sp099(j(i)+1:j(i+1))=min(S+d2099*(j(i))-sum(c1(1:i)):d2099:S+d2099*(j(i+1)-1)-sum(c1(1:i)),
                         smax(j(i)+1:j(i+1)));
    Sp0998(j(i)+1:j(i+1))=min(S+d20998*(j(i))-sum(c1(1:i)):d20998:S+d20998*(j(i+1)-1)-sum(c1(1:i)),
                          smax(j(i)+1:j(i+1)));
    end
    Sp002(j(m-1)+1:n/2+1)=max(S+d2002*(j(m-1))-sum(c1(1:m-1)):d2002:S1p002,smin(j(m-1)+1:n/2+1));
    Sp01(j(m-1)+1:n/2+1)=max(S+d201*(j(m-1))-sum(c1(1:m-1)):d201:S1p01,smin(j(m-1)+1:n/2+1));
    Sp025(j(m-1)+1:n/2+1)=max(S+d2025*(j(m-1))-sum(c1(1:m-1)):d2025:S1p025,smin(j(m-1)+1:n/2+1));
    Sp04(j(m-1)+1:n/2+1)=max(S+d204*(j(m-1))-sum(c1(1:m-1)):d204:S1p04,smin(j(m-1)+1:n/2+1));
    Sp06(j(m-1)+1:n/2+1)=min(S+d206*(j(m-1))-sum(c1(1:m-1)):d206:S1p06,smax(j(m-1)+1:n/2+1));
    Sp075(j(m-1)+1:n/2+1)=min(S+d2075*(j(m-1))-sum(c1(1:m-1)):d2075:S1p075,smax(j(m-1)+1:n/2+1));
    Sp09(j(m-1)+1:n/2+1)=min(S+d209*(j(m-1))-sum(c1(1:m-1)):d209:S1p09,smax(j(m-1)+1:n/2+1));
    Sp098(j(m-1)+1:n/2+1)=min(S+d2098*(j(m-1))-sum(c1(1:m-1)):d2098:S1p098,smax(j(m-1)+1:n/2+1));
    Sp099(j(m-1)+1:n/2+1)=min(S+d2099*(j(m-1))-sum(c1(1:m-1)):d2099:S1p099,smax(j(m-1)+1:n/2+1));
    Sp0998(j(m-1)+1:n/2+1)=min(S+d20998*(j(m-1))-sum(c1(1:m-1)):d20998:S1p0998,smax(j(m-1)+1:n/2+1));
end
    d2002=(S2p002+sum(c1(m:lk))-S1p002)/(n/2);
    d201=(S2p01+sum(c1(m:lk))-S1p01)/(n/2);
    d2025=(S2p025+sum(c1(m:lk))-S1p025)/(n/2);
    d204=(S2p04+sum(c1(m:lk))-S1p04)/(n/2);
    d206=(S2p06+sum(c1(m:lk))-S1p06)/(n/2);
    d2075=(S2p075+sum(c1(m:lk))-S1p075)/(n/2);
    d209=(S2p09+sum(c1(m:lk))-S1p09)/(n/2);
    d2098=(S2p098+sum(c1(m:lk))-S1p098)/(n/2);
    d2099=(S2p099+sum(c1(m:lk))-S1p099)/(n/2);
    d20998=(S2p0998+sum(c1(m:lk))-S1p0998)/(n/2);
if m>lk
    Sp002(n/2+2:n+1)=max(S1p002+d2002:d2002:S2p002,smin(n/2+2:n+1));
    Sp01(n/2+2:n+1)=max(S1p01+d201:d201:S2p01,smin(n/2+2:n+1));
    Sp025(n/2+2:n+1)=max(S1p025+d2025:d2025:S2p025,smin(n/2+2:n+1));
    Sp04(n/2+2:n+1)=max(S1p04+d204:d204:S2p04,smin(n/2+2:n+1));
    Sp06(n/2+2:n+1)=min(S1p06+d206:d206:S2p06,smax(n/2+2:n+1));
    Sp075(n/2+2:n+1)=min(S1p075+d2075:d2075:S2p075,smax(n/2+2:n+1));
    Sp09(n/2+2:n+1)=min(S1p09+d209:d209:S2p09,smax(n/2+2:n+1));
    Sp098(n/2+2:n+1)=min(S1p098+d2098:d2098:S2p098,smax(n/2+2:n+1));
    Sp099(n/2+2:n+1)=min(S1p099+d2099:d2099:S2p099,smax(n/2+2:n+1));
    Sp0998(n/2+2:n+1)=min(S1p0998+d20998:d20998:S2p0998,smax(n/2+2:n+1));
else
    Sp002(n/2+2:j(m))=max(S1p002+d2002:d2002:S1p002+d2002*(j(m)-n/2-1),smin(n/2+2:j(m)));
    Sp01(n/2+2:j(m))=max(S1p01+d201:d201:S1p01+d201*(j(m)-n/2-1),smin(n/2+2:j(m)));
    Sp025(n/2+2:j(m))=max(S1p025+d2025:d2025:S1p025+d2025*(j(m)-n/2-1),smin(n/2+2:j(m)));
    Sp04(n/2+2:j(m))=max(S1p04+d204:d204:S1p04+d204*(j(m)-n/2-1),smin(n/2+2:j(m)));
    Sp06(n/2+2:j(m))=min(S1p06+d206:d206:S1p06+d206*(j(m)-n/2-1),smax(n/2+2:j(m)));
    Sp075(n/2+2:j(m))=min(S1p075+d2075:d2075:S1p075+d2075*(j(m)-n/2-1),smax(n/2+2:j(m)));
    Sp09(n/2+2:j(m))=min(S1p09+d209:d209:S1p09+d209*(j(m)-n/2-1),smax(n/2+2:j(m)));
    Sp098(n/2+2:j(m))=min(S1p098+d2098:d2098:S1p098+d2098*(j(m)-n/2-1),smax(n/2+2:j(m)));
    Sp099(n/2+2:j(m))=min(S1p099+d2099:d2099:S1p099+d2099*(j(m)-n/2-1),smax(n/2+2:j(m)));
    Sp0998(n/2+2:j(m))=min(S1p0998+d20998:d20998:S1p0998+d20998*(j(m)-n/2-1),smax(n/2+2:j(m)));
    for i=m:lk-1
    Sp002(j(i)+1:j(i+1))=max(S1p002+d2002*(j(i)-n/2)-sum(c1(m:i)):d2002:S1p002+d2002*(j(i+1)-n/2-1)
                         -sum(c1(m:i)),smin(j(i)+1:j(i+1)));
    Sp01(j(i)+1:j(i+1))=max(S1p01+d201*(j(i)-n/2)-sum(c1(m:i)):d201:S1p01+d201*(j(i+1)-n/2-1)
                        -sum(c1(m:i)),smin(j(i)+1:j(i+1)));
    Sp025(j(i)+1:j(i+1))=max(S1p025+d2025*(j(i)-n/2)-sum(c1(m:i)):d2025:S1p025+d2025*(j(i+1)-n/2-1)
                         -sum(c1(m:i)),smin(j(i)+1:j(i+1)));
    Sp04(j(i)+1:j(i+1))=max(S1p04+d204*(j(i)-n/2)-sum(c1(m:i)):d204:S1p04+d204*(j(i+1)-n/2-1)
                        -sum(c1(m:i)),smin(j(i)+1:j(i+1)));
    Sp06(j(i)+1:j(i+1))=min(S1p06+d206*(j(i)-n/2)-sum(c1(m:i)):d206:S1p06+d206*(j(i+1)-n/2-1)
                        -sum(c1(m:i)),smax(j(i)+1:j(i+1)));
    Sp075(j(i)+1:j(i+1))=min(S1p075+d2075*(j(i)-n/2)-sum(c1(m:i)):d2075:S1p075+d2075*(j(i+1)-n/2-1)
                         -sum(c1(m:i)),smax(j(i)+1:j(i+1)));
    Sp09(j(i)+1:j(i+1))=min(S1p09+d209*(j(i)-n/2)-sum(c1(m:i)):d209:S1p09+d209*(j(i+1)-n/2-1)
                        -sum(c1(m:i)),smax(j(i)+1:j(i+1)));
    Sp098(j(i)+1:j(i+1))=min(S1p098+d2098*(j(i)-n/2)-sum(c1(m:i)):d2098:S1p098+d2098*(j(i+1)-n/2-1)
                         -sum(c1(m:i)),smax(j(i)+1:j(i+1)));
    Sp099(j(i)+1:j(i+1))=min(S1p099+d2099*(j(i)-n/2)-sum(c1(m:i)):d2099:S1p099+d2099*(j(i+1)-n/2-1)
                         -sum(c1(m:i)),smax(j(i)+1:j(i+1)));
    Sp0998(j(i)+1:j(i+1))=min(S1p0998+d20998*(j(i)-n/2)-sum(c1(m:i)):d20998:S1p0998+d20998*(j(i+1)-n/2-1)
                          -sum(c1(m:i)),smax(j(i)+1:j(i+1)));
    end
    Sp002(j(lk)+1:n+1)=max(S1p002+d2002*(j(lk)-n/2)-sum(c1(m:lk)):d2002:S2p002,smin(j(lk)+1:n+1));
    Sp01(j(lk)+1:n+1)=max(S1p01+d201*(j(lk)-n/2)-sum(c1(m:lk)):d201:S2p01,smin(j(lk)+1:n+1));
    Sp025(j(lk)+1:n+1)=max(S1p025+d2025*(j(lk)-n/2)-sum(c1(m:lk)):d2025:S2p025,smin(j(lk)+1:n+1));
    Sp04(j(lk)+1:n+1)=max(S1p04+d204*(j(lk)-n/2)-sum(c1(m:lk)):d204:S2p04,smin(j(lk)+1:n+1));
    Sp06(j(lk)+1:n+1)=min(S1p06+d206*(j(lk)-n/2)-sum(c1(m:lk)):d206:S2p06,smax(j(lk)+1:n+1));
    Sp075(j(lk)+1:n+1)=min(S1p075+d2075*(j(lk)-n/2)-sum(c1(m:lk)):d2075:S2p075,smax(j(lk)+1:n+1));
    Sp09(j(lk)+1:n+1)=min(S1p09+d209*(j(lk)-n/2)-sum(c1(m:lk)):d209:S2p09,smax(j(lk)+1:n+1));
    Sp098(j(lk)+1:n+1)=min(S1p098+d2098*(j(lk)-n/2)-sum(c1(m:lk)):d2098:S2p098,smax(j(lk)+1:n+1));
    Sp099(j(lk)+1:n+1)=min(S1p099+d2099*(j(lk)-n/2)-sum(c1(m:lk)):d2099:S2p099,smax(j(lk)+1:n+1));
    Sp0998(j(lk)+1:n+1)=min(S1p0998+d20998*(j(lk)-n/2)-sum(c1(m:lk)):d20998:S2p0998,smax(j(lk)+1:n+1));
end
ds7=max((smax-Sp0998)./10,e);
ds6=max((Sp0998-Sp099)./10,e);
ds5=max((Sp099-Sp098+Sp002-smin)./10,e);
ds4=max((Sp098-Sp09+Sp01-Sp002)./10,e);
ds3=max((Sp09-Sp075+Sp025-Sp01)./10,e);
ds2=max((Sp075-Sp06+Sp04-Sp025)./15,e);
ds1=(Sp06-Sp04)./20;
a1p002=logninv(0.02,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p01=logninv(0.1,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p025=logninv(0.25,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p04=logninv(0.4,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p06=logninv(0.6,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p075=logninv(0.75,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p09=logninv(0.9,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a1p098=logninv(0.98,log(a)-1/2*nu^2*T/2,nu*sqrt(T/2));
a2p002=logninv(0.02,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p01=logninv(0.1,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p025=logninv(0.25,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p04=logninv(0.4,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p06=logninv(0.6,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p075=logninv(0.75,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p09=logninv(0.9,log(a)-1/2*nu^2*T,nu*sqrt(T));
a2p098=logninv(0.98,log(a)-1/2*nu^2*T,nu*sqrt(T));
```

```
ap002=max([a:(a1p002-a)/(n/2):a1p002 a1p002+(a2p002-a1p002)/(n/2):(a2p002-a1p002)/(n/2):a2p002],amin);
ap01=max([a:(a1p01-a)/(n/2):a1p01 a1p01+(a2p01-a1p01)/(n/2):(a2p01-a1p01)/(n/2):a2p01],amin);
ap025=max([a:(a1p025-a)/(n/2):a1p025 a1p025+(a2p025-a1p025)/(n/2):(a2p025-a1p025)/(n/2):a2p025],amin);
ap04=max([a:(a1p04-a)/(n/2):a1p04 a1p04+(a2p04-a1p04)/(n/2):(a2p04-a1p04)/(n/2):a2p04],amin);
ap06=min([a:(a1p06-a)/(n/2):a1p06 a1p06+(a2p06-a1p06)/(n/2):(a2p06-a1p06)/(n/2):a2p06],amax);
ap075=min([a:(a1p075-a)/(n/2):a1p075 a1p075+(a2p075-a1p075)/(n/2):(a2p075-a1p075)/(n/2):a2p075],amax);
ap09=min([a:(a1p09-a)/(n/2):a1p09 a1p09+(a2p09-a1p09)/(n/2):(a2p09-a1p09)/(n/2):a2p09],amax);
ap098=min([a:(a1p098-a)/(n/2):a1p098 a1p098+(a2p098-a1p098)/(n/2):(a2p098-a1p098)/(n/2):a2p098],amax);
da5=max((amax-ap098+ap002-amin)./4,e);
da4=max((ap098-ap09+ap01-ap002)./4,e);
da3=max((ap09-ap075+ap025-ap01)./4,e);
da2=max((ap075-ap06+ap04-ap025)./6,e);
da1=(ap06-ap04)./8;

[X Y]=meshgrid([amin(n+1):da5(n+1):(ap002(n+1)-e) ap002(n+1):da4(n+1):(ap01(n+1)-e) ap01(n+1):da3(n+1)
      :(ap025(n+1)-e)ap025(n+1):da2(n+1):(ap04(n+1)-e) ap04(n+1):da1(n+1):(ap06(n+1)-e) ap06(n+1):
      da2(n+1):(ap075(n+1)-e) ap075(n+1):da3(n+1):(ap09(n+1)-e) ap09(n+1):da4(n+1):(ap098(n+1)-e)
      ap098(n+1):da5(n+1):(amax(n+1)-e) amax(n+1)],[smin(n+1):ds5(n+1):(Sp002(n+1)-e) Sp002(n+1):
      ds4(n+1):(Sp01(n+1)-e) Sp01(n+1):ds3(n+1):(Sp025(n+1)-e) Sp025(n+1):ds2(n+1):(Sp04(n+1)-e)
      Sp04(n+1):ds1(n+1):(Sp06(n+1)-e) Sp06(n+1):ds2(n+1):(Sp075(n+1)-e) Sp075(n+1):ds3(n+1):(Sp09(n+1)-e)
      Sp09(n+1):ds4(n+1):(Sp098(n+1)-e) Sp098(n+1):ds5(n+1):(Sp099(n+1)-e) Sp099(n+1):ds6(n+1):
      (Sp0998(n+1)-e) Sp0998(n+1):ds7(n+1):smax(n+1)-e smax(n+1)]);
f=max(c.*(Y-K),0);
p1=(1+rho)/4;
p2=(1-rho)/4;
for l=n:-1:j(lk)+1
    disc=exp(-intergratingr((l-1)*dt,l*dt));
    X1=X;
    Y1=Y;
    [X Y]=meshgrid([amin(l):da5(l):(ap002(l)-e) ap002(l):da4(l):(ap01(l)-e) ap01(l):da3(l):(ap025(l)-e)
          ap025(l):da2(l):(ap04(l)-e) ap04(l):da1(l):(ap06(l)-e) ap06(l):da2(l):(ap075(l)-e)
          ap075(l):da3(l):(ap09(l)-e) ap09(l):da4(l):(ap098(l)-e) ap098(l):da5(l):(amax(l)-e) amax(l)],
          [smin(l):ds5(l):(Sp002(l)-e) Sp002(l):ds4(l):(Sp01(l)-e) Sp01(l):ds3(l):(Sp025(l)-e)
          Sp025(l):ds2(l):(Sp04(l)-e) Sp04(l):ds1(l):(Sp06(l)-e) Sp06(l):ds2(l):(Sp075(l)-e)
          Sp075(l):ds3(l):(Sp09(l)-e) Sp09(l):ds4(l):(Sp098(l)-e) Sp098(l):ds5(l):(Sp099(l)-e)
          Sp099(l):ds6(l):(Sp0998(l)-e) Sp0998(l):ds7(l):(smax(l)-e) smax(l)]);
    Z11=min(X+nu.*X.*sqrtdt,bounda(l+1));
    Z12=max(X-nu.*X.*sqrtdt,0);
    Z21=min(Y+rdt(l).*Y+X.*exp(-(intergratingr((l-1)*dt,T)))^(1-b).*Y.^b.*sqrtdt,bounds(l+1));
    Z22=max(min(Y+rdt(l).*Y-X.*exp(-(intergratingr((l-1)*dt,T)))^(1-b).*Y.^b.*sqrtdt,bounds(l+1)),0);
    f=max(disc.*(p1.*interp2(X1,Y1,f,Z11,Z21,'cubic')+p2.*interp2(X1,Y1,f,Z11,Z22,'cubic')+
      p2.*interp2(X1,Y1,f,Z12,Z21,'cubic')+p1.*interp2(X1,Y1,f,Z12,Z22,'cubic')),c.*(Y-K));
end
l=j(lk);
    disc=exp(-intergratingr((l-1)*dt,l*dt));
    X1=X;
    Y1=Y;
    [X Y]=meshgrid([amin(l):da5(l):(ap002(l)-e) ap002(l):da4(l):(ap01(l)-e) ap01(l):da3(l):(ap025(l)-e)
          ap025(l):da2(l):(ap04(l)-e) ap04(l):da1(l):(ap06(l)-e) ap06(l):da2(l):(ap075(l)-e)
          ap075(l):da3(l):(ap09(l)-e) ap09(l):da4(l):(ap098(l)-e) ap098(l):da5(l):(amax(l)-e) amax(l)],
          [smin(l):ds5(l):(Sp002(l)-e) Sp002(l):ds4(l):(Sp01(l)-e) Sp01(l):ds3(l):(Sp025(l)-e)
          Sp025(l):ds2(l):(Sp04(l)-e) Sp04(l):ds1(l):Sp06(l)-e) Sp06(l):ds2(l):(Sp075(l)-e)
          Sp075(l):ds3(l):(Sp09(l)-e) Sp09(l):ds4(l):(Sp098(l)-e) Sp098(l):ds5(l):(Sp099(l)-e)
          Sp099(l):ds6(l):(Sp0998(l)-e) Sp0998(l):ds7(l):(smax(l)-e) smax(l)]);
    Z11=min(X+nu.*X.*sqrtdt,bounda(l+1));
    Z12=max(X-nu.*X.*sqrtdt,0);
    Z21=max(min(Y+rdt(l).*Y+X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-q1(lk),0).^b.*sqrtdt-c1(lk),bounds(l+1)),0);
    Z22=max(min(Y+rdt(l).*Y-X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-q1(lk),0).^b.*sqrtdt-c1(lk),bounds(l+1)),0);
    f=max(disc.*(p1.*interp2(X1,Y1,f,Z11,Z21,'cubic')+p2.*interp2(X1,Y1,f,Z11,Z22,'cubic')+
      p2.*interp2(X1,Y1,f,Z12,Z21,'cubic')+p1.*interp2(X1,Y1,f,Z12,Z22,'cubic')),c.*(Y-K));
```

```
for i=lk-1:-1:1
    for l=j(i+1)-1:-1:j(i)+1
    disc=exp(-intergratingr((l-1)*dt,l*dt));
    X1=X;
    Y1=Y;
    [X Y]=meshgrid([amin(l):da5(l):(ap002(l)-e) ap002(l):da4(l):(ap01(l)-e) ap01(l):da3(l):(ap025(l)-e)
    ap025(l):da2(l):(ap04(l)-e) ap04(l):da1(l):(ap06(l)-e) ap06(l):da2(l):(ap075(l)-e)
    ap075(l):da3(l):(ap09(l)-e) ap09(l):da4(l):(ap098(l)-e) ap098(l):da5(l):(amax(l)-e) amax(l)],
    [smin(l):ds5(l):(Sp002(l)-e) Sp002(l):ds4(l):(Sp01(l)-e) Sp01(l):ds3(l):(Sp025(l)-e)
    Sp025(l):ds2(l):(Sp04(l)-e) Sp04(l):ds1(l):(Sp06(l)-e) Sp06(l):ds2(l):(Sp075(l)-e)
    Sp075(l):ds3(l):(Sp09(l)-e) Sp09(l):ds4(l):(Sp098(l)-e) Sp098(l):ds5(l):(Sp099(l)-e)
    Sp099(l):ds6(l):(Sp0998(l)-e) Sp0998(l):ds7(l):(smax(l)-e) smax(l)]);
    Z11=min(X+nu.*X.*sqrtdt,bounda(l+1));
    Z12=max(X-nu.*X.*sqrtdt,0);
    Z21=min(Y+rdt(l).*Y+X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(i+1:lk)),0).^b.*sqrtdt,bounds(l+1));
    Z22=max(min(Y+rdt(l).*Y-X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(i+1:lk)),0).^b.*sqrtdt,bounds(l+1)),0);
    f=max(disc.*(p1.*interp2(X1,Y1,f,Z11,Z21,'cubic')+p2.*interp2(X1,Y1,f,Z11,Z22,'cubic')+
      p2.*interp2(X1,Y1,f,Z12,Z21,'cubic')+p1.*interp2(X1,Y1,f,Z12,Z22,'cubic')),c.*(Y-K));
    end
    l=j(i);
    disc=exp(-intergratingr((l-1)*dt,l*dt));
    X1=X;
    Y1=Y;
    [X Y]=meshgrid([amin(l):da5(l):(ap002(l)-e) ap002(l):da4(l):(ap01(l)-e) ap01(l):da3(l):(ap025(l)-e)
    ap025(l):da2(l):(ap04(l)-e) ap04(l):da1(l):(ap06(l)-e) ap06(l):da2(l):(ap075(l)-e)
    ap075(l):da3(l):(ap09(l)-e) ap09(l):da4(l):(ap098(l)-e) ap098(l):da5(l):(amax(l)-e) amax(l)],
    [smin(l):ds5(l):(Sp002(l)-e) Sp002(l):ds4(l):(Sp01(l)-e) Sp01(l):ds3(l):(Sp025(l)-e)
    Sp025(l):ds2(l):(Sp04(l)-e) Sp04(l):ds1(l):(Sp06(l)-e) Sp06(l):ds2(l):(Sp075(l)-e)
    Sp075(l):ds3(l):(Sp09(l)-e) Sp09(l):ds4(l):(Sp098(l)-e) Sp098(l):ds5(l):(Sp099(l)-e)
    Sp099(l):ds6(l):(Sp0998(l)-e) Sp0998(l):ds7(l):(smax(l)-e) smax(l)]);
    Z11=min(X+nu.*X.*sqrtdt,bounda(l+1));
    Z12=max(X-nu.*X.*sqrtdt,0);
    Z21=max(min(Y+rdt(l).*Y+X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(i:lk)),0).^b.*sqrtdt-c1(i),bounds(l+1)),0);
    Z22=max(min(Y+rdt(l).*Y-X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(i:lk)),0).^b.*sqrtdt-c1(i),bounds(l+1)),0);
    f=max(disc.*(p1.*interp2(X1,Y1,f,Z11,Z21,'cubic')+p2.*interp2(X1,Y1,f,Z11,Z22,'cubic')+
      p2.*interp2(X1,Y1,f,Z12,Z21,'cubic')+p1.*interp2(X1,Y1,f,Z12,Z22,'cubic')),c.*(Y-K));
end
for l=j(1)-1:-1:2
    disc=exp(-intergratingr((l-1)*dt,l*dt));
    X1=X;
    Y1=Y;
    [X Y]=meshgrid([amin(l):da5(l):(ap002(l)-e) ap002(l):da4(l):(ap01(l)-e) ap01(l):da3(l):(ap025(l)-e)
    ap025(l):da2(l):(ap04(l)-e) ap04(l):da1(l):(ap06(l)-e) ap06(l):da2(l):(ap075(l)-e)
    ap075(l):da3(l):(ap09(l)-e) ap09(l):da4(l):(ap098(l)-e) ap098(l):da5(l):(amax(l)-e) amax(l)],
    [smin(l):ds5(l):(Sp002(l)-e) Sp002(l):ds4(l):(Sp01(l)-e) Sp01(l):ds3(l):(Sp025(l)-e)
    Sp025(l):ds2(l):(Sp04(l)-e) Sp04(l):ds1(l):(Sp06(l)-e) Sp06(l):ds2(l):(Sp075(l)-e)
    Sp075(l):ds3(l):(Sp09(l)-e) Sp09(l):ds4(l):(Sp098(l)-e) Sp098(l):ds5(l):(Sp099(l)-e)
    Sp099(l):ds6(l):(Sp0998(l)-e) Sp0998(l):ds7(l):(smax(l)-e) smax(l)]);
    Z11=min(X+nu.*X.*sqrtdt,bounda(l+1));
    Z12=max(X-nu.*X.*sqrtdt,0);
    Z21=min(Y+rdt(l).*Y+X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(1:lk)),0).^b.*sqrtdt,bounds(l+1));
    Z22=max(min(Y+rdt(l).*Y-X.*exp(-(intergratingr((l-1)*dt,T))).*
        max(Y/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(1:lk)),0).^b.*sqrtdt,bounds(l+1)),0);
    f=max(disc.*(p1.*interp2(X1,Y1,f,Z11,Z21,'cubic')+p2.*interp2(X1,Y1,f,Z11,Z22,'cubic')+
      p2.*interp2(X1,Y1,f,Z12,Z21,'cubic')+p1.*interp2(X1,Y1,f,Z12,Z22,'cubic')),c.*(Y-K));
end
```

```
l=1;
disc=exp(-intergratingr((l-1)*dt,l*dt));
Z1a=a+nu*sqrtdt*a;
Z2a=a-nu*sqrtdt*a;
Z1s=S+rdt(l)*S+a*(S/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(1:lk)))^b*
    exp(-(intergratingr((l-1)*dt,T)))*sqrtdt;
Z2s=S+rdt(l)*S-a*(S/exp(-(intergratingr((l-1)*dt,T)))-sum(q1(1:lk)))^b*
    exp(-(intergratingr((l-1)*dt,T)))*sqrtdt;
y=max(disc*(p1*interp2(X,Y,f,Z1a,Z1s,'cubic')+p2*interp2(X,Y,f,Z1a,Z2s,'cubic')+
  p2*interp2(X,Y,f,Z2a,Z1s,'cubic')+p1*interp2(X,Y,f,Z2a,Z2s,'cubic')),c*(S-K));

function y=interest(s)
x=[0 14/365 105/365 196/365 287/365 378/365 469/365 560/365 651/365 742/365 833/365 924/365
   1015/365 1106/365 1197/365 1288/365 1379/365 1470/365 1561/365 1659/365 1750/365];
z=[0.038742466 0.038742466 0.03968778 0.04019537 0.04049118 0.04061941 0.04060938 0.04055303
   0.0404908 0.04045563 0.04041543 0.04038429 0.0435874 0.04033841 0.04032976 0.04032233 0.04031588
   0.04031232 0.04032951 0.04034591 0.0403595];
y=interp1(x,z,s);

function y=intergratingr(t1,t2)
y=quad(@(s) interest(s),t1,t2);
```

The codes of the programs diff2vput and diff2v are already given in the code for the European option with no dividends and constant interest rate.