Numerical solution for derivative models using finite difference methods and how this can be used with Monte Carlo simulation

Author: Marcus Hallabro
Supervisor: Magnus Wiktorsson
Examiner: Andreas Jakobsson

Abstract

Derivative models often come in the form of stochastic differential equations. From these equations a partial differential equation (PDE) can be derived. By discretizing the PDE the numerical solution is obtained on a form where the value of the derivative can be seen as a probabilistic weighting of future values. These probabilities can be used to simulate trajectories of the underlying assets. This connection between the finite difference scheme and the simulation is rather unique for this pricing method and can be very useful. The probability weights can be forced to have a perfect probability interpretation in some cases, meaning they are positive and less than one, but in other cases we will end up with negative weights meaning we somehow have to simulate using negative probabilities. This paper presents how to price and simulate options with these methods in a few different situations and how to solve some of the problems that may come up.

Keywords: Finite Difference Method, Option Pricing, Feynman-Kac Representation, Monte Carlo Simulation, Negative Probabilities.

Acknowledgements

I want to thank everyone that have helped me during this thesis. A special thanks to my supervisor, Professor Magnus Wiktorsson, for introducing me to the subject and for guiding me through the project with phenomenal expertise.

Contents

Chapter 1

Introduction

A financial contract with a value based on an underlying asset is called a derivative. Being able to determine a fair price for a derivative is really useful in finance but it is not very simple. In many situations it is also of interest to simulate the underlying assets. Finding a connection between these subjects is therefore of great interest.

The most common derivative models are in the form of stochastic differential equations. From these equations we can derive a partial differential equation that describes the value of the derivative as a function of the underlying asset. By discretizing this equation the numerical solution is obtained on a form where the value of the derivative can be interpreted as a probabilistic weighting of future values. These probabilities can be used to simulate trajectories of the underlying assets. In some specific situations these weights will have a perfect probability interpretation but in some other cases the weights can be negative meaning we have to somehow simulate with negative probabilities. We will explain exactly how this can be done in section 2.10.2.

The goal of this paper is to go through all the relevant theory and then show some examples of how the theory can be used in different situations.

Chapter 2

Theory and Method

2.1 Financial framework

In this paper we assume a standard financial market where people can trade derivatives. We also assume there is no arbitrage opportunities, meaning there is no way of making an immediate profit without any risk of loosing money. Basically this means that all prices are consistent with the other contracts in the market.

2.2 Derivatives

A derivative is a financial contract that specifies an event on the financial market ([Åberg, 2018, p. 1]). This event is specified in terms of another financial assets price, S. In this paper the only relevant derivative contract is the option. Therefore this section starts with some basic option theory, with a description of the relevant options for this paper.

2.2.1 Options

The notation of the following options are based on [Åberg, 2018, p. 59-72]:

Vanilla Options

The most basic and common options are the European call and put options. An option being European means it can only be exercised at a predefined date (and not earlier). We will later introduce American options that do not have this restriction. The holder of a European call option has the option to buy the underlying asset, S, to a predefined price K at predefined maturity time T. The payoff function, $\mathcal{P}$, for this option is:

$$\mathcal{P}(S_T) = \max(S_T - K, 0).$$

The holder of a European put option instead has the option to sell the underlying asset to a predefined price, so the payoff at maturity is:

$$\mathcal{P}(S_T) = \max(K - S_T, 0).$$

Binary Double No-Touch Option

The double no-touch option is an exotic contract. It gives the holder the payout one if the underlying asset price remains within a specified range until the expiration. This price range is defined by two barrier levels. One lower barrier, L_B, and one upper barrier, H_B. If the underlying asset at any point is outside of this range the payoff will be zero. This payoff function can be written as:

$$\mathcal{P}(S_T) = 1\{L_B < \inf_{0 \leq t \leq T}(S_t), \sup_{0 \leq t \leq T}(S_t) < H_B\}.$$

American Options

The American option is a generalisation of the European option. A European option can only be exercised at the time of maturity, T, but the holder of an American option must, for every time point $t \leq T$, choose to either keep or exercise the option. If the option is exercised the holder will get the payoff immediately. For example, the payoff function of an American call option is:

$$\mathcal{P}(S) = \max(S_t - K, 0).$$

2.3 Wiener Processes

The Wiener process, W, is a stochastic process in continuous time that satisfies the following conditions (see [Åberg, 2018, p. 92]):

- $W_0 = 0$.
- W has independent increments, meaning that for $0 \leq h \leq s \leq t$ $(W_{s+u} - W_s)$ is independent of $(W_{t+h} - W_t)$.
- W has Gaussian increments, meaning that $(W_{t+h} - W_t) \in \mathcal{N}(0, h)$
- W_t is continuous in t.

The Wiener process is often used as a random noise-term in modeling because of the suitable properties. Mainly the fact that it is unbiased, i.e. $E[W_t] = 0$, and that its increments are independent. However there is one problem with the Wiener Process - it is non-differentiable. A lot of things could be said about how go on about this, but it is not very important for this paper. All we need to know is that there is a very useful formula to solve this problem.

2.4 Itô Formula

Here is the one dimensional Itô Formula (see [Åberg, 2018, p. 114]). For $f(t,x) \in \mathcal{C}^{1,2}$:

$$df(t,W_t) = \Big(\partial_t f(t,W_t) + \frac{1}{2}\partial^2_{xx} f(t,W_t)\Big)dt + \partial_x f(t,W_t)dW_t. \tag{2.1}$$

Note that this is a standard notation and should be interpreted as:

$$f(T,W_T) - f(0,0) = \int_0^T \Big(\partial_t f(t,W_t) + \frac{1}{2}\partial^2_{xx} f(t,W_t)\Big)dt + \int_0^T \partial_x f(t,W_t)dW_t.$$

Itôs Formula is the rule in stochastic calculus that corresponds to the Fundamental Theorem ($f(T) - f(0) = \int_0^T f'(x)dx$) in standard calculus.

For $f(t,x,y) \in \mathcal{C}^{1,2,2}$ we can state the two dimensional Itô Formula, writing f short for $f(t,W_t^1,W_t^2)$, as:

$$df = \Big(\partial_t f + \frac{1}{2}\partial^2_{xx} f + \frac{1}{2}\partial^2_{yy} f + \rho\partial^2_{xy} f\Big)dt + \partial_x f dW_t^1 + \partial_y f dW_t^2, \tag{2.2}$$

where W_t^1 and W_t^2 are Wiener processes with correlation ρ.

2.5 Stochastic Differential Equation Models

A very common way to model the asset price, S_t, in finance is through this type of stochastic differential equation (SDE):

$$dS_t = \mu(t,S_t)dt + \sigma(t,S_t)dW_t. \tag{2.3}$$

Here $\mu(t,S_t)$ is called the drift and is used to model deterministic trends of the asset price while $\sigma(S_t,t)$ is called the diffusion and models the unpredictability of the asset price. Note that both these functions are deterministic functions. In the most simple case they are just constants but they often depend on S_t.

2.5.1 Geometric Brownian motion

For the geometric Brownian motion (GBM) both the drift term and the diffusion term are linearly dependent on the underlying asset price:

$$dS_t = \mu S_t dt + \sigma S_t dW_t.$$

This is probably the most common SDE in finance. For instance this is the model used to derive the famous Black-Scholes Formula, together with some other assumptions. For this model to not have any arbitrage opportunities

the drift constant, μ, must be equal to the risk free rate, r, i.e. the rate of return of an investment with no risk.

$$dS_t = rS_t dt + \sigma S_t dW_t. \tag{2.4}$$

In this model we only have one equation. One way to extend it is to also model the volatility of the asset with a similar SDE and make these equations depend on each other accordingly.

2.5.2 Heston Model

In the Heston Model we still only have one asset but we have two stochastic differential equations. One for the price of the underlying asset, S_t, and one for the volatility of the underlying asset, Z_t. The Heston Model can appear in some slightly different forms but for this paper we assume the following model:

$$\begin{cases} dS_t = rS_t dt + S_t\sqrt{Z_t}dW_t^1 \\ dZ_t = \kappa(\theta - Z_t)dt + \sigma\sqrt{Z_t}dW_t^2 \end{cases}, \tag{2.5}$$

where W_t^1 and W_t^2 are Wiener processes with correlation ρ. The other constants in the equation are:

- r - the rate of return of S_t.
- θ - the long variance of S_t ($E[Z_t] \to \theta$ as $t \to \infty$).
- κ - the rate which Z_t reverts to θ.
- σ - the volatility of Z_t.

2.6 The Partial Differential Equation (PDE)

The SDE models are now defined but to be able to discretize the model it first has to be written as an partial differential equation (PDE). In other words we have to derive a PDE from the SDE models. To make this as simple as possible to understand we start with the one-dimensional model, the GBM (2.4), but later we will also derive the PDE for the Heston Model (2.5).

We introduce $V(t, S_t)$ as the value of an option, i.e. the price of the option at time t with the current underlying asset price S_t. The payoff for the option at the time of maturity, $V(T, S_T)$, is known but we also want to know the value for all possible values of t and S_t. The first step in calculating how the value of the option evolves in t and S_t is to use the Itô Formula (2.1), which gives us:

$$dV = \Big(\frac{\partial V}{\partial t} + rS_t\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 V}{\partial S^2}\Big)dt + \sigma S_t\frac{\partial V}{\partial S}dW_t.$$

The next step is to understand that for there to be no arbitrage opportunities the drift term in this equation must be equal to rV, meaning we arrive at the well known Black-Scholes equation:

$$\frac{\partial V}{\partial t} + rS_t\frac{\partial V}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 V}{\partial S^2} = rV,$$

or in a more simple notation:

$$\frac{\partial V}{\partial t} + D_s^* V = rV, \tag{2.6}$$

where $D_s^* = rS_t\frac{\partial}{\partial S_t} + \frac{1}{2}S_t^2\sigma^2\frac{\partial^2}{\partial S_t^2}$.

For the Heston Model the option value, $V(t, S_t, Z_t)$, is a function of t, S_t and Z_t. The whole derivation can be done in a very similar way. With the two dimensional Itô Formula (2.2) and some arbitrage arguments we get:

$$\frac{\partial V}{\partial t}+rS_t\frac{\partial V}{\partial S_t}+\frac{1}{2}S_t^2 Z_t\frac{\partial^2 V}{\partial S_t^2}+\rho\sigma S_t Z_t\frac{\partial V^2}{\partial S_t \partial Z_t}+\kappa(\theta-Z_t)\frac{\partial V}{\partial Z_t}+\frac{1}{2}\sigma^2 Z_t\frac{\partial^2 V}{\partial Z_t^2} = rV.$$

Using the notation introduced above this can be written as:

$$\frac{\partial V}{\partial t} + D_s^* V + D_z^* V + \rho\sigma S_t Z_t\frac{\partial V^2}{\partial S_t \partial Z_t} = rV, \tag{2.7}$$

where $D_s^* = rS_t\frac{\partial}{\partial S_t} + \frac{1}{2}S_t^2 Z_t\frac{\partial^2}{\partial S_t^2}$ and $D_z^* = \kappa(\theta - Z_t)\frac{\partial}{\partial Z_t} + \frac{1}{2}\sigma^2 Z_t\frac{\partial^2}{\partial Z_t^2}$.
Note that the same D_s^* is used for both the GBM (2.6) and the Heston Model (2.7). The only difference is that the variance, σ^2, is constant in the GBM case but for the Heston Model this same variance is written as Z_t and is not constant.

The next step is to discretize this equation but first some useful theory is introduced.

2.7 Feynman-Kac Representation

The Feynman-Kac formula makes the connection between a lot of different sections in this paper a lot clearer and is thus very useful ([Åberg, 2018, p. 125]). Suppose we have a PDE on the following form. For any $f(t,x) \in \mathcal{C}^{1,2}$:

$$\begin{cases} \frac{\partial f(t,x)}{\partial t} + \mu(t,x)\frac{\partial f(t,x)}{\partial x} + \frac{\sigma(t,x)^2}{2}\frac{\partial^2 f(t,x)}{\partial x^2} = rf(t,x) \\ f(T,x) = \mathcal{P}(x) \end{cases}. \tag{2.8}$$

The Feynman-Kac Formula states that the solution to this PDE is equal to:

$$f(t,x) = e^{r(T-t)}E[\mathcal{P}(S_T)|S_t = x],$$

where $\mathcal{P}(S_T)$ is a known payoff function and S_u (for $t \leq u \leq T$) satisfies:

$$\begin{cases} dS_u = \mu(u, S_u)du + \sigma(u, S_u)dW_u \\ S_t = x \end{cases}.$$

Note that this has the form of the SDE defined above (2.3). This representation basically says that the solution to the PDE can be seen as an expected value of the contract, which makes the connection with the upcoming Monte Carlo simulation very clear. This connection will be discussed more later.

2.8 Grid

To discretize the PDE we need to first create a grid. This is done by discretizing t, S_t and Z_t into N, M and L points. We introduce the notation $V^n_{m,l} = V(n\Delta t, S_0 + m\Delta S, Z_0 + l\Delta Z)$ where $(n\Delta t, S_0 + m\Delta S, Z_0 + l\Delta Z) \in [0, (N-1)\Delta t] \times [S_0, S_0 + (M-1)\Delta S] \times [Z_0, Z_0 + (L-1)\Delta Z]$. Here S_0 and Z_0 represents the lower end values in the grid for S_t and Z_t, whereas the time is assumed to start at $t = 0$. For the GBM case the l index is not needed so for that case we use V^n_m as notation.

2.9 Finite Difference Method

Since the payoff at the time of maturity, T, is known the goal is to solve the equation backwards in time by starting with the known payoff and taking small steps backwards, one at a time, to get the value at any desired timepoint. To make it as simple as possible to follow we first create the scheme for the GBM and later extend it to the Heston model.

2.9.1 GBM Finite Difference Method

The PDE for the GBM (2.6) has the exact form of the Feynman-Kac representation (2.8). However there are some merits to instead look at the homogeneous partial differential equation where the right term is zero, but this can be done without any problem. According to the Feynman-Kac formula the rV term in the PDE only corresponds to the solution being multiplied with the constant $e^{r(T-t)}$. To make this scheme as simple as possible we can set the right term to zero and instead take care of this constant by just multiplying it with the final solution. Another reason to work with the homogeneous equation is to make the upcoming simulation part a bit more clear to understand. When we later create matrices to simulate from we want the rows in them to sum to one. If we keep the rV term in this equation it would make the rows sum to $1 + r$ instead which would make the probability

interpretation a bit unclear. So the discretized equation we work with is:

$$\frac{\partial V_m^n}{\partial t} + rS_m^n \frac{\partial V_m^n}{\partial S} + \frac{1}{2}\sigma^2 (S_n^m)^2 \frac{\partial^2 V_m^n}{\partial S^2} = 0. \tag{2.9}$$

The discretized time derivative is approximated with the standard forward difference:

$$\frac{\partial V_m^n}{\partial t} \approx \frac{V_m^{n+1} - V_m^n}{\Delta t}.$$

For the corresponding space derivatives we instead use a central difference approximation. These can be approximated in either time point n or $n+1$. To begin with we approximate them in time point n:

$$\frac{\partial V_m^n}{\partial S} \approx \frac{V_{m+1}^n - V_{m-1}^n}{2\Delta S}, \qquad \frac{\partial^2 V_m^n}{\partial S^2} \approx \frac{V_{m+1}^n - 2V_m^n + V_{m-1}^n}{(\Delta S)^2}. \tag{2.10}$$

From this we can derive the implicit solution. After inserting these approximations in equation (2.9) we get:

$$\frac{V_m^{n+1} - V_m^n}{\Delta t} + rS_m^n \frac{V_{m+1}^n - V_{m-1}^n}{2\Delta S} + \frac{1}{2}\sigma^2 (S_m^n)^2 \frac{V_{m+1}^n - 2V_m^n + V_{m-1}^n}{(\Delta S)^2} = 0$$

$$\Updownarrow$$

$$V_m^{n+1} = V_m^n - \Delta t r S_m^n \frac{V_{m+1}^n - V_{m-1}^n}{2\Delta S} - \Delta t \frac{1}{2}\sigma^2 (S_m^n)^2 \frac{V_{m+1}^n - 2V_m^n + V_{m-1}^n}{(\Delta S)^2}.$$

This holds for every m (except for the edges where some adjustments have to be made) so we can write these equations in vector form. We introduce V^n as a column vector containing all V_m^n (for $0 \le m \le M-1$).

$$V^{n+1} = V^n - \Delta t D_{s1} V^n - \Delta t D_{s2} V^n = (I - \Delta t D_{s1} - \Delta t D_{s2}) V^n = (I - \Delta t D_s) V^n$$

$$\Updownarrow$$

$$V^n = (I - \Delta t D_s)^{-1} V^{n+1} = A_s^{imp} V^{n+1}, \tag{2.11}$$

where $D_s = D_{s1} + D_{s2}$ is an $M \times M$ matrix:

$$D_{s1} = \begin{bmatrix} 0 & 0 & 0 & & & & \\ -\alpha_1 & 0 & \alpha_1 & & & & \\ & -\alpha_2 & 0 & \alpha_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\alpha_{M-2} & 0 & \alpha_{M-2} & \\ & & & 0 & 0 & 0 & \end{bmatrix}, \alpha_m = \frac{rS_m}{2\Delta S},$$

$$D_{s2} = \begin{bmatrix} 0 & 0 & 0 & & & & \\ \beta_1 & -2\beta_1 & \beta_1 & & & & \\ & \beta_2 & -2\beta_2 & \beta_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \beta_{M-2} & -2\beta_{M-2} & \beta_{M-2} & \\ & & & 0 & 0 & 0 & \end{bmatrix}, \beta_m = \frac{\sigma^2 (S_m)^2}{2(\Delta S)^2}.$$

Here the edges of the matrices are set to match the conditions for a specific contract. In this case it corresponds to the double no touch option (if the edges of the grid are set to match the barrier levels of the option). The double no touch option is very convenient to work with but the edges can also be changed to match some other type of contract one want to evaluate. Just to show an example this is how the first and the last row of D_{s1} are defined for a European call option:

$$\alpha_0 \begin{bmatrix} -3 & 4 & -1 & 0 & \dots & 0 \end{bmatrix}, \qquad \alpha_{M-1} \begin{bmatrix} 0 & \dots & 0 & 1 & -4 & 3 \end{bmatrix}. \tag{2.12}$$

The edges are the most problematic part when evaluating contracts with this kind of schemes and the greatest evaluation errors will often appear on the edges.

If we instead evaluate the space derivatives (2.10) in time point $n+1$ instead of n one can very similarly derive the explicit scheme as:

$$V^n = (I + \Delta t D_s)V^{n+1},$$

where D_s is the same matrix as for the implicit scheme.

Both these methods can be used. However, to get the best result we can have a mix of these two solutions giving us the Crank-Nicolson method [Crank, Nicolson , 1947]:

$$V^n = \Big((I - \frac{\Delta t}{2} D_s)^{-1}(I + \frac{\Delta t}{2} D_s)\Big)V^{n+1} = A_s V^{n+1}.$$

For the rest of this section we will only consider this scheme, but the implicit and explicit schemes basically work just the same.

We can simply take one step backwards in time by just multiplying the value vector V^n with A_s. But because of the clean form this equation has we can take more than one step at a time. Taking n steps is simply done by taking the n:th power of A_s and then using this new matrix as our transition matrix. This means the value at time $t = 0$ can be written as:

$$V^0 = (A_s)^{N-1} V^{N-1},$$

where V^{N-1} is the discretized payoff at the time of maturity ($t = T$).

This is true for all the European options. However if we want to price an American option we have to take every step one at a time. That is because an American option can be exercised at any time before and at the time of maturity. So at any time we want to check if the direct payoff is greater than the value of keeping the contract and if that is the case we change the

value of the option to this payoff value. But since the time is discretized we can only check this condition it at every step (and not all possible times in between). This will make the evaluation of American options a bit biased, but the error will get smaller when we increase the fineness of the time discretization, N.

2.9.2 Heston Finite Difference Method

To create the scheme for the Heston Model a lot of steps will be similar to the scheme we just derived for the GBM but there is one problem, the cross derivative term in the Heston PDE (2.7). In the case where $\rho \neq 0$ the cross term will make the scheme impossible to write in a form this simple. There is however some ways to include this correlation in the simulation and it will be discussed later but to begin with we only look at the case where $\rho = 0$. The discretized Heston equation we will work with can thus be written as:

$$\begin{aligned}&\frac{\partial V_{m,l}^n}{\partial t} + rS_{m,l}^n \frac{\partial V_{m,l}^n}{\partial S} + \frac{(S_{m,l}^n)^2 Z_{m,l}^n}{2} \frac{\partial^2 V_{m,l}^n}{\partial S^2} \\ &+ \kappa(\theta - Z_{m,l}^n) \frac{\partial V_{m,l}^n}{\partial Z} + \frac{\sigma^2 Z_{m,l}^n}{2} \frac{\partial^2 V_{m,l}^n}{\partial Z^2} = 0.\end{aligned}$$

Note that we work with the homogeneous equation (the right term is zero) just like we did for the GBM. Again we start with the derivation of the implicit scheme with some very similar approximations:

$$\frac{V_{m,l}^{n+1} - V_{m,l}^n}{\Delta t} + rS_{m,l}^n \frac{V_{m+1,l}^n - V_{m-1,l}^n}{2\Delta S} + \frac{1}{2}(S_{m,l}^n)^2 Z_{m,l}^n \frac{V_{m+1,l}^n - 2V_{m,l}^n + V_{m-1,l}^n}{(\Delta S)^2}$$

$$+\kappa(\theta - Z_{m,l}^n) \frac{V_{m,l+1}^n - V_{m,l-1}^n}{2\Delta Z} + \frac{1}{2}\sigma^2 Z_{m,l}^n \frac{V_{m,l+1}^n - 2V_{m,l}^n + V_{m,l-1}^n}{(\Delta Z)^2} = 0$$

$$\Updownarrow$$

$$V_{m,l}^{n+1} =$$

$$V_{m,l}^n - \Delta t r S_{m,l}^n \frac{V_{m+1,l}^n - V_{m-1,l}^n}{2\Delta S} - \Delta t \frac{1}{2}(S_{m,l}^n)^2 Z_{m,l}^n \frac{V_{m+1,l}^n - 2V_{m,l}^n + V_{m-1,l}^n}{(\Delta S)^2}$$

$$-\Delta t \kappa(\theta - Z_{m,l}^n) \frac{V_{m,l+1}^n - V_{m,l-1}^n}{2\Delta Z} - \Delta t \frac{1}{2}\sigma^2 Z_{m,l}^n \frac{V_{m,l+1}^n - 2V_{m,l}^n + V_{m,l-1}^n}{(\Delta Z)^2}.$$

When deriving the scheme for the GBM model we could write V as an vector at this point but now we need V to be an $M \times N$ matrix. And instead of having just one matrix operating on V we need two matrices, one from the left and one from the right. On matrix form the equation is:

$$V^{n+1} = (I - \Delta t D_s) V^n (I - \Delta t D_z)$$

$$\Updownarrow$$

$$V^n = (I - \Delta t D_s)^{-1} V^{n+1} (I - \Delta t D_z)^{-1},$$

where $D_s = D_{s1} + D_{s2}$ is an $M \times M$ matrix:

$$D_{s1} = \begin{bmatrix} 0 & 0 & 0 & & & \\ -a_1 & 0 & a_1 & & & \\ & -a_2 & 0 & a_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -a_{M-2} & 0 & a_{M-2} \\ & & & 0 & 0 & 0 \end{bmatrix} \quad , a_m = \frac{rS_m}{2\Delta S},$$

$$D_{s2} = \begin{bmatrix} 0 & 0 & 0 & & & \\ b_1 & -2b_1 & b_1 & & & \\ & b_2 & -2b_2 & b_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{M-2} & -2b_{M-2} & b_{M-2} \\ & & & 0 & 0 & 0 \end{bmatrix} \quad , b_m = \frac{(S_m)^2 Z_l}{2(\Delta S)^2},$$

and $D_z = D_{z1} + D_{z2}$ is an $L \times L$ matrix:

$$D_{z1} = \begin{bmatrix} -3c_0 & 4c_0 & -c_0 & & & \\ -c_1 & 0 & c_1 & & & \\ & -c_2 & 0 & c_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -c_{L-2} & 0 & c_{L-2} \\ & & & c_{L-1} & -4c_{L-1} & 3c_{L-1} \end{bmatrix} \quad , c_l = \frac{(\theta - Z_l)\kappa}{2\Delta Z},$$

$$D_{z2} = \begin{bmatrix} d_0 & -2d_0 & d_0 & & & \\ d_1 & -2d_1 & d_1 & & & \\ & d_2 & -2d_2 & d_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & d_{L-2} & -2d_{L-2} & d_{L-2} \\ & & & d_{L-1} & -2d_{L-1} & d_{L-1} \end{bmatrix} \quad , d_l = \frac{\sigma^2 Z_l}{2(\Delta Z)^2}.$$

The definition of the edges on D_s works exactly like they did for the GBM. The edges on D_z however is defined like this and will not be changed if we look at another contract. It would not really make sense to change them since the volatility is no tradable asset so no contract payoff is directly dependent on the volatility. Instead the edges on D_{z1} and D_{z2} are defined to always approximate the corresponding space derivative, just like for all the other rows but in a non symmetrical way. The Crank-Nicolson method can be created, just like for the GBM, giving us:

$$V^n = \Big((I - \frac{\Delta t}{2} D_s)^{-1} (I + \frac{\Delta t}{2} D_s)\Big) V^{n+1} \Big((I - \frac{\Delta t}{2} D_z)^{-1} (I + \frac{\Delta t}{2} D_z)\Big),$$

or with a more simple notation:

$$V^n = A_s V^{n+1} A_z. \tag{2.13}$$

Note that A_s is the same as in the GBM case apart from the dependency on Z_l, therefore the same notation for this matrix is used in both these schemes. This dependency will however affect the way we price the options because the dependency makes it crucial to take all the steps one at a time. Remember that the time, from $t = 0$ to $t = T$, is discretized into N time points. So we need to take $N - 1$ steps to go from the known value (payoff) at time $t = T$ to the value at time $t = 0$. If the matrices were totally independent we could have calculated just one big step by taking the $(N - 1)$:th powers of the transition matrices, just like we did for the GBM scheme. But because A_s is depending on Z_l in this Heston scheme we must take $(N - 1)$ smaller steps to get the value at $t = 0$, making the calculations a bit slower. But to evaluate an American option all these small steps are necessary to take anyway, making this a suitable method to use to price American options.

The independence of W_t^1 and W_t^2 ($\rho = 0$) makes it possible to write this scheme (2.13) in this locally one dimensional (LOD) way, meaning that we can work with one dimension at a time when taking a step. Basically it means that we have two separate matrices for the two dimensions. The whole point of this paper is to have this equation on this form so that we can see A_s and A_z as transition matrices that can be used to simulate trajectories forward in time for S_m and Z_l. If we have $\rho \neq 0$ and still want to write the equation in matrix form we would have to take all the cross terms into account and the matrix would therefore have to be huge compared to the LOD case. So being able to write the scheme in this simple LOD form means we gain a lot when it comes to computational complexity. The obvious problem with this scheme is that we can't use it to directly price options if W_t^1 and W_t^2 are correlated. However there is a way to include the correlation when simulating with the LOD scheme, which will be explained in section 2.10.4.

2.10 Simulation

This simulation section can be seen as the main part of this paper because this is what makes our option pricing method interesting. The possibility to very easily simulate from the model is what sets this model apart from other pricing methods. Having a large number of simulation samples would be very useful to analyse different attributes of the option. The most simple way to use the the samples would be to calculate the expected value with the standard Monte Carlo method, i.e. by taking the average of all the simulations. But the simulation also makes it possible to analyse the option in other ways.

The discretized solution in the finite difference scheme (2.13) can be interpreted as a Markov chain when simulating forward in time, meaning A_s and A_z can be seen as transition matrices. Recall the Feynman-Kac representation of the PDE that stated that the solution to the PDE is an expected value of the option. For the discretized scheme this should also hold, at least when N, M and L goes to infinity. But for finite values it should be a good approximation. That means the calculated values at time point n can be seen as an expected value of the option value in time point $n+1$ which implies that the weights in the transition matrix can be seen as probabilities for the different outcomes, at least if they fulfill these probability conditions:

$$\sum_j A_{ij} = 1 \forall i, \qquad 0 \leq A_{ij} \leq 1, \tag{2.14}$$

where A is a transition matrix.

2.10.1 GBM Simulation

To keep it as simple as possible to begin with we start with the GBM. We have introduced a few different options and schemes already but to begin with we assume a double no touch option and a fully implicit scheme (2.11). The reason is that this combination has some very nice characteristics when it comes to the transition matrix, A_s^{imp}. More precisely it will guarantee that the probability conditions (2.14) are met, according to [Andreasen, Huge , 2010, p. 5-6], meaning A_s^{imp} will have a perfect probability interpretation. This matrix can therefore be used to simulate trajectories with the transitional probabilities given by:

$$Pr[S^{n+1} = S_j \mid S^n = S_i] = (A_s^{imp})_{ij}. \tag{2.15}$$

While this seems like a crucial characteristic to have it is not essential for the simulation to work, to some extent. This may seem strange to begin with but it is in fact possible to use the transition matrix for simulation even though some weights are negative and/or greater than one. Exactly how to handle this will be explained later in this section but for now we just assume that the transition matrix no longer have to fulfill the probability conditions (2.14). That means the definitions of the edges of the matrices can be defined in any manner, meaning we can price other contracts than the double no touch option. It also means that the implicit scheme is not the only possible scheme to use. However, when taking a lot of time steps, the solution can become very sensitive and almost seemingly unstable if the probability conditions are greatly violated. But in the case where the weighs are "close" to fulfilling the conditions it can be handled. If we use the Crank-Nicolson

scheme the conditions are not guaranteed to be fulfilled for all rows but it will often be fulfilled "close enough" to make this method work, making this a suitable scheme in many cases.

With a given start point for the underlying asset at time $t = 0$ we simply take one step forward in time by looking at the conditional probabilities (2.15) in the row corresponding to the start point. Note that one time step has the length $\frac{T}{N}$. After taking a step we will end up in another row (or possibly the same) and we can do this again and again. This can be done for any number of steps. For the double no touch option there is no problems with taking many steps but for many other options the definition of the edges will be problematic for the simulation. This is because it can be hard, or even impossible, to make the option have the desired behavior on the edges. For the double no touch option it is easy to implement the edges correctly because if we set the edges in D_s to just zeroes it means the edge will absorb the trajectory, meaning if it ever reaches the edge it will stay there, which corresponds to exactly how the double no touch option works. This is one of the reasons the double no touch option is so well-used in this paper. However, if we start somewhere in the middle of the matrix and only simulate for a short time interval the trajectories will almost never reach the edge (depending on all the parameters). In that case the problem with the edges will not make a big impact. But for longer simulations the trajectories will reach the edge more often and then the double no touch option should theoretically get the most accurate results compared to other options.

Another concern with other edges on D_s, for example (2.12), is that the weights in A_s^{imp} don't necessarily fulfill $0 \leq A_s^{imp}$ (but each row will still sum to one), as we explained earlier in this section. The same concern comes up for all options (even the double no touch option) if we use the explicit or the Crank-Nicolson scheme. This problem is solved by "simulating with negative probabilities".

2.10.2 Simulation with negative probabilities

Assume a general transition matrix, A_s, where the entries can't be directly seen as probabilities because they might be negative or greater than one. The Feynman-Kac representation and the connections discussed earlier still obviously holds so V^n can still be seen as an expected value of V^{n+1}, however A_s can't be directly used to simulate from because the conditional simulation (2.15) only works when the probability condition (2.14) is met. So we want to somehow make the weights fulfill this condition. This can be done by first taking the absolute value of all weights and then normalizing each row so they sum to one. This can be seen as a change of numeraire, R. This means

the conditional transition probabilities can be written as:

$$Pr[S^{n+1} = S_j \mid S^n = S_i] = \frac{\mid (A_s)_{ij} \mid}{\sum_j \mid (A_s)_{ij} \mid}.$$

To make up for this change we have to multiply the final payoff with the numeraire which is updated in every step according to:

$$R_{n+1} = R_n \text{sign}((A_s)_{ij}) \sum_j^M |(A_s)_{ij}|, \tag{2.16}$$

where i corresponds to S^n and j corresponds to S^{n+1}.

This update has to be done for every step we take because it depends on every i and j and will therefore change over time. The rest of the simulation works exactly like it did before.

Now maybe it seems like there is no problem at all with probability weights being negative but there is in fact one big concern. If some weights are negative the Monte Carlo simulation will only converge in expected value. That means we can only calculate the expected value of the options with Monte Carlo simulation. If we want to do some other statistical analysis the transition matrices must fulfill the probability conditions (2.14). For instance we can calculate different percentiles for the value of an option given a start point if, and only if, the probability conditions are met.

2.10.3 Heston Simulation

The simulation for the Heston scheme is very similar to the GBM simulation but for the Heston model we also have to simulate the volatility. This is done in the same way except that we simulate from the columns of A_z because this matrix is multiplied from the right of V^n (2.13). The general conditional transition probabilities for the volatility can be written as:

$$Pr[Z^{n+1} = Z_i \mid Z^n = Z_j] = \frac{\mid (A_z)_{ij} \mid}{\sum_i \mid (A_z)_{ij} \mid}.$$

One time step in the Heston model is simulated by taking two intermediate steps. First one step to simulate S_m and then one step for Z_l. The numeraire product (2.16) is now updated after every intermediate step. For an intermediate step in S_m the numeraire is updated just like before and for a Z_l step it is updated very similarly, only the obvious changes are made. The final payoff is then multiplied with the numeraire to get the correct output.

We have already mentioned some problems when simulating with negative

probabilities but some problems can be made more clear now when the numeraires are introduced. One thing to note is that the numeraire product (2.16) grows exponentially fast. If we take a lot of steps and most numbers are greater than one the product will get too big for this method to work. But if most numbers are close to one there is no problem. However it turns out that this problem appears for the D_{z1} operator in the Heston scheme. This has to do with the term $(\theta - Z_l)$ being sometimes positive and sometimes negative depending l. But this is not an unsolvable problem. We can modify this operator into a so called upwind operator that changes the derivative approximation depending on the sign of $(\theta - Z_l)$. The upwind version of D_{z1} is defined as:

$$D^u_{z1} = \begin{bmatrix} -\gamma_0 & \gamma_0 & & & & & \\ -\gamma_1^- & (\gamma_1^- - \gamma_1^+) & \gamma_1^+ & & & & \\ & -\gamma_2^- & (\gamma_2^- - \gamma_2^+) & \gamma_2^+ & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\gamma_{L-2}^- & (\gamma_{L-2}^- - \gamma_{L-2}^+) & \gamma_{L-2}^+ & \\ & & & & -\gamma_{L-1} & \gamma_{L-1} \end{bmatrix},$$

$$\gamma_l^+ = max(0, \gamma_l), \qquad \gamma_l^- = min(0, \gamma_l), \qquad \gamma_l = \frac{\kappa(\theta - Z_l)}{\Delta Z}.$$

Using this operator and a implicit scheme will in fact guarantee that the probability conditions (2.14) are met for A_z, according to [Andreasen, Huge , 2010, p. 5-6]. This operator will generally give a slightly worse approximation but it solves the problem with the numeraire product growing to big and is thus very useful. Therefore this upwind version will be used for all simulations with the Heston Model in this paper. However if one want to only calculate the option values with the Heston finite difference scheme, and not use the calculated matrices to simulate anything, D_{z1} can be used instead of D^u_{z1} to get a slightly better result.

2.10.4 Simulation with correlation

Recall the full version of the Heston Model PDE (2.7). Until now we only considered the case where $\rho = 0$ because we want to have the final equation for the scheme on this simple LOD form (2.13). If the Wiener processes in the Heston Model (2.5) are correlated this kind of LOD finite difference scheme can't be obtained. But if we only want to simulate trajectories forward in time we can in fact include the case where $\rho \neq 0$ by extending our simulation method. For every step we take forward in time we can add a step that simulates the correlation between S_m and Z_l. We take a correlation step in the following manner:

$$\hat{V}^n_{m,l} = (1 + \Delta t D_{sz}) V^n_{m,l},$$

where $\hat{V}^n_{m,l}$ can be seen as the correlated version of $V^n_{m,l}$ and the correlation operator, D_{sz}, is defined as:

$$D_{sz}V^n_{m,l} = \frac{\sigma\rho S^n_m Z^n_l}{4\Delta s\Delta z}(V^n_{m+1,l+1} - V^n_{m+1,l-1} - V^n_{m-1,l+1} + V^n_{m-1,l-1}). \quad (2.17)$$

Recall that every step in the Heston scheme is done by taking one step for S_m and one step for Z_l, one at a time. To get the best results for the correlated simulation we can split the correlation step into two steps. One half correlation step right before the S_m step and another half correlation step right before the Z_l step. This means that $(1 + \frac{1}{2}\Delta t D_{sx})$ is the transition matrix we will use when taking a correlation step and therefore we introduce the notation $A_{sz} = (1 + \frac{1}{2}\Delta t D_{sx})$. Note that some weights in A_{sz} will be negative but this is solved by simply simulating with "negative probabilities" in a similar manner to how we solved this problem for A_s and A_z and the numeraire product is updated after every intermediate step.

This part shows how to implement the somewhat vague suggestion on how to solve the correlated case is briefly described in [Andreasen, Huge , 2010, 13]. The aim is to simulate S^{n+1} and Z^{n+1} given S^n and Z^n. To do this we need to proceed in four intermediate simulation steps, where the numeraire is updated after every intermediate step:

- Generate $S^{n+1,1}$ and $Z^{n+1,1}$ given S^n and Z^n using A_{sz}.
- Update $R_{n+1,1} = R_n \text{sign}((A_{sz})_{ij})/(1 - 4\frac{\sigma\rho S^n_m Z^n_l}{8\Delta s\Delta z})$.
- Generate $S^{n+1,2}$ given $S^{n+1,1}$ using A_s.
- Update $R_{n+1,2} = R_{n+1,1}\text{sign}((A_s)_{ij}) \sum_j^M |(A_s)_{ij}|$.
- Generate $S^{n+1,3}$ and $Z^{n+1,2}$ given $S^{n+1,2}$ and $Z^{n+1,1}$ using $(1+\frac{1}{2}\Delta t D_{sz})$.
- Update $R_{n+1,3} = R_{n+1,2}\text{sign}((A_{sz})_{ij})/(1 - 4\frac{\sigma\rho S^n_m Z^n_l}{8\Delta s\Delta z})$.
- Generate $Z^{n+1,3}$ given $Z^{n+1,2}$ using A_z.
- Update $R_{n+1,4} = R_{n+1,3}\text{sign}((A_z)_{ij}) \sum_i^L |(A_z)_{ij}|$.

Finally we set $S^{n+1} = S^{n+1,3}$, $Z^{n+1} = Z^{n+1,3}$ and $R_{n+1} = R_{n+1,4}$. The numeraire updates that correspond to the correlation step can be derived from the definition of D_{sz} (see 2.17).

Chapter 3

Results

3.1 Finite Difference Scheme Results

This paper is mainly about the connection between the direct pricing with the finite difference schemes and the Monte Carlo simulation. However it can be interesting to see some results of the direct pricing with finite difference schemes to get a better understanding of what we are doing. The finite difference (and simulation) schemes are implemented in MATLAB.

3.1.1 GBM Finite Difference Method Results

For this section we use the Crank Nicolsen method with a 1001×201 grid ($N \times M$). One European call option and one binary double no touch option are priced and the corresponding plots can be seen on page 20 and 21. For the call option S_m goes from 20 to 420 and for the double no touch option S_m goes from 75 to 125. The options are assumed to have 1 year to maturity and the model parameters are specified as $r = 0.1$ and $\sigma = 0.1$. For the European call option the true values are known so we can also show a figure of the pricing error.

3.1.2 Heston Finite Difference Method Results

The Crank-Nicolson method is used for this section as well, with a $201 \times 101 \times 101$ grid ($N \times M \times L$). We price a European call option, a double no touch option and an American put option and the corresponding plots can be seen on page 21, 22 and 23. For the call option the S_m grid goes from 0 to 400, for the double no touch option S_m goes from 50 to 150 and for the put option S_m goes from 0 to 300. For all three options Z_l goes from 0 to 1. The options are assumed to have 1 year to maturity and the model parameters set to $r = 0.1$, $\kappa = 5$, $\theta = 0.2$ and $\sigma = 0.1$. The pricing error for the European call option is also shown.

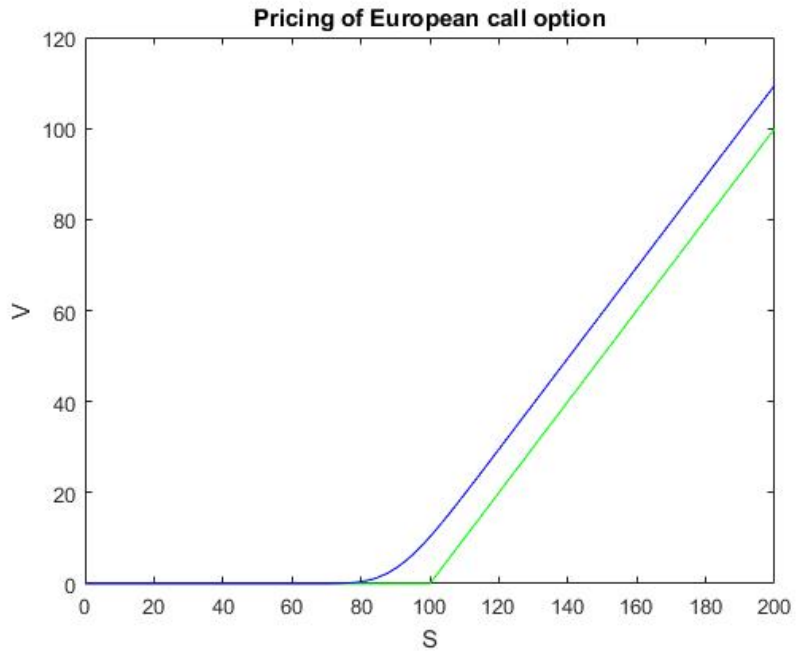

Figure 3.1: Calculated value at $t = 0$ (blue) and payoff at $t = 1$ (green) of a European call option with strike $K = 100$, with the GBM finite differenc scheme.

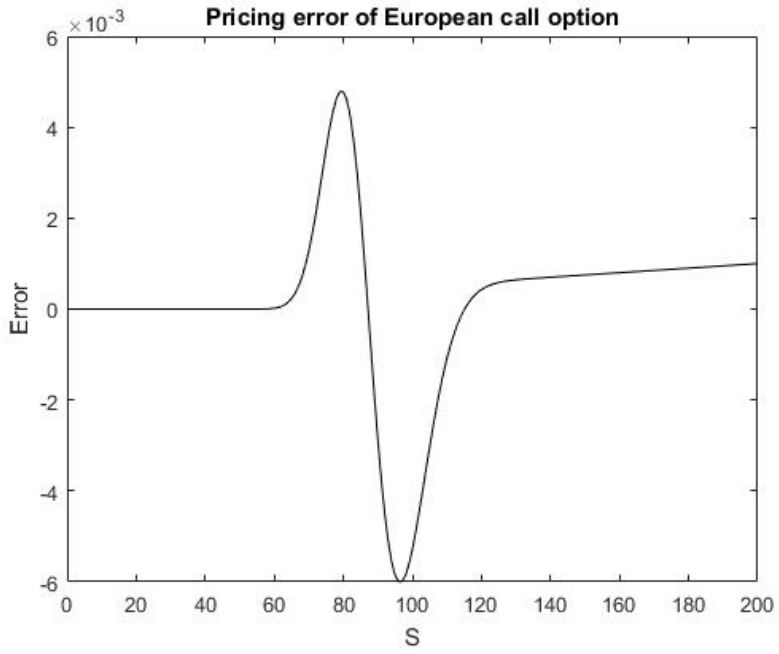

Figure 3.2: Error of the option evaluation in Figure 3.1.

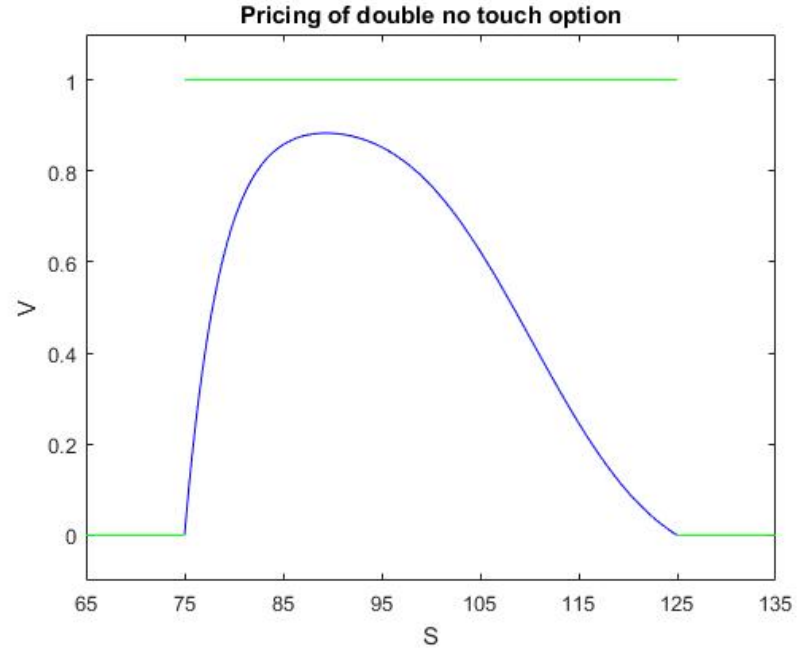

Figure 3.3: Calculated value at $t = 0$ (blue) and payoff at $t = 1$ (green) of a double no touch option with barrier levels $L_B = 75$ and $H_B = 125$, with the GBM finite difference scheme.

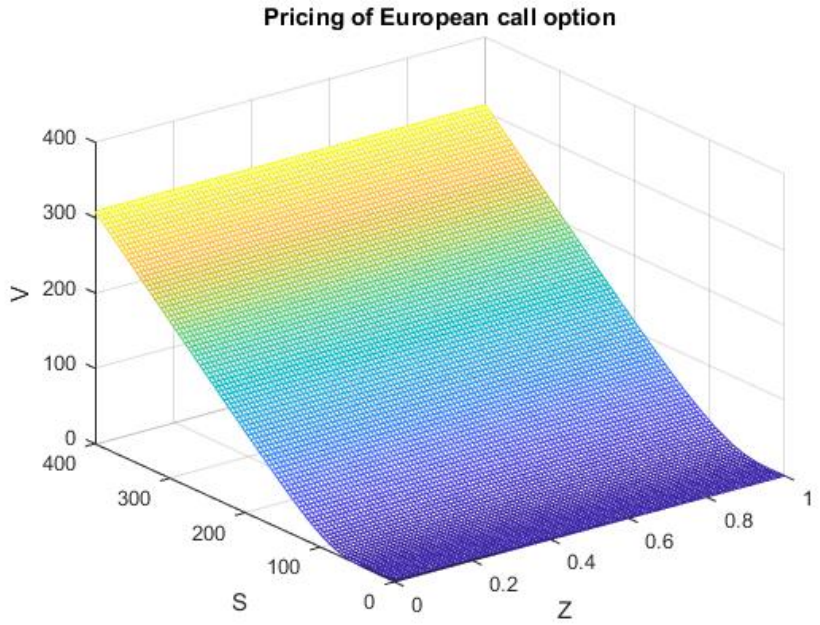

Figure 3.4: Evaluation of a European call option with strike $K = 100$ and 1 year to maturity, with the Heston finite difference scheme.

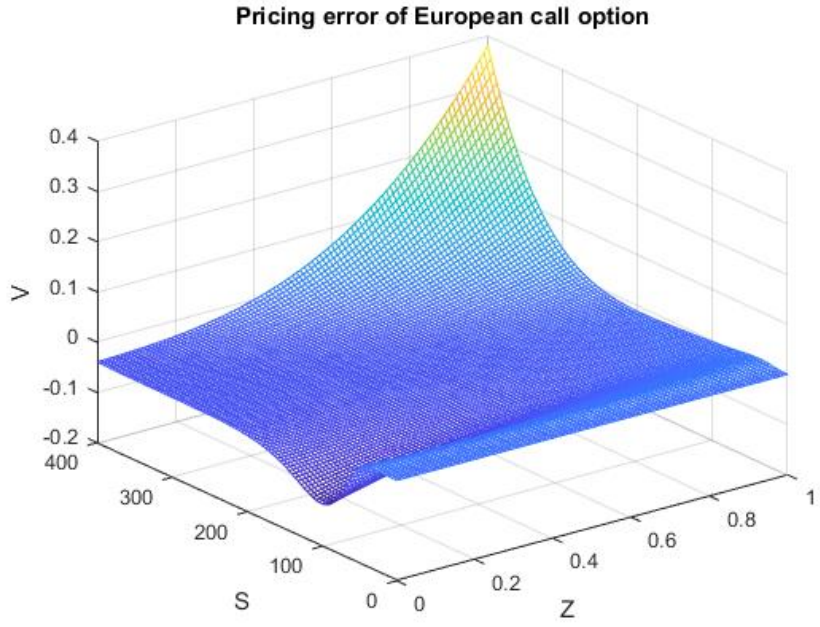

Figure 3.5: Error of the evaluation in Figure 3.4.

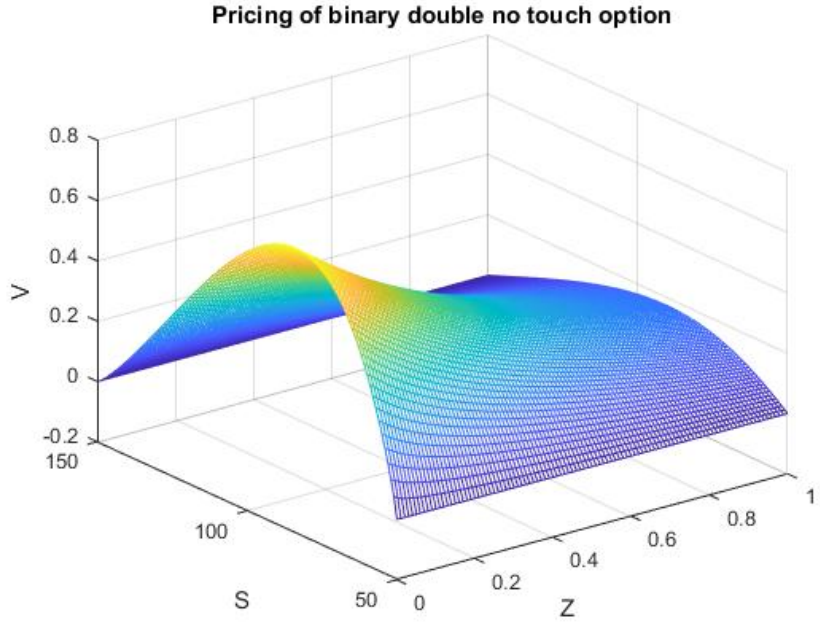

Figure 3.6: Evaluation of a double no touch option with barrier levels $L_B = 50$ and $H_B = 150$ and 1 year to maturity, with the Heston finite difference scheme.

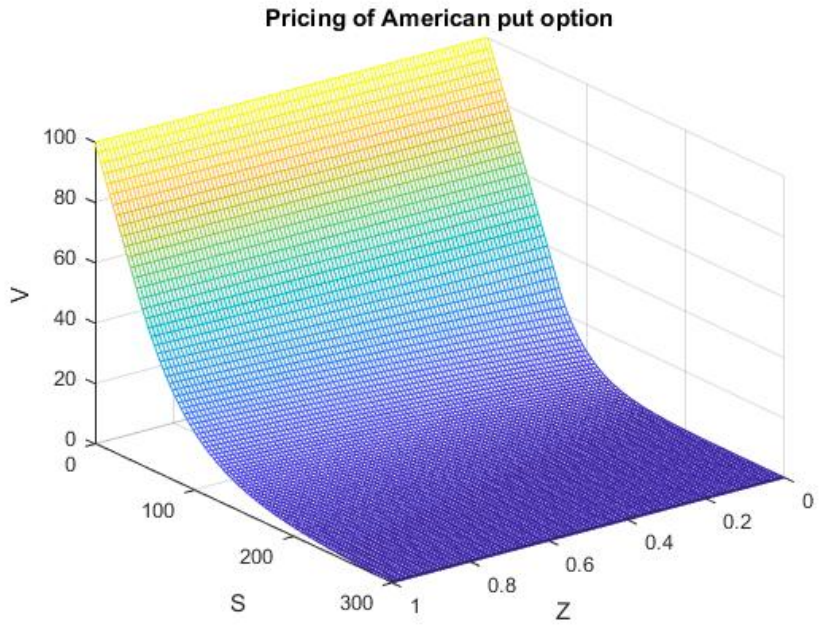

Figure 3.7: Evaluation of an American put option with strike $K = 100$ and 1 year to maturity, with the Heston finite difference scheme.

3.2 Simulation Results

For this section we only consider the Heston model with the upwind operator D^u_{z1}. Every under section to this one is based on a specific option. However, to keep it as simple as possible, we will reuse $V^n_{m,l}$ as notation for the value in the grid point (n, m, l) in every section.

3.2.1 Simulation 1

To begin with we assume a fully implicit scheme for both S_m and Z_l with the upwind operator, D^u_{z1}, and a binary double no touch option. This implies that both A_s and A_z will have a perfect probability interpretation. We use a $201 \times 101 \times 101$ grid where S_m goes from $L_B = 50$ to $H_B = 150$ and Z_l goes from 0 to 1. Furthermore the double no touch option has 1 year to maturity and the option parameters are $r = 0.1$, $\kappa = 5$, $\theta = 0.2$ and $\sigma = 0.1$. We start the simulation with $m = 51$ and $l = 21$ meaning $S_m = S_{51} = 100$ and $Z_l = Z_{21} = 0.2$ to begin with. To calculate the derivative prices we use that fact that the value of an option at time $t = 0$ can be seen as expected values of the later values:

$$V(0, S_0, Z_0) = E[V(T, S_T, Z_T)|S(0) = S_{51}, Z(0) = Z_{11}].$$

This has clear connections with the Feynman-Kac formula (2.8). The expected value is approximated by taking the average value of a large number of simulations. We run 10^6 simulations and use this to price the option according to:

$$e^{-rT}V^0_{51,11} \approx e^{-rT}\frac{1}{10^6}\sum_{i=1}^{10^6}(V^{200}_{\bullet,\bullet})_i = 0.4692,$$

where $e^{-rT}V^0_{51,11}$ is the estimated value of the option and $e^{-rT}(V^{200}_{\bullet,\bullet})_i$ is one simulated value at time $t = T$ which corresponds to $n = 200$ in this case. The interest rate term, e^{-rT}, is there to make up for the change we made in the PDE in the beginning of the finite difference scheme section.

Since the transition matrices fulfil the probability condition (2.14) we can not only use the simulation to estimate the EV of the option. By listing all the final values for all trajectories, $(V^{200}_{\bullet,\bullet})_i$, we can get an estimated value of any percentile we want. If we assume $(V^{200}_{\bullet,\bullet})_i$ is sorted from the smallest to the greatest value we can calculate the 5 and 95 percentiles as:

$$(V^{200}_{\bullet,\bullet})_{50000} = 0.2660, \qquad (V^{200}_{\bullet,\bullet})_{950000} = 0.5668.$$

This means that with 90% probability the payoff of the option at time $t = 1$ will be in the range from 0.2660 to 0.5668. We would not be able to do this kind of analysis with only the finite difference scheme so this is a prime example of why the simulation part is interesting.

3.2.2 Simulation 2

We now consider a European call option with the Crank Nicolson method, meaning we have to simulate with negative probabilities. This implies that we can't calculate any percentiles but the expected value of the option can still be calculated. We use a $201 \times 101 \times 101$ grid where S_m goes from 20 to 420 and Z_l goes from 0 to 1. The option is assumed to have 1 year to maturity and strike $K = 100$. The option parameters are $r = 0.1$, $\kappa = 5$, $\theta = 0.2$ and $\sigma = 0.1$. We start the simulation with $m = 21$ and $l = 21$ meaning $S_m = S_{21} = 100$ and $Z_l = Z_{21} = 0.2$. We run 10^6 simulations and get:

$$e^{-rT}V^0_{51,11} \approx e^{-rT}\frac{1}{10^6}\sum_{i=1}^{10^6}(V^{200}_{\bullet,\bullet})_i = 21.99.$$

This can be compared to the true value: 22.02.

3.2.3 Simulation 3

Now we consider the case were correlation is included in the simulation. The correlation is assumed to be $\rho = -0.3$. We assume that the option

only has 0.1 years to maturity. This is to make sure that no trajectories will reach the edge of the grid, because the implementation of the edges is somewhat problematic. This shorter time period makes it impossible for the trajectories to reach the edge of the grid. Apart from this change the setup is very similar to the last section (Simulation 2), meaning we have a 201×101 grid where S_m goes from 20 to 420 and Z_l goes from 0 to 1. Furthermore the option parameters are $r = 0.1$, $\kappa = 5$, $\theta = 0.2$ and $\sigma = 0.1$ and the strike price is $K = 100$. The simulation begins with $m = 21$ and $l = 21$ meaning $S_m = S_{21} = 100$ and $Z_l = Z_{21} = 0.2$. However we only run 10^5 simulations because of the increase in time complexity this scheme will have:

$$e^{-rT} V^0_{51,11} \approx e^{-rT} \frac{1}{10^5} \sum_{i=1}^{10^5} (V^{200}_{\bullet,\bullet})_i = 10.91.$$

This can be compared to the true value: 6.120.

3.2.4 Simulation 4

Again we consider a similar setup as for Simulation 2 meaning we have a $201 \times 101 \times 101$ grid where S_m goes from 20 to 420 and Z_l goes from 0 to 1. The option is assumed to have 1 year to maturity and strike $K = 100$. The option parameters are $r = 0.1$, $\kappa = 5$, $\theta = 0.2$ and $\sigma = 0.1$ and we start the simulation with $S_m = S_{21} = 100$ and $Z_l = Z_{21} = 0.2$. However one big change is made. Because of the error that might come with the definition of the grid edge we want to simulate for a shorter time period so that the trajectories never reaches the edges. One thing we can do is to simulate forward to time point $t = t_1$ (where $0 < t_1 < 1$) and calculate the value of the option at time t_1 with the finite difference methods. If we chose t_1 close enough to 0 and start the simulation somewhere in the middle of the grid the trajectories will not reach the edge. Note that this can't be done in the case were we have correlation because the correlation effect can only be included in the simulation and not in the finite difference scheme. The setup we have is identical to Simulation 2 apart from that we now don't simulate to time $t = T$ but only to time $t = t_1$. And instead of using the option payoff at time $t = T$ we use the option value at time $t = t_1$ that we can calculate with the finite difference methods. We use $t_1 = 0.1$ and run 10^6 simulations:

$$e^{-rT} V^0_{51,11} \approx e^{-rT} \frac{1}{10^6} \sum_{i=1}^{10^6} (V^{200}_{\bullet,\bullet})_i = 22.00.$$

Now $V^{200}_{\bullet,\bullet}$ is the value at time $t = 0.1$ that we can consider known because we can calculate it in advance with the finite difference methods. This can again be compared to the true value: 22.02.

Because of this decrease in the time length for the simulation part we will with very high probability never have to simulate with negative probabilities. We can verify this by looking at the final numeraire product. If the product is 1 it means that the weights have a perfect probability interpretation, which is the case for this simulation. That means we can calculate percentiles for this option. We sort $(V^{200}_{\bullet,\bullet})_i$ from the smallest to the greatest value and calculate the 5 and 95 percentiles as:

$$(V^{200}_{\bullet,\bullet})_{50000} = 9.292, \qquad (V^{200}_{\bullet,\bullet})_{950000} = 41.74.$$

This now corresponds to values at time $t = t_1 = 0.1$.

3.2.5 Simulation 5

In these sections we have simulated an underlying asset and used this to price an option. If we have more than one option that depend on the same underlying asset we can price all these options simultaneously without any extra cost because we still only have to simulate one underlying asset. This means we can very effectively simulate an portfolio of options that all are based on the same underlying asset. We now consider a portfolio of 10 different European call options with equal share (10% each) but with different strike prices $K_i = 95 + i$, where $i = 1, 2, ..., 10$.

Apart from this change we assume an identical setup as for Simulation 4, meaning the options have 1 year to maturity but we calculate the portfolio values at time $t = t_1 = 0.1$ with the finite difference scheme so that we only have to simulate forward to this time point. Note that this portfolio can be seen as a normal option with a slightly more complicated payoff function, so in practise the only adjustment we have to make is to change the payoff function so it correspond to the portfolios payoff.

The grid and the parameters are also identical to Simulation 4 meaning we have a $201 \times 101 \times 101$ grid where S_m goes from 20 to 420 and Z_l goes from 0 to 1. Furthermore the option parameters are $r = 0.1$, $\kappa = 5$, $\theta = 0.2$ and $\sigma = 0.1$. The simulation starts with $S_m = S_{21} = 100$ and $Z_l = Z_{21} = 0.2$. We run 10^6 simulations and get:

$$e^{-rT} V^0_{51,11} \approx e^{-rT} \frac{1}{10^6} \sum_{i=1}^{10^6} (V^{200}_{\bullet,\bullet})_i = 21.80,$$

where $e^{-rT} V^0_{51,11}$ is the estimated value of the portfolio and $(V^{200}_{\bullet,\bullet})_i$ is one simulated value at time $t = t_1 = 0.1$. Since this portfolio only is a linear combination of call options we have access to the true value: 21.83.

Just as for Simulation 4 the probability weights will almost certainly have

a perfect probability interpretation, even though it is not guaranteed, which means we can calculate the 5 and 95 percentiles as:

$$(V^{200}_{\bullet,\bullet})_{50000} = 9.183, \qquad (V^{200}_{\bullet,\bullet})_{950000} = 41.61.$$

This means that with 90% probability the value of the portfolio will be in the range from 9.183 to 41.61 at time $t = 0.1$.

Chapter 4

Discussion and Conclusions

4.1 Discussion of Finite Difference Method Results

Two different options were priced with the GBM finite difference method (see figure 3.1 and 3.3). We don't have access to the true values for the binary double no touch option so we can't tell exactly how accurate it is but for the European call option the true values are known. The pricing error for the European call option (see figure 3.2) is less than 0.006 for the whole pricing range which indicates that this method is fairly accurate. However one should keep in mind that small pricing errors can be enough to make a big impact in finance. Even for $S_m < 60$ where the absolute error is very small there is still a problem. While it is true that the absolute error is small, that is only because the price we want to evaluate is very small as well. If we instead would look at the relative error it would become clear that the pricing is far from perfect here as well. Luckily this paper is not about pricing options as precise as possible but rather to be able to simulate the underlying asset and for that application this method is definitely accurate enough.

For the Heston finite difference method the pricing error of the European call option (see figure 3.5) is larger than for the GBM, which is expected because of the increase in dimension. But just as for the GBM we consider this accuracy good enough for the main application, i.e. the simulation. Also note that the greatest errors are in the top corner so if we start the simulation somewhere around the bottom corner and simulate for a short time period almost no trajectory will reach the top corner and thus it wont affect the simulation results too much.

For good measure the pricing of an American option is included as well (see figure 3.7). The American call option is not very interesting because it is never optimal to exercise it before maturity so the price of an American call option is identical to the European version, provided that $r > 0$. This is

not true for the put option however, making this a more interesting option to evaluate.

To summarize we can conclude that the finite difference method is not used to get a perfect option evaluation but rather as theory to make the simulation possible. The simulation can then be used in more ways than just evaluating the option price, at least in some situations.

4.2 Discussion of Simulation Results

For all simulations with the European call option the true values are known. We see that the calculations of the expected values are about as accurate as the direct finite difference pricing in most cases. This is expected since the simulation is based on the same theory as the finite difference method. The simulation error is not constant however because of the variance that the Monte Carlo simulation will bring.

It is reasonable to assume that the accuracy for simulation with the binary double no touch option (section 3.2.1) is just as good as the corresponding simulations with the European call option. It might even be better because the definition of the grid edges are more precise for the double no touch option.

However we can note that there is one exception to all these good results. When simulating with correlation in section 3.2.3 the error is really big. This is partly a consequence of increased variance. Running the whole simulation again, with the same setup, gives a result that is not even close to neither the last simulation value nor the true value. Note that we only run 10^5 simulations (compared to the other cases where we run 10^6 simulations) which of course will make the variance higher, but not to this degree, so that does not solely explain the error. More likely is that the huge variance is a consequence of the correlation steps. Since there is no possible way to avoid negative probability weights when simulating with correlation the numeraires will get very large compared to the similar cases where we simulation without correlation and this will increase the variance. Normally a change in correlation from $\rho = 0$ to $\rho = -0.3$ would only correspond to a millesimal change for the option value. With this said we can conclude that it is not worth to consider any correlation when simulating with this method because the added variance is so much greater than the true change in value the correlation will produce. The precision would be much better by just ignoring the correlation and simulating as if $\rho = 0$, although this would give a slightly biased value.

We can conclude that the simulation with negative probabilities seems to work well if we can keep the numeraire small. The problem however is that it only converges in expected value so in the cases where the numeraire is greater than 1 we can not calculate percentiles. Luckily we found a way to avoid negative probabilities by simulating for only a short time period and use this in combination with the finite difference method, as we did in section 3.2.4 and 3.2.5. While this worked out nicely for the specific setup we used it does not generally guarantee that no weights are negative. The setup in section 3.2.1 is of course ideal if we want to avoid negative weights, since this guarantees that the probability conditions (2.14) are met, but unfortunately this setup is only possible for the double no touch option where the edges of the transition matrices are defined to be zero.

Bibliography

Åberg, S. (2018). Derivative Pricing, *KF Sigma, Lund*.

Andreasen, J and Huge, B. (2010). Finite Difference Based Calibration and Simulation. Preprint availble at: https://www.researchgate.net/profile/Jesper_Andreasen2/publication/235622407_Random_grids/links/54634c760cf2c0c6aec330dd.pdf

Crank, J and Nicolson, P. (1947). A practical method for numereical evaluation of solutions of partial differential equations of the heat-conduction type, *Cambridge Philisophical Society*, **43**, 50–67.