Numerical methods for pricing American put options under stochastic volatility

D Joubert

20405537

Dissertation submitted for the degree *Magister Scientiae* in Applied Mathematics at the Potchefstroom Campus of the North-West University

Supervisor: Mnr LM Viljoen

Co-Supervisor: Dr EHA Venter

September 2013

Numerical methods for pricing American put options under stochastic volatility

Dominique Joubert

7 July 2013 (Version 12)

Acknowledgements

I would like to thank Mr. Thinus Viljoen and Dr. Antoinetta Venter for all their guidance during the past two years, it has truly been an insightful journey. I would also like to thank the National Research Foundation (NRF) for their financial contribution.

Declaration

I declare that, apart from the assistance acknowledged, the research presented in this dissertation is my own unaided work. It is being submitted in partial fulfillment of the requirements for the degree Master of Science in Applied Mathematics at the Potchefstroom campus of the North-West University. It has not previously been submitted before for any degree or examination to any university.

Nobody, including Mr. Thinus Viljoen and Dr. Antoinetta Venter, but myself is responsible for the final version of this dissertation.

Signature

. . . . . .

Date

. . . . . .

Abstract and key terms

The Black-Scholes model and its assumptions has endured its fair share of criticism. One problematic issue is the model's assumption that market volatility is constant. The past decade has seen numerous publications addressing this issue by adapting the Black-Scholes model to incorporate stochastic volatility. In this dissertation, American put options are priced under the Heston stochastic volatility model using the Crank-Nicolson finite difference method in combination with the Projected Over-Relaxation method (PSOR). Due to the early exercise facility, the pricing of American put options is a challenging task, even under constant volatility. Therefore the pricing problem under constant volatility is also included in this dissertation. It involves transforming the Black-Scholes partial differential equation into the heat equation and re-writing the pricing problem as a linear complementary problem. This linear complimentary problem is solved using the Crank-Nicolson finite difference method in combination with the Projected Over-Relaxation method (PSOR). The basic principles to develop the methods necessary to price American put options are covered and the necessary numerical methods are derived. Detailed algorithms for both the constant and the stochastic volatility models, of which no real evidence could be found in literature, are also included in this dissertation.

Key terms: Early exercise boundary, free boundary value problem, linear complimentary problem, Crank-Nicolson finite difference method, Projected Over-Relaxation method (PSOR), stochastic volatility, Heston stochastic volatility model.

Opsomming en sleutelterme

Die Black-Scholes model en sy aannames word al vir ’n geruime tyd gekritiseer. Een van die hoof probleemareas is die model se aanname van ’n konstante volatiliteit. Oor die afgelope dekade is heelwat navorsing gedoen om hierdie probleem aan te spreek deur die Black-Scholes model te inkorporeer binne ’n stogastiese model. In hierdie dissertasie word die Amerikaanse verkoopsopsie onder die Heston stogastiese model geprys deur gebruik te maak van die Crank-Nicolson eindige differensiemetode tesame met die geprojekteerde oorverslappingsmetode (PSOR). Die proses van prysbepaling van Amerikaanse verkoopsopsies, selfs onder konstante volaliteit, is gekompliseerd omdat die Amerikaanse opsie voor die vervaldatum uitgeoefen kan word. Die konstante volaliteitprobleem word ook in hierdie dissertasie volledig beskryf. Die konstante volaliteitprobleem behels die transformasie van die Black-Scholes parsi*ë*le differensiaalvergelyking na die hittevergelyking. Die probleem word dan herskryf as ’n lin*ê*ere komplimentê*r*e probleem wat opgelos word met behulp van die Crank-Nicolson eindige differensiemetode tesame met die geprojekteerde oorverslappingsmetode (PSOR). Die basiese beginsels wat benodig word om ’n metode te ontwikkkel wat gebruik kan word om die waarde van ’n Amerikaanse verkoopsopsie te bepaal word bespreek en die nodige numeriese metodes word afgelei. Gedetaileerde algoritmes vir beide die konstante en die stogastiese volatiliteitsmodelle, word ook in die dissertasie ingesluit.

Sleutelterme: Vroe*ë* uitoefengrens, vrye grenswaardeprobleem, lin*ê*ere komplimentê*r*e probleem, Crank-Nicolson eindige differensiemetode, geprojekteerde oorverslappingsmetode (PSOR), stogastiese volatiliteit, Heston stogastiese volatiliteitsmodel.

Nomenclature

- **General**
  t: Time.
  T: Exercise date, date of maturity.
  S: Underlying stock price.
  S_t: Underlying stock price at time t.
  S_0: Underlying stock price at the beginning of the option contract.
  S_T: Underlying stock price at the end of the option contract.
  K: Strike price.
  r: Current interest rate.
  C: American call option price.
  P^{Am}: American put option price.
  c: European call option price.
  p: European put option price.

- **Chapter Two**
  μ: Constant drift of the underlying asset.
  σ: Constant volatility of the underlying asset.
  W: Wiener process.
  V: Option price.
  dS: Change in underlying asset price.
  dt: Change in time.

- **Chapter Three**
  Π: Value of a portfolio.
  $S_f(t)$: Critical asset price.
  $\Lambda(S(t))$: Option payoff function.
  $u(x,\tau)$: Function of two variables used to solve option pricing problem under constant volatility.
  x: Transforming variable relating S and K.
  τ: Transforming variable relating t and T.

$f(x,\tau)$: Function of two variables related to P^{Am}.
κ: Transforming variable relating σ and r.
α: Constant relating functions f and u.
β: Constant relating functions f and u.
$g(x,\tau)$: Transformed payoff function.

- **Chapter Four**

M: Number of intervals on the τ-axis.
m: Indexing on the τ-axis, where $m = 0,\ldots,M$.
$\delta\tau$: Interval length on the τ-axis.
x_{max}: Maximum value on the x-axis.
x_{min}: Minimum value on the x-axis.
N: Number of intervals on the x-axis.
n: Indexing on the x-axis, where $n = 0,\ldots,N$.
δx: Interval length on the x-axis.
$v(x,\tau)$: Approximation of the true solution $u(x,\tau)$.
α: Relationship between die interval lengths, $\delta\tau$ and δx.
θ: Variable that can be manipulated to select either the Explicit ($\theta = 0$), Implicit ($\theta = 1$) or Crank-Nicolson ($\theta = \frac{1}{2}$) finite difference methods.
ω: Relaxation parameter of the SOR and the PSOR iterative methods.

- **Chapter Five**

W_1: Wiener process related to the stock in the asset price model.
W_2: Wiener process related to the variance in the asset price model.
ρ: Correlation coefficient between W_1 and W_2.
γ: Volatility of volatility.
β: Long term variance.
α: Rate of mean reversion.
ϑ: Market price of risk.
Lu: Heston operator.
$u(x,y,\tau)$: Function of three variables used to solve option pricing problem under stochastic volatility.
x: Underlying stock price.
x_{max}: Maximum stock price.
x_{min}: Minimum stock price.
y: Variance.
y_{max}: Maximum variance.
m: Number of internal nodes on the x-axis.
Δx: Interval length on the x-axis.
i: Indexing on the x-axis, where $i = 0,\ldots,m+1$.
n: Number of internal nodes on the y-axis.

Δx: Interval length on the y-axis.
j: Indexing on the y-axis, where $j = 0, \ldots, n+1$.
l: Number of internal nodes on the τ-axis.
$\Delta\tau$: Interval length on the τ-axis.
k: Indexing on the τ-axis, where $k = 0, \ldots, l+1$.
a_{add}: Additional constant introduced to ensure coefficient matrix in diagonally dominant.
c_{add}: Additional constant introduced to ensure coefficient matrix in diagonally dominant.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Examining the global options and futures industry, one finds that this industry grew by $11,4\%$ in 2011 to an amount of $24,97$ billion contracts traded per year. This is slightly down on past years' numbers. Looking at the broader picture, the industry has grown by $60,9\%$ in the past 5 years. Emerging markets such as China, Brazil, India and Russia, who seem to have been only slightly affected by the downturns of 2008 and 2009, were the main contributors to this growth number (Acworth 2012, 24).

In a study conducted by the Futures Industry Association, a sample from 81 world wide exchanges revealed that the Asia-Pacific region boasted the greatest growth number in 2011, impressing with 39%. This region was followed by North America with a rise of 33% and Europe with 20% growth for 2011 (Acworth 2012, 24).

Of the total $11,4\%$, the options market grew by $15,9\%$ in 2011 compared to futures, which only grew by $7,4\%$ (Acworth 2012, 24). It is also well-known that most options traded on international exchanges and over the counter are American style. These include options on stocks, stock indexes, interest rates, foreign currencies, energy and commodities (Feng, Linesky, Morales & Nocedal 2011, 814).

Traditionally, option prices are computed using the Black-Scholes model. This model makes various assumptions about financial markets and subsequently it has shortcomings. One such an assumption is that the volatility of the underlying asset is a constant value. Since volatility is not an observable parameter, this makes the use of this model even more obsolete (Kau 2009, ii). Therefore, option pricing models that take stochastic volatility into account produce more realistic solutions that reflect current market data. **This dissertation aims to address the issue of accurately portraying market indicators by pricing American style put options using the Heston stochastic volatility model.**

However, the process of pricing an American put option using the Black-Scholes model in itself is a challenging task. This can be attributed to the early exercise facility offered by the American put as it raises the question: "Well, when is it optimal to exercise this option?" Due to this problem's complexity, no generic closed form solution exists and therefore various numerical methods are used to obtain the option's price. So, before one can even wish to solve the pricing problem under stochastic volatility, it is vital to first understand the process of pricing an option using a constant volatility model.

This dissertation is divided into two sections: the first section is devoted to solving the American put problem using the traditional Black-Scholes model. The second section covers the Heston stochastic volatility model and applies it to the pricing of these options. There are numerous methods that can be used to solve the pricing problem. However, this study focusses on the Crank-Nicolson implicit finite difference method which is used in conjunction with the Projected Successive Over Relaxation iterative method (PSOR). The aim is to develop a comprehensive algorithm that prices American put options under both constant and stochastic volatility by incorporating both the Crank-Nicolson implicit finite difference method and the PSOR method. One motivation for this aim is that algorithms are very compact for constant volatility models (Seydel 2009, 175) and no such algorithms seem to be available for stochastic volatility models.

SECTION I: CONSTANT VOLATILITY MODEL: BLACK-SCHOLES MODEL

This section comprises three chapters. Chapter two introduces both the world of options and matrix algebra. In addition, it offers a brief history of options. This chapter aims to introduce the mathematician to the financial sphere and the reader with a background in finance to matrix algebra, which is vitally important when solving the pricing problem using the finite difference method.

As previously mentioned, American put options offer their own unique computational challenges. This is due to the *early exercise feature* of these options that requires the adaptation of the Black-Scholes partial differential equation into the *Black-Scholes partial differential inequality*. This adaptation of the Black-Scholes model enables one to define the option pricing problem as a *free boundary value problem*. After a formal discussion of the free boundary value problem, the pricing problem is finally presented as a *linear complimentary problem (LCP)*. Since no analytical method is available to price American put options, a numerical method is required to solve the linear complimentary problem. The different forms of the problem's formulation can be found in chapter three.

Chapter four introduces the reader to the *finite difference numerical method*, which is used to solve the linear complimentary option pricing problem. The chapter employs the iterative *projected successive over relaxation method (PSOR)* to solve the tridiagonal system, subsequently obtained from applying the finite difference method. In addition, the chapter provides an algorithm of the whole numerical procedure, and the interested reader can find the MATLAB code in the appendix.

Additional topics that can be found in chapter four include:

- A discussion of the explicit, implicit and Crank-Nicolson implicit finite difference methods.
- The various methods, both direct and iterative, that can be used to solve tridiagonal systems of equations.
- The convergence of the iterative methods: Jacobi, Gauss-Seidel and Successive Over Relaxation (SOR).

SECTION II: STOCHASTIC VOLATILITY MODEL: HESTON MODEL

The variables of the Heston stochastic differential equations are time (t), underlying asset value (S) and variance (y). Due to the additional variable, the Heston model has an additional spacial dimension. This complicates the discretization and solution procedures and makes the numerical method computationally more expensive.

Chapter five starts with a formal definition of the *Heston operator*. It states this partial differential equation's initial and boundary conditions and this enables researchers to formulate the pricing problem under stochastic volatility in its *linear complimentary problem (LCP)* form. The remainder of chapter five is devoted to the discretization of the Heston operator with an in-depth discussion of the matrices that result from applying the finite difference method. The MATLAB code for this application can also be found in the appendix.

The goal of the dissertation is to offer a clear understanding of the American put option pricing problem solved with the use of the Heston stochastic volatility model. This will enable readers to approach more complicated pricing models with bold confidence in the future.

Chapter 2

Background Theory

This chapter addresses some of the background theory that will be used in the remainder of the dissertation. It aims to help two types of readers, those from the field of mathematics, unfamiliar to the world of finance, and those from the financial industry, unfamiliar with the intimidating topic of matrix algebra, by introducing some basic topics from both fields.

2.1 The history of options

A study of the history of options reveals that these financial instruments are not modern inventions, as is generally assumed. Their origin can be traced back to ancient Greece, where a fifth century B.C. philosopher, Thales of Miletus, engaged in trading to prove to society that if philosophers wanted to be rich, they could be. In doing so he aimed to address the eternal question, **"If you are so smart, why aren't you rich?"**

Thales noticed that Miletus' seasonal olive crop yielded good returns in favourable weather conditions and therefore he decided to put a deposit on all the olive presses in the region. During the harvest season, the demand for olive presses grew exponentially due to the exceptional yield of the olive crop and the fact that olives were not a storable resource. Thales subsequently sublet the olive presses and by doing so, made a substantial profit.

Thales created an option on the olive crop. If the crop had failed, he would merely loose his deposit. However, in the event of a successful crop, he would reap the rewards by paying the initial premium and then making a seemingly limitless profit (Forsyth 2008, 3)

Options where, however, officially only traded on an exchange on 26th April 1973. The Chicago Board Options Exchange (CBOE) was the first to create standardized, listed options. Back then, there were only calls on 16 stocks with puts being introduced in 1977. Today, options are traded on over 50 exchanges worldwide (Wilmott 2000a, 21). Many banks and other financial institutions trade them over the counter (OTC) (Hull 2000, 5).

It is interesting to note that after the 2008 financial crisis, it became clear that the growth of the over the counter markets and their severe complexity outstripped the financial industry's capacity to manage them. The value of assets traded became difficult to assess and banks lost confidence in each other. The events of 2008 helped to highlight the advantages of regulated exchanges. The Dodd-Frank bill signed in June 2010, aimed at putting regulatory measures in place for the American financial system- supports the idea of exchange trading and more standard over the counter options (DuFour 2011, 11).

2.2 Basic option theory

2.2.1 What is an option

Risk is a core component of all financial investments made. The management of risk is a highly specialized field with analysts constantly identifying, measuring and managing the risks involved in investment. A sure way to minimize risk, is to take out insurance against it. This is the premise on which derivatives were created. Derivatives offer a certain level of insurance against financial loss (Chance 2003, 1).

An options is a type of financial derivative that represents a legal contract between two parties. It is sold by one party (the option writer) and purchased by the other (the option holder). There are two main types of options. A *call option* offers its holder the right, but not the obligation, to *buy* a specific underlying asset for an agreed upon amount at a specified time in the future. On the other hand, a *put option* offers its holder the right, but not the obligation, to *sell* a specific underlying asset for an agreed upon amount at a specified time in the future (Wilmott 2000a, 22).

The two types of options encountered in this dissertation are American options and European options. The difference between the two is that American options can be exercised at any time spanning the commencement date to the date of maturity whereas their European counterparts, can only be exercised at the date of maturity (Hull 2000, 6).

Defining an option in more detail (notice the notation used) one finds:

- At time $t = 0$
  The buyer purchases the option at an option premium, X. **This option premium or price is what is calculated in later chapters**. The value of the underlying asset at time, $t = 0$, is S_0. The option contract is only binding for a specific period of time and spans the interval $0 \leq t \leq T$. Where time $t = T$ is known as the exercise date. The option offers it's holder the choice to buy or sell the underlying asset contained within the option contract at a predetermined price, the strike price K, *regardless of the actual price of the underlying asset*. If the option holder buys or sells the underlying asset, he is said to exercise the option.

- Time $0 < t < T$
  If the holder purchased an American style option, he can choose to exercise this option at any time on or before the exercise date, T. Therefore, American options can be exercised prematurely (Hull 2000, 6).

- At time $t = T$
  Both the American and the European option holder can choose to exercise the option at time T (Hull 2000, 6). Remember that if the holder wishes to do so, the underlying asset is either purchased or sold at the predetermined price (the strike price, K), and not at the asset price at maturity, S_T. Therefore, the option holder is protected from fluctuations in the asset price. If the option holder does not wish to exercise his option, he merely looses his initial premium, X.

2.2.2 Option payoffs

The option payoff depends on the type of option held and the position taken on the option. Each option contract has two potential positions:

1. **The long position**
   This position is taken by a client/investor who chooses to **buy** an option, becoming an **option holder**. He can purchase either a *put* or a *call* option, paying the option price, X (Hull 2000, 8).

2. **The short position**
   This position is taken by a client/investor who chooses to **sell or write** the option, becoming an **option writer**. He too can sell either a *put* or a *call* option. By now becoming an option writer, he receives cash upfront (the option buyer's option premium X), but faces potential liabilities in the future (Hull 2000, 8).

One can therefore define four different option positions:

- A **long** position in a **call** option.
- A **long** position in a **put** option.
- A **short** position in a **call** option.
- A **short** position in a **put** option.

Examining **European options** that can only be exercised at maturity, T, one finds the following possible scenarios. Remember that S_T is the underlying stock price at maturity T and K is the agreed upon strike price. Also note that the payoff function does not take the option premium into account, and therefore does not reflect profit.

(a) A **long** position in a **call** option.

The payoff is calculated using (Hull 2000, 9):

$$\text{Payoff} = max[0, S_T - K] \tag{2.2.1}$$

This can be seen in Fig. 2.2.1, where the payoff is positive if $S_T > K$.

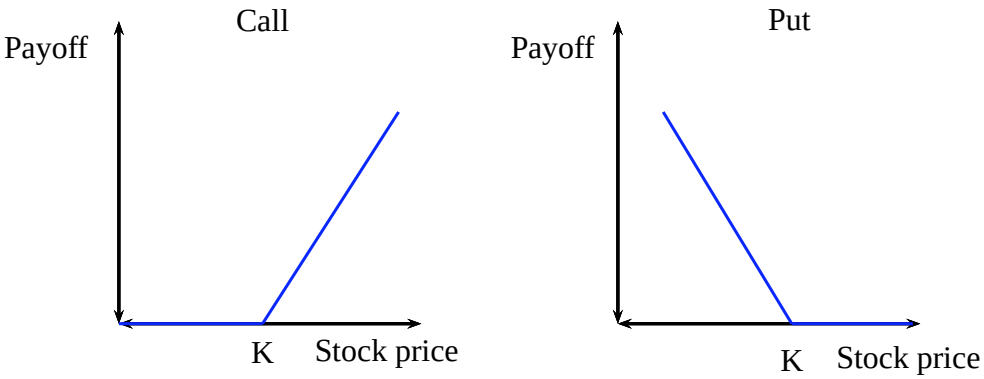

Figure 2.2.1: Long position on option

Therefore, at time $t = 0$ the holder will buy a call option with a strike price, K, in the hope that the unknown future asset price S_T will rise, thereby making a profit. A further increase in the future asset price will result in an increase in the option's payoff. At the exercise date, $t = T$, if $S_T > K$, the holder can exercise the call option and buy the underlying asset for a much lower price than it's actual value S_T. One finds that if:

- $S_T > K$: Payoff is positive and the option is said to be **in-the-money**.
- $S_T < K$: Payoff is zero and the option is said to be **out-of-the-money**.
- $S_T = K$: Payoff is zero and the option is said to be **at-the-money**.

(b) A **long** position in a **put** option

The payoff is calculated using (Hull 2000, 9):

$$\text{Payoff} = max[0, K - S_T] \tag{2.2.2}$$

This can be seen in Fig. 2.2.1, where the payoff is positive if $(K) > (S_T)$.

At time $t = 0$, the holder will buy a put option with strike price, K, in the hope that the unknown future underlying asset price, S_T, will fall and be less than the predetermined strike price. At the exercise date, $t = T$, if $K > S_T$, the holder can sell the underlying asset at the strike price K, and make a profit. In the case of a put option, if:

- $K > S_T$: Payoff is positive and the option is said to be **in-the-money**.
- $K < S_T$: Payoff is zero and the option is said to be **out-of-the-money**.
- $K = S_T$: Payoff is zero and the option is said to be **at-the-money**.

(c) A **short** position in a **call** option

The payoff is calculated using (Hull 2000, 9):

$$\text{Payoff} = min[0, K - S_T] \tag{2.2.3}$$

At time $t = 0$, the option writer receives an option premium, X, and has no say in whether an option can be exercised or not. The call option writer hopes that the unknown future asset price, S_T, will drop below the agreed upon strike price, K.

At time $t = T$, the writer of the call option is obligated to sell the underlying asset to the option holder. If the *underlying asset price rises* above the strike price and the option holder chooses to exercise his option, the option writer must sell the underlying asset to the holder at the strike price, K, losing $S_T - K$. The option writer now only has the option premium, X, left after trading. If the *underlying asset price drops* below the strike price, the option holder will not exercise the option and the option writer will have both the underlying asset and the option premium left after trading.

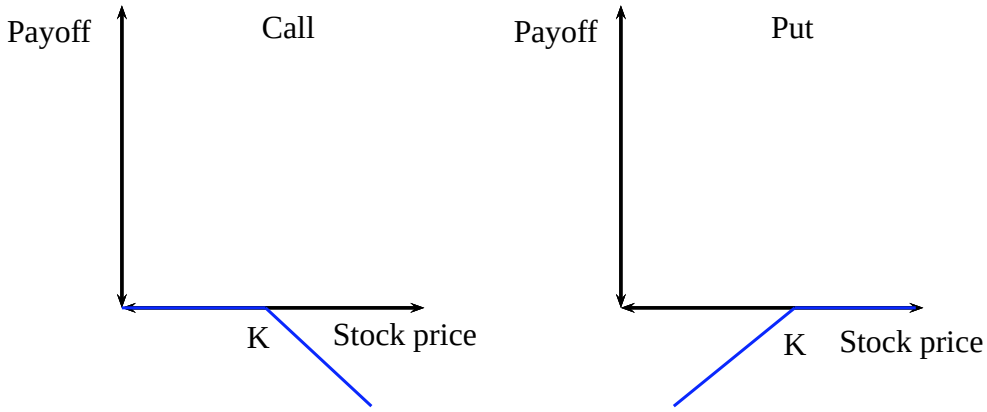

Figure 2.2.2: Short position on option

(d) A **short** position in a **put** option

The payoff is calculated using:

$$\text{Payoff} = min[0, S_T - K] \tag{2.2.4}$$

The option writer hopes that the uncertain future asset price, S_T, will rise above the strike price, K. At time $t = T$, if the *underlying asset price goes down* and $S_T < K$, the option holder will choose to exercise the option and the writer would be obliged to supply the underlying asset. Therefore, the option writer will only have the premium left after trading. If the *underlying asset price goes up* and $S_T > K$, the option holder will not exercise the option and the writer will be left with both the option premium and the underlying asset after trading.

2.2.3 Itô's Lemma

Before one can introduce the reader to the model that forms the foundation of all modern finance, the Black-Scholes model, one first has to cover some background theory in the form of Itô's Lemma. Although a detailed analysis of the complex topic of stochastic calculus falls beyond the scope of this dissertation, the most important rule of stochastic calculus deserves some attention (Wilmott 2000b, 71). The following is therefore a broad discussion that aims to introduce some of the tools that will later be used to derive the Black-Scholes partial differential equation.

An option pricing model needs a basis model that describes the movement of the underlying asset's price. Since predicting the future price of an asset is impossible, one can

use either past or current market data to calculate various properties, such as the mean and the variance of the underlying asset (Wilmott, Dewynne & Howison 1996, 18).

A simple asset price model assumes two properties:

1. An asset's past history is completely reflected in its current price and the current price does not hold any further information.
2. Markets immediately respond to new data.

These properties allows one to define a specific type of stochastic process, the Markov process. A Markov process only considers the current value of an asset and therefore disregards the asset's past values and the means by which its current value was obtained. Thus, the only information relevant in predicting an asset's future value is its current value (Hull 2012, 280).

Therefore, the changes in an asset price is a Markov process and these changes are measured as returns.

Let the price of an underlying asset at time t, be S. One now wants to investigate the change in the asset price after a small time interval, dt, where the price is defined as $S + dS$. A return is defined as the change in asset price divided by the original asset price (Wilmott et al. 1996, 19-20):

$$\frac{dS}{S}. \tag{2.2.5}$$

To model the return on this asset, one can divide the return into two sections:

1. A predictable deterministic part

   This will be equivalent to the return received from investing money in a bank at a risk-free interest rate. Mathematically, it is written as:

   $$\mu dt, \tag{2.2.6}$$

   where μ is the average rate of growth of the underlying asset and is known as the drift. The Black-Scholes model assumes this value to be constant (Wilmott et al. 1996, 20).

2. An unpredictable random change to the asset price

   This is due to external factors and is represented by a random sample that is drawn from a normal distribution. Mathematically, it is written as:

$$\sigma dW, \tag{2.2.7}$$

   where σ is the standard deviation of the underlying asset and is known as the volatility. The Black-Scholes model assumes this value to be constant (Wilmott et al. 1996, 20) and it is this assumption that is addressed in this dissertation.

Adding these two contributions to obtain the return of an asset leads to the following stochastic differential equation:

$$\frac{dS}{S} = \mu dt + \sigma dW. \tag{2.2.8}$$

This equation is used to generate different asset prices (Wilmott et al. 1996, 20) and will be used in the next chapter, which addresses the derivation of the Black-Scholes partial differential equation.

The term dW contains the uncertain randomness of asset prices and is known as a Wiener process (Wilmott et al. 1996, 20). The Wiener process has the following properties (Wilmott et al. 1996, 21):

- dW is a random variable.
- dW is taken from a standard normal distribution.

Equation 2.2.8 is known as a random walk and cannot be solved. However, it supplies information regarding the behavior of the asset price in a probabilistic sense by generating different time series (Wilmott et al. 1996, 23).

In reality, prices are quoted at discrete time intervals and therefore there is a lower limit to the time increment encountered in 2.2.8. In the process of pricing options, these discrete time intervals would lead to vast amounts of data and subsequently one uses continuous time increments, where $dt \to 0$, instead (Wilmott et al. 1996, 25). Itô's Lemma is to stochastic variables what Taylor's theorem is to deterministic variables (Wilmott 2009, 106) and relates a change in the function of a random variable, to a change in the random variable itself (Wilmott et al. 1996, 25).

The following derivation of Itô's Lemma is taken from Paul Willmott's book *The Mathematics of Financial Derivatives* (Wilmott et al. 1996, 26-27).

Itô's Lemma can be derived for a function of both one and two random variables, where the lemma for a function of two random variables is used in the following chapter to derive the Black-Scholes model, where an option price, V, is typically a function of both the underlying asset price, S and time, t.

- Itô's Lemma for a function of one random variable, $V(S)$

  Approximating the interval $V(S+dS)$ by a Taylor series, one finds:

$$V(S+dS) = dV = \frac{dV}{dS}dS + \frac{1}{2}\frac{d^2V}{dS^2}dS^2 + \ldots. \tag{2.2.9}$$

  Notice that 2.2.9 contains a term with the square of 2.2.8. One therefore defines:

$$\begin{aligned} dS^2 &= (\mu S dt + \sigma S dW)^2, & (2.2.10)\\ &= \sigma^2 S^2 dW^2 + 2\sigma\mu S^2 dt dW + \mu^2 S^2 dt^2. & (2.2.11) \end{aligned}$$

  The following properties of the Wiener process are now used (Bjork 2009, 52):

$$(dW)^2 = dt, \tag{2.2.12}$$

$$(dt)^2 = 0, \tag{2.2.13}$$

  and

$$dt.dW = 0. \tag{2.2.14}$$

  Applying equations 2.2.12, 2.2.13 and 2.2.14, equation 2.2.11 can now be written as:

$$dS^2 = \sigma^2 S^2 dt + \text{ higher order terms of } dt. \tag{2.2.15}$$

  Substituting 2.2.15 into 2.2.9 and truncating, one finds:

$$V(S+dS) = dV = \frac{dV}{dS}(\mu S dt + \sigma S dW) + \frac{1}{2}\frac{d^2V}{dS^2}\sigma^2 S^2 dt, \tag{2.2.16}$$

which, after some simplification, can be written as:

$$V(S+dS) = dV = \sigma S\frac{dV}{dS}dW + \left(\mu S\frac{dV}{dS} + \frac{1}{2}\sigma^2 S^2\frac{d^2V}{dS^2}\right)dt. \tag{2.2.17}$$

This is Itô's Lemma written for a function in one variable.

- Itô's Lemma for a function of two random variables, $V(S,t)$

  In the case of a function with two variables, if random variable S changes by a small amount, dS in a time interval, $[t,t+dt]$, one writes the Taylor series expansion as follows:

$$V(S+dS,t+dt) = dV = \frac{\partial V}{\partial S}dS + \frac{\partial V}{\partial t}dt + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}dS^2 + \ldots. \tag{2.2.18}$$

  Substituting equations 2.2.15 and 2.2.8 into 2.2.18 and truncating, one finds:

$$V(S+dS,t+dt) = dV = \frac{\partial V}{\partial S}(\mu S dt + \sigma S dW) + \frac{\partial V}{\partial t}dt + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 dt. \tag{2.2.19}$$

  Rearranging 2.2.19 one can finally write Itô's Lemma for a function of two random variables as:

$$V(S+dS,t+dt) = dV = \sigma S\frac{\partial V}{\partial S}dW + \left(\mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t}\right)dt. \tag{2.2.20}$$

2.2.4 The Black-Scholes Model

This model forms the foundation of modern finance and although it is formally derived and adapted to suit American put options in the next chapter, some important concepts are introduced in this section.

$$\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial V}{\partial S} - rV = 0. \tag{2.2.21}$$

The Black-Scholes formulae are used to analytically price **European put and call options**. The development of these formulae had a dramatic impact on both theoretical

and practical financial applications and in 1997, the Nobel Prize for Economics was awarded to Robert Merton and Myron Scholes for their work and its impact on option pricing. Unfortunately Fisher Black had passed away two years prior to the ceremony (Ugur 2008, 111).

An option's value is represented by a function that can be written as (Wilmott 2000a, 82):

$$V(S,t,\sigma,\mu,K,T,r), \tag{2.2.22}$$

where

- S - current underlying stock price.
- t - current time.
- σ - volatility of underlying asset.
- μ - drift of underlying asset.
- K - strike price of option.
- T - exercise date of option.
- r - current interest rate.

Factors such as current stock price, strike price, time to expiration, stock price volatility and the risk free interest rate, have an effect on the price of an option, as will become apparent in section 3.2 (Hull 2000, 168).

This model allows one to describe real markets in theory. Certain assumptions are made which include (Hull 2000, 245) (Merton 1976, 126):

- Underlying assets have a constant volatility.
- Non-dividend paying options.
- Stock price follows a geometric Brownian motion that produces a log-normal distribution for the stock price.
- No arbitrage is allowed.
- Risk free interest rate is constant.

- Unlimited short selling is allowed.
- No taxes or transaction costs.
- Underlying asset can be traded continuously and in infinitesimally small number units.

This dissertation attempts to expand the Black-Schoels model by addressing one of its assumptions, the fact that an underlying asset's volatility has to be constant. This is done in chapter five. For now, it is important to realize that due to the limitations of the Black-Scholes model and its assumptions, current research is focused on addressing these shortcomings by developing models to ultimately enable the pricing of complex derivatives.

2.2.5 American options

As mentioned earlier, the main difference between a European and American option, is that an American option offers it's holder the *early exercise facility*. This additional feature should not be worthless and as a result, one expects an American option to be more valuable than its European counterpart. This extra premium is known as the *early exercise premium* (Kwok 2008, 251).

This dissertation only deals with the pricing of an American put option on a single non-dividend paying underlying stock. This is due to the fact that it is never favourable to exercise this type of call option early and therefore the American call option's value is equal to the European call option and can be calculated using the traditional Black-Scholes formulae (Higham 2009, 174). This can be shown by considering the argument given in John C. Hull's book, *Options, Futures, and other Derivatives* (Hull 2000, 175-176).

An **American call option holder** who wants to know when it is most favourable to exercise his option, is faced with two possible scenarios at a certain time $t < T$:

- **If he wishes to hold the underlying stock for a period longer than the duration of the option contract**.
  - *If the option is out-of-the-money,* $(S_t < K)$.
    It is not optimal to exercise the option and the holder needs to hold on to the option.

– *If the option is in-the-money,* $(S_t > K)$.
It is not optimal to exercise the option early and the holder is better off exercising at the time of maturity, T. This statement is supported by considering the numerous advantages that not exercising the option at time t offers its holder:

1. Interest is earned on the strike price amount, K, for the duration of the contract.
2. No income is sacrificed in the case of a non-dividend paying stock option.
3. The holder is insured against a future stock price drop.

Hence there is no advantage in early exercise.

- **If he wishes to sell the underlying stock and feels that it is currently overpriced**.

  The holder now considers exercising the option and selling the underlying stock. In this case it is optimal to rather sell the option. If he were to exercise the option, he would obtain the option payoff

$$\text{Payoff} = max[0, S_t - K]. \tag{2.2.23}$$

  If he were to sell the option, the option would be bought by an individual who wishes to hold the underlying and keeping in mind that the lower boundary of a call option is given by (Higham 2009, 14):

$$C \geq max[S - Ke^{-rT}, 0], \tag{2.2.24}$$

  he would obtain a value larger than just the payoff value given in 2.2.23.

Therefore it is never optimal for the holder of an American call option to exercise prematurely.

In the case of American put options, the holder is still faced with the dilemma of finding the time when it is optimal to exercise the option and in the case of puts, things are significantly more complicated when compared to calls. Except for a few special cases, there are no analytical pricing formulas available for American options and therefore numerical methods are used to obtain the option price (Kwok 2008, 252).

This dissertation is devoted to one of these methods, the finite difference method and although it is covered in great detail in the chapters to follow, for now it is important to

understand why numerical methods are used.

In summary, due to an American put option's early exercise facility (the option to prematurely exercise the put), the problem of finding the optimal time at which to exercise can only be calculated numerically. This problem is known as a free boundary problem (Wilmott 2000a, 129). The price or premium of this put option also needs to be calculated numerically.

2.3 Basic Matrix Algebra theory

The following section broadly defines some of the terms concerning matrix algebra that the reader will encounter in this dissertation. A detailed description of these terms falls beyond the scope of this dissertation.

2.3.1 Non-singular

Before one can define a non-singular matrix, one first has to explain the concept of a matrix determinant. Let $\mathbf{A}$ be a $n \times n$ square matrix:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}.$$

The determinant of $\mathbf{A}$ is calculated by:

$$Det(A) = \sum_{k=0}^{n} a_{km}(-1)^{k+m} M_{km}, \tag{2.3.1}$$

for $1 \leq m \leq n$ and where M_{km} is the minor determinant of the $(n-1) \times (n-1)$ matrix (Yang, Cau, Chung & Morris 2005, 464).

One can now define a non-singular matrix as: a square matrix $\mathbf{A}$ is non-singular if its determinant is non-zero (Karris 2007, 4(22)):

$$Det(A) \neq 0. \tag{2.3.2}$$

2.3.2 Conjugate transpose

Start by defining the conjugate of a matrix. If a matrix **A** has complex elements and one replaces each element of **A** by its conjugate, the new matrix obtained is called the conjugate of **A** and is denoted by $\mathbf{A}^*$ (Karris 2007, 4(8)).

The transpose of a matrix $\mathbf{A}^\mathbf{T}$, is obtained when its rows and columns are interchanged (Karris 2007, 4(6)). Therefore, the conjugate transpose of a matrix $(\mathbf{A}^*)^\mathbf{T}$, is obtained when one transposes the conjugate matrix, $\mathbf{A}^*$.

2.3.3 Positive definite

A square matrix **A** is positive definite if:

$$(\mathbf{x}^*)^\mathbf{T}\mathbf{A}\mathbf{x} > 0, \tag{2.3.3}$$

for any non-zero column vector **x** (Yang et al. 2005, 468).

2.3.4 Hermitian

A square matrix **A** is called a Hermitian if (Karris 2007, 4(9)):

$$\mathbf{A} = (\mathbf{A}^*)^\mathbf{T}. \tag{2.3.4}$$

2.3.5 Spectral radius

If **A** is any matrix, then the eigenvalues of **A**, denoted by λ, are the roots of the characteristic equation of **A**:

$$Det(A - \lambda I) = 0, \tag{2.3.5}$$

where **I** is the identity matrix (Kincaid & Cheney 1991, 187).

The spectral radius of a matrix can now be defined as (Kincaid & Cheney 1991, 187):

$$\rho(A) = max\{\mid \lambda \mid : Det(A - \lambda I) = 0\}. \tag{2.3.6}$$

2.3.6 Diagonally dominant

If **A** is a matrix with elements $a_{i,j}$, then **A** is diagonally dominant if (Kincaid & Cheney 1991, 152):

$$\mid a_{ii} \mid > \sum_{j=1, j\neq i}^{n} \mid a_{ij} \mid, \quad 1 \leq i \leq n. \tag{2.3.7}$$

2.3.7 M-matrix properties

A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a M-matrix if it can be expressed as (Elhashash & Szyld 2008, 2436):

$$\mathbf{A} = s\mathbf{I} - \mathbf{B}, \tag{2.3.8}$$

where **B** is a non-negative matrix which has a spectral radius defined by (Elhashash & Szyld 2008, 2436):

$$\rho(B) \leq s \quad s \in \mathbb{R}. \tag{2.3.9}$$

2.3.8 Tridiagonal matrix

A matrix $\mathbf{A} = a_{ij}$ is tridiagonal if $a_{ij} = 0$ for all pairs (a_{ij}) that satisfy $\mid i - j \mid > 1$. Thus, in the i-th row, only the elements a_{ij-1}, a_{ij} and a_{ij+1} can be non-zero (Kincaid & Cheney 1991, 154). A tridiagonal matrix therefore has the form:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & \cdots & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1n} & a_{nn}. \end{pmatrix}.$$

Chapter four deals with tridiagonal matrices and the methods that can be used to obtain solutions of such a system, with the inclusion of both direct and iterative approaches.

Chapter 3

American put option pricing problem under constant volatility

This chapter addresses the topic of the **mathematical model for pricing American put options**. Section one discusses general *partial differential equations* (PDE's), with special attention paid to the *one-dimensional parabolic heat equation*. This heat equation is discussed in more detail because later on in this chapter, the well known Black-Scholes partial differential equation is transformed into the heat equation.

The *derivation of the Black-Scholes partial differential equation* (BS PDE) follows the section on partial differential equations and due to the early exercise facility offered by American options, the Black-Scholes partial differential equation is transformed to the *Black-Scholes inequality*. The American option pricing problem is then discussed as a *free boundary value problem*. To aid the understanding of the free boundary concept, a physical problem (the obstacle problem) is also discussed. Although far removed from the financial context, the obstacle problem serves as a tool that enables one to formally define the free boundary that exists due to the early exercise facility of an American option. In the latter sections of this chapter, the Black-Scholes partial differential equation is transformed into the one-dimensional parabolic heat equation and defined with it's initial and boundary conditions. Finally, the *linear complimentary problem* (LCP) is constructed, which forms the mathematical basis for all numerical techniques implemented in the chapters to follow.

3.1 Partial differential equations

In the process of solving the option pricing problem, the Black-Scholes partial differential equation can be transformed into the one-dimensional parabolic heat equation and therefore this section on partial differential equations is included. The heat equation that is obtained from the transformation is solved using specific initial and boundary values. The reader is now introduces to basic concepts regarding partial differential equations. Consider the one-dimensional parabolic heat equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \tag{3.1.1}$$

on the domain (Oliver 2004, 19):

$$\mathscr{D} = \{(x,t) : x \in \mathbb{R}, t \geq 0\}. \tag{3.1.2}$$

In a financial setting, most partial differential equations are either first or second order parabolic equations (Wilmott, Dewynne & Howison 2000, 75). The general heat equation given in 3.1.1 is a **homogeneous, one-dimensional, second order, linear, forward parabolic equation** (Wilmott et al. 2000, 80). This equation has served as a model for the flow of heat in a continuous medium for nearly two centuries and it is one of the most widely used models in the field of applied mathematics (Wilmott et al. 2000, 79). From a physical point of view, it describes the process of heat diffusion (a "smoothing-out" process), in which heat flows from a hot to a cooler area, cancelling out temperature differences along a heat conducting material of length L, over a certain time period, T (Wilmott et al. 2000, 81). No further details regarding the physical meaning of the heat equation in a thermodynamic setting will be discussed, since this dissertation is only concerned with the application of the heat equation in a financial setting.

The following section discusses the properties of equation 3.1.1. Begin by defining a function $u(x,t)$, that is dependant on variables x (position) and t (time).

- *Homogeneous*: When equation 3.1.1 is manipulated and equated to zero,

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0. \tag{3.1.3}$$

- *One-dimensional*: This indicates that heat can only be transferred in one direction. For equation 3.1.3, this direction is along the x-axis.
- *Second order*: The highest order derivative present in equation 3.1.3.

- *Linear*: Any linear combination of two solutions of the heat equation on an open interval I, is also a solution of the heat equation on I. Therefore, sums and constant multiples are also solutions (Kreyszig 2006, 47). For example, if u_1 and u_2 are both solutions to equation 3.1.3, then $c_1u_1 + c_2u_2$ is also a solution. Here c_1 and c_2 are any constants (Wilmott et al. 2000, 80).

- *Forward*: If the signs of the terms are the same when they are on the same side of the equation, then the equation is known as a backward parabolic equation and requires final conditions. Equation 3.1.3 is known as a **forward parabolic equation** and requires **initial conditions** to be stipulated (Wilmott et al. 2000, 46).

- *Parabolic*: Consider the partial differential equation of the form,

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial xy} + C\frac{\partial^2 u}{\partial y^2} = F\left(x, y, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right).$$

  This equation is parabolic if its discriminant, $AC - B^2$, is equal to 0 (Kreyszig 2006, 551).

Since a partial differential equation can have multiple solutions, one needs to specify initial and boundary value conditions for the forward heat equation in 3.1.3, to ensure that a unique solution is obtained as opposed to a general solution (Wilmott et al. 2000, 45).

- **Initial condition**: This will specify the value of $u(x,0)$. Therefore, time remains constant and $u(x,t)$ changes according to x. From a thermodynamic perspective, this value represents the temperature of the material at any point x, before the process of heat conduction starts. As mentioned previously, the initial condition is specified if the heat equation is of the *forward* type. Mathematically, this condition is given as:

$$u(x,0) = g(x). \tag{3.1.4}$$

- **Final condition**: The final condition is specified instead of the initial condition if the heat equation is of the *backward* type. Here, the value of $u(x,T)$ also changes according to variable x:

$$u(x,T) = h(x). \tag{3.1.5}$$

- **Boundary conditions**: These values represent the temperatures of the two terminal points of the one-dimensional material that conducts heat. As mentioned earlier, the length of the material is measured along the x-axis and can be subdivided. The boundary conditions stipulate the value of function u at end points $x = 0$ and $x = L$, for different values in time:

$$u(0,t) = a(t), \qquad 0 \leq t \leq T, \tag{3.1.6}$$

$$u(L,t) = b(t), \qquad 0 \leq t \leq T. \tag{3.1.7}$$

If no analytical solution is available, a numerical technique can be implemented to estimate a solution to the heat equation. The finite difference numerical method, will be the main focus of this dissertation. Before focussing on the process of obtaining a numerical solution to the heat equation, the Black-Scholes partial differential equation is first investigated and adapted it to suit American put options. In the latter stages of this chapter finite difference numerical method to solve the heat equation will be addressed in detail.

3.2 Derivation of the Black-Scholes partial differential equation

To better understand the Black-Scholes model, the Black-Scholes partial differential equation is derived using two different techniques. The first method involves the application of Itô's Lemma, whereas the second method involves the construction of a replicating portfolio and follows a more intuitive approach. For further reading, the book, *Frequently asked questions in Quantitative Finance* by Paul Wilmott (Wilmott 2009, 401-426), has a whole chapter dedicated to the twelve different ways to derive the Black-Scholes equation.

Before deriving the Black-Scholes partial differential equation, consider the assumptions made by the Black-Scholes model, as mentioned in section 2.2.4 of the previous chapter.

3.2.1 Using Itô's Lemma

The following derivation of the Black-Scholes partial differential equation using Itô's Lemma, is based on the text found in *Option Pricing: Mathematical models and compu-*

tation (Wilmott et al. 2000, 41-45) and John Hull's *Options, Futures, and other Derivatives* (Hull 2009, 287-289). Assuming that the underlying stock price follows a geometric Brownian motion in continuous time, the stochastic differential equation (SDE) of the stock price can defined as:

$$dS = \mu S dt + \sigma S dW, \tag{3.2.1}$$

where:
S - Stock price at time t.
μ - Constant expected rate of return of stock (drift).
σ - Constant volatility of the stock price.
W - Wiener process.

Assume that $V = V(S,t)$ is the value of either a put or a call option. Then V is sufficiently smooth with continuous first and second order derivatives with respect to S and continuous first order derivatives with respect to t on the domain:

$$D_v = \{(S,t) : S \geq 0, 0 \leq t \leq T\}. \tag{3.2.2}$$

Applying Itô's Lemma defined on page 13, one finds an equation that represents the random walk followed by V:

$$dV = \left(\frac{\partial V}{\partial t} + \frac{\partial V}{\partial S}\mu S + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right) dt + \frac{\partial V}{\partial S}\sigma S dW. \tag{3.2.3}$$

Construct a portfolio consisting of the option and $(-\Delta)$ units of the underlying asset. The value of this portfolio is given by:

$$\Pi = V - \Delta S. \tag{3.2.4}$$

The change in the value of the portfolio after one time-step is:

$$d\Pi = dV - \Delta dS. \tag{3.2.5}$$

Therefore the amount of underlying stock, $(-\Delta)$, remains constant.

Substituting 3.2.1 and 3.2.3 into 3.2.5, one finds:

$$d\Pi = \left(\frac{\partial V}{\partial t} + \frac{\partial V}{\partial S}\mu S + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right) dt + \frac{\partial V}{\partial S}\sigma S dW - \Delta(\mu S dt + \sigma S dW). \tag{3.2.6}$$

This simplifies to:

$$d\Pi = \sigma S\left(\frac{\partial V}{\partial S} - \Delta\right) dW + \left(\mu S\frac{\partial V}{\partial S} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 + \frac{\partial V}{\partial t} - \mu\Delta S\right) dt. \tag{3.2.7}$$

Noting that fluctuations caused by increments of the Wiener process have a coefficient

$$\left(\frac{\partial V}{\partial S} - \Delta\right), \tag{3.2.8}$$

one can remove the randomness component of the random walk by choosing

$$\frac{\partial V}{\partial S} = \Delta. \tag{3.2.9}$$

One now finds a totally deterministic portfolio (Wilmott et al. 2000, 43):

$$d\Pi = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right) dt. \tag{3.2.10}$$

Also notice that the drift rate μ, has been cancelled out. The remaining σ reflects the stochastic behaviour of the Black-Scholes equation and is assumed to be constant.

If an amount of Π is invested in a a risk-less asset with constant interest rate r, the capital growth on this amount would be $r\Pi dt$ after a time dt. With the assumption of no transaction costs and applying the concept of no arbitrage, one finds that:

$$r\Pi dt = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right) dt. \tag{3.2.11}$$

Substituting equations 3.2.4 and 3.2.9 into 3.2.11 and simplifying, one obtains the **Black-Scholes partial differential equation** (Wilmott et al. 2000, 44):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial V}{\partial S} - rV = 0. \tag{3.2.12}$$

3.2.2 Using a replicating portfolio

This discussion is based on a derivation given in *An Introduction to Financial Option Valuation: Mathematics, Stochastics and Computation* by Desmond J. Higham (Higham

2009, 74-79), where the author uses the concept of a replicating portfolio. Start by defining two different time increments. Assume the lifetime of the option contract to span the interval $0 \leq t \leq T$ and divide this option lifetime into large, equally spaced time increments, Δt. From Fig. 3.2.1 follows that Δt denotes the time-step over interval $[t, t+\Delta t]$. If one divides each large Δt increment into L smaller δt increments, the interval $[t_0, t_1 \ldots t_i, t_{i+1} \ldots t_L]$ is obtained, where $t_0 = t$ and $t_L = t + \Delta t$. Therefore, there are $L+1$ points and L intervals, as illustrated in Fig. 3.2.2.

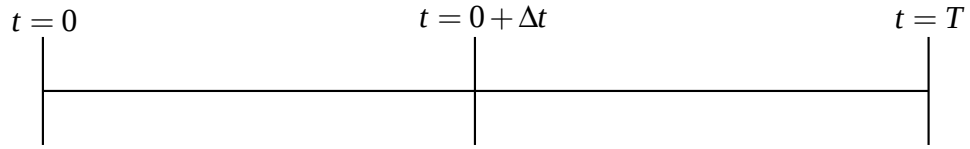

Figure 3.2.1: Put option duration divided into larger time increments, Δt

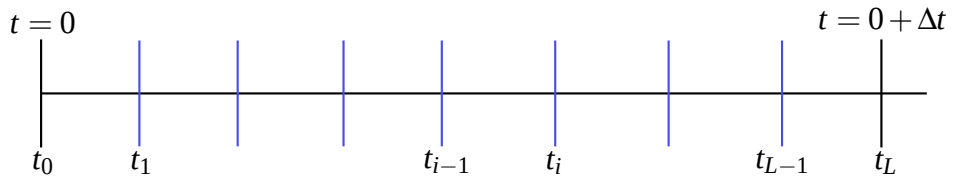

Figure 3.2.2: Put option duration divided into smaller time increments, δt

One can define δt as:

$$\delta t = t_{i+1} - t_i. \tag{3.2.13}$$

To derive the Black-Scholes partial differential equation, begin by constructing a replicating portfolio consisting of the asset underlying the option, S, and a certain amount of cash, D. This portfolio has exactly the same risk as the option at all times. Let $A(S,t)$ represent the number of units of underlying asset, S, where $A(S,t)$ is a function of both the asset price, S and time, t and let the amount of cash, $D(S,t)$ also be a function of both S and t.

The value of the replicating portfolio can now be defined as:

$$\Pi(S,t) = A(S,t)S + D(S,t). \tag{3.2.14}$$

Keeping equation 3.2.13 and the fact that the stock price, S, is also a function of time in mind, one can now define the following increments:

$$\delta S_i = S(t_{i+1}) - S(t_i), \tag{3.2.15}$$

$$\delta V_i = V(S(t_{i+1}), t_{i+1}) - V(S(t_i), t_i), \tag{3.2.16}$$

$$\delta \Pi_i = \Pi(S(t_{i+1}), t_{i+1}) - \Pi(S(t_i), t_i), \tag{3.2.17}$$

$$\delta (V - \Pi)_i = \delta V_i - \delta \Pi_i, \tag{3.2.18}$$

where function $V(S,t)$ represents the option value for any asset price, $S \geq 0$, at any time $0 \leq t \leq T$. As time varies, the amount of the underlying asset, $A(S,t)$, remains constant. This implies that a change in the value of the portfolio, $\delta\Pi$, has two origins:

- Asset price fluctuation, δS_i.
- Interest gained from the cash invested for time period δt, $rD_i\delta t$.

One can now define the portfolio value after time δt, as:

$$\delta \Pi_i = A_i \delta S_i + rD_i \delta t. \tag{3.2.19}$$

Since $V(S,t)$ is assumed to be a smooth function of both S and t, the Taylor series expansion of V is given by:

$$\delta V_i \approx \frac{\partial V_i}{\partial t}\delta t + \frac{\partial V_i}{\partial S}\delta S_i + \frac{1}{2}\frac{\partial^2 V_i}{\partial S^2}\delta S_i^2. \tag{3.2.20}$$

Subtracting 3.2.19 from 3.2.20 one finds:

$$\delta (V - \Pi)_i \approx \left(\frac{\partial V_i}{\partial t} - rD_i\right)\delta t + \left(\frac{\partial V_i}{\partial S} - A_i\right)\delta S_i + \frac{1}{2}\frac{\partial^2 V_i}{\partial S^2}\delta S_i^2. \tag{3.2.21}$$

This equation **compares the change in portfolio value to the change in option value**.

In the process of replicating the option, one needs the difference between the portfolio and option values to be predictable and therefore the unpredictable term, δS_i, must be eliminated. This is done by equating it's coefficient to zero:

$$\frac{\partial V_i}{\partial S} - A_i = 0. \tag{3.2.22}$$

Or

$$\frac{\partial V_i}{\partial S} = A_i. \tag{3.2.23}$$

This is similar to the strategy followed in the derivation using Itô's Lemma. Substituting 3.2.23 into 3.2.21 one obtains:

$$\delta(V - \Pi)_i \approx \left(\frac{\partial V_i}{\partial t} - rD_i\right)\delta t + \frac{1}{2}\frac{\partial^2 V_i}{\partial S^2}\delta S_i^2. \tag{3.2.24}$$

The goal is to ultimately remove all the random elements from the value comparison equation in 3.2.24. This is done by adding all these increments over the interval $0 \leq i \leq L-1$. The summation of 3.2.24 yields:

$$\Delta(V - \Pi) \approx \sum_{i=0}^{L-1}\left(\frac{\partial V_i}{\partial t} - rD_i\right)\delta t + \frac{1}{2}\sum_{i=0}^{L-1}\frac{\partial^2 V_i}{\partial S^2}\delta S_i^2. \tag{3.2.25}$$

Using the fact that $L\delta t = \Delta t$, one can now re-write 3.2.25 as:

$$\Delta(V - \Pi) \approx \left(\frac{\partial V}{\partial t} - rD\right)\Delta t + \frac{1}{2}\sum_{i=0}^{L-1}\frac{\partial^2 V_i}{\partial S^2}\delta S_i^2. \tag{3.2.26}$$

It can also be shown that the sum of the δS_i^2 terms is non-random.

$$\sum_{i=0}^{L-1}\delta S_i^2 \approx S(t)^2\sigma^2\Delta t. \tag{3.2.27}$$

Substituting 3.2.27 and assuming that all approximations are exact in the limit $\delta t \to 0$, 3.2.26 can be re-written as:

$$\Delta(V-\Pi) = \left(\frac{\partial V}{\partial t} - rD + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right)\Delta t, \tag{3.2.28}$$

where $\Delta(V-\Pi)$ denotes the change in $(V-\Pi)$ over time interval $[t, t+\Delta t]$. This can also be written as:

$$\Delta(V-\Pi) = V(S(t+\Delta t), t+\Delta t) - \Pi(S(t+\Delta t), t+\Delta t) - [V(S(t),t) - \Pi(S(t),t)]. \tag{3.2.29}$$

Assuming the no arbitrage principle, the change in portfolio $(V-\Pi)$ must equal the growth offered by the risk free interest rate and therefore the following holds:

$$\Delta(V-\Pi) = r\Delta t(V-\Pi). \tag{3.2.30}$$

Finally, by combining equations 3.2.14, 3.2.28 and 3.2.30, one obtains:

$$\frac{\partial V}{\partial t} - rD + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = r(V - AS - D). \tag{3.2.31}$$

Substituting A, from equation 3.2.23 into 3.2.31, one obtains the **Black-Scholes partial differential equation** (BS PDE):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial V}{\partial S} - rV = 0. \tag{3.2.32}$$

This PDE is satisfied for any option whose value can be expressed as some smooth function $V(S,t)$. As mentioned previously, this equation can now be transformed into the heat equation. The price of both **European put and call options** can be obtained analytically by using the appropriate initial and boundary conditions (Wilmott et al. 2000, 98).

- The **call option value** at time t, as derived in (Wilmott et al. 2000, 97-100) is given by:

  $$c(S,t) = SN(d1) - Ke^{-r(T-t)}N(d2), \tag{3.2.33}$$

  where

  $$d1 = \frac{log(S/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{(T-t)}}, \tag{3.2.34}$$

and

$$d2 = \frac{log(S/K) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{(T-t)}}. \tag{3.2.35}$$

$N(\cdot)$ is the distribution function for a standardized normal random variable (Wilmott et al. 1996, 48):

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}y^2} dy. \tag{3.2.36}$$

The final condition is the known payoff at this time (Wilmott et al. 1996, 46):

$$c(S,T) = max[S-K, 0]. \tag{3.2.37}$$

The boundary conditions are given by (Wilmott et al. 1996, 46):

$$c(0,t) = 0, \qquad S = 0, \tag{3.2.38}$$

$$c(S,t) = S, \qquad S \to \infty. \tag{3.2.39}$$

- The **put option value** is given by (Wilmott et al. 1996, 48):

$$p(S,t) = Ke^{-r(T-t)}N(-d2) - SN(-d1), \tag{3.2.40}$$

using equations 3.2.34 and 3.2.35 for $d1$ and $d2$ respectively.

The final condition is the known payoff at this time (Wilmott et al. 1996, 46):

$$p(S,T) = max[K-S, 0]. \tag{3.2.41}$$

The boundary conditions are given by (Wilmott et al. 1996, 47):

$$p(0,t) = Ke^{-r(T-t)}, \qquad S = 0, \tag{3.2.42}$$

$$P(S,t) = 0, \qquad S \to \infty. \tag{3.2.43}$$

3.3 Derivation of the Black-Scholes inequality for American put options

In the following section, the Black-Scholes partial differential equation is adapted to suit American put options. This is done to accommodate the early exercise facility of American options. Due to their greater flexibility, these options can potentially be more valuable than their European counterparts (Wilmott et al. 1996, 106). From the above derivation, equation 3.2.28 was obtained. Its notation is now altered to suit American put options by letting P^{Am} denote the value of an American put option. Therefore 3.2.28 can is now written as:

$$\Delta(P^{Am} - \Pi) = \left(\frac{\partial P^{Am}}{\partial t} - rD + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P^{Am}}{\partial S^2}\right)\Delta t. \tag{3.3.1}$$

Accepting the previous assumptions made, that due to the elimination of the random elements this comparison value, $\Delta(P^{Am} - \Pi)$, must equal the growth offered by a risk free investment at a fixed interest rate r, the following two scenarios are now examined:

1. $\Delta[P^{Am} - \Pi] > r\Delta t[P^{Am} - \Pi]$.

   This indicates that a combination of $P^{Am} - \Pi$ will offer better gains than money invested in the bank. One can thus proceed by **buying the put option,** P^{Am} and **selling the portfolio,** Π (selling the underlying asset and loaning out the cash).

2. $\Delta[P^{Am} - \Pi] < r\Delta t[P^{Am} - \Pi]$.

   This suggests that the combination of $P^{Am} - \Pi$ performs worse than the cash investment in the bank. Subsequently, one will **sell the put option,** P^{Am} and **buy the portfolio,** Π (buying the underlying asset and borrowing money) (Higham 2009, 174-175).

For European options with no early exercise facility, the no arbitrage theory rules out both scenarios. However, for American options this is not the case. Scenario one states that the arbitrageur buys the American put option and sells the portfolio. This means that he controls the early exercise facility and therefore an arbitrage opportunity still exists. In scenario two, the arbitrageur sells the American put option and is therefore at the mercy of the early exercise facility because the new option holder can at any time exercise the option against the arbitrageur. In this case **the arbitrageur can no longer guarantee a return greater than the risk-less bank investment**.

The no arbitrage principle rules out scenario one, but not scenario two and one finds that the following inequality is valid:

$$\Delta(P^{Am} - \Pi) \leq r\Delta t(P^{Am} - \Pi). \tag{3.3.2}$$

Substituting equations 3.3.1 and 3.2.14 into equation 3.3.2 one finds:

$$\left(\frac{\partial P^{Am}}{\partial t} - rD + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P^{Am}}{\partial S^2}\right)\Delta t \leq r\Delta t(P^{Am} - AS - D). \tag{3.3.3}$$

According to (Higham 2009, 175), cancelling the Δt's and substituting equation 3.2.23 - adapted in terms of the P^{Am} notation - into equation 3.3.3, one finds that the Black-Scholes partial differential equation in equation 3.2.32 changes to the **Black-Scholes partial differential inequality**:

$$\frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\frac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial P^{Am}}{\partial S} - rP^{Am} \leq 0. \tag{3.3.4}$$

Similar to the Black-Scholes partial differential equation, the Black-Scholes partial differential inequality can be transformed into the heat equation. This equation can then be solved to obtain a unique price for an American put option. From the section on partial derivatives, one now appreciates the necessity of initial and boundary values to obtain such a unique solution. One of the difficulties encountered with the American pricing problem is that one of the boundaries needed cannot be specified beforehand. This stems from the early exercise facility of American options and is known as a **free boundary problem**. The following section addresses this topic.

3.4 The free boundary problem

As mentioned previously, American options have the advantage of permitting early exercise of the option at any time during the lifetime of the option contract (Wilmott et al. 2000, 54). This early exercise advantage complicates the Black-Scholes analysis and often places strain on computational methods (Higham 2009, 173) because the explicit formulae available for European options do not necessarily offer meaningful values for American options (Wilmott et al. 2000, 54).

The American option pricing function $P^{Am}(S,t)$, that ultimately represents the price of an American option, is a function of both time and the underlying asset's price, S. The free boundary problem is essentially **the task of determining the specific stock**

price $S_f(t)$ (critical asset price), at each point on the time interval, $0 \leq t \leq T$, at which it is optimal to exercise the American put option.

The payoff formula of the put option, defined in (Hull 2009, 183), is given by:

$$\Lambda(S(t)) = max[K - S, 0]. \tag{3.4.1}$$

If the stock price S is large, the put option payoff would be 0 and it will not be worthwhile to exercise the American put. On the other hand, as S approaches 0, the payoff from exercising the put option approaches the maximum value, K, and it would be optimal to exercise the option.

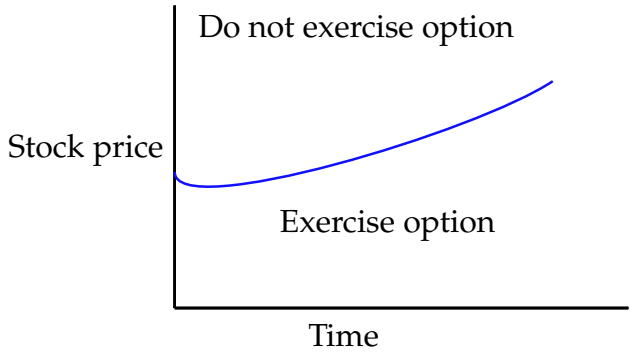

Figure 3.4.1: The early exercise boundary of an American put option

Interpolating between these two extreme scenarios, one finds an optimal exercise boundary (Higham 2009, 177). Therefore, at each point on the time interval, $0 \leq t \leq T$, there is a critical asset price $S_f(t)$, which marks the boundary between two regions. On the one side of this value, the option should be exercised, but on the other side of this value the option in question should be held (Wilmott et al. 2000, 55). This can be seen in Fig. 3.4.1. The critical asset price depends on the time remaining to expiry, as well as other variables of the partial differential equation, such as volatility (Wilmott et al. 2000, 106).

For American put options where (Higham 2009, 177):

- $S < S_f(t)$, it is **optimal to exercise the option** and $P^{Am} = \Lambda(S(t))$.
- $S > S_f(t)$, it is **optimal to hold the option** and $P^{Am} > \Lambda(S(t))$.

One needs the unknown critical asset prices, $S_f(t)$, at each point on the time interval, $0 \leq t \leq T$. These prices will form the optimal exercise boundary which is the second

boundary condition needed to uniquely solve the partial differential equation. The process of obtaining this optimal exercise boundary is known as the free boundary problem.

The concept of a free boundary is not unique to the American pricing problem. In order to understand it better, the obstacle problem is discussed in the section below. Although this problem deals with an elastic string, it is mathematically simple and it's physical interpretations are applicable to the American option pricing problem.

3.4.1 The obstacle problem as a free boundary value problem

The obstacle problem is discussed in much more mathematical detail in later sections of this chapter. For now, the basic idea behind this problem is focused on to aid understanding of the free boundary concept. Begin by considering an elastic string, tied at points x_0 and x_1. The string is stretched over a smooth obstacle that lies between these two points. This can be seen in Fig.3.4.2. The obstacle is defined by a function $g(x)$ and the string by function $u(x)$. Initially, one does not know the region of contact between the string and the obstacle (region between points a and b). What is known, is that the string is either in contact with the obstacle (the position is known) or it is not in contact (in which case the string must be straight). Additionally, two constraints are also known that enables one to find a unique solution to the obstacle problem. The four constraints can be summarised as (Wilmott et al. 2000, 55):

(i) The string must either lie on or above the obstacle.

(ii) The string must have a negative or zero curvature.

(iii) The string must be continuous.

(iv) The string's slope must be continuous.

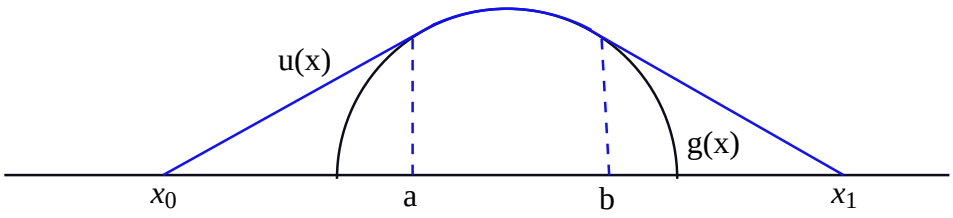

Figure 3.4.2: The obstacle probelm

A unique solution to the American option pricing problem can also be obtained by specifying a similar set of constraints to those of the obstacle problem (Wilmott et al. 2000, 56). These are:

(i) The option value must be greater or equal to the option payoff.

(ii) One must have the Black-Scholes inequality.

(iii) The option value must be a continuous function of S.

(iv) The option delta (slope) must be continuous.

The following section examines each of these constraints on their own merit. Each constraint will hereafter be written in mathematical notation:

(i) $P^{Am} \geq max[K-S,0], \qquad 0 \leq t \leq T$ and $S \geq 0$.

This constraint (lower boundary), indicates that **early exercise can occur, but that arbitrage shouldn't** (Wilmott et al. 2000, 56)(Higham 2009, 174). Desmond Higham, in his book, *An Introduction to Financial Option Valuation: Mathematics, Stochastics and Computation* (Higham 2009, 14), proves the lower boundary of the European put option by considering the following two portfolios: Portfolio Π_A contains a put option and its underlying stock, whilst portfolio Π_B contains a cash amount invested at a risk free interest rate, r:

$$\Pi_A = P + S, \tag{3.4.2}$$

and

$$\Pi_B = Ke^{-rT}. \tag{3.4.3}$$

At the exercise date, T, the following two possible scenarios for portfolio Π_A are identified:

- $S < K$: If the stock price is less than the strike price, the put option should be exercised. One obtains a payoff with a maximum value of K and the portfolio can have a maximum value of K.
- $S > K$: If the stock price is more than the strike price, the put option will not be exercised. The portfolio can have a maximum value of S, the value of the underlying stock also held in the portfolio.

Taking both of the above-mentioned scenarios into account, portfolio Π_A has a potential value of $max[K,S]$.

The value of portfolio Π_B at the exercise date will be K, the growth of the the initial invested amount at interest rate, r. Thus, portfolio Π_B only has a potential value of $max[K]$.

Portfolio Π_A is therefore more valuable than portfolio Π_B and this can be expressed as:

$$P + S \geq Ke^{-rt}. \tag{3.4.4}$$

Since a put option can never have a negative value, 3.4.4 can be re-written as (Oliver 2004, 14)(Higham 2009, 14):

$$P^{Am} \geq max[Ke^{-rt} - S, 0]. \tag{3.4.5}$$

In the case of the early exercise facility of the American option, one immediately receives the cash amount K, rather than having to wait until time T to receive K (which has a present value of Ke^{-rt}). Therefore, one can re-write 3.4.5 to obtain the first constraint. This is known as the lower boundary of the American put option:

$$P^{Am} \geq max[K - S, 0]. \tag{3.4.6}$$

(ii) $\dfrac{\partial P^{Am}}{\partial t} + \dfrac{1}{2}\dfrac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\dfrac{\partial P^{Am}}{\partial S} - rP^{Am} \leq 0.$

This inequality has already been established in section 3.3.

(iii) $P^{Am}(S,t)$ must be a continuous function of S.

Although the stochastic variable, S, is not a smooth function of time, $P^{Am}(S,t)$ can be a continuous function of S (Higham 2009, 73). This assumption is used when deriving the Black-Scholes equation and follows from the arbitrage principle. If there was a discontinuity in the option value as a function of S and if this discontinuity occurred for longer than an infinite decimal period of time, then the portfolio of options could make a risk-free profit, with probability one, if the underlying asset price reached the value at which the discontinuity occurred. However, discontinuous option prices do occur in some instances and are known as jumps (Wilmott et al. 2000, 57).

(iv) $\dfrac{\partial P^{Am}}{\partial S}$ must be continuous.

Examining the payoff function of the put option, $\Lambda(S(t)) = max[K - S, 0]$, one finds that its slope (Δ) is -1. This can be seen in Fig. 3.4.3.

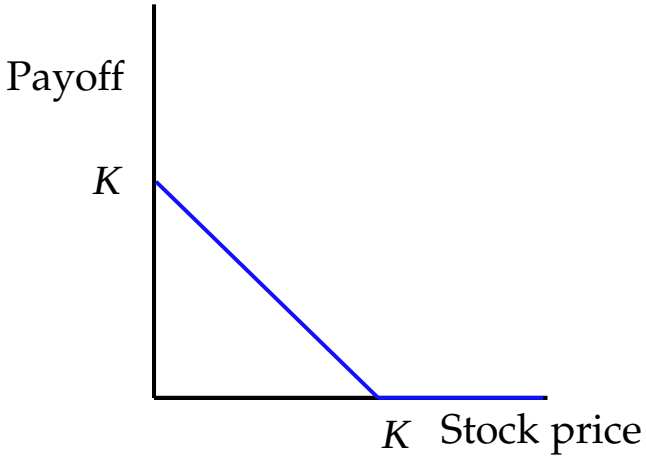

Figure 3.4.3: American put option payoff

As mentioned earlier, the American put option has an early exercise boundary at $S_f(t)$ and the option should be exercised when $S < S_f(t)$ (Wilmott et al. 2000, 57). The slope (Δ) of the option, at this critical asset price, $S_f(t)$, is now investigated. In *Option Pricing: Mathematical models and computation* (Wilmott et al. 2000, 56-58), the authors suggest that Δ at $S_f(t)$ can have three possible values:

- $\dfrac{\partial P^{Am}}{\partial S_f(t)} < -1.$
- $\dfrac{\partial P^{Am}}{\partial S_f(t)} > -1.$
- $\dfrac{\partial P^{Am}}{\partial S_f(t)} = -1.$

The following section explores each of these scenarios according to (Wilmott et al. 2000, 56-58).

- Keeping in mind that the Δ value of the option represents the relationship or ratio between the option value and the underlying asset value, begin by examining the first potential value, $\frac{\partial P^{Am}}{\partial S_f(t)} < -1$.

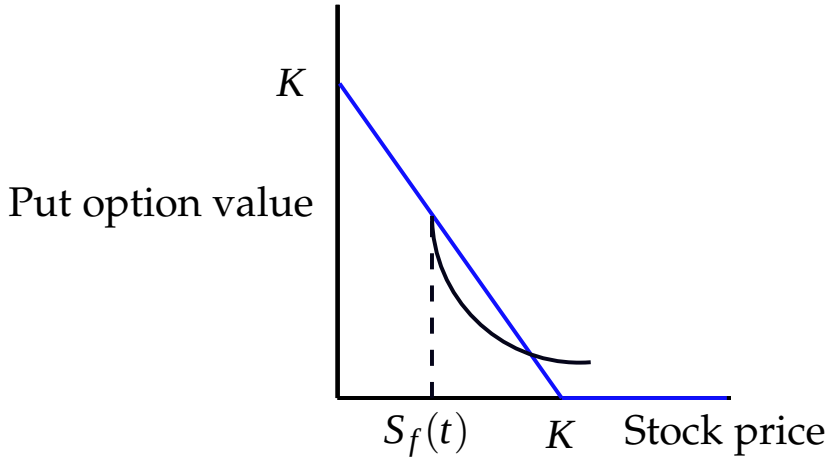

Figure 3.4.4: $\frac{dP^{Am}}{dS_f(t)} < -1$

Choose $S_f(t) < K$ and $\frac{\partial P^{Am}}{\partial S_f(t)} < -1$. If S **is increased**, the value of the put option will dip below the put payoff value, $\Lambda = max[K-S,0]$ (since the slope is more negative than -1). Therefore, $P^{Am}(S,t) < max[K-S,0]$ as in Fig. 3.4.4. The lower boundary of an American put option discussed on page 36, $P^{Am} \geq max[K-S,0]$, is now violated and subsequently $\frac{\partial P}{\partial S_f(t)} < -1$ is ruled out.

- The next task is to rule out $\frac{\partial P^{Am}}{\partial S_f(t)} > -1$. As before, choose $S_f(t) < K$ and S slightly larger than $S_f(t)$. The $S_f(t)$ value is now reduced along the lower boundary of the American put option, thereby increasing the option payoff and consequently the option value $P^{Am}(S,t)$ (Rodolfo 2007, 44). This is seen in the American put option payoff formula, $\Lambda = max[K-S,0]$. Whilst decreasing $S_f(t)$, one finds that $\frac{\partial P^{Am}}{\partial S}$ also decreases, as the slope becomes more steep (Rodolfo 2007, 44).

  By decreasing $S_f(t)$ sufficiently, one arrives at an underlying asset value where the inequality, $\frac{\partial P^{Am}}{\partial S_f(t)} > -1$, no longer holds. The option is therefore miss-valued (Rodolfo 2007, 44). This is seen in Fig. 3.4.5. Therefore, $\frac{\partial P}{\partial S_f(t)} > -1$ is also ruled out.

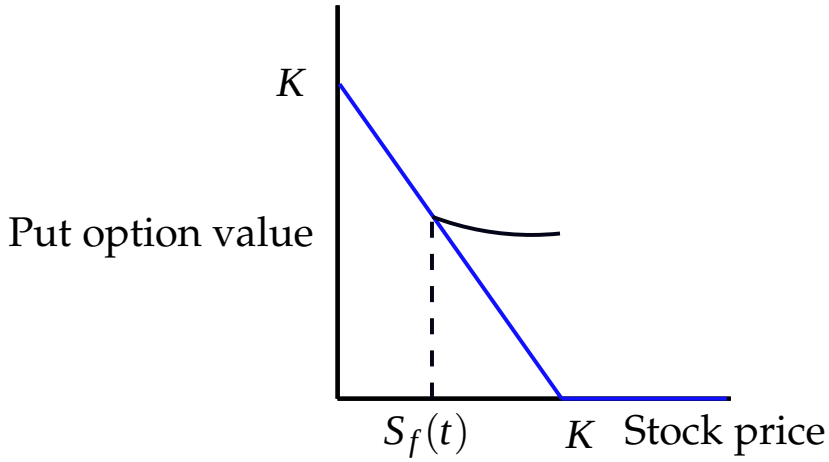

Figure 3.4.5: $\frac{dP^{Am}}{dS_f(t)} > -1$

- At this critical asset value, $S_f(t)$, neither of the two inequalities hold and one is able to conclude that $\frac{\partial P^{Am}}{\partial S_f(t)} = -1$, as in Fig. 3.4.6.

Constraints three and four, requiring that $P^{Am}(S,t)$ must be a continuous function of S and that $\frac{\partial P^{Am}}{\partial S}$ must be continuous, are collectively known as the *smooth-pasting*

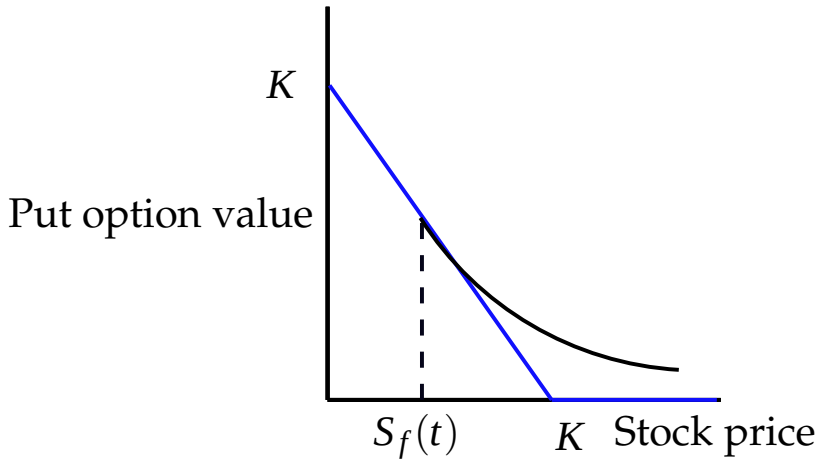

Figure 3.4.6: $\frac{dP^{Am}}{dS_f(t)} = -1$

condition. It indicates that the option value function, $P^{Am}(S,t)$, is **tangential** to it's payoff function, $\Lambda = max[K-S,0]$, in the point $S_f(t)$ (Wilmott 2009, 238).

3.4.2 The American put problem as a free boundary value problem

The following is a summary of the free boundary value problem of the American put option (Oliver 2004, 17) (Wilmott et al. 2000, 108-109) (Higham 2009, 177):

When early exercise is optimal:

$$\begin{gathered} S < S_f(t), \\ P^{Am}(S,t) = \max[K-S,0], \\ \frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\frac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial P^{Am}}{\partial S} - rP^{Am} \leq 0. \end{gathered} \tag{3.4.7}$$

When early exercise is not optimal:

$$\begin{gathered} S > S_f(t), \\ P^{Am}(S,t) > \max[K-S,0], \\ \frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\frac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial P^{Am}}{\partial S} - rP^{Am} = 0. \end{gathered} \tag{3.4.8}$$

Boundary conditions:

$$\lim_{S\to\infty} P^{Am}(S,t) = 0, \tag{3.4.9}$$

$$\frac{\partial P^{Am}}{\partial S_f(t)} = -1, \tag{3.4.10}$$

$$P^{Am}(S_f(t),t) = \max[K-S_f(t),0]. \tag{3.4.11}$$

Final condition:

$$P^{Am}(S,T) > \max[K-S,0]. \tag{3.4.12}$$

It is important to note that although $\frac{\partial P^{Am}}{\partial S}$ is continuous, as the point $S_f(t)$ is crossed, $\frac{\partial^2 P^{Am}}{\partial S^2}$ and $\frac{\partial P^{Am}}{\partial t}$ are not. This lack of continuity along the early exercise boundary affects the accuracy of numerical approximations (Seydel 2009, 165).

As mentioned earlier, the ordinary heat equation can uniquely be solved by stipulating

two boundary values and a time condition. For the American pricing problem, because the optimal exercise boundary is unknown beforehand, one has to include an additional boundary condition, as can be seen in the above section. The next task is to attempt the re-formulation of the pricing problem and reduce it to a **fixed boundary problem** that does not contain the free boundary explicitly and from which the free boundary can be obtained afterwards (Wilmott et al. 2000, 316).

There are many different transformations available to do exactly this, but this dissertation will focus on only one such method, the formulation of a **linear complimentary problem**. An alternative method, variational inequalities, is discussed in numerous literary sources, such as the book, *Option pricing: Mathematical models and computation* (Wilmott et al. 2000, 316). This formulation uses the method of finite elements to solve the pricing problem.

In section 3.5, the pricing problem is formulated as the linear complimentary problem, but first, the topic of the free boundary and the critical asset price $S_f(t)$ is further addressed by a closer inspection of the free boundary.

3.4.3 Asymptotic behaviour of the critical exercise price near expiry

The behaviour of the early exercise boundary in the vicinity of the expiry time, T, is important when pricing American options (Rodolfo 2007, 50). Although the precise details of this behaviour falls outside the scope of this dissertation, it is important to note that the assumed value of $S_f(T) = K$, is incorrect. For a better understanding of the correct value and a detailed description of the behaviour of the early exercise boundary near and at the expiry time, the reader can refer to the following resources: *A Comparative Study of American Option Evaluation and Computation* by K. Rodolfo (Rodolfo 2007, 45-51), *The Mathematics of Financial Derivatives* (Wilmott et al. 1996, 121-129) and *Mathematical models of Financial Derivatives* (Kwok 2008, 257-262).

3.5 Linear complimentary problem

In this section the American option pricing problem is reformulated as a **linear complimentary problem** (LCP). This is done in an attempt to state the option pricing problem **without explicit dependence on the unknown free boundary** (Wilmott et al. 2000, 123). Because the free boundary (a boundary required beforehand to uniquely solve the pricing problem) is initially unknown and because the problem can seldom be solved us-

ing an explicit method, numerical techniques are used (Wilmott et al. 2000, 122). One therefore proceeds by numerically finding a solution to the pricing problem, without having to solve the free boundary beforehand. The location of the boundary can subsequently be obtained once the linear complimentary problem has been solved (Wilmott et al. 2000, 123).

In a linear complimentary problem, no objective function is optimized (Murty 1988, 1). Let $\mathbf{M}$ be a known square matrix of order n and $\mathbf{g}$ is a known column vector in $\mathbb{R}^n$. Unknown column vectors, $\mathbf{w}$ and $\mathbf{z}$ are both in $\mathbb{R}^n$. The problem is to find $\mathbf{w} = (w_1, \ldots, w_n)^T$ and $\mathbf{z} = (z_1, \ldots, z_n)^T$ such that the following equations are satisfied (Murty 1988, 1):

$$\begin{aligned} \mathbf{w} - \mathbf{M}\mathbf{z} &= \mathbf{q}, \\ \mathbf{w} &\geq 0, \\ \mathbf{z} &\geq 0, \\ w_i z_i = 0 \qquad & i = 1, \ldots, n. \end{aligned} \tag{3.5.1}$$

In the section to follow, the obstacle problem, mentioned earlier is discussed as a linear complimentary problem. This simplistic mathematical model is included to aid the reader's understanding of the more complicated American pricing model to follow later on in this chapter.

3.5.1 The obstacle problem as a linear complimentary problem

Consider Fig. 3.4.2 on page 34. Let a function $g(x)$ be given, where $x \in \mathbb{R}$, $g \in C^2$ and $g''(x) < 0$ (Oliver 2004, 18). Function $g(x)$ **represents the height of the obstacle** (Wilmott et al. 2000, 123). Let $u(x)$ **be a function that represents a string stretched over** $g(x)$, where $u \in C^1[x_0, x_1]$ and $u(x_0) = u(x_1) = 0$ (Oliver 2004, 18). On the interval $[a,b]$, $u(x)$ and $g(x)$ coincide, thus, $u(x) = g(x)$. With the string and the obstacle in contact, this implies that the string is bent and therefore $u''(x) < 0$ holds. At all the other points, where the string is not in contact with the obstacle, $u(x) > g(x)$ holds and the string is straight, implying that $u''(x) = 0$ (Oliver 2004, 18). **Initially,** a **and** b **are unknown**.

The obstacle problem as a free boundary value problem as stipulated in (Wilmott et al. 2000, 124) is defined as:

$$
\begin{array}{ll}
x = x_0: & u(x_0) = 0, \\
x_0 < x < a: & u(x) > g(x) \text{ and } u''(x) = 0, \\
x = a: & u(a) = g(a) \text{ and } u'(a) = g'(a), \\
a < x < b: & u(x) = g(x) \text{ and } u''(x) = g''(x), \\
x = b: & u(b) = g(b) \text{ and } u'(b) = g'(b), \\
b < x < x_1: & u(x) > g(x) \text{ and } u''(x) = 0, \\
x = x_1: & u(x_1) = 0.
\end{array} \tag{3.5.2}
$$

This free boundary value problem can also be formulated as a **linear complimentary problem** (Oliver 2004, 19) (Wilmott et al. 2000, 124) (Seydel 2009, 166):

Find $u(x)$ such that:

$$
\begin{array}{c}
u''(u-g) = 0, \\
-u''(x) \geq 0, \\
u(x) - g(x) \geq 0,
\end{array} \tag{3.5.3}
$$

subject to the constraints: $u(x_0) = u(x_1) = 0$ and $u \in C^1[x_0, x_1]$.

3.5.2 Transforming the Black-Scholes PDE to the heat equation

Before formally presenting the American pricing problem as a linear complimentary problem, the **Black-Scholes partial differential equation** in 3.2.32 is transformed into the one-dimensional parabolic heat equation. Start by rewriting equation 3.2.32 to suit American put options:

$$
\frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\frac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial P^{Am}}{\partial S} - rP^{Am} = 0, \tag{3.5.4}
$$

on the domain (Ugur 2008, 116),

$$
\mathscr{D}_{P^{Am}} = \{(S,t) : S > 0, 0 \leq t \leq T\}. \tag{3.5.5}
$$

3.5.4 is to be transformed into the **one-dimensional parabolic heat equation** of the form (Oliver 2004, 19):

$$
\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}, \tag{3.5.6}
$$

for x and τ on the domain (Ugur 2008, 116),

$$\mathscr{D}_u = \{(x,\tau) : -\infty < x < \infty, 0 \leq \tau \leq \frac{\sigma^2}{2}T\}. \tag{3.5.7}$$

The following derivation follows the transformation to the heat equation as given in (Ugur 2008, 117-118) and (Wilmott et al. 2000, 98-99, 127). The transformation of 3.5.4 is achieved by using various substitutions, the first of which is to re-define the independent variables S and t as (Wilmott et al. 2000, 127):

$$S = Ke^x, \qquad t = T - \frac{\tau}{\frac{1}{2}\sigma^2}. \tag{3.5.8}$$

Here time $t = T$ corresponds to $\tau = 0$ and therefore the final condition stipulated in 3.4.12 on page 41, now becomes an initial condition for 3.5.6 (Oliver 2004, 20). Next the dependant variable, $f(x,\tau)$, is defined as (Ugur 2008, 117):

$$f(x,\tau) = \frac{1}{K}P^{Am}(S,t) = \frac{1}{K}P^{Am}\left(Ke^x, T - \frac{\tau}{\frac{1}{2}\sigma^2}\right). \tag{3.5.9}$$

The change of the independent variables ensures that the domain of the new dependent variable f, is D_u, as stipulated in 3.5.7 (Ugur 2008, 117).

Using the chain rule for several variables, the following derivatives are obtained (Ugur 2008, 117):

$$\frac{\partial P^{Am}}{\partial t} = K\left(\frac{\partial f}{\partial \tau}\frac{\partial \tau}{\partial t}\right) = -\frac{\sigma^2}{2}K\frac{\partial f}{\partial \tau}, \tag{3.5.10}$$

$$\frac{\partial P^{Am}}{\partial S} = K\left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial S}\right) = \frac{K}{S}\frac{\partial f}{\partial x}, \tag{3.5.11}$$

$$\frac{\partial^2 P^{Am}}{\partial S^2} = \frac{\partial}{\partial S}\left(\frac{\partial P^{Am}}{\partial S}\right) = \frac{K}{S^2}\left(\frac{\partial^2 f}{\partial x^2} - \frac{\partial f}{\partial x}\right). \tag{3.5.12}$$

Substituting equations 3.5.10, 3.5.11 and 3.5.12 into equation 3.5.4, one obtains:

$$\frac{\partial f}{\partial \tau} = \frac{\partial^2 f}{\partial x^2} + \left(\frac{2r}{\sigma^2} - 1\right)\frac{\partial f}{\partial x} - \frac{2r}{\sigma^2}f. \tag{3.5.13}$$

A new constant, κ, is also defined (Wilmott et al. 2000, 127) (Ugur 2008, 117):

$$\kappa = \frac{2r}{\sigma^2}, \tag{3.5.14}$$

and 3.5.13 can be re-written as (Ugur 2008, 117):

$$\frac{\partial f}{\partial \tau} = \frac{\partial^2 f}{\partial x^2} + (\kappa - 1)\frac{\partial f}{\partial x} - \kappa f. \tag{3.5.15}$$

Now, 3.5.15 only has one constant, κ, instead of four, K, T, σ^2 and r (Wilmott et al. 2000, 98). The next step is to consider a function $u(x,\tau)$, defined by (Wilmott et al. 2000, 98):

$$f(x,\tau) = e^{\alpha x + \beta \tau} u(x,\tau), \tag{3.5.16}$$

where α and β are unknown constants.

Applying the chain rule to the function $f(x,\tau)$, one obtains the following derivatives:

$$\frac{\partial f}{\partial \tau} = e^{\alpha x + \beta \tau}\left(\beta u + \frac{\partial u}{\partial \tau}\right), \tag{3.5.17}$$

$$\frac{\partial f}{\partial x} = e^{\alpha x + \beta \tau}\left(\alpha u + \frac{\partial u}{\partial x}\right), \tag{3.5.18}$$

$$\frac{\partial^2 f}{\partial x^2} = e^{\alpha x + \beta \tau}\left(\alpha^2 u + 2\alpha\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2}\right). \tag{3.5.19}$$

Substituting equations 3.5.17, 3.5.18 and 3.5.19 into equation 3.5.15, the following equation is obtained (Wilmott et al. 2000, 98):

$$\beta u + \frac{\partial u}{\partial \tau} = \alpha^2 u + 2\alpha\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + (\kappa - 1)\left(\alpha u + \frac{\partial u}{\partial x}\right) - \kappa u. \tag{3.5.20}$$

Rearranging 3.5.20, one finds:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x}(2\alpha + \kappa - 1) + u(-\beta + \alpha^2 + \alpha\kappa - \alpha - \kappa) \tag{3.5.21}$$

To eliminate terms u and $\dfrac{\partial u}{\partial x}$ from 3.5.21, thereby obtaining the heat equation, let:

$$2\alpha + \kappa - 1 = 0, \tag{3.5.22}$$

and

$$-\beta + \alpha^2 + \alpha\kappa - \alpha - \kappa = 0. \tag{3.5.23}$$

Solving these equations simultaneously, one finds that (Wilmott et al. 2000, 98):

$$\alpha = -\frac{1}{2}(\kappa - 1), \tag{3.5.24}$$

and

$$\beta = -\frac{1}{4}(\kappa + 1)^2. \tag{3.5.25}$$

Substituting these values into 3.5.16, one obtains (Wilmott et al. 2000, 98):

$$f(x,\tau) = e^{-\frac{1}{2}(\kappa-1)x - \frac{1}{4}(\kappa+1)^2\tau}u(x,\tau), \tag{3.5.26}$$

and now for the same values of α and β, 3.5.21 reduces to the heat equation:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}, \qquad (x,\tau) \in \mathscr{D}_u. \tag{3.5.27}$$

Considering the Black-Scholes inequality suited to American put options as defined on page 32:

$$\frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\frac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial P^{Am}}{\partial S} - rP^{Am} \leq 0, \tag{3.5.28}$$

the following inequality to the heat equation is obtained to suit American put options (Oliver 2004, 20):

$$\frac{\partial u}{\partial \tau} \geq \frac{\partial^2 u}{\partial x^2}. \tag{3.5.29}$$

3.5.3 The American put problem as a linear complimentary problem

Keeping the lower boundary of an American put option in mind, one knows that $P^{Am} \geq max[K-S,0]$. Using the definition of the function $f(x,\tau)$ in 3.5.9 on page 45, the lower boundary of $f(x,\tau)$ is now defined as (Wilmott et al. 2000, 98):

$$P^{Am}(S,t) = Kf(x,\tau) \geq K(max[1-e^x,0]). \tag{3.5.30}$$

Therefore,

$$f(x,\tau) \geq max[1-e^x,0]. \tag{3.5.31}$$

And because the function $f(x,\tau)$ is defined in terms of function $u(x,\tau)$ in equation 3.5.26, one can re-write 3.5.26 as (Oliver 2004, 21):

$$u(x,\tau) \geq e^{\frac{1}{2}(\kappa-1)x+\frac{1}{4}(\kappa+1)^2\tau} f(x,\tau). \tag{3.5.32}$$

Using the definition of f in 3.5.31, u has the final form (Oliver 2004, 21):

$$u(x,\tau) \geq e^{\frac{1}{2}(\kappa-1)x+\frac{1}{4}(\kappa+1)^2\tau} max[1-e^x,0]. \tag{3.5.33}$$

A new function, $g(x,\tau)$, is also defined. This replaces the original payoff function, $\Lambda = max[K-S,0]$ (Oliver 2004, 21) (Wilmott et al. 1996, 119):

$$g(x,\tau) = e^{\frac{1}{2}(\kappa-1)x+\frac{1}{4}(\kappa+1)^2\tau} max[1-e^x,0]. \tag{3.5.34}$$

This can be simplified to (Wilmott et al. 2000, 127):

$$g(x,\tau) = e^{\frac{1}{4}(\kappa+1)^2\tau} max[e^{\frac{1}{2}(\kappa-1)x-\frac{1}{2}(\kappa+1)x},0]. \tag{3.5.35}$$

As mentioned earlier, the **initial condition** for the heat equation, $\tau = 0$, equals the final condition $t = T$ as stipulated in equation 3.4.12 on page 41. Therefore, one finds that (Wilmott et al. 2000, 127):

$$u(x,0) = g(x,0) = max[e^{\frac{1}{2}(\kappa-1)x-\frac{1}{2}(\kappa+1)x},0], \tag{3.5.36}$$

and for all other values of τ (Wilmott et al. 2000, 127):

$$u(x,\tau) \geq g(x,\tau). \tag{3.5.37}$$

Boundary conditions can also be defined for large ($x \to \infty$) and small ($x \to -\infty$) values (Wilmott et al. 2000, 128):

$$u(\infty,\tau) = g(\infty,\tau) = 0, \tag{3.5.38}$$

and

$$u(-\infty,\tau) = g(-\infty,\tau). \tag{3.5.39}$$

Consider how equations 3.4.7 and 3.4.8 on page 41 resemble equation 3.5.2 on page 44. Therefore, the American pricing problem can also be formulated as a linear complimentary problem, where the free boundary, $S_f(t)$, is not explicitly mentioned, but will become apparent once the linear complimentary problem has been solved (Oliver 2004, 19) (Wilmott et al. 2000, 316).

All the equations needed to formally define the linear complimentary problem have now been mentioned and therefore **the linear complimentary problem for the American put option pricing problem** can be written as (Oliver 2004, 21):

Find $u(x,\tau)$ **such that:**

$$\left(\frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2}\right)(u(x,\tau) - g(x,\tau)) = 0,$$

$$\left(\frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2}\right) \geq 0, \tag{3.5.40}$$

$$u(x,\tau) - g(x,\tau) \geq 0,$$

where:

$$g(x,\tau) = e^{\frac{1}{4}(\kappa+1)^2\tau} max[e^{\frac{1}{2}(\kappa-1)x - \frac{1}{2}(\kappa+1)x}, 0], \tag{3.5.41}$$

subject to the **initial condition**:

$$u(x,0) = g(x,0), \qquad -\infty < x < \infty, \tag{3.5.42}$$

and **boundary conditions at** $x = \pm\infty$:

$$u(x,\tau) = g(x,\tau), \qquad 0 \leq \tau \leq \frac{1}{2}\sigma^2 T, \tag{3.5.43}$$

and the constraints that u and $\dfrac{\partial u}{\partial x}$ are continuous (Wilmott et al. 2000, 129).

This can now be compared to the linear complimentary problem of the obstacle problem in equation 3.5.3 on page 44. Note that the two different scenarios in equation 3.5.40 correspond to the different option exercise scenarios. When it is optimal to exercise the option, $u = g$, and when it is better to hold the option, $u > g$ (Wilmott et al. 2000, 129). It is also important to note that there is no explicit mention of the unknown free-boundary (Wilmott et al. 2000, 131). It can be shown that linear complimentary formulation is equivalent to the free boundary problem stated earlier. However, the technique of doing so relies on functional analysis and falls outside the scope of this dissertation (Wilmott et al. 1996, 120).

This concludes the formal definition of the American option pricing problem. First **the Black-Scholes equation** was derived and then adapted to suit the American put by changing it to **the Black-Scholes inequality**. The next step was to define the American pricing problem as a **free boundary problem**. Then the Black-Scholes inequality was transformed to the **one dimensional heat equation**. Finally the pricing problem was stipulated as a **linear complimentary problem**.

The next chapter addresses the finite difference numerical method, which is the focus of

this dissertation, in detail. An understanding of the finite difference method is essential before one attempts the complex task of solving the American pricing problem using this finite difference technique.

Chapter 4

Finite Difference Methods

4.1 Introduction

The previous two chapters were tasked with formally defining the American put option pricing problem. Since no general analytical formulae are available to solve this problem, there are a number of finite difference methods that can be employed to do just that. The second order accurate Crank-Nicolson finite difference method will be used in this dissertation. The goal is to find a price, P^{Am} of an American put option containing an underlying asset with price, S. As will be seen in the following sections, because of the discretization process, one obtains a surface of prices. This surface contains all the option values, P^{Am}, on the half strip $S > 0$ and $0 \leq t \leq T$ (Seydel 2009, 141).

After considering the transformation of the Black-Scholes equation to the heat equation, $\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}$, one finds that the original half strip is now transformed to become the strip $-\infty < x < \infty$ and $0 \leq \tau \leq \frac{1}{2}\sigma^2 T$ (Seydel 2009, 143). Because discretization is done on a finite interval, the infinite interval, $-\infty < x < \infty$, needs to be replaced by a finite interval $x_{min} \leq x \leq x_{max}$, where $x_{min} < 0$ and $x_{max} > 0$. Using the original transformation on page 45, one can now define $S_{min} = Ke^{x_{min}}$ and $S_{max} = Ke^{x_{max}}$ (Seydel 2009, 146).

The finite difference method is used to solve the heat equation defined in the linear complimentary problem, given by equation 3.5.40 on page 50. This solution is then converted back to variables S and t using transformations given in 3.5.8 on page 45.

In the following section, some foundational concepts are covered. This will aid the understanding of the discretization process of the finite difference method. Figures will also be given to further explain the finite difference grid and its notation. In sections

4.3, 4.4 and 4.5, the three variations of the finite difference method are discussed in great detail. These are the Explicit, the Implicit and the Crank-Nicolson finite difference methods. The more general θ - finite difference notation is also introduced. Later in the chapter, the different methods available to solve tridiagonal systems are covered and finally the American option pricing problem is solved by introducing the reader to a comprehensive algorithm that systematically incorporates all the different numerical methods one needs to implement. The author is not aware of of such a comprehensive algorithm in the literature.

4.2 Foundations

The first step is to discretize both the x and τ axes. Thus, one subdivides these axes into equally spaced intervals of length δx and $\delta\tau$ respectively (Seydel 2009, 146). This process can be summarised as follows:

- $\tau = 0, \ldots, \frac{1}{2}\sigma^2 T$,

  $\tau_m = m\delta\tau, \qquad m = 0, \ldots, M,$

  where M, the number of intervals on the τ-axis, can be chosen beforehand. The upper boundary of τ stems from the substitution used in 3.5.8 on page 45. The subinterval length $\delta\tau$, is defined as:

  $$\delta\tau = \frac{(\frac{1}{2}\sigma^2 T - 0)}{M}.$$

- $x = x_{min}, \ldots, x_{max}$,

  $x_n = x_{min} + n\delta x, \qquad n = 0, \ldots, N,$

  where N, the amount of intervals on the x-axis, can be chosen beforehand and where the subinterval length δx, is defined as:

  $$\delta x = \frac{(x_{max} - x_{min})}{N}.$$

- The following notation will be used to denote the exact solution of the heat equation at a specific node (n,m), $u_n^m = u(n\delta x, m\delta\tau)$ (Wilmott et al. 2000, 274).

- The notation, v_n^m, denotes an approximate value to the exact solution, u_n^m. Therefore, $v_n^m \approx u_n^m$ (Wilmott et al. 2000, 326).

Fig. 4.2.1 is a representation of the $x - \tau$ discretisized grid. The blue dots represent values known at the initial time, $\tau = 0$. The red dots represent values known at boundaries, x_{max} and x_{min}. Values at all the other nodes are unknown and need to be computed.

4.3 Explicit finite difference method

Start by considering the heat equation, $\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}$. Using a **forward difference Taylor approximation** to approximate $\frac{\partial u}{\partial \tau}$ and the symmetric central difference Taylor approximation to approximate $\frac{\partial^2 u}{\partial x^2}$, one finds (Wilmott et al. 2000, 270):

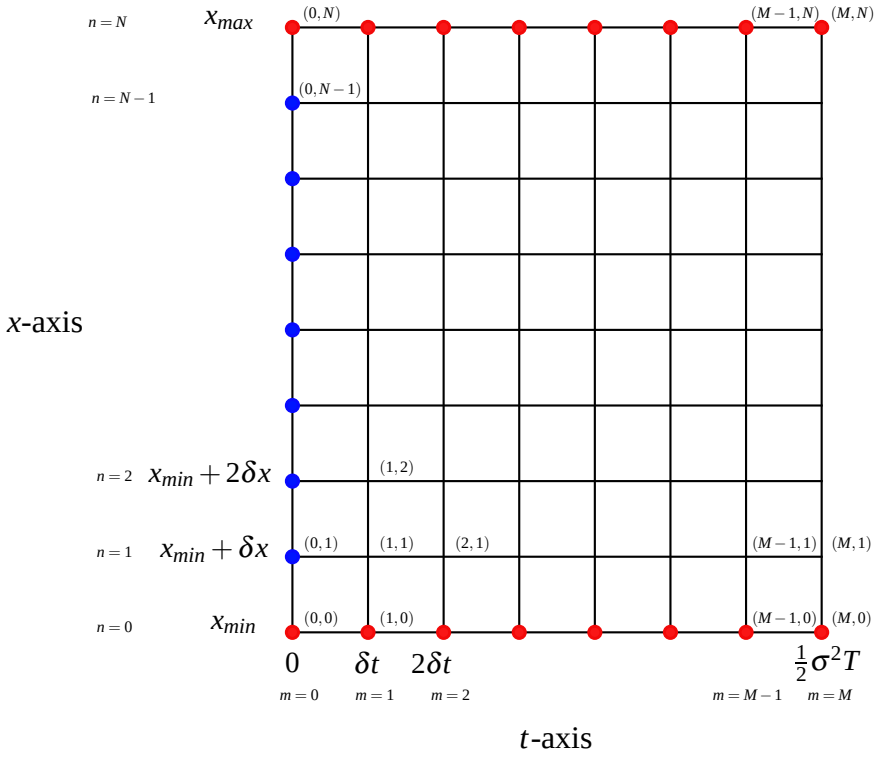

Figure 4.2.1: $x - \tau$ grid after discretization

$$\frac{\partial u}{\partial \tau} = \frac{u_n^{m+1} - u_n^m}{\delta \tau} + O(\delta \tau), \tag{4.3.1}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2). \tag{4.3.2}$$

Therefore, the parabolic heat equation approximated by Taylor expansion is (Wilmott et al. 2000, 278)(Kincaid & Cheney 1991, 574):

$$\frac{u_n^{m+1} - u_n^m}{\delta \tau} + O(\delta \tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2). \tag{4.3.3}$$

Re-arranging 4.3.3, by separating the terms containing u^{m+1} and u^m and then implementing the approximating notation, v_n^m, one obtains the following equation (Wilmott et al. 2000, 278):

$$v_n^{m+1} = v_n^m + \alpha(v_{n-1}^m - 2v_n^m + v_{n+1}^m), \quad 0 < n < N, \quad 0 < m \leq M, \tag{4.3.4}$$

where

$$\alpha = \frac{\delta \tau}{(\delta x)^2}. \tag{4.3.5}$$

This can be re-written as (Seydel 2009, 147):

$$v_n^{m+1} = \alpha v_{n-1}^m + (1 - 2\alpha) v_n^m + \alpha v_{n+1}^m, \quad 0 < n < N, \quad 0 < m \leq M, \tag{4.3.6}$$

subject to the **initial condition** (Wilmott et al. 2000, 280) (Seydel 2009, 147):

$$v_n^0 = g(x_n, 0), \qquad 0 \leq n \leq N, \tag{4.3.7}$$

where $g(x, \tau)$ is defined in equation 3.5.41 on page 50.

The **boundary conditions** are (Wilmott et al. 2000, 280):

$$v_0^m = g(x_{min}, \tau_m), \qquad 0 < m \leq M, \tag{4.3.8}$$

and

$$v_N^m = g(x_{max}, \tau_m), \qquad 0 < m \leq M. \tag{4.3.9}$$

When written in matrix notation, one finds (Kincaid & Cheney 1991, 576)(Seydel 2009, 148):

$$v^{m+1} = \mathbf{A}v^m, \qquad m = 0, \ldots, M, \tag{4.3.10}$$

where matrix $\mathbf{A}$ is an $(N-1) \times (N-1)$ tridiagonal matrix (Seydel 2009, 148) (Wilmott et al. 1996, 146):

$$\mathbf{A} = \begin{pmatrix} (1-2\alpha) & \alpha & 0 & \cdots & 0 \\ \alpha & (1-2\alpha) & \alpha & \cdots & 0 \\ 0 & \alpha & (1-2\alpha) & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \alpha \\ 0 & \cdots & 0 & \alpha & (1-2\alpha) \end{pmatrix},$$

and vector v^m is of length $(N-1)$ and of the form (Wilmott et al. 1996, 146)(Seydel 2009, 147):

$$v^m = \begin{pmatrix} v_1^m \\ \vdots \\ \vdots \\ \vdots \\ v_{N-1}^m \end{pmatrix}.$$

It is important to note that the explicit method is only stable for $0 < \alpha \leq \frac{1}{2}$ (Seydel 2009, 151). Therefore, special attention has to be paid when choosing interval lengths δx and $\delta\tau$. Due to this restriction, a more flexible method that remains stable for any values of δx and $\delta\tau$ is preferred.

Using **a backward difference Taylor approximation** to approximate $\dfrac{\partial u}{\partial \tau}$ and the symmetric central difference Taylor approximation to approximate $\dfrac{\partial^2 u}{\partial x^2}$, one finds (Wilmott et al. 2000, 294)(Seydel 2009, 151):

$$\frac{\partial u}{\partial \tau} = \frac{u_n^m - u_n^{m-1}}{\delta\tau} + O(\delta\tau), \tag{4.3.11}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2). \tag{4.3.12}$$

Using these Taylor expansions, one now finds an approximation for the parabolic heat equation to be (Wilmott et al. 2000, 294):

$$\frac{u_n^m - u_n^{m-1}}{\delta\tau} + O(\delta\tau) = \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} + O((\delta x)^2). \tag{4.3.13}$$

Equation 4.3.13 is re-arranged by separating the terms containing u^m and u^{m-1}. Using the approximating notation, v_n^m, one now obtains the following equation (Wilmott et al. 2000, 295) (Seydel 2009, 151):

$$-\alpha v_{n-1}^m + (2\alpha+1)v_n^m - \alpha v_{n+1}^m = v_n^{m-1}, \quad 0 < n < N, \quad 0 < m \leq M, \tag{4.3.14}$$

where α is defined in 4.3.5 on page 56. Equation 4.3.14 is subject to the **initial condition** (Wilmott et al. 2000, 295) (Seydel 2009, 147):

$$v_n^0 = g(x_n, 0), \qquad 0 \leq n \leq N, \tag{4.3.15}$$

where $g(x,\tau)$ is defined in equation 3.5.41 and also subject to **boundary conditions** (Wilmott et al. 2000, 295):

$$v_{-N}^m = g(x_{min}, \tau_m), \qquad 0 < m \leq M, \tag{4.3.16}$$

and

$$v_N^m = g(x_{max}, \tau_m), \qquad 0 < m \leq M. \tag{4.3.17}$$

When written in matrix notation, one finds (Kincaid & Cheney 1991, 581)(Seydel 2009, 152):

$$v^m = \mathbf{A}^{-1} v^{m-1}, \qquad m = 1, \ldots, M, \tag{4.3.18}$$

where matrix $\mathbf{A}$ is a $(N-1)\times(N-1)$ tridiagonal matrix (Seydel 2009, 152)(Wilmott et al. 1996, 146):

$$\mathbf{A} = \begin{pmatrix} (2\alpha+1) & -\alpha & 0 & \cdots & 0 \\ -\alpha & (2\alpha+1) & -\alpha & \cdots & 0 \\ 0 & -\alpha & (2\alpha+1) & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & -\alpha \\ 0 & \cdots & 0 & -\alpha & (2\alpha+1) \end{pmatrix},$$

and vector v^m is of length $(N-1)$ and has the form (Wilmott et al. 1996, 146)(Seydel 2009, 147):

$$v^{\mathbf{m}} = \begin{pmatrix} v_1^m \\ \vdots \\ \vdots \\ \vdots \\ v_{N-1}^m \end{pmatrix}.$$

The implicit finite difference method is stable for all values of α (Seydel 2009, 152).

4.4 Crank-Nicolson implicit finite difference method

In comparison to both the explicit and implicit methods, where the discretization of $\frac{\partial u}{\partial \tau}$ was of order $O(\delta\tau)$, the Crank-Nicolson finite difference method uses a discretization of order $O((\delta\tau)^2)$. This method is also stable for all values of α (Seydel 2009, 153).

The **Crank-Nicolson implicit method is an average of both the implicit and explicit methods** (Wilmott et al. 2000, 306). Using a **forward difference approximation at node (m) on the τ-axis** and the approximating notation, v_n^m (Seydel 2009, 153):

$$\frac{v_n^{m+1} - v_n^m}{\delta\tau} = \frac{v_{n+1}^m - 2v_n^m + v_{n-1}^m}{(\delta x)^2}, \tag{4.4.1}$$

and a **backward difference approximation at node (m$+1$) on the τ-axis** and the approximating notation, v_n^{m+1} (Seydel 2009, 153):

$$\frac{v_n^{m+1} - v_n^m}{\delta\tau} = \frac{v_{n+1}^{m+1} - 2v_n^{m+1} + v_{n-1}^{m+1}}{(\delta x)^2}, \tag{4.4.2}$$

the addition of these two equations yields (Seydel 2009, 153):

$$\begin{aligned} &\frac{v_n^{m+1} - v_n^m}{\delta\tau} = \frac{1}{2(\delta x)^2}(v_{n+1}^m - 2v_n^m + v_{n-1}^m + v_{n+1}^{m+1} - 2v_n^{m+1} + v_{n-1}^{m+1}), \\ &0 < n < N, \quad 0 < m \leq M. \end{aligned} \tag{4.4.3}$$

Re-arranging 4.4.3 by separating terms containing v^{m+1} and v^m, one obtains the following (Seydel 2009, 154):

$$-\frac{\alpha}{2}v_{n-1}^{m+1}+(1+\alpha)v_n^{m+1}-\frac{\alpha}{2}v_{n+1}^{m+1}=\frac{\alpha}{2}v_{n-1}^{m}+(1-\alpha)v_n^{m}+\frac{\alpha}{2}v_{n+1}^{m}, \qquad 0<n<N, \quad 0<m\leq M. \tag{4.4.4}$$

Again, 4.4.4 is subject to the **initial condition** (Wilmott et al. 2000, 308) (Seydel 2009, 154):

$$v_n^0=g(x_n,0), \qquad 0\leq n\leq N, \tag{4.4.5}$$

where $g(x,\tau)$ is defined in equation 3.5.41 on page 50.

It is also subject to **boundary conditions** (Wilmott et al. 2000, 295):

$$v_{-N}^m=g(x_{min},\tau_m), \qquad 0<m\leq M, \tag{4.4.6}$$

and

$$v_N^m=g(x_{max},\tau_m) \qquad 0<m\leq M. \tag{4.4.7}$$

Writing 4.4.4 in matrix notation, one finds (Seydel 2009, 155):

$$\mathbf{A}v^{m+1}=\mathbf{B}v^m, \qquad m=0,\ldots,M, \tag{4.4.8}$$

where both $\mathbf{A}$ and $\mathbf{B}$ are $(N-1)\times(N-1)$ tridiagonal matrices (Seydel 2009, 154):

$$\mathbf{A}=\begin{pmatrix} (1+\alpha) & -\frac{\alpha}{2} & 0 & \cdots & 0 \\ -\frac{\alpha}{2} & (1+\alpha) & -\frac{\alpha}{2} & \cdots & 0 \\ 0 & -\frac{\alpha}{2} & (1+\alpha) & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & -\frac{\alpha}{2} \\ 0 & \cdots & 0 & -\frac{\alpha}{2} & (1+\alpha) \end{pmatrix},$$

$$\mathbf{B} = \begin{pmatrix} (1-\alpha) & \frac{\alpha}{2} & 0 & \cdots & 0 \\ \frac{\alpha}{2} & (1-\alpha) & \frac{\alpha}{2} & \cdots & 0 \\ 0 & \frac{\alpha}{2} & (1-\alpha) & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \frac{\alpha}{2} \\ 0 & \cdots & 0 & \frac{\alpha}{2} & (1-\alpha) \end{pmatrix},$$

and vector v^m is of length $(N-1)$ and has the form (Wilmott et al. 1996, 309)(Seydel 2009, 147):

$$v^m = \begin{pmatrix} v_1^m \\ \vdots \\ \vdots \\ \vdots \\ v_{N-1}^m \end{pmatrix}.$$

Before addressing the topic of solving a tridiagonal system of equations, a generic finite difference notation, θ-notation, is first introduced. Here, θ can be manipulated to resemble any one of the three above-mentioned methods.

4.5 θ - finite difference notation

This notation generalizes the finite difference method, combining both the explicit and implicit methods (Kincaid & Cheney 1991, 582). Here, θ can be adjusted to form either the explicit ($\theta = 0$), implicit ($\theta = 1$) or Crank-Nicolson implicit method ($\theta = \frac{1}{2}$) (Kincaid & Cheney 1991, 582). Using the appropriate Taylor expansions, one finds (Wilmott et al. 2000, 325 - 326):

$$\frac{\partial u}{\partial \tau} = \frac{u_n^{m+1} - u_n^m}{\delta\tau} + O(\delta\tau), \tag{4.5.1}$$

$$\frac{\partial^2 u}{\partial x^2} = \theta\left(\frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2}\right) + (1-\theta)\left(\frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2}\right) + O((\delta x)^2), \tag{4.5.2}$$

where $0 \leq \theta \leq 1$. The parabolic heat equation approximated by Taylor expansion can be written as (Wilmott et al. 2000, 325 - 326) (Kincaid & Cheney 1991, 582):

$$\frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2} \approx \frac{u_n^{m+1} - u_n^m}{\delta \tau} - \theta \left(\frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{(\delta x)^2} \right) - (1-\theta) \left(\frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{(\delta x)^2} \right). \tag{4.5.3}$$

Re-arranging 4.5.3 by separating the terms containing u^{m+1} and u^m and implementing the approximating notation, v_n^m, one obtains the following equation (Wilmott et al. 2000, 313):

$$\begin{aligned} &v_n^{m+1} - \alpha\theta(v_{n-1}^{m+1} - 2v_n^{m+1} + v_{n+1}^{m+1}) = v_n^m + \alpha(1-\theta)(v_{n-1}^m - 2v_n^m + v_{n+1}^m), \\ &0 < n < N, \quad 0 < m \leq M, \end{aligned} \tag{4.5.4}$$

where as before, $\alpha = \dfrac{\delta \tau}{(\delta x)^2}$.

4.6 Finite difference method for solving the obstacle problem

One is required to find a function, $u(x)$, as stipulated in the linear complimentary problem in equation 3.5.3 on page 44. Start by approximating the second derivative found in equation 3.5.3 using the appropriate Taylor-expansion (Wilmott et al. 2000, 317):

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{n+1} - 2u_n + u_{n-1}}{(\delta x)^2} + O((\delta x)^2), \tag{4.6.1}$$

where $u_n = u(n\delta x)$. The following notation is used:

$$x_n = x_0 + n\delta x, \qquad n = 0, \ldots, N,$$

where $[x_0, x_1]$ are the lower and upper boundaries on the x-axis (Refer to Fig. 3.4.2) and $x_1 = x_0 + N\delta x$. Now one can define δx as:

$$\delta x = \frac{x_1 - x_0}{N}.$$

Keeping in mind that v_n is used as an approximation to u_n, one can re-write the linear complimentary problem in 3.5.3 as (Wilmott et al. 2000, 318)(Seydel 2009, 167):

$$\begin{gathered}(v_{n+1}-2v_n+v_{n-1})(v_n-g_n)=0,\\ -v_{n+1}+2v_n-v_{n-1}\geq 0,\\ v_n\geq g_n,\end{gathered} \tag{4.6.2}$$

for $0<n<N$ and subject to the boundary conditions $v_0=v_N=0$ (Seydel 2009, 167).

Rewriting 4.6.2 in **constrained matrix notation**, one finds (Seydel 2009, 167):

$$\begin{gathered}(v-\mathbf{g})^{\mathbf{Tr}}\mathbf{B}v=\mathbf{0},\\ \mathbf{B}v\geq\mathbf{0},\\ v\geq\mathbf{g}.\end{gathered} \tag{4.6.3}$$

Matrix $\mathbf{B}$ is a $(N-1)\times(N-1)$ tridiagonal matrix and vectors v and $\mathbf{g}$ are of length $(N-1)$. These matrices can be defined as (Wilmott et al. 2000, 319)(Seydel 2009, 167):

$$\mathbf{B}=\begin{pmatrix}2&-1&0&\cdots&0\\-1&2&-1&\cdots&0\\0&-1&2&\ddots&\vdots\\\vdots&\vdots&\vdots&\ddots&-1\\0&\cdots&0&-1&2\end{pmatrix},$$

$$v=\begin{pmatrix}v_1\\\vdots\\\vdots\\\vdots\\v_{N-1}\end{pmatrix},$$

and

$$\mathbf{g}=\begin{pmatrix}g_1\\\vdots\\\vdots\\\vdots\\g_{N-1}\end{pmatrix}.$$

The solution to problem 4.6.3 can be found by solving $\mathbf{B}v=\mathbf{0}$, subject to the condition that $v\geq\mathbf{g}$ (Seydel 2009, 167).

4.7 Finite difference method for solving the American pricing problem

In a similar manner, one can now define the finite difference formulation of the linear complimentary problem in equation 3.5.40 on page 50. Since $\frac{\partial u}{\partial \tau} - \frac{\partial^2 u}{\partial x^2}$ was defined using Taylor approximations in equation 4.5.3 on page 62 and was simplified in equation 4.5.4, one finds that the finite difference formulation of the linear complimentary formulation of the American option pricing problem in equation 3.5.40 on page 50 has the following form (Wilmott et al. 2000, 326-327) (Seydel 2009, 196-170):

$$\begin{aligned}
&\{v_n^{m+1} - \theta\alpha(v_{n-1}^{m+1} - 2v_n^{m+1} + v_{n+1}^{m+1}) - v_n^m \\
&-\alpha(1-\theta)(v_{n-1}^m - 2v_n^m + v_{n+1}^m)\}(v_n^{m+1} - g_n^{m+1}) = 0, \\
&v_n^{m+1} - \theta\alpha(v_{n-1}^{m+1} - 2v_n^{m+1} + v_{n+1}^{m+1}) \geq \\
&v_n^m + \alpha(1-\theta)(v_{n-1}^m - 2v_n^m + v_{n+1}^m), \\
&v_n^m \geq g_n^m, \quad 0 < n < N, \quad 0 < m \leq M.
\end{aligned} \tag{4.7.1}$$

It is subject to the **initial condition** (Seydel 2009, 170):

$$v_n^0 = g(x_n, 0), \qquad 0 \leq n \leq N, \tag{4.7.2}$$

where $g(x,\tau)$ is once again defined in equation 3.5.41 on page 50.

Equation 4.7.1 is also subject to the following **boundary conditions** (Wilmott et al. 2000, 327)(Seydel 2009, 170):

$$v_0^m = g(x_{min}, \tau_m), \qquad 0 < m \leq M, \tag{4.7.3}$$

and

$$v_N^m = g(x_{max}, \tau_m), \qquad 0 < m \leq M. \tag{4.7.4}$$

One can re-write the finite difference formulation in equation 4.7.1 in **constrained matrix notation** (Wilmott et al. 2000, 327-328)(Seydel 2009, 170):

$$\begin{aligned}(\mathbf{C}v^{\mathbf{m+1}}-\mathbf{b^m})^{\mathbf{Tr}}(v^{\mathbf{m+1}}-\mathbf{g^{m+1}})&=\mathbf{0},\\ (\mathbf{C}v^{\mathbf{m+1}}-\mathbf{b^m})&\geq\mathbf{0},\\ (v^{\mathbf{m+1}}-\mathbf{g^{m+1}})&\geq\mathbf{0}, \qquad m=0,\ldots,M.\end{aligned} \tag{4.7.5}$$

Matrix $\mathbf{C}$ is a $(N-1)\times(N-1)$ tridiagonal matrix of the form (Wilmott et al. 2000, 328)(Seydel 2009, 170)(Wilmott et al. 1996, 169):

$$\mathbf{C}=\begin{pmatrix}(1+2\alpha\theta) & -\alpha\theta & 0 & \cdots & 0\\ -\alpha\theta & (1+2\alpha\theta) & -\alpha\theta & \cdots & 0\\ 0 & -\alpha\theta & (1+2\alpha\theta) & \ddots & \vdots\\ \vdots & \vdots & \vdots & \ddots & -\alpha\theta\\ 0 & \cdots & 0 & -\alpha\theta & (1+2\alpha\theta)\end{pmatrix}.$$

Vectors $\mathbf{b^m}$, v^m and $\mathbf{g^m}$ are of length $(N-1)$ and are defined as (Wilmott et al. 2000, 327)(Seydel 2009, 170):

$$v^m=\begin{pmatrix}v_1^m\\ \vdots\\ \vdots\\ \vdots\\ v_{N-1}^m\end{pmatrix},$$

$$\mathbf{g^m}=\begin{pmatrix}g_1^m\\ \vdots\\ \vdots\\ \vdots\\ g_{N-1}^m\end{pmatrix},$$

and

$$\mathbf{b^m}=\begin{pmatrix}b_1^m\\ \vdots\\ \vdots\\ \vdots\\ b_{N-1}^m\end{pmatrix},$$

where

$$b^m = v_n^m + \alpha(1-\theta)(v_{n-1}^m - 2v_n^m + v_{n+1}^m), \qquad 2 \leq n \leq N-2. \tag{4.7.6}$$

Unlike the rest of the terms of vector $\mathbf{b^m}$, the first and the last terms, b_1^m and b_{N-1}^m, include the boundary conditions at $n = 0$ and $n = N$. One therefore finds the following adaptation to these two b^m terms (Wilmott et al. 2000, 328)(Seydel 2009, 170):

$$b_1^m = v_1^m + \alpha(1-\theta)(g_0^m - 2v_1^m + v_2^m) + \alpha\theta g_0^{m+1}, \tag{4.7.7}$$

$$b_{N-1}^m = v_{N-1}^m + \alpha(1-\theta)(g_N^m - 2v_{N-1}^m + v_{N-2}^m) + \alpha\theta g_N^{m+1}. \tag{4.7.8}$$

The American option pricing problem which was defined as a linear complimentary problem to remove the free boundary, has now been formulated in constrained matrix finite difference notation (equation 4.7.5). Before proceeding with the task of finding the price of an American put option, first consider the different methods that can be used to solve the tridiagonal system that results from applying any of the above mentioned finite difference methods.

4.8 Methods available to solve tridiagonal systems

In many cases, the application of the finite difference method yields a large system of linear equations that can be rewritten in matrix notation as a tridiagonal system of equations. This is the case with both the obstacle and the American pricing problems.

The next section discusses two different approaches to solving these tridiagonal systems of equations. The first is a **direct elimination method**, which analytically solves the system. The second is the use of **iterative methods** (Seydel 2009, 171). For large systems, direct elimination methods such as LU-decomposition, are inefficient (Duffy 2006, 257) and in practice, large tridiagonal systems are often solved using iterative processes. (Seydel 2009, 171).

Special attention is paid when solving tridiagonal systems since large amounts of storage can be saved by not reserving space for all the elements of the matrix, but only storing the non-zero elements as three vectors (Press, Teukolsky, Vetterling & Flannery 2007, 56) (Wilmott et al. 1996, 147).

Start by considering *a general tridiagonal system of equations*. Both the direct and

itterative methods are described using this general notation in the hope that the reader will gain more insight into these methods. One can then proceed to solve the American option pricing problem written in constrained matrix notation in 4.7.5 on page 65.

Consider the following general set of linear equations written in matrix notation:

$$\mathbf{Ax} = \mathbf{b}. \tag{4.8.1}$$

One is required to solve vector $\mathbf{x}$, where $\mathbf{A}$ is a tridiagonal matrix of the form (Brandimarte 2006, 160):

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & 0 & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & \cdots & 0 \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \cdots & 0 & 0 & a_{n,n-1} & a_{n,n} \end{pmatrix}.$$

4.8.1 Direct elimination methods

The **Thomas algorithm**, a special case of Gaussian elimination is used to obtain the solution to the vector $\mathbf{x}$. Assume that the coefficient matrix, $\mathbf{A}$, is *symmetric* and *positive definite* and therefore no row pivoting is required (Kincaid & Cheney 1991, 154). Because of this, problems can arise even when working with non-singular matrices when one arrives at a zero pivot. For this reason an additional requirement is added by stating that the coefficient matrix, $\mathbf{A}$, *has to be diagonally dominant* (Press et al. 2007, 57).

Begin by re-writing matrix $\mathbf{A}$ as:

$$\mathbf{A} = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & 0 & a_n & b_n \end{pmatrix}.$$

Therefore, the tridiagonal system, $\mathbf{Ax} = \mathbf{b}$, can now be written as follows (Dukkipati 2010, 55-57):

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_3 & b_3 & c_3 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

By implementing the following steps, one can solve **x**.

Step 1:

$$y_1 = b_1,$$

$$y_i = b_i - \frac{a_i c_{i-1}}{y_{i-1}}, \qquad i = 2, \dots, n.$$

Step 2:

$$z_1 = \frac{d_1}{b_1},$$

$$z_i = \frac{d_i - a_i z_{i-1}}{y_i}, \qquad i = 2, \dots, n.$$

Step 3:

$$x_n = z_n,$$

$$x_i = z_i - \frac{c_i x_{i+1}}{y_i}, \qquad i = n-1, \dots, 1.$$

4.8.2 Iterative numerical methods

As mentioned earlier, iterative methods are particularly useful for large systems of equations. There are three main reasons for this:

- Storing all the zero values of a large tridiagonal matrix, as is the case when using certain direct methods, may lead to **unnecessary usage of computer memory**. Iterative methods allows one to store only the required information (Brandimarte 2006, 161).

- Direct methods may lead to a **delay in computational time**, due to all the additional data that has to be handled (Kincaid & Cheney 1991, 181).
- Iterative methods are usually stable and may dampen errors as the iterative process continues (Kincaid & Cheney 1991, 182).

For these reasons, financial literature often uses an iterative method, which generates a sequence of solution vectors that theoretically converges to the desired solution (Brandimarte 2006, 161). A discussion of the convergence of these iterative methods will follow later, however, for now it is important to note that as was the case with the Thomas algorithm, these iterative methods will only converge if the matrix **A** is diagonally dominant.

The following iterative methods are used to solve a tridiagonal system of equations, with a coefficient matrix of form on page 67:

- **Jacobi**. This is the simplest of the three methods. The following is an algorithm for the Jacobi method (Brandimarte 2006, 163-164):

1. Choose an initial value $\mathbf{x^k}$, where $k = 0$.
2. Decide on an acceptable error tolerance ε.
3. Compute $\mathbf{x^{k+1}}$ using:

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^k \right) \quad i = 1, \ldots, n \quad , a_{ii} \neq 0. \tag{4.8.2}$$

4. Calculate the error to attain whether or not the method has converged sufficiently. As an example, the following error can the calculated:

$$\|\mathbf{x^{k+1}} - \mathbf{x^k}\| < \varepsilon \|\mathbf{x^k}\|. \tag{4.8.3}$$

If sufficient convergence has been achieved, $\mathbf{x^{k+1}}$ is the solution and the iterative process is complete and thereby terminated. If not, the iterative process is repeated by setting $\mathbf{x^k} = \mathbf{x^{k+1}}$ and computing a new $\mathbf{x^{k+1}}$ using equation 4.8.2 and then repeating the error test of equation 4.8.3. Repeat till sought after degree of convergence has been obtained.

- **Gauss-Seidel**. Gauss-Seidel is an improved variant of the Jacobi method. This method uses each improved x_i^{k+1} **immediately**. The following is an algorithm for the Gauss-Seidel method (Brandimarte 2006, 168):

1. Choose two initial values $\mathbf{x^k}$ and $\mathbf{x^{k+1}}$, where $k = 0$.
2. Decide on an acceptable error tolerance ε.
3. Compute $\mathbf{x^{k+1}}$ using:

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right) \quad i = 1, \ldots, n \quad , a_{ii} \neq 0. \tag{4.8.4}$$

4. Calculate the error to attain whether or not the method has converged sufficiently. As an example, the following error can the calculated:

$$\|\mathbf{x^{k+1}} - \mathbf{x^k}\| < \varepsilon \|\mathbf{x^k}\|. \tag{4.8.5}$$

   If sufficient convergence has been achieved, $\mathbf{x^{k+1}}$ is the solution and the iterative process is complete and thus stopped. If not, the iterative process is repeated by setting $\mathbf{x^k} = \mathbf{x^{k+1}}$ and computing a new $\mathbf{x^{k+1}}$ using equation 4.8.4 and then repeating the error test of equation 4.8.5. Repeat till sought after degree of convergence has been obtained.

- **Successive Over Relaxation (SOR)**. This method aims to accelerate the convergence of the Gauss-Seidel method (Brandimarte 2006, 168) by allowing one to select an additional parameter, ω. A suitable choice of ω may speed up acceleration and guarantee convergence (Brandimarte 2006, 169). The following is an algorithm for Successive Over Relaxation method (Brandimarte 2006, 169):

  1. Choose two initial values $\mathbf{x^k}$ and $\mathbf{x^{k+1}}$, where $k = 0$.
  2. Decide on an acceptable error tolerance ε.
  3. Select an appropriate ω.
  4. Repeat for $i = 0, \ldots, n$:
     Compute y_i^{k+1} using:

$$y_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right) \quad , a_{ii} \neq 0. \tag{4.8.6}$$

     Compute x_i^{k+1} using:

$$x_i^{k+1} = \omega y_i^{k+1} + (1 - \omega) x_i^k \tag{4.8.7}$$

  5. Calculate the error to attain whether or not the method has converged sufficiently. As an example, the following error can the calculated:

$$\|\mathbf{x^{k+1}} - \mathbf{x^k}\| < \varepsilon \|\mathbf{x^k}\|. \tag{4.8.8}$$

     If sufficient convergence has been achieved, $\mathbf{x^{k+1}}$ is the solution and the iterative process is complete and is stopped. If not, repeat the iterative process by setting $\mathbf{x^k} = \mathbf{x^{k+1}}$ and computing a new $\mathbf{x^{k+1}}$ using equations 4.8.6 and 4.8.7 and then repeating the error test of equation 4.8.8. Repeat till sought after degree of convergence has been obtained.

Convergence of iterative methods

- *Jacobi*
  Theorem 1 - If $\mathbf{A}$ is diagonally dominant, then the sequence produced by the Jacobi iteration converges to the solution of $\mathbf{Ax} = \mathbf{b}$, for any starting vector (Kincaid & Cheney 1991, 185).

- *Gauss-Seidel*

Start by decomposing matrix **A**, using the splitting matrix, **Q** and re-write 4.8.1 on page 67 as (Kincaid & Cheney 1991, 183):

$$\mathbf{Qx} = (\mathbf{Q} - \mathbf{A})\mathbf{x} + \mathbf{b}. \tag{4.8.9}$$

Iteratively this can be written as:

$$\mathbf{Q}x^k = (\mathbf{Q} - \mathbf{A})x^{k-1} + b^k. \tag{4.8.10}$$

To solve **x**, one therefore solves (Kincaid & Cheney 1991, 183):

$$\mathbf{x} = (\mathbf{I} - \mathbf{Q^{-1}A})\mathbf{x} + \mathbf{Q^{-1}b}. \tag{4.8.11}$$

Corollary - The iterative process in 4.8.10 will produce a sequence converging to the solution of $\mathbf{Ax} = \mathbf{b}$, for any starting vector if (Kincaid & Cheney 1991, 189):

$$\rho(\mathbf{I} - \mathbf{Q^{-1}A}) < 1, \tag{4.8.12}$$

where ρ is the spectral radius.

Theorem 2 - If **A** is diagonally dominant, then the Gauss-Seidel method converges to the solution of $\mathbf{Ax} = \mathbf{b}$, for any starting vector (Kincaid & Cheney 1991, 189).

- *Successive Over Relaxation (SOR)*

  Begin by choosing **Q** as $\alpha\mathbf{D} - \mathbf{C}$, where α is a real parameter, **D** is a positive definite Hermitian matrix and **C** is any matrix that satisfies:

$$\mathbf{C} + \mathbf{C}^* = \mathbf{D} - \mathbf{A}, \tag{4.8.13}$$

  where $\mathbf{C}^*$ is the conjugate transpose of **C** (Kincaid & Cheney 1991, 192).

  Theorem 3 - If **A** is a positive definite Hermitian, **Q** is non-singular and $\alpha > \frac{1}{2}$, then the SOR iteration converges for any starting vector (Kincaid & Cheney 1991, 192):

The prerequisites for convergence of these iterative methods can be manipulated by implementing an adapted discretization resulting in irregular grid interval lengths. This however falls beyond the scope of this dissertation.

This concludes the discussion on the general iterative methods that can be implemented to solve linear systems of equations. Now one can return to the set of matrices in equation 4.7.5 on page 65 that must be used to price an American put option.

4.9 Projected Successive Over Relaxation method for American puts

The **projected SOR method** (PSOR) is used when solving the constrained matrix problems in 4.6.3 and 4.7.5 on pages 63 and 65 respectively. This method has one additional feature, when compared to the classical SOR method discussed in the section above (Wilmott et al. 2000, 320). It ensures that every element of the vector, v^{m+1}, adheres to the constraint $v_i^{m+1} \geq g_i^{m+1}$ (Wilmott et al. 2000, 320).

Also note that the grid-axes are numbered from $0,\ldots,M$ and $0,\ldots,N$ for the τ and x respectively. *When programming, one therefore has to keep in mind that indices only start at 1 and therefore the iterative process has to stop at $M+1$ in stead of M.*

To solve the American put option pricing problem, one is required to solve the following matrix equation obtained from 4.7.5 on page 65:

$$\mathbf{C}v^{\mathbf{m+1}} = \mathbf{b^m},$$

subject to the constraint:

$$v^{\mathbf{m+1}} \geq \mathbf{g^{m+1}}.$$

The following is a detailed algorithm of the Projected Successive Over Relaxation method (PSOR) that is used in the American pricing problem. Sections of this algorithm can be found in *Option Pricing: Mathematical models and computation* by P. Wilmott (Wilmott et al. 2000, 330). No evidence was found in literature of such a comprehensive algorithm. The algorithm solves the first unknown vector v^1, at time increment $m = 1$. It is important to note that because *one only has known data at the initial time node, $m = 0$ and at the boundaries, $n = 0$ and $n = N$*, the algorithm has to be repeated for all unknown v^m vectors at time nodes $m = 1,\ldots,M$.

1. Set **m = 0**.
2. Calculate **KNOWN initial values** ($m = 0$) of the finite difference grid using equation 4.7.2 on page 64, where g is defined on page 50 as:

$$g(x,\tau) = e^{\frac{1}{4}(\kappa+1)^2\tau}\max[e^{\frac{1}{2}(\kappa-1)x-\frac{1}{2}(\kappa+1)x}, 0]. \tag{4.9.1}$$

   One now has the the following known vectors to one's disposal:

$$v^m = v^0 = \begin{pmatrix} v_1^0 \\ \vdots \\ \vdots \\ \vdots \\ v_{N-1}^0 \end{pmatrix},$$

and

$$\mathbf{g^m} = \mathbf{g^0} = \begin{pmatrix} g_1^0 \\ \vdots \\ \vdots \\ \vdots \\ g_{N-1}^0 \end{pmatrix}.$$

3. Calculate **KNOWN boundary values** ($n = 0$) and ($n = N$) of grid using equations 4.7.3 and 4.7.4 on page 64. One now has additional known values:

$$v_0 = \begin{pmatrix} v_0^0 \\ \vdots \\ \vdots \\ \vdots \\ v_0^M \end{pmatrix},$$

and

$$v_N = \begin{pmatrix} v_N^0 \\ \vdots \\ \vdots \\ \vdots \\ v_N^M \end{pmatrix}.$$

4. Compute g^{m+1}, where $m = 0$, using 4.9.1:

$$\mathbf{g^{m+1}} = \mathbf{g^1} = \begin{pmatrix} g_1^1 \\ \vdots \\ \vdots \\ \vdots \\ g_{N-1}^1 \end{pmatrix}.$$

5. **Choose an initial vector, $\mathbf{x^{old}}$**. As in the case of the iterative processes discussed earlier, the vector $\mathbf{x^{old}}$ will iteratively be used to approximate the unknown vector v^1. Using the constraint given in equation 4.7.5 on page 65, $(v^{m+1} - \mathbf{g^{m+1}}) \geq \mathbf{0}$, one selects $\mathbf{x^{old}}$ as:

$$\mathbf{x^{old}} = \max[v^0, \mathbf{g^1}], \tag{4.9.2}$$

and finds an initial $\mathbf{x^{old}}$:

$$\mathbf{x^{old}} = \begin{pmatrix} x_1^{old} \\ \vdots \\ \vdots \\ \vdots \\ x_{N-1}^{old} \end{pmatrix}.$$

6. **Compute $\mathbf{b^m}$**, using equations 4.7.6, 4.7.7 and 4.7.8 on page 66. One now has a vector of the form:

$$\mathbf{b^m} = \mathbf{b^0} = \begin{pmatrix} b_1^0 \\ \vdots \\ \vdots \\ \vdots \\ b_{N-1}^0 \end{pmatrix}.$$

7. Choose variable, iter $= 1$ (this will allow access to the While-loop).

8. Choose variable, error $= 0,00001$ (for example).

9. Choose relaxation parameter, $\omega = 1$ (for example).

10. **While iter = 1 do**:

(Enter into a While-loop that will iteratively solve $v^{m+1} = v^1$ and will continue until the solution has converged sufficiently).

(a) **Compute a y value**. Notice that the following formulae correspond to the formulae given at the Successive Over Relaxation (SOR) discussion on page 72. In this case, the entries of the main diagonal of matrix **C** on page 65, $(1+2\alpha\theta)$, are used.

- **For k = 1:N-1** (spans all the unknown grid points at a single time interval)

  k = 1:

$$y = \frac{b_k^0 + \alpha\theta(x_{k+1}^{old})}{(1+2\alpha\theta)}. \tag{4.9.3}$$

  Because the Projected Successive Over Relaxation (PSOR) is an adaptation of the Gauss-Seidel method, the proceeding x^{new}-value is used to compute the following one, one first needs to compute $x_k^{new} = x_1^{new}$. This value must satisfy the constraint $(v^{m+1} - \mathbf{g^{m+1}}) \geq \mathbf{0}$ on page 65 and therefore:

$$x_k^{new} = max[g_{k+1}^1, x_k^{old} + \omega(y - x_k^{old})]. \tag{4.9.4}$$

  k = 2: to N-2:

$$y = \frac{b_k^0 + \alpha\theta(x_{k-1}^{new} + x_{k+1}^{old})}{(1+2\alpha\theta)} \tag{4.9.5}$$

$$x_k^{new} = max[g_{k+1}^1, x_k^{old} + \omega(y - x_k^{old})]. \tag{4.9.6}$$

  k = N-1:

$$y = \frac{b_k^0 + \alpha\theta x_{k-1}^{new}}{(1+2\alpha\theta)} \tag{4.9.7}$$

$$x_k^{new} = max[g_{k+1}^1, x_k^{old} + \omega(y - x_k^{old})]. \tag{4.9.8}$$

- **end (For)**

One now has a vector, $\mathbf{x^{new}}$:

$$\mathbf{x^{new}} = \begin{pmatrix} x_1^{new} \\ \vdots \\ \vdots \\ \vdots \\ x_{N-1}^{new} \end{pmatrix}.$$

(b) **Test convergence**. This will determine whether the While-loop has converged sufficiently or whether the iterative process needs to be repeated.

- **If** $\| \mathbf{x^{old}} - \mathbf{x^{new}} \| \leq$ error
  then $iter = 0$ (this will stop the While-loop)
  else
  $\mathbf{x^{old}} = \mathbf{x^{new}}$ and one re-calculates $\mathbf{x^{new}}$ by re-entering the For-loop described above.
- **end (If)**

end (While)

After the While-loop, the solution has converged sufficiently to $\mathbf{x^{new}}$.

11. $\mathbf{x^{new}}$ represents solution at time interval, $m = 1$. One can now transfer the values of vector $\mathbf{x^{new}}$ to the value matrix v.

$$v^1 = \mathbf{x^{new}} \tag{4.9.9}$$

12. Repeat While-loop and test for convergence for the other time increments, $m = 2, \ldots, M$. The final result is a matrix v, containing all the solutions.

4.10 Algorithm to solve put price under constant volatility using PSOR

Keeping in mind that when programming, the numbering of nodes will start at 1, as opposed to the numbering system used thus far, $m = 0, \ldots, M$.

1. **INITIAL PARAMETERS**

   (a) Given values K, T (per annum), r (per annum), σ (per annum).

   (b) Given current value of underlying stock price, $S_{current}$.

(c) Choose number of intervals on each axis of the grid:

$m = M$, number of intervals on τ-axis.

$n = N$, number of intervals on x-axis.

2. **CALCULATING AMERICAN PUT OPTION PRICES**

(a) Compute κ using equation 3.5.14 on page 46,

$$\kappa = \frac{2r}{\sigma^2}. \tag{4.10.1}$$

(b) Choose $x_{\min}$ and x_{max} values, where as mentioned earlier, $x_{min} < 0$ and $x_{max} > 0$.

(c) Compute δx, the length of an interval on the x-axis:

$$\delta x = \frac{x_{max} - x_{min}}{N}. \tag{4.10.2}$$

(d) Compute $\tau_{max} = 0,5\sigma^2 T$.

(e) Compute $\delta\tau$, the length of an interval on the τ-axis:

$$\delta\tau = \frac{\tau\max}{M}. \tag{4.10.3}$$

(f) Choose numerical parameter, θ. This value dictates whether the Explicit, Implicit of Crank-Nicolson finite difference method is to be used.

(g) Compute α, using equation 4.3.5 on page 56,

$$\alpha = \frac{\delta\tau}{(\delta x)^2}. \tag{4.10.4}$$

(h) Test if method chosen is stable:

If $\theta < 0,5$ and if $\alpha > 0,5$, then method is unstable and either θ or number of intervals need to be altered.

(i) Discretize both x and τ axes:

$$x = (x_{min} : \delta x : x_{\max}),$$

$$\tau = (0 : \delta\tau : \tau_{max}).$$

(j) Define in-line function g using 4.9.1 on page 74.

(k) Compute matrix **g**. These values are to be used in the PSOR routine. By doing so one doesn't have to re-calculate a vector **g** repeatedly and one can only refer to the relevant column needed in the PSOR iteration. One now has:

$$\mathbf{g} = \begin{pmatrix} g_0^0 & g_0^1 & g_0^2 & \cdots & g_0^{M-1} & g_0^M \\ g_1^0 & g_1^1 & g_1^2 & \cdots & g_1^{M-1} & g_1^M \\ g_2^0 & g_2^1 & g_2^2 & \cdots & g_2^{M-1} & g_2^M \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ g_N^0 & g_N^1 & g_N^2 & \cdots & g_N^{M-1} & g_N^M \end{pmatrix}.$$

(l) Define initial matrix v with zero values.

(m) Complete PSOR algorithm, discussed in section 4.10. This will result in a matrix v, containing the solutions for variables x and τ. One now has to convert these values to values related to the variables S and t.

3. **TRANSFORM TO S, t and P^{Am} (OPTION PRICE) VALUES**

(a) Convert x and τ values on the axes to S and t values using equation 3.5.8 on page 45,

$$S = Ke^x, \qquad t = T - \frac{\tau}{\frac{1}{2}\sigma^2}. \tag{4.10.5}$$

(b) Convert matrix v, *containing values which are approximations to values in matrix* **u**, to a matrix $\mathbf{P^{Am}}$, using equations on pages 45 and 47:

$$f(x,\tau) = e^{-\frac{1}{2}(\kappa-1)x-\frac{1}{4}(\kappa+1)^2\tau}u(x,\tau), \tag{4.10.6}$$

$$f(x,\tau) = \frac{1}{K}P^{Am}(S,t) = \frac{1}{K}P^{Am}\left(Ke^x, T - \frac{\tau}{\frac{1}{2}\sigma^2}\right), \tag{4.10.7}$$

and therefore,

$$P^{Am}(S,t) = Kf(x,\tau) = Ke^{-\frac{1}{2}(\kappa-1)x-\frac{1}{4}(\kappa+1)^2\tau}u(x,\tau). \tag{4.10.8}$$

Finally, one obtains a matrix containing a broad range of American put option values that is suited to each individual stock price at each time interval:

$$\mathbf{P^{Am}} = \begin{pmatrix} P_0^0 & P_0^1 & P_0^2 & \cdots & P_0^{M-1} & P_0^M \\ P_1^0 & P_1^1 & P_1^2 & \cdots & P_1^{M-1} & P_1^M \\ P_2^0 & P_2^1 & P_2^2 & \cdots & P_2^{M-1} & P_2^M \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P_N^0 & P_N^1 & P_N^2 & \cdots & P_N^{M-1} & P_N^M \end{pmatrix}.$$

4. **COMPUTE EARLY EXERCISE BOUNDARY**

Because the early exercise boundary is valid for all t values, where $0 \leq t \leq T$, one needs to compute an S_f value at each node on the time axis.

(a) Create initial vector, $\mathbf{S_f}$ of length $M+1$ containing zeros.

(b) Choose err $= 0,000001$ (for example). A value slightly larger than zero.

(c) **For j = 0:M** (for all time-axis values)

- Exercise is optimal when $S < S_f$ and $P^{Am} = max[K-S,0]$. Refer to equation 3.4.7 on page 41. Therefore, one will exercise when $P^{Am} - K + S = 0$.
- One now wants a value $S(i)$, where $0 \leq i \leq N$ (all the x values at one time increment), in each column j (one time increment), of the matrix, $\mathbf{P^{Am}}$. Here, $P_i^j - K + S(i) < err$, for this is the last option value where one will exercise the option. Because at $P_i^j \geq max[K - S(i), 0]$, one will hold on to the option and $S(i) > S_f$ (See equation 3.4.8 on page 41). Therefore,

$$i = \text{find}(\text{abs}(P^{Am}(:,j) - K + S) < \text{err}, 1, '\text{last}').$$

- This is the stock price value that separates the exercise from the non-exercise region and therefore, $S_f(j) = S(i)$.

end (For)

5. **FIND CURRENT OPTION VALUE**

Interpolate S and P^{Am} time zero values to find option value corresponding to current stock price, $S_{current}$. This is done in cases where the $S_{current}$ value does not fall precisely on the S grid. For example, spline interpolation can be used.

$$P^{Am}_{current} = interpl(S, P^{Am}(:,0), S_{current}, 'spline').$$

Chapter 5

American put option pricing problem under stochastic volatility

5.1 Introduction

In the first part of this dissertation, American put options were priced under constant volatility. This was done in accordance with the assumptions made by the Black-Scholes model. However, recent world events have proven that markets are indeed volatile and subsequently current research trends focus on the development of mathematical models that take fluctuating volatility into account. This has led to the development of numerous sophisticated models that can be used to value an option (Ikonen & Tiovanen 2008, 105):

- Value and time dependant volatility functions.
- Jump processes for the value.
- Combinations of value and time dependant volatility functions and jump processes.
- Stochastic volatility models.
- Stochastic volatility models with jumps (Ikonen & Tiovanen 2008, 105).

The second part of this dissertation addresses one of the shortcomings of the Black-Scholes model by exploring *stochastic volatility models*. As was the case previously the Black-Scholes partial differential equation on page 29 can be adapted to compensate for an expanded asset price model by re-writing the problem in linear complimentary form. This linear complimentary problem is then discretized using the appropriate finite difference formulas and solved numerically.

5.2 Black-Scholes model under stochastic volatility

The stochastic volatility model discussed in this dissertation is the Heston model. This model generalizes the Black-Scholes equation by introducing stochastic volatility and therefore compensates for the shortcomings of Black-Scholes model's assumptions on constant volatility. This generalization is achieved by modelling the price of an underlying asset in a more realistic way. As was the case with constant volatility (see equation 3.2.1 on page 24), start by constructing an **asset price model of the underlying stock price**. The following stochastic differential equations are assumed to govern the asset *price process*, S and its *variance*, y (Duffy 2006, 240-241):

$$dS = \mu S dt + \sqrt{y} S dW_1, \tag{5.2.1}$$

$$dy = \alpha[\beta - y]dt + \gamma\sqrt{y}dW_2, \tag{5.2.2}$$

where

- S - Stock price at time t.
- μ - Constant expected rate of return of stock (drift).
- y - Variance.
- W_1 - A Wiener process.
- W_2 - A second Wiener process related to the first by correlation coefficient, ρ.
- γ - Volatility of the volatility.
- β where $(0 < \beta)$ - Long term variance.
- α where $(\beta < \alpha)$ - Rate of mean reversion.

The correlation between the two Wiener processes is given as (Duffy 2006, 241):

$$dW_1 dW_2 = \rho dt. \tag{5.2.3}$$

It can also be described as the correlation between the price of the underlying asset and its variance (Ikonen & Tiovanen 2008, 106).

Now the Black-Scholes partial differential equation has to be adjusted to allow for the

changes made to the asset price model (Oosterlee 2003, 168-169). Because American put options are the focus of this dissertation, consider the Black-Scholes partial differential inequality which was derived on page 32 as:

$$\frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\frac{\partial^2 P^{Am}}{\partial S^2}\sigma^2 S^2 + rS\frac{\partial P^{Am}}{\partial S} - rP^{Am} \leq 0. \tag{5.2.4}$$

If 5.2.4 is changed to take the asset price model in 5.2.1 and 5.2.2 into account, the result is a two-dimensional parabolic partial differential inequality (Ikonen & Tiovanen 2009, 302):

$$\begin{aligned}\frac{\partial P^{Am}}{\partial t} + \frac{1}{2}\left(yS^2\frac{\partial^2 P^{Am}}{\partial S^2} + 2\rho\gamma yS\frac{\partial^2 P^{Am}}{\partial S\partial y} + \gamma^2 y\frac{\partial^2 P^{Am}}{\partial y^2}\right) + rS\frac{\partial P^{Am}}{\partial S} + \\ (\alpha(\beta - y) - \vartheta\gamma\sqrt{y})\frac{\partial P^{Am}}{\partial y} - rP^{Am} \leq 0,\end{aligned} \tag{5.2.5}$$

where ϑ is the market price of the risk (Oosterlee 2003, 168).

It is important to notice that *the same path as in the case of constant volatility, where the Black-Scholes equation was transformed into the one dimensional heat equation is not followed. Instead, one uses only one transformation to change the problem from a backward equation, requiring final conditions, to a forward equation that requires initial conditions* (Oosterlee 2003, 169) (Higham 2009, 257). This concept was covered in paragraph 3.1.

Equation 5.2.5 is transformed into a forward equation by defining a variable τ as:

$$\tau = T - t. \tag{5.2.6}$$

New symbols are also introduced for simplicity. Let u and x denote the American option price, P^{Am} and the underlying stock price, S, respectively. One can now rewrite equation 5.2.5 and define the **Heston operator** as (Ikonen & Tiovanen 2008, 106) (Duffy 2006, 239-241) (Ikonen & Tiovanen 2009, 302):

$$\begin{aligned}Lu = \frac{\partial u}{\partial \tau} - \frac{1}{2}yx^2\frac{\partial^2 u}{\partial x^2} - \rho\gamma yx\frac{\partial^2 u}{\partial x\partial y} - \frac{1}{2}\gamma^2 y\frac{\partial^2 u}{\partial y^2} - rx\frac{\partial u}{\partial x} \\ - (\alpha(\beta - y) - \vartheta\gamma\sqrt{y})\frac{\partial u}{\partial y} + ru \geq 0.\end{aligned} \tag{5.2.7}$$

The option pricing problem is defined on the unbounded domain (Ikonen & Tiovanen 2008, 107) (Ikonen & Tiovanen 2009, 302):

$$\{(x,y,\tau) : x \geq 0, y \geq 0, \tau \in [0,T]\}. \tag{5.2.8}$$

The values on the x and y axes need to be truncated. However, the Heston model is not entirely clear on how large the computational domain needs to be to ensure a sufficiently small truncation error (Ikonen & Tiovanen 2008, 107). Therefore, redefine the domain in 5.2.8 as (Ikonen & Tiovanen 2009, 302):

$$\{(x,y,\tau) : 0 = 0 \leq x \leq x_{max}, 0 \leq y \leq y_{max}, \tau \in [0,T]\}, \tag{5.2.9}$$

where x_{max} and y_{max} are sufficiently large numbers. Remembering that for a put option, the payoff is (Ikonen & Tiovanen 2009, 302):

$$g(x) = max[K-x,0], \tag{5.2.10}$$

one can now proceed to investigate the initial and boundary conditions of the pricing problem.

The **initial condition** implies that (Ikonen & Tiovanen 2008, 106) (Ikonen & Tiovanen 2009, 303):

$$u(x,y,0) = g(x) = max[K-x,0] \qquad (x,y) \in [0,x_{max}] \times [0,y_{max}]. \tag{5.2.11}$$

Keep in mind that due to the transformation using τ, the problem is solved by providing a $\tau = 0$ value, but that this value in actual fact corresponds to the option price at time T. Therefore, when the pricing problem is numerically solved for all unknown τ values, the value obtained at $\tau = T$ corresponds to the option price at $t = 0$ (current price).

Boundary conditions at (Ikonen & Tiovanen 2008, 107):

- x = 0, Dirichlet boundary condition (Ikonen & Tiovanen 2009, 303):

$$u(0,y,\tau) = g(0) = K \qquad (y,\tau) \in [0,y_{max}] \times [0,T]. \tag{5.2.12}$$

- x = x_{max}, Neumann boundary condition (Ikonen & Tiovanen 2008, 107) (Ikonen & Tiovanen 2009, 303):

$$\frac{\partial u}{\partial x}(x_{max}, y, \tau) = 0 \qquad (y, \tau) \in [0, y_{max}] \times [0, T]. \tag{5.2.13}$$

- y = 0, Dirichlet boundary condition (Ikonen & Tiovanen 2004, 5) (Clarke & Parrott 1999, 180):

$$u(x, 0, \tau) = g(x) = max[K - x, 0] \qquad (x, \tau) \in [0, x_{max}] \times [0, T]. \tag{5.2.14}$$

- y = y_{max} (Ikonen & Tiovanen 2009, 303) (Ikonen & Tiovanen 2008, 107):

$$\frac{\partial u}{\partial y}(x, y_{max}, \tau) = 0 \qquad (x, \tau) \in [0, x_{max}] \times [0, T]. \tag{5.2.15}$$

Due to the early exercise facility of an American option, an additional early exercise constraint is included (Ikonen & Tiovanen 2009, 303):

$$u(x, y, \tau) \geq g(x) \qquad (x, y, \tau) \in [0, x_{max}] \times [0, y_{max}] \times [0, T]. \tag{5.2.16}$$

The **linear complimentary problem for the American option price under stochastic volatility** can now be defined as (Ikonen & Tiovanen 2008, 107) (Ikonen & Tiovanen 2009, 303):

$$\begin{aligned} (Lu)(u - g) &= 0, \\ Lu &\geq 0, \\ u - g &\geq 0. \end{aligned} \tag{5.2.17}$$

5.3 Finite difference method

The linear complimentary problem in 5.2.17 is to be solved using the finite difference method. This requires the discretization of the Heston operator in 5.2.7 on page 84 (Ikonen & Tiovanen 2009, 304). **The spatial derivatives are discretized using a seven point stencil** and the time axis is discretized using a uniform space-time finite difference grid on the domain described in 5.2.9 (Ikonen & Tiovanen 2009, 304).

- Number of internal nodes on the x-axis $= m$,

  $\Delta x = \frac{x_{max}}{m+1}$.

- Number of internal nodes on the y-axis $= n$,

  $\Delta y = \frac{y_{max}}{n+1}$.

- Number of internal nodes on the τ-axis $= l$,

  $\Delta \tau = \frac{T}{l+1}$.

- The following grid point notation is used:

  $u_{i,j}^{(k)} \approx u(x_i, y_j, \tau_k) = u(i\Delta x, j\Delta y, k\Delta\tau)$,
  where $i = 0, \ldots, m+1$, $j = 0, \ldots, n+1$ and $k = 0, \ldots, l+1$ (Ikonen & Tiovanen 2009, 304).

Fig. 5.3.1 on page 88 is a visual representation of the grid obtained after the space discretization of equation 5.2.7 on page 84. The blue dots represent the known values on the $x = 0$ boundary. These values are obtained using equation 5.2.12. The red dots represent the known values on the $y = 0$ boundary and are obtained from 5.2.14. The green dots represent unknown values that are used to calculate the option price on the boundary $y = y_{max}$. The details of this process will be discussed in the next section, but for now one can mention that the idea behind solving the boundary values at $y = y_{max}$ is to approximate 5.2.15 on page 86 using a central difference formulae, therefore using the two adjacent nodes. This too is the case on the $x = x_{max}$ boundary, where the pink dots represent the adjacent nodes after approximating 5.2.13. The use of a central difference formula to approximate 5.2.15 and 5.2.13 results in the use of ghost points (Ikonen & Tiovanen 2004, 8). These points fall outside the grid dimensions and as just mentioned, one will learn how to deal with these ghost or fictitious grid points in the following section.

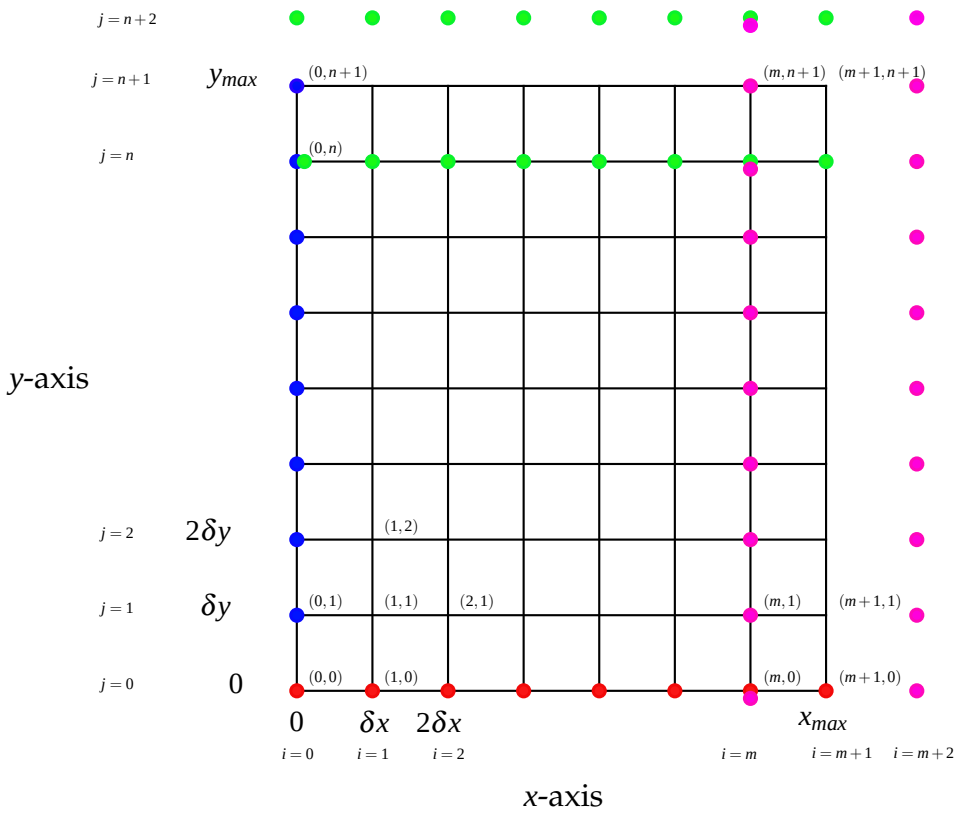

Figure 5.3.1: $x - \tau$ - grid after discretization

The coloured dots are discussed on page 87.

5.3.1 Space discretization

All the partial derivatives in the Heston operator on page 84, have variable coefficients and in parts of the domain, a first order derivative dominates a second order one (Ikonen & Tiovanen 2009, 304). Notice that the operator also contains a second-order cross-derivative term, $\rho\gamma yx\frac{\partial^2 u}{\partial x \partial y}$, and therefore it is difficult to construct a grid with good properties and accuracy (Ikonen & Tiovanen 2009, 304).

Normal finite difference approximations may result in some *positive off-diagonal elements in the coefficient matrix*, due to the *cross-derivative and dominating first-order derivative terms*. These positive elements may lead to oscillations in the solution (Ikonen & Tiovanen 2009, 304).

To remedy the problem of positive off-diagonal elements, one needs to construct a *strictly diagonal dominant matrix* with *positive diagonal elements* and *non-positive off-diagonal elements*, called a M-matrix (Ikonen & Tiovanen 2009, 304).

The **first and second-order spatial derivatives** in 5.2.7, with the exception of the cross-derivative term, are approximated with **standard second-order accurate central finite difference formulas** (Ikonen & Tiovanen 2009, 305). In cases where a first-order derivative term dominates the related second-order derivative term, one needs to increase the second-order derivative term's coefficient, thereby avoiding positive off-diagonal elements (Ikonen & Tiovanen 2009, 305).

- **First-order derivative terms** (Ikonen & Tiovanen 2009, 305).

$$\begin{aligned} \delta_x u_{i,j} &= \frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}, \\ \delta_y u_{i,j} &= \frac{\partial u}{\partial y} = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}. \end{aligned} \tag{5.3.1}$$

- **Second-order derivative terms** (Ikonen & Tiovanen 2009, 305).

$$\begin{aligned} \delta_x^2 u_{i,j} &= \frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}, \\ \delta_y^2 u_{i,j} &= \frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}. \end{aligned} \tag{5.3.2}$$

- **Second order cross-derivative term**.

  Begin by assuming that the coefficient of this term, $\rho\gamma xy$, is non-positive (Ikonen & Tiovanen 2009, 305). By expanding the Taylor series and truncating, one finds (Ikonen & Tiovanen 2009, 305):

$$\begin{aligned} u(x_{i+1},y_{j+1}) &\approx u+\Delta x\frac{\partial u}{\partial x}+\Delta y\frac{\partial u}{\partial y} \\ &+\frac{1}{2}\left(\Delta x^2\frac{\partial^2 u}{\partial x^2}+2\Delta x\Delta y\frac{\partial^2 u}{\partial xy}+\Delta y^2\frac{\partial^2 u}{\partial y^2}\right), \\ u(x_{i-1},y_{j-1}) &\approx u-\Delta x\frac{\partial u}{\partial x}-\Delta y\frac{\partial u}{\partial y} \\ &+\frac{1}{2}\left(\Delta x^2\frac{\partial^2 u}{\partial x^2}+2\Delta x\Delta y\frac{\partial^2 u}{\partial xy}+\Delta y^2\frac{\partial^2 u}{\partial y^2}\right), \end{aligned} \tag{5.3.3}$$

  where u and it's derivatives on the right hand side of these equations are evaluated in the grid point (x_i,y_j).

  By adding these two equations, simplifying and rearranging the terms, one obtains an approximation for $\dfrac{\partial^2 u}{\partial x\partial y}$ (Ikonen & Tiovanen 2009, 306):

$$\begin{aligned} \frac{\partial^2 u}{\partial x\partial y} &\approx \frac{1}{2\Delta x\Delta y}\left[u(x_{i+1},y_{j+1})-2u(x_i,y_j)+u(x_{i-1},y_{j-1})\right] \\ &-\frac{\Delta x}{2\Delta y}\frac{\partial^2 u}{\partial x^2}-\frac{\Delta y}{2\Delta x}\frac{\partial^2 u}{\partial y^2}. \end{aligned} \tag{5.3.4}$$

Using the approximation of the second order derivative terms in 5.3.2 and the approximation of the second-order cross derivative term in 5.3.4, one can re-write these terms in 5.2.7 as (Ikonen & Tiovanen 2009, 306):

$$
\begin{aligned}
&-\frac{1}{2}y_j x_i^2 \frac{\partial^2 u}{\partial x^2} - \rho\gamma y_j x_i \frac{\partial^2 u}{\partial x \partial y} - \frac{1}{2}\gamma^2 y_j \frac{\partial^2 u}{\partial y^2} \approx \\
&\left[-\frac{1}{2}y_j x_i^2 + \frac{\rho\gamma y_j x_i}{2}\frac{\Delta x}{\Delta y}\right]\frac{\partial^2 u}{\partial x^2} + \left[-\frac{1}{2}\gamma^2 y_j + \frac{\rho\gamma y_j x_i}{2}\frac{\Delta y}{\Delta x}\right]\frac{\partial^2 u}{\partial y^2} \\
&-\frac{\rho\gamma y_j x_i}{2\Delta x \Delta y}\left[u(x_{i+1}, y_{j+1}) - 2u(x_i, y_j) + u(x_{i-1}, y_{j-1})\right].
\end{aligned}
\tag{5.3.5}
$$

Finally, by substituting the central differences in 5.3.1 and 5.3.2, together with 5.3.5 into 5.2.7 (Ikonen & Tiovanen 2009, 306), one obtains:

$$
\begin{aligned}
&\frac{\partial u}{\partial \tau} + \left[-\frac{1}{2}y_j x_i^2 + \frac{\rho\gamma y_j x_i}{2}\frac{\Delta x}{\Delta y} + a_{add}\right]\frac{\partial^2 u}{\partial x^2} + \left[-\frac{1}{2}y_j\gamma^2 + \frac{\rho\gamma y_j x_i}{2}\frac{\Delta y}{\Delta x} + c_{add}\right]\frac{\partial^2 u}{\partial y^2} \\
&-rx_i\frac{\partial u}{\partial x} - \left[\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}\right]\frac{\partial u}{\partial y} \\
&-\frac{\rho\gamma y_j x_i}{2\Delta x \Delta y}\left[u_{i+1,j+1} - 2u_{i,j} + u_{i-1,j-1}\right] + ru_{i,j} \geq 0,
\end{aligned}
\tag{5.3.6}
$$

where a_{add} and c_{add} are additional coefficients chosen to ensure that all off-diagonal elements of the coefficient matrix are non-positive (Ikonen & Tiovanen 2009, 306). Therefore (Ikonen & Tiovanen 2009, 306):

$$
\begin{aligned}
a_{add} &= min\left(\frac{1}{2}y_j x_i^2 - \frac{\rho\gamma y_j x_i}{2}\frac{\Delta x}{\Delta y} - rx_i\frac{\Delta x}{2}, \frac{1}{2}y_j x_i^2 - \frac{\rho\gamma y_j x_i}{2}\frac{\Delta x}{\Delta y} + rx_i\frac{\Delta x}{2}, 0\right), \\
c_{add} &= min\left(\frac{1}{2}y_j x_i^2 - \frac{\rho\gamma y_j x_i}{2}\frac{\Delta y}{\Delta x} - [\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}]\frac{\Delta y}{2}, \right. \\
&\left. \frac{1}{2}y_j x_i^2 - \frac{\rho\gamma y_j x_i}{2}\frac{\Delta x}{\Delta y} + [\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}]\frac{\Delta y}{2}, 0.\right)
\end{aligned}
\tag{5.3.7}
$$

Using 5.3.1 and 5.3.2, one can re-write 5.3.6 to define a seven point discretization stencil in points $u_{i,j}, u_{i-1,j}, u_{i+1,j}, u_{i,j-1}, u_{i,j+1}, u_{i+1,j+1}$ and $u_{i-1,j-1}$ referred to in (Ikonen & Tiovanen 2009, 306), but written in detail as:

$$\begin{aligned}
&\frac{\partial u}{\partial \tau} + \left[\frac{y_j x_i^2}{\Delta x^2} - \frac{\rho\gamma y_j x_i}{\Delta y \Delta x} - \frac{2a_{add}}{\Delta x^2} + \frac{y_j \gamma^2}{\Delta y^2} - \frac{2c_{add}}{\Delta y^2} + r\right] u_{i,j} \\
&\quad + \left[-\frac{y_j x_i^2}{2\Delta x^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{a_{add}}{\Delta x^2} + \frac{r x_i}{2\Delta x}\right] u_{i-1,j} \\
&\quad + \left[-\frac{y_j x_i^2}{2\Delta x^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{a_{add}}{\Delta x^2} - \frac{r x_i}{2\Delta x}\right] u_{i+1,j} \\
&+ \left[-\frac{y_j \gamma^2}{2\Delta y^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{c_{add}}{\Delta y^2} + \frac{[\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}]}{2\Delta y}\right] u_{i,j-1} \\
&+ \left[-\frac{y_j \gamma^2}{2\Delta y^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{c_{add}}{\Delta y^2} - \frac{[\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}]}{2\Delta y}\right] u_{i,j+1} \\
&\quad + \left[-\frac{\rho\gamma y_j x_i}{2\Delta y \Delta x}\right] u_{i+1,j+1} + \left[-\frac{\rho\gamma y_j x_i}{2\Delta y \Delta x}\right] u_{i-1,j-1} \geq 0,
\end{aligned} \tag{5.3.8}$$

for $i = 1, \ldots, m+1$ and $j = 1, \ldots, n+1$.

Equation 5.3.8 can be simplified as:

$$\begin{aligned}
&\frac{\partial u}{\partial \tau} + Au_{i,j} + Bu_{i-1,j} + Cu_{i+1,j} + Du_{i,j+1} + Eu_{i,j-1} \\
&\quad + Fu_{i+1,j+1} + Gu_{i-1,j-1} \geq 0,
\end{aligned} \tag{5.3.9}$$

where there is a different value for each letter of the alphabet corresponding to the different values, $i = 1, \ldots, m+1$ and $j = 1, \ldots, n+1$.

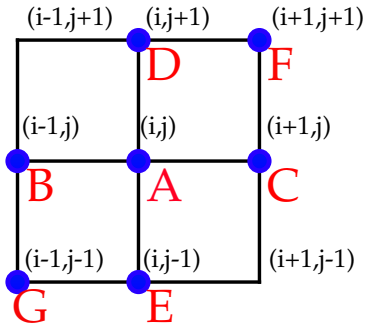

Figure 5.3.2: Seven point discretization stencil

Fig. 5.3.2 is a graphical representation of 5.3.9.

One uses the *initial condition* defined in 5.2.11 and *boundary conditions* at $x = 0$ and $y = 0$, defined in 5.2.12 and 5.2.14 respectively. At the boundary $x = x_{max}$, where $i = m+1$, one uses 5.2.13 on page 86. The following is a central difference approximation to 86 (Ikonen & Tiovanen 2004, 8):

$$\frac{\partial u}{\partial x}(x_{max}, y, \tau) = \frac{u_{m+2,j} - u_{m,j}}{2\Delta x} = 0, \tag{5.3.10}$$

for all values, $j = 0, \ldots, n+1$. Simplifying 5.3.10 one finds that a ghost point outside the grid, $(m+2, j)$, now equals an unknown internal grid point, (m, j). Therefore, the unknown grid point on the boundary, $(m+1, j)$, can be calculated as part of the pricing problem using the additional information:

$$u_{m+2,j} = u_{m,j}. \tag{5.3.11}$$

This same arguement is used on the boundary $y = y_{max}$, where $j = n+1$. Using 5.2.15 on page 86, one finde the following central difference approximation:

$$\frac{\partial u}{\partial y}(x, y_{max}, \tau) = \frac{u_{i,n+2} - u_{i,n}}{2\Delta y} = 0, \tag{5.3.12}$$

for all values $i = 0, \ldots, m+1$. Simplifying 5.3.12 one finds that a ghost point outside the grid, $(i, n+2)$, now equals an unknown internal grid point, (i, n). Therefore, the unknown grid point on the boudanry, $(i, n+1)$, can also be calculated as part of the pricing problem using the additional information:

$$u_{i,n+2} = u_{i,n}. \tag{5.3.13}$$

Applying the space discretization formula in 5.3.9 for $i = 1, \ldots, m+1$ and $j = 1, \ldots, n+1$, results in a semi-discrete equation with matrix representation (Ikonen & Tiovanen 2009, 307):

$$\frac{\partial u}{\partial \tau} + \mathbf{A}\mathbf{u} \geq 0. \tag{5.3.14}$$

where $\mathbf{A}$ is a $(m+1)(n+1) \times (m+1)(n+1)$ tridiagonal block-matrix and $\mathbf{u}$ is a vector of length $(m+1)(n+1)$ containing indices for all the unknown nodes(Ikonen & Tiovanen 2004, 8). The following section discusses the composition of matrix $\mathbf{A}$. No published information seems to exists that addresses the structure of $\mathbf{A}$ in such detail. Matrix $\mathbf{A}$ has the form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A_{1,1}} & \mathbf{A_{1,2}} & 0 & 0 & 0 & \dots & 0 & 0 \\ \mathbf{A_{2,1}} & \mathbf{A_{2,2}} & \mathbf{A_{2,3}} & 0 & 0 & \dots & 0 & 0 \\ 0 & \mathbf{A_{3,2}} & \mathbf{A_{3,3}} & \mathbf{A_{3,4}} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & \mathbf{A_{m,m-1}} & \mathbf{A_{m,m}} & \mathbf{A_{m,m+1}} \\ 0 & 0 & 0 & \dots & 0 & 0 & \mathbf{A_{m+1,m}} & \mathbf{A_{m+1,m+1}} \end{pmatrix},$$

where **each matrix in A is of size** $(n+1) \times (n+1)$. Matrices on the diagonal of **A** can be defined as:

$$\mathbf{A_{r,r}} = \begin{pmatrix} A & E & 0 & \dots & 0 & 0 & 0 \\ D & A & E & 0 & \dots & 0 & 0 \\ 0 & D & A & E & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & D & A & E \\ 0 & 0 & 0 & \dots & 0 & D+E & A \end{pmatrix},$$

for $r = 1, \dots, m+1$. Keep in mind that these constants are different for each value of i and j.

The matrices on the top diagonal of **A** are of the form:

$$\mathbf{A_{r,r+1}} = \begin{pmatrix} C & -F & 0 & \dots & 0 & 0 & 0 \\ 0 & C & -F & 0 & \dots & 0 & 0 \\ 0 & 0 & C & -F & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & C & -F \\ 0 & 0 & 0 & \dots & 0 & -F & C \end{pmatrix},$$

where $r = 1, \dots, m$. Keep in mind that these constants are different for each value of i and j.

The matrices on the bottom diagonal of **A**, with the exception of matrix $\mathbf{A_{m+1,m}}$ are of the form:

$$\mathbf{A_{r,r-1}} = \begin{pmatrix} B & 0 & 0 & \dots & 0 & 0 & 0 \\ -G & B & 0 & 0 & \dots & 0 & 0 \\ 0 & -G & B & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -G & B & 0 \\ 0 & 0 & 0 & \dots & 0 & -G & B \end{pmatrix},$$

where $r = 2, \dots, m$. Keep in mind that these constants are different for each value of i and j.

Matrix $\mathbf{A_{m+1,m}}$ is of the form:

$$\mathbf{A_{m,m-1}} = \begin{pmatrix} B+C & -F & 0 & \dots & 0 & 0 & 0 \\ -G & B+C & -F & 0 & \dots & 0 & 0 \\ 0 & -G & B+C & -F & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -G & B+C & -F \\ 0 & 0 & 0 & \dots & 0 & -F-G & B+C \end{pmatrix}.$$

The coefficients used in these matrices are defined in 5.3.8 and 5.3.9 on page 92 and as repeatedly mentioned, it is important to note that these coefficients change in value corresponding to the changes in i and j values on the x- and y axes respectively.

The structure of **A** is quite complicated to comprehend, yet a thorough understanding of it's structure is vital when programming numerical procedures to price American put options. The following is a practical illustration of how one is to go about implementing equation 5.3.9 to ultimately obtain 5.3.14. Keeping Fig. 5.3.1 on page 88 in mind, option values on boundaries where $i = 0$ and $j = 0$ are known and therefore one only needs to compute option values for unknown grid points.

Example on how to obtain equation 5.3.14

- **Compiling matrix A**
  - Start by choosing parameters m and n that represent the number of internal nodes on the x- and y-axes respectively. For example choose $m = 2$ and $n = 3$.
  - Create a column vector for each individual letter of the alphabet. The length of each vector should be $(m+1)(n+1)$, the total amount of unknown nodes on the grid. For this example the vector length is 12.

– Declare an initial matrix **A** containing zero values. The size of matrix **A** for this example is 12×12.

– Declare a counter that will keep track of the entry position of each individual value of an alphabet letter into its column vector, $count = 1$.

– Keeping equation 5.3.7 on page 91 and equations 5.3.8, 5.3.9 on page 92 in mind, one can now proceed by calculating each coefficient $A, B, \ldots, F, G$ for the different i and j values.

∗ **For tel = 1:2+1** (this will keep track of the i values as one moves along the x-axis)

∗ **For teller = 1:3+1** (this will keep track of the j values as one moves along the y-axis)

$$A(count) = \left[\frac{y_j x_i^2}{\Delta x^2} - \frac{\rho\gamma y_j x_i}{\Delta y \Delta x} - \frac{2a_{add}}{\Delta x^2} + \frac{y_j \gamma^2}{\Delta y^2} - \frac{2c_{add}}{\Delta y^2} + r\right], \tag{5.3.15}$$

$$B(count) = \left[-\frac{y_j x_i^2}{2\Delta x^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{a_{add}}{\Delta x^2} + \frac{r x_i}{2\Delta x}\right], \tag{5.3.16}$$

$$C(count) = \left[-\frac{y_j x_i^2}{2\Delta x^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{a_{add}}{\Delta x^2} - \frac{r x_i}{2\Delta x}\right], \tag{5.3.17}$$

$$D(count) = \left[-\frac{y_j \gamma^2}{2\Delta y^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{c_{add}}{\Delta y^2} + \frac{[\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}]}{2\Delta y}\right], \tag{5.3.18}$$

$$E(count) = \left[-\frac{y_j \gamma^2}{2\Delta y^2} + \frac{\rho\gamma y_j x_i}{2\Delta y \Delta x} + \frac{c_{add}}{\Delta y^2} - \frac{[\alpha(\beta - y_j) - \vartheta\gamma\sqrt{y_j}]}{2\Delta y}\right], \tag{5.3.19}$$

$$F(count) = \left[-\frac{\rho\gamma y_j x_i}{2\Delta y \Delta x}\right], \tag{5.3.20}$$

$$G(count) = \left[-\frac{\rho\gamma y_j x_i}{2\Delta y \Delta x}\right]. \tag{5.3.21}$$

count = count + 1;

End(For(teller))
End(For(tel))

One therefore obtains 12 values for each coefficient $A, B, \ldots, F, G$. This includes coefficients at the unknown points on the boundaries, $x = x_{max}$ and $y = y_{max}$. These 12 different values can be placed in the correct position in matrix **A**.

– Place coefficients into correct places in original matrix **A**.
The final form of of matrix **A** for this example is:

$$\mathbf{A}=\begin{pmatrix}
A_1 & E_1 & 0 & 0 & C_1 & F_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
D_2 & A_2 & E_2 & 0 & 0 & C_2 & F_2 & 0 & 0 & 0 & 0 & 0 \\
0 & D_3 & A_3 & E_3 & 0 & 0 & C_3 & F_3 & 0 & 0 & 0 & 0 \\
0 & 0 & (D_4+E_4) & A_4 & 0 & 0 & F_4 & C_4 & 0 & 0 & 0 & 0 \\
B_5 & 0 & 0 & 0 & A_5 & E_5 & 0 & 0 & C_5 & F_5 & 0 & 0 \\
G_6 & B_6 & 0 & 0 & D_6 & A_6 & E_6 & 0 & 0 & C_6 & F_6 & 0 \\
0 & G_7 & B_7 & 0 & 0 & D_7 & A_7 & E_7 & 0 & 0 & C_7 & F_7 \\
0 & 0 & G_8 & B_8 & 0 & 0 & (D_8+E_8) & A_8 & 0 & 0 & F_8 & C_8 \\
0 & 0 & 0 & 0 & (B_9+C_9) & F_9 & 0 & 0 & A_9 & E_9 & 0 & 0 \\
0 & 0 & 0 & 0 & G_{10} & (B_{10}+C_{10}) & F_{10} & 0 & D_{10} & A_{10} & E_{10} & 0 \\
0 & 0 & 0 & 0 & 0 & G_{11} & (B_{11}+C_{11}) & F_{11} & 0 & D_{11} & A_{11} & E_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & (G_{12}+F_{12}) & (B_{12}+C_{12}) & 0 & 0 & (D_{12}+E_{12}) & A_{12}
\end{pmatrix}.$$

- Declare column vector **u** of length $(m+1)(n+1)$ containing zeros. This vector will contain values for all the unknown grid points and will have the following $u_{i,j}$ entries:

$$\mathbf{u}=\begin{pmatrix}
u_{1,1} \\ u_{1,2} \\ u_{1,3} \\ u_{1,4} \\ u_{2,1} \\ u_{2,2} \\ u_{2,3} \\ u_{2,4} \\ u_{3,1} \\ u_{3,2} \\ u_{3,3} \\ u_{3,4}
\end{pmatrix}.$$

Matrix **A** and vector **u** have now been defined. Therefore equation 5.3.14 can now be defined. The next step is to discretize the time axis.

5.3.2 Time discretization

The stability properties of the time discretization process are vital in option pricing problems because the initial boundary condition in 5.2.11 on page 85 has a discontin-

uous first derivative (Ikonen & Tiovanen 2009, 307). Therefore, an unconditionally stable scheme with no restrictions on the interval length $\Delta\tau$ is chosen. The second-order accurate Crank-Nicolson method is given by (Ikonen & Tiovanen 2009, 307):

$$\left(\mathbf{I}+\frac{1}{2}\Delta\tau\mathbf{A}\right)\mathbf{u}^{(k+1)} \geq \left(\mathbf{I}-\frac{1}{2}\Delta\tau\mathbf{A}\right)\mathbf{u}^{(k)}. \tag{5.3.22}$$

One can simplify 5.3.22 as:

$$\mathbf{B}\mathbf{u}^{k+1} \geq \mathbf{C}\mathbf{u}^{k}, \tag{5.3.23}$$

where

$$\mathbf{B} = \left(\mathbf{I}+\frac{1}{2}\Delta\tau\mathbf{A}\right) \tag{5.3.24}$$

and

$$\mathbf{C} = \left(\mathbf{I}-\frac{1}{2}\Delta\tau\mathbf{A}\right). \tag{5.3.25}$$

5.4 Linear complimentary problem under stochastic volatility

One can now formally define the linear complimentary problem for the American put option under stochastic volatility as (Ikonen & Tiovanen 2009, 308)(Ikonen & Tiovanen 2008, 113):

$$\begin{aligned}
(\mathbf{B}\mathbf{u}^{(k+1)} - \mathbf{C}\mathbf{u}^{(k)})^{\mathrm{Tr}}(\mathbf{u}^{(k+1)} - \mathbf{g}) = 0,\\
\mathbf{B}\mathbf{u}^{(k+1)} - \mathbf{C}\mathbf{u}^{(k)} \geq 0,\\
\mathbf{u}^{(k+1)} - \mathbf{g} \geq 0, \qquad k = 0,\ldots,l+1.
\end{aligned} \tag{5.4.1}$$

To solve this linear complimentary problem, one has to solve the following matrix equation taken from 5.4.1:

$$\mathbf{B}\mathbf{u}^{(k+1)} = \mathbf{C}\mathbf{u}^{(k)},$$

subject to the constraint:

$$\mathbf{u}^{(k+1)} \geq \mathbf{g}, \qquad k = 0, \ldots, l+1.$$

5.5 Algorithm to solve put price under stochastic volatility using PSOR

Keep in mind that when programming, the numbering of nodes will start at 1, as opposed to the numbering system used in this dissertation, $i = 0, \ldots, m+1$, $j = 0, \ldots, n+1$ and $k = 0, \ldots, l+1$. This algorithm is written for $k = 0$ but needs to be repeated for all values of k.

1. **INITIAL PARAMETERS**

   (a) Stipulate values K, T (per annum), r (per annum), x_{max}, y_{max}, γ, ρ, ϑ, β, α, ω and ε.

   (b) Given current value of underlying stock price, $S_{current}$ and volatility level where price should be specified, $\sigma_{current}$.

   (c) Choose number of intervals on each axis of the grid:

   m, number of internal nodes on x-axis.

   n, number of internal nodes on y-axis.

   l, number of internal nodes on τ-axis.

   (d) Create initial matrix **A**, containing zeros.

   (e) Create initial matrix **Z**, containing zeros. In this matrix all prices will be stored. It is therefore a combination of the values for all $x-y$ grids at a specific time increment for all the values of k, $k = 0, \ldots, l+1$ and contains prices of the three dimensional grid.

   (f) Calculate interval lengths, $\Delta x, \Delta y, \Delta \tau$.

2. **CALCULATING AMERICAN PUT OPTION PRICES**

   Steps $(a), \ldots, (l)$ *need to be repeated for* $k = 0, \ldots, l+1$.

(a) Compute boundary conditions using 5.2.12 and 5.2.14 and store them in the correct positions in matrix **Z**.

(b) Compute different a_{add} and c_{add} values for $i = 0,\ldots,m+1$ and $j = 0,\ldots,n+1$. There is now a different a_{add} and c_{add} value for each node of the grid on page 88. These values are used to compute the elements of the coefficient matrix **A**. The most effective way to store these different values, is to create a vector for a_{add} and c_{add} respectively. The length of this vector will be $(m+1)\times(n+1)$.

(c) Compute coefficient matrix entries: A,B,C,D,E,F for each $i = 0,\ldots,m+1$ and $j = 0,\ldots,n+1$ using equations 5.3.15 to 5.3.21. The most effective way to store these coefficients is to create a different vector for each letter of the alphabet. The length of this vector will be $(m+1)\times(n+1)$.

(d) Using the different alphabet vectors created in the previous step, one now has to place the correct coefficient in the correct position in matrix **A**. This is quite a complicated task, but using the general structure of the tridiagonal block matrix given on page 92 as guideline along with the subsequent breakdown of each individual matrix in the pages to follow, after some effort one obtains the coefficient matrix **A**.

(e) Create identity matrix **I**, with same dimensions as **A**.

(f) Obtain matrix **B** by applying equation 5.3.24 on page 98.

(g) Obtain matrix **C** by applying equation 5.3.25 on page 98.

(h) Applying the initial condition in equation 5.2.11 on page 85, calculate the values of the grid on the first time level, $k = 0$ and store these in the correct position in the matrix containing solutions at all time increments, **Z**.

(i) Create a vector **g**, with length corresponding to all the unknown grid points at one time increment, thus for $i = 1,\ldots,m+1$ and $j = 1,\ldots,n+1$. The entries of **g** correspond to the grid points when $k = 0$ and is calculated using equation 5.2.11 on page 85.

(j) Create a vector $\mathbf{u^0}$, with length corresponding to all the unknown grid points at one time increment, thus for $i = 1,\ldots,m+1$ and $j = 1,\ldots,n+1$. The entries of $\mathbf{u^0}$ correspond to the grid points when $k = 0$ and is calculated using equation 5.2.11 on page 85.

(k) Calculate vector **b** as $\mathbf{b} = \mathbf{Cu^0}$. One now has the right hand side of equation 5.3.23 on page 98 and one needs to solve $\mathbf{u^1}$ using the following equation:

$$\mathbf{Bu}^1 = \mathbf{b}, \tag{5.5.1}$$

subject to the constraint:

$$\mathbf{u}^1 \geq \mathbf{g}. \tag{5.5.2}$$

The elements of $\mathbf{u^1}$ are solved iteratively using the PSOR method. To speed up computation time, the PSOR method can be adapted to handle only the non-zero elements of matrix **B** and the value of ω can be optimized.

(l) Once $\mathbf{u^1}$ has been solved, its values are stored into the right positions in matrix **Z**. Keep in mind that this matrix will eventually contain all the option values at all time increments and as mentioned earlier, this matrix represents all the values of the three dimensional grid with axes time, stock price and volatility.

3. **FIND CURRENT OPTION VALUE**

After solving all the option prices, the last entries of **Z** corresponding to the dimensions of the $x-y$-grid will contain the current option prices for a range of stock prices and volatilities. The current stock price $S_{current}$ and specific volatility level $\sigma_{current}$ are stipulated beforehand. The final step is to interpolate to find the option price that corresponds the current stock price $S_{current}$ and specific volatility level $\sigma_{current}$ respectively.

Chapter 6

Numerical Experiments

The following models were solved using a computer with specifications:

- Intel Core 2 Duo @ 2.8GHz,
- 1 GB Ram,
- 160 Gig.

6.1 Constant volatility

The following example is taken from the book *The mathematics of financial derivatives* (Wilmott et al. 1996, 174), where the parameters are chosen as:

- $K = 10$ (strike price),
- $r = 0,1$ (constant interest rate),
- $T = 0,25$ (3 months duration of the option contract),
- $\sigma = 0,4$ (constant volatility) ,
- $\omega = 1,8$ (PSOR relaxation parameter),
- $m = 300$ (amount of intervals on the τ-axis),
- $n = 300$ (amount of intervals on the x-axis).

The following solutions were obtained:

Stock price	European put	American put	(Wilmott et al. 1996, 176)
4	5,753100148	6	6
6	3,756838358	4,000000051	4
8	1,901630105	2,019254756	2,02
10	0,667864364	0,6892706	0,6913
12	0,166486116	0,169866883	0,1711

Table 6.1.1: American and European put options under constant volatility

Table 6.1.1 compares the values of European and American put options for different underlying stock prices. Notice that the price of European put options are less than their American counterparts. As mentioned earlier, this is due to the early exercise facility offered by American options that allows a greater flexibility to the option holder. Option values decrease significantly as the option shifts from in the money to out of the money. The results are comparable to the American put option values in (Wilmott et al. 1996, 176), since the algorithm used in this dissertation is based on descriptions in (Wilmott et al. 1996, 167 - 177).

The following figures were also generated:

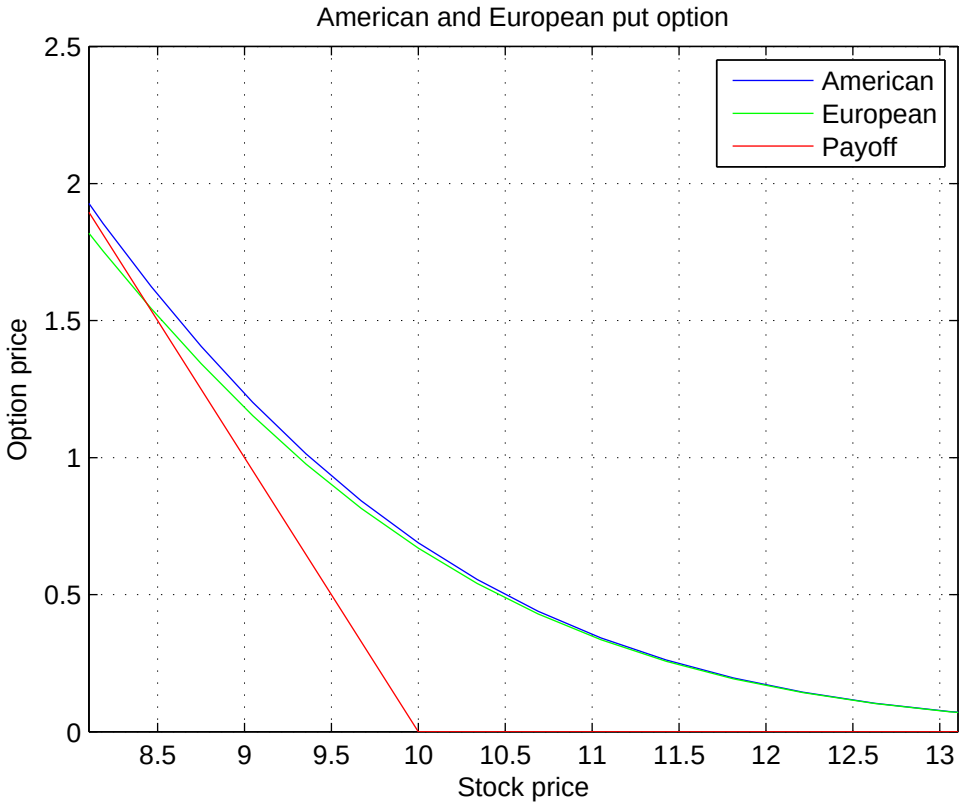

Figure 6.1.1: European and American put option prices - under constant volatility

Fig. 6.1.1 is calibrated to show option prices for values of the underlying between $S = 8$ and $S = 13$. It illustrates the price differences between American and European put options. These price differences are more significant for in the money options and the general pattern follows the traditional form of the option payoff, with prices decreasing gradually as the underlying price is increased. This is directly related to the payoff function:

$$\text{Payoff} = max[K - S_t, 0]. \tag{6.1.1}$$

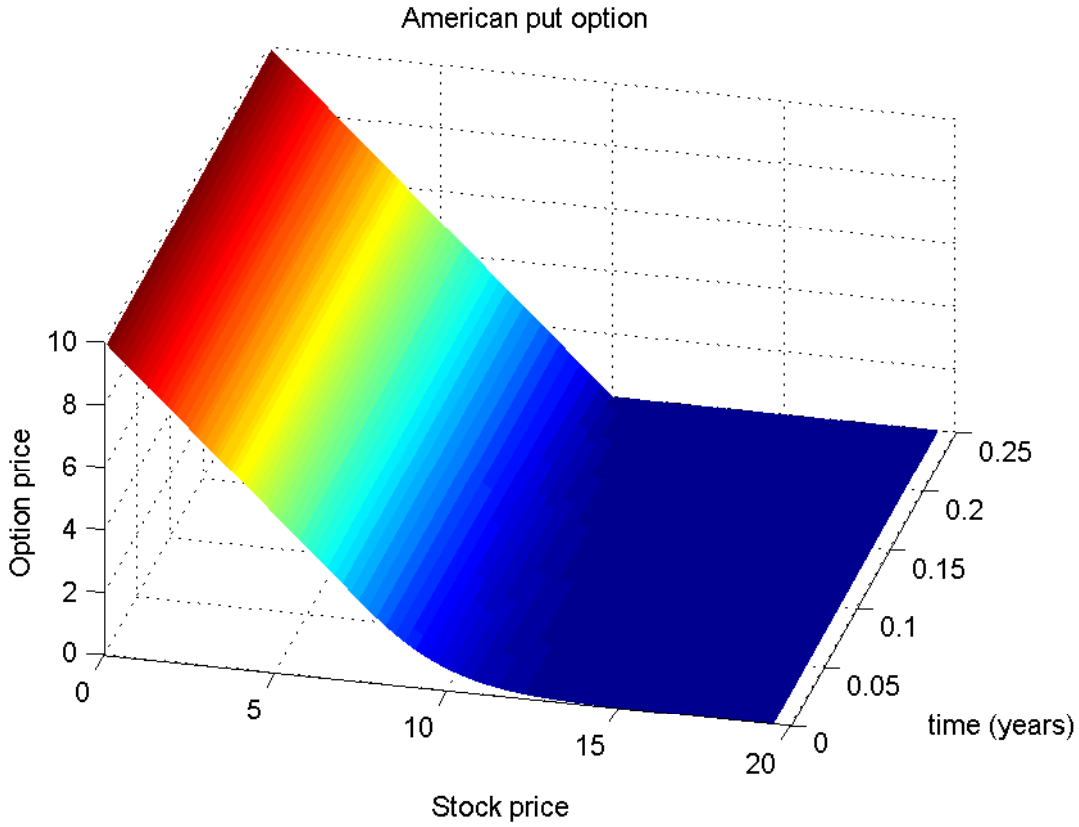

Figure 6.1.2: American put option price surface

Fig. 6.1.2 is a graphical representation of the option price surface. This is a three dimensional graph with stock price, time and American put option prices as axes. It too illustrates that the value of the option increases as the underlying stock price decreases. Additionally it shows that of options with a longer time to maturity are more valuable than options where the exercise date is looming. The reason for this is that for both in the money and out of the money options, a longer duration implies that there is a greater probability for making more money.

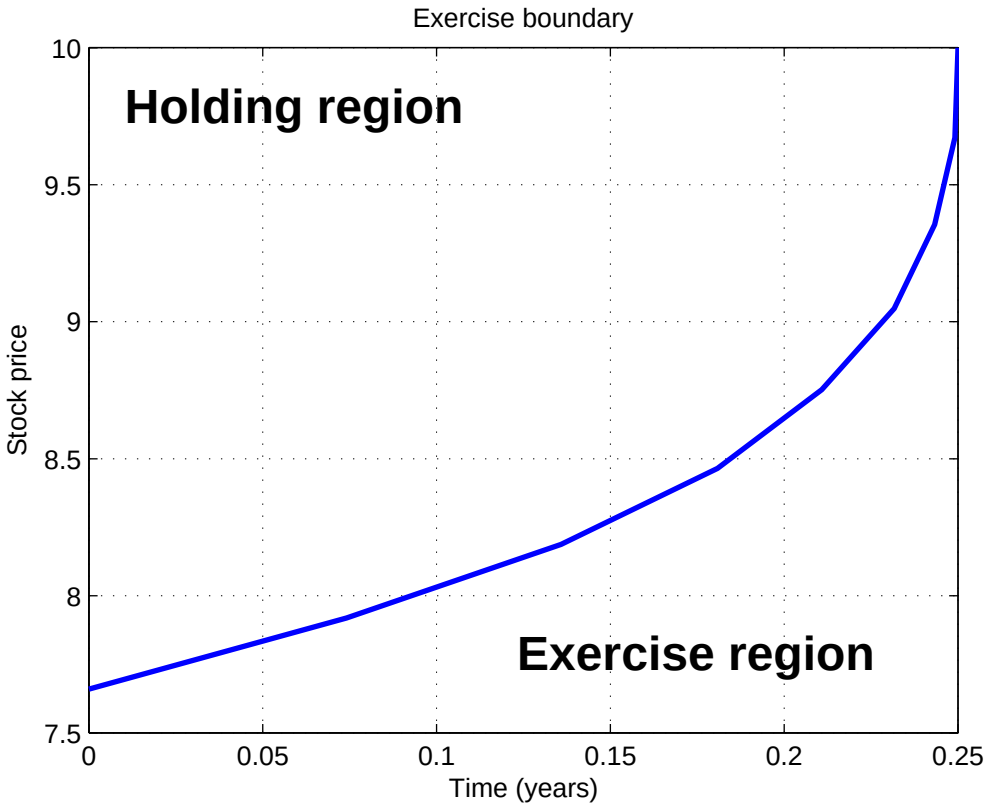

Figure 6.1.3: American put option early exercise boundary

Fig. 6.1.3 shows the early exercise boundary. It shows what the option holder's mindset should be at a specific point in time for a specific stock price value. For stock prices above the boundary, the option should be held and for stock prices under the boundary, the option should be exercised. Recent studies have covered topics surrounding the exercise boundary, such as the asymptotic behaviour of the boundary near expiry. This asymptotic behaviour can be observed from Fig. 6.1.3. As mentioned in section 3.4.3, the interested reader can refer to the following resources: *A Comparative Study of American Option Evaluation and Computation* by K. Rodolfo (Rodolfo 2007, 45-51), *The Mathematics of Financial Derivatives* (Wilmott et al. 1996, 121-129) and *Mathematical models of Financial Derivatives* (Kwok 2008, 257-262).

A parameter sensitivity analysis was performed to investigate the effect different volatility values have on American put option prices under constant volatility. Table 6.1.2 contains data gathered for both in the money put options, with current stock price $S = 9$ and out of the money put options, with $S = 11$. The other parameters were chosen as stipulated on page 99. As volatility increases, so does the price of the option and so the data supports the underlying option theory.

σ	Put (in the money)	Put (out of the money)
0,0625	1,000106586	0
0,125	0,999797503	0,006744304
0,1875	0,999330151	0,04629278
0,25	1,027205002	0,118003376
0,3125	1,102204667	0,209091841
0,375	1,194133908	0,311480472
0,4375	1,294194631	0,420767228
0,5	1,398832562	0,534490685
0,5625	1,506183937	0,651146405
0,625	1,615220401	0,769784632
0,6875	1,725338893	0,889783531
0,75	1,836144582	1,010714493
0,8125	1,947392995	1.13229042
0,875	2,058824557	1,254218422
0,9375	2,17031157	1,376335943

Table 6.1.2: Volatility (σ) parameter sensitivity analysis

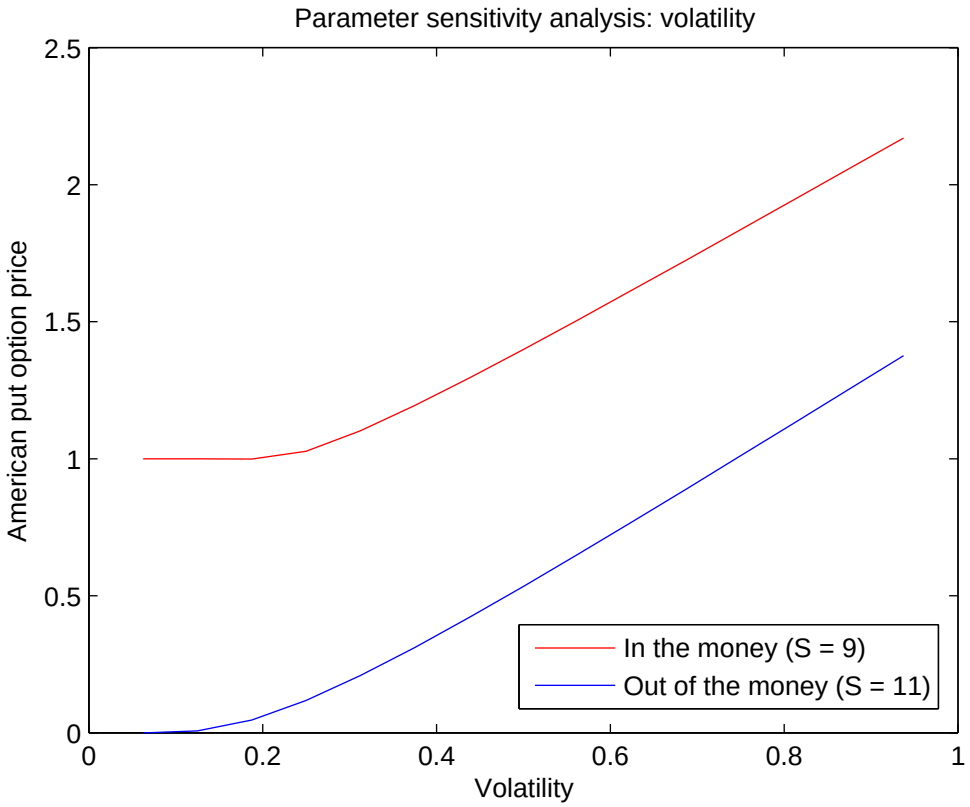

Figure 6.1.4: Volatility (σ) sensitivity analysis

Fig. 6.1.4 is a graphical representation of table 6.1.2. Puts under constant volatility present with constant prices for lower levels of volatility and exhibit a linear increase in price as volatility passes a certain percentage mark. For in the money options, this increase occurs at roughly the 20% mark, whereas for out of the money options, the linear price increase occurs roughly the 10% mark. The out of the money options are priced at significantly lower levels than their in the money counterparts.

6.2 Stochastic volatility

6.2.1 Experiment one

The following example is taken form the paper, *Operator splitting methods for pricing American options under stochastic volatility* (Ikonen & Tiovanen 2004, 11), where the parameters are chosen as:

- $K = 10$ (strike price),
- $r = 0,1$ (constant interest rate),
- $T = 0,25$ (3 months duration of the option contract),
- $\omega = 1,8$ (PSOR relaxation parameter),
- $\alpha = 5$ (rate of mean reversion),
- $\beta = 0,16$ (long term variance),
- $\gamma = 0,9$ (volatility of volatility),
- $\rho = 0,1$ (correlation between two Wiener processes),
- $x_{max} = 20$ (maximum stock price on grid),
- $\vartheta = 0$ (market price of the volatility risk),
- $y_{max} = 1$ (maximum volatility on the grid),
- $m = 80$ (amount of internal nodes on the x-axis, the stock price axis)
  x_i denotes a specific stock price at a node and $i = 0, \ldots, m+1$,
- $n = 32$ (amount of internal nodes on the y-axis, the volatility axis)
  y_j denotes a specific volatility level at a node and $j = 0, \ldots, n+1$,
- $l = 16$ (amount of internal nodes on the τ-axis, the time axis)
  τ_k denotes a specific point in time at a node and $k = 0, \ldots, l+1$.

Additionally, for the PSOR iterative procedure, one chooses x_{old} and x_{new}, two vectors that contain initial guesses for the unknown option values at one time interval. These vectors are chosen using the payoff formula:

$$\text{Payoff} = max[0, K - x_i], \tag{6.2.1}$$

where $i = 1, \ldots, m+1$.

The stop criteria for the PSOR iterative process was chosen as $\varepsilon = 0,00001$.

Table 6.2.1 contains the solutions for American put option prices calculated using the stochastic volatility model. Prices were calculated for two different variances, $y = 0,0625$ and $y = 0,25$ respectively. Values given in the reference article are also included (Ikonen & Tiovanen 2004, 12).

Stock price	Put ($y = 0,0625$)	Article	Put ($y = 0,25$)	Article
8	2,0000	2	2,1968	2,07744
8,25	1,7504		2,0076	
8,5	1,5011		1,8282	
8,75	1,2544		1,6590	
9	1,0149	1,10435	1,5005	1,33192
9,25	0,7852		1,3530	
9,5	0,5691		1,2167	
9,75	0,3774		1,0917	
10	0,2341	0,50755	0,9778	0,79388
10,25	0,1468		0,8747	
10,5	0,0964		0,7819	
10,75	0,0668		0,6987	
11	0,0485	0,20462	0,6244	0,44650
11,25	0,0366		0,5583	
11,5	0,0284		0,4994	
11,75	0,0226		0,4471	
12	0,0182	0,07909	0,4006	0,242170

Table 6.2.1: American put option prices under stochastic volatility

Since the algorithm in (Ikonen & Tiovanen 2004, 1-19) is not described in detail, the algorithm used in this dissertation is the author's interpretation of the information stipulated in (Ikonen & Tiovanen 2004, 1-19).

Regarding the numerical solutions obtained, notice that in the region of the strike price, $K = 10$ and for out of the money options, the solutions obtained differ from the solutions stipulated in (Ikonen & Tiovanen 2004, 12). Three factors have been identified as possible explanations for these price differences:

1. The price differences could be attributed to the fact that the pricing model is expected to deliver inaccurate prices at points of discontinuity, in this case when $S = K$ (Duffy 2004, 71).

2. The choice of the boundary values might also be altered. It has been suggested that the choice of the boundary at $y = 0$, which was chosen as,

$$u(x,0,\tau) = g(x) = max[K-x,0] \qquad (x,\tau) \in [0,x_{max}] \times [0,T], \qquad (6.2.2)$$

   according to (Ikonen & Tiovanen 2004, 5) (Clarke & Parrott 1999, 180), can be adjusted to improve results (Chockalingam & Muthuram 2011, 796).

3. The interval lengths of the grid in all three dimensions are chosen very coarsely. Although a smaller increment does not remedy the pricing discrepancies on its own, when one factors in a finer grid along with the previous two points of discussion, a more accurate solution may possibly be obtained. This could be further investigated in the future.

No additional data was found in the literature to compare the obtained values to. The data behaviour supports the underlying option theory, as will be seen in the figures to follow.

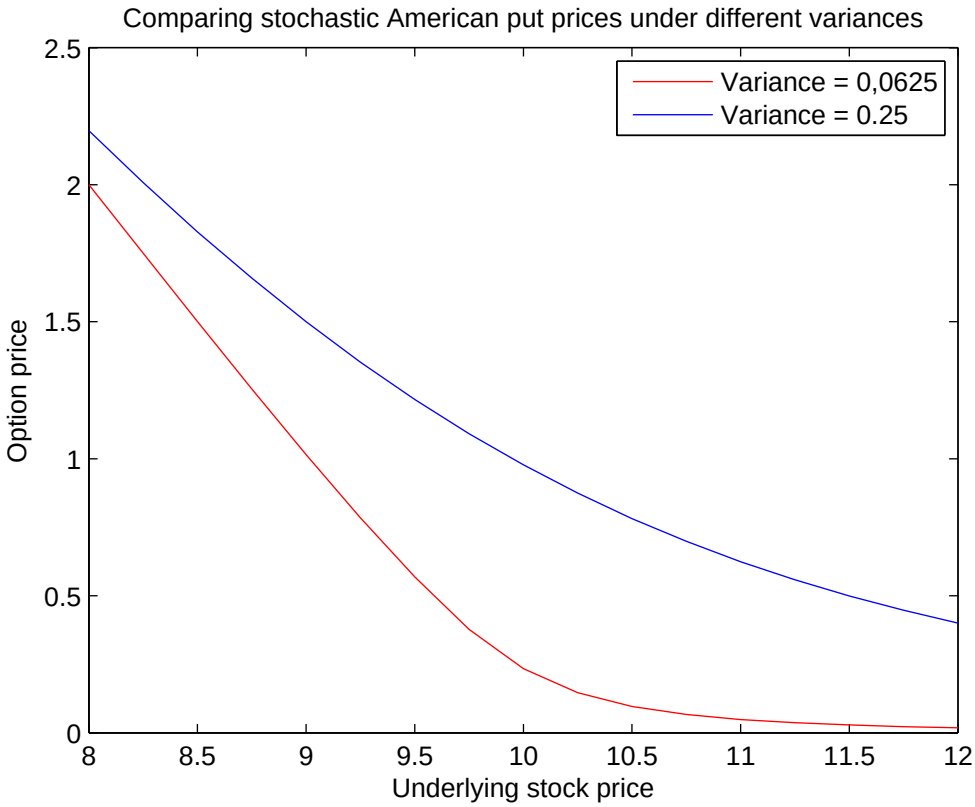

Figure 6.2.1: American put option prices under stochastic volatility

Fig. 6.2.1 compares the stochastic put prices obtained for variances $y = 0,0625$ and $y = 0,25$ respectively. As expected the prices computed using the higher variance, $y = 0,25$, are more expensive than the prices computed for the lower variance. Thus, the behaviour of the data is consistent with the expected theoretical results.

y	Put(in the money)	Put(out of the money)
0,0625	1,014872419	0,04850057
0,125	1,149453031	0,220689055
0,1875	1,328675604	0,428881734
0,25	1,500532212	0,624436156
0,3125	1,650723005	0,793946879
0,375	1,777129738	0,936076032
0,4375	1,881188553	1,052882599
0,5	1,965317701	1,147257662
0,5625	2,032135782	1,222192113
0,625	2,084177111	1,280537048
0,6875	2,123761979	1,324896066
0,75	2,152938171	1,35756825
0,8125	2,173456442	1,380525851
0,875	2,186777349	1,395418406
0,9375	2,194102904	1,403605618

Table 6.2.2: Variance parameter sensitivity analysis

A parameter sensitivity analysis was performed to investigate the effect different variance values have on American put option prices in a stochastic scenario. Table 6.2.2 contains the data gathered for both in the money put options, with current stock price $S = 9$ and out of the money put options with $S = 11$. The other parameters were chosen as stipulated on pages 107 and 108. As volatility increases, so does the price of the option. Therefore, the data supports the underlying option theory.

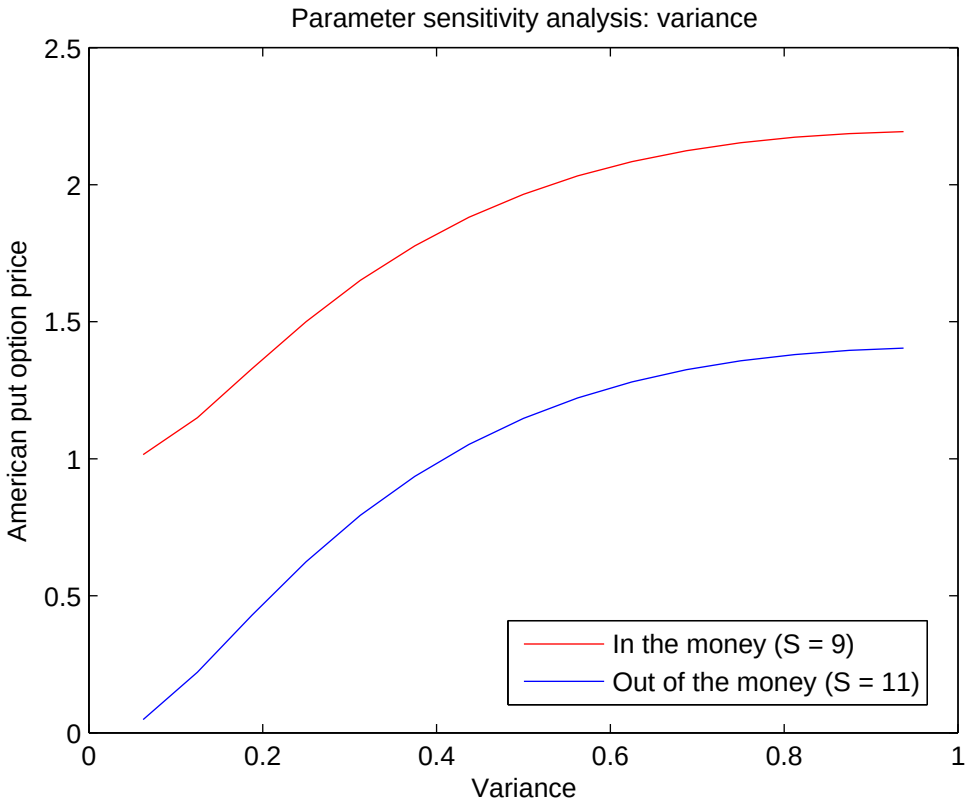

Figure 6.2.2: Variance sensitivity analysis

Fig. 6.2.2 is a graphical representation of table 6.2.2. There is a sharp increase in the price of American puts as the variance parameter, y, is increased. This increase in option price stabilizes as the volatility reaches the 80% mark. This supports the notion that then volatility is large, a marginal increase in volatility has very little effect on the option price (Chockalingam & Muthuram 2011, 796). In the money options are more valuable than out of the money options with the same parameters.

No evidence was found in the literature of such a parameter sensitivity analysis.

ω	American put (stochastic)	Computation time
1	1,500529657	3,017956
1,05	1,500529931	2,687155
1,1	1,500530114	2,594131
1,15	1,500530277	2,532479
1,2	1,500530481	2,530937
1,25	1,500530615	2,3672
1,3	1,500530767	2,236525
1,35	1,500530897	2,147487
1,4	1,50053107	2,17602
1,45	1,500531114	2,016279
1,5	1,50053127	2,04633
1,55	1,500531581	2,024228
1,6	1,500531844	1,962723
1,65	1,500532168	2,032544
1,7	1,500532238	2,154058
1,75	1,500532144	2,138417
1,8	1,500532212	2,114659

Table 6.2.3: ω sensitivity analysis: PSOR computational time

Table 6.2.3 contains data obtained by performing an ω parameter sensitivity analysis. Options where priced using the stochastic volatility model and because this model uses the Projected Over-Relaxation iterative method (PSOR), different values for ω where chosen and their computational times were compared. Parameters defined on pages 104 and 105 were used with the current stock price chosen as $S = 9$ and variance chosen as $y = 0,25$.The chosen ω values range from $\omega = 1$, which represents the Gauss-Seidel method, to $\omega = 1,9$. The solutions converge to similar values for each ω but computational times differ considerably.

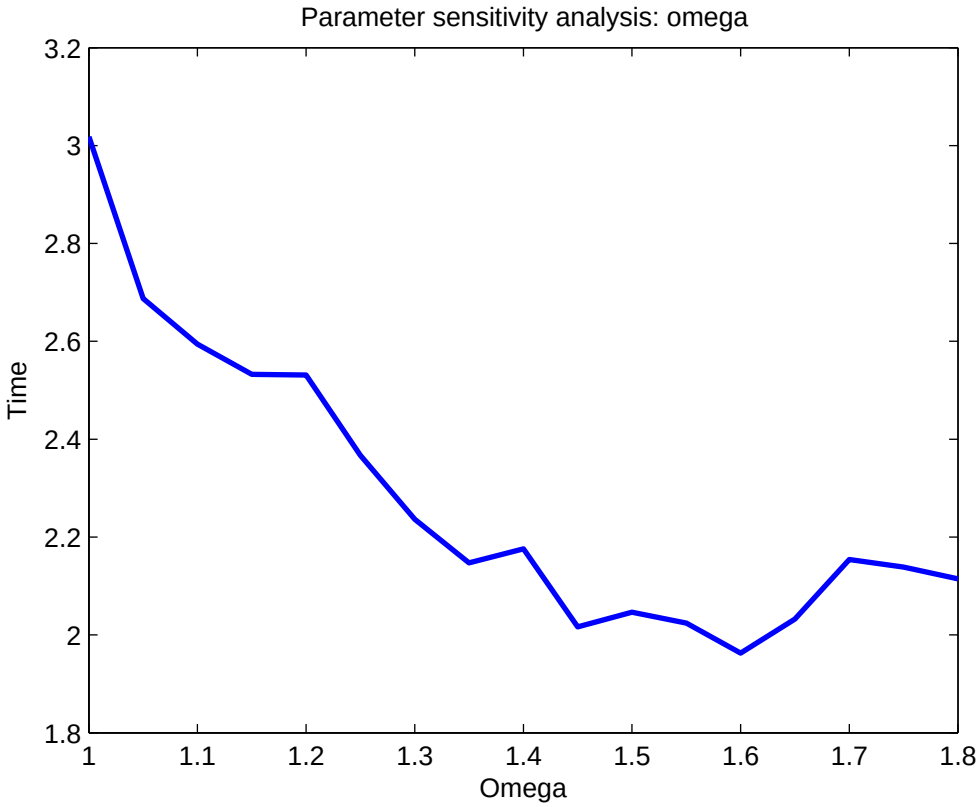

Figure 6.2.3: ω sensitivity analysis: PSOR computational time

Fig. 6.2.3 is a graphical representation of the data contained in table 6.2.3. It reveals that for $\omega = 1,6$, the computational time is at a minimum ($1,962723$ seconds). As ω is increased from this point (in this dissertation $\omega = 1,8$) the computational time increases. The maximum computational time occurs when the Gauss-Seidel method is implemented, $\omega = 1$ ($3,017956$ seconds).

6.2.2 Experiment two

The aim of this experiment is to validate the numerical PSOR algorithm by comparing its solutions to values obtained solving the matrices directly. The following parameters were chosen:

- $K = 10$ (strike price),
- $r = 0,1$ (constant interest rate),

- $T = 0,25$ (3 months duration of the option contract),
- $\omega = 1,8$ (PSOR relaxation parameter),
- $\alpha = 5$ (rate of mean reversion),
- $\beta = 0,16$ (long term variance),
- $\gamma = 0,9$ (volatility of volatility),
- $\rho = 0,1$ (correlation between two Wiener processes),
- $x_{max} = 15$ (maximum stock price on grid),
- $\vartheta = 0$ (market price of the volatility risk),
- $y_{max} = 1$ (maximum volatility on the grid),
- $m = 2$ (amount of internal nodes on the x-axis, the stock price axis)
  x_i denotes a specific stock price at a node and $i = 0, \ldots, m+1$,
- $n = 3$ (amount of internal nodes on the y-axis, the volatility axis)
  y_j denotes a specific volatility level at a node and $j = 0, \ldots, n+1$,
- $l = 1$ (amount of internal nodes on the τ-axis, the time axis)
  τ_k denotes a specific point in time at a node and $k = 0, \ldots, l+1$,
- $\varepsilon = 0,00001$.

The solutions obtained using both the numerical and the direct methods are summarised in a table containing the option values at individual grid points. This table has the form:

$u_{0,4}$	$u_{1,4}$	$u_{2,4}$	$u_{3,4}$
$u_{0,3}$	$u_{1,3}$	$u_{2,3}$	$u_{3,3}$
$u_{0,2}$	$u_{1,2}$	$u_{2,2}$	$u_{3,2}$
$u_{0,1}$	$u_{1,1}$	$u_{2,1}$	$u_{3,1}$
$u_{0,0}$	$u_{1,0}$	$u_{2,0}$	$u_{3,0}$

Table 6.2.4: Summary of values at grid points

The entries of table 6.2.4 resemble the entries on Fig. 5.3.1. Using direct analytical method to solve the pricing problem one obtains the following prices:

10	5	1,3509	0,7810
10	5	1,3281	0,7571
10	5	1,0927	0,5449
10	5	0,4640	0,1721
10	5	0	0

Table 6.2.5: Prices obtained using analytical method

The following prices where obtained using the numerical PSOR method:

10	5	1,5261	0,8561
10	5	1,4882	0,8271
10	5	1,2056	0,5901
10	5	0,5136	0,1858
10	5	0	0

Table 6.2.6: Prices obtained using numerical PSOR method

This example verifies that the PSOR numerical procedure is sufficiently accurate and as mentioned earlier, it is implemented to save on both computation time and effort by excluding calculations with the zero elements of the coefficient matrix. In experiment one, where solutions were obtained using the numerical PSOR method, the following analytical results where obtained. These results are contained within the brackets and prove that the numerical solutions are indeed accurate.

Stock price	Put ($y = 0,0625$)	Article	Put ($y = 0,25$)	Article
8	2,0000 (2)	2	2,1968 (2,19)	2,07744
9	1,0149 (1,002)	1,10435	1,5005 (1,49)	1,33192
10	0,2341 (0,217)	0,50755	0,9778 (0,972)	0,79388
11	0,0485 (0,047)	0,20462	0,6244 (0,62)	0,44650
12	0,0182 (0,018)	0,07909	0,4006 (0,398)	0,242170

Table 6.2.7: American put prices obtained

Chapter 7

Conclusion

The aim of this dissertation was to develop comprehensive algorithms, incorporating both the Crank-Nicolson implicit finite difference and the PSOR methods, that could be used to price American put options under both constant and stochastic volatility. Motivated by the fact that, for constant volatility models, algorithms are very compact (Seydel 2009, 175) and that no algorithms for stochastic volatility models could be found, the following detailed algorithms were developed in this dissertation:

- *Constant volatility*.
  A detailed description of the PSOR algorithm along with an algorithm to find the current price of an American put can be found on pages 72 - 79.
- *Stochastic volatility*.
  This algorithm can be found on pages 96 - 98. The PSOR method referred to in this section, follows the same steps as the one described in the constant volatility section with the only difference being the vector dimensions. When programming this iterative process, the program can be manipulated to exclude all zero elements. Also note that the choice of the two initial vectors of the PSOR method, which are not explicitly stated in literature, is covered in this algorithm. This algorithm also goes into great detail describing the structure of the tridiagonal block-matrix **A**. No evidence of this could be found in literature.

Both models are explained thoroughly, with emphasis placed on equipping the reader to implement the models and not merely understand the theory behind it.

The Black-Scholes model's assumption of constant volatility is not its only drawback. More concerns are raised when considering the models' statistical properties. It is often criticised for not accurately reflecting market behaviour. One of the main concerns is

that the probability distribution of asset returns has characteristics not taken into consideration by this model. These include heavy tails, skewed distribution with high peaks and volatility clustering and cannot be explained by the log-normal assumption that underlies the Black-Scholes model (Salmi & Toivanen 2011, 821). Modern approaches have been developed to address all of the unsatisfactory elements just mentioned, these fall into two main categories:

- Stochastic volatility without jumps.
- Stochastic volatility with jumps.

This dissertation only focused on the Heston stochastic volatility model without jumps. However, both of the above mentioned approaches, perform well only in special cases, with stochastic volatility models without jumps, offering more realistic outcomes in cases involving long maturity terms (Ballestra & Sgarra 2010, 1571). This has naturally led to models that incorporate elements of both stochastic volatility and jumps. The three most popular of these models are (Ballestra & Sgarra 2010, 1572):

- BNS model, introduced by Barndorff-Nielsen and Shephard in 2001.
- Bates model, introduced in 1996.
- Time-changed Levy models, introduced by Carr, Geman and Madan in 2003.

Future research should therefore be based on a model that takes stochastic volatility with jumps into account.

When considering the option prices, accurate results where obtained under the constant volatility model and these are summarized in chapter 6.1. Satisfactory results were obtained under stochastic volatility and the behaviour of solutions support the underlying theoretical principles.

Future research could focus on the refinement of the stochastic volatility model, were issues such as the boundary conditions and the Crank-Nicolson method itself can be further investigated.

Chapter 8

Appendix

8.1 Constant volatility MATLAB code

```
%MAIN PROGRAM THAT CALLS ALL OTHE SUB-PROGRAMS:
%AmericanOption_kiki(K,T,r,sigma,type,m,n);
%FreeBoundary_kiki(S,t,V,K,type);

%ONLY THIS SECTION NEEDS TO BE EXICUTED

% *Parameters*
%
%    K       : Strike price.
%    T       : Expiery time (expressed in years).
%    r       : Annualized, continuously compounded risk-free rate,
%              expressed as a positive decimal number.
%    sigma   : Constant volatility.
%    type    : 'put' option stipulated.
%    Scurrent : Current price of underlying stock.
%    n       : Number of intervals on stock price axis
%              (The number of discrete stock values is n+1).
%    m     : Number of intervals on time axis
%            (The number of discrete time values is m+1).

% *Numerical Results*
%
%    current_option_value : Provides price of American put option.
%    Elapsed time         : Duration of the problem solving process
%                           (this wil increase as number of nodes are
%                           increasd).
%    Fig. 1               : Option price vs. Stock price of both American
%                           and European options.
%    Fig. 2               : Option price vs. Stock price of both American
%                           and European options.
%    Fig. 3               : Graphical representation of exercise boundary
%                           Stock price vs. Time.

%=========================================================================
%=========================================================================
```

```
%% Define parameters
tic
format long

% Financial parameters

K     = 10;
T     = 0.25;
r     = 0.1;
sigma = 0.4;
type  = 'put';
Scurrent = 8;

% Numerical parameters

m = 300; %x axis
n = 300; %t axis

%=========================================================================
%=========================================================================

%% Compute American option prices

[S,t,V] = AmericanOption_kiki(K,T,r,sigma,type,m,n);

%=========================================================================
%=========================================================================

%% Compute free boundary

Sf = FreeBoundary_kiki(S,t,V,K,type);

%Compute indices for smoothing the free boundary
%find = returns all non zero elements
%diif calculates difference between adjacent entries

FreeBoundaryIndices = [1, find(abs(diff(Sf))>1e-5)+1]

%=========================================================================
%=========================================================================

%% Define plot variables

%Reduce field to stock prices only withing range 3*K from 0.
plotrange = S>=0 & S<=2*K;


%Vector
%Only values within plotrange
Sp = S(plotrange)
%Only s values for plotrange, but these values over all time indices
%Matrix
Vp = V(plotrange,:)

%=========================================================================
%=========================================================================

%% Graph of American and European option values at time t=0 and payoff

figure('Color','White')
```

```
%Compute correspoding European option values
%Using BS toolbox
%Send stock values for range chosen Sp
%K
%T
%sigma - volatility
%delta = 0
[EurCall,EurPut]=blsprice(Sp,K,r,T,sigma);

switch type
    case 'call'
        V_payoff = [0 0 K];
        V_euro   = EurCall;
        location = 'NorthWest';
    case 'put'
        V_payoff = [K 0 0];
        V_euro   = EurPut;
        location = 'NorthEast';
end

plot(Sp,Vp(:,1),'b',Sp,V_euro,'g',[0 K 2*K],V_payoff,'r')
title(['American and European ',type,' option'])
legend('American','European','Payoff','Location',location)
grid on
xlabel('Stock price')
ylabel('Option price')

%=========================================================================
%=========================================================================

%% 3-D plot of option values versus stock prices and time

figure('Color','White')

[t_grid,Sp_grid]=meshgrid(t,Sp);
surf(Sp_grid,t_grid,Vp,'LineStyle','none')
title(['American ',type,' option'])
xlabel('Stock price')
ylabel('time (years)')
zlabel('Option price')

%=========================================================================
%=========================================================================

%% Plot free boundary

figure('Color','White')

plot(t(FreeBoundaryIndices),Sf(FreeBoundaryIndices),'LineWidth',2)
title('Exercise boundary')
grid on
xlabel('Time (years)')
ylabel('Stock price')

switch type
    case 'call'
        location_hold = [0.15 0.15 0.5 0.1];
        location_ex   = [0.5  0.8  0.5 0.1 ];
    case 'put'
        location_hold = [0.15 0.8  0.5 0.1];
        location_ex   = [0.5  0.15 0.5 0.1];
end
```

```
%HEADINGS
annotation('textbox','String',{'Holding region'},...
    'FontWeight','bold',...
    'FontSize',18,...
    'FontName','Arial',...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'Position',location_hold);

annotation('textbox','String',{'Exercise region'},...
    'FontWeight','bold',...
    'FontSize',18,...
    'FontName','Arial',...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'Position',location_ex);

%=========================================================================
%=========================================================================

%% Interpolating option values to obtain the option price corresponding to the ↩
    current stock price
%At time t=0, option values in the first column of the value matrix V

%The stock price range
tt = S;
%The different option values correspoding to stock prices
pp = V(:,1);

%Compute using matlab spline interpolation function
current_option_value = interp1(tt,pp,Scurrent, 'spline')

%Stop timing
toc
```

```
function [S,t,V] = AmericanOption_kiki(K,T,r,sigma,type,m,n)

% Input:
%
%   K       : Strike price.
%   T       : Expiery time (expressed in years).
%   r       : Annualized, continuously compounded risk-free rate,
%             expressed as a positive decimal number.
%   sigma   : Constant volatility.
%   type    : 'put' option stipulated.
%    n      : Number of intervals on time axis
%             (The number of discrete time values is n+1).
%    m      : Number of intervals on stock price axis
%            (The number of discrete stock values is m+1).%
% Output:
%
%   S = range of stock prices
%   t = range of time points from 0 to T
%   V = corresponding option prices, i.e.
%       V(i,j) is an approximation of V(S(i),t(j))

%=========================================================================
%=========================================================================
```

```
%% Define parameters

%Define fransformation parameters used to obtain the heat equation
% q = 2*r/sigma^2
% S = Ke^x
% S_min = Ke^x_min
% S_max = Ke^x_max
%tau = 0.5sigma^2T

%Dimensionless parameters
delta = 0;
q       = 2*r/sigma^2;
q_delta = 2*(r-delta)/sigma^2;

%Asset space
x_min = -5;
x_max = 5;

%Calculate step length on x axis
dx      = (x_max-x_min)/m;

%Time space
tau_max = .5*sigma^2*T;

%Calculate step length on tau axis
dtau    = tau_max/n;

%=========================================================================
%=========================================================================
%HERE WE CAN MANIPULATE NUMERICAL
%Stop criteria paramenter.
eps     = 1e-6;
%Stipulate Crank-Nicolson is used when = 0.5.
theta   = 0.5;
%Relaxation parameter of SOR
omega_R = 1.8;

lambda  = dtau/dx^2;
alpha   = lambda*theta;

%=========================================================================
%=========================================================================
%Verify stability condition.
%When theta < 0.5, then explicit method is chosen and we need to
%test lambda.
if theta < 0.5

    if lambda > 0.5
    error(strcat('The algorithm is unstable. Stability can be obtained ',...
          ' by increasing the value of n or by decreasing the value of m, thus by ↩
              changing the interval lengths or by changing theta to > 0.5 and thus ↩
              selecting an implicit method'))
    end
end

%-------------------------------------------------------------------------
function boundary = g(x,tau)

    %It is also used for
    %the boundary and initial conditions.
```

```
    abb1 = exp(((q_delta-1)^2+4*q)*tau/4);
    abb2 = exp((q_delta-1)*x/2);
    abb3 = exp((q_delta+1)*x/2);
    switch type
        case 'put'
            boundary = abb1.*max(abb2-abb3,0);
        case 'call'
            boundary = abb1.*max(abb3-abb2,0);
    end
end

%=========================================================================
%=========================================================================

%% Initialization

%Discretize time and space axes
%x is on the vertical axis
%tau is on the horizontal axis
x   = (x_min:dx:x_max)';
tau = 0:dtau:tau_max;

%For performance reasons we compute one matrix with all the g values
X = repmat(x,1,n+1);
Y = repmat(tau,m+1,1);
G = g(X,Y);

%Define values matrix with ,initial 0 values. (for all point on
%grid)(dimensions (m+1)x(n+1))
%Dimensionless option value
 w = zeros(m+1,n+1);

% Calculate bouindary contitions of grid
% initial values i = = (1 .. m+1)
w(:,1)   = G(:,1);
%Lower boundary
 w(1,:)   = G(1,:); %j = 1..n+1
%Upper boundary
 w(m+1,:) = G(end,:); %j = 1..n+1

%Righthandside is needed in core algorithm
%Initail vector of zeros
%Vector length = m-1 = internal unknown points
 b = zeros(m-1,1);

%SOR iteration vector needs to be pre-allocated only once
%Initial vector of zeros = kies as beginpunt
%Vector length = m-1 = internal unknown points
vnew = zeros(m-1,1);
%
%=========================================================================
%=========================================================================

%% Core algorithm
%Whole core algorithm is repeat for:
%j = 2 :n+1 rest of unkown time steps
% j = 1 is kknown inital condition
for j = 2:n+1 %TIME AXIS% %for time step 2 ... n+1 = tau_max = 0.5sogma^2T

    %Create righthandside b
    for k = 1:m-1 %X AXIS %internal points
        switch k
```

```
          %Use known w from earlier time increment (j-1) and w(1,j) is
          %known boundary condition.
           case 1
               b(k) = w(2,j-1)+lambda*(1-theta)*(w(1,j-1)-2*w(2,j-1)+w(3,j-1))+alpha↩
                   *w(1,j);
           case m-1
               b(k) = w(m,j-1)+lambda*(1-theta)*(w(m-1,j-1)-2*w(m,j-1)+w(m+1,j-1))+↩
                   alpha*w(m+1,j);
           otherwise
               b(k) = w(k+1,j-1)+lambda*(1-theta)*(w(k,j-1)-2*w(k+1,j-1)+w(k+2,j-1))↩
                   ;
       end
   end
%Initialize vector v
%This is one of the equations of the constrained matric notation
%Assign values to unknown node in grid at a time interval and for x nodes (2..m)
   v    = max(w(2:m,j-1),G(2:m,j));

   %The variable iter is introduced to manage the SOR iteration
   %Iteration is ended by iter = 0
   %when the derised degree of convergence has taken place
   iter = 1;

%=========================================================================
%=========================================================================
%SOR iteration
%Will remian in while looop till sufficuent convergence has taken place
while iter == 1
       %b length = 1...m-1
       %v length = 1 ... m-1
        for k = 1:m-1%X AXIS  %unknown values in grid at a specific time step
            %gauss seidel step
            switch k
                %generic gauss seidel
                %y = (b(k)+alpha*(vnew(k-1)+v(k+1)))/(1+2*alpha);

                case 1 % dont' have (k-1 = 0 )vnew(k-1) = vnew(0) term
                     %thus adjusted
                    y = (b(k)+alpha*v(k+1))/(1+2*alpha);
                case m-1 % dont' have (k+1 = m v(k+1) = vnew(m) term
                    %thus adjusted
                    y = (b(k)+alpha*vnew(k-1))/(1+2*alpha);
                otherwise
                    %the rest
                    y = (b(k)+alpha*(vnew(k-1)+v(k+1)))/(1+2*alpha);
            end

           %Projected sor = makes sure inequatiily in constraint matrix
           %quation is met Vj >= gj+1
           %vnew(k) = max[G(k+1,j),omega_R*(y) + (1 - omega_R*)(v(k))];
           %Sor uses relaxation parameter = omega_R
           %G(k+1,j) = same position as vnew(k) in matrix w
            vnew(k) = max(G(k+1,j),v(k)+omega_R*(y-v(k)));
       end

         if norm(v-vnew) <= eps
            iter = 0;
       else
            v = vnew;
       end
end
   w(2:m,j) = vnew;
```

```
end

%=========================================================================
%=========================================================================
%% Transformation to original dimensions

S = K*exp(x);
t = T-2*tau/sigma^2;
V = K*exp(-.5*(q_delta-1)*x)*exp(-(.25*(q_delta-1)^2+q)*tau).*w
%Notice, Smax is at bottom, tmax = at beginning
%Re-raagenge to fit graphical representation

%Re-arrange t and V in increasing time order
S
t = fliplr(t)
V = fliplr(V)

end
```

```
function Sf = FreeBoundary_kiki(S,t,V,K,type)

Sf = zeros(1,length(t));
eps_star = K*1e-5;

switch type
    case 'put'
        for j = 1:length(t)

            een = abs(V(:,j)-K)
            twee = abs(V(:,j)-K+S)
            drie = find(abs(V(:,j)-K+S)< eps_star, 1, 'last')
            Sf(j) = S(find(abs(V(:,j)-K+S)< eps_star, 1, 'last'))
        end
    case 'call'
        for j = 1:length(t)
            Sf(j) = S(find(abs(V(:,j)+K-S)< eps_star, 1, 'first'));
        end


end
```

8.2 Stochastic volatility MATLAB code

```
%=========================================================================
tic
format long
%=========================================================================
%PROGRAM PAREMETERS
k = 10;   %option strike price
T = 0.25;   %option expiery date / time to expiery
r = 0.1; %current constant intereest rate
xmax = 15;    %maximum value on x axis
ymax = 1;  %maximum value on the y axis
gamma = 0.9; %volatility of volatility
cor = 0.1;   %correlation coefficient between two markov processes
vega = 0;  %market price of risk
beta = 0.16;  %long term variance
alpha = 5; % rate of mean reversion
l = 1;    %number of internal nodes on the time axis
omega = 1.8;   % for PSOR convergence
eps = 0.00001;  %sufficient convergence limit

%current volatility and stock price values
y_current = 0.25
s_current = 10;

%=========================================================================
%=========================================================================
%matrix AA - MATRIX DIMENSTIONS - x-y axis matrix - at one fixed time
%period

m = 2; %number of internal nodes on the x axis
n = 3;  %number of internal nodes on the y axis

%=========================================================================
%=========================================================================
%for that increases omega values


%stipulate siza of AA ans amount of unknown nodes of the x axis * amount of
%nodes on the y axis + 1 additional row and column to compensate for known
%boundary value information
%unknown values also at boundaries x = x_max and y = y_max
size = (m+1)*(n+1);

AA = zeros(size,size);

%=========================================================================
%=========================================================================
%GRID DISCRETIZATION

dx = xmax/(m+1);
dy = ymax/(n+1);

%x axis values
for tel = 1:(m+2)
    x(tel) = (tel-1)*dx;
end

x; %displays the x-axis values after discretization
```

```
%y axis values
for teller = 1:(n+2)
    y(teller) = (teller-1)*dy;
end

y; %displays the y-axis values after discretization

%boundary conditions
%x axis - y = 0
for telx = 1:(m+2)
    bx(telx) = max(k-x(telx),0);
end

%y axis - x = 0
for tely = 1:(n+2)
    by(tely) = k;
end

bx;
by;

%=========================================================================
%=========================================================================
%time discretization
dt = T/(l+1);

%t axis values
for teller = 1:(l+2)
    t(teller) = (teller-1)*dt;
end

%display t values
t = fliplr(t);

%=========================================================================
%=========================================================================
%define a matrix containing all solutions that will be used as data for
%graph

solution = zeros((l+2)*(n+2), m+2);

%define a matrix containing all solutions at one time interval
%this will be added to solution matrix - at bottom
oplossing = zeros(n+2, m+2);

%vector that contains soutions at each new time increment u_k+1
uu = zeros(size,1);
%=========================================================================
%=========================================================================

%MATRIX COEFFICIENTS

%a_add values (1...(m+1)*(n+1))
pos = 1;
for tel = 2:((m)+2)
    for teller = 2:((n+1)+1)
        inbetween1 = min(0.5*y(teller)*x(tel)^2 - cor*gamma*y(teller)*x(tel)*dx/(2*dy↩
            ) - r*x(tel)*dx/2, 0.5*y(teller)*x(tel)^2 - cor*gamma*y(teller)*x(tel)*dx↩
            /(2*dy) + r*x(tel)*dx/2);
        add(pos) = min(inbetween1, 0);
        pos= pos+1;
```

```
    end
end

add;

%=========================================================================
%c_add values (1...(m+1)*(n+1))
pos = 1;
for tel = 2:((m+1)+1)
    for teller = 2:((n+1)+1)
        inbetween2 = min(0.5*y(teller)*x(tel)^2 - cor*gamma*y(teller)*x(tel)*dy/(2*dx
            ) - (alpha*(beta-y(teller))-vega*gamma*sqrt(y(teller)))*(dy/2), 0.5*y(
            teller)*x(tel)^2 - cor*gamma*y(teller)*x(tel)*dy/(2*dx) + (alpha*(beta-y(
            teller))-vega*gamma*sqrt(y(teller)))*(dy/2));
        cadd(pos) =  min(inbetween2,0);
        pos= pos+1;
    end
end

cadd;

%=========================================================================
%A - (1..(m+1)*(n+1))
%u_ij
pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
      A(pos) = (x(tel)^2*y(teller))/(dx^2) - (cor*gamma*x(tel)*y(teller))/(dx*dy) -
          2*add(pos)/(dx^2) + y(teller)*gamma^2/(dy^2) - 2*cadd(pos)/(dy^2) + r;
      pos = pos + 1;
 end
end

 A;

 %=========================================================================
 %B - (1..(m+1)*(n+1))
 %u_i-1j
pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
      B(pos) = (-x(tel)^2*y(teller))/(2*dx^2) + (cor*gamma*x(tel)*y(teller))/(2*dx*dy
          ) + add(pos)/(dx^2) + r*x(tel)/(2*dx);
      pos = pos + 1;
 end
end

B;

 %=========================================================================
 %C - (1..(m+1)*(n+1))
 %u_i+1j
 pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
      C(pos) = (-x(tel)^2*y(teller))/(2*dx^2) + (cor*gamma*x(tel)*y(teller))/(2*dx*dy
          ) + add(pos)/(dx^2) - r*x(tel)/(2*dx);
      pos = pos + 1;
 end
end

 C;
```

```
%=========================================================================
 %D - (1..(m+1)*(n+1))
 %u_ij-1
pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
    D(pos) = (-gamma^2*y(teller))/(2*dy^2) + (cor*gamma*x(tel)*y(teller))/(2*dx*dy) +↩
         cadd(pos)/(dy^2) - ((alpha*(beta - y(teller))) - vega*gamma*sqrt(y(teller)))↩
        /(2*dy);
    pos = pos + 1;
 end
end

D;

%=========================================================================
 %E - (1..(m+1)*(n+1))
 %u_ij+1
 pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
      E(pos) = (-gamma^2*y(teller))/(2*dy^2) + (cor*gamma*x(tel)*y(teller))/(2*dx*dy)↩
           + cadd(pos)/(dy^2) + ((alpha*(beta - y(teller))) - vega*gamma*sqrt(y(↩
          teller)))/(2*dy);
      pos = pos + 1;
 end
end

E;

%=========================================================================
 %F - (1..(m+1)*(n+1))
  %u_i+1j+1
pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
      F(pos) = -(cor*gamma*x(tel)*y(teller))/(2*dx*dy);
      pos = pos + 1;
 end
end

F;

 %=========================================================================
 %G - (1..(m+1)*(n+1))
  %u_i-1j-1
pos = 1;
for tel = 2:(m+2)
 for teller = 2:(n+2)
      G(pos) = -(cor*gamma*x(tel)*y(teller))/(2*dx*dy);
      pos = pos + 1;
 end
end

G;

%=========================================================================
%=========================================================================
%u_0 = known - vector of all unknown values = known at time t = 0.
%lenght of vector u is (m+1)*(n+1)
%used as starting point to compute all further u(time+1) vectors containing
```

```
%unknown values
%uses u(x,y,0) = max(k-x,0)

pos = 2;
for tel = 1:n+1:(m+1)*(n+1)
u(tel:tel*n+1) = bx(pos);
pos = pos +1;
end

% %cut u to have only values of importance
 u = u(1:(m+1)*(n+1));
%add final element that matches known values and additional row in
%coefficient matrix AA
%u(m*n+1) = 1;

%display initial vector u at time 0
u;

%=========================================================================
%=========================================================================
%Compile matrix AA

%=========================================================================
%main doagonal of AA - A
for tel = 1:(m+1)*(n+1)
    AA(tel,tel) = A(tel);
end

%=========================================================================
%top codiagonal of AA - E

for tel = 1:(m+1)*(n+1)
          AA(tel,tel+1) = E(tel);
end

for tel = n+1:n+1:(m+1)*(n+1)
    AA(tel,tel+1) = 0;
end

%=========================================================================
%bottom codiagonal of AA - D

for tel = 2:(m+1)*(n+1)
          AA(tel,tel-1) = D(tel);
end

for tel = n+1:n+1:(m+1)*(n+1)
    AA(tel,tel-1) = D(tel) + E(tel);
end

for tel = n+1:n+1:(m+1)*(n+1)
    AA(tel+1,tel) = 0;
end
%=========================================================================
%top diagonal of AA - contains C and F

for tel = 1:((m+1)*(n+1) - (n+1))
          AA(tel,tel+(n+1)) = C(tel);
          AA(tel, tel+(n+1)+1) = F(tel);
end

for tel = n+1:n+1:((m+1)*(n+1)-(n+1))
```

```
     AA(tel, tel+(n+1)+1) =0;
end

for tel = n+1:n+1:((m+1)*(n+1)-(n+1))
     AA(tel, tel+ (n+1)-1) = F(tel);
end

%=========================================================================
%bottom diagonal of AA - contains B, G, C and F

for tel = ((n+1)+1):((m+1)*(n+1) - (n+1))
          AA(tel,tel-(n+1)) = B(tel);
          AA(tel+1,tel-(n+1)) = G(tel+1);
end

for tel = 2*(n+1)+1:n+1:(m+1)*(n+1)
    AA(tel,tel-(n+1)-1) = 0;
end


%last n rows - bottom diagonal = B+C

for tel = ((m+1)*(n+1)-(n+1)+1):(m+1)*(n+1)
     AA(tel, tel - (n+1)) = B(tel) + C(tel);
     AA(tel, tel - (n+1) + 1) = F(tel);
end

for tel = ((m+1)*(n+1)-(n+1)+2):((m+1)*(n+1)-1)
     AA(tel, tel - (n+1)-1) = G(tel);
end

for tel = ((m+1)*(n+1) - (n+1) + 1): ((m+1)*(n+1)-1)
    AA(tel + 1, tel - (n+1)) = G(tel+1);
end

AA((m+1)*(n+1), (m+1)*(n+1) - (n+1) - 1) = F(m*n) + G(m*n);
AA((m+1)*(n+1), (m+1)*(n+1) - (n+1) + 1) = 0;

%shrink AA to be (m*n)*(m*n)-matrix
AA = AA(1:(m+1)*(n+1),1:(m+1)*(n+1));


%=========================================================================
%=========================================================================

%with time discretization:
%(I + 0.5 delta t *AA) uu = (I - 0.5 delta t *AA) u
%where u is the known vector from previous time step and uu is new unknown
%vector at new time increment

I = eye(size,size);

%COMPILE MATRICES B AND C
XX = I + 0.5*dt*AA %- in text defined as B
YY = I - 0.5*dt*AA %- in text defined as C

% %analytical solution
% ZZ = YY*u';
% uu = XX\ZZ;
u = u';

%Define initial vector xold to be used in iterative process
```

```
xold = zeros(size,1);

%initial gues to xold
%For loops calculates initial xold's value - using the constraint that
%u(k+1) >= g
%choose xold and xnew = g(x) = max(k - x( ),0)

pos = 2;
for tel = 1:n+1:(m+1)*(n+1)
xold(tel:tel*n+1) = bx(pos);
pos = pos +1;
end

% %cut u to have only values of importance
 xold = xold(1:(m+1)*(n+1));
 xnew = xold;

%numerical solution
uu = psoramericann(XX,YY,A,bx,x,size,k,u,m,n,omega,eps,xold,xnew);


%display uu - solution obtained at 1 time increment
uu;
% %=========================================================================
% %=========================================================================
% %APPLY CONSTRAINT THAT u>=g
 teller = 1;
%
for tel = 1:(m+1)
    %devides vector uu into column matrix with amount of rows = n+1
    %and columns m+1
    constraint(:,tel) = uu(teller:teller+(n+1)-1);
    teller = teller + n+1;
end

%=========================================================================
%=========================================================================
%put visually into grid of values - oplossing - MATRIX form
%DIMENSION OF oplossing = (m+2)*(n+2)
%we only need to add the initial boundary values at x = 0 and y = 0


%display as on grid
constraint = flipud(constraint)

%put constraint into its place in oplossing
oplossing(1:n+1,2:m+2) = constraint;

%boundary where x = 0
    for teller = 1:n+2
        oplossing(teller,1) = by(teller);
    end


%boundary where y = 0

     for teller = 2:m+2
         oplossing(n+2, teller) = bx(teller);
     end

%Display boudanries
%y = 0 boundary
```

```
bx;
%x = 0 boundary
by;

%putting tihs grid into its correct position in solution matrix
solution(n+3:2*(n+2),:) = oplossing

%=========================================================================
%=========================================================================
%display initial vector u at time 0 also as grid as just done
%this will be the first part of the matrix solution
%which as mentioend at the start contains all the values of the option from
%k = 0 .... l+1
%or written k = 1 ... l+2
u;

teller = 1;

for tel = 1:(m+1)
    %devides vector u into column matrix with amount of rows = n+1
    %and columns m+1
    umatrix(:,tel) = u(teller:teller+(n+1)-1);
    teller = teller + n+1;
end

%put constraint into its place in oplossing
oplossing(1:n+1,2:m+2) = umatrix;

%boundary where x = 0
    for teller = 1:n+2
        oplossing(teller,1) = by(teller);
    end


%boundary where y = 0

     for teller = 2:m+2
         oplossing(n+2, teller) = bx(teller);
     end

%putting tihs grid into its correct position in solution matrix
solution(1:n+2,:) = oplossing;

 %=========================================================================
 %=========================================================================
 %WE NOW HAVE SOLUTION AT TIME 0 AND TIME INTERVAL 1
 %which when programmed translates to u(1) and u(2)
 %REPEAT FOR TIME STEPS 3...l+2
 %that translates to finding the final price at time 0

 %set u_1 = u_0 (or u(old) = u(new))
 %now new uu value for time 2
  %which when programmed translates to u(3)
 u = uu;
 YY;
for counter = 3:(l+2)

xold = u;
xnew = xold;


%numerical solution
```

```
uu = psoramericann(XX,YY,A,bx,x,size,k,u,m,n,omega,eps,xold,xnew);


% %=============================================================================
% %=============================================================================
% %APPLY CONSTRAINT THAT u>=g
 teller = 1;
%
 for tel = 1:(m+1)
     %devides vector uu into column matrix with amount of rows = n+1
     %and columns m+1
     constraint(:,tel) = uu(teller:teller+(n+1)-1);
     teller = teller + n+1;
 end

%=============================================================================
%=============================================================================

%put visually into grid of values - oplossing - MATRIX form
%DIMENSION OF oplossing = (m+2)*(n+2)
%we only need to add the initial boundary values at x = 0 and y = 0


%display as on grid
constraint = flipud(constraint);

%put constraint into its place in oplossing
oplossing(1:n+1,2:m+2) = constraint;

%boundary where x = 0
    for teller = 1:n+2
         oplossing(teller,1) = by(teller);
    end


%boundary where y = 0

     for teller = 2:m+2
         oplossing(n+2, teller) = bx(teller);
     end

oplossing;
%Display boudanries
%y = 0 boundary
bx;
%x = 0 boundary
by;

%putting tihs grid into its correct position in solution matrix
solution((counter-1)*(n+2)+1:counter*(n+2),:) = oplossing;

%set u(i) as x(i+1) and compute new u(i+1)
u = uu;

end

%display final matrix with all values at all time values - (1...l+2)
solution
%
%only interested in values at final u(l+2)---- which correlates to pricing
%solution at time 0 of option
answer = solution((l+2)*(n+2) - (n+1):(l+2)*(n+2),:)
```

```
answer = flipud(answer)

%=========================================================================
%=========================================================================
%INTERPOLATE TO FIND CORRECT VALUE IN Y - X GRID

%isolate rows from answer that relate to y_current
for tel = 1:n+2
    if y_current > y(tel) | y_current == y(tel)
        if y_current < y(tel+1) | y_current == y(tel+1)
            y_under = tel;
            y_top = tel+1;
        end
    end
end

y_under;
y_top;
y;
dy;

y_interval = y_current/dy - (y_under-1);

%isolate these two rows form answer
solution1 = answer(y_under,:);
solution2 = answer(y_top,:);
%combined
soll = [solution1; solution2];

%isolate columns from answer that relate to s_current
for tel = 1:m+2
    if s_current > x(tel) | s_current == x(tel)
        if s_current < x(tel+1) | s_current == x(tel+1)
            x_under = tel;
            x_top = tel+1;
        end
    end
end

x_under;
x_top;
x;
dx;

x_interval = s_current/dx - (x_under-1);

%isolate these two columns from soll(2 isolated rows)
solution1 = soll(:,x_under);
solution2 = soll(:,x_top);

%interpolate two rows to find appropriate y value
final1 = (1-y_interval)*solution1(1,1) + y_interval*solution1(2,1);
final2 = (1-y_interval)*solution2(1,1) + y_interval*solution2(2,1);
toetsy = (1-y_interval)*y(y_under) + y_interval*y(y_top)

%interpolate two final values to find appropriate x value
finalsolution = (1-x_interval)*final1 + x_interval*final2;
toetsx = (1-x_interval)*x(x_under) + x_interval*x(x_top)

finalsolution

 toc
```

```
function [uu] = psoramericann(XX,YY,A,bx,x,size,k,u,m,n,omega,eps,xold,xnew);

%THE AIM IS TO DO CALCULATIONS - NOT INVOLVING ANY ZEROS contained in the
%coefficient matrix XX

%create vector containing all the payoff values for the different x values, g
%this vector is used in the PSOR stage where we obtain the maximum

g = xold;

%Now the iterative process , that converges to the true numerical solution
%- denoted as uu starts.
%We have the following XXu(k+1) = YYu(k)
%We have to solve u(k+1) but we have already chosen an initial guess xold
%We also have u
%Next we have to multiply YYu(k) to write the equation as
%XXu(k+1) = b = Au(k+1)

b = YY*u;

%=========================================================================
%=========================================================================
%% y
%vector y containing all intermediate numerical solutions
%refer to section on SOR in text

 y = zeros((n+1)*(m+1),1);

%iter = 1 = this will enter while loop that runs till method converged
iter = 1;
res = 0;

while iter == 1

%=========================================================================
%only rows 1...n+1
y(1) = (1/XX(1,1))*(b(1) - xold(2)*XX(1,2) - xold(n+2)*XX(1,n+2) - xold(n+3)*XX(1,n↩
    +3));

 xnew(1) = max(omega*(y(1) - xold(1)) + xold(1), g(1));


for tel = 2:n

    y(tel) = (1/XX(tel,tel))*(b(tel) - xnew(tel-1)*XX(tel,tel-1) - xold(tel+1)*XX(tel↩
        ,tel+1) - xold(tel+(n+1))*XX(tel,tel+(n+1)) - xold(tel+(n+2))*XX(tel,tel+(n↩
        +2)));

    xnew(tel) = max(omega*(y(tel) - xold(tel)) + xold(tel), g(tel));


end

y(n+1) = (1/XX(n+1,n+1))*(b(n+1) - xnew(n)*XX(n+1,n) - xold(n+(n+1))*XX(n+1,n+(n+1)) ↩
    - xold(n+(n+2))*XX(n+1,n+(n+2)));

xnew(n+1) = max(omega*(y(n+1) - xold(n+1)) + xold(n+1), g(n+1));
```

```
%=========================================================================
%rows (n+1)+1... (m+1)*(n+1)-(n+1)

y((n+1)+1) = 1/XX(n+1+1,n+1+1)*(b(n+1+1) - xnew(n+1+1-(n+1))*XX(n+1+1,1) - xold(n↩
    +1+2)*XX(n+1+1,n+1+2) - xold(n+1+(n+2))*XX(n+1+1,n+1+(n+2)) - xold(n+1+(n+3))*XX(↩
    n+1+1,n+1+(n+3)));

xnew(n+1+1) = max(omega*(y(n+2) - xold(n+1+1)) + xold(n+1+1), g(n+1+1));


for tel = (n+1)+2 : (m+1)*(n+1)-(n+1)-1

    y(tel) = (1/XX(tel,tel))*(b(tel) - xnew(tel - (n+2))*XX(tel,tel - (n+2)) - xnew(↩
        tel - (n+1))*XX(tel ,tel - (n+1)) - xnew(tel-1 )*XX(tel ,tel-1) - xold(tel+1)↩
        *XX(tel,tel+1) - xold(tel+n)*XX(tel,tel+n) - xold(tel+n+1)*XX(tel ,tel+n+1) -↩
         xold(tel+n+2)*XX(tel ,tel+n+2));

    xnew(tel) = max(omega*(y(tel) - xold(tel)) + xold(tel), g(tel));


end

tel = (m+1)*(n+1)-(n+1);

y(tel) = (1/XX(tel,tel))*(b(tel) - xnew(tel - (n+2))*XX(tel,tel - (n+2)) - xnew(tel -↩
     (n+1))*XX(tel ,tel - (n+1)) - xnew(tel-1 )*XX(tel ,tel-1) - xold(tel+1)*XX(tel,↩
    tel+1) - xold(tel+n)*XX(tel,tel+n) - xold(tel+n+1)*XX(tel ,tel+n+1));

xnew(tel) = max(omega*(y(tel) - xold(tel)) + xold(tel), g(tel));


%=========================================================================
%rows (m+1)*(n+1)-(n+1)+ 1 ... (m+1)*(n+1)

for tel = (m+1)*(n+1)-(n+1)+ 1:(m+1)*(n+1)-1

    y(tel) = 1/XX(tel,tel)*(b(tel) - xnew(tel - (n+2))*XX(tel,tel - (n+2)) - xnew(tel↩
         - (n+1))*XX(tel,tel - (n+1)) - xnew(tel - (n))*XX(tel,tel - (n)) - xnew(tel ↩
        - (1))*XX(tel,tel - (1)) - xold(tel +1)*XX(tel,tel +1));

    xnew(tel) = max(omega*(y(tel) - xold(tel)) + xold(tel), g(tel));


end

tel = (m+1)*(n+1);

%last row of vector y
y(tel) = 1/XX(tel,tel)*(b(tel) - xnew(tel - (n+2))*XX(tel,tel - (n+2)) - xnew(tel - (↩
    n+1))*XX(tel,tel - (n+1)) - xnew(tel - (n))*XX(tel,tel - (n)) - xnew(tel - (1))*↩
    XX(tel,tel - (1)));

xnew(tel) = max(omega*(y(tel) - xold(tel)) + xold(tel), g(tel));
%
 if norm(xnew-xold)<=eps
     iter = 0;

 else
     xold = xnew;
 end

end
```

```
uu = xnew;
end
```

Bibliography

Acworth, W. 2012. Annual volume survey: Volume climbs 11.4% to 25 billion contracts worldwide, Futures Industry: The magazine for futures, options and cleared swaps. http://www.futuresindustry.org/downloads/Volume Survey(03-12 FI).pdf Date of access: 20 Feb. 2012.

Ballestra, L. V. and Sgarra, C. 2010. The evaluation of american options in a stochastic volatility model with jumps: An efficient finite element approach, *Computers and Mathematics with Applications* **60**: 1571–1590.

Bjork, T. 2009. *Arbitrage theory in continuous time*, Oxford University Press, New York.

Brandimarte, P. 2006. *Numerical methods in finance and economics: A Matlab-based introduction*, John Wiley Sons, Inc., New Jersey.

Chance, D. M. 2003. *Analysis of derivatives for the CFA program*, AIMR, New York.

Chockalingam, A. and Muthuram, K. 2011. American options under stochastic volatility, *Operations Research* **59**(4): 793–809.

Clarke, N. and Parrott, K. 1999. Multigrid for american option pricing with stochastic volatility, *Applied Mathematical Finance* **6**(3): 177–195.

Duffy, D. J. 2004. A critique of the crank nicolson scheme strengths and weaknesses for financial instrument pricing, WILMOTT magazine. http://www.wilmott.com/pdfs/071203 duffy.pdf Date of access: 18 Sept. 2012.

Duffy, D. J. 2006. *Finite difference methods in financial engineering : a partial differential equation approach*, John Wiley Sons Ltd, West Sussex.

DuFour, R. D. 2011. Otc solutions in an exchange environment, FOCUS: The monthly newsletter of regulated exchanges. http://www.world-exchanges.org/insight/views/otc-solutions-exchange-environment Date of access: 5 March 2012.

Dukkipati, R. V. 2010. *Numerical methods*, New Age International Limited, New Delhi.

Elhashash, A. and Szyld, D. B. 2008. Generalizations of m-matrices which may not have a nonnegative inverse, *Linear Algebra and its Applications* **429**: 2435–2450.

Feng, L., Linesky, V., Morales, J. L. and Nocedal, J. 2011. On the solution of complimentary problems arising in american options pricing, *Optimization methods and Software* **20**(4-5): 813–825.

Forsyth, P. A. 2008. An introduction to computational finance without agonizing pain. https://cs.uwaterloo.ca/ paforsyt/agon.pdf date of access: 10 may 2012.

Higham, D. J. 2009. *An introduction to financial option valuation: Mathematics, stochastics and computation*, Cambridge University Press, New York.

Hull, J. C. 2000. *Options, futures, and other derivatives*, Pearson Education, Inc., New Jersey.

Hull, J. C. 2009. *Options, futures, and other derivatives*, Pearson Education, Inc., New Jersey.

Hull, J. C. 2012. *Options, futures, and other derivatives, Global edition*, Pearson Education Limited, Essex.

Ikonen, S. and Tiovanen, J. 2004. Operator splitting methods for american options with stochastic volatility. european congress on computational methods in applied sciences and engineering.

Ikonen, S. and Tiovanen, J. 2008. Efficient numerical methods for pricing american options under stochastic volatility, *Numerical Methods for Partial Differential Equations* **24**(1): 104–126.

Ikonen, S. and Tiovanen, J. 2009. Operator spitting methods for american options with stochastic volatility, *Numerische Mathematik* **113**(2): 299–324.

Karris, S. T. 2007. *Numerical analysis using Matlab and Excel*, Orchard publications, Fremont.

Kau, J. 2009. *Stochastic volatility models with jumps and high frequency data*, Master's thesis, University of Aarhus.

Kincaid, D. and Cheney, W. 1991. *Numerical analysis mathematics of scientific computing*, Wadsworth, Inc., Belmont, California.

Kreyszig, E. 2006. *Advanced engineering mathematics*, John Wiley Sons, Inc., New Jersey.

Kwok, Y. 2008. *Mathematical models of financial derivatives*, Springer, Berlin.

Merton, R. C. 1976. Option pricing when underlying stock returns are discontinuous, *Journal of Financial Economics* **3**(1-2): 125–144.

Murty, K. G. 1988. *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann Verlag, Berlin.

Oliver, P. 2004. *Numerical simulation of american options*, Master's thesis, University of Ulm.

Oosterlee, C. W. 2003. On multigrid of linear complimentary problems with application to american-style options, *Electronic Transactions on Numerical analysis* **15**: 165–185.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. 2007. *Numerical recipes: The art of scientific computing*, Cambridge University Press, New York.

Rodolfo, K. 2007. *A comparative study of American option evaluation and computation*, PhD thesis, University of Sydney.

Salmi, S. and Toivanen, J. 2011. An iterative method for pricing american options under jump-diffusion models, *Applied Numerical Mathematics* **61**: 821–831.

Seydel, R. U. 2009. *Tools for computational finance*, Springer-Verlag, Berlin.

Ugur, O. 2008. *Series in quantitative finance - Vol. 1: An introduction to computational finance*, Imperial College Press, River Edge, New Jersey.

Wilmott, P. 2000a. *Derivatives: The theory and practice of financial engineering*, John Wiley Sons, Ltd., West Sussex.

Wilmott, P. 2000b. *On quantitative finance: Volume one*, John Wiley Sons, Ltd., West Sussex.

Wilmott, P. 2009. *Frequently asked questions in quantitative finance*, John Wiley Sons, Ltd., West Sussex.

Wilmott, P., Dewynne, J. and Howison, S. 1996. *The mathematics of financial derivatives*, Press syndicate of the University of Cambridge, Cambridge.

Wilmott, P., Dewynne, J. and Howison, S. 2000. *Option pricing: Mathematical models and computation*, Oxford Financial Press, Oxford.

Yang, W. Y., Cau, W., Chung, T. and Morris, J. 2005. *Applied numerical methods using Matlab*, John Wiley Sons, Ltd., Hoboken, New Jersey.