Numerical Models for Pricing Barrier Options

Oskar Milton

5th March 1999

Contents

1 Abstract

A Barrier options is a typ of option that is path dependent. That is, the payoff is dependent of the motion of the underlying asset from the day the contract is written until it expires. In this working paper I am evaluating the best suited numerical model for pricing this type of contract. Of the four different types of models, I have found one that is very fast and robust. There are some earlier works in this field but non of them has came up with a model as fast as the one presented here. The model is a Finite Difference Model with Cranc-Nicolson, the results is accurate due to earlier works but here is the results presented with a higher precision, this is not scientifically proved and shall just be used as a help to understand the very fast convergence of this model.

2 Variable specification

Symbol	Description	Code notation
S	Current price of the underlying asset.	S
$\tilde{S}(t)$ or $\tilde{S}$	Continuous price of the underlying asset at any time t.	-
$S_{i,j}$	Price of the underlying asset at position (i, j) in a tree or a grid.	St[i][j]
σ	Standard deviation of return, volatility, of asset price (S).	sig_S
r	Continuously compounded interest rate.	r
K	Strike or exercise price of a contingent claim.	K
μ	Drift of asset price (S).	mu
t	Time.	t
T	Maturity date of contingent claim.	T
q	Continuous dividend yield on an asset.	q
c(K,T)	European call price with strike price K and time to maturity T.	-
p(K,T)	European put price with strike price K and time to maturity T.	-
$\tilde{C}(t,s)$ or $\tilde{C}$	The contingent claim price at time t and asset price s for some conract.	-
$C_{i,j}$	The contingent claim price at position (i, j) in a tree or a grid.	C[i][i]
y	The natural logarithm of the asset price S.	y
ν	Risk neutral drift of y.	nu
u	Size of proportional upward move of stochastic variable or subscript indicating upward move of a stochastic variable.	u
d	Size of proportional downward move of stochastic variable or subscript indicating downward move of a stochastic variable.	d

Symbol	Description	Code notation
p_u	Probability of upward move in a tree.	pu
p_m	Probability of straight-forward move in a tree.	pm
p_d	Probability of downward move in a tree.	pd
i	Time step index in a tree or a grid.	i
j	Asset price state index in a tree or a grid.	j
N_i	Number of discretizations, i.e. the number of time steps, between current date and maturity date.	Ni
N_j	Number of discretizations of the asset price between the lowest and the highest node in the tree or the grid.	Nj
M	Number of simulations for the MonteCarlo simulation model.	M
B	Barrier level.	B
X_{rebate}	Cash rebate paied when barrier knocks out.	Xrebate
D_i	Dividend payment at time step i.	Di
D^i_{sum}	Discounted cumulative value of all future dividend payments at time step i, including payment at time step i.	Dsum[i]

Hedge sensitivities:

δ	The rate of change of the value of an option with respect to changes in the stock price.	delta
γ	The rate of change of the delta with respect to changes in the stock price.	gamma
ω	The rate of change of the gamma with respect to changes in the stock price.	omega
θ	The rate of change of the value of an option with respect to time.	theta
ρ	The rate of change of the value of an option with respect to the risk-free rate of interest.	rho
v	The rate of change of the value of an option with respect to volatility.	vega

For futher reading about the hedge sensitivities, go to section 10.

3 Introduction

I've been commissioned by ABN-Amro Software AB to evaluate different numerical models for pricing barrier options.

This project is my thesis for MSc at the Royal Institute of Technology, KTH, in Stockholm.

The models that are handled are the following:

- The Binomial Tree Model
- The Trinomial Tree Model
- The Finite Difference Model
- Monte Carlo Simulation

When comparing different models for pricing derivatives it is normally the tree or grid size, i.e. the number of discretizations, that is considered. When a model is used in a transaction system where derivative traders need to recalculate their portfolio several times per hour it is not the number of discretisations that is important but the calculation time. These big systems are time-critical because of the heavy load and must therefore use models that give the best value at the lowest amount of time. In the comparison later on in this paper I will show both the number of discretizations and the calculation-time for the different models. When I say that one model is faster than an-other it is normally with respect to the calculation-time that I have made the comparison.

To get a clear idea of the behaviour of the different models, the diagrams are displayed with option value on the vertical axis and computer-power or real calculation time on the horizontal axis. The computer-power is a relative factor that indicates how much time a calculation of an option's price takes, the real computer time on the other hand is the true calculation time measured by the computer and is displayed in milliseconds. If the computer-power or calculation time needed for one model is twice as high as for an other model for the same accuracy in the option's price, we come to the conclusion that the second model is twice as fast and twice as good as the first one.

A computer program has been developed to simplify the evaluation process and the comparison. This tool makes it possible to draw graphs from several different models in the same diagram and makes it easy to compare the convergence of the different models. All the diagrams shown in the paper comes from this tool. It also allows making diagrams for hedge parameters such as delta, gamma, omega etc. The hedge parameters tell us much about the model and it's accuracy near the barrier.

The tool is constructed so that implementation and testing of different derivatives are made easy and fast. This was already from the start a pronounced goal because ABN-Amro Software AB wants to implement several new derivatives in their system. All these derivatives need to be evaluated and tested. A great thing about the tool is that anyone, including those with no programming experience, can use the tool and get the maximum of help in their evaluation process from it. In the end of this paper I will show you how the evaluation process is to be done and how to use the tool to make it fast. The tool belongs to ABN-Amro Software AB and can not be distributed with this paper.

3.1 Briefly about options

Options are traded all around the world in bigger and bigger quantities, but what is an option and why trade with them? All options have an underlying asset. As an example we have the stock option that has a stock as its underlying asset. Commodities as metals or corn are other underlying assets that an option can be based on. Often when dealing with options we hear words like put and call or European and American but what do they really mean?

If we start with the call, when you by a call option contract you have bought the right to buy a certain amount of the underlying asset to a fixed price, also called the strike price with the notation K, specified in the contract. The put on the other hand gives you the right to sell a certain amount of the underlying asset for the strike price, K, specified in the option contract. In both cases the one that must sell or by the asset when the contract is exercised is the one who originally wrote the option contract.

If the option is a European style option it can only be exercised at the maturity day. For the American style option contract it can be exercised at any time during the options life, i.e. from the day it was written until maturity.

These types are the most common but there are plenty of different types on the market, more or less traded. Next we are going to look at a type called barrier option, it is yet not traded in very big quantities..

3.2 What is a barrier option?

A barrier option is a path dependent option with some kind of restriction of its validity. That is, if the barrier is crossed the option will either be valid and active or invalid and void. The barrier is a fixed value due to the stock price, e.g. if the current stock price is SEK 100 the barrier can be placed at SEK 120, the barrier will be crossed if the stock price rises over SEK 120.

The option can start it's life inactive and become active when/if the asset price crosses the barrier, i.e. knock in, or start active and become void when/if it crosses the barrier, i.e. knock out. The barrier can be placed above or below the current stock price. The option can be either a call or a put.

If an option starts active as for knock out barrier options and when the option goes void a rebate is often paid to the owner of the option contract.

When combining these different types we will get eight different single barrier options described below.

3.2.1 Up-and-out call

The up-and-out call starts active and stays so until the barrier is crossed or the expiry date is reached. If the barrier is crossed the rebate will be paid. If the expiry date is reached and the stock price is above the strike price a sum equal $S_T - K$ will be paid, else the value of the option is 0. If it's an American option the option can be exercised at any time, t, as long as it is active. The sum paid will then be $S_t - K$. Figure 1 shows two different paths, 1 and 2, for an "Up and Out" call option. Path 1 crosses the barrier marked B, as soon as it does it gets void and the option's price equals zero even if it gets back below the barrier before maturity day. Path 2 that never crosses the barrier will get a payoff at maturity day that is greater than zero because the asset price is between the strike price K, and the barrier, B.

3.2.2 Up-and-in call

The up-and-in call starts inactive and stays so until the barrier is crossed. If the expiry date is reached and the barrier hasn't been crossed, the option will be void and no money will be paid. If the barrier is crossed the option becomes active but will not become inactive if it

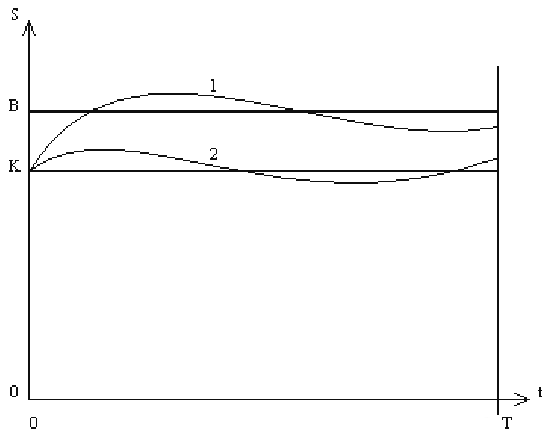

Figure 1: Up and Out call example. The asset price following path 1 will knock out when it crosses the barrier, the option becomes void and gets a value equal to zero. If the asset price is following path 2 the option gets a value equal to $S_T - K$.

crosses the barrier again. Therefore, the value of the option is equal $S_T - K$ for a European option and $S_t - K$ for an American option if it is exercised before maturity date. If the option becomes void no rebate will be paid.

3.2.3 Down-and-out call

The down-and-out call has the same behaviour as the up-and-out call with the difference that this instrument's barrier is placed below the stock price at time 0. In this case the stock price has to cross the barrier to become void, that is, the stock price has to go below the barrier to inactivate the instrument. The sum paid is the same as for a down-and-out call; if the barrier is crossed a rebate will be paid. If the expiry date is reached and the stock price is above the strike price a sum equal $S_T - K$ will be paid, else the value of the option is 0. If it's an American option the option can be executed at any time, t, as long as it is active. Then, the sum paid will be $S_t - K$.

Note, this is the option that is going to be the standard example option, all tests and evaluations are done due to this barrier type. See section 3.5

3.2.4 Down-and-in call

The down-and-in call has the same behaviour as the up-and-in-call with the difference that this instrument's barrier is placed below the stock price at time 0. In this case the stock price has to cross the barrier to become active, that is, the stock price has to go below the barrier to activate the instrument. When the barrier once is crossed it can never be void. The sum paid is the same as for the up-and-in call; the value of the option is equal $S_T - K$ for a European option and $S_t - K$ for an American option if it is exercised at time t. If the option becomes void no rebate will be paid.

3.2.5 Up-and-out put

The up-and-out put starts active just like the up and out call option, if the option's payoff shall be greater than zero the asset price has to lay under the strike price and it shall never during its life have raised above the barrier. Here, if the barrier never has been crossed, the payoff is $K - S_T$ for a European option and $K - S_t$ for a American option if it is exercised in advance.

3.2.6 Up-and-in put

The Up-and-in put starts inactive and gets valid when crossing the barrier, just like the Up-and-in call option. Here, the option pays off if the asset price lay under the strike price and if it at least one time during its life has been above the barrier.As for the Up-and-out put, if the barier never has been crossed the payoff is $K - S_T$ for a European option and $K - S_t$ for a American option if it is exercised in advance.

3.2.7 Down-and-out put

The Down-and-out put knocks out and gets void like the Down-and-out call if the barrier is crossed. Therefore, the payoff will be greater than zero only if the asset price lay below the strike price and if it never crosses the barrier during its life. As for the Up-and-out put, if the barier never has been crossed the payoff is $K - S_T$ for a European option and $K - S_t$ for a American option if it is exercised in advance.

3.2.8 Down-and-in put

The Down-and-in put knocks in and gets valid and active when it crosses the barrier from above. It pays off if the asset prive is lower than the strike price and if the barrier has been crossed at least once. As for the Up-and-out put, if the barrier never has been crossed the payoff is $K - S_T$ for a European option and $K - S_t$ for a American option if it is exercised in advance.

3.3 Double barrier option

The two-barrier option, also referred to as corridor option, can be of two types, double knock in or double knock out. The first starts inactive and becomes active when a barrier is crossed from the outside, the other starts active with the current asset price somewhere between the barriers, it gets void if any of the barriers are crossed. This type of option is not going to be evaluated in this paper, this is a limitation and is discussed next.

3.4 Delimitations

A barrier option can have one or more barriers as discussed above. In addition, the barrier can be non-constant. In this paper, only single constant barrier options will be discussed and evaluated. In section 13, "Further Development", I briefly discuss how other contracts can be treated.

3.5 Standard option contract

To make the comparison between different models easy, we will use one standard option contract in the examples throughout this paper. It will be a single barrier option of the type "Down-and-out" call with the following data:

$$S = 95$$

$$K = 100$$

$$B = 90$$

$$r = 10\%$$

$$T = 1$$

No dividend is payed during the options life. The definition of all variables can be found in section 2.

4 Black-Scholes analytic formula and theory

In 1973, Fischer Black and Myron Scholes received the Nobel price for their option pricing formula. They have revolutionized the world with their formula that makes it possible to calculate the option value for some options in an analytic way. Before presenting the simple analytic formulas, I will show you the Black-Scholes differential equation from which the formulas derive,

$$\frac{\partial \tilde{C}}{\partial t} + r\tilde{S}\frac{\partial \tilde{C}}{\partial \tilde{S}} + \frac{1}{2}\sigma^2\tilde{S}^2\frac{\partial^2 \tilde{C}}{\partial \tilde{S}^2} = r\tilde{C}. \tag{1}$$

It is outside the framework of this paper to derive this formula but more details can be found in Hull [16] section 11. This differential equation is also the basis for the finite difference model, see section 7.

For the Black-Scholes pricing formulas, that will be presented below, there are some assumptions that restrict the use. They are as follows:

- The stock pays no dividends during the option's life.
- European exercise terms are used.
- An arbitrage-free world is assumed.
- Interest rates and volatility remain constant and known during the option's life.
- The underlying asset price follows a geometric Brownian motion, i.e. returns are log-normally distributed.

The formula and all its variables is described below. Where

$$c(K,T) = \tilde{S}N(d_1) - Ke^{-rT}N(d_2) \tag{2}$$

and

$$p(K,T) = -\tilde{S}N(-d_1) + Ke^{-rT}N(-d_2). \tag{3}$$

Here,

$$d_1 = \frac{log(\frac{\tilde{S}}{K}) + (r + \frac{\sigma}{2})T}{\sigma\sqrt{T}} \tag{4}$$

and

$$d_2 = d_1 - \sigma\sqrt{T}. \tag{5}$$

The variables used above:

c() =	Theoretical Call Premium.
p() =	Theoretical Put Premium.
$\tilde{S}$ =	Current stock price.
T =	Time until expiration.
K =	Option striking price.
R =	Risk-free interest rate.
N() =	Cumulative standard normal distribution.
σ =	Standard deviation of stock returns (Volatility).

In order to understand the formula itself, equation 2, we divide it into two parts. The first part, $\tilde{S}N(d_1)$, derives the expected benefit from acquiring a stock outright. This is found by multiplying stock price, $\tilde{S}$, by the change in the call premium with respect to a change in the underlying stock price, $N(d_1)$. The second part of the model, $Ke^{-rt}N(d_2)$, gives the present value of paying the exercise price on the expiration day. The fair market value of the call option is then calculated by taking the difference between these two parts.

The assumptions mentioned above make formula 2 and 3 applicable on a limited number of derivatives. The assumption that says that no dividends can be paid during the option's life restrict the use dramatically, most companies pay dividend to their share holders and when a dividend is paid the call option value, for example, gets lower. Continuous dividend payment is an approximate way to deal with discrete dividend payment; Robert Merton came up with a solution for this in 1973. The assumption that European exercise terms are used means that the option can only be exercised on the maturity day in contrary to American exercise terms that means that the option can be exercised at any time during the option's life.

In addition to the assumptions above the Black-Scholes formula is not applicable on path dependent options such as barrier options. We then have to find different models that relax the assumptions of no dividends, only European exercise terms and not path dependent options. For numerical models in discrete time and asset price are all these assumptions relaxed, which gets us to the following chapters where different numerical models will be examined and evaluated.

5 Binomial Tree Model

The binomial tree model is a simple and powerful model for pricing standard options such as European and American style options. For these, the model is widely used. The results retrieved with around 100 time steps is often a good approximation to the accurate theoretically value and the calculation time required is very short. Two different binomial models will be presented here, the first one proposed by Cox, Ross and Rubinstein, see [16], is the most intuitive one and is easy to understand. The second model was proposed by Trigeorgis 1992, see [15] page 18-19, and is an even better approximation to the theoretically correct option price. I will present this model as an educational example to make it easier to understand the log-transformed calculation method also used for the trinomial and the finite difference models presented later in this paper.
These models are also applicable on options with discrete and continuous dividends. Furthermore, it is very simple to implement early exercise conditions as needed for American option pricing. Options with path dependency as barrier options can also be treated.

When using these models with the barrier condition attached to the model for pricing standard options, we will get a very unstable behaviour, especially near the barrier. I will show a way to decrease this instability as suggested by Boyle and Lau [4] which is applicable on the model by Cox-Ross-Rubinstein. Moreover, I will show how to implement early exercise condition and discrete dividends within the binomial tree framework. Finally I discuss hedge sensitivities such as delta, gamma and omega.

5.1 Cox-Ross-Rubinstein Binomial Model

The Cox-Ross-Rubinstein binomial model was presented 1979 in a working paper called; "Option Pricing: A Simplified Approach". Their theory has laid ground to the binomial model that I will present below.

To begin with, the binomial tree has equal time steps, Δt, througout the whole tree, so are also the up ratio, u, the down ratios, d, and the transition probability, p. For simplifying the notation we decide upon a numbering system (i, j) as in figure 2. Each variable has it's index derived from this numbering system.

We will only look at recombining trees for this model, that is, one up move followed by a down move will result in the same node as one down move followed by one up move, $ud = du$.

Figure 3 shows a one step binomial model of the asset price, also called the jump process.

Cox, Ross and Rubinstein have chosen the multiplicative jumpsizes u and d to have the relationship

$$u = \frac{1}{d}, \tag{6}$$

which satisfies the jump condition mentioned above. They also show that u must satisfy the condition

$$u = e^{\sigma\sqrt{\Delta t}}. \tag{7}$$

Since we are working in a risk-neutral and arbitrage-free world, see [16] page 198 and 12, the expected return from a stock is equal the risk-free interest rate r. Then, for one time step Δt, the expected value for the option given the values one time step ahead is:

$$S_{i,j}e^{r\Delta t} = pS_{i+1,j+1} + (1-p)S_{i+1,j} \tag{8}$$

or

$$e^{r\Delta t} = pu + (1-p)d, \tag{9}$$

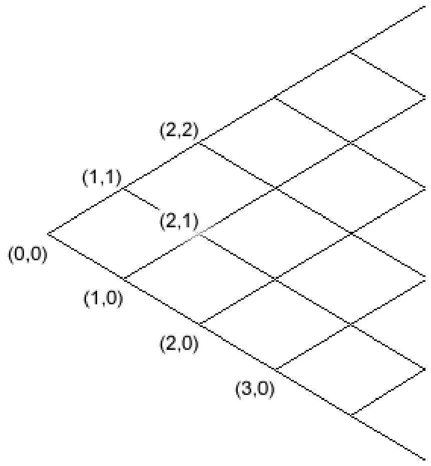

Figure 2: Number system for the Binomial Tree.

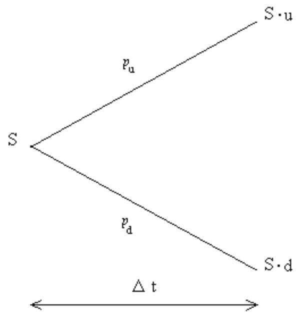

Figure 3: Jump process for the asset price for the binomial tree.

which gives the explicit expression of the transition probability

$$p = \frac{e^{r\Delta t} - d}{u - d}. \tag{10}$$

5.2 Trigeorgis Model

Trigeorgis has developed a model where the asset price is log-transformed. This means that he assumes that it is the logarithm of the asset price that evolves as a brownian motion with constant drift. The logarithm of the asset price is normally distributed with constant mean and variance. The numbering system will be the same in this model but we'll declare the up and down transition probabilities as p_u and p_d where $p_d = 1 - p_u$. An up move takes you to $y + \Delta y_u$ and a down move takes you to $y + \Delta y_d$, here we will choose equal jump size and therefore $\Delta y_d = -\Delta y_u$. Figure 4 shows a one step binomial model of the logarithm of the asset.

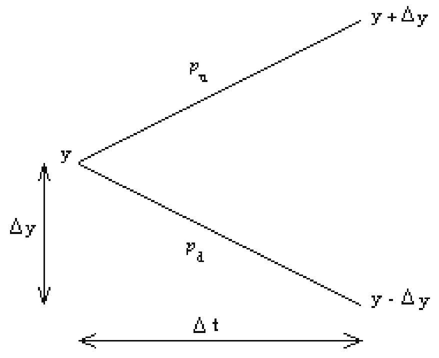

Figure 4: Jump process with equal jump sizes for the logarithm of the asset price.

Applying Itô's formula, the continuous time risk-neutral process for $y = ln(\tilde{S}(t))$ can be shown to be, [15],

$$dy = \nu dt + \sigma dz \tag{11}$$

where

$$\nu = r - \frac{1}{2}\sigma^2. \tag{12}$$

The mean and variance of the continuous time process can be approximated with the binomial process for y, this leads us to equation

$$E[\Delta y] = p_u \Delta y_u + p_d \Delta y_d = \nu \Delta t, \tag{13}$$

and

$$E[\Delta y^2] = p_u \Delta y_u^2 + p_d \Delta y_d^2 = \sigma^2 \Delta t + \nu^2 \Delta t^2. \tag{14}$$

The assumption that we have equal jump size gives us $\Delta y = \Delta y_u = -\Delta y_u$. This applied to equation 13 and 14 gives

$$p_u \Delta y - p_d \Delta y = \nu \Delta t \tag{15}$$

and

$$p_u \Delta y^2 + p_d \Delta y^2 = \sigma^2 \Delta t + \nu^2 \Delta t^2 \tag{16}$$

which takes us to the goal:

$$\Delta y_u = \sqrt{\sigma^2 \Delta t + \nu^2 \Delta t^2}, \tag{17}$$

$$\Delta y_d = -\Delta y_u, \tag{18}$$

$$p_u = \frac{1}{2} + \frac{1}{2}\frac{\nu \Delta t}{\Delta y}, \tag{19}$$

$$p_d = 1 - p_u. \tag{20}$$

Next I'm going to show how to use this model for pricing barrier options.

5.3 Calculating a barrier option's value using Trigeorgis Model

The following procedure to calculate an option's price is allmost the same as for the Cox-Ross-Rubinstein Model, to get a more exact description for this model, I direct you to the litterature in the field. Note, the calculations are done for a Down-and-out call option as for the rest of the examples in this paper, see section 3.5.
We have all the information we need to build the tree, at maturity date we start with computing the stock price at the lowest position,

$$S_{N,0} = Se^{-N_i \Delta y}. \tag{21}$$

We can then compute all other values from $S_{N_i,0}$ to S_{N_i,N_i} by adding $e^{2\Delta y}$ to the asset price at the node above as;

$$S_{N,j+1} = S_{N,j} e^{2\Delta y}. \tag{22}$$

Knowing all the stock prices at maturity date we can now get the call option's price as the maximum value of $S_{N_i,j} - K$ and 0 for all j as;

$$C_{N,j} = max(0, S_{N_i,j} - K). \tag{23}$$

We now do the calculation of the option's price in an iterative way from maturity day back to current time. We compute the option prices at time t knowing the prices at $t + \Delta t$ as;

$$C_{i,j} = p_u C_{i+1,j+1} + p_d C_{i+1,j}. \tag{24}$$

After iterating N_i times we obtain the option's price $C_{0,0}$ at current time. If we during the iteration want to apply the early exercise condition as for American style options, we have to calculate the stock price spread at time t from the spread at time $t + \Delta t$. This can be done in a very simple way;

$$S_{i,j} = \frac{S_{i+1,j}}{e^{-\Delta y}}, \qquad for \;\; j = 0..i \;\; . \tag{25}$$

The early exercise condition can then be added where the option's price is set to

$$C_{i,j} = max(C_{i,j}, S_{i,j} - K). \tag{26}$$

We can finally add the barrier condition to the model. This condition is also very simple to implement. The only check that has to be done is if the barrier has been crossed at any node, if it has, we set the option's value at that node to zero.

When we use the method above for implementing the barrier condition we will get an error. The calculated value will always be above the theoretically correct value (for a Down- and-out call option). Figure 5 below shows how the barrier is represented in the model. You can see that the node just below the true barrier lies where the model will represent an imaginary barrier. The distance between the true barrier and the imaginary one is the source of error that will result in a error in the option price. The closer the barrier lies the current asset price the lower will the option price be. As we have seen, the imaginary barrier will always lay below the true one and therefore the error will always be positive.

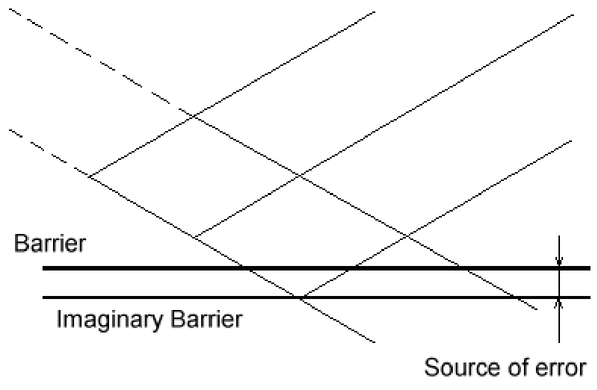

Figure 5: Error generated by the barrier.

While changing the number of time steps the distance between the true barrier and the imaginary barrier will change. It will not always get smaller, in a periodical manner the imaginary barrier will make a jump so that the distance grows to a maximum. This occurs when a node passes the true barrier from below, when passed, the imaginary barrier will lay on a new node below the true barrier. The fluctuation in the option price is presented in Figure 6.

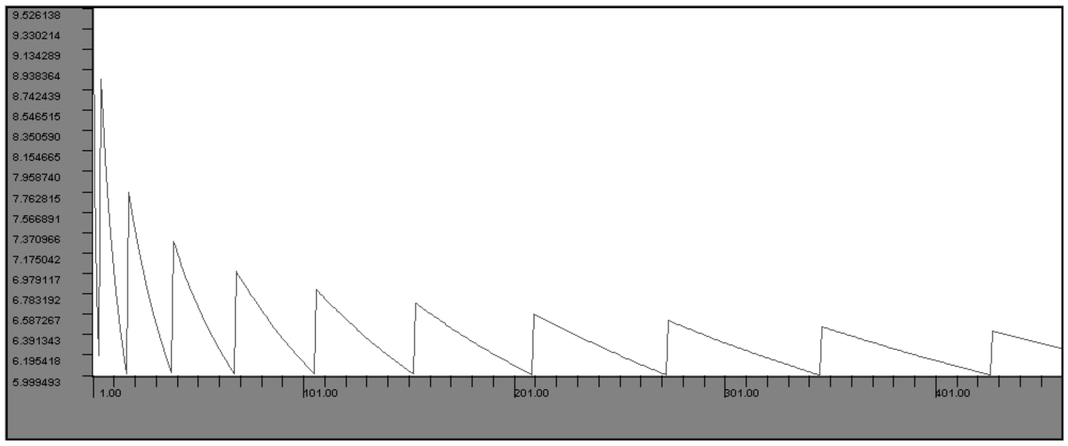

Figure 6: The sawtooth pattern for the binomial tree method. Option value is presented on the vertical axis and the power-rate on the horizontal axis.

How this behaviour can be eliminated by choosing only the best points I will show you next.

5.4 How to choose only the best points

The way the best points is chosen for a binomial tree model was suggested by Phelim P. Boyle and Sok Hoon Lau in 1994 and their working paper was published in the Journal of Derivatives, [4]. The model is based on the binomial model sugested by Cox, Ross and Rubinstein earlier described in this chapter. Their method makes the option price converge much faster than for a standard binomial tree model, I will show why this model gives a better approximation of the option price. Looking att Figure 6, we can se that the option price converge towards a value that is close to the lowest value in the graph. We know that the error, Figure 5, is at a minimum while the barrier lies just above a layer of horizontal nodes. What Boyle and Lau did were that they chose the number of time steps in a way that makes this criteria fulfilled. They came up with a formula that choose the number of time steps for us as

$$F(m) = \frac{m^2\sigma^2 T}{(log(\frac{\tilde{S}}{B}))^2} \qquad m = 1, 2, 3, ... \quad . \tag{27}$$

For some m, the largest integer that is smaller than $F(m)$ represents the number of time steps for the point at the bottom between two sawteeth, see figure 6. For a specific m, knowing the F(m) we can choose the best number of time steps, N, as the largest integer that is smaller than F(m).

Here, the barrier, B, will lie between the asset price after m down jumps and the asset price after m+1 down jumps, this gives the following relationship:

$$\tilde{S}d^m > B > \tilde{S}d^{m+1} \tag{28}$$

The factor d is the multiplicative down move as defined by Cox, Ross and Rubenstein presented earlier. For different values of m, we will get different numbers of time steps.

When I implemented this function I wanted to calculate the option's price at a specific power-rate, I then calculated the m value from the number of time steps where I got the suitable power-rate (see secion 3). In general terms: we assume that we want to calculate the value of a barrier option with a specific number of time-steps, N', we then calculate m' from:

$$m' = \sqrt{\frac{N'(log(\frac{\tilde{S}}{B}))^2}{\sigma^2 T}} \tag{29}$$

Now we calculate m as the smallest integer value bigger than m'. Then we know that the number of time steps given by equation (27) gives the best point closest to but greater than, N', the number of time steps requested.

I've made a calculation using this method and found that the result is very good. Figure 7 shows the values calculated in the same interval as for the standard Cox-Ross-Rubinstein model. Please, make a comparison with figure 6 but don not forget that the scale on the y axes are different.

As you have seen, this model has a disadvantage, the value of the option does not converge strictly, this makes it hard to know when you have reached a correct value. E.g. often you want to double the time step to se how the value converge, but this isn't possible to do for this model. Here you have to look at a much bigger range of different values for different number of time steps to get a picture of the convergence.

5.5 Computing hedge sensitivities

Here I will present how to compute the hedge sensitivities for a binomial tree method. Those hedge parameters that do not differs from the standard way of calculation will not be discussed here. They can be found under section 10 where you also can find definitions of all the hedge sensitivities mentioned in this paper.

Delta, Gamma and Omega is calculated from the binomial tree. We can also calculate

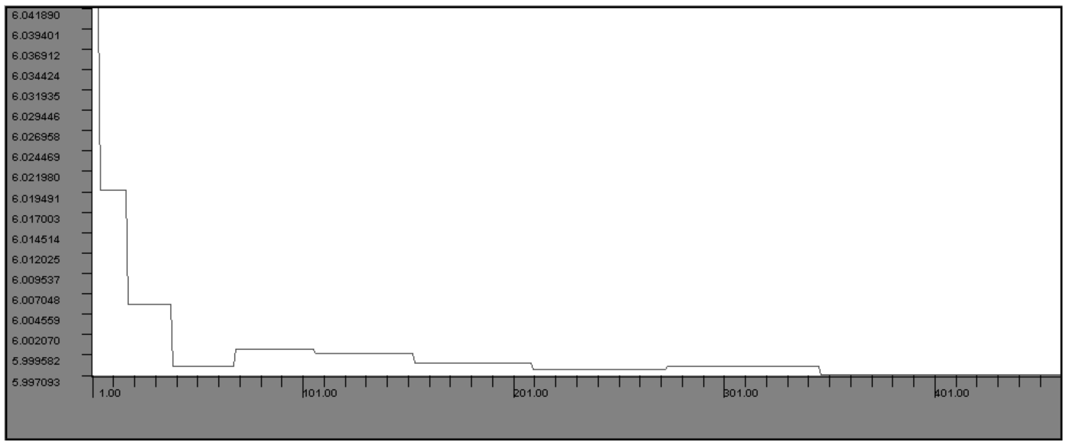

Figure 7: The number of discretizations is chose in a way that minimize the error. Option value is presented on the vertical axis and the power-rate on the horizontal axis.

Theta from the tree in a simple way. To begin with, Delta is calculated as

$$\delta = \frac{\partial \tilde{C}}{\partial \tilde{S}} \simeq \frac{C_{2,2} - C_{2,0}}{S_{2,2} - S_{2,0}}. \tag{30}$$

To be abel to calculate the Delta you must have more values than the one we already have at current time $(0, j)$, for this we need asset values and option values two time steps forward at time $(2, j)$. (Note: These can be saved during the iterative calculation process as earlier described.) The same holds for the Gamma, this parameter is calculated as

$$\gamma = \frac{\partial^2 \tilde{C}}{\partial \tilde{S}^2} \simeq \frac{\frac{C_{\ ,} - C_{\ ,}}{S_{\ ,} - S_{\ ,}} - \frac{C_{\ ,} - C_{\ ,}}{S_{\ ,} - S_{\ ,}}}{\frac{1}{2}(S_{2,2} - S_{2,0})}. \tag{31}$$

The Omega is then calculated as follows;

$$\Omega = \frac{\partial^3 \tilde{C}}{\partial \tilde{S}^3} \simeq \frac{\frac{\frac{C_{\ ,}\quad C_{\ ,}}{S_{\ ,}\quad S_{\ ,}} - \frac{C_{\ ,}\quad C_{\ ,}}{S_{\ ,}\quad S_{\ ,}}}{-(S_{\ ,} - S_{\ ,}\ )} - \frac{\frac{C_{\ ,}\quad C_{\ ,}}{S_{\ ,}\quad S_{\ ,}} - \frac{C_{\ ,}\quad C_{\ ,}}{S_{\ ,}\quad S_{\ ,}}}{-(S_{\ ,} - S_{\ ,}\ )}}{\frac{1}{4}(S_{2,2} - S_{2,0})}. \tag{32}$$

The last hedge parameter that can be obtained directly from the tree is Theta. This is the rate of change of the option price with time. It is calculated as

$$\theta = \frac{\partial \tilde{C}}{\partial t} \simeq \frac{C_{2,1} - C_{0,0}}{2\Delta t}. \tag{33}$$

In the next section we're going to look at a different type of model called Trinomial tree model.

6 Trinomial Tree Model

Here we assume that the asset price follows a geometric Brownian motion (GBM) presented by the stochastic differential equation (SDE) below,

$$d\tilde{S} = \mu \tilde{S} dt + \sigma \tilde{S} dz \tag{34}$$

It is more convenient to work with the log-transformed asset price $y(t) = ln(\tilde{S}(t))$, also described for the Trigeorgis binomial tree model. This transform equation 34 and gives us,

$$dy = \nu dt + \sigma dz, \tag{35}$$

where

$$\nu = \mu - \frac{1}{2}\sigma^2. \tag{36}$$

Before continuing the calculation we will present the model and its jump probabilities. Going from one time step t to $t + \Delta t$ the log-transformed asset price y can change to either $y + \Delta y$ with probability p_u, $y - \Delta y$ with probability p_d or stay the same with probability p_m. The model is illustrated below in figure 8.

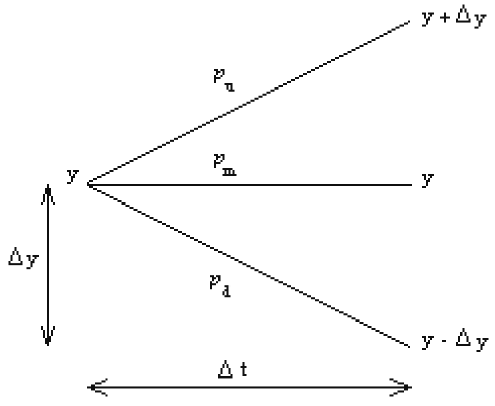

Figure 8: The jump process for the natural logarithm of the asset price for the trinomial tree model.

I have also defined a numbering notation system, which reminds of the numbering system for the binomial model. Normally, the numbering system for the trinomial model has a center with a constant log-transformed asset price at index 0, it is then growing upwards with positive indexes and downwards with negative indexes. When implementing the model in a computer calculation function using vectors for storing the values, this numbering system is disturbing because most programming languages use vectors with non-negative indexes. Therefore we choose the lowest position at each time step in the tree to have index 0. Figure 9 presents this numbering notation system.

Each position in the tree has an index i representing the time step and an index j representing the log-transformed asset price index. The index i goes from 0 to N, at maturity date j goes from 0 at the lowest position in the tree up to 2N at the highest position.

We can now continue the calculation of the jump probabilities and the jump size. The relationship between the parameters of the continuous time process and the trinomial process is obtained by equating the mean and variance over the time interval Δt and requiring that the probabilities sum to one, this gives

$$E[\Delta y] = p_u \Delta y + p_m \cdot 0 - p_d \Delta y = \nu \Delta t, \tag{37}$$

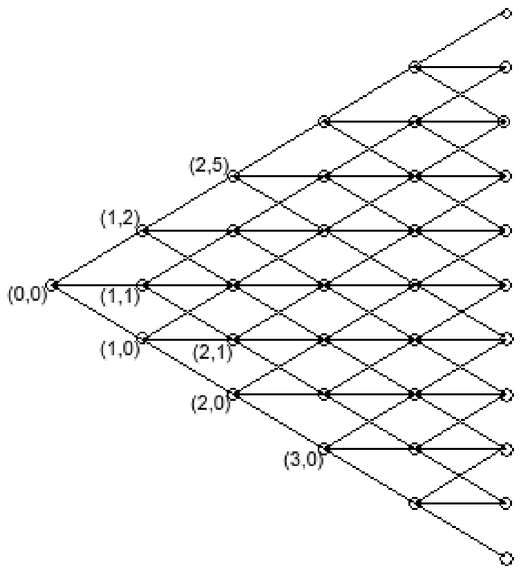

Figure 9: The Trinomial model's numbering system.

$$E[\Delta y^2] = p_u \Delta y^2 + p_m \cdot 0 - p_d \Delta y^2 = \sigma^2 \Delta t + \nu^2 \Delta t^2 \tag{38}$$

and

$$p_u + p_m + p_d = 1. \tag{39}$$

When we solve the equations (37), (38) and (39) with respect to the probabilities, we obtain

$$p_u = \frac{1}{2}\left(\frac{\sigma^2 \Delta t + \nu^2 \Delta t^2}{\Delta y^2 + \frac{\nu \Delta t}{\Delta y}}\right), \tag{40}$$

$$p_m = 1 - \left(\frac{\sigma^2 \Delta t + \nu^2 \Delta t^2}{\Delta y^2 + \frac{\nu \Delta t}{\Delta y}}\right) \tag{41}$$

and

$$p_d = \frac{1}{2}\left(\frac{\sigma^2 \Delta t + \nu^2 \Delta t^2}{\Delta y^2 - \frac{\nu \Delta t}{\Delta y}}\right). \tag{42}$$

To determine the range of asset price values in the grid, we consider what Clewlow and Strickland, [15] (page 65) says; "A reasonable range of asset price values at maturity date of the option is three standard deviations on either side of the mean". So, knowing the standard deviation σ we can calculate the Δy as:

$$\Delta y = \frac{6\sigma\sqrt{T}}{N_i} \tag{43}$$

We can now start to initialize the tree variables, starting with $S_{N_i,j}$ at maturity. We first set the asset price at the lowest position in the tree at maturity date to

$$S_{N_i,0} = \tilde{S}(t_0) e^{-N_i \Delta y}. \tag{44}$$

Then we can calculate all the asset prices above in an iterative way as,

$$S_{N_i,j} = S_{N_i,j-1}e^{\Delta y}. \tag{45}$$

The option prices at maturity day can now be calculated from the following expression,

$$C_{N_i,j} = max(0, S_{N_i,j} - K). \tag{46}$$

We now start the iterative process to calculate the option prices throughout the tree, for each time step back to 0 we will calculate the option prices. In general terms, at time step i we will calculate the prices from $j = 0$ to $j = 2i$ as

$$C_{i,j} = e^{-r\Delta t}\left(p_u C_{i+1,j+1} + p_m C_{i+1,j} + p_d C_{i+1,j-1}\right). \tag{47}$$

This is the discounted expectation where $e^{-r\Delta t}$ is the discount factor. When we reach the current time step at $i = 0$ we can get the option value at $C_{0,0}$. If we want to make the model valid for American style options we have to include an exercise condition at every time step, this condition is the same as when we calculate the option prices at maturity, written in a more general way we get

$$C_{i,j} = max(C_{i,j}, S_{i,j} - K). \tag{48}$$

Here you can see that we need to know the asset price in every node throughout the tree, we can find these values one time step ahead in the tree where

$$S_{i,j} = S_{i+1,j+1}, \tag{49}$$

see figur 9. It is still possible to keep all asset prices in one singel vector, but using equation (49) requires that we start at the bottom $j = 0$ working upwards to $j = 2i$ so that we don't overwrite the data. Finally, we can implement the barrier condition by adding an extra check to see if the barrier is crossed. This is done for every node, if an asset price is below the barrier, that is for a "Down-and-out" call option, the value of the option will be set to zero in that node. That is, if

$$S_{i,j} \leq B \tag{50}$$

then

$$C_{i,j} = 0. \tag{51}$$

As for the standard binomial tree model we get a very alternating behaviour when the current asset price lies near the barrier. Peter Ritchken [1] found a solution for this problem in 1994. He define a stretch parameter, λ, that adjust the lattice so that a horizontal row of nodes always will lie on the barrier. Then, the error as we discussed in section 5 will be limited. In next section we discus how to implement the stretch parameter.

6.1 Implementing a stretch parameter

Implementing a stretch parameter is an efficient way of adjusting the standard trinomial model, it is well known and often used for pricing barrier options. Maybe, that is because the procedure is quite easy to understand and not to hard to implement. I will start with defining the stretch parameter, λ, in general terms by looking at a specific case with a "Down-and-out" option. First we define a variable, n_0, that is the number of down moves that leads to the horizontal layer of nodes just above the barrier. The exact or decimal number of down moves to the barrier is defined as η and is calculated as,

$$\eta = \frac{ln(\frac{\tilde{S}}{B})}{\sigma\sqrt{\Delta t}}. \tag{52}$$

Now we can calculate n_0 as the largest integer value that is smaller than η and then define λ as,

$$\lambda = \frac{\eta}{n_0} \quad when \quad 1 \leq \lambda < 2. \tag{53}$$

Furthermore, we redefine the jump probabilities and the jumpsize adjusted to the λ. The jump size Δy as defined in the standard model is chosen as

$$\Delta y = \lambda \sigma \sqrt{\Delta t} \tag{54}$$

and the jump sizes also defined above, eq. (40) - (42), is chosen as

$$p_u = \frac{1}{2\lambda^2} + \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}, \tag{55}$$

$$p_m = 1 - \frac{1}{\lambda^2} \tag{56}$$

and

$$p_d = \frac{1}{2\lambda^2} - \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma}. \tag{57}$$

Notice that if $\lambda = 1$ the lattice collapses to the usual binomial lattice. As I mentioned, this model is valid in general and not only for the "Down-and-out" barrier option. For an "Up-and-out" barrier option the λ will take a value based on the number of up moves before crossing the barrier instead of the number of down moves.

6.2 Testing the adjusted trinomial model

The convergence is the most important measurement of the options behaviour. This is easiest presented in a graph with the option's price on the vertical axis and the power-rate, i.e., growing number of N_i, on the horizontal axis (see section 3). In figure 10 this is displayed for the adjusted trinomial model where the current asset price lies near the barrier. The standard case parameters are used as described in the introduction of this paper.

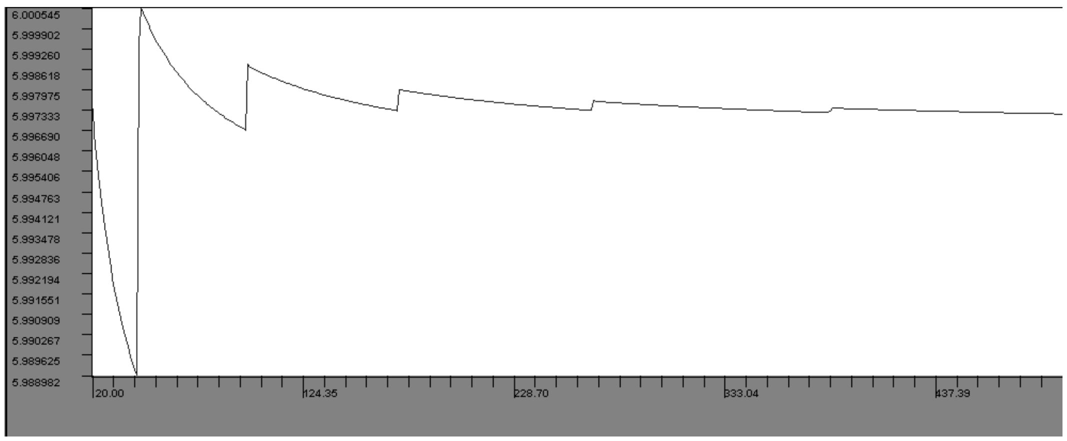

Figure 10: The adjusted trinomial tree model. The option's value is represented on the vertical axis and the power-rate on the horizontal axis.

This model does not converge strictly, you can clearly see the singularities where the price jumps. To see a comparison with other models you should look at section 9 where the adjusted binomial tree model, this adjusted trinomial model and the finite difference model are compared.

6.3 Computing hedge sensitivities

Here I will present how to compute the hedge sensitivities for a trinomial tree model. Those hedge parameters that do not differs from the standard way of calculation will not be discussed here, they can be found under chapter 10 where you also can find definitions of all the other hedge sensitivities mentioned in this paper.

The way of calculating the hedges does not differ much from the way they are calculated from the binomial tree model. Here we can calculate the Delta one time step from current time as

$$\delta = \frac{\partial \tilde{C}}{\partial \tilde{S}} \simeq \frac{C_{1,2} - C_{1,0}}{S_{1,2} - S_{1,0}}. \tag{58}$$

To be able to calculate the Delta we must have more values than the one we already have at current time $(0, j)$, for this we need asset values and option values one time step forward at time $(1, j)$. The same goes for the Gamma, but here we need option values two time steps forward at time $(2, j)$. The Gamma is calculated as

$$\gamma = \frac{\partial^2 \tilde{C}}{\partial \tilde{S}^2} \simeq \frac{\frac{C_{,} - C_{,}}{S_{,} - S_{,}} - \frac{C_{,} - C_{,}}{S_{,} - S_{,}}}{(S_{2,3} - S_{2,1})}. \tag{59}$$

The Omega is evaluated at the third time step from current time $(3, j)$, it is calculated as

$$\omega = \frac{\partial^3 \tilde{C}}{\partial \tilde{S}^3} \simeq \frac{\frac{\frac{C_{,} \quad C_{,}}{S_{,} \quad S_{,}} - \frac{C_{,} \quad C_{,}}{S_{,} \quad S_{,}}}{(S_{,} - S_{,})} - \frac{\frac{C_{,} \quad C_{,}}{S_{,} \quad S_{,}} - \frac{C_{,} \quad C_{,}}{S_{,} \quad S_{,}}}{(S_{,} - S_{,})}}{(S_{3,4} - S_{3,2})}. \tag{60}$$

The last hedge parameter that can be obtained directly from the tree is the Theta. This is the rate of change of the option price with respect to time. It is calculated as

$$\theta = \frac{\partial \tilde{C}}{\partial t} \simeq \frac{C_{1,1} - C_{0,0}}{\Delta t}. \tag{61}$$

In the following section we will look at the Finite Difference model.

7 Finite Difference Model

I will present for you a robust and fast finite difference model that converges to a correct value much faster than earlier published methods do, see the titles below. The method is robust because its stability depends on neither the number of time steps nor it's number of price discretizations between the current price of the underlying asset and the barrier. This method converges fast when the price of the underlying asset is close to the barrier. That is because the nodes of the grid hit the barrier and the current price for the underlying asset exactly. Furthermore, there is always at least one step of discretization between the current price of the underlying asset and the barrier. I will compare this method with some earlier published work on the field;

- "On pricing barrier options" by Peter Ritchken, [1] (Trinomial Tree Model).
- "An explicit finite difference approach to the pricing of barrier options", by P. Boyle and Y. Tian, [2].
- "Application of Finite Difference Method for Pricing Barrier Options" by G. Ioffe and M. Ioffe, [3].

When comparing different models for pricing derivatives it's normally the number of discretizations that is considered, but different numerical models use different types of discretizations. Therefore, we will mainly consider the calculation time as the dimensioning factor in this section.

What we are going to do is to solve the Black-Scholes Partial Differential Equation (PDE). That is done using numerical approximations of the derivatives. We rewrite the Black-Scholes PDE equation 1 presented in section 4,

$$\frac{\partial \tilde{C}}{\partial t} + r\tilde{S}\frac{\partial \tilde{C}}{\partial \tilde{S}} + \frac{1}{2}\sigma^2\tilde{S}^2\frac{\partial^2 \tilde{C}}{\partial \tilde{S}^2} = r\tilde{C}.$$

Before continuing we have to do some changes in the PDE because we want to log-transform the asset price. That is,

$$y(t) = log(\tilde{S}(t)). \tag{62}$$

Boyle and Tian [2] show how to write the log-transformed Partial Differential Equation (PDE), where the outcome is,

$$\frac{\partial \tilde{C}}{\partial t} + (r - \frac{\sigma^2}{2})\frac{\partial \tilde{C}}{\partial y} + \frac{1}{2}\sigma^2\frac{\partial^2 \tilde{C}}{\partial y^2} = r\tilde{C}. \tag{63}$$

The derivatives can be defined in several different ways, here we'll look at three different approximations, first an explicit model and then two different implicit models.

7.1 Explicit approach

The model I will present here was proposed by P. Boyle and Y. Tian in 1998 [2]. Their model, which they refer to as "Modified Explicit Finite Difference" (MEFD) makes the option price converge to a theoretically correct value much faster than earlier proposed models, such as the binomial tree model [4] or the trinomial tree model [1]. Their main idea is to put a horizontal line of nodes on the barrier. The drawback is that, most of the time, there will be no node that is laying on the current asset price implying that no node will possess the right option value. They propose a way of interpolate the option value at current asset price by quadratic interpolation using the closest nodes, above and below the current asset price. To get these values at current time, t_0 we also need to make a calculation one time step back in time, t_{-1}, this makes the implementation a bit harder, but more about that later.

This model is explicit, which means that by knowing the option values at time step $i+1$, makes it possible to calculate the value of the option at the current time step, i. This is illustrated in figure 11 where the option value $C_{i,j}$ kan be calculated knowing $C_{i+1,j+1}$, $C_{i+1,j}$ and $C_{i+1,j-1}$.

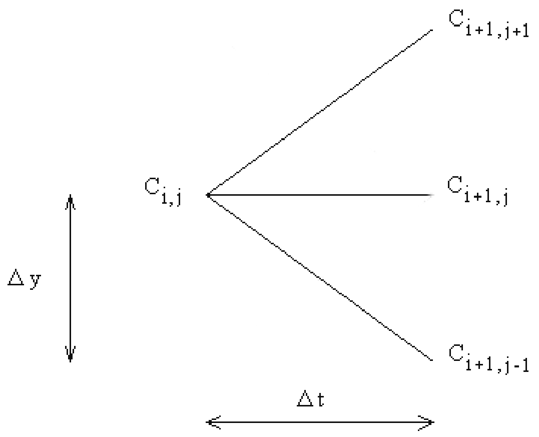

Figure 11: An explicit finite difference model

The evaluation point is $(i\Delta t, S_{i,j})$ where we denote,

$$\tilde{C} = \tilde{C}(i\Delta t, S_{i,j}) \simeq C_{i,j}. \tag{64}$$

How are the derivatives approximated in that point? We use numerical approximation where the derivatives is calculated as,

$$\frac{\partial \tilde{C}}{\partial t} \simeq \frac{C_{i+1,j} - C_{i,j}}{\Delta t}, \tag{65}$$

$$\frac{\partial \tilde{C}}{\partial y} \simeq \frac{C_{i+1,j+1} - C_{i+1,j-1}}{2\Delta y} \tag{66}$$

and

$$\frac{\partial^2 \tilde{C}}{\partial y^2} \simeq \frac{C_{i+1,j+1} - 2C_{i+1,j} + C_{i+1,j-1}}{\Delta y^2}. \tag{67}$$

We can now put the derivatives into the PDE, eq. (63), and then we get an explicit expression of the option price $C_{i,j}$ at a node i,j as,

$$\frac{C_{i+1,j} - C_{i,j}}{\Delta t} + (r - \frac{\sigma^2}{2})\frac{C_{i+1,j+1} - C_{i+1,j-1}}{2\Delta y} + \frac{1}{2}\sigma^2 \frac{C_{i+1,j+1} + C_{i+1,j-1} - 2C_{i+1,j}}{\Delta y^2} = rC_{i,j}. \tag{68}$$

If we rewrite this equation we can see the explicit behaviour,

$$C_{i,j} = p_u C_{i+1,j+1} + p_m C_{i+1,j} + p_d C_{i+1,j-1} \tag{69}$$

where p_u p_m p_d are given by

$$p_u = \Delta t \left(\frac{\sigma^2}{2\Delta y^2} + \frac{\nu}{2\Delta y} \right), \tag{70}$$

$$p_m = 1 - \Delta t \frac{\sigma^2}{\Delta y^2} - r\Delta t, \tag{71}$$

$$p_d = \Delta t \left(\frac{\sigma^2}{2\Delta y^2} - \frac{\nu}{2\Delta y} \right) \tag{72}$$

and

$$\nu = r - \frac{\sigma^2}{2}. \tag{73}$$

We now have all the formulas for starting the implementation process. In the following section I will show how this model can be implemented and how the quadratic interpolation is done.

7.1.1 Implementation

As before we will start the calculation by defining a grid, which has a asset price spread of $3\sigma\sqrt{T}$ on either side of the current asset price, as described in section 5. We will have N_i time steps between current time and maturity time and $2N_i + 2$ discretizations of the asset price. Note that the grid is built from one step back in time, t_{-1}. Starting at the lowest nodes $j = 0$ the grid is growing upwards to the highest layer of nodes at index $j = N_j = 2N_i + 2$. Now, we choose the barrier to lay on a horizontal line of nodes with $j = N_d$. Because there are no nodes laying on the current asset price we have to find the layer of nodes laying closest. Defining y_0 as the log-transformed current asset price, Boyle and Tian propose a way of finding the number of nodes, j_0, between the barrier and the nodes closest to the current asset price as,

$$j_0 = \langle \frac{y_0 - y_d}{\Delta y} + 0.5 \rangle \tag{74}$$

where $\langle . \rangle$ returns the integer portion of the argument. We know that there are $N_i + 1$ layers of nodes below this point, the barrier will then lay on a layer with index $j = N_i + 1 - j_0$ because the layer of nodes laying closest to the current asset price has index $j = N_i + 1$. The grid is illustrated in figure 12. To calculate the asset prices at different nodes going from

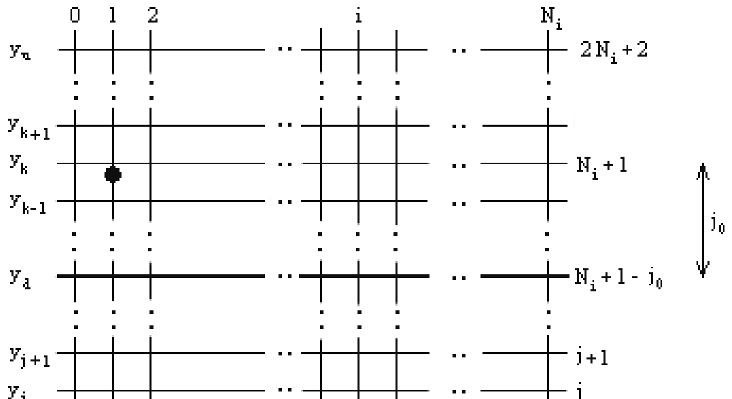

Figure 12: An explicit finite difference model where y_d lies on the barrier and y_k is the layer of nodes that is closest to the current asset price. The uppermost layer of nodes lay at y_u.

$j = 0$ up to $j = N_j$ we start with calculating the lowest asset price as,

$$S_0 = e^{y_d - (N_i + 1 - j\ )\Delta y}. \tag{75}$$

From this position we can easily calculate the rest of the asset prices simply by adding a factor $e^{\Delta y}$ to the asset price one level below, in an iterative way this can be done using the following expression,

$$S_j = e^{\Delta y} S_{j-1}. \tag{76}$$

Now we can compute the option values at maturity date by adding the exercise condition; if the barrier is crossed then,

$$C_{N_i+1,j} = max(0, S_j - K) \tag{77}$$

else

$$C_{N_i+1,j} = 0. \tag{78}$$

In general,

$$\Delta y = \lambda \sigma \sqrt{\Delta t} \tag{79}$$

but the λ has to be chosen in a way so that the model is stable, see section 7.1.2. Here we choose,

$$\lambda = \sqrt{3} \tag{80}$$

which according to Boyle and Tian gives better accuracy of the option price then other λ. As I mentioned before the time interval goes from t_{-1} to t_T and does not start at current time, t_0, as normally. This makes the implementation of the model confusing because the arrays, as I discussed in section 5, starts at index 0 in most programming languages. When the calculations at maturity day is done we have N_i iterations left to do, starting at t_{T-1} stepping back until we reaches t_0. Within this iterative process we will calculate the option values with the barrier condition added as follows; if the barrier is crossed then,

$$C_{i,j} = p_u C_{i+1,j+1} + p_m C_{i+1,j} + p_d C_{i+1,j-1} \tag{81}$$

else,

$$C_{i,j} = 0. \tag{82}$$

Then we add the exercise condition if it is an American style option as,

$$C_{i,j} = max(C_{i,j}, 0). \tag{83}$$

When writing the code, only one array is needed for storing data and not a matrix as above because the model is explicit and the old values can be overwritten.

As you have seen, we do not need to calculate the values at t_{-1}, it is just the grid that is built that way so we get three values at current time, t_0. The option values located in the array C, while finished the iteration process, is used to calculate the option value at current asset price. Boyle and Tian describe the basic principle of the quadratic interpolation as follows; *Imagine a real-valued function $g(x)$. Suppose that we wish to obtain the value of the function at x_0. The functional form of g may not be known, but the values of the function at x_1, x_2, and x_3 are known and denoted by $g(x_1)$, $g(x_2)$ and $g(x_3)$, respectively. Without loss of generality, let $x_1 < x_2 < x_3$. If $x_0 \in (x_1, x_3)$ the value of $g(x_0)$ may be approximated by:*

$$g(x_0) \approx \left(\frac{x_0 - x_2}{x_1 - x_2}\right)\left(\frac{x_0 - x_3}{x_1 - x_3}\right) g(x_1) + \left(\frac{x_0 - x_1}{x_2 - x_1}\right)\left(\frac{x_0 - x_3}{x_2 - x_3}\right) g(x_2) + \left(\frac{x_0 - x_1}{x_3 - x_1}\right)\left(\frac{x_0 - x_2}{x_3 - x_2}\right) g(x_3) \tag{84}$$

I have written a simple function for this quadratic interpolation in which I put the output values from the iterative calculation process, with Boyle and Tian notation it gives,

$x_3 = log(S_{N_i+2})$ $\quad g_3 = C_{N_i+2}$
$x_2 = log(S_{N_i+1})$ $\quad g_2 = C_{N_i+1}$
$x_1 = log(S_{N_i})$ $\quad g_1 = C_{N_i}$
$x_0 = log(\tilde{S}(t_0))$

Here, the current asset price, $\tilde{S}$, is closest to S_{N_i+1} as defined in the beginning of this section.

In this model we have a one-dimensional freedom λ as shown above, this parameter can be chosen in a way so that both the current asset price and the barrier is hit exactly by a layer of nodes. In this case the model has collapsed to the trinomial tree model, see figure 6, which gives a faster convergence.
If we have two barriers, the parameter can be chosen so that both barriers but not the current asset price are hit exactly. Without any tests, I believe that this model will be both accurate and fast for pricing this type of contract. Furthermore, the implicit model presented later will maybe deliver an even faster convergence if this interpolation method is used. More about this in section 13, "Further Development". Now we are going to look at the stability and convergence conditions for the explicit model.

7.1.2 Stability and convergence

The stability is important to study in order to know in which cases the model is applicable. As you have already seen, the trinomial tree model can also be derived from the PDE, eq. 1 and not only as we did in section 6. This makes the stability conditions derived below valid for the trinomial tree as well.

We want the stability conditions for the transformed PDE, eq. (63), but those are the same as for the homogeneous equation below,

$$\frac{\partial \tilde{C}}{\partial t} + (r - \frac{\sigma^2}{2})\frac{\partial \tilde{C}}{\partial y} + \frac{1}{2}\sigma^2 \frac{\partial^2 \tilde{C}}{\partial y^2} = 0. \tag{85}$$

If we apply the discrete derivatives, eq. (65) - (67) as before we will get the following discrete partial differential equation,

$$\frac{C_{i+1,j} - C_{i,j}}{\Delta t} + (r - \frac{\sigma^2}{2})\frac{C_{i+1,j+1} - C_{i+1,j-1}}{2\Delta y} + \frac{1}{2}\sigma^2 \frac{C_{i+1,j+1} + C_{i+1,j-1} - 2C_{i+1,j}}{\Delta y^2} = 0. \tag{86}$$

If we rewrite this equation we can see the explicit behaviour as earlier but with different jump probabilities,

$$C_{i,j} = p_1 C_{i+1,j+1} + p_2 C_{i+1,j} + p_3 C_{i+1,j-1}. \tag{87}$$

Here p_1, p_2 and p_3 are given by,

$$p_1 = \Delta t \left(\frac{\sigma^2}{2\Delta y^2} + \frac{\nu}{2\Delta y} \right), \tag{88}$$

$$p_2 = 1 - \Delta t \frac{\sigma^2}{\Delta y^2} \tag{89}$$

and

$$p_3 = \Delta t \left(\frac{\sigma^2}{2\Delta y^2} - \frac{\nu}{2\Delta y} \right). \tag{90}$$

If we now add the dependency condition between Δy and Δt, eq. (79), we get the following three equations,

$$p_1 = \frac{1}{2\lambda^2} + \frac{\nu\sqrt{\Delta t}}{2\lambda\sigma}, \tag{91}$$

$$p_2 = 1 - \frac{1}{\lambda^2} \tag{92}$$

and

$$p_3 = \frac{1}{2\lambda^2} - \frac{\nu\sqrt{\Delta t}}{2\lambda\sigma}. \tag{93}$$

A sufficient condition for stability is that p_1, p_2 and p_3 is non-negative, see Boyle and Tian [2]. The λ can not be negative because the steps of discretization can not be negative. We start with looking at p_2 that restrict the stability as,

$$\lambda \geq 1. \tag{94}$$

The other conditions $p_3 \geq 0$ and $p_1 \geq 0$ are satisfied if the restrictions below are fulfilled,

$$0 \leq \lambda \leq \frac{\sigma}{|\nu| \sqrt{\Delta t}}. \tag{95}$$

For the trinomial model proposed by Ritchken, [1], we have another condition for the λ that makes both the barrier and the current asset price hit by a node, se section 6. Then, the stability condition makes it difficult to choose the best-suited time steps and jump sizes. For the implicit models below, we'll have no such problems. They are allways stable and therefore we can choose time steps and jump sizes independently, this gives this model an advantage to the explicit because it is easier to adjust these parameters for different contracts, i.e. contracts where the current asset price is close to the barrier.

7.2 Implicit, Cranc-Nicolson

This method is a so-called fully centered method, which means that it replaces the space and time derivatives with finite differences centered at an imaginary time step at $(i + \frac{1}{2})$. The dependencies for this model is described in figure 13.

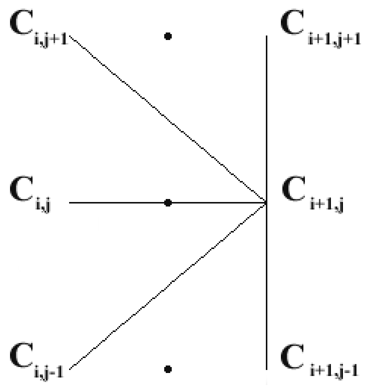

Figure 13: An implicit finite difference model using Cranc-Nicolson.

We see here that the evaluation point position is $((i + \frac{1}{2})\Delta t, S_{i,j})$, which means that we approximate the continuous option value in that point to the following value,

$$\tilde{C} = \tilde{C}((i + \frac{1}{2})\Delta t, S_{i,j}) \simeq \frac{C_{i+1,j} + C_{i,j}}{2}. \tag{96}$$

The partial derivatives are approximated with Crank-Nicolson and are described in discrete terms in the grid as follows;

$$\frac{\partial \tilde{C}}{\partial t} \simeq \frac{C_{i+1,j} - C_{i,j}}{\Delta t}, \tag{97}$$

$$\frac{\partial \tilde{C}}{\partial y} \simeq \frac{C_{i+1,j+1} - C_{i,j+1} - C_{i+1,j-1} + C_{i,j-1}}{4\Delta y}, \tag{98}$$

and

$$\frac{\partial^2 \tilde{C}}{\partial y^2} \simeq \frac{(C_{i+1,j+1} - 2C_{i+1,j} + C_{i+1,j-1}) + (C_{i,j+1} - 2C_{i,j} + C_{i,j-1})}{2\Delta y^2}. \tag{99}$$

The PDE, eq. (63), can now be presented in a discrete way,

$$\frac{C_{i+1,j}-C_{i,j}}{\Delta t}+(r-\frac{\sigma^2}{2})\frac{C_{i+1,j+1}-C_{i,j+1}-C_{i+1,j-1}+C_{i,j-1}}{4\Delta y}+$$
$$\frac{1}{2}\sigma^2\frac{(C_{i+1,j+1}-2C_{i+1,j}+C_{i+1,j-1})+(C_{i,j+1}-2C_{i,j}+C_{i,j-1})}{2\Delta y^2}=r\frac{C_{i+1,j}+C_{i,j}}{2}. \quad (100)$$

Rewritten as

$$p_uC_{i,j+1}+p_mC_{i,j}+p_dC_{i,j-1}=-p_uC_{i+1,j+1}-(p_m-2)C_{i+1,j}-p_dC_{i+1,j-1}. \quad (101)$$

Were the p_u, p_m and p_d are given by,

$$p_u=-\frac{1}{4}\Delta t\left(\frac{\sigma^2}{\Delta y^2}+\frac{\mu}{\Delta y}\right), \quad (102)$$

$$p_m=1+\Delta t\frac{\sigma^2}{2\Delta x^2}+\frac{r\Delta t}{2} \quad (103)$$

and

$$p_d=-\frac{1}{4}\Delta t\left(\frac{\sigma^2}{\Delta y^2}-\frac{\mu}{\Delta y}\right). \quad (104)$$

7.2.1 Construction of the grid

The main idea with this method is to place both the barrier and the spot price on the grid's nodes. This is always possible while dealing with single barrier options as in this paper. To determine the range of asset price values in the grid we consider what Clewlow and Strickland, [15] page 65, says; "A reasonable range of asset price values at maturity date of the option is three standard deviations either side of the mean". To make this model converge to the sixth decimal we must choose an even bigger range than proposed above, for the standard option contract a range of 10 standard deviations above the current asset price is enough.

To understand the complex grid presented below better, we look at figure 14 where we can se all the variables and their position in the grid.

The lowest nodes in the grid is placed on the barrier, that is because we know that all option values below the barrier is zero (for a "Down-and-out" call option). We then decide how many nodes we want between the barrier and the spot price. If the model shall be valid, the minimum amount of discretizations, k, is equal to one because we must do at least one calculation in this area. When we have determined the number of nodes between the two critical points in the grid, we are going to decide upon how many steps of discretization we will have above the node laying on the spot price.

We start with determine the Δy as:

$$\Delta y=\frac{y_0-y_d}{k}. \quad (105)$$

Where

$$y_0=log(\tilde{S}) \quad (106)$$

and

$$y_d=log(B). \quad (107)$$

The log-transformed asset price at the uppermost horizontal layer of nodes, y_{N_j}, will lie next above y_u which is calculated as below,

$$y_u=log((1+3\sigma\sqrt{3T})\tilde{S}). \quad (108)$$

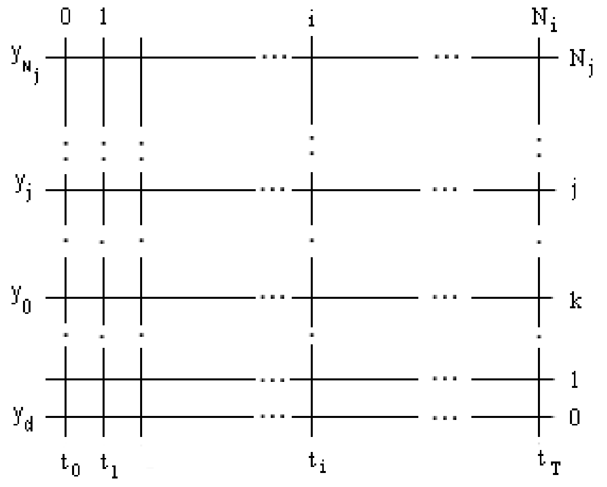

Figure 14: The grid used for the Cranc-Nicolson implicit model. Here, y_d lies on the barrier and y_k lies on the current asset price. The uppermost layer of nodes, y_{N_j}, lies next above y_u.

When having the discretisation step of the stock price, we can determine how many nodes N_j that are required to get the needed price spread. N_j must be an integer value, so we have to choose the uppermost node to lie above the upper limit of the price spread. Which gives,

$$N_j = \langle k\frac{y_u - y_d}{y_0 - y_d}\rangle = \langle\frac{y_u - y_d}{\Delta y}\rangle. \tag{109}$$

Where $\langle\cdot\rangle$ is the operator that returns the upper integer value of the argument. Then, the price spread of the underlying asset in the model goes from the barrier, B, at position 0 up to S_{N_j} at position N_j as,

$$S_{N_j} = e^{y_d + N_j \Delta y}. \tag{110}$$

The stock price at node j is calculated as,

$$S_j = e^{y_d + j\Delta y}. \tag{111}$$

When we have determined the stock price for the nodes, we will determine the discretisation in time between the current time and the maturity date. How many time steps N_i are reasonable to choose? The model works with any number of steps greater than 0, but of course, one step won't give an accurate value. I have chosen $N_i = \frac{N_j}{2}$ which makes the model converge fast. Then, the time step Δt is given by,

$$\Delta t = \frac{T}{N_i}. \tag{112}$$

We have chosen a spread of the stock price for the grid because we want the value at the current time, node (t_0, y_k), to depend of all values at all nodes on this interval. In this model I have chosen Cranc-Nicolson as the discretization model, this is an implicit method because it requires knowing the values not only one time step forward but also the values above and below the present value. This gives us an equation system that includes all the equations given at each node one step back from maturity. That is, the value at the current time, node (t_0, y_k), will always more or less depend on the uppermost value at maturity.

If we take equation (101) and generate an equation for each j where $1 \leq j \leq N_j$, we will get an equation system with the appearance as below:

$$
\begin{bmatrix}
1 & -1 & 0 & & & & \\
p_u & p_m & p_d & 0 & & & \\
0 & p_u & p_m & p_d & 0 & & \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & 0 & p_u & p_m & p_d & 0 \\
 & & & 0 & p_u & p_m & p_d \\
 & & & & 0 & p_u & p_m
\end{bmatrix}
\begin{bmatrix}
C_{i,N_j} \\ C_{i,N_j-1} \\ C_{i,N_j-2} \\ C_{i,N_j-3} \\ \vdots \\ C_{i,4} \\ C_{i,3} \\ C_{i,2} \\ C_{i,1}
\end{bmatrix}
=
$$

$$
\begin{bmatrix}
\lambda_u \\
-p_u C_{i+1,N_j} - (p_m - 2)C_{i+1,N_j-1} - p_d C_{i+1,N_j-2} \\
-p_u C_{i+1,N_j-1} - (p_m - 2)C_{i+1,N_j-2} - p_d C_{i+1,N_j-3} \\
-p_u C_{i+1,N_j-2} - (p_m - 2)C_{i+1,N_j-3} - p_d C_{i+1,N_j-4} \\
\vdots \\
-p_u C_{i+1,5} - (p_m - 2)C_{i+1,4} - p_d C_{i+1,3} \\
-p_u C_{i+1,4} - (p_m - 2)C_{i+1,3} - p_d C_{i+1,2} \\
-p_u C_{i+1,3} - (p_m - 2)C_{i+1,2} - p_d C_{i+1,1} \\
-p_u C_{i+1,2} - (p_m - 2)C_{i+1,1}
\end{bmatrix}
\tag{113}
$$

7.2.2 Boundary conditions

The upper and the lower equation in the system above are results of two boundary conditions. First, the upper one is a result of the assumption that the rate of change of the option value with respect to the stock price is 1 far from the current asset price, i.e.;

$$\frac{\partial \tilde{C}}{\partial \tilde{S}} \approx 1 \quad if \quad S_j >> S_{y_k} \tag{114}$$

gives

$$C_{i,N_j} - C_{i,N_j-1} = \lambda_u \tag{115}$$

where

$$\lambda_u = S_{N_j} - S_{N_j-1}. \tag{116}$$

The barrier gives the lower boundary condition. On the barrier, the option values are equal to zero. Applied on equation (101) at the center position, $j = 1$ gives

$$C_{i,j-1} = 0, \tag{117}$$

$$C_{i+1,j-1} = 0 \tag{118}$$

and the following equation,

$$p_u C_{i,2} + p_m C_{i,1} = -p_u C_{i+1,2} - (p_m - 2)C_{i+1,1}. \tag{119}$$

The option values are known at maturity date, this gives the final boundary condition necessary for starting the calculation.

We solve the equation system for each time step starting at the maturity date to get the option values one time-step back in time. The equation system to solve is tri-diagonal, which makes the calculation more efficient with respect to calculation time.

Next we are going to look at the third finite difference model; this one is implicit but not fully centered.

7.3 Implicit, not fully centered

We will finally take a look at another implicit finite difference model. The implementation is made almost the same way as for the one above using Cranc-Nicolson approximations. The difference is the way of approximating the derivatives. I am going to show you that this model is not as fast as the Cranc-Nicolson, though, it is interesting to see the difference in convergence for the models. Even though a fully centered model as the one using Crank-Nicolson is faster with respect to the number of discretizations, it is not necessarily faster with respect to calculation time. The equation system for the Cranc-Nicolson implementation is more complex which may make a difference. Moreover, the result retrieved from this model was presented in a working paper by I. Ioffe and M. Ioffe [3], but, the Implementation I will present below does not converge as fast as their similar implementation. The reason for this is hard to say since they have not described their implementation.

To get a clear idea how the derivatives is computed we look at figure 15 that illustrates this. As you see, standing at time step i, the option price one time step forward is calculated

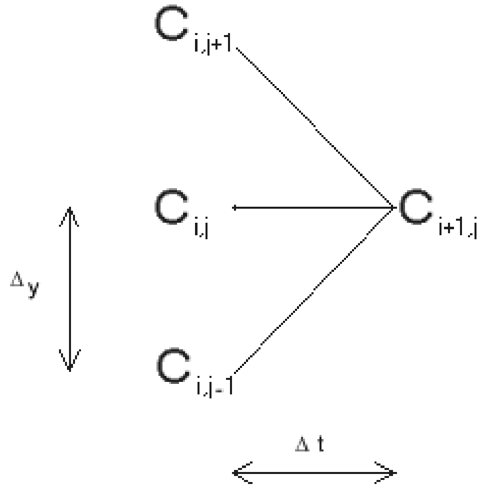

Figure 15: An Implicit Finite Difference model

from the option prices at position i as well as from the option prices at position i+1. This is the implicit behaviour in this model. We want to calculate the option prices from values we do not have explicitly, therefore we have to solve an equation system, (128), that is derived below.

The continuous option value at time, $i\Delta t$ and asset price $S_{i,j} = e^{j\Delta y}$, i.e. $\tilde{C}(i\Delta t, S_{i,j})$ is approximated to the value in the grid at position (i, j),

$$\tilde{C} = \tilde{C}(i\Delta t, S_{i,j}) \simeq C_{i,j}. \tag{120}$$

Before presenting that equation we approximate the following derivatives as,

$$\frac{\partial \tilde{C}}{\partial t} \simeq \frac{C_{i+1,j} - C_{i,j}}{\Delta t}, \tag{121}$$

$$\frac{\partial \tilde{C}}{\partial y} \simeq \frac{C_{i,j+1} - C_{i,j-1}}{2\Delta y} \tag{122}$$

and

$$\frac{\partial^2 \tilde{C}}{\partial y^2} \simeq \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta y^2}. \tag{123}$$

Which gives us the following equation when approximating the continuous derivatives in the transformed Black-Scholes PDE, eq. (63),

$$p_u C_{i,j+1} + p_m C_{i,j} + p_d C_{i,j-1} = C_{i+1,j} \tag{124}$$

where

$$p_u = -\frac{1}{2}\Delta t \left(\frac{\sigma^2}{\Delta y^2} + \frac{\nu}{\Delta y} \right), \tag{125}$$

$$p_m = 1 + \Delta t \frac{\sigma^2}{\Delta y^2} + r\Delta y \tag{126}$$

and

$$p_d = -\frac{1}{2}\Delta t \left(\frac{\sigma^2}{\Delta y^2} - \frac{\nu}{\Delta y} \right). \tag{127}$$

Now we can write the equation system,

$$\begin{bmatrix} 1 & -1 & 0 & & & & \\ p_u & p_m & p_d & 0 & & & \\ 0 & p_u & p_m & p_d & 0 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 0 & p_u & p_m & p_d & 0 \\ & & & 0 & p_u & p_m & p_d \\ & & & & 0 & p_u & p_m \end{bmatrix} \begin{bmatrix} C_{i,N_j} \\ C_{i,N_j-1} \\ C_{i,N_j-2} \\ \vdots \\ C_{i,3} \\ C_{i,2} \\ C_{i,1} \end{bmatrix} = \begin{bmatrix} \lambda_u \\ C_{i+1,N_j-1} \\ C_{i+1,N_j-2} \\ \vdots \\ C_{i+1,3} \\ C_{i+1,2} \\ C_{i+1,1} \end{bmatrix} \tag{128}$$

As described before in section 7.2 this system is solved using an algorithm solving tri-diagonal equation systems efficiently. This is important, if using a function based on Gaussian elimination the calculation will be much slower. For further description of the implementation I direct you to the previous section 7.2. The results retrieved from the different models can be viewed in next section where we make a comparison between all the Finite Difference models.

7.4 Comparison between the Finite Difference Models

As I mentioned in the beginning of this paper, the main idea when comparing different models is not the number of time steps required to reach a certain level of accuracy, but to compare the total calculation time. Though, there are some exceptions, when comparing one of the models presented in this paper with a similar model proposed in some other paper, the lack of calculation time data makes this legit. In the working paper by M. Ioffe and I. Ioffe, [3], the implementation is not presented, so I only have the results and have been unable to reproduce the exact results. The results from the two similar models are presented in table 1.

We can see that the results retrieved from the model proposed by Ioffe and Ioffe converge faster than those coming from my non fully centered implicit model. My implementation produce results less accurate than their implementation, so we come to conclusion that it is possible to make a more efficient implementation. Therefore, I should compare the model proposed by M. Ioffe and I. Ioffe, and not my non fully centered implicit model, with other different models to evaluate which one that is the fastest with respect to the calculation time. Anyway, the difference is not that big and we can use the results retrieved here to continue the evaluation process.

When generating the results presented in table 1 I used the ratio $N_i = \frac{N_j}{2}$ which I believe Ioffe and Ioffe have chosen in their model too. But if we choose a greater N_i we get a better result, i.e., if we choose $N_i = 2N_j$ the result will converge faster with respect to calculation time, though, N_j will not be the same in the two cases. This behaviour is, probably, caused by the approximation of the derivatives, the accuracy depends more on how many time properties there are than how many discretizations of the asset price that is done, see table 2.

Number of time partitions	Down and Out Call price (Ioffe-Ioffe [3], implicit)	Down and Out Call price not fully centered
50	5.9721	5.9780
100	5.9860	5.9875
200	5.9928	5.9922
400	5.9950	5.9945
800	5.9960	5.9957
1000	5.9962	5.9959
2000	5.9965	5.9964
3000	5.9966	5.9965
4000	5.9967	5.9966
Accurate value	5.9968	5.9968

Table 1: Comparison between the implicit model proposed by Ioffe and Ioffe, [3], and a non fully centered implicit model presented in section 7.3.

As I mentioned above, the explicit model collapse into the trinomial model if we try to put both the current asset price and the barrier on a layer of nodes. Therefore, this model will not be taken into consideration here because it will be treated in section 9 where the best one of all the different types of models is compared. Here we want to find which one of the two implicit models that is producing the fastest convergence with respect to time.

In table 2 we can see the difference when comparing option prices, with respect to calculation time, retrieved from different models. Here, we have implemented an non fully centered implicit model with a ratio, $N_i = \frac{N_j}{2}$ as well as one with a ratio $N_i = 10N_j$. The third model shown in the table is the Implicit model using Crank-Nicolson approximations.

Six decimals is way to exact when pricing options, but when evaluating which one that produces the best results we have to look at this many decimals because, as you see in the right column, the price has already converged to four decimals after 10 ms. Even if we improve the non fully centered model as in the middle column we still have a great difference, this model doesn't converge to four decimals before calculation time greater than 128000 ms.

We are also going to look at a graphical representation of the results retrieved from these models in figure 16. Finally, we can appoint the model using Cranc-Nicolson approximations to be the fastest of the models presented in this section. To see comparisons between the different types of models, please go to section 9. Next we are going to look at the Monte Carlo simulation model.

Calculation time (± 5ms)	Down and Out Call price Implicit, section 7.3 $N_i = \frac{N_j}{2}$	Down and Out Call price Implicit, section 7.3 $N_i = 10N_j$	Down and Out Call price Implicit, Cranc-Nicolson
10	5.982606 (146,65,130)	5.993079 (12,300,30)	5.996892 (115,53,107)
20	5.986693 (216,91,183)	5.994016 (26,380,38)	5.996857 (200,88,157)
30	5.988218 (256,107,214)	5.994539 (39,450,45)	5.996858 (230,99,199)
40	5.989024 (286,118,237)	5.994929 (49,530,53)	5.996857 (255,107,214)
50	5.990107 (338,137,275)	5.995205 (51,610,61)	5.996854 (280,118,237)
100	5.992115 (480,195,390)	5.995680 (89,840,84)	5.996847 (445,179,359)
200	5.993126 (620,248,497)	5.996043 (131,1220,122)	5.996844 (600,241,482)
500	5.994304 (920,363,727)	5.996268 (199,1680,168)	5.996843 (900,355,711)
1000	5.995032 (1300,509,1018)	5.996456 (290,2440,244)	5.996842 (1300,509, 1018)
2000	5.995594 (1900,738,1477)	5.996563 (410,3360,336)	5.996842 (1800,700,1400)
4000	5.995926 (2600,1006,2013)	5.996641 (580,4660,466)	5.996842 (2600,1006,2013)
8000	5.996179 (3600,1389,2778)	5.996704 (850,6730,673)	
16000	5.996373 (5100,1963,3926)	5.996744 (1200,9410,941)	
32000	5.996495 (6900,2652,5304)	5.996772 (1700,13240,1324)	
64000	5.996600 (9900,3800,7600)	5.996792 (2400,18590,1859)	
128000	5.996664 (13500,5178,10356)	5.996806 (3300,25480,2548)	
256000	5.996716 (19000,7283,14566)	5.996818 (4900,37730,3773)	

Table 2: Comparison between two different implicit models, the non fully centered and one with Cranc-Nicolson which is fully centered. Two different implementations of the non fully centered model are present. The comparison is made with respect to calculation time.

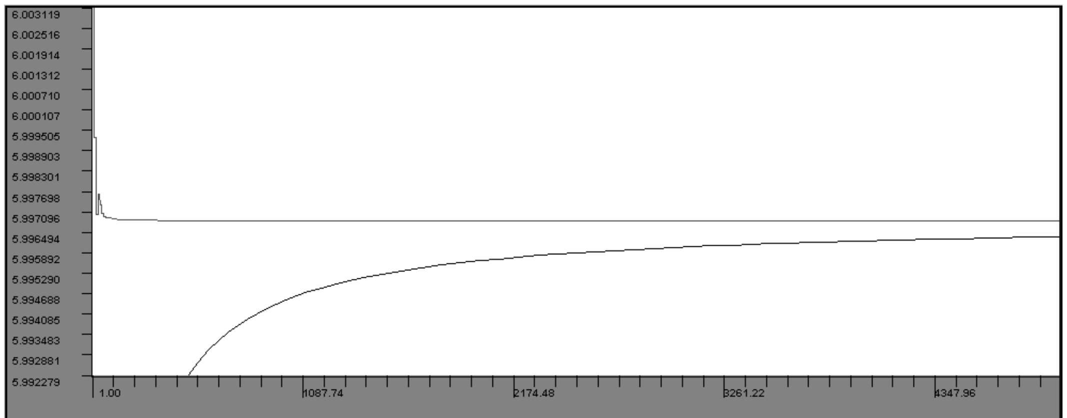

Figure 16: Graphs representing two implicit models, one using central difference and the other Cranc-Nicolson approximations to the derivatives

8 Monte Carlo Simulation

Monte Carlo simulation is a different way of estimating an option's price. It differs from the tree approaches earlier described. For this model we make many simulations and the option value is the average of the outcomes. It is well known that the Monte Carlo simulation converge slow compared to the tree approaches for pricing path-dependent derivatives. Hull [16] for example says: *The main problem in using Monte Carlo simulation to value path-dependent derivatives is that the computation time necessary to achieve the required level of accuracy can be unacceptably high.* Furthermore, American-style path-dependent derivatives cannot be handled.

In the following section I am going to show, as an educationally example, the theory behind the model and how the model can be implemented.

We will make M simulations when pricing an option, for each simulation we'll simulate the asset price's path with N_i time steps from the start at current time, t_0, until maturity, t_T. The value of the option, $C_{0,j}$, for the j:th simulation at time t_0 will then be

$$C_{0,j} = max(S_{N_i} - K, 0) \tag{129}$$

if the barrier is not crossed and

$$C_{0,j} = 0 \tag{130}$$

otherwise, i.e. if $S_{i,j} \leq B$. That goes for a Down-and-out call option.

When we have simulated the price M times we get the option price as the average of all the outcomes:

$$\hat{C}_0 = \frac{1}{M}\sum_{j=1}^{M} C_{0,j}. \tag{131}$$

How is the asset price path simulated then? Because we have divided the path into N_i parts we want to simulate a jump at each time step as

$$S_{i+1,j} = S_{i,j}e^{\nu+\sigma\epsilon}. \tag{132}$$

Here, ϵ is a sample from a standard normal distribution and ν is calculated as,

$$\nu = (r - q - \frac{1}{2}\sigma^2)\Delta t. \tag{133}$$

When all simulations are made we get the option price from equation (131).

Figure 17 shows 100 simulated paths with data for the standard barrier option contract. On the vertical axis we see the asset price and on the horizontal axis is the time presented in years. We see that if the barrier is crossed in a simulation, that simulation ends and a horizontal line is drawn in the graph until maturity.

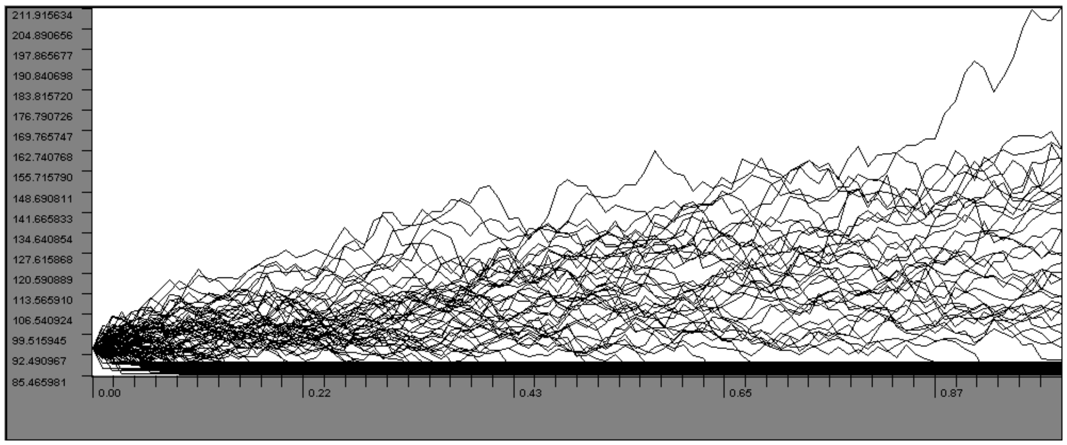

Figure 17: 100 simulated paths. On the vertical axis we see the asset price and on the horizontal axis is the time presented in years. We see that if the barrier is crossed in a simulation, that simulation ends and a horizontal line is drawn in the graph until maturity.

In figure 18 we can see the slow convergence of this model. On the horizontal axis is the calculation time presented in milliseconds while the option value is presented on the vertical axis. As you see, even a calculation that takes over 400 seconds doesn't give a good result.

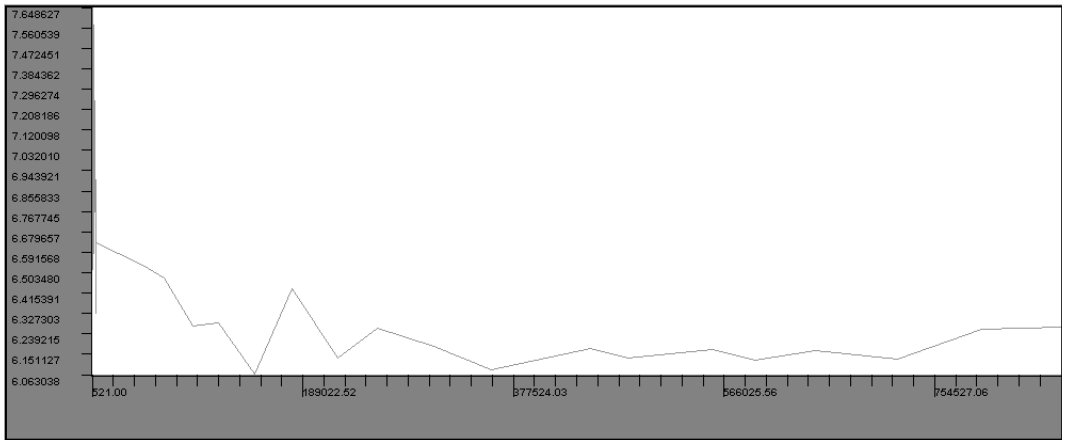

Figure 18: Monte Carlo Simulation of value for the standard barrier option contract. Option prices on the vertical axis for different calculation times on the horizontal axis.

In the comparison in the coming section this model will be neglected because of its very slow convergence.

9 Comparison

We have earlier in each section that treats different models done comparisons to find the best model of each type. We have come to conclusion that we have one binomial model where we try to find the best number of time steps, N_i, to eliminate the oscillating behaviour, section 5. Furthermore, the trinomial model with a stretch parameter, as presented in section 6, will also be considered in this evaluation. The MonteCarlo Simulation will we neglect because of the very slow convergence, section 8. Finally, the Implicit finite difference model using Cranc-Nicolson will be shown to be the best model for pricing single barrier options, see section 7.

We will see both graphs and tables with data that we compare to and from which we make our decisions.

9.1 Evaluation of the best suited numerical model for pricing the standard contract

We will start the evaluation by comparing the binomial tree model with the trinomial tree model. We are not only going to look at prices retrieved from a fix calculation time but also in which way the models converge. We start with looking at a graphical representation of the results in figure 19, here we see that the models converge almost equally fast but that the trinomial model converge in a less oscillating way than the binomial tree. An other disadvantage for the binomial tree model is that it is not possible to choose whatever number of time discretizations. That is, if we want to make a calculation at a certain number of time steps the model will choose a time step so that the nearest higher best point is hit.

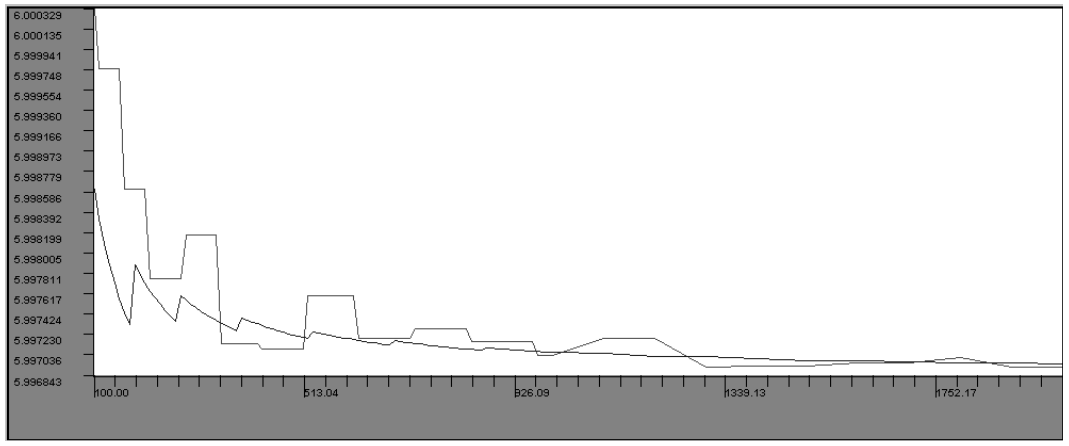

Figure 19: Comparison between the trinomial tree model and the binomial tree model. Option prices is displayed on the vertical axis for different power-rate on the horizontal axis.

From figure 19 we come to conclusion that the trinomial tree model is the best of those.

Now we are going to compare the trinomial tree model with the finite difference model. Here we choose to put the real calculation time on the x-axis. The pattern is not as easy to see as before, that is because the calculation time is varying from time to time, i.e. making two calculations with the same settings can result in different calculation times measured by the computer. Here we make calculations with a small difference in the power-rate, which is why the pattern is a bit diffuse.
Figure 20 shows the option price with respect to calculation time for the trinomial tree model

and the finite difference model.

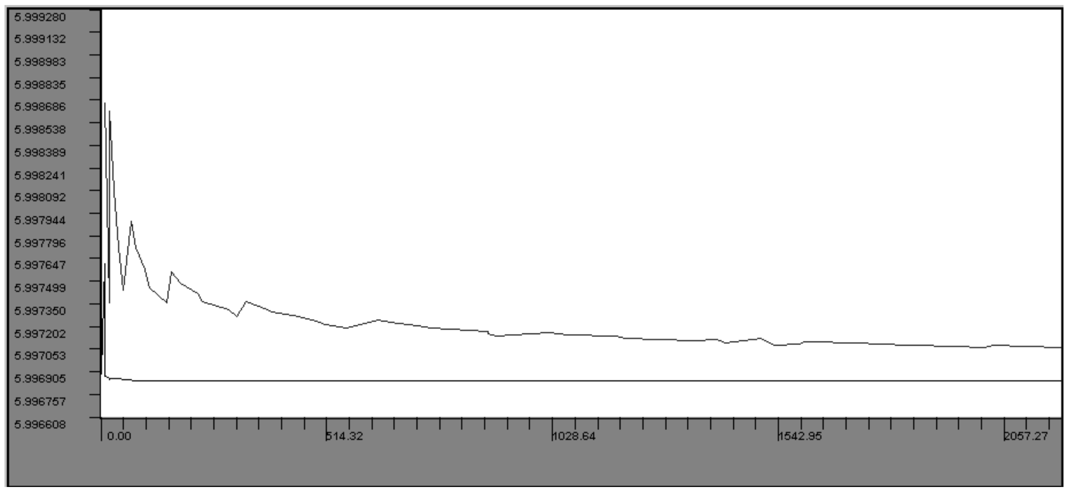

Figure 20: Comparison between the trinomial tree model and the finite difference model using Cranc-Nicolson. Option prices is displayed on the vertical axis for different calculation time on the horizontal axis.

Looking at the two graphs, we see that the finite difference model converges fast compared to the trinomial tree model. Furthermore, the finite difference model does not oscillate much, the visible oscillating behaviour is gone after 50 ms calculation time. We are finally going to see data of the different models in table 3. From this we can draw no further conclusions then above, but we can again appoint that the finite difference model have converged to the 6:th decimal already after 1000 ms. For pricing a standard contract, described in section 3.5, with current asset price, S, equal 95 we see that the finite difference model is the fastest and most suitable model.

We have seen that the finite difference model using Cranc-Nicolson approximations to the derivatives is, without doubt, the fastest model for pricing the standard barrier options contract, section 3.5. But what will the result be if we are far from, or very close to the barrier? Is the finite difference model equally fast in these cases? Those are questions that we are going to answer next.

9.2 Evaluation of the best suited numerical model for pricing contracts with asset price very close to the barrier

Here we are going to see what happens when the current asset price gets very close to the barrier. We have already seen that the finite difference model is very fast when we are close to the barrier, compared to other models this model is extraordinary and gives correct results within a fraction of time used by those. But what happens if we go even closer to the barrier. In table 4 we can see the results retrieved from the different models when the barrier lies at 90.2, all other parameters are the same as for the standard contract.

The closed form is 0.2583 due to Boyle and Tian [2] , here I have presented a more exact result but I can not confirm the accuracy because I have not tested the calculation with double precision.

As you see, calculations at a very short amount of time are impossible to do. This is because the models are not valid for a very small amount of disctetizations. E.g., for the binomial model is the smallest amount of time steps 12684. For the trinomial model by Ritchken, to

Calculation time (± 5ms)	Down and Out Call price Binomial tree	Down and Out Call price Trinomial tree	Down and Out Call price Finite Difference
10	6.000329 (80,534,534)	5.998663 (60,120,120)	5.996892 (115,53,107)
20		5.998614 (100,200,200)	5.996857 (200,88,157)
30	5.999755 (150,769,769)	5.998092 (120,240,240)	5.996858 (230,99,199)
40		5.994929 (49,530,53)	5.996857 (255,107,214)
50		5.997571 (150,300,300)	5.996854 (280,118,237)
100	5.997756 (210,1368,1368)	5.997575 (220,440,440)	5.996847 (445,179,359)
200	5.998179 (340,1731,1731)	5.997450 (310,620,620)	5.996844 (600,241,482)
500		5.997224 (490,980,980)	5.996843 (900,355,711)
1000	5.998179 (740,1731,1731)	5.997168 (690,1380,1380)	5.996842 (1300,509, 1018)
2000	5.997028 (1050,5473,5473)	5.997070 (960,1920,1920)	5.996842 (1800,700,1400)
4000	5.996924 (1500,7718,7718)	5.996999 (1400,2800,2800)	5.996842 (2600,1006,2013)
8000	5.996892 (2100,11310,11310)	5.996953 (1925,3850,3850)	
16000	5.996898 (3000,11586,15586)	5.996911 (2750,5500,5500)	
32000	5.996907 (4200,21893,21893)	5.996879 (3875,7750,7750)	
64000	5.996886 (5600,29269,29269)	5.996872 (5500,11000,11000)	
128000	5.996855 (8000,41392,41392)	5.996870 (8000,16000,16000)	
256000	5.996891 (11000,55609,55609)	5.996863 (10700,21400,21400)	

Table 3: Comparison between the binomial tree, trinomial tree and the finite Difference models with respect to calculation time for the standard option contract

Calculation time (± 5ms)	Down and Out Call price Binomial tree	Down and Out Call price Trinomial tree	Down and Out Call price Cranc-Nicolson
731			0.2582957 (1,564,1128)
1000			
2000			
4000			
8000			
11667			
34660		0.2582972	
64000	0.2583017 (12684)		
84912		0.2582966	
101476		0.2582963	
229360	0.2582972 (50737)		
1304336	0.2582964 (114159)		
1528728			0.2582957 (1000,19365,38731)

Table 4: Comparison between the binomial tree, trinomial tree and the finite difference models with respect to calculation time for the standard barrier contract with current asset price at 90.2.

obtain non-negative probabilities, is the minimum number of time steps needed 12685 for this contract, [2]. The finite difference model has not exactly the same problem, for this model we have a criteria that the number of discretizations between the barrier and the current asset price must be at least one. This makes the number of discretizations between the barrier and the uppermost asset price level very big. For this contract the number of discretizations in price will be at least 1128 and because we have the condition that the number of time steps shall be the half of that it is set to 564. If we allow two discretizations between the barrier and the current asset price there will be twice as many steps in the asset price etc.

In figure 21 you can see the results presented, one graph for each model. We can clearly

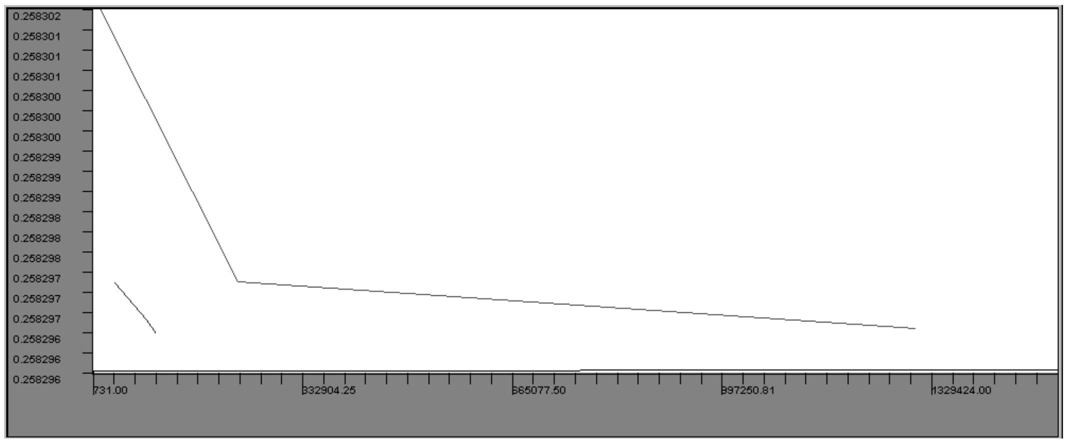

Figure 21: Comparison between the Binomial Tree model, Trinomial tree model and the Finite Difference model using Cranc-Nicolson.

see that the finite difference model is the fastest, though, it isn't as fast as for the standard contract tested above in section 9.1. The closer the barrier the asset price comes, the slower will the model become.

9.3 Evaluation of the best suited numerical model for pricing contracts with asset price far away from the barrier

The finite difference model has been shown to be the best suited model for pricing the standard contract and contracts where the asset price lie very close to the barrier. Here, we will vary the asset price to see if the model is equally accurate for a barrier option contract with the asset price far from the barrier. The asset price that will be evaluated is 200.

In table 5 as well as in figure 22 we can see that the trinomial model has a very slow convergence when the asset price is far from the barrier. Both the binomial model and the finite difference model show much better convergence. Looking at the table, it seems like the binomial model and the finite difference model converges equally fast. After 4000 ms booth the models have converged to the 6:th decimal but the behaviour of the binomial tree model is still very oscillating which does not show in the table. Figure 23 shows us a magnified picture of the two models behaviour, we now see the great difference in the way of convergence.

The finite different model using Cranc-Nicolson has then been shown to be the fastest and most suitable model for pricing all kinds of single barrier options.

Calculation time (± 5ms)	Down and Out Call price Binomial tree	Down and Out Call price Trinomial tree	Down and Out Call price Cranc-Nicolson
10	109.522629 (85,426,426)	109.517982 (90,180,180)	109.522873 (900,67,134)
20	109.522598 (100,508,508)	109.518791 (110,220,220)	109.522797 (1100,8,163)
30	109.522629 (150,759,759)	109.519135 (120,240,240)	109.522781 (1200,89,178)
40	109.522614 (180,903,903)	109.519600 (140,280,280)	109.522736 (1500,110,221)
50	109.522606 (190,960,960)	109.519821 (150,300,300)	109.522728 (2600,118,236)
100	109.522644 (260,1318,1318)	109.520729 (220,440,440)	109.522682 (2400,176,352)
200	109.522636 (350,1760,1760)	109.521332 (320,440,440)	109.522667 (3600,263,526)
500	109.522636 (500,2509,2509)	109.521797 (500,1000,1000)	109.522659 (4800,350,700)
1000	109.522652 (800,4039,4039)	109.522049 (700,1400,1400)	109.522659 (7000,509, 101
2000	109.522652 (1100,5505,5505)	109.522202 (950,1900,1900)	109.522652 (10000,727,145
4000	109.522652 (1600,8017,8017)	109.522346 (1400,2800,2800)	109.522652 (14000,1017,20
8000	109.522652 (2200,11000,11000)	109.522438 (2000,4000,4000)	
16000		109.522499 (2800,5600,5600)	
32000		109.522545 (400,8000,8000)	
64000		109.522575 (5300,10600,10600)	
128000		109.522598 (7800,15800,15800)	
256000		109.522614 (11000,22000,22000)	

Table 5: Comparison between the binomial tree, trinomial tree and the finite difference models with respect to calculation time for the standard barrier conteract with current asset price at 200.

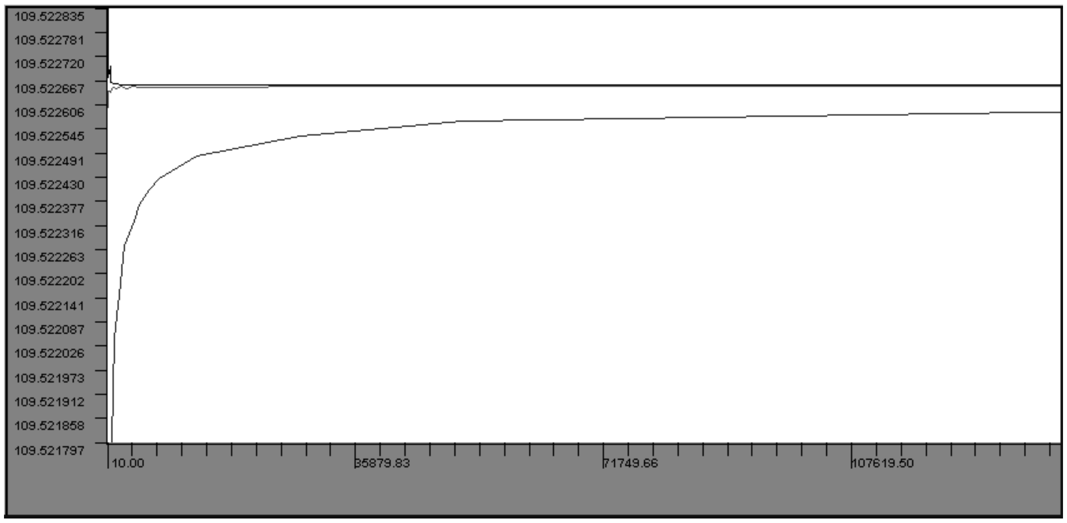

Figure 22: Comparison between the binomial tree model, trinomial tree model and the finite difference model using Cranc-Nicolson.

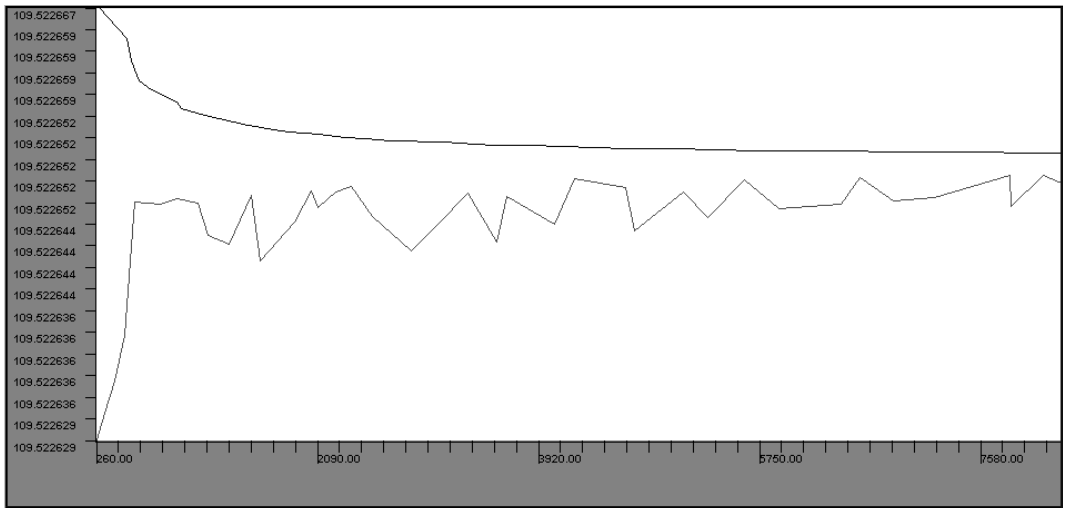

Figure 23: Comparison between the binomial tree model and the finite difference model, shows that the finite difference model converge more stict han the binomial tree model.

10 Computing Hedge Sensitivities

Hedge sensitivities is often used by traders to understand the behaviour of the contract they possess. How will the contingent claim price change when the price of the underlying asset changes? How much will the contingent claim price rise if the underlying asset rise one unit? The δ gives us the answer to these questions. There are several similar ratios that are interesting for the trader to know. I have gathered some of them together here below.

Some of the ratios can be taken from one single calculation, to do that some data need to be saved during the calculation. In each section treating different models the way of calculating these ratios from-out the tree or the grid is described. For those ratios that can not be computed directly from the tree or grid one more calculation has to be made. How this is done and how the different hedge sensitivities is described can be read below.

10.1 Delta,

Delta is the rate of change of an option's value with respect to the underlying stock price. Roughly, if we know the delta, then we can predict how much the option's value will change when the underlying's price changes by a certain amount. This parameter is computed directly from the tree for the binomial and trinomial tree models and from the grid for the Finite Difference model, for the MonteCarlo simulation model an extra calculation ought to be done.

In continuous time this parameter is in fact the partial derivative of the option value with respect to the asset price.

10.2 Gamma,

Gamma is the rate of change of the Delta, δ, with respect to the underlying's stock price. This parameter is taken directly from the tree for the binomial and trinomial tree models, likewise it can be calculated from the grid of the Finite Difference model but for the MonteCarlo simulation model an extra calculation ought to be done.

In continuous time this parameter is in fact the second partial derivative of the option value with respect to the asset price.

10.3 Omega,

Omega is the rate of change of the Gamma with respect to the underlying's stock price. This parameter is taken directly from the tree for the binomial and trinomial tree models, for the MonteCarlo simulation model an extra calculation ought to be done.

In continuous time this parameter is in fact the third partial derivative of the option value with respect to the asset price.

10.4 Theta,

Theta is the rate of change of the value of an option with respect to time. For example, it tells us how the option value is going to change if the time to maturity change. Theta can be calculated from-out the tree for the binomial and trinomial model or from-out the grid for the Finite Difference model. for the MonteCarlo simulation model an extra calculation ought to be done.

In continuous time this parameter is in fact the partial derivative of the option value with respect to the time, t.

10.5 Vega,

Vega isthe rate of change of the value of an option with respect to volatility. That is, how will the option value change if the market becomes more volatile? A high v indicates that an option's price is very sensitive to small changes in the volatility of the underlying asset.

The v has to be calculated with an extra calculation. To do this we first save the old option value, C_{old}, and the old volatility value, σ_{old}. After this we change the volatility with one percent, $\sigma_{new} = 1.01\sigma_{old}$, then we calculate a new option value, C_{new}, with this new parameter. Finally we compute the v as the rate of change as,

$$v = \frac{C_{old} - C_{new}}{\sigma_{old} - \sigma_{new}} \tag{134}$$

In continuous time this parameter is in fact the partial derivative of the option value with respect to the volatility.

10.6 Rho,

Rho is the rate of change of the value of an option with respect to the risk-free interest rate. This hedge parameter tells us how sensitive the option value is to changes in the interest rate, high ρ indicates high sensitivity.

As for the Vega, the ρ can not be computed from the tree or the grid. Here we have to make an extra calculation in the same way as for the Vega. First we save the option, C_{old} value calculated with the proper Rho, r_{old}. Then we change the interest rate one percent to $r_{new} = 1.01r_{old}$ and calculate the option value, C_{new}. Having all these values we can calculate the ρ as,

$$v = \frac{C_{old} - C_{new}}{r_{old} - r_{new}} \tag{135}$$

In continuous time this parameter is in fact the partial derivative of the option value with respect to the interest rate.

11 Including Discrete Dividends

In real-life applications it is necessary to take lumpy dividends into consideration. If there is a discrete dividend payment on an asset during the option's life the value will dramatically differ from the value for the same option with no dividend payment, even if the dividend payment is small. I will present a way of dealing with lumpy dividend payments so that no changes have to be done in the tree for binomial models or in the grid for Finite Difference and Trinomial models.

What we are going to do is adjusting the asset prices in the tree/grid with the dividend payments. The asset prices will be lowered with the sum of all dividend payments during the option's life discounted to current time with the interest rate. To do this we first found a vector, D, with elements from 0 to N_i, each element representing a time steps. At each time step we lock if there are any dividend payments, if there are we put them in the vector at its proper position. E.g., if there is a payment at time step i, we'll put the value of the payment at position i in the vector D. The dividend payment time, $t_{dividend}$, does seldom hit a time step, $i\Delta t$, exactly so we have to do some kind of approximation to put the dividend payment date in the vector D. That is, a dividend payment, $V_{dividend}$, will belong to the closest time step, i, and we'll put the discounted dividend value in the vector at that position,

$$D_i = e^{r(i\Delta t - t_{dividend})} V_{dividend}, \qquad \Delta t(i-0.5) < t_{dividend} \leq \Delta t(i+0.5) \tag{136}$$

Now we compute the discounted cumulative value of all the future dividend payments in a new vector D_{sum} also with elements from 0 to N_i. Each element, i, in the vector is calculated as below:

$$D_{sum}{}^{i} = \sum_{k=i}^{N_i} e^{-rk\Delta t} D_k \tag{137}$$

We can finally calculate the asset price spread at maturity date for the binomial method and for the whole grid for the Trinomial and the Finite Difference model. We'll do this in the same way as before, described for each model, by computing the lowest asset price in the spread, S_{low}, and subtract the value $D_{sum}{}^{0}$ from it. We now have a new lowest asset price

$$S^{*}_{low} = S_{low} - D_{sum}{}^{0} \tag{138}$$

from which we compute the spread in a normal way. If the option has the European style we don't have to change anything else. The option's price will fall out as before on the current time. But if we're dealing with American style options we have to adjust the options price at the exercise check. We'll then add the discounted cumulative dividend value at that time step in the following way:

$$C_{i,j} = max(C_{i,j}, S_{i,j} + D_{sum}{}^{i} - K) \tag{139}$$

This way of dealing with lumpy dividends is very easy to understand as you see, in fact it is also very easy to implement in the calculation process. There are no limitations in how many dividends that can be added to this model but the option price converges much slower if many dividends is present. That is important to remember and it's better informing the traders about it. It is also possible to make an adaptive system that chooses the power-rate necessary.

12 Conclutions and Results

We have seen that the implicit finite difference model using Cranc-Nicolson approximation of Black and Scholes partial differential equation is the fastest model for pricing single barrier options, section 7. This model is fast, robust, easy to understand and easy to implement. Robust because the stability is independent of the number of discretizations. Futhermore, even if the current asset price lies close to the barrier we get a correct result in a short amount of calculation time which other models presented in this paper does not manage to do.

12.1 Difficulties in the evaluation process

There have been some difficulties in the evaluation process. When getting a model from a working paper published in some journal it is often difficult to reproduce the results. Often, there is no description how to implement the model and there is often a lack of evident theoretically aspects too. Therefore, the results retrieved from my implementations is not always exactly the same as those presented in the paper. In this particular case we try to compare the models by calculation time. Firstly, almost no other makes their comparisons with respect to calculation time, therefore, I've been forced to implement all different models to make the comparison. Otherwise I could have taken the results directly from the published papers, though, it would probably be almost impossible to use the values because the computer configuration and performance would have to be exactly the same for me and for the one used to get the results in the paper.

12.2 Notes about the Finite Difference model

The Finite Difference Model is the best suited model for pricing single barrier options, though, when the current asset price is close to the barrier even this model becomes slow. It is possible to make an interpolation in that case to speed up the calculation but in real life, options with asset prices closer to the barrier than 0.05 when the current asset price is about 100 is really not interesting to calculate because that the asset is traded with at maximum one decimal.

I'm going to present the results for the standard contract, section 3.5, in two different ways so that the results presented in this paper can be used in future comparisons. First, one table with different values with respect to the number of discretizations, second, one table with values with respect to calculation time.

To get an idea of the behaviour of the option price close to the barrier we are going to look at the delta, see section 10, for different values of the current asset price. In figure 24 below the horizontal axis shows the current asset price and the verical axis shows the δ value. We can see from the figure that the option value gets more and more sensitive for changes in the current asset price the closer it is to the barrier, this is a typical behaviour for the barrier option.

Calculation time (± 5ms)	Down and Out Call price Finite Difference
10	5.996892 (115,53,107)
20	5.996857 (200,88,157)
30	5.996858 (230,99,199)
40	5.996857 (255,107,214)
50	5.996854 (280,118,237)
100	5.996847 (445,179,359)
200	5.996844 (600,241,482)
500	5.996843 (900,355,711)
1000	5.996842 (1300,509, 1018)
2000	5.996842 (1800,700,1400)
4000	5.996842 (2600,1006,2013)

Table 6: The option value for the standard barrier option contract, see section 3.5 is presented for different calculation time, the asset price is here $N_i = \frac{N_j}{2}$

Discretizations in asset price	Down and Out Call price Finite Difference
49	5.996010
98	5.996705
196	5.996814
393	5.996831
803	5.996840
1606	5.996840
3213	5.996841
6403	5.996842
2016	039486

Table 7: The option value for the standard barrier option contract, see section 3.5 is presented for different number of discretizations of the asset price, here $N_i = \frac{N_j}{2}$

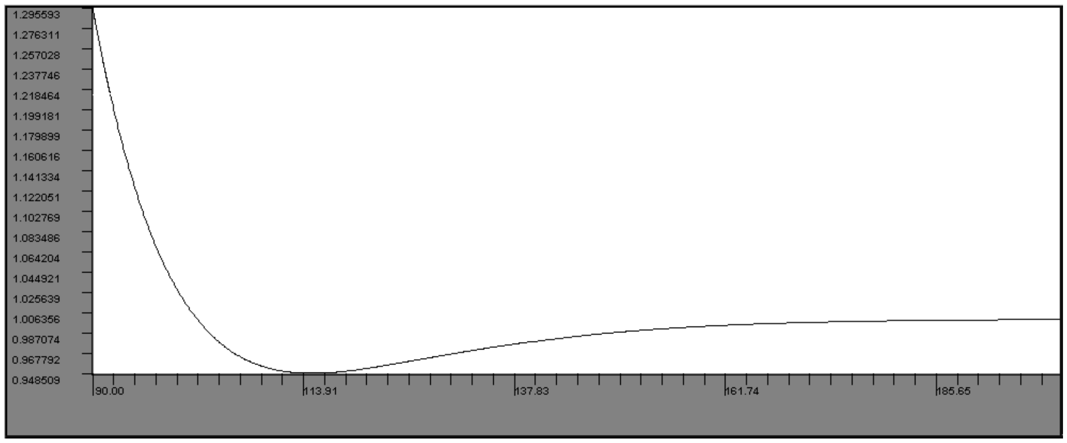

Figure 24: Graph representing the delta value with respect to current asset price, all other data as for the standard barrier contract

13 Further Development

The standard barrier option contract used in this paper, see section 3.5, is just one of many different types of single barrier option contracts. Adjusting the finite difference model to these contracts is a development process that has to be done. Furthermore, make the option pricing functions more efficient is evident. For example, the function for solving tri-diagonal equation systems may be improved. One can also use a performance monitoring tool that computes the time spent in different partitions of the calculation functions to see where the greatest effort should be used to maximize the calculation speed.

All models treated in this paper are used for pricing single barrier options where the barrier is constant. This is the most common type of barrier option contract but there exist many other types such as those with non-constant barriers or those with two barriers or combinations of them.

First, I would like to recommend further developing of the models for double barrier option pricing. That is because I think these will be traded more in the future. We have already looked at a model which is originally constructed for pricing double barrier options, [2]. I think the Finite Difference Model using Cranc-Nicolson for approximating the derivatives with the implementation and interpolation suggested in that paper will give the best results. Other paper I've found where information and pricing models for double barrier options can be found are Intermark's working papers [18] and [19].

The barrier option contract with non-constant barrier is treated in several publications. Peter Ritchken [1] for example is evaluating barrier options with exponential boundaries.

I don't think trading with non-constant-barrier option contracts will be common in a near future because of its complexity, it is not easy to intuitively "see" how the contract is going to develop when the price of the underlying asset is changing. Anyway, to be prepared for the future is not a waste of time. So, I recommend finding a fast model for the non-constant barrier contract.

14 Appendix, About the evaluation tool built in Java

The evaluation tool is built for the purpose to price different types of derivatives. The program is modelled in a way such that it is easy to extend with new derivatives without having much programming experience. The graphical interface of the program is copied from the calculation module in Extra. Extra is a trader program developed by ABN-Amro Software AB. At the moment only standard options can be valued with Extra. Therefore I have extended the interface so that other types of derivatives can be calculated in the same module.

Mainly, the program consist in three different parts; the inputpanel where we write the contract specification, the outputfield where we get all the output data such as option values, finally we have the graphwriter that makes it possible to have a graphical representation of the models behaviour.

On the inputpanel we specify the data of the option contract that is going to be valued, typical data here is the underlying asset's current price, time to maturity and volatility etc. There are some fields here that is common for all different types of derivative contracts, each type of derivative has also its own inputfields, for the barrier option this is for example which type of barrier option that is being valued, i.e. down-and-out or down-and-in etc. Other inputs as interest-rate and dividends are added using dialog boxes.

In the outputfield we get the results from the calculations, they are presented in a list with one row for each calculation. In addition to the option's value we also get values of hedge sensitivities. Furthermore, the program measures the calculation time and displays this in the outputfield, it is measured and displayed in milliseconds. This is a limitation of the Java language; we can't get a more exact measurement as for example in nanoseconds. A more accurate calculation time can be interesting when dealing with very fast models as the Binomial model for pricing standard options. For the barrier option on the other hand this is less important. In the outputfield is also some of the input data displayed so that we know which contract that got a specific value.

The graphwriter is a very useful tool where we can test the options behaviour. That is, we display all option values from the output field in a diagram with the option value on the vertical axis and the power-rate or calculation time on the horizontal axis. The most interesting to see is how the value of the derivative converges with higher calculation time and then especially compared to other models. We can also place the power-rate on the horizontal axis. When doing this we get a more exact picture of the model because the power-rate doesn't give an alternating error as with the calculation time, i.e. a measurement of the calculation time is not exact because of the load on the computer. Furthermore, to be able to compare two different models using the power-rate, we need to weight the number of discretizations so that the models have the same calculation time for the same power-rate.

We can also display the behaviour of the hedge sensitivities for different levels of the current asset price for a certain contract.

14.1 How to add new features to the program.

As I mentioned in the introduction to this paper, the program is built in Java. This makes the program very easy to extend because it's easy to change old classes or add new classes to the program without need to recompile the whole program.

The structure of the program makes it easy to find and change the classes. In figure 25 we see where different parts of the program is placed. In *Source* is all source-code files placed, to make changes to the program, copy this folder and rebuild the program using for

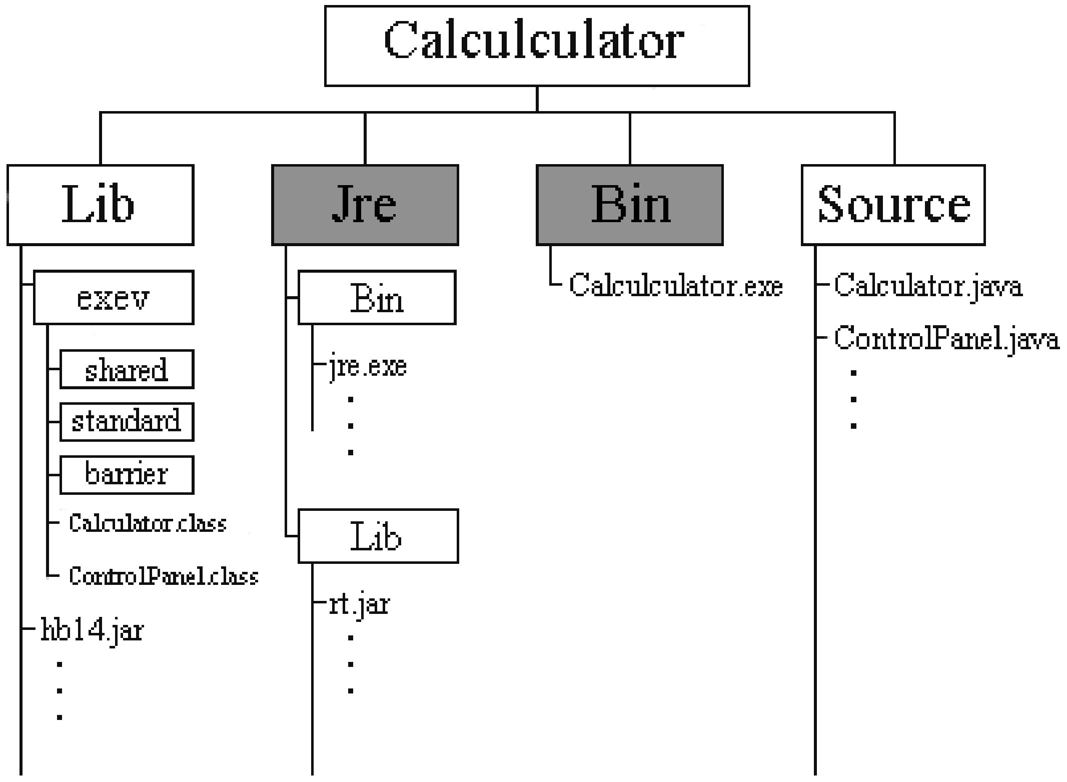

Figure 25: The program structure of the Evaluation Tool built in Java. The folder *Calculator* includes all parts of the program, i.e. the program specific class files placed in *Lib* and the Java Runtime environment placed in *Jre*. A launcher program, Calculator.exe, is placed in *Bin*. All source-code for the program is placed in *Source*. The two marked folders in the middle is platform dependent and has to be changed if the application shall run under an-other operating system.

example JDK by Sun Microsystems Inc. When doing this, make sure all files in the folder *Calculator\Lib\*, excluding exev, is linked to the project.

Java is an OS independent programming language, but to make the program easy to use I have made it semi-dependent. That is, to run the program on any other platform than the Windows 95/98/NT some changes has to be done to the program. The launcher, here called Calculator.exe, has to be rewritten for the new platform. What this program does is to launch the Jre.exe and feed this with the location of all the class files and libraries. Furthermore, the Java Runtime environment, Jre, has to be changed for this application if it shall run under an-other operating system.

Then, how do we add new features as for example a new derivative? I'll take an example; we want to be able to evaluate the best-suited numerical model for pricing Asian Options using this tool. We start with copying all .java files in the folder *Source* to a new project in a development environment, e.g. JDK. What we then have to do is making a new Class called AsianOption, this class inherit from the base class Option that on the other hand inherit its functionality from the class Derivative. We override the calculation functions that we want to use as for example calculateBT() to make a calculation with the binomial tree model. We can now make the inputpanel for this specific instrument; as I mentioned earlier we have a base class with all common inputs from which we inherit to the new panel called for example AsianPanel. To this panel we add input fields specific to this instrument. Finally, to add this panel to the program we have to make a change in one file. This file is called *Control-Panel.java.*

When all the program is written, compiled and debugged we can add the new .class files to the existing program in its proper position. Don't forget to add the new source files in the *Source* folder so that the program can be rebuilt next time someone want to extend the functionality of the program.

14.2 The developers environment.

I have built a developer environment with automatic generation of code for adding new derivatives to the program. Here, you just open a dialog box from the toolbar and type the name of the new derivative. You can on the same time choose the number of additional input fields you want, then you choose names for those variables. The application will then automatically generate files with the code for the new derivative. It constructs a new panel associated with it and adds it to the evaluation tool. Finally it compiles the newly created and changed files and opens the file with the calculation functions for editing.

To add functionality to the model you type code inside the pre-generated functions. When done, just press "save" followed by "compile".

A more detailed description of how to add new derivatives to the application can be found in "The Derivative Evaluation Tool, Developers Guide" which belongs to ABN-Amro Software AB and can not be distributed with this paper.

15 Bibliography

References

[1] Peter Ritchken
"On Pricing Barrier Options."
The Journal of Derivatives, Volume 3, Number 2, (Winter 1995).

[2] Phelim P. Boyle and Yisong (Sam) Tian
"An explicit finite difference approach to the pricing of barrier options"
Applied Mathematical Finance 5, 17-43 (1998)

[3] G.Ioffe and M. Ioffe
"Application of Finite Difference Method for Pricing Barrier Options."
Intermark's Working Papers (Last modified 10 June 1998) http://www.intermarkit.com

[4] Boyle, P., and S.H. Lau.
"Bumping up against the Barrier with the Binomial Method"
Journal of Derivatives, 1,4 (1994), pp 6-14

[5] Black, F. and M. Scholes.
"The Pricing of Options and Corporate Liabilities."
Journal of Political Economy, Vol. 81, No. 3 (1973) pp. 673-659.

[6] Schwartz E. S.
"The Valuation of Warrants: Implementing a New Approach."
Journal of Financial Economics, 4,1 (1977) pp. 79-93

[7] Emanuel Derman and Iraj Mani and Neil Chriss
Implied Trinomial Trees of the Volatility Smile

[8] Yuh-Bauh Lyun
"Very Fast Algorithms for Barrier Option Pricing and the Ballot Problem"
The Journal Of Derivatives, Volume 5, Number 3, Spring 1998

[9] N.K. Chidambaran and stephen Figlewski
"Streamlining Monte Carlo Simulation with the Quasi-Analytic Method; Analysis of a path-dependent Option Strategy"
Journal of Derivatives, winter 1995

[10] Mark Broadie and Paul Glasserman and Gautam Jain
"Enhanced monte Carlo Estimates for American ooption Prices"
The Journal of Derivatives Volume 5, Number 1, Fall 1997

[11] Silvio Galanti and Alan JUNG
"Low-Discrepanay Sequences; Monte Carlo Simulation of Option Prices"
Journal of Derivatives, Volume 5, Number 1, Fall 1997

[12] "A More Accurate Finite Difference Approximation for the valuation of options"
Journal of Financeial and Quantitative Analysis, December 1982, pp 697-703

[13] Black, F. and M. Scholes.
"Finite Difference Methods and Jump Processes Arising in the Pricing of Contingent Claims: A Synthesis."
Journal of Financial and Quantitative Analysis, September 1978, pp. 461-474

[14] Hull J. and White A.
"Valuing Derivative Securities Using the Explicit Finite Difference Method."
Journal of Financial and Quantitative Analysis, March 1990, pp. 237-252

[15] Les Clewlow and Chris Strickland
"Implementing Derivatives Models"
ISBN 0-471-96651-7
John Wiley and Sons, 1998

[16] John C. Hull
"Options, Futures, and other Derivatives"
ISBN 0-13-264367-7
Prentice-Hall, Inc, 1997

[17] Neil A. Chriss
"Black-Scholes and Beyond; Option Pricing Models"
ISBN 0-7863-1025-1
McGraw-Hill, 1997

[18] G. Ioffe
"Differences in theoretical and actual prices of double Knock-Out and binary range FX options"
Intermark's Working Papers (Last modified 10 June 1998) http://www.intermarkit.com

[19] Ioffe M., Ioffe G., Krell D.
"Pricing formulas for Double Knock Out and Binary Range Options"
Intermark's Working Papers (Last modified 10 June 1998) http://www.intermarkit.com