Pricing multi-asset options with sparse grids

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties, in het openbaar te verdedigen
op vrijdag 13 juni 2008 om 12:30 uur

door

Coenraad Cornelis Willem LEENTVAAR
Natuurkundig ingenieur en wiskundig ingenieur

geboren te 's Gravenhage

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. C.W. Oosterlee

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. C.W. Oosterlee	Technische Universiteit Delft, promotor
Prof.dr.ir.drs. H. Bijl	Technische Universiteit Delft
Prof.dr.ir. A.W. Heemink	Technische Universiteit Delft
Prof.dr. A.K. Parrott	The University of Greenwich
Prof.dr. J.M. Schumacher	Universiteit van Tilburg
Prof.dr. R.U. Seydel	Universität zu Köln
Prof.dr.ir. C. Vuik	Technische Universiteit Delft

Pricing multi-asset options with sparse grids.
Proefschrift aan de Technische Universiteit te Delft

This research was financially supported by the Dutch Technology Foundation STW DWI 6322.

ISBN 978-90-8559-382-9

Samenvatting

Het prijzen van opties op meerdere aandelen met behulp van dunne roosters.

Coenraad Cornelis Willem Leentvaar

In tegenstelling tot standaard opties, zijn opties op een mandje met aandelen gebaseerd op meerdere onderliggende aandelen. Dit fenomeen maakt het bepalen van de optieprijs tot een grotere uitdaging. Een van de belangrijkste problemen is de stijgende probleemdimensionaliteit. Bij het stijgen van de dimensionaliteit, neemt de complexiteit van het onderliggende probleem exponentieel toe, omdat het aantal onbekenden dat opgelost dient te worden exponentieel groeit. Huidige computersystemen kunnen niet overweg met een dergelijk grote hoeveelheid data.

Teneinde het meerdimensionale optieprobleem op te lossen, moet er een geavanceerde numerieke oplostechniek gevonden worden. Een van de technieken is de zogeheten dunne roostertechniek. Deze techniek splitst het probleem op in een substantieel aantal deelproblemen van een lagere complexiteit, die allen hanteerbaar zijn voor een modern computersysteem. Omdat ieder deelprobleem dat uit deze splitsing ontstaat onafhankelijk van ieder ander deelprobleem is, leent de dunne rooster techniek zich optimaal voor parallellisatie. Dat wil zeggen dat in het optimale geval alle deelproblemen simultaan opgelost kunnen worden. Echter, gezien de dimensionaliteit, kan een deelprobleem nog steeds te groot zijn. In dat geval dient dat deelprobleem verder geparallelliseerd te worden.

De dunne rooster methode kan niet straffeloos worden toegepast. De beperkingen aan de toepasbaarheid van de dunne rooster methode liggen in de begrensdheid van de gemengde afgeleiden van de oplossing van het probleem. Omdat de eindvoorwaarde van vele optieprijsproblemen niet differentieerbaar is, dient met deze beperking zorgvuldig te worden omgegaan.

In de eerste oplosmethode in dit proefschrift, wordt aan de hand van experimenten aangetoond dat zonder gebruik te maken van geavanceerde roostertransformatietechnieken, een partiële differentiaal vergelijking in combinatie met dunne roosters niet tot een gewenst convergentieresultaat leidt. Indien een coördinatentransformatie wordt toegepast, verbetert dit

resultaat aanzienlijk. De coördinatentransformatie dient er voor de niet-differentieerbaarheid in de eindvoorwaarde langs een roosterlijn te leggen. Echter, niet alle prijzen van opties op meerdere aandelen kunnen voldoende nauwkeurig bepaald worden met een coördinatentransformatie. Soms is transformatie niet nodig, zoals bij opties die gebaseerd zijn op het best of slechtst presterend aandeel. Deze optiecontracten hebben een eindvoorwaarde die automatisch langs een roosterlijn ligt. Omdat het niet eenvoudig is om passende randvoorwaarden voor dit probleem te definiëren, is voor het prijzen van deze opties gebruik gemaakt van de tweede, alternatieve methode in het proefschrift. Deze methode is gebaseerd op het bepalen van een meervoudige integraal die voortkomt uit de risicovrije verwachtingswaarde van een optie. De integraal kan zeer efficiënt doorgerekend worden met behulp van een meervoudige discrete Fouriertransformatie.

De snelle Fourier transformatie is een efficiënt algoritme om de discrete Fourier transformatie te berekenen. Dit algoritme is ook de basis voor een algoritme voor het parallel uitrekenen van de transformatie, door het probleem op te splitsen in stukken. In dit proefschrift is een volledig communicatievrij parallel algoritme ontwikkeld dat ervoor zorgt dat het probleem op een slimme manier wordt gesplitst. In combinatie met de dunne roostermethode, kunnen hier voldoende nauwkeurige resultaten worden behaald voor hoogdimensionale problemen.

Summary

Pricing multi-asset options with sparse grids.

Coenraad Cornelis Willem Leentvaar

Multi-asset options are based on more than one underlying asset, in contrast to standard vanilla options. A very significant problem within the pricing techniques for multi-asset options is the curse of dimensionality. This curse of dimensionality is the exponential growth of the complexity of the problem when the dimensionality increases, because the number of unknowns to solve simultaneously grows exponentially. Modern computer systems cannot handle this huge amount of data.

In order to handle the multi-dimensional option pricing problem, the curse of dimensionality has to be dealt with. The sparse grid solution technique is one of the key techniques to do this. The sparse grid technique divides the original problem into many smaller sized sub-problems, which can be handled efficiently on a modern computer system. Because every sub-problem is independent of all others, this technique is parallelisable at a high efficiency rate. This means, that every sub-problem can be solved simultaneously. However, because of the dimensionality, the size of the sub-problems may remain too large to solve and should be parallelised further.

The main restriction to the application of the sparse grid method is that the mixed derivative of the solution of a multi-dimensional option pricing problem has to be bounded. Because of the typical non-differentiability of the final condition of the option pricing problem, this restrictions has to be taken seriously.

In the first part of this thesis, it is shown, experimentally, that indeed the sparse grid technique does not lead to a satisfactory accuracy without the use of advanced techniques. If a coordinate transformation is used, the accuracy increases significantly. This transformation aligns the non-differentiability along a grid line.

Coordinate transformations are not applicable to any type of multi-asset option, which seriously restricts the sparse grid solution technique for real life financial applications. Sometimes, however, it is not necessary to use it, because the non-differentiability is already aligned with grid line. These

types of options are the options based on the best or worst performing underlying asset. The boundary conditions of these contracts are unknown en henceforth these options are computed and analysed with a second alternative method in this thesis. This method arises from the risk-neutral expectation valuation of the final condition which can be written as a multi-dimensional integral over the transition density. By use of a discrete Fourier transform, we can solve this integral efficiently.

The fast Fourier transform is a fast algorithm to compute the discrete Fourier transform. This algorithm serves as the basis for a sophisticated algorithm to parallelise the computation of the discrete Fourier transform, by dividing the transform in several parts. In this thesis, a complete parallel algorithm which does not require communication between the sub-problems is developed, which subdivides the problem in a sophisticated way. In combination with the sparse grid technique, the numerical results have a satisfactory accuracy.

Contents

List of Tables

List of Figures

Preface

In the last twenty years of the 20th century, financial derivative products have become increasingly important in the world of finance. Many different derivatives are traded over all major stock exchanges in the world. Typical examples of derivatives are options, swaps, futures, forward contracts and many others. We have now reached the stage where anyone in finance needs the knowledge of how derivatives work, how they are used and priced.

Derivatives can be defined as financial instruments whose value depends on the value of other underlying variables. These underlying variables are also tradable, but they are of a more basic type than the derivatives, although derivative contracts with other derivatives as underlying do indeed exist. In far the most examples of derivatives, the underlying variable is the price of an asset. A stock option, for example, is a derivative contract with the price of the stock as underlying. But derivatives can be based on almost every variable, for example the amount of sunshine in a popular region of Spain [29].

Options occur in many forms. Examples are vanilla options, barrier options, digital options and multi-asset options. The basic example of an option is the vanilla option. An option contract is an agreement between a buying party (the holder) and a selling party (the underwriter). The holder of the option contract has no obligation to use his option contract, whereas the underwriter is obliged to agree with the holder if the holder uses the option contract.

A simple example of a vanilla option is the European call option on a stock. A call option gives the holder the right to buy the stock at a prescribed moment in the future, the maturity date, for a prescribed price, the strike price. Because the holder has the *right* to use his option, he can decide what to do at maturity. The use of the right prescribed in the option contract is called exercising the option. Basically, two scenarios may occur at the maturity date. First, the price of the stock is less than the strike price. Then, if the holder exercises his option, he buys a stock for an amount equal to the strike price. This is not in his favour, because the holder may buy the same stock on the stock exchange for a smaller amount. Contrarily, if the stock price is above the strike price, then the holder can make a profit if he exercises his option. This kind of option trading is called speculating.

A second example of trading options, is hedging. Suppose an investor possesses a stock and he expects a decrease of the stock price. Then he may buy a European put option, which gives the holder the right to sell his stock for the strike price. If the stock price goes down below the strike, the holder uses the put option to sell his stock for the strike price. Otherwise, the holder has a stock in his possession with a higher price. This type of trading, hedging, is used to reduce the risk.

Another group of option traders are the so-called arbitrageurs. These persons may buy the same option contracts on different stock exchanges with different currencies (for example in Amsterdam and in London). Then if the currency rate fluctuates, the options at either side of the North-Sea can be cheaper. The arbitrageur can sell the cheaper options at the other stock exchange and he will make a risk-free profit. This is called arbitrage. Arbitrage cannot last for long, because the forces of supply and demand will cause the currency rate between the Sterling and the Euro to change.

We already discussed the put and call options. There is no greater distinction in the class of options than between these two options. Another distinction is the time point of exercising the option. A European contract can only be exercised at the end of the lifetime of the option, while the American option can be exercised on at any time point prior to the exercise date (early exercise). In 1973, a paper from Fischer Black and Myron Scholes appeared [8], which led to the famous Black-Scholes equation. In 1997, the Nobel prize in economics was awarded for this work. This framework is still in use nowadays as the basic fundamental understanding of the option pricing theory, although the framework clearly has its drawbacks.

One of the exciting aspects is the creation of non-standard products by financial engineers. These non-standard products, the so-called exotic options, are important for investors because these options are mostly more profitable than the plain vanilla counterparts. Exotic options are basically derived from the vanilla options, but they have some extra properties. For example a barrier option may become worthless if the underlying stock price hits a certain barrier, which means that the stock price crosses a certain value. The vanilla options have well-defined properties and are traded actively. Their prices are quoted by supply and demand on stock exchanges by regular brokers. The exotic options are traded in the over-the-counter derivatives market.

Exotic options are developed because of numerous reasons. Sometimes they meet a genuine hedging need in the market or there are other reasons - e.g. tax, legal, accounting - to make exotic options attractive for financial institutions. One type of exotic option that is traded also by regular brokers, is the index option. This option makes the link between the options on only one underlying variable to the options on more underlying variables. The options on more than one underlying asset, the so-called multi-asset options, are the topic of this thesis. There exist, for example, cross-currency

options that involve the exchange of more than two currencies against a base currency at expiration. The basket option buyer purchases the right to receive designated currencies in exchange for a base currency, either at the prevailing foreign exchange market rate or at a prearranged rate of exchange. Multinational corporations with multi-currency cash flows frequently use basket options because it is generally cheaper to buy an option on a basket of currencies than to buy individual options on each of the currencies that make up the basket.

Another type of a multi-asset option is a basket option. This type is an option on a basket of stocks. For example a basket based on a stock of Shell and a stock of Philips is an example of a two-asset basket option. For a basket call option, if the value of the basket is above the strike price, it is favourable to buy all the stocks in the basket. The price of a basket option is usually cheaper than the two individual options, since the value of the option is based on the average of the stocks. Furthermore, the transaction costs, are based on one option in contrast to the two individual options. Other more sophisticated options, for example, are rainbow options, i.e. options based on the best performing stock, geometric options or highly exotic examples with additional properties like barriers.

Financial institutions work with sophisticated software programs to calculate the value of a portfolio with assets and options. Also options on several assets attract big interest. To price these options partial differential equations from computational finance may need to be solved. However, they cannot yet be solved with a similar efficiency as the traditional options on a single asset. At the same time, basket options lead to interesting challenging questions in numerical mathematics. Two factors that determine the difficulty of numerically pricing and hedging these options are the number of underlying assets, i.e., the problem dimensionality and the so-called 'early exercise' possibility. For low-dimensional problems (fewer than four dimensions) well-known classical discretisation techniques are an obvious choice for solving the partial differential equations with methods from numerical mathematics. These methods can cope well with early exercise and are relatively fast. For higher dimensions (above ten dimensions), Monte Carlo simulations are in principle adequate, but relatively slow and not very efficient for American-style options. Nowadays, basket option pricing problems between three and ten dimensions occur frequently.

The main goal of the research in this thesis was to find novel numerical methods for solving the medium dimensional (3-10 dimensions) problems. There is currently no numerical method that copes well with such problems. Notice that, without advanced numerical techniques, the solution of a discrete five-dimensional partial differential equation, for example, with 32 points in each dimension will already give rise to 33 million computational points each time step. The computational work is therefore huge for higher-dimensional problems. For dimensions less than ten, however, it must still

be possible to reduce the total number of computational points based on the sparse grids technique and solve the discrete equations with the most efficient solution methods on parallel computers.

Expertise from computer science in solving discrete partial differential equations on parallel machines is necessary, when working with an extremely large number of computational points. The ingredients for a breakthrough, like accurate discretisation techniques, non-uniform sparse grids and fast iterative solution methods, have been developed and used successfully in the computational fluid dynamics (CFD) area, however, only for 3D problems. These modern numerical techniques will enable to reduce the total number of grid points and will also provide an accurate approximation of the derivatives of the option value. These derivatives, called the Greeks, are of major interest to financial engineers, as they indicate the sensitivity of a portfolio under consideration.

In this thesis, two different numerical approaches to pricing options are discussed and tested in combination with the sparse grid techniques developed in [10]. This technique breaks the curse of dimensionality [3] by dividing the problem in a large number of significantly smaller-sized subproblems. This thesis shows the advantages and the drawbacks of the sparse grid method in combination with the option pricing problem. This thesis can therefore be seen as a detailed numerical study of combining existing methods to reach the highest possible dimensionality.

Chapter 1

Introduction

1.1 Motivation

The work in this thesis is a description of some pricing techniques for multi-asset options under the Black-Scholes framework. The focus is on the methods and the results that they yield. We combine several different methods together to improve the computational times as well as the accuracy and to obtain higher levels of dimensionality. The *curse of dimensionality* is the main course in the menu of this thesis, because it has a direct influence on the maximum problem size and the computational time. This thesis leads the reader from the classic stochastic derivation of the multi-dimensional Black-Scholes equation to the computational science of high-dimensional problems in terms of partial differential equations or multi-dimensional Fourier transforms.

1.2 Overview

The thesis is divided into five chapters. The first chapter, viz: the introduction, presents the derivation of option pricing techniques and a description of several option contracts. The derivation of the pricing techniques is done via the risk-neutral valuation of the option price and via the risk-free portfolio and Itô's lemma. Both the partial differential and the integral approaches are presented.

Chapters 2 and 3 present the numerical solution of the pricing problem using a partial differential equation. Chapter 2 covers the numerical solution of the single-asset problem. The discretisation of the derivatives is discussed as well as grid stretching, an iterative method for pricing American style options and interpolation techniques for discrete dividend. In Section 2.5 we show at work an accurate method for pricing single-asset option contracts.

In Chapter 3, the PDE method for pricing options continues; however we treat the multi-dimensional case there. The discretisation used in Chap-

ter 2 is extended to the general multi-dimensional problem using Kronecker products. We prove that the Kronecker products can be used for each derivative and discretisation, if the discretised solution can be written as a linear combination of the approximated solutions.

The curse of dimensionality is an important issue in this thesis and in Section 3.3, the sparse grid technique for general multi-dimensional partial differential equations is presented. The method can be used for high-dimensional problems under some restrictions. For option pricing, we will see that some restrictions get violated and therefore we have to use advanced coordinate transformations to obtain a good accuracy. These coordinate transformations can be applied to basket options and we end with the presentation of numerical results for basket options.

Chapter 4 contains the option pricing technique through a Fourier transform. The Fourier transform is discretised and leads to the computation of a multi-dimensional discrete Fourier transform (DFT). The DFT is computed with the efficient fast Fourier transform (FFT). However, the curse of dimensionality is still an issue and we discuss the possibilities of parallelisation of the DFT. In combination with the sparse grid technique, we find a sophisticated algorithm to solve multi-dimensional option pricing problems.

Finally, in Chapter 5 , we draw the conclusions of this thesis and discuss the possibilities of both methods. Furthermore some remarks about further quantitative research are given.

1.3 Computational finance

1.3.1 Price processes

The derivation of the price and the derivative processes in this section are based on [7]. The theory of financial derivatives is based on the behaviour of the prices of their underlying assets. For example a price of a call option increases with the price of the underlying asset. The question is: What kind of process drives the price? First we define two types of processes: deterministic processes and stochastic processes.

Definition 1.3.1. A price process B is said to be deterministic if it has the dynamics

$$dB(t) = r(t)B(t)dt, \tag{1.1}$$

where $r(t)$ is a given deterministic function, $dB(t)$ is the change of B in a period of dt.

A typical example is the risk-free interest rate applied by a bank if the money is put on an account. In contrast to a risk-free price process, a stock price process can be modelled as:

$$d\bar{S}(t) = \left(\mu(t, \bar{S}(t)) - \delta\right) \bar{S}(t)dt + \sigma(t, \bar{S}(t))\bar{S}(t)d\bar{W}(t) \tag{1.2}$$

This process describes the change in the stock price $d\bar{S}(t)$ in terms of deterministic drift part and a stochastic part. Here the functions $\mu(t, \bar{S}(t))$ and $\sigma(t, \bar{S}(t))$ are deterministic as well, but the term $\bar{W}(t)$ is a *Wiener* process or a geometric Brownian motion. The term $\delta\bar{S}dt$ is the dividend payment if the underlying stock pays dividend. This dividend payment is independent of time except through the dependence of $\bar{S}$. The *dividend yield*, δ is defined as the proportion of the asset paid out per unit time dt. If the stock price does not fall down the amount $\delta\bar{S}dt$, then a trader can make a risk-free profit by buying the asset.This is an example of *arbitrage*.

Equation (1.2) is known as a stochastic differential equation. An important case and also the key model in this thesis is the celebrated *Black-Scholes model*:

Definition 1.3.2. The $d-$dimensional Black-Scholes model consists of $d+1$ assets with dynamics:

$$dB(t) = rB(t)dt \tag{1.3}$$
$$d\bar{S}_i(t) = (\mu_i - \delta_i)\bar{S}_i(t)dt + \sigma_i\bar{S}_i(t)d\bar{W}_i(t), \tag{1.4}$$

where r, μ_i, δ_i and σ_i are deterministic constants and $\mathbb{E}\{d\bar{W}_i(t)d\bar{W}_j(t)\} = \rho_{ij}dt$ represents the correlation between the stocks.

This model describes the behaviour of a multi-dimensional price process when the constants are given. The constant μ_i is also called the drift term and it incorporates the risk-free interest rate r. The deterministic constants are a drawback of the model as many improvements of this model are made. However, for multi-asset option pricing, this model is still in use.

The Wiener processes are taken on a real-world probability measure P. To apply the Feynman-Kac̃ theorem for contingent claims (see next section), the probability measure P is replaced by another probability measure Q in such way that $\frac{S_i(t)}{B(t)}$ is a martingale with respect to the so-called Q probability measure. The use of probability measure Q is also referred to the *risk-neutral valuation*. In the risk-neutral world the stock price will have the $Q-$dynamics. Secondly, a future stochastic payment would be equal to the expected value of the payments discounted by the risk-free interest rate. In this case we have $Q = P$. The Black-Scholes in the risk-neutral world reads:

Definition 1.3.3. The $d-$dimensional risk-neutral Black-Scholes model consists of d assets with dynamics:

$$dS_i(t) = (r - \delta_i)S_i(t)dt + \sigma_iS_i(t)dW_i(t), \tag{1.5}$$

where r is the interest rate, δ_i is the continuous dividend yield and $\mathbb{E}^Q\left(dW_i(t)dW_i(t)\right) = \rho_{ij}dt$ represents the correlation between the stocks w.r.t. the risk-neutral measure.

1.3.2 Derivatives

Options on stocks or other types of products driven by the Black-Scholes model are called *derivatives* of the stocks. Derivatives have a contract function, which is called Φ. A typical example of an option is a European call option:

Definition 1.3.4. A European call option with strike price K and exercise date T on one underlying asset is a contract defined as follows:

- The holder of the option has the right to buy the underlying stock S for the exercise price K at the exercise date T from the underwriter of the option.
- The holder has the right to buy the underlying asset at maturity date T only, but is **not** obliged to buy the underlying stock.

The financial terminology of *exercising the option* means the execution of the right of the option. The European call option is just one simple example of an option contract. More contract functions are presented in Section 1.4.

The option contract stated in Definition 1.3.4 is a contingent claim:

Definition 1.3.5. Consider a price process with stocks, S_i, as used in the Black-Scholes model. A **contingent claim** with exercise date T (also called $T-$claim) is a stochastic variable driven by the price process as defined in Definition 1.3.3.

If at the exercise date the stock price is greater than K, a profit of $S(T)-K$ can be made by exercising the option and selling the stock immediately. However, if $S(T) < K$, then it is not profitable to exercise the option and the value of the contingent claim is zero. In other words:

$$\Phi(T, S(T)) = \max\{S(T) - K, 0\}. \tag{1.6}$$

Now, the question arises as to what the today's price of the contingent claim will be. This is the main topic of this thesis: finding a method to compute the price of certain contingent claims. First of all, the pricing equation should be known. The true value of the contingent claim is only known at the end of the lifetime of the claim.

In Definition 1.3.3 the risk-neutral price process of the underlying stocks is presented in the celebrated Black-Scholes model. Together with the properties of the contingent claim, the following definition can be postulated:

Definition 1.3.6. The risk-neutral price of the claim, Φ, on the underlying stocks, S_i, as in Definition 1.3.3 is given by a function, $V(t, S_i(t))$:

$$V(t, S_i) = e^{-r(T-t)}\mathbb{E}^Q\{\Phi(T, S_i(T))\}. \tag{1.7}$$

The economic background of Definition 1.3.6 is that the price is the expectation of the value of the contract at exercise date T and then discounting it by the factor $e^{-r(T-t)}$. The expectation can be taken with respect to the risk-neutral probability distribution:

$$\mathbb{E}^Q\{\Phi(T, S_i(T))\} = \int_{\mathbb{R}^d} \Phi(T, S_i(T)) f(\mathbf{S}(t)|\mathbf{S}(T)))dS_1(T)\dots dS_d(T), \quad (1.8)$$

with $\mathbf{S}(t) = [S_1, \dots, S_d]^T$. The probability density function $f(\mathbf{S}(t)|\mathbf{S}(T))$ is here the log normal conditional distribution function, which means that the logarithms of the stock prices are multivariate normally distributed with correlation coefficients ρ_{ij} and volatilities σ_i as given in Definition 1.4. The expectation is the computation of the value of V at time t and asset prices $S_i(t)$ when $S_i(T)$ is given. This is called a *transition density*. This integral representation forms the basis of the Fourier based pricing techniques, that will be described in detail in Chapter 4

Chapters 2 and 3 cover the pricing technique using a partial differential equation. The next theorem provides a link between the integral representation and the partial differential equation:

Theorem 1.3.7 (Feynman-Kač)**.** *Given the system of stochastic differential equations:*

$$dS_i(t) = (r - \delta_i)S_i(t)dt + \sigma_i S_i dW_i(t)$$

with $\mathbb{E}^Q\{dW_i(t)dW_j(t)\} = \rho_{ij}dt$ *and a contingent claim,* V*, such that*

$$\begin{aligned} V(t, S_i) &= e^{-r(T-t)}\mathbb{E}^Q\{\Phi(T, S_i(T))\} \\ &= e^{-r(T-t)} \int_{\mathbb{R}^d} \Phi(T, S_i(T)) f(\mathbf{S}(t)|\mathbf{S}(T)))dS_1(T)\dots dS_d(T), \end{aligned}$$

with the sum of the first derivatives of the option square integrable. Then the value, $V(t, S_i)$*, of the contingent claim at time,* t *is the unique solution of the final condition problem:*

$$\begin{cases} \dfrac{\partial V}{\partial t} + \displaystyle\sum_{i=1}^{d}(r - \delta_i)S_i\frac{\partial V}{\partial S_i} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} - rV = 0 \\ V(T, S_i) = \Phi(T, S_i(T)), \end{cases} \quad (1.9)$$

with $V(t, S_i) : \mathbb{R}^d_+ \times \mathbb{R}_+ \to \mathbb{R}$ *and* $\rho_{ii} = 1$*.*

For a proof see [7] or [35]. Equation (1.9) is a second order parabolic partial differential equation in d $S-$dimensions and plus a time dimension. The numerical solution of this type of equation is of our interest, as very few problems admit an analytic solution. The main problems when solving this equation are its dimensionality and the type of final condition.

Equation (1.9) in one dimension is the celebrated **Black-Scholes partial differential equation**:

$$\begin{cases} \dfrac{\partial V}{\partial t} + (r-\delta)S\dfrac{\partial V}{\partial S} + \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 V}{\partial S^2} - rV = 0 \\ V(T,S) = \Phi(T, S(T)), \end{cases} \tag{1.10}$$

with $\Phi(T, S(T))$ the contract function of an option.

Another result of the Feynman-Kač Theorem 1.3.7, is that the solution of (1.8) with Φ in equation (1.6) for a call option can be written as:

$$V(t,S) = Se^{-\delta(T-t)}N(d_1) - Ke^{-r(T-t)}N(d_2) \tag{1.11}$$

with

$$d_1 = \frac{\ln S - \ln K + (r - \delta + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{1.12}$$

$$d_2 = \frac{\ln S - \ln K + (r - \delta - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \tag{1.13}$$

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}y^2} dy. \tag{1.14}$$

Figure 1.1 presents the contract function or *payoff* for a European call option and the solution (1.11) is also presented for three different times to maturity.

1.4 Contract functions

We will now focus on the contract functions, Φ. They determine the type of the option and thus also its price. We will distinguish between single-asset and multi-asset options. Standard single-asset options like calls and puts are traded on stock exchanges as the AEX in Amsterdam, Xetra DAX in Frankfurt or CAC40 in Paris. Other options are often referred as exotic options. The multi-asset option itself is also an exotic option contract. These options are typically traded between banks, brokers and some industrial customers. We will assume that all options considered in this thesis are simple contingent claims in terms of Definition 1.3.5.

1.4.1 Single-asset contracts

Definition 1.3.4 is already a fine definition of a standard European call. The definition of a put option is given in Definition 1.4.1.

Definition 1.4.1. A European put option with strike price K and exercise date T on one underlying asset is a contract defined as follows:

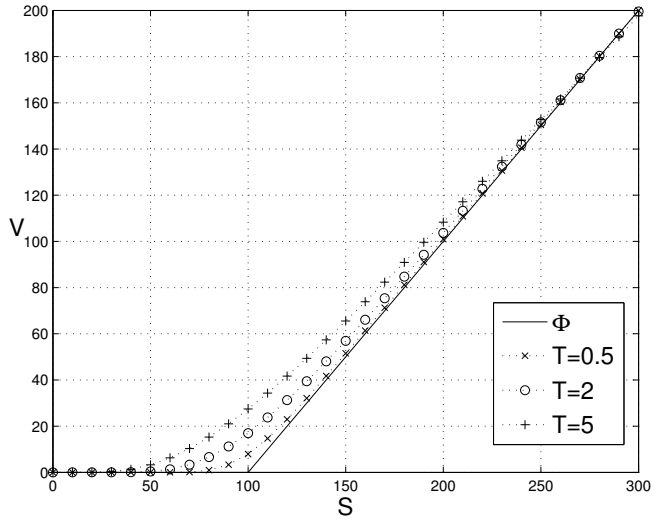


Figure 1.1: Contract function and exact solution for today's price ($t = 0$) of a European call option for three different exercise dates. $K = 100, r = 0.06, \delta = 0.02, \sigma = 0.25$

- The holder of the option has the right to sell the underlying stock S for the exercise price K at the exercise date T to the writer of the option.
- The holder has the right to sell the underlying asset at maturity date T only, but is not obliged to sell the underlying stock.

The contract function of a put option reads:

$$\Phi(T, S(T)) = \max\{K - S(T), 0\}. \tag{1.15}$$

The contract function for a put and the prices of the put option at different maturity times are presented in Figure 1.2.

Put options are often bought when a speculator expects that the stock price will go down. He will make a profit when he buys the option price for $S(T)$ and sells it for K. Even if the holder of the option does not possess any stock, he is allowed to buy this option and exercise it eventually. This is called *short selling*. An analogous analytic solution for the European put option is also available:

$$V(t, S) = Ke^{-r(T-t)}N(-d_2) - Se^{-\delta(T-t)}N(-d_1), \tag{1.16}$$

with d_1, d_2 and $N(\cdot)$ given in equations (1.12)-(1.14). Combining (1.11) and (1.16) gives the *put-call parity* for European options:

$$V_p(t, S) = Ke^{-r(T-t)} + V_c(t, S) - Se^{-\delta(T-t)}. \tag{1.17}$$

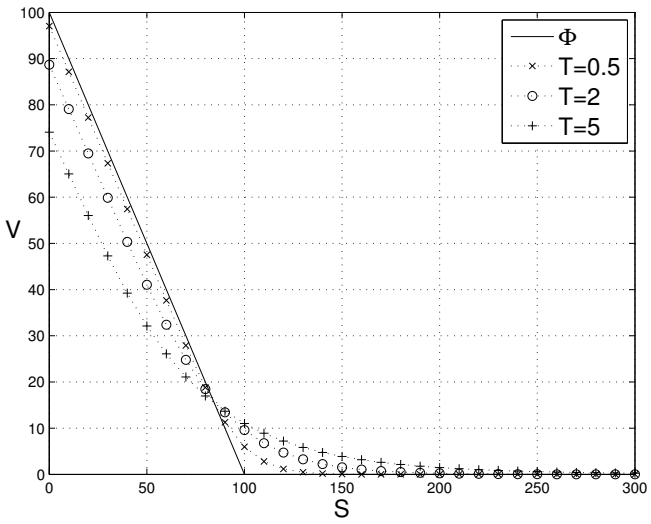


Figure 1.2: Contract function and analytic solution for today's price of a European put option for three different exercise dates. $K = 100, r = 0.06, \delta = 0.02, \sigma = 0.25$

where the subscript represents the type of the contract.

We can observe that the contract functions for both call option (1.6) and (1.15) are non differentiable at $S = K$. This is a common property of contract functions of options. However, contract functions with a discontinuity, called *digital options*, also exist. We have:

Definition 1.4.2. A digital option with strike price, K, and exercise date, T on one underlying asset is a contract defined by:

- The holder of the option has the right to exercise at the exercise date T only.
- If the holder decides to exercise, he receives either a fixed amount V_0, if the contract type is a **cash-or-nothing** option or the asset, S, if the contract is a **asset-or-nothing**.

For both types of digital options, put and call versions exist. An analytic solution exists [29, 18, 57] for these four options. Table 1.1 presents the analytic solutions together with the contract functions Φ. The contract function and the solutions for three different maturity times of a cash-or-nothing call option are plotted in Figure 1.3.

Other examples of single-asset options are:

- Barrier options [18]: There are several types, but the main property is that the option becomes worthless if the asset price passes a barrier at B or it stays worthless if it does not pass the barrier.

Table 1.1: Contract functions and analytic solutions of four different types of digital options. d_1, d_2 and $N(\cdot)$ as in equations (1.12)-(1.14).

Cash-or-nothing	Call	Put
$\Phi(T, S(T))$	$\begin{cases} V_0 & S(T) > K \\ 0 & S(T) < K \end{cases}$	$\begin{cases} 0 & S(T) > K \\ V_0 & S(T) < K \end{cases}$
$V(t, S)$	$V_0 e^{-r(T-t)} N(d_2)$	$V_0 e^{-r(T-t)} N(-d_2)$
Asset-or-nothing	**Call**	**Put**
$\Phi(T, S(T))$	$\begin{cases} S(T) & S(T) > K \\ 0 & S(T) < K \end{cases}$	$\begin{cases} 0 & S(T) > K \\ S(T) & S(T) < K \end{cases}$
$V(t, S)$	$S e^{-\delta(T-t)} N(d_1)$	$S e^{-\delta(T-t)} N(-d_1)$

- Asian options [57]: The option depends on functions of the average asset price during the lifetime of the option.
- Compound option [52]: An option with an option as underlying.
- Spread options [29]. A linear combination of puts and calls or digitals [37].

1.4.2 Multi-asset contracts

Efficient pricing of options that are dependent on more than one asset is the core of this thesis. The holder of a multi-asset contract has the right to buy a *set* of assets if the conditions are profitable. Such a set is often described as a *basket* of assets and can even be a whole index. This class of basket options can be described by a general equation for the contract function:

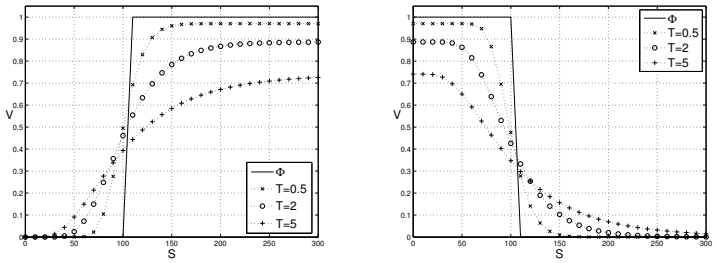
$$\Phi(T, \mathbf{S}(T)) = \max\left(\sum_{i=1}^{d} w_i S_i(T) - K, 0\right), \tag{1.18}$$

where K is the exercise price of the complete basket and w_i the percentages in the set of assets. The contract function of a basket put option is similar:

$$\Phi(T, \mathbf{S}(T)) = \max\left(K - \sum_{i=1}^{d} w_i S_i, 0\right) \tag{1.19}$$

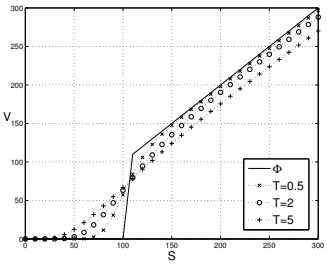
These options are often traded because a basket option is cheaper than the total of single-asset options on each particular asset [15]. The contract functions of these options are plotted in Figure 1.4.

Other options similar to the basket options are:

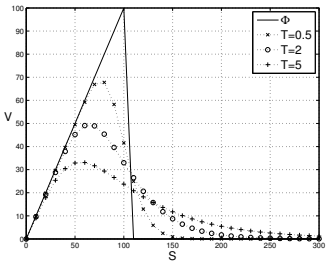


(a) Cash-or-nothing call

(b) Cash-or-nothing put



(c) Asset-or-nothing call



(d) Asset-or-nothing put

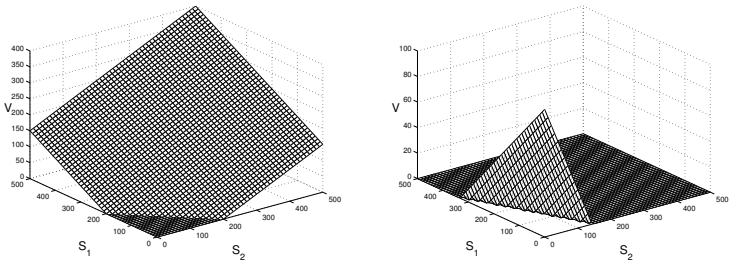
Figure 1.3: Contract function and analytic solution for today's price of four different European digital options for three different exercise dates. $K = 100, r = 0.06, \delta = 0.02, \sigma = 0.25, Q = 1$

- Index options: The basket is now replaced by complete stock index and every asset is an underlying. These options are often treated as a single-asset option.
- Exchange options: The basket of this option contains two assets, with $w_1 = -w_2$ and $K = 0$. In other words, this is an option which allows the holder to exchange one asset for another.
- Cross-currency options: Options on one asset, but priced in different currencies. With this option, the exchange rates are important.

Option on the geometric average of the assets

A multi-asset option contract based on the geometric average of the assets has the nice property [4], that after a transformation, this contract can be valuated with the one-dimensional equations (1.11) or (1.16). The geometric average of the assets is defined as:

$$\widehat{S} = \prod_{j=1}^{d} S_j^{\frac{1}{d}}. \tag{1.20}$$



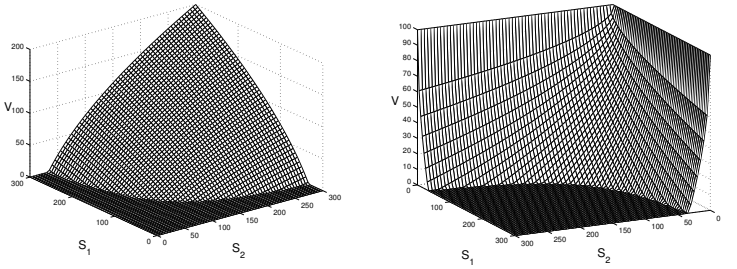
(a) Basket call, $w_1 = w_2 = 0.5$ (b) Basket put, $w_1 = 0.3, w_2 = 0.7$

Figure 1.4: Contract function of the basket option with $K = 100$

If coordinate $\widehat{S}$ is used in equation (1.9) then with

$$\widehat{\sigma} = \sqrt{\frac{1}{d^2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j}, \quad \widehat{\delta} = \frac{1}{d}\sum_{i=1}^{d}\left(\delta_i + \frac{1}{2}\sigma_i^2\right) - \frac{1}{2}\widehat{\sigma}^2.$$

the single-asset pricing equation (1.10) can be used to compute the price of the option on the geometric average. The contract functions of a call and a put on the geometric average are presented in Figure 1.5.



(a) Geometric call (b) Geometric put

Figure 1.5: Contract function of an option based on the geometric average with $K = 100$.

Option on the maximum or minimum of assets

A type of multi-asset option, which can not be described by equation (1.18) is the option on the maximum or minimum of assets. This option finds its application in a wide variety of contingent claims. One example is the option bond where payment at expiry can be chosen by the holder in a currency from a list of currencies if all possibilities are available [47]. The pay-off of

a call (put) on the maximum of d risky assets reads:

$$\text{Max call} \quad \Phi(T, \mathbf{S}(T)) = \max\{\max\{S_1, S_2, \ldots, S_d\} - K, 0\}, \tag{1.21}$$

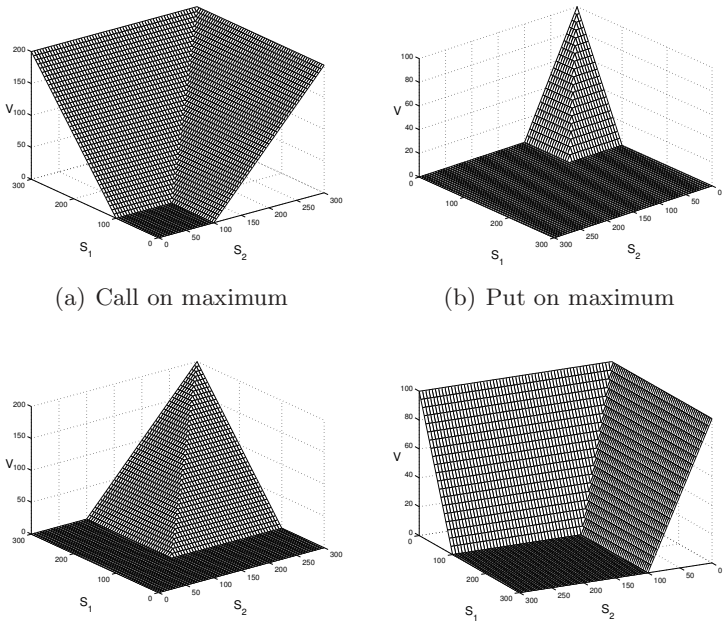
$$\text{Max put} \quad \Phi(T, \mathbf{S}(T)) = \max\{K - \max\{S_1, S_2, \ldots, S_d\}, 0\} \tag{1.22}$$

and of a call (put) on the minimum of d risky assets reads:

$$\text{Min call} \quad \Phi(T, \mathbf{S}(T)) = \max\{\min\{S_1, S_2, \ldots, S_d\} - K, 0\}, \tag{1.23}$$

$$\text{Min put} \quad \Phi(T, \mathbf{S}(T)) = \max\{K - \min\{S_1, S_2, \ldots, S_d\}, 0\}. \tag{1.24}$$

In Figure 1.6 the contract functions of these options are presented.



(a) Call on maximum

(b) Put on maximum

(c) Call on minimum

(d) Put on minimum

Figure 1.6: Contract functions for 2D options on the maximum or minimum of assets.

1.5 Risk-free portfolio and Greeks

The derivation of the famous Black-Scholes equation (1.9) or (1.10) can also be done via an alternative route. This derivation is used in many references like [52, 31, 57]. In this setting, it is straightforward to determine the hedge parameters for the portfolio.

Consider again the risk-neutral Black-Scholes price process presented in Definition 1.3.3:

$$dS_i = (r - \delta_i)S_i dt + \sigma_i S_i dW_i \tag{1.25}$$

and the value of a certain derivative $V(t, S_1, S_2, \ldots, S_d)$ dependent on the time and the underlying assets S_i. Then Itô's lemma, which can be seen as the stochastic counterpart of Taylor's lemma, reads:

Theorem 1.5.1 (Itô)**.** *Given a vector of Wiener processes* $\mathbf{W} = (W_1, W_2, \ldots, W_d)$ *and correlation coefficients* ρ_{ij} *as given and let furthermore the function* $V(t, S_1.S_2, \ldots, S_d)$ *be a stochastic process (1.5) w.r.t. to the risk-neutral probability measure, then the differential of the function* V *can be written as:*

$$\begin{aligned} dV = &\left\{ \frac{\partial V}{\partial t} + \sum_{i=1}^{d} (r - \delta_i) S_i \frac{\partial V}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \rho_{ij} \sigma_i \sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} \right\} dt \\ &+ \sum_{i=1}^{d} \sigma_i S_i \frac{\partial V}{\partial S_i} dW_i. \end{aligned} \tag{1.26}$$

Proofs of this famous theorem can be found in [7, 35].

Now, we set-up a *portfolio*, Π, containing one derivative (for example a basket call option) and $-\Delta_i$ underlying assets S_i. The value of this portfolio reads:

$$\Pi = V - \sum_{i=1}^{d} \Delta_i S_i.$$

Basically the portfolio Π follows the same process as of the asset price S and the $Q-$dynamics of the derivative V. The change in the value of the portfolio is affected by the payment of the dividend yield. A continuous dividend yield gives an amount $\delta_i S_i dt$ per time unit and since we have $-\Delta_i$ assets S_i in our portfolio, we have [52]:

$$d\Pi = dV - \sum_{i=1}^{d} \Delta_i dS_i - \sum_{i=1}^{d} \Delta_i \delta_i S_i dt. \tag{1.27}$$

In other words, the value of the portfolio increases with $\sum_{i=1}^{d} \Delta_i \delta_i S_i dt$ during its lifetime. We assume that Δ_i is a fixed number within the time-interval dt according to the Black-Scholes theory [8]. Now we substitute the differential (1.26) into $d\Pi$ (1.27):

$$\begin{aligned} d\Pi &= dV - \sum_{i=1}^{d} \Delta_i dS_i - \sum_{i=1}^{d} \Delta_i \delta_i S_i dt \\ &= \left\{ \frac{\partial V}{\partial t} + \sum_{i=1}^{d} (r - \delta_i) S_i \frac{\partial V}{\partial S_i} + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \rho_{ij} \sigma_i \sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} \right\} dt \\ &+ \sum_{i=1}^{d} \sigma_i S_i \frac{\partial V}{\partial S_i} dW_i - \sum_{i=1}^{d} \Delta_i dS_i - \sum_{i=1}^{d} \Delta_i \delta_i S_i dt, \end{aligned}$$

and substitute dS_i from equation (1.25):

$$d\Pi = \left\{ \frac{\partial V}{\partial t} + \sum_{i=1}^{d}(r-\delta_i)S_i\frac{\partial V}{\partial S_i} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} \right\} dt$$
$$+ \sum_{i=1}^{d}\sigma_i S_i \frac{\partial V}{\partial S_i} dW_i - \sum_{i=1}^{d}\Delta_i \left[(r-\delta_i)\, S_i dt + \sigma_i S_i dW_i\right] - \sum_{i=1}^{d}\Delta_i \delta_i S_i dt.$$

To eliminate the risk from $d\Pi$, we have to eliminate the term with dW_i, so we choose $\Delta_i = \dfrac{\partial V}{\partial S_i}$. Then we have by rearranging terms:

$$d\Pi = \left\{ \frac{\partial V}{\partial t} - \sum_{i=1}^{d}\delta_i S_i \frac{\partial V}{\partial S_i} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} \right\} dt \qquad (1.28)$$

From an economical point of view [52], the return on the portfolio invested in risk-free assets would see a increase of $r\Pi dt$ in a time period dt. If the right-hand side of (1.28) is larger than this amount, a trader can make a guaranteed risk-free profit by borrowing an amount Π and invest it in the portfolio. This is again an example of *arbitrage*. Conversely, if the right-hand side is smaller the trader will go short on the portfolio and invest Π in a risk-free bank account. Therefore, we need to have $d\Pi = r\Pi dt$. Combining the growth of the portfolio during dt with the portfolio change (1.28), we find:

$$r\Pi dt = \left\{ \frac{\partial V}{\partial t} - \sum_{i=1}^{d}\delta_i S_i \frac{\partial V}{\partial S_i} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} \right\} dt. \qquad (1.29)$$

Substituting the value of the portfolio $\Pi = V - \sum_{i=1}^{d}\Delta_i S_i$ and the value of Δ_i in (1.29), we again obtain the Black-Scholes equation (1.9) after division by dt.

The Δ_i in the derivation of the Black-Scholes equation is known as a *hedge parameter* and it is one of the *Greeks*. This parameter gives thus the amount of assets to purchase in combination with an option to eliminate the risk in the portfolio. The analytic solution value of Δ for a single-asset European call and put option by use of equation (1.11) or (1.16) reads:

$$\Delta_c(t,S) = e^{-\delta(T-t)}N(d_1) \qquad (1.30)$$
$$\Delta_p(t,S) = e^{-\delta(T-t)}\left(N(d_1)-1\right) \qquad (1.31)$$

In Figure 1.7, the Δ of a European call for different maturity times is presented.

Another hedge parameter which is important in finance is the Gamma, Γ. This parameter is a measure to adjust the amount Δ to maintain a

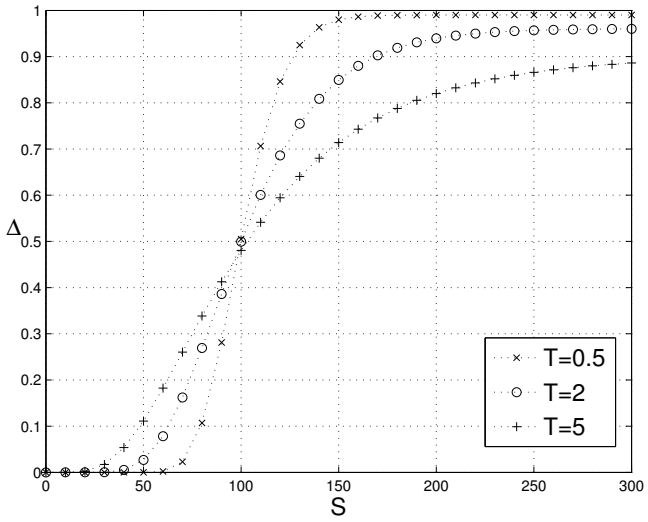


Figure 1.7: Analytic solution of a Δ of a European call option for three different maturity times. $K = 100, r = 0.06, \delta = 0.02, \sigma = 0.25$

risk-neutral portfolio. The version in the multi-asset case $\Gamma_{i,j}$ is the second derivative of the option price:

$$\Gamma_{i,j} = \frac{\partial^2 V}{\partial S_i \partial S_j}. \tag{1.32}$$

It easily follows that the Γ for single-asset put and call options is equal:

$$\Gamma_{p,c}(t,S) = \frac{e^{-\frac{1}{2}d_1^2}}{\sigma S\sqrt{2\pi(T-t)}}. \tag{1.33}$$

In Figure 1.8, the gamma, Γ, for a European option with different maturity times is presented.

1.6 Early exercise

In contrast to European options, which can only be exercised at the maturity date T, American options can be exercised at any time prior to T. Consequently, identifying the optimal exercise strategy is an integral part of the valuation problem.

Let $V(t,S)$ be the value of an American option with contract function $\Phi(T, S(T)))$ at exercise. The possibility of early exercise requires

$$V(t,S) \geq \Phi(S), \quad \forall t \in [0,T] \text{ and } S \in [0,\infty),$$

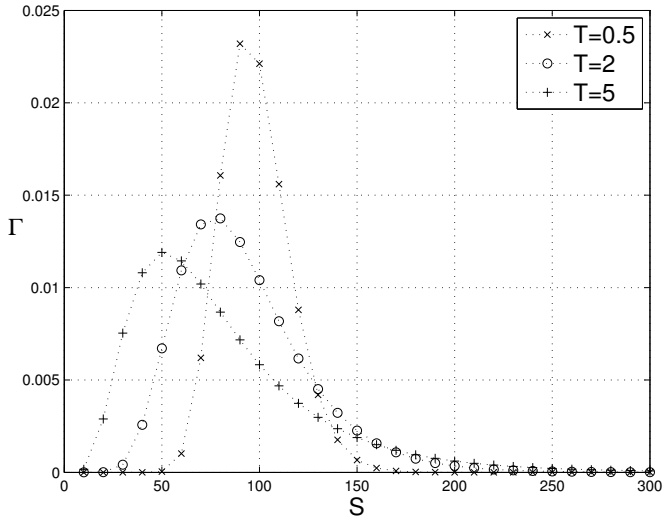


Figure 1.8: Analytic solution of a Γ of a European call option for three different maturity times. $K = 100, r = 0.06, \delta = 0.02, \sigma = 0.25$

as otherwise a holder would immediately exercise this option [29, 52, 51], and an arbitrage opportunity would exist.

To illustrate this, two different portfolios are constructed. The first portfolio contains one European call option on a non-dividend paying stock plus an amount of money Ke^{-rT}. The second portfolio is only one asset S_0. In the first portfolio, the cash, if it is invested at the risk-free interest rate, would grow to K in time T. If $S(T) > K$, the option is exercised at maturity and therefore the portfolio is worth $S(T)$. If $S(T) < K$, the option is worthless and the value of the portfolio is K. Hence at maturity time, the value of the portfolio is $\max\{S_T, K\}$.

The value of the second portfolio is worth S_T at time T. We therefore see that the value of the first portfolio is always larger than or equal to the value of the second portfolio. In the absence of arbitrage opportunities, this must be the same during the lifetime of the option:

$$V(t, S(t)) + Ke^{-r(T-t)} \geqslant S(t) \Leftrightarrow V(t, S(t)) \geqslant S(t) - Ke^{-r(T-t)}. \quad (1.34)$$

If the option is an American option, it is allowed to exercise the option during its lifetime. To exercise the option during its lifetime the value of the American call option is at least equal to the contract function:

$$V(t, S(t)) \geqslant S(t) - K.$$

Using the inequality from (1.34), we have:

$$V(t, S(t)) \geqslant S(t) - Ke^{-r(T-t)} \geqslant S(t) - K$$

and we deduce that in the absence of dividends it is never optimal to exercise early.

However, for a put option the solution is different. Again, we illustrate this with two different portfolios. The first portfolio is a combination of a European put on a non-dividend paying stock and one asset. The second portfolio is an amount of cash equal to Ke^{-rT}. At maturity time the value of the first portfolio equals K if $S_T < K$, because the option will be exercised. The value equals S_T if $S_T > K$, so again, the value of the first portfolio at maturity time is $\max\{S(T), K\}$. The second portfolio would grow to an amount K if it was invested at the risk-free interest rate. We deduce that the value of the first portfolio is larger than or equal to the value of the second portfolio and by the same arbitrage arguments, this holds as well during the lifetime of the option. So we have:

$$V(t, S(t)) + S(t) \geqslant Ke^{-r(T-t)} \Leftrightarrow V(t, S(t)) \geqslant Ke^{-r(T-t)} - S(t). \quad (1.35)$$

If it is an American option, it is possible to exercise early and we have:

$$V(t, S(t)) \geqslant K - S(t). \quad (1.36)$$

From (1.35), we see that the price of the European option can be lower than the contract function and consequently the price of an American option is larger than a European option.

Mathematically, the valuation of the American option is similar to what is known as solving a free boundary problem. The free boundary stock price $S_f(t)$, also called early exercise boundary, divides the (t, S) half strip into two parts, namely the continuation region and the stopping region. The continuation region $\{(t, S) \in [0, T] \times \mathbb{R}_+ : V(t, S) > \Phi(T, S(T))\}$ is the set of points (t, S) at which the option is worth more alive, whereas in the stopping region $\{(t, S) \in [0, T] \times \mathbb{R}_+ : V(t, S) = \Phi(T, S(T))\}$ early exercise is advisable as the option is worth its contract function.

Under the Black-Scholes framework, the price $V(t, S)$, satisfies, in the continuation region

$$V(t, S) > \Phi(T, S(T)), \quad \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta)S\frac{\partial V}{\partial S} - rV = 0;$$

or in the stopping region

$$V(t, S) = \Phi(S), \quad \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \delta)S\frac{\partial V}{\partial S} - rV < 0.$$

Additionally, the continuity of V and $\partial V/\partial S$ form the boundary conditions at $S_f(t)$:

$$V(t, S_f(t)) = \Phi(t, S_f(t)), \quad \frac{\partial V(t, S_f(t))}{\partial S} = \Phi'(S_f(t)).$$

It is known as the *smooth fit principle*.

This free boundary problem can be reformulated into a *linear complementarity problem* [44] which reads as follows:

$$\begin{aligned} V(t,S) &\geq \Phi(T,S(T)), \\ -\Big(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-\delta)S\frac{\partial V}{\partial S} - rV\Big) &\geq 0, \\ \Big(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-\delta)S\frac{\partial V}{\partial S} - rV\Big)\big(V(t,S) - \Phi(T,S(T))\big) &= 0 \end{aligned}$$

with final and boundary conditions. The optimal exercise boundary, $S_f(t)$, is automatically captured by this formulation and can be determined a-posteriori. It is the set of points for which we have equality sign for the inequalities in this problem.

A second type of early exercise option, is the so-called *Bermudan* option. Having this option, a holder is allowed to exercise the option prior to the exercise date T, *at certain prescribed dates*, the *exercise moments*. Hence if at these moments the condition

$$V(t,S(t)) > \Phi(T,S(t))$$

holds, the option should not be exercised. Otherwise, if the price of the option equals the contract function, a holder obviously exercises his option. At exercise moments, the option value is equal to:

$$V(t_E,S(t_E)) = \max\{\Phi(t_E,S(t_E)), V(t_E,S(t_E))\}. \tag{1.37}$$

In Figure 1.9 the solutions of a European put and an Bermudan put are compared.

1.7 Dividends

In equation (1.10), the parameter δ represents a continuous dividend yield. However, on most stocks, dividend is paid at discrete moments, for example once or twice a year. This is called a *discrete dividend*, D.

As an illustration of a dividend payment, we consider two portfolios. The first portfolio contains a European call option on the dividend paying stock and an amount of cash equal to $D + Ke^{-rT}$. The second portfolio contains one asset. By similar arguments as in the previous section, the value of the first portfolio is equal to $S(T) + D$ if $S(t) > K$. Otherwise the value is equal to $K + D$, because the option is not exercised and the amount D is not invested at the risk-free rate. The value of the second portfolio is $S(T) + D$, because during the lifetime of the option, the amount D is paid. We see that at maturity time, the value of the first portfolio is larger than

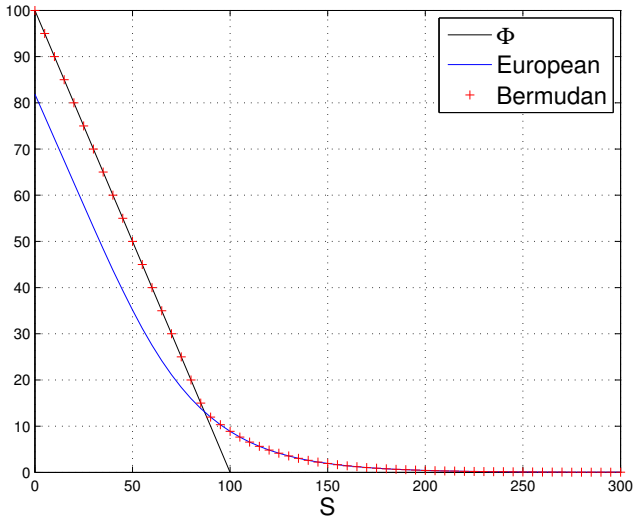


Figure 1.9: Contract function, European and Bermudan put price with $K = 100, r = 0.1, \delta = 0.02, \sigma = 0.3, T = 2$

the value of the second portfolio. Under the absence of arbitrage, this holds for the period before the dividend payment too, so we have:

$$V(t, S(t)) + D + Ke^{-r(T-t)} \geqslant S(t) \Leftrightarrow V(t, S(t)) \geqslant S(t) - D - Ke^{-r(T-t)}.$$

Considering early exercise when $V(t, S(t)) \geqslant \Phi(t, S(t)) = \max\{S(t) - K, 0\}$, we see that early exercise is optimal if

$$D \geqslant K(1 - e^{-rT}). \tag{1.38}$$

We see that for some values of D (and also for the continuous dividend yield), exercising a call option could be optimal. In an analogous way, we can derive that it is optimal to exercise early in the case of an American put option if[31, 2]:

$$D \leqslant K(1 - e^{-rT}). \tag{1.39}$$

Furthermore Meyer [34] derived that a free boundary as defined in Section 1.6 can disappear during a period:

$$\delta t = \frac{1}{r} \ln(1 + \frac{D}{K}). \tag{1.40}$$

We see that the dividend payment has an influence on both the asset price as well as on the portfolio. However, the price of an option does not

change over the ex-dividend date, t_d. We adopt the technique of modelling discrete dividend by a *jump condition* at the ex-dividend date t_d [52]:

$$V(t_d^-, S_{t_d^-}) = V(t_d^+, S_{t_d^+}). \tag{1.41}$$

where t_d^-, t_d^+ represent the times just before and after the ex-dividend date, respectively. In fact, we solve the Black-Scholes partial differential equation in two parts. First, the part from T to t_d^+ and then from t_d^- to $t = 0$. This will be explained in more detail in the section describing the numerical procedures.

Chapter 2

Single-asset option pricing with the PDE method

2.1 Introduction

This chapter covers the numerical solution of the single-asset option pricing problem by solving the one-dimensional partial differential equation (1.10). Efficient solution of the one-dimensional option pricing problem can be seen as a basic requirement for solving the multi-dimensional problem. Since the domain of the partial differential equation is the whole positive real axis, some problems may occur when using numerical techniques. In Section 2.2, the boundary conditions for the one-dimensional option pricing problem are discussed as well as the truncation of the domain of computation. Section 2.3 contains the discussion of the numerical solution technique. The derived difference equations are generalised to the multi-dimensional case in Chapter 3. In Section 2.4 grid stretching is discussed. Grid stretching is a method to improve the accuracy of the numerical solution in the region of interest. Furthermore, the numerical solution of the hedge parameters in combination with the grid stretching is presented. Section 2.5 shows some numerical experiments. We present a plain vanilla call option comparing the different discretisation techniques. A digital option is also considered, because it is a well-known example of contract with a discontinuity. Some options on dividend paying stocks are discussed as well as early exercise examples. We draw our conclusions in Section 2.6.

2.2 Boundary conditions

We recall equation (1.10) for the single-asset problem:

$$\begin{cases} \dfrac{\partial V}{\partial t} + (r-\delta)S\dfrac{\partial V}{\partial S} + \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 V}{\partial S^2} - rV = 0 \\ V(T,S) = \Phi(T,S(T)) \end{cases} \tag{2.1}$$

This equation is a parabolic partial differential equation of the anti-diffusion type. In the introduction, we have already discussed the final conditions, that determine the type of the option. Since parabolic differential equations are typically *stiff* [27], the numerical solutions can be naturally obtained by implicit time integration. In that case boundary conditions are mandatory to have a well-posed problem.

The solution is computed on the domain $[0, \infty) \times [0, T]$. For numerical treatment, the domain has to be truncated to some maximum value S_{max}. In the work by Kangro and Nicolaides [30], the error is estimated when S is truncated at a certain value for S_{max}. By rigorous analysis, they state that if S_{max} is chosen as:

$$S_{max} = Ke^{\sqrt{2\sigma T \log 100}} \tag{2.2}$$

the error is of prescribed size (i.e. typically less than 0.01). For a (European) put option, we have $P(t, S_{max}) \approx 0$. Using the put-call parity (1.17) from Section 1.4.1, the boundary condition for a call option, if $S = S_{max}$, reads:

$$V(t, S_{max}) = S_{max}e^{-\delta(T-t)} - Ke^{-r(T-t)}, \tag{2.3}$$

This type of boundary condition is called an inhomogeneous *Dirichlet condition*. We see that the value of the option shows a linear behaviour if S increases towards S_{max}. Another type of boundary condition is the *linearity condition*. Such a condition on a general boundary $\partial\Omega$ reads:

$$\frac{\partial^2 V}{\partial S^2} = 0 \quad S \in \partial\Omega. \tag{2.4}$$

This type of boundary conditions for the second order partial differential equation does not guarantee a well-posed problem. It may cause inaccuracies by numerical solution. However, Tavella [48] and Forsyth [53] show that option problems with a contract function that is linear at the boundaries, can be treated with sufficient accuracy. This kind of boundary condition is used in the multi-dimensional case as well when an exact boundary solution or Dirichlet boundary conditions are not known. The linearity condition holds for both put and call options.

If $S = 0$ in equation (2.1), then the Black-Scholes equation reduces to an ordinary differential equation:

$$\begin{cases} \dfrac{dV}{dt} - rV & = 0 \\ V(0) & = \Phi(0, 0). \end{cases} \tag{2.5}$$

Equation (2.5) can serve as the boundary condition for $S = 0$ in equation (2.1). This boundary condition comes directly from the problem itself and therefore is is known as a *natural boundary conditions*. If these boundary conditions are available (even for multi-asset options), then these boundary

conditions should be preferred to linearity conditions, because the implementation is straightforward. From the contract function of a call option, we have $\Phi(T,0) = 0$ and so the solution of equation (2.5) reads $V(t) = 0$ for a call option. For a put option $\Phi(T,0) = K$ and the solution of equation (2.5) reads $V(t) = Ke^{-r(T-t)}$. The Black-Scholes partial differential equation for a European call option defined on a truncated domain with proper initial and boundary conditions now reads:

$$\begin{cases} \dfrac{\partial V}{\partial \tau} = (r-\delta)S\dfrac{\partial V}{\partial S} + \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 V}{\partial S^2} - rV \\ V(0,S) = \max(S-K,0) \\ V(\tau,0) = 0 \\ V(\tau,S_{max}) = S_{max}e^{-\delta\tau} - Ke^{-r\tau}, \end{cases} \tag{2.6}$$

where we already transformed the final condition into an initial condition by replacing $T-t$ by τ. Henceforth we use t as the time coordinate, but we assume that all partial differential equations are transformed to well-posed problems with initial conditions.

2.3 Numerical solution

2.3.1 Discretisation

Equation (2.6) can be generalised to a more standard form:

$$\begin{cases} \dfrac{\partial V}{\partial t} & = f(S)\dfrac{\partial^2 V}{\partial S^2} + g(S)\dfrac{\partial V}{\partial S} - rV, \\ V(0,S) & = \Phi(T,S(T)), \\ V(t,S_{min}) & = L(t) \quad \text{or} \quad \dfrac{\partial^2 V}{\partial S^2}|_{S=S_{min}} = 0, \\ V(t,S_{max}) & = R(t) \quad \text{or} \quad \dfrac{\partial^2 V}{\partial S^2}|_{S=S_{max}} = 0. \end{cases} \tag{2.7}$$

The functions $L(t), R(t)$ and $\Phi(t,S)$ determine the type of contract and the value S_{min} is typically zero. The discretisation of the partial differential equation is independent of the chosen boundary conditions. The solution, $V(t,S)$, is discretised first at an equidistant $(t,S)-$mesh. The value of the solution at a certain point, (t_ν, S_i), will be abbreviated by V_i^ν. The distance between two grid points, i.e. the values S_i and S_{i+1} equals h and the time step, Δt, is the difference between t_ν and $t_{\nu+1}$. The number of grid points is equal to N and the number of time steps is M, so:

$$h = \frac{S_{max} - S_{min}}{N}, \tag{2.8}$$

$$\Delta t = \frac{T}{M}. \tag{2.9}$$

The partial differential equation is discretised according to the finite difference method with for example the so-called $\theta-$scheme:[48]

$$\begin{aligned}\frac{V_i^{\nu+1}-V_i^{\nu}}{\Delta t} = (1-\theta)\left(f_i\frac{V_{i+1}^{\nu}-2V_i^{\nu}+V_{i-1}^{\nu}}{h^2}+g_i\frac{V_{i+1}^{\nu}-V_{i-1}^{\nu}}{2h}-rV_i^{\nu}\right)+\\ +\theta\left(f_i\frac{V_{i+1}^{\nu+1}-2V_i^{\nu+1}+V_{i-1}^{\nu+1}}{h^2}+g_i\frac{V_{i+1}^{\nu+1}-V_{i-1}^{\nu+1}}{2h}-rV_i^{\nu+1}\right).\end{aligned} \tag{2.10}$$

If $\theta = 1$,the scheme is the so-called BDF1 (backward differentiation formula) or backward Euler scheme. The error of this discretisation follows from the Taylor expansion and reads $\mathbf{O}(h^2+\Delta t)$. The value $\theta = 0$ leads to the explicit Euler scheme which is not used in this thesis. The value of θ that may lead to a higher accuracy is $\theta = \frac{1}{2}$, the Crank-Nicolson method with accuracy $\mathbf{O}(h^2+\Delta t^2)$, so second order in asset price and time.

If the linearity boundary conditions are used, then a one-sided difference scheme is used for the $S-$coordinate at the boundary. For example if $S = S_N = S_{max}$, the scheme reads:

$$\begin{aligned}\frac{V_N^{\nu+1}-V_N^{\nu}}{\Delta t} = (1-\theta)\left(g_i\frac{3V_N^{\nu}-4V_{N-1}^{\nu}+V_{N-2}^{\nu}}{2h}-rV_N^{\nu}\right)+\\ +\theta\left(g_i\frac{3V_N^{\nu+1}-4V_{N-1}^{\nu+1}+V_{N-2}^{\nu+1}}{2h}-rV_N^{\nu+1}\right),\end{aligned} \tag{2.11}$$

and this scheme also has an accuracy of $\mathbf{O}(h^2+\Delta t^2)$, if $\theta = \frac{1}{2}$.

The initial conditions (i.e. contract functions of the options) are non-differentiable in general. Hence, some difficulties with the Crank Nicolson method may occur [37], since this time discretisation is not L-stable [26]. Therefore, the so-called Rannacher time-marching can be applied [39, 22] which means that the first step is preformed with $\theta = 1$, followed by $M-1$ steps of the Crank-Nicolson scheme ($\theta = \frac{1}{2}$). Instead of the $M-1$ Crank-Nicolson steps, the time-integration can also be done with the BDF2 scheme:

$$\begin{aligned}\frac{\frac{3}{2}V_i^{\nu+1}-2V_i^{\nu}+\frac{1}{2}V_i^{\nu-1}}{\Delta t} = f_i\frac{V_{i+1}^{\nu+1}-2V_i^{\nu+1}+V_{i-1}^{\nu+1}}{h^2}+\\ +g_i\frac{V_{i+1}^{\nu+1}-V_{i-1}^{\nu+1}}{2h}-rV_i^{\nu+1}\end{aligned} \tag{2.12}$$

The accuracy is again $\mathbf{O}(h^2+\Delta t^2)$. This method requires two previous solution (V_i^{ν} and $V_i^{\nu-1}$), and therefore if $\nu = 1$, the BDF 1 scheme (equation (2.10) with $\theta = 1$) is used.

Higher order accuracy can be obtained by using higher order difference methods. These methods use more grid points to discretise a derivative. A higher order time-integration method employs more previous solutions to

obtain a higher order time accuracy. The combined schemes for an accuracy of $\mathbf{O}(h^4 + \Delta t^4)$ based on a BDF4 scheme and long-stencil central differences in asset space reads:

$$\begin{aligned}\frac{1}{\Delta t}\left(\frac{25}{12}V_i^{\nu+1} - 4V_i^{\nu} + 3V_i^{\nu-1} - \frac{4}{3}V_i^{\nu-2} + \frac{1}{4}V_i^{\nu-3}\right) = \\ f_i\frac{-V_{i+2}^{\nu+1} + 16V_{i+1}^{\nu+1} - 30V_i^{\nu+1} + 16V_{i-1}^{\nu+1} - V_{i-2}^{\nu+1}}{12h^2} + \\ g_i\frac{-V_{i+2}^{\nu+1} + 8V_{i+1}^{\nu+1} - 8V_{i-1}^{\nu+1} + V_{i-2}^{\nu+1}}{12h} - rV_i^{\nu+1}.\end{aligned} \tag{2.13}$$

For the computation at point S_i, five adjacent spatial points are required and for the time point $\nu + 1$, four preceding steps are needed. The first two steps can be done by the use of BDF1 and BDF2 steps. A third order time-integration, BDF3, is the used between steps $\nu = 2$ and $\nu = 4$.

Finally, for the discretisation in the points S_1 and S_{N-1}, a one-sided difference scheme is used, but now for both the first and second derivatives. For example, the $\mathbf{O}(h^4 + \Delta t^4)$ scheme in the point S_{N-1} reads:

$$\begin{aligned}\frac{1}{\Delta t}\left(\frac{25}{12}V_i^{\nu+1} - 4V_i^{\nu} + 3V_i^{\nu-1} - \frac{4}{3}V_i^{\nu-2} + \frac{1}{4}V_i^{\nu-3}\right) = \\ f_{N-1}\frac{10V_N^{\nu+1} - 15V_{N-1}^{\nu+1} - 4V_{N-2}^{\nu+1} + 14V_{N-3}^{\nu+1} - 6V_{N-4}^{\nu+1} + V_{N-5}^{\nu+1}}{12h^2} + \\ g_{N-1}\frac{3V_N^{\nu+1} + 10V_{N-1}^{\nu+1} - 18V_{N-2}^{\nu+1} + 6V_{N-3}^{\nu+1} - V_{N-4}^{\nu+1}}{12h} - rV_i^{\nu+1}.\end{aligned} \tag{2.14}$$

2.3.2 Linear system, early exercise and dividends

The schemes (2.10), (2.12) or (2.13) form a set of linear equations. These equations can be denoted by an algebraic matrix equation of the type $\mathbf{Pv} = \mathbf{b}$, where $\mathbf{P}$ is the discretisation matrix, $\mathbf{v}$ the vector of unknowns (i.e. containing $V_i^{\nu+1}$) and vector $\mathbf{b}$ containing the right hand sides. In this latter vector, the boundary conditions appear as well which is called *non-eliminated boundary conditions*. A reason to keep the boundary conditions in the matrix lies in the generalisation of the discretisation to the multi-dimensional case. When eliminated boundary conditions are used, the use of Kronecker products is not straight-forward to discretise mixed derivatives. The boundary conditions are placed on the first and last positions of the vector $\mathbf{b}$ and therefore the first and last row of matrix $\mathbf{P}$ contain a 1 as diagonal entry.

The construction of the matrix $\mathbf{P}$ can be done by constructing difference matrices for every derivative separately and then adding these together. For

example, difference matrix $\mathbf{A}_1$ for the second derivative with:

$$(\mathbf{A}_1)_{ij} = \begin{cases} \frac{-2}{h^2} & j = i \\ \frac{1}{h^2} & j = i \pm 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

and a difference matrix $\mathbf{B}_1$ for the first derivative with

$$(\mathbf{B}_1)_{ij} = \begin{cases} \frac{1}{2h} & j = i + 1 \\ \frac{-1}{2h} & j = i - 1. \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

or we may use the long stencils as used in equations (2.13). The first and last row of these matrices may change when the boundary stencils (2.11) or (2.14) are used. If these matrices are multiplied by the coefficient functions, we obtain:

$$\mathbf{F}\mathbf{A}_1 + \mathbf{G}\mathbf{B}_1 - r\mathbf{I}$$

with $\mathbf{F}$ and $\mathbf{G}$ diagonal matrices containing $f_0, f_1, \ldots, f_N$ and $g_0, g_2, \ldots, g_N$, respectively and I is the identity matrix. Now, the construction of the θ scheme in matrix formulation reads:

$$\begin{aligned} \mathbf{I}\mathbf{v}^{\nu+1} - \mathbf{I}\mathbf{v}^{\nu} = (1-\theta)\,\Delta t\,[\mathbf{F}\mathbf{A}_1 + \mathbf{G}\mathbf{B}_1 - r\mathbf{I}]\,\mathbf{v}^{\nu} + \\ + \theta\Delta t\,[\mathbf{F}\mathbf{A}_1 + \mathbf{G}\mathbf{B}_1 - r\mathbf{I}]\,\mathbf{v}^{\nu+1}, \end{aligned} \tag{2.17}$$

and we have to solve $\mathbf{P}\mathbf{v}^{\nu+1} = \mathbf{b}$ with:

$$\begin{aligned} \mathbf{P} &= \mathbf{I} - \theta\Delta t\,[\mathbf{F}\mathbf{A}_1 + \mathbf{G}\mathbf{B}_1 - r\mathbf{I}] \\ \mathbf{b} &= \mathbf{I} + (1-\theta)\,\Delta t\,[\mathbf{F}\mathbf{A}_1 + \mathbf{G}\mathbf{B}_1 - r\mathbf{I}]\,\mathbf{v}^{\nu}. \end{aligned}$$

The construction of these matrices derivative-wise is useful for the generalisation of the discretisation to the multi-dimensional case, because they can be combined per coordinate to construct the multi-dimensional version of $\mathbf{P}$. This construction is described in detail in Section 3.2.2.

The solution of the algebraic equation $\mathbf{P}\mathbf{v}^{\nu+1} = \mathbf{b}$, with typically large vectors and matrices can be performed by a direct solution method like an LU-decomposition or Gauss elimination. For higher-dimensional cases, we use iterative solvers, like Bi-CGSTAB in combination with multigrid.

The numerical computation of American options or options with dividend is also of interest. In Section 1.6, we presented the linear complementary problem for the early exercise options. The discretised version of this problem, with the use of the discretisation matrix $\mathbf{P}$ reads:

$$\begin{aligned} \left(\mathbf{P}\mathbf{v}^{\nu+1} - \mathbf{b}\right) \cdot \left(\mathbf{v}^{\nu+1} - \mathbf{\Phi}\right) = 0 \\ \text{such that } \mathbf{P}\mathbf{v}^{\nu+1} - \mathbf{b} \geqslant \mathbf{0} \quad \text{and} \quad \mathbf{v}^{\nu+1} \geqslant \mathbf{\Phi}, \end{aligned} \tag{2.18}$$

where $\mathbf{\Phi} = [\Phi(t_\nu, S_0(t_\nu)), \Phi(t_\nu, S_0(t_\nu)), \ldots, \Phi(t_\nu, S_N(t_\nu))]^T$. This system of equations holds for every time step, ν and it can be used with both second and fourth order asset and time discretisations.

System (2.18) is then solved by the *projected successive over-relaxation method* (PSOR) [44, 14]. This method is an element-wise iterative solution method. Consider the computation of the vector $\mathbf{v}^\nu$ and a desired tolerance Tol. Let $\mathbf{x}^k$ be the $k-$th iterated solution of $\mathbf{v}^\nu$, then by processing each element, we define:

$$z_i^k = b_i - \sum_{j=0}^{i-1} p_{ij} x_j^k - \sum_{j=i}^{N} p_{ij} x_j^{k-1} \tag{2.19}$$

with p_{ij} the entries of matrix $\mathbf{P}$ and x_j^k the $j-$th position of vector $\mathbf{x}^k$. Then with equation (2.19), the new iterated solution for position j reads:

$$x_i^k = \max\{\Phi(t_\nu, S_i(t_\nu)), x_i^{k-1} + \frac{z_i^k}{p_{ii}}\}. \tag{2.20}$$

We observe that in equation (2.19) the iterated solution at step $k+1$ already occurs. The reason is the element-wise processing. For element j, the known values of $\mathbf{X}^k$ are used. The iteration stops when the solution of the original problem is lower than the desired tolerance:

$$\|\mathbf{P}\mathbf{v}^{\nu+1} - \mathbf{b}\|_\infty \leqslant Tol. \tag{2.21}$$

The computation of System (2.18) is now summarised in Algorithm 1.

Algorithm 1: PSOR

```
1  Initialise and compute P
2  for ν = 1 to  ν = M do
3  |  Compute Φ^ν and b^ν
4  |  Set x_i^0 = v_i^ν
5  |  while ‖Pv^{ν+1} − b^ν‖ ⩾ Tol do
6  |  |  for j = 1 to  j = N do
7  |  |  |  Compute z_i^k via (2.19).
8  |  |  |  Compute x_i^k via (2.20).
9  |  |  endfor
10 |  endw
11 endfor
```

2.4 Grid stretching

The accuracy of a finite difference approximation depends on the existence of several derivatives in the Taylor's expansion of a solution, but in option pricing the final condition is not differentiable (or even discontinuous in the case of a digital option). Therefore, local grid refinement seems a logical choice to obtain a satisfactory accuracy. It is well-known that local grid refinement near sharp corners in the domain or near singularities in an equation often improves the overall discretisation accuracy drastically. By an h-refinement in the vicinity of a singularity the discretisation error is locally decreased, due to the smaller h, and the global accuracy is not spoiled by the well-known *pollution effect* [49], as it is encountered for elliptic or parabolic equations. The principle of local refinement is simple: Place more points in the neighbourhood of the grid points where the non-differentiable condition occurs. This can be done by adaptive grid refinement for some regions, based on an error indicator, or by an *analytic coordinate transformation*, which results in an a-priori stretching of the grid. To avoid confusion, we will call the one-to-one one-dimensional coordinate transformation derived here *coordinate stretching* (in contrast to true coordinate transformation as is used in Sections 3.5) A coordinate stretching is an elegant way in our applications as the region of interest is known beforehand. An equidistant grid discretisation can be used after the analytic stretching, as only the coefficients in front of the derivatives change. We explain the principle for the general parabolic partial differential equation (2.7):

$$\begin{cases} \dfrac{\partial V}{\partial t} & = f(S)\dfrac{\partial^2 V}{\partial S^2} + g(S)\dfrac{\partial V}{\partial S} - rV \\ V(0,S) & = \Phi(T, S(T)) \\ V(t, S_{min}) & = L(t) \quad \text{or} \quad \dfrac{\partial^2 V}{\partial S^2}|_{S=S_{min}} = 0 \\ V(t, S_{max}) & = R(t) \quad \text{or} \quad \dfrac{\partial^2 V}{\partial S^2}|_{S=S_{max}} = 0. \end{cases}$$

Consider a coordinate stretching $y = \psi(S)$, which must be one-to-one, with inverse $S = \varphi(y) = \psi^{-1}(y)$ and let $\hat{V}(t,y) := V(t,S)$ (unknowns with "hat" exist on the stretched grid). By the chain rule, the first and second derivatives with respect to S of $V(t,S)$ become:

$$\frac{\partial V}{\partial S} = \frac{\partial \hat{V}}{\partial y}\frac{dy}{dS} = \frac{\partial \hat{V}}{\partial y}\left(\frac{dS}{dy}\right)^{-1} = \frac{1}{\varphi'(y)}\frac{\partial \hat{V}}{\partial y}, \tag{2.22}$$

$$\begin{aligned} \frac{\partial^2 V}{\partial S^2} &= \frac{\partial}{\partial y}\frac{dy}{dS}\left(\frac{1}{\varphi'(y)}\frac{\partial \hat{V}}{\partial y}\right) = \frac{1}{\varphi'(y)}\frac{\partial}{\partial y}\left(\frac{1}{\varphi'(y)}\frac{\partial \hat{V}}{\partial y}\right) \\ &= \frac{1}{\varphi'(y)^2}\frac{\partial^2 \hat{V}}{\partial y^2} - \frac{\varphi''(y)}{\varphi'(y)^3}\frac{\partial \hat{V}}{\partial y} \end{aligned} \tag{2.23}$$

Application of (2.22) and (2.23) to (2.7) changes the factors f and g into:

$$\widehat{f}(y) = \frac{f(\varphi(y))}{(\varphi'(y))^2}, \quad \widehat{g}(y) = \frac{g(\varphi(y))}{\varphi'(y)} - f(\varphi(y))\frac{\varphi''(y)}{(\varphi'(y))^3} \tag{2.24}$$

The boundary points S_{min} and S_{max} are also transformed into $\psi(S_{min})$ and $\psi(S_{max})$, respectively. The equidistant grid size for the transformed equation is $h = (\psi(S_{max}) - \psi(S_{min}))/N$, assuming function ψ to be a monotonically increasing function.

The spatial stretching used for the one-dimensional Black-Scholes equation here originates from [13] and is also presented in [48]:

$$y = \psi(S) = \frac{\sinh^{-1}(\xi\,(S - S_R)) - c_1}{c_2 - c_1}, \tag{2.25}$$

with normalisation constants $c_1 = \sinh^{-1}(\xi(S_{min} - S_R))$ and $c_2 = \sinh^{-1}(\xi(S_{max} - S_R))$, so that $y \in [0, 1]$. The grid is refined around $S = S_R$, which is typically set equal to the strike price K. Parameter ξ determines the amount of stretching. In the analytic function (2.25) the combination ξS_R appears. For satisfactory accuracy, especially on coarse grids, it appears advantageous to keep this quantity constant. Setting $\xi S_R = 15$, for example, has proven to be an appropriate choice over a variety of option pricing parameters (by numerical experiments). Figure 2.1 shows

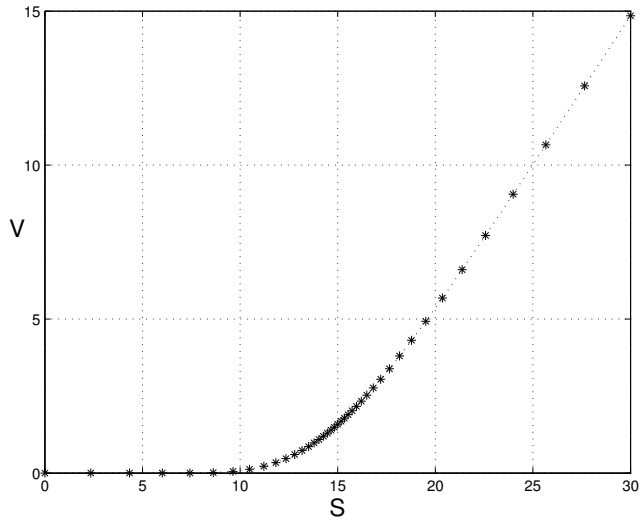


Figure 2.1: Solution of the European call pricing problem on a stretched grid with $\xi = 1$.

the solution of a European call on a stretched grid with stretching around $S_R = 15$ for $\xi = 1$ ($\xi S_R = 15$). The difference in the number of points per

S-interval with $\xi = 1$ and $\xi = 12$, for example. is depicted in Figure 2.2. The number of points per interval is displayed for three grid sizes of 20-, 40- and 80 points with different colours (from light to dark in Fig. 2.2). Thus, larger ξ means fewer points in the outer regions. When ξ decreases the grid approaches an equidistant one.

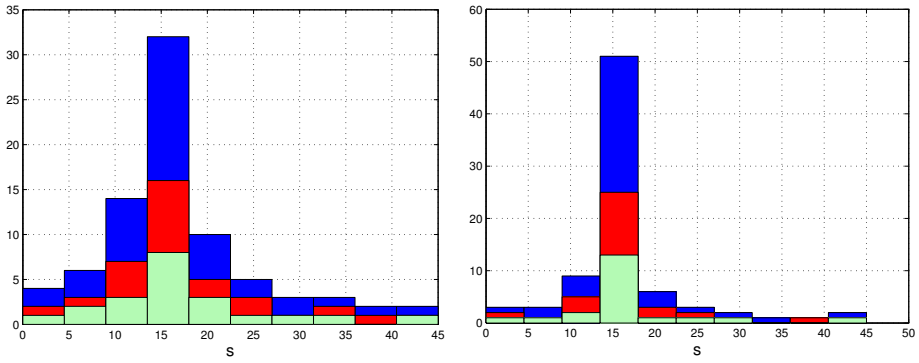


Figure 2.2: Number of grid points in an interval on S-axis for $\xi = 1$ (left) and $\xi = 12$ (right). The number of points is 20, 40 and 80 for the colours from light to dark.

For stretching (2.25), the inverse and the first two derivatives are:

$$\varphi(y) = \frac{1}{\xi}\sinh\left(c_2 y + c_1(1-y)\right) + S_R, \tag{2.26}$$

$$J(y) = \frac{c_2 - c_1}{\xi}\cosh\left(c_2 y + c_1(1-y)\right), \tag{2.27}$$

$$H(y) = \frac{(c_2 - c_1)^2}{\xi}\sinh\left(c_2 y + c_1(1-y)\right). \tag{2.28}$$

Here $J(y)$, the Jacobian, is the first derivative of $\varphi(y)$ and $H(y)$, the Hessian, denotes its second derivative. Applying stretching (2.26) to the final condition gives:

$$\hat{V}(T,y) = \max\left(\frac{1}{\xi}\sinh\left(c_2 y + c_1(1-y)\right) + S_R - K, 0\right). \tag{2.29}$$

The kink in the final condition of a European option does not disappear; the condition is still not differentiable.

For the valuation of the hedge parameters in Section 1.5, we use numerical differentiation. If the numerical solution is known, then a difference equation (see for example equation (2.10)) can be used to compute the Δ on the $S-$grid. However, if the coordinate stretching (2.25) is used, then the equation for Δ changes into:

$$\Delta = \frac{\partial V}{\partial S} = \frac{1}{J(y)}\frac{\partial \widehat{V}}{\partial y}, \tag{2.30}$$

where $\frac{\partial \widehat{V}}{\partial y}$ is obtained by a difference equation with respect to y and $J(y)$ can be found in equation (2.27).

The other hedge parameter of our interest, Γ, is the derivative of the Δ, or the second derivative of the option price. Using the coordinate stretching, we have:

$$\Gamma = \frac{\partial^2 V}{\partial S^2} = J^{-2}(y)\frac{\partial^2 \widehat{V}}{\partial y^2} - H(y)J^{-3}(y)\frac{\partial \widehat{V}}{\partial y}. \tag{2.31}$$

In this equation, $\frac{\partial^2 \widehat{V}}{\partial y^2}$ is the numerical differentiation based on the central difference discretisation on the $y-$grid. $H(y)$ is the second derivative of the stretch function (2.28).

2.5 Numerical experiments

We start with some numerical experiments and solve a European call option and a digital call option. Both options have an analytic solution and therefore these analytic solutions are used to investigate the behaviour of the numerical methods. We will perform numerical experiments with the stretching parameter, ξ, in one dimension in the next two sections and we will use the results for the options with dividend and early exercise. These options do not have an analytic solution.

2.5.1 European option with continuous dividend yield

We start with a reference option:

$$K = 15,\ S_0 = K,\ \sigma = 0.3,\ r = 0.05,\ \delta = 0.03,\ T = 0.5. \tag{2.32}$$

The European (vanilla) call is computed to gain some insight in the properties of the numerical techniques. The numerical solution, its first and second derivatives at initial time $t = 0$ are compared to the analytic solution in the infinity norm. Next to this, the tables below present the error reduction factors c_∞, defined as: $c_\infty = |V_h - V_{ex}|_\infty / |V_{2h} - V_{ex}|_\infty$ for some vector V, where V_h and V_{ex} denote the solution on mesh size h and the exact solution, respectively. We aim for accuracy with only a few grid points in order to the reduce the computational time as much as possible, therefore the grids are typically not finer than 40×40.

Table 2.1 presents results obtained on both a non-stretched as a stretched grid in space based on a fourth order scheme proposed in (2.13). The outer boundary S_{max} has been placed at three times the exercise price, in accordance with equation (2.2).

It is shown in Table 2.1 that the accuracy of the results in V, Δ and Γ is nicely improved with the grid stretching ($\xi = 1$). We do not observe a

Table 2.1: Comparison of error and accuracy in V, Δ and Γ ($t = 0$) for a European call option on non-stretched (Top) and stretched grids with $\xi = 1$ (Bottom).

Grid	$\|V - Vex\|_\infty$	c_∞	$\|\Delta - \Delta_{ex}\|_\infty$	c_∞	$\|\Gamma - \Gamma_{ex}\|_\infty$	c_∞
10×10	1.6×10^{-1}		9.5×10^{-2}		2.2×10^{-2}	
20×20	3.7×10^{-2}	4.3	1.7×10^{-2}	5.6	3.7×10^{-3}	6.1
40×40	7.1×10^{-3}	5.2	2.0×10^{-3}	8.5	6.5×10^{-4}	5.7
10×10	1.1×10^{-2}		1.9×10^{-2}		6.3×10^{-3}	
20×20	1.0×10^{-3}	10.1	3.1×10^{-3}	6.2	1.3×10^{-3}	4.8
40×40	9.3×10^{-5}	11.2	2.9×10^{-4}	10.8	9.7×10^{-5}	13.6

fourth order error reduction on these grids with a "moderate stretching", but the error for 20×20 points is already less than one Euro cent with the transformation. This is a satisfactory result. Other experiments in [36] show that with more severe stretching fourth order convergence could be obtained. However, the error on the coarser grids is significantly larger than that obtained for $\xi = 1$ in Table 2.1. Therefore, $\xi = 1$ is to be preferred. The stretched grid, the solution and derivatives for $\xi = 1$ and $\xi = 12$ are displayed in Figure 2.3.

2.5.2 Digital option with continuous dividend yield

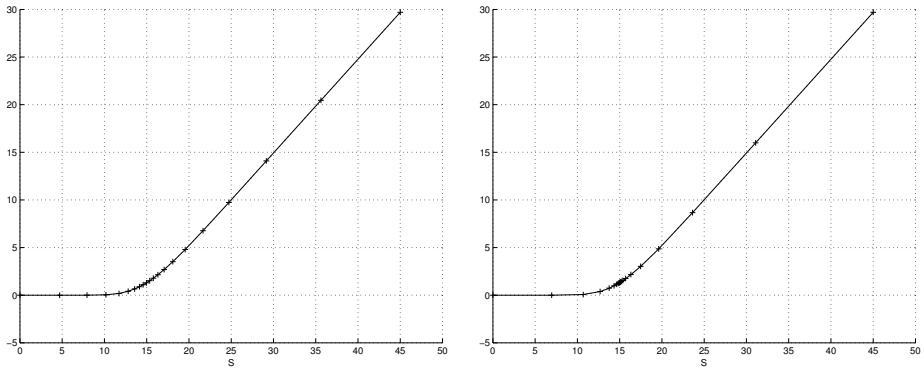
In this section, we evaluate a digital call option. According to [37], the exact position of the discontinuity in the final condition with respect to the position of the grid points is important for satisfactory accuracy. Test results in [37] show that if K is not *e*xactly between two grid points when applying a numerical scheme to a digital option, a satisfactory accuracy and regular grid convergence are not easily obtained.

In addition, for digital options with the discontinuous final condition, we need a proper time integration. The Greeks are the quantities in which numerical oscillations may occur with an improper numerical time integration. Examples can be constructed for which the lack of damping properties of the Crank Nicolson scheme can clearly be seen in Γ, see, for example [37]. Therefore we use the Rannacher time-marching defined in Section 2.3.1 and we start first with equation (2.10), with $\theta = 1$ we continue with the BDF2 (2.12), BDF3 and BDF4 (2.13).

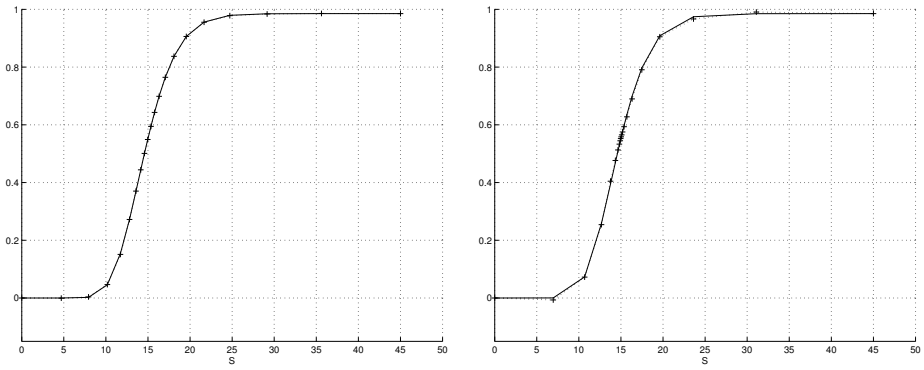
The parameters chosen here for the digital call are:

$$K = 15, \quad \sigma = 0.3, \quad r = 0.05, \quad \delta = 0.03, \quad T = 0.5. \tag{2.33}$$

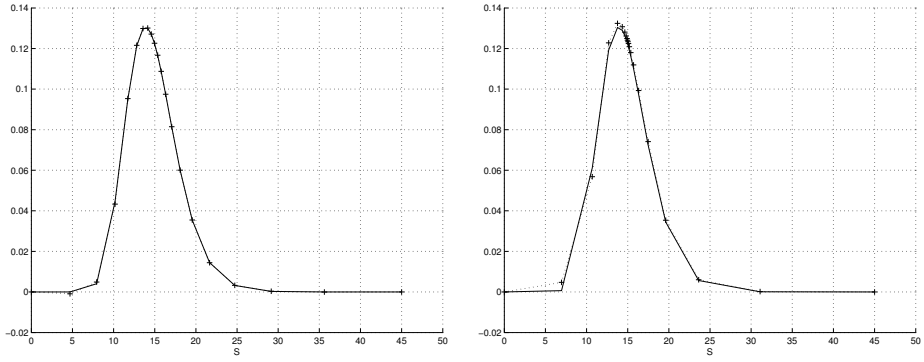
Table 2.2 shows the convergence results for the digital call (2.33), and its



(a) V, $\xi = 1$ (b) V, $\xi = 12$



(c) Δ, $\xi = 1$ (d) Δ, $\xi = 12$



(e) Γ, $\xi = 1$ (f) Γ, $\xi = 12$

Figure 2.3: Plots of numerical option price V, Δ and Γ of a European call, $K = 15, \sigma = 0.3, \delta = 0.03, r = 0.05, T = 0.5$, versus the analytic solution with the 20 points stretched grids.

Table 2.2: Comparison of error and accuracy in V, Δ and Γ ($t = 0$) for a digital call. Fourth order scheme (2.13). Top: without stretching. Bottom: stretched grid with $\xi = 1$.

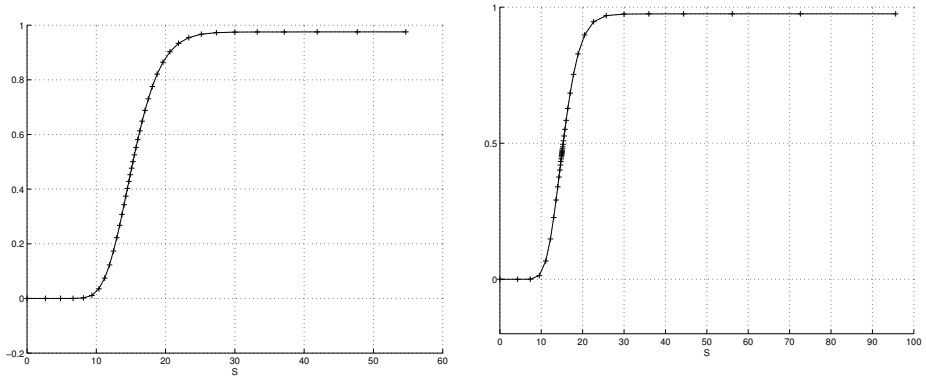
Grid	$\|V - V_{ex}\|_\infty$	c_∞	$\|\Delta - \Delta_{ex}\|_\infty$	c_∞	$\|\Gamma - \Gamma_{ex}\|_\infty$	c_∞
10×10	3.6×10^{-2}		3.1×10^{-2}		1.4×10^{-2}	
20×20	1.5×10^{-2}	2.4	5.1×10^{-3}	6.1	5.8×10^{-3}	2.4
40×40	1.9×10^{-3}	8.0	7.7×10^{-4}	6.7	5.8×10^{-4}	9.9
10×10	5.4×10^{-3}		1.6×10^{-2}		6.3×10^{-3}	
20×20	1.0×10^{-3}	5.4	2.1×10^{-3}	7.6	2.0×10^{-4}	31.0
40×40	7.7×10^{-5}	13.0	2.6×10^{-4}	8.3	9.8×10^{-5}	2.1

derivatives on the stretched grid with $\xi = 1$. The accuracy observed on the coarse grids for the problem with discontinuous final condition resembles the accuracy for the plain vanilla call from Table 2.1 quite well. mesh. When the grid points are not placed properly - such that $S_R = K$ is exactly between two grid points - the accuracy is drastically reduced.

Figure 2.4 displays graphically the option value, Delta and Gamma for the digital call on the stretched grids. It can be observed that with a severe stretching ($\xi = 12$) the grid resolution, around interesting positions for the derivatives may not be sufficient on a relatively coarse grid. This is an explanation of the irregular convergence behaviour that is sometimes observed on these coarse grids. A moderate stretching is again to be preferred.

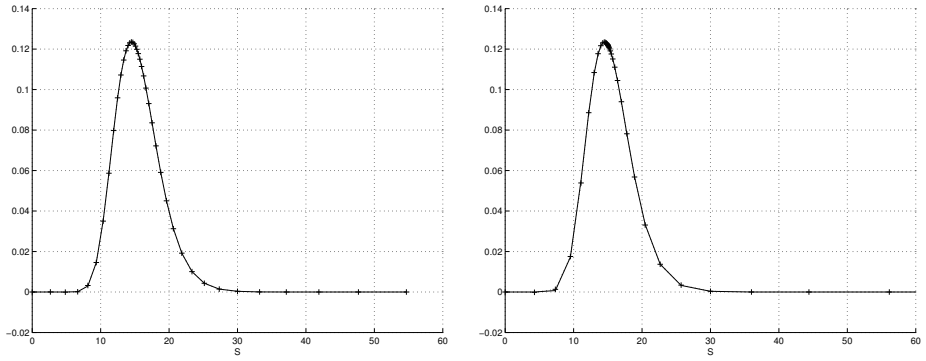
2.5.3 European options with discrete dividend

We now present some numerical examples in which the continuous dividend payment is replaced by a discrete dividend payment. These dividend payments are paid once or twice a year (see Section 1.7) and it may well happen that the dividend payment falls in the lifetime of the option. In order to deal with the jump condition (1.41) accurately, the following procedure is applied. We first perform a regular Black-Scholes computation from the maturity date until the ex-dividend date. This is done with the stretched grid and the fourth order discretisation in space. At the ex-dividend date, (1.41) is taken into account by means of interpolation. Lagrange interpolation of fourth order has been applied to the numerical solution on the stretched grid, since typically $S - D$ (D is the dividend payment) may not be exactly a grid point. After the ex-dividend date the Black-Scholes computation is restarted with the initialisation by the BDF1, BDF2, BDF3 sequence preceding the BDF4 time integration. In our computations we place t_d exactly on a time line, t_d^- and t_d^+ are assumed to exist at the same line. The numerical solutions obtained are compared to exact values for discrete dividends



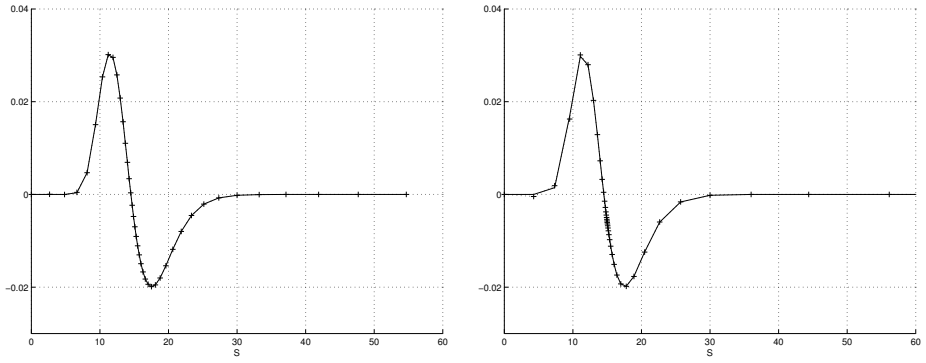
(a) V, $\xi = 1$

(b) V, $\xi = 12$



(c) Δ, $\xi = 1$

(d) Δ, $\xi = 12$



(e) Γ, $\xi = 1$

(f) Γ, $\xi = 12$

Figure 2.4: Plots of numerical option price V, Δ and Γ of a digital call, $K = 15, \sigma = 0.3, \delta = 0.03, r = 0.05, T = 0.5$, versus the analytic solution with the 40 points stretched grids.

found in [18]. The parameters from [18], that we also choose here, are $K = S_{spot} = 100, T = 1, r = 0.06, \sigma = 0.3$ and $D = 7$ (a dividend payment of €7). The dividend payment may take place on three different time points, $t = 0.0001, t = 0.5$ and $t = 0.9999$.

Table 2.3 presents the convergence on different grid sizes, with $\xi = 0.15$, $(\xi K = 15)$. As for the plain vanilla call we observe a small error

Table 2.3: Convergence of option value at $t = 0$ with discrete dividend payment at different t values, moderate stretching $\xi = 0.15$.

Grid	$\|V - V_{ex}\|_\infty$ $t = 0.0001$	c_∞	$\|V - V_{ex}\|_\infty$ $t = 0.5$	c_∞	$\|V - V_{ex}\|_\infty$ $t = 0.9999$	c_∞
10×10	1.7×10^{-2}		1.5×10^{-2}		1.3×10^{-1}	
20×20	4.5×10^{-3}	3.9	4.9×10^{-3}	3.1	1.7×10^{-3}	73.5
40×40	1.0×10^{-3}	4.4	1.1×10^{-3}	4.6	4.0×10^{-4}	4.3
80×80	7.9×10^{-5}	12.7	1.0×10^{-4}	10.6	2.0×10^{-4}	2.0

with moderate stretching (and, not shown, a fourth order convergence with more severe stretching). Also for the extreme cases $t = 0.0001,\ t = 0.9999$ satisfactory results are obtained, i.e. dividend payment just after the start of the contract or just before the maturity date.

We also present some results for multiple discrete dividends, as in [18]. With parameters $S_0 = K = 100, r = 0.06, \sigma = 0.25$ and multiple dividends of four (ex-dividend date is each half year). Table 2.4 presents the numerical results for $T = 1, T = 2$ and $T = 3$, with one, two and three dividend payments, respectively. It compares the numerical approximation to the exact solution from [18]. $S_{max} = 3K$ according to [30] and stretching parameter $\xi = 0.15$. For larger values of T the number of points in time increases proportionally. The numerical results obtained with only a few grid points

Table 2.4: Multiple discrete dividends payments, $K = 100, D = 4, \xi = 0.15$.

	$T = 1$		$T = 2$		$T = 3$
Grid	$V\ (t = 0)$	Grid	$V\ (t = 0)$	Grid	$V\ (t = 0)$
10×10	10.654	10×20	15.229	10×30	18.734
20×20	10.665	20×40	15.210	20×60	16.616
40×40	10.661	40×80	15.202	40×120	18.604
80×80	10.661	80×160	15.201	80×240	18.602
160×160	10.661	160×320	15.201	160×480	18.602
320×320	10.661	320×640	15.201	320×960	18.602
Price [18]	10.661		15.199		18.598

Table 2.5: American put reference problem from [34], $K = 100, D = 2, \xi = 0.15$.

Grid	$V(t = 0, 80)$	$V(t = 0, 100)$	$W(t = 0, 120)$
20×20	0.223	0.105	0.043
40×40	0.223	0.105	0.043
Meyer [34]:	0.223	0.105	0.043

are very satisfactorily. Other discrete dividend results from [18] can also be confirmed accurately with our discretisation techniques.

2.5.4 American options with discrete dividend

We finally consider an American put with parameters from [34]: $K = 100, \sigma = 0.4, r = 0.08, D = 2, t_d = 0.3, T = 0.5$ and $\mu = 0.15, S_{max} = 3K$. Results for $S = 80, 100, 120$ at $t = 0$ on a 20×20- and 40×40-grid are compared to those in [34] in Table 2.5. Algorithm 1 is now used with a suitable tolerance of 10^{-5}. With only 20 points in space and time and grid stretching with $\xi = 0.15$, the results from [34] are reproduced.

We also consider an American put with two ex-dividend dates, $t_{d_1} = 0.3, t_{d_2} = 0.8$ ($T = 1$) and visualise the free boundary as a function of time for different dividend payment strategies (as in [34]). Figure 2.5 shows the free boundary for an American put with problem parameters: $K = 100, \sigma = 0.4, r = 0.08, T = 1$. In the figure the free boundary functions presented are without any dividend payment ($D = 0$, solid line), with a fixed dividend payment $D = 2$ at t_{d_1}, t_{d_2} (dashed line) and with a payment proportional to the asset price $D = 0.02S$ (dotted line) at the ex-dividend dates. It can be seen that after the discrete dividend payment the free boundary may disappear, and reappear, confirming that early exercise is not favourable after an ex-dividend date according to Section 1.7.

2.6 Conclusions

We solved the one-dimensional Black-Scholes equation for a set of reference options with only a few grid points. Fourth order accurate space and time discretisation were proposed, using spatial grid stretching by means of an analytical coordinate transformation. With the proper choices of grid and stretching parameters, the fourth order accuracy can be achieved. Important for our applications is, however, a small discretisation error with only a few grid points. This was achieved by the techniques proposed with a moderate stretching. Furthermore, we have observed a satisfactory accuracy of the

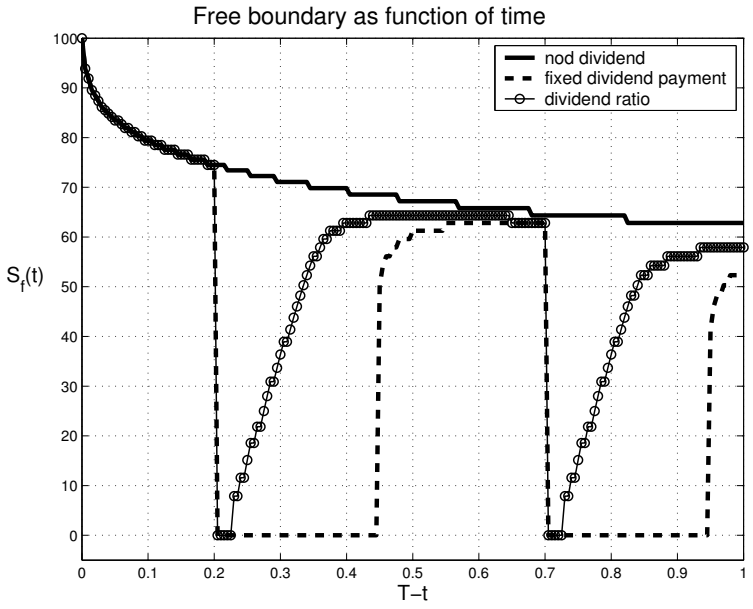


Figure 2.5: Free boundary as function of time with two ex-dividend dates and different forms of dividend payment: $D = 0$ (solid), $D = 2$ (dashed) vs. $D = 0.02S$ (dotted).

hedge parameters. For the reference problem, about 20 space- and time steps were sufficient to get the solution accurate to one Euro cent.

Discrete dividend payment is handled very satisfactorily by the stretched grid discretisation and a fourth order Lagrange interpolation at the ex-dividend date, even with multiple dividend dates. The accuracy is in this case high. American put options with discrete dividend can be solved with the PSOR algorithm and the techniques used for the European options. The PSOR algorithm is a straightforward method for early exercise.

The discretisation in this method can be generalised to the multi-asset option pricing problem. Grid stretching seems an interesting way to concentrate grid points around a desired point. The fourth order scheme is indeed a technique to reduce the number of grid points and time steps.

Chapter 3

Multi-asset option pricing with the PDE method

3.1 Introduction

This chapter covers the numerical solution of the multi-dimensional Black-Scholes partial differential equation (1.9). Each underlying asset represents one of the coordinates in the problem and therefore the problem is $d-$dimensional. The numerical solution ingredients - e.g. the discretisation matrix and final conditions - are constructed automatically. This is done by the use of *Kronecker products*. Kronecker products are powerful matrix operations to set-up a matrix for a $d-$dimensional problem. In fact, only one-dimensional discretisation matrices as developed in the previous chapter are the basic ingredients of this technique. We also show that the mathematical solution of higher-dimensional problems is not necessarily much more difficult than a lower-dimensional problem.

However, since the discretisation of the problem is done on a tensor-based Cartesian grid with $N_i,\ i = 1, \ldots, d$, grid points per coordinate, the total number of grid points is the product of these numbers. When we climb in the dimensions the overall number of grid points or unknowns grows drastically. We introduce the *sparse grid technique* [10, 25] in Section 3.3 to avoid serious problems with memory management. This technique combines solutions on many coarser sub-grids to obtain a quite accurate approximation of the full grid solution. Since each sub-problem, belonging to a sparse grid solution, is independent from the others, this method is straightforward to use in parallel.

Although the sparse grid technique is a powerful tool to overcome the curse of dimensionality and we find satisfactory test results in Section 3.4, the sparse grid technique is not usable under general circumstances. This is certainly a problem for the multi-asset option pricing problem as we will see in Section 3.4.3. Therefore some coordinate transformations are presented in

Section 3.5 to align the kink in the contract function to a grid line. In Section 3.6, we present the results of the sparse grid used in combination with the coordinate transformation and the coordinate stretching (see Section 2.4). Finally we draw our conclusions in Section 3.7.

3.2 Discretisation of the equation

3.2.1 Preliminaries

We will now focus on the discretisation of the multi-dimensional Black-Scholes equation (1.9). We rewrite this partial differential equation as a general partial differential equation:

$$\frac{\partial V}{\partial t} = \sum_{i=1}^{d}\sum_{j=1}^{d} f_{ij}(\mathbf{S})\frac{\partial^2 V}{\partial S_i \partial S_j} + \sum_{i=1}^{d} g_i(\mathbf{S})\frac{\partial V}{\partial S_i} - rV(t,\mathbf{S}). \tag{3.1}$$

The discretisation of this equation is similar to the discretisation of the single asset problem in terms of the matrix based equation (2.17). For example, the $\theta-$scheme for the multi-dimensional partial differential equation reads:

$$\begin{aligned}\mathbf{I}\mathbf{v}^{\nu+1} - \mathbf{I}\mathbf{v}^{\nu} = (1-\theta)\Delta t \left[\sum_{i=1}^{d}\left(\mathbf{F}_{ii}\mathbf{A}_i^d + \mathbf{G}_i\mathbf{B}_i^d + 2\sum_{j=i+1}^{d}\mathbf{F}_{ij}\mathbf{C}_{ij}^d\right) - r\mathbf{I}\right]\mathbf{v}^{\nu} \\ + \theta\Delta t\left[\sum_{i=1}^{d}\left(\mathbf{F}_{ii}\mathbf{A}_i^d + \mathbf{G}_i\mathbf{B}_i^d + 2\sum_{j=i+1}^{d}\mathbf{F}_{ij}\mathbf{C}_{ij}^d\right) - r\mathbf{I}\right]\mathbf{v}^{\nu+1}.\end{aligned}$$

In this equation, the matrices $\mathbf{F}_{ij}$ and $\mathbf{G}_i$ represent the coefficients of the derivatives in equation (3.1). These matrices contain the discrete functions f_{ij} and g_i as in (3.1). Matrices $\mathbf{A}_i^d$ are the difference matrices on a $d-$dimensional Cartesian grid of the second derivative with respect to coordinate S_i. Similarly, the matrices $\mathbf{B}_i^d$ are the difference matrices of the first derivative with respect to coordinate S_i. Finally, the matrices $\mathbf{C}_{ij}^d$, which have a double index, represent the difference matrices of the mixed derivatives with respect to coordinates S_i and S_j. The inner sums run from $j = i+1$ to d and are multiplied by 2, because the coefficients are symmetric, i.e. $f_{ij}(\mathbf{S}) = f_{ji}(\mathbf{S})$ and the mixed derivative itself is a symmetric operator.

The focus in this section is on the construction of the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ by the use of Kronecker products. The Kronecker products can only be used, when a certain grid ordering is obeyed.

Definition 3.2.1. A grid ordering is a system of enumeration of all grid points in a $d-$dimensional grid. If a coordinate $i \in \{1,\ldots,d\}$ has N_i grid points, then a grid point on this grid will be presented by a set of d numbers

$(i_1, i_2, \ldots, i_d)$. Every grid point can also be presented by one unique number m:

$$m = i_1 + \sum_{j=2}^{d} i_j \prod_{k=1}^{j-1} (N_k + 1). \tag{3.2}$$

where i_j is the $j-$th element of the $d-$dimensional $(i_1, i_2, \ldots, i_d)$ set.

Remark 3.2.2. Note that the numbers i with subscripts are different from the number i itself. i is used for the $i-$th dimension and i_j represents the $i-$th grid point in the $j-$th coordinate.

Example 3.2.3. *The point* $\mathbf{S}_0 = (S_1(i_1 = 4), S_2(i_2 = 3), S_3(i_3 = 5))$ *on a three-dimensional grid with* $N_1 = N_2 = N_3 = 32$ *corresponds with row number 5548 in every difference matrix or coefficient matrix. So the matrix element of* $\mathbf{F}_{11}$ *with row and column number 5548 is the value of the coefficient* f_{11} *in the point* $\mathbf{S}_0$*. Note that the matrix has in total 35937 rows.*

3.2.2 Kronecker products

The construction of the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ is now considered in terms of Kronecker products. The Kronecker product is a matrix operation, which is applicable for generally sized matrices, but here we only consider square matrices. (This in contrast to the regular matrix product, whereby the sizes must coincide.) However, the Kronecker product is not commutative in general. The Kronecker product is defined as:

Definition 3.2.4. The Kronecker product $\otimes$ of matrix $\mathbf{E}$ with size $(N_2 + 1) \times (N_2 + 1)$ and matrix $\mathbf{D}$ with size $(N_1 + 1) \times (N_1 + 1)$ is defined by [46]:

$$\mathbf{K} = \mathbf{E} \otimes \mathbf{D} = \begin{pmatrix} d_{00}\mathbf{E} & d_{01}\mathbf{E} & \ldots & d_{0N_1}\mathbf{E} \\ d_{10}\mathbf{E} & d_{11}\mathbf{E} & \ldots & d_{1N_1}\mathbf{E} \\ \vdots & \vdots & & \vdots \\ d_{N_10}\mathbf{E} & d_{N_11}\mathbf{E} & \ldots & d_{N_1N_1}\mathbf{E} \end{pmatrix}. \tag{3.3}$$

The elements in the matrix $\mathbf{K}$ are related to the entries in the original matrices $\mathbf{D}$ and $\mathbf{E}$ as follows:

$$(\mathbf{K})_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} = (\mathbf{D})_{i_2,k_2} (\mathbf{E})_{i_1,k_1} \tag{3.4}$$

Lemma 3.2.5. *Given matrices* $\mathbf{P}, \mathbf{Q}$ *and* $\mathbf{R}$ *and identity matrices* $\mathbf{I}_i$ *and* $\mathbf{I}_j$ *of sizes* N_i *and* N_j*, respectively, then*

$$\mathbf{I}_i \otimes \mathbf{I}_j = \mathbf{I}_j \otimes \mathbf{I}_i$$
$$\mathbf{P} \otimes (\mathbf{Q} + \mathbf{R}) = \mathbf{P} \otimes \mathbf{Q} + \mathbf{P} \otimes \mathbf{R}$$

Proof. See [23] □

Definition 3.2.6. The repeated Kronecker product $\bigotimes$ is defined as:

$$\bigotimes_{j=1}^{d} \mathbf{E}_j = \mathbf{E}_1 \otimes \mathbf{E}_2 \otimes \ldots \otimes \mathbf{E}_d. \tag{3.5}$$

Furthermore: $\bigotimes_{j=n}^{n} \mathbf{E}_j = \mathbf{E}_n$ and $\bigotimes_{j=n+1}^{n} \mathbf{E}_j = 1$.

3.2.3 General difference matrix

The construction of the matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ is similar and for simplicity we focus on a discretisation of a derivative with a general difference equation. In this section, the matrix $\mathbf{A}$ can therefore be replaced by $\mathbf{B}$. In Section 3.2.4, the matrices $\mathbf{C}_{ij}$ are presented in detail. They are treated separately, because they are dependent on more than one coordinate.

First we start with the definition of the difference matrix.

Definition 3.2.7. A difference matrix $\mathbf{A}_i^d$ is a matrix with as the entries the coefficients of the discretisation of a derivative with respect to coordinate i and based on a $d-$dimensional grid.

Example 3.2.8. *Equations (2.15) and (2.16) are examples of difference matrices. In the notation of Definition 3.2.7, the matrix in equation (2.15) is presented as* $\mathbf{A}_1^1$ *and the matrix in equation (2.16) as* $\mathbf{B}_1^1$.

Remark 3.2.9. The notation of Definition 3.2.7 is usable for general cases and we also use cases when $d < i$. This notation is a representation of the discretisation of a derivative with respect to coordinate i. The grid has d coordinates and i is one of the coordinates. The matrix $\mathbf{A}_2^1$ has in fact nothing to do with the dimensionality of the problem. It is an abbreviation of the discretisation of a derivative with respect to coordinate 2 on a standard one-dimensional grid. It can be seen as a one-dimensional problem based on a grid of coordinate 2.

Consider a discretisation of a derivative with respect to the first coordinate. It can be either a first derivative or a second derivative. The central difference schemes in Section 2.3.1 are used. The difference matrix on a $d-$dimensional grid is derived using mathematical induction. Therefore the two-dimensional difference matrix $\mathbf{A}_1^2$ is necessary as a first induction step.

Lemma 3.2.10. *Let* $\mathbf{A}_1^1$ *be the matrix corresponding the difference equation of the discretisation with respect to the first coordinate. The difference matrix based on a two-dimensional grid respecting the grid ordering from Definition 3.2.1 reads:*

$$\mathbf{A}_1^2 = \mathbf{I}_2 \otimes \mathbf{A}_1^1 \tag{3.6}$$

where $\mathbf{I}_2$ *is the identity matrix of size* $(N_2 + 1) \times (N_2 + 1)$.

Proof. Using the Kronecker product of two matrices defined element-wise according to equation (3.4) we have:

$$\begin{aligned}\left(\mathbf{A}_1^2\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} &= \left(\mathbf{I}_2 \otimes \mathbf{A}_1^1\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} \\ &= \left(\mathbf{I}_2\right)_{i_2,k_2} \left(\mathbf{A}_1^1\right)_{i_1,k_1} \\ &= \begin{cases} \left(\mathbf{A}_1^1\right)_{i_1,k_1} & i_2 = k_2 \\ 0 & i_2 \neq k_2 \end{cases}\end{aligned} \tag{3.7}$$

If a general difference matrix is used for the second derivative to the first coordinate (see for example equation (2.15)), then we can write for the approximated value in the coordinate (i_1, i_2) in a one-dimensional representation:

$$\left[\frac{\partial^2 V}{\partial S_1^2}\right]_{i_1,i_2} \approx \sum_{k_1=0}^{N_1} \left(\mathbf{A}_1^1\right)_{i_1,k_1} V^{\nu}_{k_1+i_2(N_1+1)}$$

and similar for the first derivative. The index number i_2 is fixed. In the two-dimensional case, we can set-up a difference equation with a contribution of all grid points:

$$\left[\frac{\partial^2 V}{\partial S_1^2}\right]_{i_1+i_2(N_1+1)} \approx \sum_{k_1=0}^{(N_1+1)(N_2+1)-1} \left(\mathbf{A}_1^2\right)_{i_1+i_2(N_1+1),k_1} V^{\nu}_{k_1}.$$

However, the finite difference schemes for the derivatives with respect to one coordinate are based on contributions of values along that coordinate. Hence:

$$\left[\frac{\partial^2 V}{\partial S_1^2}\right]_{i_1+i_2(N_1+1)} \approx \sum_{k_1=0}^{N_1} \left(\mathbf{A}_1^2\right)_{i_1+i_2(N_1+1),k_1+i_2(N_1+1)} V^{\nu}_{k_1+i_2(N_1+1)}.$$

The coefficients are the same as in the one-dimensional representation, so we have:

$$\left(\mathbf{A}_1^2\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} = \begin{cases} \left(\mathbf{A}_1^1\right)_{i_1,k_1} & i_2 = k_2 \\ 0 & i_2 \neq k_2 \end{cases}$$

and this is the same as equation (3.7). □

Corollary 3.2.11. *Let $\mathbf{A}_2^1$ be the matrix corresponding the difference equation of the discretisation with respect to the second coordinate. The difference matrix based on a two-dimensional grid respecting the grid ordering from Definition (3.2.1) reads:*

$$\mathbf{A}_2^2 = \mathbf{A}_2^1 \otimes \mathbf{I}_1 \tag{3.8}$$

where $\mathbf{I}_1$ is the identity matrix of size $(N_1 + 1) \times (N_1 + 1)$.

Proof. Using again the Kronecker product of two matrices defined element-wise according to equation (3.4) we have:

$$\begin{aligned}\left(\mathbf{A}_2^2\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} &= \left(\mathbf{A}_2^1 \otimes \mathbf{I}_1\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} \\ &= \left(\mathbf{A}_2^1\right)_{i_2,k_2} \left(\mathbf{I}_1\right)_{i_1,k_1} \\ &= \begin{cases} \left(\mathbf{A}_2^1\right)_{i_2,k_2} & i_1 = k_1 \\ 0 & i_1 \neq k_1 \end{cases}\end{aligned} \tag{3.9}$$

Now, if the general difference matrix is used for the second derivative to the second coordinate, then we can write in a one-dimensional representation:

$$\left[\frac{\partial^2 V}{\partial S_2^2}\right]_{i_1,i_2} \approx \sum_{k_2=0}^{N_2} \left(\mathbf{A}_2^1\right)_{i_2,k_2} V^{\nu}_{i_1+k_2(N_1+1)}$$

and similar for the first derivative. The index number i_1 is fixed. Similar to the proof of Lemma 3.2.10, the difference equation in two dimensions can be constructed using all grid points. However, the finite difference schemes for the derivatives with respect to one coordinate are based on contributions of values along that coordinate. Hence:

$$\left[\frac{\partial^2 V}{\partial S_2^2}\right]_{i_1+i_2(N_1+1)} \approx \sum_{k_2=0}^{N_2} \left(\mathbf{A}_2^2\right)_{i_1+i_2(N_1+1),i_1+k_2(N_1+1)} V^{\nu}_{i_1+k_2(N_1+1)}.$$

The coefficients are the same as in the one-dimensional representation, so we have:

$$\left(\mathbf{A}_2^2\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} = \begin{cases} \left(\mathbf{A}_2^1\right)_{i_2,k_2} & i_1 = k_1 \\ 0 & i_1 \neq k_1 \end{cases}$$

and this is the same as equation (3.9). □

Now the general $d-$dimensional difference matrix is constructed.

Proposition 3.2.12. *The difference matrix of the discretisation of a derivative with respect to the coordinate i based on a $d-$dimensional grid reads:*

$$\mathbf{A}_i^d = \left(\bigotimes_{j=i+1}^{d} \mathbf{I}_j\right) \otimes \mathbf{A}_i^1 \otimes \left(\bigotimes_{j=1}^{i-1} \mathbf{I}_j\right) \tag{3.10}$$

Proof. The proof consists of three parts. First the case $i = 1$. Using Lemma 3.2.10 as the induction step, it follows for $d > 2$:

$$\mathbf{A}_1^{d+1} = \mathbf{I}_{d+1} \otimes \mathbf{A}_1^d = \mathbf{I}_{d+1} \otimes \left(\bigotimes_{j=2}^{d} \mathbf{I}_j\right) \otimes \mathbf{A}_1^1 = \left(\bigotimes_{j=2}^{d+1} \mathbf{I}_j\right) \otimes \mathbf{A}_1^1$$

where

$$\mathbf{I}_{d+1} \otimes \bigotimes_{j=2}^{d} \mathbf{I}_j = \left(\bigotimes_{j=2}^{d} \mathbf{I}_j \right) \otimes \mathbf{I}_{d+1} = \bigotimes_{j=2}^{d+1} \mathbf{I}_j$$

follows from Lemma 3.2.5.
The next step is $i = d$. Corollary 3.2.11 is used as induction step. Hence for $d > 2$, we have:

$$\mathbf{A}_d^d = \mathbf{A}_d^{d-1} \otimes \mathbf{I}_1 = \mathbf{A}_d^{d-2} \otimes \mathbf{I}_2 \otimes \mathbf{I}_1 = \ldots = \mathbf{A}_d^1 \otimes \bigotimes_{j=1}^{d-1} \mathbf{I}_j$$

Finally, for $2 \leqslant i \leqslant d$, we have:

$$\mathbf{A}_i^{d+1} = \mathbf{I}_{d+1} \otimes \mathbf{A}_i^d.$$

This is in fact an extension of the $i = 1$ part of the proof. Now we use Lemma 3.2.10 and Corollary 3.2.11 as induction steps and we have:

$$\begin{aligned}
\mathbf{A}_i^{d+1} = \mathbf{I}_{d+1} \otimes \mathbf{A}_i^d &= \mathbf{I}_{d+1} \otimes \left[\left(\bigotimes_{j=i+1}^{d} \mathbf{I}_j \right) \otimes \mathbf{A}_i^{d-i-1} \right] \\
&= \left(\bigotimes_{j=i+1}^{d+1} \mathbf{I}_j \right) \otimes \mathbf{A}_i^{d-i-1} \\
&= \left(\bigotimes_{j=i+1}^{d+1} \mathbf{I}_j \right) \otimes \mathbf{A}_i^1 \otimes \left(\bigotimes_{j=1}^{i-1} \mathbf{I}_j \right).
\end{aligned}$$

□

3.2.4 The mixed derivative

The discretisation of the mixed second derivative leads to a difference matrix with respect to two different coordinates. The mixed derivative can be written as:

$$\frac{\partial^2 V}{\partial S_i \partial S_j} = \frac{\partial}{\partial S_i} \left(\frac{\partial V}{\partial S_j} \right) = \frac{\partial}{\partial S_j} \left(\frac{\partial V}{\partial S_i} \right) \tag{3.11}$$

Since the mixed derivative is symmetric. the discretisation does not depend on the order. Therefore $i < j$ is assumed without loss of generality.

Consider the two-dimensional case. In this case there is only one possibility: the derivative with respect to S_1 and S_2. First we use a discretisation for a first derivative with respect to S_1. The discretisation matrix is known as $\mathbf{B}_1^1$, where the method of discretisation can be a second or fourth order central scheme. We then have a difference equation in each point (i_1, i_2), which reads:

$$\left[\frac{\partial V}{\partial S_1} \right]_{i_1,i_2}^{\nu} = \left(\mathbf{B}_1^1 \mathbf{v}^{\nu} \right)_{i_1,i_2} = \sum_{k_1=0}^{N_1} \left(\mathbf{B}_1^1 \right)_{i_1,k_1} V_{k_1,i_2}^{\nu}$$

Now we apply the difference equation of the first derivative with respect to the second coordinate:

$$\left[\frac{\partial}{\partial S_2}\frac{\partial V}{\partial S_1}\right]^{\nu}_{i_1,i_2} = \sum_{k_2=0}^{N_2}\left(\mathbf{B}_2^1\right)_{i_2,k_2}\sum_{k_1=0}^{N_1}\left(\mathbf{B}_1^1\right)_{i_1,k_1}V^{\nu}_{k_1,k_2}$$

We write this in grid ordering notation according to Definition (3.2.1):

$$\left[\frac{\partial}{\partial S_2}\frac{\partial V}{\partial S_1}\right]^{\nu}_{i_1+i_2(N_1+1)} = \sum_{k_2=0}^{N_2}\sum_{k_1=0}^{N_1}\left(\mathbf{B}_2^1\right)_{i_2,k_2}\left(\mathbf{B}_1^1\right)_{i_1,k_1}V^{\nu}_{k_1+k_2(N_1+1)}$$

The subscript of $V^{\nu}_{k_1+k_2(N+1)}$ is the representation of column numbers of the large matrix which represents the two-dimensional matrix for the mixed derivative. Of course the row number defined in Definition 3.2.1 is indicated by the point (i_1, i_2) in which the mixed derivative is discretised. The difference matrix for the mixed derivative is called $\mathbf{C}^d_{ij}$, representing the mixed derivative to coordinate i and j with $i < j$ based on a $d-$dimensional grid:

$$\left(\mathbf{C}^2_{1,2}\right)_{i_1+i_2(N_1+1),k_1+k_2(N_1+1)} = \left(\mathbf{B}_1^2\right)_{i_2,k_2}\left(\mathbf{B}_1^1\right)_{i_1,k_1} \tag{3.12}$$

By use of the definition of the Kronecker product (Definition 3.2.4) we have proven Proposition 3.2.13.

Proposition 3.2.13. *Let $\mathbf{B}_1^1$ be the difference matrix of the first derivative with respect to S_1 and $\mathbf{B}_2^1$ the difference matrix of the first derivative with respect to S_2 (see Remark 3.2.9), then the difference matrix of the mixed derivative to the coordinates S_1 and S_2, $\mathbf{C}^2_{1,2}$ reads:*

$$\mathbf{C}^2_{1,2} = \mathbf{B}_2^1 \otimes \mathbf{B}_1^1 \tag{3.13}$$

According to Proposition 3.2.12, this property for the mixed derivative with respect to coordinate S_1 and S_2 can be extended to higher dimensions. Hence:

$$\mathbf{C}^d_{1,2} = \left(\bigotimes_{j=3}^{d}\mathbf{I}_j\right)\otimes\mathbf{C}^2_{1,2} = \left(\bigotimes_{j=3}^{d}\mathbf{I}_j\right)\otimes\mathbf{B}_2^1\otimes\mathbf{B}_1^1 = \mathbf{B}_2^{d-1}\otimes\mathbf{B}_1^1.$$

This property holds for all mixed derivatives with respect to coordinates S_i and S_{i+1} by using Proposition 3.2.12:

$$\mathbf{C}^d_{i,i+1} = \left(\bigotimes_{j=i+2}^{d}\mathbf{I}_j\right)\otimes\mathbf{B}_{i+1}^1\otimes\mathbf{B}_i^1\otimes\left(\bigotimes_{j=1}^{i-1}\mathbf{I}_j\right) = \mathbf{B}_{i+1}^{d-i}\otimes\mathbf{B}_i^i.$$

For other mixed derivatives, for example with respect to coordinate S_1 and S_3, we can write:

$$\mathbf{C}^3_{1,3} = \left[\frac{\partial^2}{\partial S_1\partial S_3}\right]^3 = \left[\frac{\partial}{\partial S_3}\frac{\partial}{\partial S_1}\right]^3 = \mathbf{B}_3^1\otimes\mathbf{B}_1^2 \tag{3.14}$$

since $\mathbf{C}^2_{1,3}$ does not exist in our grid ordering. We thus have to use a two-dimensional representation of the difference matrix of one of the two coordinates and a standard one-dimensional difference matrix for the other coordinate. Note that equation (3.14) can be written using Proposition 3.2.12 as:

$$\mathbf{C}^3_{1,3} = \mathbf{B}^1_3 \otimes \mathbf{B}^2_1 = \mathbf{B}^1_3 \otimes \mathbf{I}_2 \otimes \mathbf{B}^1_1. \tag{3.15}$$

We now have to construct the discretisation for a general mixed derivative, and equation (3.15) can be used as induction step.

Proposition 3.2.14. *Let $\mathbf{B}^1_i$ and $\mathbf{B}^1_j$ be difference matrices of the discretisation of the first derivative to the coordinate S_i and S_j respectively with $i < j$. Then the difference matrix of the mixed derivative to the coordinates S_i and S_j on a $d-$dimensional grid reads:*

$$\mathbf{C}^d_{i,j} = \left(\bigotimes_{k=j+1}^{d} \mathbf{I}_k\right) \otimes \mathbf{B}^1_j \otimes \left(\bigotimes_{k=i+1}^{j-1} \mathbf{I}_k\right) \otimes \mathbf{B}^1_i \otimes \left(\bigotimes_{k=i}^{i-1} \mathbf{I}_k\right) \tag{3.16}$$

Proof. First the case with $j = d$:

$$\mathbf{C}^d_{i,d} = \mathbf{B}^1_d \otimes \mathbf{B}^{d-1}_i = \mathbf{B}^1_d \otimes \left(\bigotimes_{k=i+1}^{d-1} \mathbf{I}_k\right) \otimes \mathbf{B}^1_i \otimes \left(\bigotimes_{k=1}^{i-1} \mathbf{I}_k\right)$$

and this is equation (3.16) where the first repeated Kronecker product is 1 according to Definition 3.2.6. Furthermore for $i = 1$ we have:

$$\mathbf{C}^d_{1,j} = \mathbf{B}^{d-1}_j \otimes \mathbf{B}^1_1 = \left(\bigotimes_{k=j+1}^{d} \mathbf{I}_k\right) \otimes \mathbf{B}^1_j \otimes \left(\bigotimes_{k=2}^{j-1} \mathbf{I}_k\right) \otimes \mathbf{B}^1_1$$

where the last repeated Kronecker product of equation (3.16) is equal to 1 according to Definition 3.2.6. Finally with $j = i + 1$:

$$\mathbf{C}^d_{i,i+1} = \mathbf{B}^{d-i}_{i+1} \otimes \mathbf{B}^i_i = \left(\bigotimes_{k=i+1}^{d} \mathbf{I}_k\right) \otimes \mathbf{B}^1_{i+1} \otimes \mathbf{B}^1_i \otimes \left(\bigotimes_{k=1}^{i-1} \mathbf{I}_k\right)$$

and we see that the middle repeated Kronecker product vanishes. For the other cases we use induction and Proposition 3.2.13 and equation (3.15) are the induction steps. Now:

$$\begin{aligned}
\mathbf{C}^{d+1}_{i,j} &= \mathbf{I}_{d+1} \otimes \mathbf{C}^d_{i,j} \\
&= \mathbf{I}_{d+1} \otimes \left(\bigotimes_{k=j+1}^{d} \mathbf{I}_k\right) \otimes \mathbf{B}^1_j \otimes \left(\bigotimes_{k=i+1}^{j-1} \mathbf{I}_k\right) \otimes \mathbf{B}^1_i \otimes \left(\bigotimes_{k=i}^{i-1} \mathbf{I}_k\right) \\
&= \left(\bigotimes_{k=j+1}^{d+1} \mathbf{I}_k\right) \otimes \mathbf{B}^1_j \otimes \left(\bigotimes_{k=i+1}^{j-1} \mathbf{I}_k\right) \otimes \mathbf{B}^1_i \otimes \left(\bigotimes_{k=i}^{i-1} \mathbf{I}_k\right)
\end{aligned}$$

□

3.3 Sparse grids

Theoretically, it is possible to solve the discrete system for a general number of dimensions. However, in computational science, a major problem occurs when d increases. The size of the discrete solution and the necessary matrices increases drastically. A five-dimensional problem with 32 points per coordinate in the discrete system employs vectors having 32 million components and the corresponding matrix is square and has the order 32 million. Increasing d further implies severe memory requirements on desktop computers, which is usually not an option. This is called the *curse of dimensionality* [3]. In this section, the sparse grid technique will be presented. This technique is developed by Zenger and co-workers [10, 55]. The basic idea of the technique is that a mimic of the original tensor-based grid is computed by the interpolative use of the solutions on particular coarse grids of the same dimensionality. The number of sub-problems to solve will increase, while the computational time per problem decreases drastically. The computational structure - based on the multi-index - can be used in parallel very easily as each sub-grid generated from the multi-index is independent of the others. However, this combination technique will cause inaccuracy, however since this error can be modelled, it is possible to compute the required number of sub-problems and interpolate them to get a reasonable mimic of the full grid solution. With the sparse grid method, the curse of dimensionality can be broken, so that reaching a dimensionality up to seven dimensions with standard computational systems is possible.

First of all, the two-dimensional sparse grid technique will be explained in detail; the error analysis of the complete method is based on the case $d = 2$. Thereafter, the error analysis will be extended to $d = 3$ and generalised to $d > 3$. Other combination techniques than the standard ones will be presented as well.

3.3.1 Basic combination technique in two dimensions

Consider a general two-dimensional discretised problem based on an equidistant grid with N points in both directions and suppose N is a power of 2. Later, we will generalise the analysis to basic rectangular grids with different numbers of points in each direction. We recall the *full grid solution* of this problem as the problem solved on the $N \times N$ grid. Now consider the computation of two solutions. One is on a $\frac{1}{2}N \times N$ sub-grid and the other is on a $N \times \frac{1}{2}N$ sub-grid. If these two solutions are added together, then the solution in some points is taken twice. Therefore the solution computed on a $\frac{1}{2}N \times \frac{1}{2}N$ sub-grid can be subtracted to correct for these double points. Now, this step is repeated on both the $\frac{1}{2}N \times N$ sub-grid as on the $N \times \frac{1}{2}N$ sub-grid. Then we have one solution on a $\frac{N}{4} \times N$ sub-grid, one on a $N \times \frac{N}{4}$ sub-grid and two solutions on a $\frac{N}{2} \times \frac{N}{2}$ sub-grid. Again,

when adding these solutions, some solutions on some points occur more than once and this must be corrected. The sum of the solutions on the sub-grids $\frac{N}{4} \times \frac{N}{2}$ and $\frac{N}{2} \times \frac{N}{4}$ has to be subtracted. This procedure can be continued until we reach the lowest possible number of grid points. Suppose we need at least two points per coordinate (i.e. two cells and three grid points). Then the set of sub-grids, $\mathcal{G}_A$, on which the solution is computed, is $\mathcal{G}_A = [(N \times 2), (\frac{N}{2} \times 4), (\frac{N}{4} \times 8), \ldots, (4 \times \frac{N}{2}), (2 \times N)]$. Then the sum of all solutions has too many contributions of some points. If the solution is computed as well on all sub-grids from the set $\mathcal{G}_B = [(\frac{N}{2} \times 2), (\frac{N}{4} \times 4), \ldots, (4 \times \frac{N}{4}), (2 \times \frac{N}{2})]$, and subtracted then, the combination is done in a proper way. This procedure is presented in Figure 3.1 for $N = 16$. The four top sub-grids represent the solution on the sub-grids $[(16 \times 2), (8 \times 4), (4 \times 8), (2 \times 16)]$ and the next three sub-grids represent the solution on the sub-grids $[(8 \times 2), (4 \times 4), (2 \times 8)]$.

The sub-grids in both sets are ordered according to their number of points. In the first set, all sub-grids have 32 grid points and in the second set, the sub-grids have 16 grid points. To develop the combination technique of the sparse grids, it is a convenient choice to use the logarithm of these numbers of points. We define a new number, n_f, which corresponds to the number of points in each direction of the full grid: $N = 2^{n_f}$. Furthermore, we define n_i, with $i = 1, 2$ as the numbers corresponding to the number of grid points of the sub-grids in $\mathcal{G}_A$ and $\mathcal{G}_B$, such that $N_i = 2^{n_i}$. Hence, if new sets $\mathcal{I}_A$ and $\mathcal{I}_B$ are constructed based on the numbers n_1 and n_2, then each element in the sets $\mathcal{I}_A$ and $\mathcal{I}_B$ contains two numbers n_1 and n_2 according to the sub-grids defined in $\mathcal{G}_A$ and $\mathcal{G}_B$. In the sets $\mathcal{G}_A$ and $\mathcal{G}_B$ the product of the number of grid points is the same. By the definition of n_i, we see that $2^{n_1} \times 2^{n_2}$ is equal for all sub-grids in $\mathcal{G}_A$ or $\mathcal{G}_B$. In the new sets $\mathcal{I}_A$ and $\mathcal{I}_B$, we see that $n_1 + n_2$ is equal for all elements. The counterparts of $\mathcal{G}_A$ and $\mathcal{G}_B$ reads:

$$\mathcal{I}_A = [(n_f, 1), (n_f - 1, 2), (n_f - 2, 3), \ldots, (2, n_f - 1), (1, n_f)]$$
$$\mathcal{I}_B = [(n_f - 1, 1), (n_f - 2, 2), \ldots, (2, n_f - 2), (1, n_f - 1)].$$

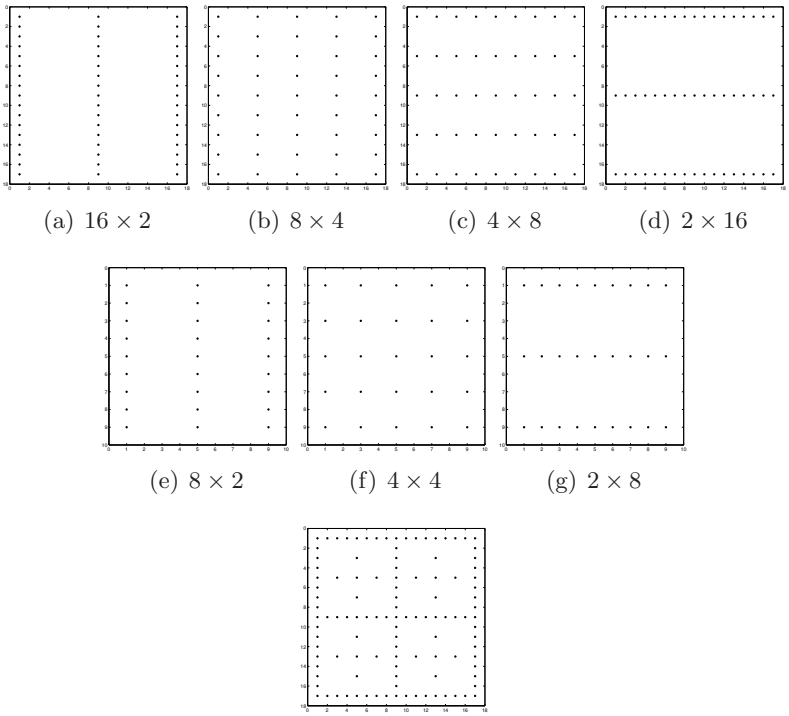
The sets $\mathcal{I}_A$ $\mathcal{I}_B$ are called *multi-indices* here. The layer number l is the sum of the numbers, n_i, in each element of the multi-index. Then we see that our set of sub-grids $\mathcal{I}_A$ has a layer number of $l = n_f + 1$ and $\mathcal{I}_B$ has a layer number of $l = n_f$.

We now arrive at the combination formula developed by Griebel et al. [25] as the sum over the two-dimensional sub-grids from the multi-index $\mathcal{I}$ for which the layer number is $n_f + 1$ and n_f:

$$V_{n_f}^c = \sum_{\{(n_1,n_2) \in \mathcal{I}: l = n_f + 1\}} V_{n_1,n_2} - \sum_{\{(n_1,n_2) \in \mathcal{I}: l = n_f\}} V_{n_1,n_2}, \tag{3.17}$$

with V_{n_1,n_2} the solution on a sub-grid from the multi-index. The solution $V_{n_f}^c$ is computed on a *sparse grid* defined by (3.17). This combined sparse

grid is presented for a two-dimensional grid with $N = 16$ ($n_f = 4$) in Figure 3.1h as well as the construction from the sub-grids from the multi-indices. In this example we have for the top sub-grids (a-d) $l = 5$ and for the lower sub-grids (e-g) $l = 4$.



(a) 16×2 (b) 8×4 (c) 4×8 (d) 2×16

(e) 8×2 (f) 4×4 (g) 2×8

(h) Combined

Figure 3.1: Construction of a 2D sparse grid; (a)–(d): sub-grids on layer with $l = 5$, (e)–(g): sub-grids on layer with $l = 4$; (h) combined sparse grid solution

3.3.2 Error expansion of the two-dimensional sparse grid

The combination expression for a two-dimensional sparse grid solution in equation (3.17) is a mimic or approximation of a solution on a full grid with size 2^{n_f} in each direction. The accuracy of the solution by using this combination equation is the subject of this section. Suppose that an analytic solution U of the numerical problem is sufficiently smooth and that for every point in the grid an error expansion of the type

$$U - V_{n_1,n_2} = C_1(h_{n_1})h_{n_1}^2 + C_2(h_{n_2})h_{n_2}^2 + D(h_{n_1}, h_{n_2})h_{n_1}^2 h_{n_2}^2. \qquad (3.18)$$

exists, where $C_1(h_{n_1}), C_2(h_{n_2})$ and $D(h_{n_1}, h_{n_2})$ are appropriate coefficient functions bounded by a certain constant κ, i.e. $|C_1(h_{n_1})| \leqslant \kappa, |C_2(h_{n_2})| \leqslant$

κ and $|D(h_{n_1}, h_{n_2})| \leqslant \kappa, \forall h_{n_1}, h_{n_2}$. V_{n_1,n_2} corresponds to the numerical solution on a sub-grid with h_{n_1} as mesh size for the first coordinate and h_{n_2} as mesh size for the second coordinate. The numbers n_1 and n_2 come from the multi-index. This expansion holds for e.g. Poisson's problem on a unit cube with sufficiently smooth Dirichlet or Neumann boundary conditions and a sufficiently smooth source function. For the Laplace equation a proof of this expansion is provided in [28, 11].

Consider the combined solution on the sparse grid as given by equation (3.17), then the error reads:

$$\begin{aligned} U - V_{n_f}^c &= U - \sum_{\{(n_1,n_2)\in\mathcal{I}:l=n_f+1\}} V_{n_1,n_2} + \sum_{\{(n_1,n_2)\in\mathcal{I}:l=n_f\}} V_{n_1,n_2} \\ &= \sum_{\{(n_1,n_2)\in\mathcal{I}:l=n_f+1\}} (U - V_{n_1,n_2}) - \sum_{\{(n_1,n_2)\in\mathcal{I}:l=n_f\}} (U - V_{n_1,n_2})\,. \end{aligned}$$

The sums are running over all combinations, n_1, n_2, where $n_1 + n_2 = n_f + 1$ and $n_1 + n_2 = n_f$, respectively, where $n_1, n_2 \geqslant 1$:

$$U - V_{n_f}^c = \sum_{n_1+n_2=n_f+1} (U - V_{n_1,n_2}) - \sum_{n_1+n_2=n_f} (U - V_{n_1,n_2})$$

Using the error expansion (3.18), we have:

$$\begin{aligned} U - V_{n_f}^c = &\sum_{n_1+n_2=n_f+1} \left(C_1(h_{n_1})h_{n_1}^2 + C_2(h_{n_2})h_{n_2}^2 + D(h_{n_1}, h_{n_2})h_{n_1}^2 h_{n_2}^2\right) \\ &- \sum_{n_1+n_2=n_f} \left(C_1(h_{n_1})h_{n_1}^2 + C_2(h_{n_2})h_{n_2}^2 + D(h_{n_1}, h_{n_2})h_{n_1}^2 h_{n_2}^2\right) \end{aligned}$$

If the terms in the sums are rearranged, we have:

$$\begin{aligned} &\sum_{n_1+n_2=n_f+1} C_1(h_{n_1})h_{n_1}^2 - \sum_{n_1+n_2=n_f} C_1(h_{n_1})h_{n_1}^2 \\ &= \sum_{n_1=1}^{n_f} C_1(h_{n_1})h_{n_1}^2 - \sum_{n_1=1}^{n_f-1} C_1(h_{n_1})h_{n_1}^2 = C_1(h_{n_f})h_{n_f}^2 \end{aligned}$$

and

$$\sum_{n_1+n_2=n_f+1} C_2(h_{n_2})h_{n_2}^2 - \sum_{n_1+n_2=n_f} C_2(h_{n_2})h_{n_2}^2 = C_2(h_{n_f})h_{n_f}^2$$

Now using $h_{n_1} h_{n_2} = 2^{-(n_1+n_2)}$, the combined error term reads:

$$\begin{aligned}
&\sum_{n_1+n_2=n_f+1} D(h_{n_1}, h_{n_2}) h_{n_1}^2 h_{n_2}^2 - \sum_{n_1+n_2=n_f} D(h_{n_1}, h_{n_2}) h_{n_1}^2 h_{n_2}^2 \\
&= h_{n_f+1}^2 \sum_{n_1+n_2=n_f+1} D(h_{n_1}, h_{n_2}) - h_{n_f}^2 \sum_{n_1+n_2=n_f} D(h_{n_1}, h_{n_2}) \\
&= h_{n_f}^2 \left(\frac{1}{4} \sum_{n_1+n_2=n_f+1} D(h_{n_1}, h_{n_2}) - \sum_{n_1+n_2=n_f} D(h_{n_1}, h_{n_2}) \right)
\end{aligned}$$

Summarising, we have for the error:

$$\begin{aligned}
U - V_{n_f}^c = {} & C_1(h_{n_f}) h_{n_f}^2 + C_2(h_{n_f}) h_{n_f}^2 \\
& + h_{n_f}^2 \left(\frac{1}{4} \sum_{n_1+n_2=n_f+1} D(h_{n_1}, h_{n_2}) - \sum_{n_1+n_2=n_f} D(h_{n_1}, h_{n_2}) \right)
\end{aligned}$$

Now the upper bound of this error equals:

$$\begin{aligned}
|U - V_{n_f}^c| &\leqslant |C_1(h_{n_f})| h_{n_f}^2 + |C_2(h_{n_f})| h_{n_f}^2 \\
&\quad + \left| h_{n_f}^2 \left(\frac{1}{4} \sum_{n_1+n_2=n_f+1} D(h_{n_1}, h_{n_2}) - \sum_{n_1+n_2=n_f} D(h_{n_1}, h_{n_2}) \right) \right| \\
&\leqslant |C_1(h_{n_f})| h_{n_f}^2 + C_2(h_{n_f}) | h_{n_f}^2 \\
&\quad + \frac{1}{4} h_{n_f}^2 \left| \sum_{n_1+n_2=n_f+1} D(h_{n_1}, h_{n_2}) \right| + h_{n_f}^2 \left| \sum_{n_1+n_2=n_f} D(h_{n_1}, h_{n_2}) \right| \\
&\leqslant \kappa h_{n_f}^2 + \kappa h_{n_f}^2 + \frac{1}{4} \kappa n_f h_{n_f}^2 + \kappa (n_f - 1) h_{n_f}^2 \\
&= \kappa h_{n_f}^2 \left(1 + \frac{5}{4} n_f \right) \\
&= \kappa h_{n_f}^2 \left(1 + \frac{5}{4} \log_2 h_{n_f}^{-1} \right).
\end{aligned}$$

All terms with $n_1, n_2 \neq n_f$ cancel out and the h_{n_f} terms remain. However, all terms which are dependent on the product of h_{n_1} and h_{n_2} accumulate leading to an additional $\log_2(h_{n_f}^{-1})$ term. This is characteristic for the sparse grid combination technique. Summarising, the error for the two-dimensional can be stated with the following order of accuracy [10, 9]:

$$U - V_{n_f}^c = \mathbf{O}\left(h_{n_f}^2 \log_2 h_{n_f}^{-1} \right). \tag{3.19}$$

This error expansion exists if the analytic solution is sufficiently smooth, which means that the solution should have bounded mixed derivatives [25, 20, 11].

3.3.3 Basic combination in higher dimensions

We continue with the general combination of solutions on $d-$dimensional coarse sub-grids to obtain a mimic of the full grid solution of a $d-$dimensional problem with 2^{n_f} grid points per coordinate. For the two-dimensional case, we combined two *layers* of solutions to obtain the combined sparse grid solution. It is obvious that for a general $d-$dimensional problem, the number of layers to combine a proper sparse grid solution is d. First of all, we generalise the multi-index to a d-dimensional version.

Definition 3.3.1. A multi-index $\mathcal{I}$ belonging to a $d-$dimensional sparse grid based on a mimic of a full grid with 2^{n_f} grid points per coordinate is a collection of sets of d values $(n_1, n_2, \ldots, n_d)$ such that:

- $1 \leqslant n_i \leqslant n_f$
- The sum of the numbers in each element for each sub-grid is equal to the layer number.

Furthermore, we define a layer as a set of sub-grids and we shall see that a layer and a multi-index are closely related to each other

Definition 3.3.2. A layer in a $d-$dimensional sparse grid solution represents a set of sub-grids. The total number of grid points in each sub-grid on a particular layer is the same. All sub-grids are ordered according to a multi-index. Each layer has a unique layer number that is based on the dimensionality, d, and the value of n_f. A $d-$dimensional combined sparse grid solution has d layers and the layer number l_j of the $j-$th layer reads:

$$l_j = n_f + d - j \qquad 1 \leqslant j \leqslant d.$$

Now the sparse grid combination for a $d-$dimensional grid is defined as:

Definition 3.3.3. The solution of a sparse grid problem on a $d-$dimensional grid based on a mimic of a full grid with 2^{n_f} grid points per coordinate is the combination of all solutions on all sub-grids provided by d layers. This combination is defined as:

$$V_{n_f}^c = \sum_{j=1}^{d} (-1)^{d-j} \binom{d-1}{j-1} \sum_{\sum_{i=1}^{d} n_i = l_j} V_{n_1, n_2, \ldots, n_d} \tag{3.20}$$

with l_j presented in Definition 3.3.2.

Remark 3.3.4. In equation (3.20), we recognise equation (3.17) by setting $d = 2$ and for the layer numbers $l_1 = n_f + d - 1$ and $l_2 = n_f + d - 2$.

Example 3.3.5. *Now the construction for a three-dimensional problem follows from equation (3.20). Choosing $d = 3$ we have:*

$$V_{n_f}^c = \sum_{l_1=n_f+2} V_{n_1,n_2,\ldots,n_d} - 2 \sum_{l_2=n_f+1} V_{n_1,n_2,\ldots,n_d} + \sum_{l_3=n_f} V_{n_1,n_2,\ldots,n_d} \quad (3.21)$$

If U is again the exact solution of the problem, then we have for the error:

$$\begin{aligned} U - V_{n_f}^c &= U - \left(\sum_{l_1=n_f+2} V_{n_1,n_2,n_3} - 2 \sum_{l_2=n_f+1} V_{n_1,n_2,n_3} + \sum_{l_3=n_f} V_{n_1,n_2,n_3} \right) \\ &= \sum_{l_1=n_f+2} (U - V_{n_1,n_2,n_3}) - 2 \sum_{l_2=n_f+1} (U - V_{n_1,n_2,n_3}) \\ &\quad + \sum_{l_3=n_f} (U - V_{n_1,n_2,n_3}) \end{aligned}$$

For the three-dimensional case, a similar error expansion as equation (3.18) exists for sufficiently smooth solutions:

$$\begin{aligned} U - V_{n_1,n_2,n_3} &= C_1(h_{n_1})h_1^2 + C_2(h_{n_2})h_{n_2}^2 + C_3(h_{n_3})h_{n_3}^2 \\ &\quad + D_1(h_{n_1}, h_{n_2})h_{n_1}^2 h_{n_2}^2 + D_2(h_{n_1}, h_{n_3})h_{n_1}^2 h_{n_3}^2 \\ &\quad + D_3(h_{n_2}, h_{n_3})h_{n_2}^2 h_{n_3}^2 + E(h_{n_1}, h_{n_2}, h_{n_3})h_{n_1}^2 h_{n_2}^2 h_{n_3}^2 \end{aligned} \quad (3.22)$$

Assume that the functions $C_k(h_{n_i}), D_k(h_{n_i}, h_{n_j})$ and $E(h_{n_1}, h_{n_2}, h_{n_3})$ with $k = 1, 2, 3$ are bounded above by a constant κ. The three sums are running over all combinations of n_1, n_2 and n_3 such that the sum of these n_i is equal to l_j:

$$\begin{aligned} U - V_{n_f}^c &= \sum_{n_1+n_2+n_3=n_f+2} (U - V_{n_1,n_2,n_3}) - 2 \sum_{n_1+n_2+n_3=n_f+1} (U - V_{n_1,n_2,n_3}) \\ &\quad + \sum_{n_1+n_2+n_3=n_f} (U - V_{n_1,n_2,n_3}) \end{aligned}$$

Using equation (3.22), we distinguish different error terms. First the error terms which are dependent on only one h_{n_1}:

$$
\begin{aligned}
&\sum_{n_1+n_2+n_3=n_f+2} C_1(h_{n_1})h_{n_1}^2 - 2\sum_{n_1+n_2+n_3=n_f+1} C_1(h_{n_1})h_{n_1}^2 \\
&+ \sum_{n_1+n_2+n_3=n_f} C_1(h_{n_1})h_{n_1}^2 \\
&= \sum_{n_1=1}^{n_f} (n_f - n_1 + 1)C_1(h_{n_1})h_{n_1}^2 - 2\sum_{n_1=1}^{n_f-1} (n_f - n_1)C_1(h_{n_1})h_{n_1}^2 \\
&+ \sum_{n_1=1}^{n_f-2} (n_f - n_1 - 1)C_1(h_{n_1})h_{n_1}^2 \\
&= \sum_{n_1=1}^{n_f-2} \left((n_f - n_1 + 1) - 2(n_f - n_1) + (n_f - n_1 - 1)\right) C_1(h_{n_1})h_{n_1}^2 \\
&+ \left((n_f - (n_f - 1) + 1) - 2(n_f - (n_f - 1))\right) C_1(h_{n_f-1})h_{n_f-1}^2 \\
&+ (n_f - n_f + 1)\, C_1(h_{n_f})h_{n_f}^2 \\
&= C_1(h_{n_f})h_{n_f}^2
\end{aligned}
$$

where we used

$$
\sum_{n_1+n_2+n_3=n_f+2} C_1(h_{n_1})h_{n_1}^2 = \sum_{n_1=1}^{n_f} (n_f - n_1 + 1)\, C_1(h_{n_1})h_{n_1}^2 .
$$

This relation holds because if $n_1 = n_f$, then $n_2 = n_3 = 1$ gives one possible combination. If for example $n_1 = 1$, then there are n_f combinations to make $n_2 + n_3 = n_f + 1$ as in Section 3.3.2. The same argumentation hold for the other terms. For C_2 and C_3, the same relations hold. Next the terms dependent on h_{n_1} and h_{n_2}. It is easy to see that:

$$
\sum_{n_1+n_2+n_3=n_f+2} D_1(h_{n_1}, h_{n_2})h_{n_1}^2 h_{n_2}^2 = \sum_{n_1+n_2 \leqslant n_f+1} D_1(h_{n_1}, h_{n_2})h_{n_1}^2 h_{n_2}^2 .
$$

Combining all D_1 terms, we find:

$$\begin{aligned}
&\sum_{n_1+n_2+n_3=n_f+2} D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 - 2\sum_{n_1+n_2+n_3=n_f+1} D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 \\
&+ \sum_{n_1+n_2+n_3=n_f} D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 \\
&= \sum_{n_1+n_2\leqslant n_f-1} (1-2+1)\, D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 \\
&+ \sum_{n_1+n_2=n_f} (1-2)\, D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 + \sum_{n_1+n_2=n_f+1} D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 \\
&= \sum_{n_1+n_2=n_f+1} D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 - \sum_{n_1+n_2=n_f} D_1(h_{n_1},h_{n_2})h_{n_1}^2h_{n_2}^2 \\
&= \left(\frac{1}{4}\sum_{n_1+n_2=n_f+1} D_1(h_{n_1},h_{n_2}) - \sum_{n_1+n_2=n_f} D_1(h_{n_1},h_{n_2})\right) h_{n_f}^2
\end{aligned}$$

where the last step is already shown in Section 3.3.2. Again similar results hold for D_2 and D_3. Finally for the $E-$terms we find:

$$\begin{aligned}
&\left|\sum_{n_1+n_2+n_3=n_f+2} E(h_{n_1},h_{n_2},h_{n_3})h_{n_1}^2h_{n_2}^2h_{n_3}^2\right| \\
\leqslant &\sum_{n_1+n_2+n_3=n_f+2} |E(h_{n_1},h_{n_2},h_{n_3})|\, h_{n_1}^2h_{n_2}^2h_{n_3}^2 \\
= &h_{n_f+2}^2 \sum_{n_1+n_2+n_3=n_f+2} |E(h_{n_1},h_{n_2},h_{n_3})| \\
\leqslant &\kappa\frac{n_f(n_f+1)}{2}h_{n_f+2}^2
\end{aligned}$$

The last expression follows from the number of summations in the combination $n_1+n_2+n_3=n_f+2$, which is equal to $\frac{1}{2}n_f(n_f+1)$. Now combining

all terms and taking the upper bound of the error we obtain:

$$
\begin{aligned}
|U - V^c_{n_f}| &\leqslant |C_1(h_{n_f})|h^2_{n_f} + |C_2(h_{n_f})|h^2_{n_f} + |C_3(h_{n_f})|h^2_{n_f} \\
&+ \left| \left(\frac{1}{4} \sum_{n_1+n_2=n_f+1} D_1(h_{n_1}, h_{n_2}) - \sum_{n_1+n_2=n_f} D_1(h_{n_1}, h_{n_2}) \right) h^2_{n_f} \right| \\
&+ \left| \left(\frac{1}{4} \sum_{n_1+n_3=n_f+1} D_2(h_{n_1}, h_{n_3}) - \sum_{n_1+n_3=n_f} D_2(h_{n_1}, h_{n_3}) \right) h^2_{n_f} \right| \\
&+ \left| \left(\frac{1}{4} \sum_{n_2+n_3=n_f+1} D_3(h_{n_2}, h_{n_3}) - \sum_{n_2+n_3=n_f} D_3(h_{n_2}, h_{n_3}) \right) h^2_{n_f} \right| \\
&+ \left| \kappa \frac{n_f(n_f+1)}{2} h^2_{n_f+2} - 2\kappa \frac{n_f(n_f-1)}{2} h^2_{n_f+1} + \kappa \frac{(n_f-2)(n_f-1)}{2} h^2_{n_f} \right| \\
&\leqslant 3\kappa h^2_{n_f} + 3\kappa h^2_{n_f} \left(\frac{1}{4} n_f + n_f - 1 \right) \\
&+ \kappa h^2_{n_f} \left(\frac{1}{16} \frac{n_f(n_f+1)}{2} + \frac{2}{4} \frac{n_f(n_f-1)}{2} + \frac{(n_f-2)(n_f-1)}{2} \right) \\
&= \kappa h^2_{n_f} \left(1 + \frac{65}{32} \log_2 h^{-1}_{n_f} + \frac{25}{32} \left(\log_2 h^{-1}_{n_f} \right)^2 \right) \approx \mathbf{O}\left(h^2_{n_f} \left(\log_2 h^{-1}_{n_f} \right)^2 \right).
\end{aligned}
$$

Only terms of the highest order are taken into account and we see that we have a square of the extra $\log_2$ term. This square comes from the combination of the three different values of h_{n_1}, h_{n_2} and h_{n_3}. It is obvious that a power of $d-1$ occurs in $d-$dimensional problems, by use of the combination of all different values of h_{n_1}. Therefore the order of the accuracy of a standard sparse grid combination technique for a $d-$dimensional problem with a sufficiently smooth solution reads:

$$
|U - V^c_{n_f}| = \mathbf{O}\left(h^2_{n_f} \left(\log_2 h^{-1}_{n_f} \right)^{d-1} \right). \tag{3.23}
$$

Similar to the two-dimensional case, the meaning of *sufficiently smooth solution* that it should have a bounded mixed derivative. Option pricing problems have a non-differentiability in the contract function. Due to this non-differentiability, the solution will have mixed derivatives, which are not bounded. We will show that, with the help of coordinate transformations in the option pricing problem (see Section 3.5), this effect will be reduced and accurate solutions can be obtained.

With equation (3.20) and (3.23), we have the ingredients of the sparse grid method for a $d-$dimensional problem. We conclude with the computation of the total number of sub-grids needed for a single sparse grid computation. Per layer, the number of sub-grids reads:

$$
Z_j = \binom{n_f + d - j - 1}{d-1} \tag{3.24}
$$

and the total number of sub-problems to solve in a combined sparse grid problem reads:

$$\begin{aligned} Z = \sum_{j=1}^{d} Z_j &= \sum_{j=1}^{d} \binom{n_f + d - j - 1}{d-1} \\ &= \frac{n_f + d - 1}{d} \binom{n_f + d - 2}{d-1} - \frac{n_f - 1}{d} \binom{n_f - 2}{d} \end{aligned} \tag{3.25}$$

Example 3.3.6. *For the two-dimensional case, we see that with $n_f = 4$ and $d = 2$ we have $Z_1 = 4$ and this gives the multi-index $[(4,1),(3,2),(2,3),(1,4)]$ corresponding to the sub-grids $[(16 \times 2),(8 \times 4),(4 \times 8),(2 \times 16)]$ and we find $Z_1 = 3$ with the multi-index $[(3,1),(2,2),(1,3)]$ corresponding to the sub-grids $[(8 \times 2),(4 \times 4),(2 \times 8)]$. We see that $Z = 7$ according to equation (3.25).*

The sub-grids coming from the multi-index belonging to the $j-$th layer of a $d-$dimensional sparse grid have, according to the layer number in Definition 3.3.2,

$$\mathcal{N}_j = 2^{n_f + d - j} \tag{3.26}$$

points per sub-grid. Finally, the total complexity - i.e. the total number of unknowns - reads:

$$\mathcal{N} = \sum_{j=1}^{d} \mathcal{N}_j Z_j = \sum_{j=1}^{d} 2^{n_f + d - j} \binom{n_f + d - j - 1}{d-1} \tag{3.27}$$

Example 3.3.7. *Consider a seven-dimensional problem with $n_f = 10$. Then the full grid problem should have 2^{70} unknowns. In terms of computer memory this is more than 2^{40} GB for storing the solution vector. This is impossible, even on a hard-disc drive. However, the total sparse grid complexity has approximately 2^{29} unknowns and this can be handled by many computer systems, since each sub-problem need not be computed at the same time on the same machine. The maximum problem size is 2^{16} unknowns.*

Remark 3.3.8. A final remark is about a special case when $n_f < d$. Then following the definition of the multi-index, there is at least one empty multi-index. Otherwise, there is no combination possible such that $\sum_{i=1}^{d} n_i = d - d'$, $d' > 0$ with the restriction that $n_i \geqslant 1$. This case is known as an incomplete sparse grid solution.

3.3.4 Combination technique for general grids

The one-sided difference equation defined in equation (2.11) requires at least three adjacent grid points. Furthermore the fourth order scheme (2.13) requires five adjacent points. Therefore, the minimum number of grid points

for each coordinate should be higher than two for the sparse grid solution. The consequence is a modification in the combination technique for the sparse grid case. Another reason to modify the combination technique, is the construction of grids with different numbers of grid points per coordinate, so-called *non-equidistant grids*, for example a four-dimensional grid with $(256 \times 64 \times 32 \times 16)$ points. The reason to mimic a full grid with different numbers of grid points per coordinate comes from the coordinate transformation to be developed in Section 3.5. With this coordinate transformation, it may be possible to choose fewer grid points for certain coordinates.

The construction of the multi-indices does not change, but each sub-grid is multiplied with constants c_i, to mimic the original full grid. Hence each element from a multi-index represents a sub-grid with $c_i 2^{n_i}$ grid points per coordinate. Now the basis is defined as the minimum number of grid points in each sub-grid. With the basis and the coefficients c_i together it is possible to construct a mimic of non-equidistant grids.

Definition 3.3.9. The basis b, of a sparse grid combination technique, represents the minimum number of grid points per coordinate, which is 2^b. If the basis is equal to one, then we deal with the standard sparse combination technique.

A different basis than $b = 1$ leads to a different sparse grid combination, if a basis is applied as then the sparse grid technique combination is a mimic of the full grid which has sizes 2^{n_s} with n_s defined as:

$$n_s = n_f - b + 1. \tag{3.28}$$

To obtain the proper sub-grids, each sub-grid coming from the multi-index is multiplied by $c_i = 2^{b-1}$.

Example 3.3.10. *Consider a sparse grid combination to mimic a grid of* 64×64. *Then we have* $n_f = 6$. *Suppose, we require at least eight points per coordinate and thus we have* $b = 3$ *and by equation (3.28)* $n_s = 4$. *The multi-indices and sub-grids are now:*

$$\begin{aligned} [(4,1),(3,2),(2,3),(1,4)] &\longrightarrow [(64,8),(32,16),(16,32),(8,64)] \\ [(3,1),(2,2),(1,3)] &\longrightarrow [(32,8),(16,16),(8,32)] \end{aligned}$$

and we see that these multi-indices are the same ones as in Example 3.3.6, but the corresponding sub-grids are different.

For the error expansion in two dimensions, we now have:

$$U - V_{n_f}^c = \sum_{(n_1,n_2)\in\mathcal{I}:l=n_s+1} (U - V_{n_1,n_2}) - \sum_{(n_1,n_2)\in\mathcal{I}:l=n_s} (U - V_{n_1,n_2}),$$

but now V_{n_1,n_2} is a solution on a sub-grid with mesh sizes $\widehat{h}_{n_i} = (c_i 2^{n_i})^{-1} = 2^{-b+1}2^{-n_i} = 2^{-b+1}h_{n_i}$. With the error expansion in equation (3.18) we have:

$$\begin{aligned}
U - V^c_{n_f} = &\sum_{n_1+n_2=n_s+1} \left(C_1(\widehat{h}_{n_1})\widehat{h}^2_{n_1} + C_2(\widehat{h}_{n_2})\widehat{h}^2_{n_2} + D(\widehat{h}_{n_1},\widehat{h}_{n_2})\widehat{h}^2_{n_1}\widehat{h}^2_{n_2} \right) \\
&- \sum_{n_1+n_2=n_s} \left(C_1(\widehat{h}_{n_1})\widehat{h}^2_{n_1} + C_2(\widehat{h}_{n_2})\widehat{h}^2_{n_2} + D(\widehat{h}_{n_1},\widehat{h}_{n_2})\widehat{h}^2_{n_1}\widehat{h}^2_{n_2} \right) \\
&= \sum_{n_1=1}^{n_s} C_1(\widehat{h}_{n_1})\widehat{h}^2_{n_1} - \sum_{n_1=b}^{n_s-1} C_1(\widehat{h}_{n_1})\widehat{h}^2_{n_1} \\
&+ \sum_{n_2=1}^{n_s} C_2(\widehat{h}_{n_2})\widehat{h}^2_{n_2} - \sum_{n_2=b}^{n_s-1} C_2(\widehat{h}_{n_2})\widehat{h}^2_{n_2} \\
&+ \sum_{n_1+n_2=n_s+1} D(\widehat{h}_{n_1},\widehat{h}_{n_2})\widehat{h}^2_{n_1}\widehat{h}^2_{n_2} - \sum_{n_1+n_2=n_s} D(\widehat{h}_{n_1},\widehat{h}_{n_2})\widehat{h}^2_{n_1}\widehat{h}^2_{n_2}
\end{aligned}$$

Because $\widehat{h}_{n_s} = 2^{-b+1}h_{n_s} = h_{n_f}$ according to equation (3.28) and $\widehat{h}_{n_1}\widehat{h}_{n_2} = 2^{-b+1}h_{n_1}2^{-b+1}h_{n_2}$ the error reads:

$$\begin{aligned}
U - V^c_{n_f} &= C_1(\widehat{h}_{n_s})h^2_{n_f} + C_2(\widehat{h}_{n_s})h^2_{n_f} + \sum_{n_1+n_2=n_s+1} D(\widehat{h}_{n_1},\widehat{h}_{n_2})2^{-4b+4}h^2_{n_s+1} \\
&\quad - \sum_{n_1+n_2=n_s} D(\widehat{h}_{n_1},\widehat{h}_{n_2})2^{-4b+4}h^2_{n_s} \\
&= C_1(\widehat{h}_{n_s})h^2_{n_f} + C_2(\widehat{h}_{n_s})h^2_{n_f} \\
&\quad + 2^{-2b+2}h^2_{n_f}\left(\frac{1}{4} \sum_{n_1+n_2=n_s+1} D(\widehat{h}_{n_1},\widehat{h}_{n_2}) - \sum_{n_1+n_2=n_s} D(\widehat{h}_{n_1},\widehat{h}_{n_2}) \right)
\end{aligned}$$

We see that only the number of combinations in the mixed term is different and as it is multiplied by a different factor. Following the same procedure for the error expansion as in Section 3.3.2, we have:

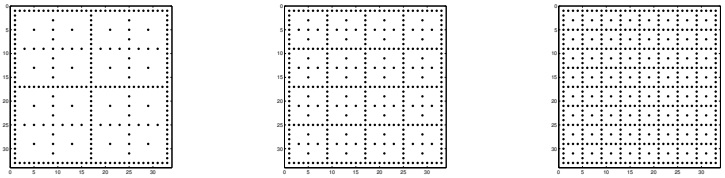
$$\begin{aligned}
|U - V^c_{n_f}| &\leqslant 2\kappa h^2_{n_f} + 2^{-2b+2}\kappa h^2_{n_f}\left(\frac{1}{4}n_s + (n_s - 1) \right) \\
&= 2\kappa h^2_{n_f} + 2^{-2b+2}\kappa h^2_{n_f}\left(\frac{5}{4}n_s - 1 \right) \\
&= 2\kappa h^2_{n_f} + 2^{-2b+2}\kappa h^2_{n_f}\left(\frac{5}{4}(n_f - b) + \frac{1}{4} \right) \\
&\leqslant \kappa h^2_{n_f}\left(2 + 5 \cdot 2^{-2b}n_f \right) \approx \mathbf{O}\left(h^2_{n_f}\log_2\left(h^{-1}_{n_f} \right) \right).
\end{aligned}$$

The approximation in the last step is dependent on the value of b. If b is larger, then the influence of logarithmic term is weaker and the convergence ratio will tend to the full grid convergence ratio. For the three-dimensional

case, we can find that:

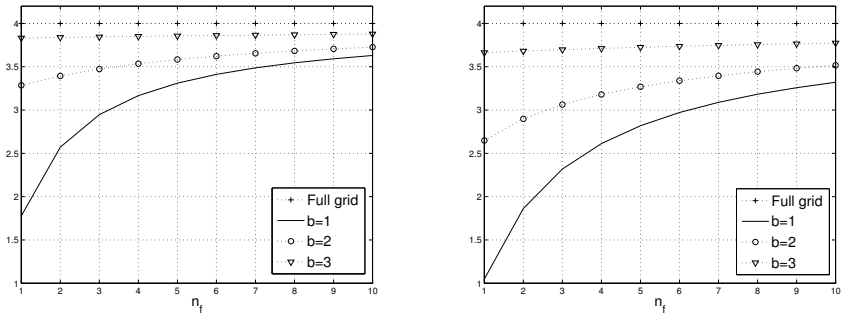
$$
\begin{aligned}
|U - V_{n_f}^c| \leqslant\ & 3\kappa h_{n_f}^2 + 3\kappa h_{n_f}^2 2^{-2b+2}\left(\frac{5}{4}(n_f - b)\right) \\
& + 2^{-(4b+4)}\kappa h_{n_f}^2 \frac{1}{16}\frac{(n_f - b + 1)(n_f - b + 2)}{2} \\
& + 2^{-(4b+4)}\kappa h_{n_f}^2 \frac{(n_f - b + 1)(n_f - b)}{4} \\
& + 2^{-(4b+4)}\kappa h_{n_f}^2 \frac{(n_f - b - 1)(n_f - b)}{2}.
\end{aligned}
$$

Three different two-dimensional sparse grids are presented in Figure 3.2 and the effect on the theoretical convergence ratio is presented in Figure 3.3.



(a) b=1 (b) b=2 (c) b=3

Figure 3.2: Construction of a 2D sparse grid: combined solution



(a) d=2 (b) d=3

Figure 3.3: Theoretical error convergence of sparse grids with a different value of b.

Now, the non-equidistant grids follows from choosing c_i differently. The basis will be defined by $b = \log_2(\min c_i) + 1$. The sparse grid combination is again a mimic of a full grid with 2^{n_s} grid points per coordinate and n_s is defined in equation (3.28), whereby $n_f = \log_2 \min N_i$. The error expansion of these grids can be approximated by the same expansion.

Example 3.3.11. *Consider a two-dimensional grid with* (16×64) *grid points. Now we have* $n_f = 4$ *and* $c_1 = 1$ *and* $c_2 = 4$*. So, it follows that* $b = 1$ *and* $n_s = 4$*. Hence the multi-indices and set of sub-grids are:*

$$[(4,1),(3,2),(2,3),(1,4)] \Rightarrow [(16,8),(8,16),(4,32),(2,64)]$$
$$[(3,1),(2,2),(1,3)] \Rightarrow [(8,8),(4,16),(2,32)].$$

and again we see that these multi-indices are the same as in Example 3.3.6 or Example 3.3.10, but the corresponding sub-grids are different.

The next example is a combination of the two different cases.

Example 3.3.12. *Consider a two-dimensional grid with sizes* (256×64) *and furthermore we require at least eight points per coordinate, so* $b = 3$*. We already have* $c_1 = 4$ *and* $c_2 = 1$*, but with the extra requirement, we simply multiply the basis with the coefficients:* $c_1 = 4 \cdot 2^{b-1} = 16$ *and* $c_2 = 4$*. Then it follows that* $n_s = 4$ *and therefore we see for the multi-index:*

$$[(4,1),(3,2),(2,3),(1,4)] \Rightarrow [(256,8),(128,16),(64,32),(32,64)]$$
$$[(3,1),(2,2),(1,3)] \Rightarrow [(128,8),(64,16),(32,32)].$$

Again, we have different sub-grids from the same multi-indices.

Remark 3.3.13. If $c_1 = c_2 = 5$, then the basis cannot be defined, but we can construct the multi-index in a similar way. For example, if $n_s = 4$ (note that n_s can be prescribed as well), we have:

$$[(4,1),(3,2),(2,3),(1,4)] \Rightarrow [(80,10),(40,20),(20,40),(10,80)]$$
$$[(3,1),(2,2),(1,3)] \Rightarrow [(80,10),(20,20),(10,40)].$$

and we see that with proper choices of c_i it is possible to construct general grids. In this thesis, we do not use this and we only construct (non-)equidistant grids with numbers of grid points that are a power of two.

If the basis of the sparse grid combination is known, then by using equation (3.28) and (3.24), the number of sub-grids in a layer j reads:

$$Z_j = \binom{n_f + d - b - j}{d - 1}. \tag{3.29}$$

The complexity in a sub-grid or number of unknowns in the j-th layer reads:

$$\mathcal{N}_j = 2^{n_s + d - j} \prod_{i=1}^{d} c_i. \tag{3.30}$$

The total complexity reads:

$$\mathcal{N} = \sum_{j=1}^{d} \mathcal{N}_j Z_j = \prod_{i=1}^{d} c_i \sum_{j=1}^{d} 2^{n_s + d - j} \binom{n_f + d - b - j}{d - 1}. \tag{3.31}$$

3.3.5 Combinations with an extra layer

Equation (3.20) is referred as the standard combination technique. However we will also present an experiment with a different type of combination technique, where the coefficients come from the backward difference formulae:

$$\begin{aligned} V_{n_f}^c = \frac{3}{2} \sum_{\sum_{i=1}^d n_i = n_f+2} V_{n_1,n_2} - 2 \sum_{\sum_{i=1}^d n_i = n_f+1} V_{n_1,n_2} \\ + \frac{1}{2} \sum_{\sum_{i=1}^d n_i = n_f} V_{n_1,n_2}, \end{aligned} \tag{3.32}$$

with the coefficients of a BDF2 scheme [26]. With vectors $\mathbf{q} = \frac{1}{2}[3, -4, 1]$ and:

$$\mathbf{w} = \left[\binom{d-2}{0}, \binom{d-2}{1}, \ldots, \binom{d-2}{d-2} \right], \tag{3.33}$$

the d-dimensional combination technique in this setting is written as (3.20):

$$V_{n_f}^c = \sum_{j=1}^{d} (-1)^j a_j \sum_{l=n+j} V_{n_1,n_2,\ldots,n_d} \tag{3.34}$$

with a_j the j-th element of $\mathbf{a} = \mathbf{q} \star \mathbf{w}$, where $\star$ is the convolution operator. Note that this method uses one additional finer layer compared to the basic sparse grid method (3.20). With the splitting of the error as given in (3.18), the error for the two-dimensional Poisson case can be bounded by:

$$|U - V_{n_f}^c| \leq \frac{35}{32} h_{n_f}^2 \log_2(h_{n_f}^{-1}).$$

3.3.6 Reisinger's sparse grid method

The third combination technique evaluated is based on the basic combination technique combined with a multivariate extrapolation [40, 41]. The combination equation then reads:

$$\begin{aligned} V_{n_f}^c = \sum_{\ell=n_f}^{n_f+2d-1} \sum_{l=\ell} V_I \left(\sum_{j=\max\{0,\ell-n_f-d\}}^{j=\min\{\ell-n_f,d-1\}} a_j \alpha_{\ell-n_f-d} \binom{N(I)}{\ell-n_f-j} \right), \\ a_j = (-1)^{d-1-j}, \quad \alpha_j = (-4)^j (-3)^{-d}. \end{aligned} \tag{3.35}$$

$N(I)$ denotes the number of nonzero elements in multi-index $\mathcal{I}$. (The elements of the multi-indices for this type of combination can be zero.) It can be proven that this method is **fourth order accurate** and the absolute error for the Poisson equation reads [40]:

$$|U - V_{n_f}^c| \leq \frac{10}{3} \frac{K}{(d-1)!} \left(\frac{85}{24} \right)^{d-1} \left(\log_2(h_{n_f}^{-1}) + 2d - 1 \right)^{d-1} h_{n_f}^4. \tag{3.36}$$

An error splitting as in (3.18) does not guarantee fourth order accuracy when employing fourth order finite differences within the basic sparse grid technique [40], but the combination in equation (3.35) does..

A final remark on the three variants of the sparse grid technique is on the number of underlying grids, and thus on the method's complexity. For a d-dimensional problem the basic sparse grid method uses $d-1$ "layers" of grids, the BDF sparse grid method employs d-, and Reisinger's method $2d-1$-layers.

3.4 Sparse grid test experiments

It is now reasonable to evaluate the behaviour of the sparse grid combination technique with some test experiments. The first two experiments are based on the multi-dimensional Poisson equation and the time-dependent heat equation respectively. For both the equations we use a test function so that the solution is already prescribed and serves for a comparison with the numerical solution. The Poisson equation is solved for up to $d = 8$ and the heat equation is solved up to $d = 5$ plus the additional time dimension. In Section 3.4.3, we show that the Black-Scholes equation can be transformed to the heat equation [31] and we evaluate the sparse grid technique for the transformed Black-Scholes equation.

3.4.1 Laplace equation

The sparse grid technique is used for the $d-$dimensional Poisson equation. The test function chosen here for this sparse grid stationary experiment reads:

$$V(S_1, S_2, \ldots, S_d) = \prod_{i=1}^{d} e^{S_i^2} = \exp\left(\sum_{i=1}^{d} S_i^2\right), \tag{3.37}$$

with $\Omega = [0,1]^d$. The approximation is done for $2 \leqslant d \leqslant 8$ with a mimic of the grid with $n_f = 10$. The mimic of the full grid is done with $c_i = 1$ and $b = 1$. The discretisation technique in Section 3.2 is used for each sub-grid. Each discretisation leads to a linear system, that is iteratively solved by the Bi-CGSTAB method with multigrid as a preconditioner. We chose to use the Bi-CGSTAB method, because the problems we discuss in the rest of the thesis have non-symmetric matrices and the Bi-CGSTAB methods perform satisfactorily for the computations. The iterative solver is constructed in another project. Details can be found in [6] and [5]. The boundary conditions for the $d-$dimensional problem are directly computed from the test function and therefore we have Dirichlet boundary conditions.

In a full grid setting the maximum problem would have a size of $1024^8 = 2^{80}$, i.e., 2^{53} GB of memory, which is - of course - immensely huge. The maximum size of an 8D problem in the sparse grid setting chosen here is

$1024 \times 2^7 = 2^{17}$, which is only 1 MB. The solution at the central point in the domain $x_i = 0.5$ is computed here. The results are described in detail for $d = 2$ and $d = 8$ in Table 3.1 and the error and convergence for all values of d are plotted in Figure 3.4. In the table, the time indicated is the total computational time for the computation of the solutions on all sub-grids (see equation (3.25)) and the interpolation to the central point. The number of problems Z is as in equation (3.25) and the theoretical convergence $Th.Conv$ from equation (3.23).

Table 3.1: Time-independent experiments of problem (3.37) using sparse grids. TOP: The two-dimensional case. BOTTOM: The eight-dimensional case. Column one gives n_f, corresponding with the mimic of the full grid.

n_f	Value	Error	Conv	Time	Z	Th. Conv
d=2						
4	1.65	$5.52 \cdot 10^{-3}$	3.05	0.04	7	3.00
5	1.65	$1.72 \cdot 10^{-3}$	3.21	0.07	9	3.20
6	1.65	$5.16 \cdot 10^{-4}$	3.34	0.10	11	3.33
7	1.65	$1.50 \cdot 10^{-4}$	3.43	0.12	13	3.43
8	1.65	$4.30 \cdot 10^{-5}$	3.50	0.16	15	3.50
9	1.65	$1.21 \cdot 10^{-5}$	3.55	0.25	17	3.56
10	1.65	$3.36 \cdot 10^{-6}$	3.60	0.33	19	3.60
d=8						
4	7.63	$2.43 \cdot 10^{-1}$	1.50	18.51	165	0.53
5	7.54	$1.48 \cdot 10^{-1}$	1.64	111.07	495	0.84
6	7.47	$8.33 \cdot 10^{-2}$	1.77	578.55	1287	1.12
7	7.43	$4.40 \cdot 10^{-2}$	1.89	2404.47	3003	1.36
8	7.41	$2.20 \cdot 10^{-2}$	2.00	9067.30	6435	1.57
9	7.40	$1.04 \cdot 10^{-2}$	2.10	32925.66	12869	1.75
10	7.39	$4.76 \cdot 10^{-3}$	2.19	106826.59	24301	1.91

The table and the figures show the dependence of the number of dimensions in the convergence according to the theoretical convergence ratio in equation (3.23). Although the theoretical convergence of the sparse grid method is low when d is high at small numbers of n_f, the convergence in this test experiment is reasonable. A reason may be the smoothness of the analytic solution. The computational time shown in Table 3.1 is the total time required to solve the Z sub-problems. For the eight-dimensional problem, we have eight layers. In the case $n_f = 10$, the complexity ranges from 2^{10} to 2^{17} unknowns. The latter number of unknowns occur in the first layer and this layer requires the biggest number of sub-problems: $Z_1 = 11440$. Hence, the largest part of the computational time is used to solve the problems in

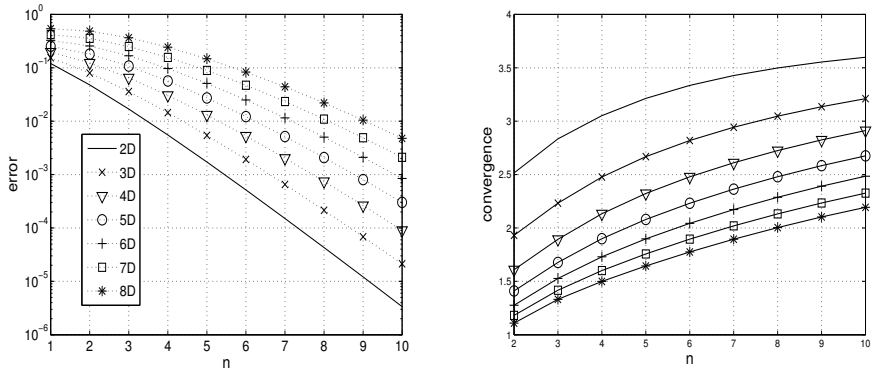


Figure 3.4: LEFT: Decay of the error $|U - V^c_{n_f}|$, with U the exact solution (3.37). RIGHT: Convergence of the error in the left picture.

this layer. However, if the solution of the sub-problems would be parallelised over ten processors, then the total computational time should be 2.8 hours. The parallelisation of the sparse grid method will be discussed in Section 4.3.

Now, we employ the different combination techniques, as defined in Section 3.3.5 and 3.3.6, with a different prescribed solution.

$$V(S_1, S_2, \ldots, S_d) = \exp \prod_{i=1}^{d} S_i \tag{3.38}$$

Again, second order central differences are used for the discretisation of the Poisson equation. We only perform a two-dimensional experiment, since the two-dimensional case gives sufficient information. The numerical solution is computed at the central point of the grid: $(\frac{1}{2}, \frac{1}{2})$. In Table 3.2 the error and the convergence ratio are presented for the full grid and the three different sparse grid combination techniques. We see that the BDF2 combination technique shows an irregular convergence pattern, but shows smaller errors. The method developed by Reisinger performs very satisfactorily. For higher-dimensional problems, however, these techniques with extra layers of sub-problems become too expensive, especially since in computational finance we are not interested in extremely high accuracy.

3.4.2 Multi-dimensional heat equation

For the time-dependent test case, we choose as the analytic solution,

$$V(t, S_1, S_2, \ldots, S_d) = e^t \prod_{i=1}^{d} e^{\frac{S_i}{\sqrt{d}}} = \exp\left(t + \frac{1}{\sqrt{d}} \sum_{i=1}^{d} S_i\right), \tag{3.39}$$

Table 3.2: 2D Poisson test problem. Second order discretisation: error in point $(\frac{1}{2},\frac{1}{2})$ with $h_{n_f} = 2^{-n_f}$ in the full grid case. Three different combination techniques.

	Full		(3.20)	
n_f	error	Conv	error	Conv
1	$2.1 \cdot 10^{-4}$		$2.1 \cdot 10^{-4}$	
2	$1.1 \cdot 10^{-4}$	1.9	$2.6 \cdot 10^{-4}$	0.8
3	$3.2 \cdot 10^{-5}$	3.4	$1.3 \cdot 10^{-4}$	1.9
4	$8.4 \cdot 10^{-6}$	3.8	$5.3 \cdot 10^{-5}$	2.5
5	$2.1 \cdot 10^{-6}$	4.0	$1.8 \cdot 10^{-5}$	2.9
6	$5.3 \cdot 10^{-7}$	4.0	$5.8 \cdot 10^{-6}$	3.1
7	$1.3 \cdot 10^{-7}$	4.0	$1.8 \cdot 10^{-6}$	3.3
	(3.34)		(3.35)	
n_f	error	Conv	error	Conv
1	$2.8 \cdot 10^{-4}$		$3.3 \cdot 10^{-5}$	
2	$7.2 \cdot 10^{-5}$	3.9	$3.5 \cdot 10^{-6}$	9.4
3	$1.2 \cdot 10^{-5}$	5.9	$3.3 \cdot 10^{-7}$	10.6
4	$9.3 \cdot 10^{-7}$	13.1	$2.9 \cdot 10^{-8}$	11.2
5	$3.7 \cdot 10^{-7}$	2.5	$2.5 \cdot 10^{-9}$	11.7
6	$2.5 \cdot 10^{-7}$	1.5	$2.0 \cdot 10^{-10}$	12.3
7	$1.8 \cdot 10^{-7}$	2.4	$1.6 \cdot 10^{-11}$	12.4

with $\Omega = [0,1]^d$, $t \geqslant 0$. The approximation is done for $2 \leqslant d \leqslant 5$ with $n_f = 10$ as the maximum number per coordinate in the sparse grid combination technique and $c_i = 1$. The solution of the central point in the domain $x_i = 0.5$ is computed at time $t = 0.1$. The number of time steps used is fixed at 400. Boundary as well as the initial conditions are obtained from the prescribed solution.

The *grid convergence* results are described in detail for $d = 2$ and $d = 5$ in Table 3.3 and the errors and convergence for all values of d are plotted in Figure 3.5. In the table, again "time" is the total computational time for the complete sparse grid solution including the interpolation and time integration. The number of problems is as in (3.25) and the theoretical convergence from equation (3.23).

Again, the results in the table and figures show the dependence on the number of dimensions in the error. The total time for the $d = 5$ computation is relatively small per sub-grid and the accuracy results are also satisfactory for the time-dependent case.

Table 3.3: Time-dependent experiments with solution (3.39) using sparse grids. TOP: is the two-dimensional case. BOTTOM: five dimensional case. Column one gives the maximum number of cells per coordinate.

	$X = 0.5^d$			Control		
Grid	Value	Error	Conv	Time	#probl	Th. Conv
d=2						
8	2.24	$1.37 \cdot 10^{-4}$	2.96	2.14	5	2.67
16	2.24	$4.36 \cdot 10^{-5}$	3.14	5.67	7	3.00
32	2.24	$1.33 \cdot 10^{-5}$	3.28	10.87	9	3.20
64	2.24	$3.93 \cdot 10^{-6}$	3.38	19.12	11	3.33
128	2.24	$1.14 \cdot 10^{-6}$	3.46	31.53	13	3.43
256	2.24	$3.23 \cdot 10^{-7}$	3.52	49.01	15	3.50
512	2.24	$9.07 \cdot 10^{-8}$	3.56	70.35	17	3.56
1024	2.24	$2.56 \cdot 10^{-8}$	3.54	99.82	19	3.60
d=5						
8	3.38	$9.67 \cdot 10^{-5}$	1.98	9.02	21	0.79
16	3.38	$4.45 \cdot 10^{-5}$	2.17	36.99	56	1.27
32	3.38	$1.92 \cdot 10^{-5}$	2.32	129.77	126	1.64
64	3.38	$7.83 \cdot 10^{-6}$	2.45	436.85	251	1.93
128	3.38	$3.06 \cdot 10^{-6}$	2.56	1288.44	456	2.16
256	3.38	$1.15 \cdot 10^{-6}$	2.65	3658.62	771	2.34
512	3.38	$4.24 \cdot 10^{-7}$	2.72	11297.58	1231	2.50
1024	3.38	$1.50 \cdot 10^{-7}$	2.83	32266.03	1876	2.62

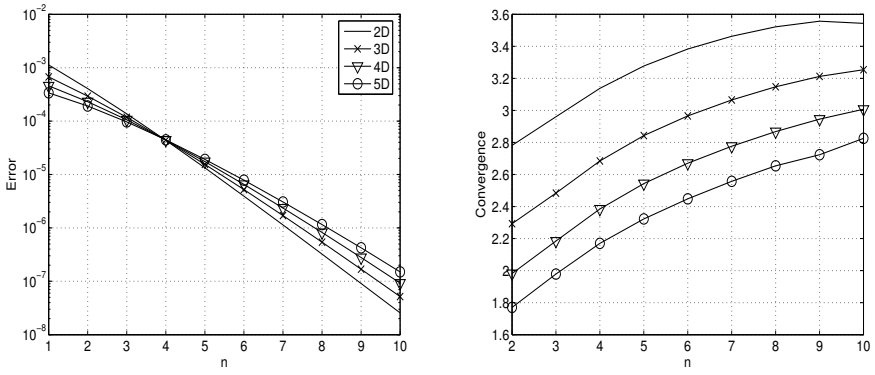


Figure 3.5: LEFT: Decay of the error $|U - V_{n_s}^c|$, U from (3.39). RIGHT: Convergence of the error in the left picture.

3.4.3 Black-Scholes equation

The results of the solution of the heat equation in the previous section show that the sparse grid method is a satisfactory alternative to reach reasonable dimensionality for sufficiently smooth solutions. Moreover, the Black-Scholes equation (1.9) can be transformed into a heat equation by using the coordinates

$$y_i = \frac{1}{\sigma_i}\log S_i + \frac{1}{\sigma_i}(r - \delta_i - \frac{1}{2}\sigma_i^2)(T-t)\ ,\ \tau = T - t\ ,\ V' = Ve^{-r(T-t)}$$

Then it follows that the transformed equation reads:

$$\frac{\partial V'}{\partial \tau} = \frac{1}{2}\sum_{i=1}^{d}\rho_{ij}\frac{\partial^2 V'}{\partial y_i \partial y_j}$$

This can be written in matrix notation [31]:

$$\frac{\partial V'}{\partial \tau} = \frac{1}{2}\nabla^T \mathbf{\Sigma}\nabla V',$$

where $\mathbf{\Sigma}$ is a symmetric positive semidefinite matrix for which $\mathbf{\Sigma}_{ij} = \sigma_i\sigma_j\rho_{ij}$. Then there exists an orthogonal transformation:

$$\mathbf{Q\Sigma Q}^T = diag(\lambda_i) = \mathbf{\Lambda},$$

with $\mathbf{\Lambda}$ the diagonal matrix with the eigenvalues of $\mathbf{\Sigma}$. The transformation reads $\mathbf{x} = \mathbf{Qy}$ and it follows that the d-dimensional Black Scholes equation can be reduced to the d-dimensional heat equation:

$$\frac{\partial V'}{\partial \tau} = \frac{1}{2}\sum_{i=1}^{d}\lambda_i\frac{\partial^2 V'}{\partial x_i^2}. \tag{3.40}$$

Advantages of this transformation are the constant coefficients, the disappearance of the first order terms and mixed derivatives, diagonal diffusion and the availability of a Green's function to change the partial differential equation to an integral problem [57, 31, 40, 20]. But there are also disadvantages. Boundaries are now at $\pm\infty$. This is not useful for numerical methods. The contract function, $\sum_{i=1}^{d} S_i - K$, does not lie on a grid line [40].

Equation (3.40) is now used as the last test experiment for the sparse grid technique. The contract function of a basket call changes into the initial condition:

$$V'(0,\boldsymbol{x}) = \max\left(\sum_{i=1}^{d} w_i e^{\sigma_i x_i} - K, 0\right), \tag{3.41}$$

This payoff function has a non-differentiability in the hyperplane where the transformed basket sum equals the strike price. This will turn out to be

problematic for the sparse grid solution of this problem. The option parameters chosen are: $K = 40, r = 0.06, T = 1, \sigma_i = 0.25, \delta_i = 0.04, \rho_{ij} = 0.25$ and $w_i = d^{-1}$.

The price of the option is computed for $2 \leqslant d \leqslant 5$ where d represents the number of assets in the basket. The outer domain boundaries are placed at $S = 5K$ to mimic infinity in (1.9). In the $x-$domain, this means that $\Omega = [-\sigma_i^{-1} \log 5, \sigma_i^{-1} \log 5]^d$. The sparse grid approximation contains grids with at maximum 1024 cells along a coordinate and with 128 time steps. The results of the experiments are summarised in Table 3.4. The errors in this table are computed as:

$$\epsilon_{n_f} = |V_{n_f-1} - V_{n_f}| \tag{3.42}$$

and these value present the decay of the error when the number of grid points increases.

Table 3.4: Option prices of basket calls. TOP: Sparse grid option prices for $d = 2$ and $d = 3$. BOTTOM: Option prices for the higher dimensions. n represents the maximum number of point in one dimension

	d=2		d=3	
n_f	Value	Error	Value	Error
3	2.291	$5.11 \cdot 10^{-1}$	2.102	$4.51 \cdot 10^{-1}$
4	2.727	$7.60 \cdot 10^{-2}$	2.498	$5.46 \cdot 10^{-2}$
5	2.801	$1.10 \cdot 10^{-3}$	2.562	$1.00 \cdot 10^{-2}$
6	2.807	$4.23 \cdot 10^{-3}$	2.563	$9.04 \cdot 10^{-3}$
7	2.811	$8.56 \cdot 10^{-3}$	2.562	$9.87 \cdot 10^{-3}$
8	2.807	$4.65 \cdot 10^{-3}$	2.555	$3.28 \cdot 10^{-3}$
9	2.803	$7.72 \cdot 10^{-4}$	2.554	$1.39 \cdot 10^{-3}$
10	2.803	$6.52 \cdot 10^{-4}$	2.553	$3.61 \cdot 10^{-4}$
	d=4		d=5	
3	1.983	$4.32 \cdot 10^{-1}$	1.896	$4.32 \cdot 10^{-1}$
4	2.364	$5.05 \cdot 10^{-2}$	2.273	$5.42 \cdot 10^{-2}$
5	2.429	$1.45 \cdot 10^{-2}$	2.343	$1.57 \cdot 10^{-2}$
6	2.427	$1.19 \cdot 10^{-2}$	2.341	$1.36 \cdot 10^{-2}$
7	2.422	$7.02 \cdot 10^{-3}$	2.331	$3.97 \cdot 10^{-3}$
8	2.420	$5.35 \cdot 10^{-3}$	2.335	$7.88 \cdot 10^{-3}$
9	2.417	$2.69 \cdot 10^{-3}$	2.330	$2.28 \cdot 10^{-3}$
10	2.413	$1.29 \cdot 10^{-3}$	2.326	$1.77 \cdot 10^{-3}$

In the table, satisfactory grid convergence is observed for the lower dimensional cases, but it is no longer regular. In particular when d increases the convergence becomes irregular. The reason may be due to the non-differentiability which is not at a grid line. An alternative is to use the

sparse grid technique based on a larger number of points in each dimension (b at least 4 or 8) see Definition 3.3.1, but also in this case, the accuracy is hampered by the fact that the initial condition is non-differentiable.

3.5 Coordinate transformation

Coordinate transformations -typically- are employed to transform a given partial differential equation into another one whose solution is easier to achieve. A first example of a multi-dimensional coordinate transformation has been presented in Section 3.4.3. In basket option pricing an important reason for using a coordinate transformation is to simplify the payoff function (1.6) or (1.19). This function is non-differentiable along a hyperplane $\sum_{k=1}^{d} w_k S_k = K$ in the d-dimensional domain. This plane crosses the Cartesian S_i-grid, which hampers the accuracy of the sparse grid combination technique, described in Section 3.3 and presented in Section 3.4.3. A general coordinate transformation from S_k to x_i can be written in the form

$$x_i = \zeta_i \left(S_1, S_2, \ldots, S_d\right), \tag{3.43}$$

$$S_k = \zeta_k^{-1} \left(x_1, x_2, \ldots, x_d\right). \tag{3.44}$$

We write $x_i = x_i(\mathbf{S})$ and $S_k = S_k(\mathbf{x})$ where $\mathbf{S}$ and $\mathbf{x}$ are d-dimensional vectors. If the transformations (3.43) and (3.44) are applied to the partial differential equation (1.9), it changes into

$$\frac{\partial V}{\partial t} + \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_{ij} \frac{\partial^2 V}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \beta_i \frac{\partial V}{\partial x_i} - rV = 0, \tag{3.45}$$

where

$$\alpha_{ij} = \sum_{k=1}^{d} \sum_{\ell=1}^{d} a_{k\ell} \frac{\partial x_i}{\partial S_k} \frac{\partial x_j}{\partial S_\ell}, \tag{3.46}$$

$$\beta_i = \sum_{k=1}^{d} \sum_{\ell=1}^{d} a_{k\ell} \frac{\partial^2 x_i}{\partial S_k \partial S_\ell} + \sum_{k=1}^{d} b_k \frac{\partial x_i}{\partial S_k}, \tag{3.47}$$

with $a_{k\ell} = \frac{1}{2}\rho_{k\ell}\sigma_k\sigma_\ell S_k(\mathbf{x}) S_\ell(\mathbf{x})$ and $b_k = (r - \delta_k) S_k(\mathbf{x})$.

The new coordinate x_1 is now chosen equal to the basket value

$$x_1 = \sum_{k=1}^{d} w_k S_k. \tag{3.48}$$

With this coordinate the transformed contract function for the basket call reads

$$\Phi(T, \mathbf{x}) = \max\{x_1 - K, 0\}. \tag{3.49}$$

This contract function is only dependent on x_1 and thus non-differentiable in only one coordinate direction. It now makes sense, for example, to use a truncation of this coordinate as in the 1D case presented by Kangro [30]

$$x_1^{max} = K \exp(\sqrt{2\sigma^2 T \log 100}).$$

We can, however, also safely use $x_1^{max} = 3K$. With this coordinate transformation, it may be possible to reduce the number of points in the other coordinates, as stated in [40, 48].

For the definition of the remaining coordinates, two basic choices are available: via a linear transformation or via a non-linear, normalised, transformation.

3.5.1 Linear transformation

A linear coordinate transformation can be written in the form

$$\mathbf{x} = \mathbf{G}\mathbf{S}, \tag{3.50}$$

with $\mathbf{G}$ the transformation matrix. The first row of matrix $\mathbf{G}$ is defined by the weights of the basket option. The other coefficients are chosen as follows (see for example [48], Chapter 5)

$$g_{ij} = \begin{cases} -w_j & j = i-1, i \neq 1, \\ w_j & j \neq i-1, i \neq 1 \vee i = 1. \end{cases} \tag{3.51}$$

Applying (3.51) to (3.46) and (3.47) gives

$$\alpha_{ij} = \sum_{k=1}^{d}\sum_{\ell=1}^{d} a_{k\ell} g_{ik} g_{j\ell}, \quad \beta_i = \sum_{k=1}^{d} b_k g_{ik}.$$

Note that $\partial^2 x_i / \partial S_k \partial S_\ell = 0$ with this transformation, because x_i is linear in S_k. If α_{ij} is written out in terms of the x_i-coordinates, using $\mathbf{S} = \mathbf{G}^{-1}\mathbf{x}$, we obtain:

$$\alpha_{ij} = \frac{1}{2}\sum_{k=1}^{d}\sum_{\ell=1}^{d}\sum_{m=1}^{d}\sum_{p=1}^{d} \rho_{k\ell}\sigma_k\sigma_\ell g_{ik} g_{j\ell}\gamma_{km}\gamma_{\ell p} x_m x_p$$

with γ_{km} elements of the inverse transformation matrix $\mathbf{G}^{-1}$.

The coordinate transformation must be non-singular. It is easy to see that (3.51) is non-singular, as it can be transformed into

$$\begin{pmatrix} w_1 & w_2 & \dots & w_{d-1} & w_d \\ -w_1 & w_2 & \dots & w_{d-1} & w_d \\ w_1 & -w_2 & \dots & w_{d-1} & w_d \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_1 & w_2 & \dots & -w_{d-1} & w_d \end{pmatrix} \rightarrow \begin{pmatrix} w_1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 2w_d \\ 0 & -2w_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -2w_{d-1} & 0 \end{pmatrix},$$

which is obviously non-singular.

The boundary conditions transformation accordingly. Coordinate x_1 is defined on $[0, x^{max}]$. At $x_1 = 0$ all asset prices are zero and therefore the option price itself is also set to zero. This is a Dirichlet condition. The linearity condition for x_1 towards infinity remains valid; x_1^{max} corresponds with $\sum w_k S_k^{max}$. For the other coordinates $x_i,\ i \neq 1$ we set linearity conditions on the both boundaries, as these transformed coordinates do not have their left-hand boundaries at $x_k = 0$. Therefore, it is not true in general that the coefficients of the particular derivatives vanish, which implies that the use of the linearity conditions both at $x_i = x^{min}$ as on $x_i = x^{max}$ with $i > 1$ makes good sense.

3.5.2 Non-linear transformation

It is also reasonable to employ a non-linear transformation, with normalised coordinates $x_j,\ j > 1$. By normalisation one can guarantee that the transformed coordinate directions remain in a $(d-1)$-dimensional *unit* hypercube. This transformation was developed for equally distributed basket put options ($\forall i, j\ w_i = w_j$) in [40, 41]. With basket weights w_k included, it reads

$$x_i = \begin{cases} \sum_{k=1}^{d} w_k S_k & i = 1, \\ \dfrac{w_{i-1} S_{i-1}}{\sum_{k=i-1}^{d} w_k S_k} & i > 1. \end{cases} \tag{3.52}$$

Correspondingly, we find the inverse transformation

$$S_k = \begin{cases} \dfrac{1}{w_1} x_1 x_2 & k = 1, \\ \dfrac{1}{w_k} x_1 x_{k+1} \prod_{j=1}^{k} (1 - x_j) & 1 < k < d, \\ \dfrac{1}{w_d} x_1 \prod_{j=1}^{d} (1 - x_j) & k = d. \end{cases} \tag{3.53}$$

Again the sum of the weighted assets in the basket is used for the first coordinate. Before the new coefficients (3.46) and (3.47) are derived, we define the following function

$$\hat{f}_{ik} := \begin{cases} x_{k+1} \prod_{j=i+1}^{k} (1 - x_j) & i < k < d, \\ \prod_{j=i+1}^{k} (1 - x_j) & i < k = d, \\ x_{k+1} & i = k < d, \\ 1 & i = k = d, \\ 0 & i > k. \end{cases} \tag{3.54}$$

Using (3.54), the coefficients (3.46) are transformed to

$$\alpha_{11} = x_1^2 \sum_{k=1}^{d}\sum_{\ell=1}^{d} \hat{\rho}_{k\ell}\hat{f}_{1k}\hat{f}_{1\ell},$$
$$\alpha_{1j} = x_1 x_j (1-x_j) \sum_{k=1}^{d}\sum_{\ell=1}^{d} (\hat{\rho}_{k,j-1} - \hat{\rho}_{k\ell})\, \hat{f}_{1k}\hat{f}_{j\ell}, \quad \forall 1 < j \leq d,$$

for $i = 1$. For $i > 1$:

$$\alpha_{ij} = x_i(1-x_i)x_j(1-x_j) \sum_{k=1}^{d}\sum_{\ell=1}^{d} (\hat{\rho}_{k\ell} - \hat{\rho}_{i-1,\ell} - \hat{\rho}_{k,j-1} + \hat{\rho}_{i-1,j-1})\, \hat{f}_{ik}\hat{f}_{j\ell},$$

with $\hat{\rho}_{k\ell} = \hat{\rho}_{\ell k} = \frac{1}{2}\rho_{k\ell}\sigma_k\sigma_\ell$, and $\alpha_{ij} = \alpha_{ji}$. The coefficients (3.47) now become

$$\beta_1 = \sum_{k=1}^{d} (r - \delta_k)\hat{f}_{1k},$$
$$\beta_i = x_i(1-x_i)\left(r - \delta_1 - \sum_{k=1}^{d}\left((r-\delta_k)\,\hat{f}_{ik}\right)\right) +$$
$$+ x_i(1-x_i)\left(\sum_{\ell=1}^{d}(-2\hat{\rho}_{i-1,i-1}x_i + (2x_i - 1)(\hat{\rho}_{k,i-1} + \hat{\rho}_{\ell,j-1}))\right)$$
$$+ x_i(1-x_i)\sum_{\ell=1}^{d} 2(1-x_i)\hat{\rho}_{k\ell}\hat{f}_{ik}\hat{f}_{i\ell}$$

The boundary conditions for x_1 are the same as in the case of the linear transformation. Furthermore, it can be shown that $\alpha_{ij} = 0$ and $\beta_j = 0$ for $x_j = 0$ and $x_j = 1$ with $i \geqslant 1$ and $j > 1$. This means that on these boundaries, the coefficients of the derivatives with respect to x_j vanish and the natural boundary conditions can be applied.

We will compare the accuracy of basket option prices and hedge parameters after employing one of these grid transformations. In addition we will evaluate the use of grid stretching in coordinate x_1 as presented in Section 2.4.

3.5.3 Hedge parameters and coordinate transformation

The Greeks for the singe-asset option contracts were discussed in Section 1.5. For the multi-dimensional option contracts, we concentrate on Δ_k, the first derivative w.r.t. asset price k, $\Gamma_{k,k}$, the second derivative of the price and the correlation parameter $\Gamma_{k,\ell}$, $(k \neq \ell)$, based on the mixed derivative of the price. We use numerical differentiation of the solution originating

from the sparse grid combination technique to obtain these Greeks, similar to the method in Section 2.4. When using transformation and stretching the equations for Δ_k and $\Gamma_{k,\ell}$ read

$$\Delta_k = \frac{\partial u}{\partial S_k} = \sum_{i=1}^{d} \frac{\partial u}{\partial x_i} \frac{\partial x_i}{\partial S_k}, \tag{3.55}$$

$$\Gamma_{k,\ell} = \frac{\partial^2 u}{\partial S_k \partial s_\ell} = \sum_{i=1}^{d} \frac{\partial u}{\partial x_i} \frac{\partial^2 x_i}{\partial S_k \partial S_\ell} + \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\partial^2 u}{\partial x_i \partial x_j} \frac{\partial x_i}{\partial S_k} \frac{\partial x_j}{\partial S_\ell}. \tag{3.56}$$

In higher dimensions, we typically do not have the solution on the whole domain available, as we work with only a set of sparse grid solutions. The solution of the partial differential equation on a region in the $d-$domain can, however, be obtained relatively easily by interpolation of the sparse grid solutions.

Consider a point $\mathbf{x} = \mathbf{x}_0$, where we wish to evaluate the option price V, Δ_k and $\Gamma_{k,\ell}$. Then from each sub-grid we interpolate the solution to a region around $\mathbf{x_0}$ with N_R points in each direction, such that there are enough points to use the proper difference equation to compute the hedge parameter. This region is a subset of the finest full grid. This means a successive interpolation to N_R^d points. The combination of all sub-grids is then straightforward, because we need to combine the interpolated solutions to the part of the finest grid. After this, we apply numerical differentiation for obtaining the Greeks on the relevant part of the finest grid. Schematically this is depicted in Figure 3.5.3.

As we use a higher order Lagrange interpolation for this purpose, we need $4 \times d$ points adjacent to point $\mathbf{x}_0$. The point $\mathbf{x}_0$ is placed in the middle of the region of interest. On each side of $\mathbf{x}_0$, we thus need two adjacent points for the first and second derivatives and four adjacent points for the mixed derivative.

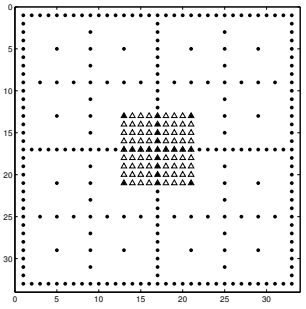


Figure 3.6: Representation of the interpolated Ω_R from the sparse grid

3.6 Option pricing experiments

In this section, the multi-dimensional Black-Scholes equation is solved numerically for European basket call options, European digital basket call options and Bermudan basket put options defined on three, four and five assets. The aim here is to evaluate numerically the spatial accuracy achieved by the numerical techniques presented above. In the test experiments, the time step is therefore fixed at $\Delta t = 10^{-3}$. The upper bound of the domain for the asset prices is $S_i^{max} = 3K$. As in our test experiments we do not use more than 256 points per coordinate direction, the error of the time integration, $\mathbf{O}(\Delta t^2)$, is negligible compared to the spatial discretisation error. So, in these model experiments we focus on the sparse grid spatial accuracy and neglect the effect of a discretisation in time.

3.6.1 Basket options

Three-dimensional full grid computations for a three-asset option are used as a reference to evaluate the influence of the coordinate transformation, the grid stretching, the use of fewer points in certain grid directions, and the use of sparse grids. Four- and five-asset options are computed with the techniques, that perform best in the three-asset reference computations. The option parameters used in the experiments are $K = 100, T = 1, r = 0.04, \rho_{ij} = 0.5, \delta_i = 0$ and all assets are equally weighted, hence $w_i = d^{-1}$. The volatilities for the different options can be found in Table 3.5.

Table 3.5: Volatilities for the call option pricing experiment.

d	σ_1	σ_2	σ_3	σ_4	σ_1
3	0.3	0.35	0.4		
4	0.3	0.35	0.4	0.45	
5	0.3	0.35	0.4	0.45	0.25

The spot price is chosen to be $S_1 = S_2 = S_3 = K$ (so $\sum_{k=1}^{3} w_k S_k = \sum_{k=1}^{3} \frac{S_k}{3} = K$). The reference value $V(\text{spot})$=13.245 is computed by use of an accurate FFT-based pricing technique (see Chapter 4) . In Table 3.6, option prices obtained on a 3D equidistant full grid are presented for the original formulation of the pricing PDE (1.9) as well as for the two types of transformations (linear and nonlinear). The total number of unknowns is 2^{3n_f}. In the experiments with equidistant grids we choose the same number of points for every coordinate.

The errors in Table 3.6 are computed as the absolute error between the computed value and $V(spot)$. We see in Table 3.6 that the use of only

Table 3.6: Three-asset option with the three formulations on *an equidistant full grid* of $(2^{n_f} \times 2^{n_f} \times 2^{n_f})$.

	Eq (1.9)		Eq (3.45) and (3.51)		Eq (3.45) and (3.52)	
n_f	Price	Error	Price	Error	Price	Error
3	12.862	$3.8 \cdot 10^{-1}$	13.937	$6.9 \cdot 10^{-1}$	13.904	$6.6 \cdot 10^{-1}$
4	13.150	$9.5 \cdot 10^{-2}$	13.196	$4.9 \cdot 10^{-2}$	13.173	$7.2 \cdot 10^{-2}$
5	13.221	$2.4 \cdot 10^{-2}$	13.236	$9.4 \cdot 10^{-3}$	13.232	$1.3 \cdot 10^{-2}$
6	13.239	$5.9 \cdot 10^{-3}$	13.242	$3.3 \cdot 10^{-3}$	13.241	$4.1 \cdot 10^{-3}$
7	13.243	$1.5 \cdot 10^{-3}$	13.244	$7.7 \cdot 10^{-4}$	13.244	$9.6 \cdot 10^{-4}$

a grid transformation does improve accuracy significantly on a full grid, as expected. Furthermore, the accuracy improves by a factor 4 with decreasing mesh sizes, which is also expected.

Table 3.7: Three-asset option on a *non-equidistant full grid* of size $(c_1 2^{n_f} \times c_2 2^{n_f} \times c_3 2^{n_f}), c_1 = 16, c_2 = c_3 = 4$.

	Eq (1.9)		Eq (3.45) and (3.51)		Eq (3.45) and (3.52)	
n_f	Price	Error	Price	Error	Price	Error
1	13.098	$1.5 \cdot 10^{-1}$	13.241	$4.3 \cdot 10^{-3}$	13.232	$1.3 \cdot 10^{-2}$
2	13.207	$3.8 \cdot 10^{-2}$	13.243	$2.2 \cdot 10^{-3}$	13.241	$4.0 \cdot 10^{-3}$
3	13.236	$9.4 \cdot 10^{-3}$	13.244	$5.1 \cdot 10^{-4}$	13.244	$9.5 \cdot 10^{-4}$
4	13.243	$2.3 \cdot 10^{-3}$	13.245	$1.4 \cdot 10^{-4}$	13.245	$2.4 \cdot 10^{-4}$

In Table 3.7, we can observe an interesting improvement in accuracy with the two coordinate transformations when non-equidistant grids are used with $c_1 = 16$ and $c_i = 4$, $i = 2, 3$. Four times fewer points have been used in these tests. Whereas the accuracy without transformation is worse compared to the results in Table 3.6, the effect of coordinate transformation is positive in this respect. The size of the grid at number $n_f = 3$ is $128 \times 32 \times 32$. This solution is comparable to the solution on the $128 \times 128 \times 128$ grid from Table 3.6.

Results obtained with the *sparse grid combination technique* (from section 3.3.4) corresponding to those in Tables 3.6 and 3.7, are presented in Table 3.8, for the equidistant case, and in Table 3.9, for the non-equidistant finest grids.

We observe the negative effect of the payoff function not being aligned with a grid line on the sparse grid accuracy in the second and third columns of Table 3.8, where the results with the original non-transformed grid are presented. The need to align the payoff function with a grid line can clearly

Table 3.8: Three-asset option with the three formulations on a regular sparse grid, representing a $(c_1 2^{n_s} \times c_2 2^{n_s} \times c_3 2^{n_s})$-grid $c_1 = c_2 = c_3 = 4$.

	Eq (1.9)		Eq (3.45) and (3.51)		Eq (3.45) and (3.52)	
n_s	Price	Error	Price	Error	Price	Error
1	12.868	$3.8 \cdot 10^{-1}$	13.937	$6.9 \cdot 10^{-1}$	13.904	$6.6 \cdot 10^{-1}$
2	13.440	$1.9 \cdot 10^{-1}$	13.198	$4.7 \cdot 10^{-2}$	13.173	$7.2 \cdot 10^{-2}$
3	13.150	$9.5 \cdot 10^{-2}$	13.237	$8.1 \cdot 10^{-3}$	13.232	$1.3 \cdot 10^{-2}$
4	13.326	$8.1 \cdot 10^{-2}$	13.242	$2.7 \cdot 10^{-3}$	13.241	$4.0 \cdot 10^{-3}$
5	13.230	$1.5 \cdot 10^{-2}$	13.244	$6.3 \cdot 10^{-4}$	13.244	$9.5 \cdot 10^{-4}$
6	13.233	$1.2 \cdot 10^{-2}$	13.245	$1.7 \cdot 10^{-4}$	13.245	$2.4 \cdot 10^{-4}$

be observed as the methods based on transformed coordinates show a very satisfactory accuracy.

We further notice that there is no significant difference between the linear and nonlinear coordinate transformations. In Table 3.9, grid stretching (2.25) is also included. A slightly better result is observed by using the stretching. It shows, however, that the reduction of grid points in the other (not the first) directions, by the choice for non-equidistant grids, is much more significant than the additional stretching of the first coordinate. The accuracy is dictated by the fewer grid points in the directions 2 and 3. Moreover, a fixed analytic grid stretching may not place the clustered points at the desired position for accuracy in the Greeks. The Greeks need not have their largest gradients near the exercise price. The errors in the hedge parameters in the sparse grid computation technique presented in Table 3.10 are computed as the difference between the values of two preceding layers, i.e: $|\Delta_{n_s+1} - \Delta_{n_s}|$ where Δ_{n_s} corresponds to a hedge parameter on layer n_s. The difference in accuracy between a linear and a non-linear transformation is negligible. The grid stretching slightly decreases the Greek's accuracies. We conclude that the use of grid stretching does not really pay off in these model examples.

An interesting notion is about the number of sub-grids Z that we need to evaluate with the sparse grid computation technique. Because of the choice of the c_i ($c_1 = 16, c_2 = c_3 = 4$), the number n_s can be chosen differently in the Tables 3.8 and 3.9 and therefore the number of grids employed is different. The number of grids for the equidistant case using equation (3.25) is 46, whereas for the non-equidistant case it is only 19. This is because in the latter case, the sparse grid evaluation is based on a $32 \times 8 \times 8$-grid rather than on an $8 \times 8 \times 8$-grid. The finest sub-grids in both cases have 2^{14} points per sub-grid and therefore the non-equidistant case has a lower complexity than the equidistant case, since fewer solutions have to be computed.

Table 3.9: Three-asset option with the two coordinate transformation methods on a non-equidistant sparse grid, representing a $(c_1 2^{n_s} \times c_2 2^{n_s} \times c_3 2^{n_s})$-grid, $c_1 = 16, c_2 = c_3 = 4$.

Without stretching				
	Eq (3.45) and (3.51)		Eq (3.45) and (3.52)	
n_s	Price	Error	Price	Error
1	13.241	$4.3 \cdot 10^{-3}$	13.232	$1.3 \cdot 10^{-2}$
2	13.243	$2.3 \cdot 10^{-3}$	13.241	$4.0 \cdot 10^{-3}$
3	13.244	$5.0 \cdot 10^{-4}$	13.244	$9.4 \cdot 10^{-4}$
4	13.245	$1.4 \cdot 10^{-4}$	13.245	$2.4 \cdot 10^{-4}$
With stretching				
	Eq (3.45) and (3.51)		Eq (3.45) and (3.52)	
n_s	Price	Error	Price	Error
1	13.265	$2.0 \cdot 10^{-2}$	13.259	$1.4 \cdot 10^{-2}$
2	13.249	$3.6 \cdot 10^{-3}$	13.247	$2.5 \cdot 10^{-3}$
3	13.246	$7.2 \cdot 10^{-4}$	13.245	$4.5 \cdot 10^{-4}$
4	13.245	$3.7 \cdot 10^{-5}$	13.245	$9.6 \cdot 10^{-5}$

Table 3.10: Greeks of the three-asset option on a non-equidistant sparse grid of size $(c_1 2^{n_s} \times c_2 2^{n_s} \times c_3 2^{n_s})$ with $c_1 = 16, c_2 = c_3 = 4$

	Non-linear transformation					
n_s	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error	$\Gamma_{1,2}$ (3.56)	Error
3	0.196		$1.6 \cdot 10^{-3}$		$1.53 \cdot 10^{-3}$	
4	0.197	$8.3 \cdot 10^{-4}$	$1.588 \cdot 10^{-3}$	$8.3 \cdot 10^{-4}$	$1.460 \cdot 10^{-3}$	$2.4 \cdot 10^{-6}$
5	0.197	$1.7 \cdot 10^{-4}$	$1.588 \cdot 10^{-3}$	$6.6 \cdot 10^{-4}$	$1.460 \cdot 10^{-3}$	$5.9 \cdot 10^{-7}$
6	0.197	$4.5 \cdot 10^{-5}$	$1.588 \cdot 10^{-3}$	$1.3 \cdot 10^{-4}$	$1.460 \cdot 10^{-3}$	$1.3 \cdot 10^{-7}$
n_s	Non-linear transformation and stretching					
1	0.198		$1.582 \cdot 10^{-3}$		$1.454 \cdot 10^{-3}$	
2	0.197	$7.8 \cdot 10^{-4}$	$1.586 \cdot 10^{-3}$	$7.9 \cdot 10^{-4}$	$1.458 \cdot 10^{-3}$	$3.8 \cdot 10^{-6}$
3	0.197	$2.00 \cdot 10^{-4}$	$1.587 \cdot 10^{-3}$	$5.9 \cdot 10^{-4}$	$1.459 \cdot 10^{-3}$	$8.1 \cdot 10^{-7}$
4	0.197	$4.9 \cdot 10^{-5}$	$1.587 \cdot 10^{-3}$	$1.5 \cdot 10^{-4}$	$1.450 \cdot 10^{-3}$	$2.0 \cdot 10^{-7}$

Table 3.11: Four-asset option price, Δ_1 and $\Gamma_{1,1}$. The sparse grid solution mimics a $(c_1 2^{n_s} \times c_2 2^{n_s} \times c_3 2^{n_s} \times c_4 2^{n_s})$-grid, $c_1 = 16, c_2 = c_3 = c_4 = 4$.

	4D linear, no stretching					
n_s	Price	Error	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error
1	13.672		0.1450		$8.697 \cdot 10^{-4}$	
2	13.662	$1.0 \cdot 10^{-2}$	0.146	$5.4 \cdot 10^{-4}$	$8.715 \cdot 10^{-4}$	$1.8 \cdot 10^{-6}$
3	13.661	$2.1 \cdot 10^{-3}$	0.146	$1.3 \cdot 10^{-4}$	$8.731 \cdot 10^{-4}$	$1.6 \cdot 10^{-6}$
4	13.659	$6.6 \cdot 10^{-4}$	0.146	$4.4 \cdot 10^{-5}$	$8.735 \cdot 10^{-4}$	$4.0 \cdot 10^{-7}$
	4D linear and stretching					
n_s	Price	Error	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error
1	13.686		0.146		$8.694 \cdot 10^{-4}$	
2	13.664	$2.1 \cdot 10^{-2}$	0.146	$5.0 \cdot 10^{-4}$	$8.718 \cdot 10^{-4}$	$2.342 \cdot 10^{-6}$
3	13.660	$4.4 \cdot 10^{-3}$	0.146	$1.3 \cdot 10^{-4}$	$8.729 \cdot 10^{-4}$	$1.124 \cdot 10^{-6}$
4	13.659	$1.1 \cdot 10^{-3}$	0.146	$3.3 \cdot 10^{-5}$	$8.731 \cdot 10^{-4}$	$2.484 \cdot 10^{-7}$
	4D non-linear					
n_s	Price	Error	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error
1	13.647		0.145		$8.734 \cdot 10^{-4}$	
2	13.656	$8.1 \cdot 10^{-3}$	0.146	$5.6 \cdot 10^{-4}$	$8.736 \cdot 10^{-4}$	$1.9 \cdot 10^{-7}$
3	13.658	$2.9 \cdot 10^{-3}$	0.146	$1.2 \cdot 10^{-4}$	$8.736 \cdot 10^{-4}$	$7.3 \cdot 10^{-9}$
4	13.659	$6.4 \cdot 10^{-4}$	0.146	$3.1 \cdot 10^{-5}$	$8.737 \cdot 10^{-4}$	$1.7 \cdot 10^{-8}$
	4D non-linear and stretching					
n_s	Price	Error	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error
1	13.671		0.147		$8.701 \cdot 10^{-4}$	
2	13.661	$1.0 \cdot 10^{-2}$	0.146	$5.8 \cdot 10^{-4}$	$8.726 \cdot 10^{-4}$	$2.4 \cdot 10^{-6}$
3	13.659	$1.7 \cdot 10^{-3}$	0.146	$1.4 \cdot 10^{-4}$	$8.731 \cdot 10^{-4}$	$5.5 \cdot 10^{-7}$
4	13.658	$4.4 \cdot 10^{-4}$	0.146	$3.6 \cdot 10^{-5}$	$8.732 \cdot 10^{-4}$	$1.4 \cdot 10^{-7}$

For the four- and five-asset option examples discussed next, we evaluate the coordinate transformation with and without grid stretching. The non-equidistant grids are also employed. For the five-asset basket call we focus only on the non-linear transformation. In the Tables 3.11 and 3.12, the results of these two option contracts are presented. The errors are computed as the difference between the option values in the point $S_i = K \ \forall i$ in two preceding layers. We observe that the non-equidistant grid also leads to very satisfactory accuracy here. The determination of the hedge parameters also works fine in higher dimensions. Grid stretching again does not seem to be necessary for obtaining small truncation errors. Note that the reason for a slight decrease in the grid convergence of the 4D and 5D sparse grid solutions is due to the term $(\log(h_l^{-1}))^{d-1}$ in equation (3.23).

3.6.2 Digital option

The techniques presented can also be used for pricing options with discontinuous contract functions. An example is the digital basket option. The

Table 3.12: Five-asset option price, Δ_1 and $\Gamma_{1,1}$. The sparse grid solution mimics a full grid of $(c_1 2^{n_s} \times c_2 2^{n_s} \times c_3 2^{n_s} \times c_4 2^{n_s} \times c_5 2^{n_s})$ points, $c_1 = 16, c_2 = c_3 = c_4 = c_5 = 4$.

Non-linear transformation, no grid stretching						
n_s	Price	Error	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error
1	12.670		0.118		$6.057 \cdot 10^{-4}$	
2	12.679	$9.113 \cdot 10^{-3}$	0.118	$5.4 \cdot 10^{-4}$	$6.056 \cdot 10^{-4}$	$1.2 \cdot 10^{-7}$
3	12.682	$3.276 \cdot 10^{-3}$	0.118	$1.1 \cdot 10^{-4}$	$6.055 \cdot 10^{-4}$	$7.3 \cdot 10^{-8}$
4	12.683	$7.495 \cdot 10^{-4}$	0.118	$2.9 \cdot 10^{-5}$	$6.055 \cdot 10^{-4}$	$8.3 \cdot 10^{-9}$
Non-linear transformation and stretching						
n_s	Price	Error	Δ_1 (3.55)	Error	$\Gamma_{1,1}$ (3.56)	Error
1	12.700		0.119		$6.033 \cdot 10^{-4}$	
2	12.687	$1.3 \cdot 10^{-2}$	0.118	$5.0 \cdot 10^{-4}$	$6.049 \cdot 10^{-4}$	$1.7 \cdot 10^{-6}$
3	12.684	$2.4 \cdot 10^{-3}$	0.118	$1.2 \cdot 10^{-4}$	$6.053 \cdot 10^{-4}$	$3.3 \cdot 10^{-7}$
4	12.683	$6.3 \cdot 10^{-4}$	0.118	$3.1 \cdot 10^{-5}$	$6.054 \cdot 10^{-4}$	$8.4 \cdot 10^{-8}$

contract function of a digital basket call reads:

$$V(T, \mathbf{S}) = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{d} w_i S_i > K \\ 0 & \text{elsewhere.} \end{cases} \tag{3.57}$$

The discontinuity is not aligned with a grid line, unless coordinate transformation from Section 3.5 is used. Here, other option parameters are chosen, in particular, a shorter time to maturity (leading to solutions with steep gradients):

$$K = 100 \quad r = 0.05\% \quad T = 0.25 \quad \delta = \begin{pmatrix} 0.02 & 0.03 & 0.06 & 0.04 & 0.07 \end{pmatrix}$$
$$w_i = 1/d \quad \sigma = \begin{pmatrix} 0.30 & 0.35 & 0.40 & 0.33 & 0.27 \end{pmatrix}$$
$$\rho = \begin{pmatrix} 1.00 & 0.50 & 0.25 & 0.17 & 0.10 \\ 0.50 & 1.00 & -0.25 & -0.65 & -0.30 \\ 0.25 & -0.25 & 1.00 & 0.50 & 0.45 \\ 0.17 & -0.65 & 0.50 & 1.00 & 0.07 \\ 0.10 & -0.30 & 0.45 & 0.07 & 1.00 \end{pmatrix}$$

In Table 3.13, the results for a digital basket call are presented. Again, the errors are computed as the difference between the option values in the point $S_i = K \ \forall i$ in two preceding layers. As expected, we now observe a lower convergence of $\mathbf{O}(h)$ due to the discontinuity, but the accuracy is still satisfactory. The grid stretching gives the same error convergence, but a more accurate result. It is a helpful technique in the case of solutions with steep gradients.

Table 3.13: Digital basket call option with 3,4 and 5 underlying assets. The sparse grid solution mimics a full grid of $16 \cdot 2^{n_s}$ points in the first direction and $4 \cdot 2^{n_s}$ in the other directions.

Non-linear transformation, no stretching						
	3D		4D		5D	
n_s	Price	Error	Price	Error	Price	Error
1	0.543		0.559		0.568	
2	0.455	$8.8 \cdot 10^{-2}$	0.451	$1.1 \cdot 10^{-1}$	0.441	$1.3 \cdot 10^{-1}$
3	0.496	$4.1 \cdot 10^{-2}$	0.500	$4.9 \cdot 10^{-2}$	0.497	$5.6 \cdot 10^{-2}$
4	0.476	$2.0 \cdot 10^{-2}$	0.476	$2.4 \cdot 10^{-2}$	0.470	$2.7 \cdot 10^{-2}$
Non-linear transformation, stretching						
n_s	Price	Error	Price	Error	Price	Error
1	0.483		0.486		0.481	
2	0.470	$1.3 \cdot 10^{-2}$	0.470	$1.6 \cdot 10^{-2}$	0.463	$1.8 \cdot 10^{-2}$
3	0.477	$6.7 \cdot 10^{-3}$	0.478	$7.9 \cdot 10^{-3}$	0.472	$8.9 \cdot 10^{-3}$
4	0.480	$3.4 \cdot 10^{-3}$	0.482	$4.0 \cdot 10^{-3}$	0.476	$4.5 \cdot 10^{-3}$

3.6.3 Bermudan option

We conclude the experiments with a Bermudan put option. The option parameters are again chosen differently:

$$
\begin{aligned}
K = 50 \quad r = 5\% \quad T = 0.25 \\
\sigma &= \begin{pmatrix} 0.41 & 0.38 & 0.39 & 0.37 & 0.42 \end{pmatrix} \\
\rho &= \begin{pmatrix} 1.00 & 0.95 & 0.90 & 0.86 & 0.81 \\ 0.95 & 1.00 & 0.95 & 0.90 & 0.86 \\ 0.90 & 0.95 & 1.00 & 0.95 & 0.90 \\ 0.86 & 0.90 & 0.95 & 1.00 & 0.95 \\ 0.81 & 0.86 & 0.90 & 0.95 & 1.00 \end{pmatrix} \\
\delta &= \begin{pmatrix} 0.02 & 0.03 & 0.06 & 0.04 & 0.07 \end{pmatrix} \\
w_{3D} &= \begin{pmatrix} 0.45 & 0.30 & 0.25 \end{pmatrix} \\
w_{4D} &= \begin{pmatrix} 0.4 & 0.2 & 0.1 & 0.3 \end{pmatrix} \\
w_{5D} &= \begin{pmatrix} 0.32 & 0.28 & 0.18 & 0.10 & 0.12 \end{pmatrix}
\end{aligned}
$$

A Bermudan put gives the holder the right to exercise at discrete moments prior to the maturity date (See Section 1.6). In this experiment 10 exercise dates are allowed, that are equally spaced along the duration of the option contract. At each exercise date, the option value is the maximum of the computed value at the current date t_m and the payoff function:

Table 3.14: 10-times exercisable Bermudan basket put option with 3, 4 and 5 underlying assets. The sparse grid solution mimics a full grid of $16 \cdot 2^{n_s}$ points in the first direction and $4 \cdot 2^{n_s}$ in the other directions.

Non-linear transformation, no stretching						
	3D		4D		5D	
n_s	Price	Error	Price	Error	Price	Error
3	10.365		10.042		10.344	
4	10.370	$4.7 \cdot 10^{-3}$	10.046	$4.1 \cdot 10^{-3}$	10.349	$4.9 \cdot 10^{-3}$
5	10.370	$5.2 \cdot 10^{-4}$	10.048	$1.2 \cdot 10^{-3}$	10.349	$4.7 \cdot 10^{-4}$
6	10.370	$1.8 \cdot 10^{-4}$	10.048	$2.0 \cdot 10^{-5}$	10.350	$4.2 \cdot 10^{-4}$

$V(t_m, \mathbf{S}) = \max\{V(T, \mathbf{S}(T)), V(t_m, \mathbf{S}(t_m))\}$. In Table 3.14, we present the results of the computation with up to 5 underlying assets. We observe a satisfactory accuracy for all option contracts, although the convergence is irregular. This may be due to the choice of correlation coefficients and, in particular due to the early exercise feature. The non-linear transformation works well for this case.

Although the results of these experiments are positive for the use of sparse grids for basket options, it also gives rise to some serious thoughts on the applicability of the sparse grid method. Satisfactory sparse grid accuracy can be achieved for options whose payoff function coincides with a grid line after a coordinate transformation. This may, however, not be easily possible for complex payoff functions, as they are usually encountered in the financial industry.

3.7 Conclusions

For pricing basket options with the multi-dimensional Black-Scholes equation and the sparse grid combination technique, a linear or a non-linear coordinate transformation can be employed in order to align the payoff function with a grid line. An additional stretching function concentrates points in the region around the exercise price. With the coordinate transformations it is possible to reduce the number of grid points in some coordinates, which is highly advantageous from a computational point of view. The effect of grid stretching is mainly significant on these non-equidistant grids if the maturity time is short for digital options (as then steep gradients in the solution occur). With the coordinate transformation the sparse grid combination technique can be efficiently employed to achieve very satisfactory grid accuracy in space. A significant reduction in the number of sub-grids that need to be processed can be achieved by a clever definition of the base grid. For the

model problems evaluated, the difference in the accuracy between the linear or the nonlinear coordinate transformations is not significant. This includes the evaluation of the hedge parameters. Both the linear and the nonlinear transformation perform very well. The nonlinear transformation gives rise to a basket option problem with easier boundary conditions. A critical observation is about the generality of the sparse grid method in multi-asset option pricing. For highly complicated payoff functions that typically cannot be transformed to a low-dimensional hyper-plane the efficient use of sparse grids is not so evident. The transformation also works for options with discontinuous payoff functions and options with early exercise.

Chapter 4

Option pricing using a parallel FFT method

4.1 Introduction

In this chapter, we present a pricing technique based on the risk-neutral integral equation (1.8) presented in Section 1.3.2. This type of integrals can be solved numerically by the use of the fast Fourier transform (FFT). The FFT is a very efficient algorithm for the discrete Fourier transform following from the discretisation of the integral in equation (1.8). Again, the curse of dimensionality plays an important role in the solution, since large vectors have to be stored in the memory of a computer. However, a major advantage of the FFT is the availability of nice division techniques. When the FFT is divided in parts, the method can be applied to larger problems than the PDE method in Chapter 3. The outline of this chapter is as follows. We start with a short introduction of transform-based option pricing methods and present the CONV method [33] in detail. Section 4.3 gives some insight in the fast Fourier transform as well as in its parallelisation. Furthermore, the parallelisation of the CONV method and of the sparse grid technique is discussed. We conclude the chapter with the numerical experiments in Section 4.4 and we draw the conclusions of the FFT-based option pricing method in Section 4.5.

4.2 The multi-dimensional CONV method

4.2.1 Background

The CONV method to be presented falls in the category of transform-based methods (see for example [1, 43]). In particular, the method in [16] employs the FFT to price options on more than one asset. These methods originate from the risk-neutral valuation formula (1.8), which for single-asset problems

reads:

$$V(t, S(t)) = e^{-r(T-t)}\mathbb{E}^Q \left[V(T, S(T))\right], \tag{4.1}$$

where again V denotes the value of the option, r is the risk-free interest rate, t is the current time, T is the maturity date and S represents the price of the underlying asset. The interest rate, r, is again assumed to be deterministic here. Equation (4.1) is an expectation and can be evaluated using numerical integration provided that the probability density function is known. Equation (4.1) can be written as,

$$V(t, x(t)) = e^{-r(T-t)} \int_{K^*}^{\infty} \left(e^{x(T)} - e^{K^*}\right) f(x)dx, \tag{4.2}$$

with $K^* = \ln K$, $x(t) = \ln S(t)$ and $f(x)$ the probability density function. The value of $V(t, x(t))$ tends to $S(0)$ as K^* tends to $-\infty$ and hence the call price is not square integrable. Therefore, the contract function should be damped in any way. Commonly, it is multiplied by damping factor $\exp(\alpha K^*)$, with $\alpha > 0$. The computation of the Fourier transform of the option value, ψ, can be done by using the characteristic function, $\varphi(\omega)$, as proposed by Carr and Madan [12]:

$$\begin{aligned} \psi(\omega) &= e^{-r(T-t)} \int_{-\infty}^{\infty} e^{i\omega K^*} \int_{K^*}^{\infty} e^{\alpha K^*} \left(e^{x} - e^{K^*}\right) f(x)dxdK^* \\ &= \frac{e^{-r(T-t)}\varphi(\omega - (\alpha+1)i)}{\alpha^2 + \alpha - v^2 + i(2\alpha+1)v}, \end{aligned} \tag{4.3}$$

where the characteristic function of the underlying is defined by

$$\varphi(u) = \mathbb{E}\left[e^{iu \ln S(T)}\right]. \tag{4.4}$$

For a European call an exact solution exists provided φ is known. To compute the call price, the inverse Fourier transform [21] has to be computed:

$$V(t, x(t)) = \frac{e^{-\alpha K^*}}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega K^*} \psi(\omega)d\omega. \tag{4.5}$$

Following the same steps for basket options, however, would require the characteristic function of a basket value, which is not known in general.

For common baskets quite accurate approximations can be obtained by assuming that the basket itself is an asset following the same distribution as one of the underlying assets [19, 54]. Here, we evaluate a method which does not rely on such an approximation.

4.2.2 The CONV-method

Like all transform-based methods, the CONV method [33] is also based on the risk-neutral valuation formula (1.8). In the multi-dimensional version

we need to compute:

$$V(t, \mathbf{x}(t)) = e^{-r(T-t)} \int_{\mathbb{R}^d} \Phi(T, \mathbf{y}) f(\mathbf{y}|\mathbf{x}) d\mathbf{y}, \tag{4.6}$$

where $\mathbf{x} = \ln \mathbf{S}(t)$ is a vector of the log of the asset prices, $\mathbf{y} = \ln \mathbf{S}(T)$ and $f(\mathbf{y}|\mathbf{x})$ is the probability density function of the transition of $\mathbf{x}$ at time t to $\mathbf{y}$ at time T.

With the right to exercise at certain times, t_n, before the maturity date, T, the *Bermudan option price* is defined by:

$$V(t_n, \mathbf{x}(t_n)) = \max \left\{ \Phi(t_n, \mathbf{x}(t_n)) \ , \ e^{-r(t_{n+1}-t_n)} \int_{\mathbb{R}^d} V(t_{n+1}, \mathbf{y}) f(\mathbf{y}|\mathbf{x}) d\mathbf{y} \right\}, \tag{4.7}$$

with $\Phi(t_n, \mathbf{x}(t_n))$ the contract function at t_n.

Remark 4.2.1. (Smooth Fit Principle) It is well-known that in the case of American options under Black-Scholes dynamics the derivative of the value function is continuous (smooth fit principle). This is however not the case anymore when pricing Bermudan options, for which the function V will have a discontinuous first derivative. Though at the final exercise time the location of this discontinuity is known, this is not the case at previous exercise times. This may hamper the numerical treatment of options in the present context.

At each exercise date, t_n, valuation of (4.7) can be interpreted as a European-style contract with maturity time t_{n+1} and "initial" time t_n. For the derivation of the multi-asset CONV method, we can therefore focus on (4.6) and keep the notation as in (4.6).

The main premise of the CONV method is that the transition density function $f(\mathbf{y}|\mathbf{x})$ equals the density of the difference of $\mathbf{y}$ and $\mathbf{x}$:

$$f(\mathbf{y}|\mathbf{x}) = f(\mathbf{y} - \mathbf{x}). \tag{4.8}$$

This holds for several models, such as geometric Brownian motion and, more generally, Lévy processes, which have independent increments. Then with $\mathbf{z} = \mathbf{y} - \mathbf{x}$, we have:

$$V(t, \mathbf{x}) = e^{-r(T-t)} \int_{\mathbb{R}^d} \Phi(T, \mathbf{x} + \mathbf{z}) f(\mathbf{z}) d\mathbf{z}, \tag{4.9}$$

which is a cross-correlation between Φ and f. The cross-correlation operator can be treated as a convolution operator [24] and therefore the method is called the CONV method in [33].

The valuation of equation (4.9) can be done numerically for known probability density functions. For several asset price models, including the Lévy processes, however, only the characteristic function is known. When transforming the cross-correlation operator to the Fourier space, we have to deal

with a product of the Fourier transform of the contract function and the Fourier transform of the probability density function, which is the characteristic function.

The Fourier transform of a function is only available if the function is $\mathbf{L}_1-$integrable. This is typically not the case for multi-asset contract functions as damping techniques [12] and [33] are not available for multi-asset options in general. As an example we try to integrate a contract function of a two-dimensional basket call, which is damped by $e^{\alpha_1 x_1+\alpha_2 x_2}$,

$$\begin{aligned}
&\int_{\mathbb{R}^2} e^{\alpha_1 x_1+\alpha_2 x_2} K\left(c_1 e^{x_1}+c_2 e^{x_2}-1\right)^+ dx_1 dx_2 \\
=&K\int_{-\infty}^{-\ln c_2}\int_{\ln(1-c_2 e^{x_2})-\ln c_1}^{\infty} e^{\alpha_1 x_1+\alpha_2 x_2}\left(c_1 e^{x_1}+c_2 e^{x_2}-1\right) dx_1 dx_2 \\
+&K\int_{-\ln c_2}^{\infty}\int_{-\infty}^{\infty} e^{\alpha_1 x_1+\alpha_2 x_2}\left(c_1 e^{x_1}+c_2 e^{x_2}-1\right) dx_1 dx_2.
\end{aligned} \tag{4.10}$$

The second term in (4.10) is unbounded because of the integration over $\mathbb{R}$ for x_1. It is not possible to find a proper value for α_1 in order to bound this integral.

Instead of performing the FT analytically, in the multi-dimensional CONV method the computation is done *numerically* and therefore the domain of integration has to be truncated. Since the density in (4.9) decays to zero rapidly as $\mathbf{z}\to\pm\infty$, we truncate the infinite integration range without loosing significant accuracy to $[a_i,b_i]\subset\mathbb{R}^d,\ i=1,\ldots,d$. The direct construction of the discretised multi-dimensional CONV formula, below, via a Fourier series expansion of the continuation value replaces L^1-integrability on $(-\infty,\infty)$ with L^1-*summability* on a truncated computational domain, so that the restriction on α is removed. Experiments in [33] showed that one can choose $\alpha=0$ in the CONV method for many contracts, and obtain accurate option values. The discrete version will however resemble its continuous counterpart more and more as the domain size increases.

So, our point of departure will be a truncated domain, Ω_d. The size of Ω_d is chosen such that the error made due to truncation is negligible compared to the discretisation error. To show the accuracy of the truncation, we refer Table 4.1. In this table a numerical experiment with a basic European call option is included, illustrating the effect of the truncation. It is shown numerically that a domain of size $\Omega_i=[-L_i,+L_i]$ with $L_i=20\sigma_i$ gives highly accurate results, with σ_i denoting the standard deviation of the density. This domain size is set in all experiments to follow.

We now take the Fourier transform of equation (4.9):

$$\begin{aligned}
e^{r(T-t)}\mathcal{F}\{V(t,\mathbf{x})\}(\boldsymbol{\omega}) &= \int_{\Omega^d} e^{i\boldsymbol{\omega}\mathbf{x}}\left[\int_{\Omega^d}\Phi(T,\mathbf{x}+\mathbf{z})f(\mathbf{z})d\mathbf{z}\right]d\mathbf{x} \\
&= \int_{\Omega^d}\int_{\Omega^d} e^{i\boldsymbol{\omega}(\mathbf{x}+\mathbf{z})}\Phi(T,\mathbf{x}+\mathbf{z})f(\mathbf{z})e^{-i\boldsymbol{\omega}\mathbf{z}}d\mathbf{z}d\mathbf{x}.
\end{aligned} \tag{4.11}$$

Table 4.1: At the money error caused by truncation to Ω_d with 2^{20} discretisation points (see section 4.2.4) for a European call. $K = 40, r = 0.06, \delta = 0.04, \sigma = 0.25, T = 1$

L	$Ke^{x_{min}}$	$Ke^{x_{max}}$	Error
σ	31.15	51.36	2.45×10^{0}
2σ	24.26	65.95	1.42×10^{0}
4σ	14.72	108.73	1.92×10^{-1}
8σ	5.41	295.56	2.40×10^{-4}
12σ	1.99	803.42	7.54×10^{-9}
16σ	0.73	2183.91	6.86×10^{-10}
20σ	0.27	5936.47	2.07×10^{-10}

Here the multi-dimensional Fourier transform, and its inverse, are defined as:

$$\mathcal{F}\{h(\mathbf{x})\}(\boldsymbol{\omega}) = \int_{\Omega^d} e^{i\boldsymbol{\omega}\mathbf{x}} h(\mathbf{x}) d\mathbf{x},$$
$$\mathcal{F}^{inv}\{H(\boldsymbol{\omega})\}(\mathbf{x}) = \frac{1}{(2\pi)^d} \int_{\Omega^d} e^{-i\boldsymbol{\omega}\mathbf{x}} H(\boldsymbol{\omega}) d\boldsymbol{\omega},$$

with, for example,

$$\int_{\Omega^d} e^{i\boldsymbol{\omega}\mathbf{x}} h(\mathbf{x}) d\mathbf{x} = \int_{-L_d}^{L_d} \dots \int_{-L_1}^{L_1} e^{i\omega_1 x_1} \dots e^{i\omega_d x_d} h(x_1, \dots, x_d) dx_1 \dots dx_d.$$

Changing the order of integration in (4.11) and using $\mathbf{y} = \mathbf{x} + \mathbf{z}$, we find:

$$\begin{aligned} e^{r(T-t)} \mathcal{F}\{V(t,\mathbf{x})\}(\boldsymbol{\omega}) &= \int_{\Omega^d} \int_{\Omega^d} e^{i\boldsymbol{\omega}\mathbf{y}} \Phi(T,\mathbf{y}) f(\mathbf{z}) e^{-i\boldsymbol{\omega}\mathbf{z}} d\mathbf{y} d\mathbf{z} \\ &= \int_{\Omega^d} e^{i\boldsymbol{\omega}\mathbf{y}} \Phi(T,\mathbf{y}) d\mathbf{y} \int_{\Omega^d} e^{-i\boldsymbol{\omega}\mathbf{z}} f(\mathbf{z}) d\mathbf{z} \\ &= \mathcal{F}\{\Phi(T,\mathbf{y})\}(\boldsymbol{\omega}) \varphi(-\boldsymbol{\omega}), \end{aligned} \tag{4.12}$$

with φ the characteristic function. After taking the inverse Fourier transform, the option price is found to be:

$$V(t,\mathbf{x}) = e^{-r(T-t)} \mathcal{F}^{inv}\left\{\mathcal{F}\{\Phi(T,\mathbf{y})\}\varphi(-\boldsymbol{\omega})\right\}. \tag{4.13}$$

Aliasing, a commonly observed feature when dealing with a convolution of sampled signals by means of the FFT, is not a problem in our application, as we encounter a convolution of a characteristic function and the DFT of a vector with option values. The DFT is periodical but this would make the convolution circular only if the characteristic function would also be obtained by a DFT. However, we work with the analytical characteristic function, which is not periodic.

4.2.3 Characteristic function and hedge parameters

In the multi-asset case considered here, the asset prices are modelled as correlated log-normal distributions. The characteristic function can be found via the probability density function,

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}\right)^T \mathbf{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}\right)\right),$$

where $\mu_j = \left(r - \delta_j - \frac{1}{2}\sigma_j^2\right)(T-t)$, $\mathbf{\Sigma}$ is the correlation matrix with $[\sigma]_{jk} = \rho_{jk}\sigma_j\sigma_k(T-t)$ and $|\mathbf{\Sigma}|$ its determinant. Dividend-yields δ_j, volatilities σ_j and the correlation coefficients ρ_{jk} are assumed to be known. The characteristic function reads, by using equation (4.4),

$$\varphi(\boldsymbol{\omega}) = \exp\left(i\sum_{k=1}^{d}\mu_k\omega_k - \frac{1}{2}(T-t)\sum_{j=1}^{d}\sum_{k=1}^{d}\rho_{jk}\sigma_j\sigma_k\omega_j\omega_k\right). \tag{4.14}$$

The hedge parameters can easily be obtained using the CONV method. The hedge parameters can be computed in an analytic way by the use of the derivative properties of the Fourier transform [24]:

$$\mathcal{F}\left(\frac{df}{dx}\right) = -i\omega\mathcal{F}\left(f\right).$$

Now the hedge parameters are:

$$\Delta_j(t,\mathbf{x}) = \frac{\partial V}{\partial S_j} = -\frac{e^{-r(T-t)}}{S_j}\mathcal{F}^{inv}\left\{i\omega_j\mathcal{F}\{\Phi(T,\mathbf{y})\}\varphi(-\boldsymbol{\omega})\right\}, \tag{4.15}$$

$$\Gamma_j(t,\mathbf{x}) = \frac{\partial^2 V}{\partial S_j^2} = \frac{e^{-r(T-t)}}{S_j^2}\mathcal{F}^{inv}\left\{\left(i\omega_j + \omega_j^2\right)\mathcal{F}\{\Phi(T,\mathbf{y})\}\varphi(-\boldsymbol{\omega})\right\}. \tag{4.16}$$

4.2.4 Discretisation of the CONV-method

Equation (4.13) can now be solved numerically with the help of multi-dimensional quadrature rules. A computation using the DFT requires equidistant grids for $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\omega}$. The number of grid points per coordinate is again chosen to be N_j. The domains are chosen to be symmetric around the origin and hence $\Omega^d = [-L_1, L_1]\times[-L_2, L_2]\times\ldots\times[-L_d, L_d]$. The values of L_j are dependent on σ_j according to [33]. We use a fixed region of $L_j = 20\sigma_j$. Now the meshwidth for coordinate j is defined as $dx_j = dy_j = \frac{2L_j}{N_j}$ and therefore we have:

$$x_j(k_j) = -L + k_j dx_j = (k_j - \frac{1}{2}N_j)dx_j \quad 0 \leqslant k_j \leqslant N_j - 1 \tag{4.17}$$

$$y_j(k_j) = -L + k_j dy_j = (k_j - \frac{1}{2}N_j)dy_j \quad 0 \leqslant k_j \leqslant N_j - 1. \tag{4.18}$$

To avoid aliasing, the Nyquist relation between the grid size in the frequency domain ω and log-asset x must be obeyed:

$$dx_j \cdot d\omega_j = \frac{2\pi}{N_j}. \tag{4.19}$$

and for the discrete frequency domain, we have:

$$\omega_j(n_j) = (n_j - \frac{1}{2}N_j)d\omega_j \quad 0 \leqslant n_j \leqslant N_j - 1. \tag{4.20}$$

Remark 4.2.2. We will use k_j and m_j for the index numbers in the log-asset domain and n_j for the index numbering in the frequency domain according to the standard literature [24].

The functions defined on the discretised log-asset prices are abbreviated:

$$\begin{aligned}\Phi(T, y_1(k_1), \ldots, y_d(k_d)) &= \Phi_{\mathbf{k}} \\ V(t, x_1(k_1), \ldots, x_d(k_d)) &= V(t, \mathbf{x_k}).\end{aligned}$$

with $\mathbf{k} = (k_1, k_2, \ldots, k_d)$. Similar for the functions in the frequency domain:

$$\varphi(-\omega_1(n_1), -\omega_2(n_2), \ldots, -\omega_d(n_d)) = \varphi_{\mathbf{n}},$$

with $\mathbf{n} = (n_1, n_2, \ldots, n_d)$. The Fourier transform $\mathcal{F}\{\Phi(T, \mathbf{y})\}_{\mathbf{n}}$ of Φ is abbreviated as $\widehat{\Phi}_{\mathbf{n}}$. Finally we use the abbreviation

$$\sum_{k_1=0}^{N_1-1} \cdots \sum_{k_d=0}^{N_d-1} = \sum_{\mathbf{k=0}}^{\mathbf{N-1}}$$

when approximating the Fourier integrals to summations. The approximation of the inner integral of (4.13) by the trapezoidal rule gives:

$$\widehat{\Phi}_{\mathbf{n}} := \int_{\mathbb{R}^d} \Phi(T, \mathbf{y}) e^{i\boldsymbol{\omega}\mathbf{y}} d\mathbf{y} \approx d\mathbf{Y} \sum_{\mathbf{k=0}}^{\mathbf{N-1}} Z_{\mathbf{k}} \Phi_{\mathbf{k}} \exp(i\omega_{n_1} y_{k_1} + \ldots + i\omega_{n_d} y_{k_d}), \tag{4.21}$$

with $d\mathbf{Y} = \prod_{j=1}^{d} dy_j$, $Z_{\mathbf{k}} = \prod_{j=1}^{d} R_j(k_j)$ and the trapezoidal weights,

$$R_j(k_j) = \begin{cases} \frac{1}{2} & k_j = 0 \vee k_j = N_j - 1 \\ 1 & \text{otherwise.} \end{cases}$$

Using the Nyquist relation (4.19), the terms $\exp(i\omega_{n_j} y_{k_j}), j = 1 \ldots d$, can be rewritten as,

$$\begin{aligned}\exp(i\omega_{n_j} y_{k_j}) &= \exp(i(n_j - \frac{1}{2}N_j)d\omega_j(k_j - \frac{1}{2}N_j)dx_j) \\ &= \exp(\frac{2\pi i n_j k_j}{N_j}) \exp(-\pi i(k_j + n_j)) \exp(\frac{\pi i N_j}{2})\end{aligned}$$

Introducing the twiddle factors[1], $W_{N_j} = \exp(-\frac{2\pi i}{N_j})$ as one of N_j complex $N-$th roots of unity, we have:

$$\exp(\frac{2\pi i n_j k_j}{N_j}) \exp(-\pi i(k_j + n_j)) \exp(\frac{\pi i N_j}{2}) = (-1)^{k_j}(-1)^{n_j + \frac{N_j}{2}} W_{N_j}^{-n_j k_j}.$$

Equation (4.21) can be written as:

$$\widehat{\Phi}_{\mathbf{n}} \approx d\mathbf{Y} \sum_{\mathbf{k=0}}^{\mathbf{N-1}} G_{\mathbf{k}} \Phi_{\mathbf{k}} \prod_{j=1}^{d} (-1)^{n_j + \frac{N_j}{2}} W_{N_j}^{-n_j k_j}, \tag{4.22}$$

with $G_{\mathbf{k}} = Z_{\mathbf{k}} \prod_{j=1}^{d} (-1)^{k_j}$.

Recognising that

$$\mathcal{D}_d \{f\}_{\mathbf{n}} = \sum_{\mathbf{k=0}}^{\mathbf{N-1}} f_{\mathbf{k}} \prod_{j=1}^{d} e^{\frac{2\pi i n_j k_j}{N_j}} = \sum_{\mathbf{k=0}}^{\mathbf{N-1}} f_{\mathbf{k}} W_{\mathbf{N}}^{-\mathbf{nk}} \tag{4.23}$$

$$\mathcal{D}_d^{inv} \{F\}_{\mathbf{k}} = \frac{1}{\prod_{j=1}^{d} N_j} \sum_{\mathbf{n=0}}^{\mathbf{N-1}} F_{\mathbf{n}} W_{\mathbf{N}}^{\mathbf{nk}} \tag{4.24}$$

are the discrete Fourier transform (DFT) and inverse Fourier transform, respectively, we have:

$$\widehat{\Phi}_{\mathbf{n}} \approx d\mathbf{Y} \prod_{j=1}^{d} \left((-1)^{n_j + \frac{N_j}{2}} \right) \cdot \mathcal{D}_d \left[G_{\mathbf{k}} \Phi_{\mathbf{k}} \right]_{\mathbf{n}}, \tag{4.25}$$

where $\mathcal{D}_d$ is the $d-$dimensional (or d-times repeated) DFT.

The outer integral of equation (4.13) is treated by the left-hand rectangle rule in accordance with the error analysis [33]. So, we have,

$$\begin{aligned} V(t, \mathbf{x_m}) &= e^{-r(T-t)} \mathcal{F}^{inv} \left(\widehat{\Phi}_{\mathbf{n}} \cdot \varphi_{\mathbf{n}} \right) \approx \frac{e^{-r(T-t)}}{(2\pi)^d} \int_{\mathbb{R}^d} \widehat{\Phi}_{\mathbf{n}} \varphi_{\mathbf{n}} e^{-i\boldsymbol{\omega}\mathbf{x}} d\boldsymbol{\omega} \\ &= d\boldsymbol{\Omega} \frac{e^{-r(T-t)}}{(2\pi)^d} \sum_{\mathbf{n=0}}^{\mathbf{N-1}} \widehat{\Phi}_{\mathbf{n}} \varphi_{\mathbf{n}} e^{-i\omega_{n_1} x_{m_1} - \ldots - i\omega_{n_d} x_{m_d}}. \end{aligned} \tag{4.26}$$

By using (4.19):

$$d\boldsymbol{\Omega} = \prod_{j=1}^{d} d\omega_j = \prod_{j=1}^{d} \frac{2\pi}{N_j dy_j} = \frac{(2\pi)^d}{N^d d\mathbf{Y}}. \tag{4.27}$$

Again we can replace,

[1]The twiddle factors in finance are the complex conjugate of the twiddle factors in most standard literature. There is no difference in the computations

$$e^{-i\omega_{n_j} x_{m_j}} = (-1)^{m_j}(-1)^{n_j+\frac{N_j}{2}} W_{N_j}^{n_j m_j}. \tag{4.28}$$

Combining (4.26), (4.27) and (4.28) gives:

$$V(t,\mathbf{x_m}) \approx \frac{e^{-r(T-t)}}{N^d d\mathbf{Y}} \sum_{\mathbf{n=0}}^{\mathbf{N-1}} \widehat{\Phi}_{\mathbf{n}} \cdot \varphi_{\mathbf{n}} . \prod_{j=1}^{d} (-1)^{m_j}(-1)^{n_j+\frac{N_j}{2}} W_{N_j}^{n_j m_j}. \tag{4.29}$$

We see that the products $(-1)^{n_j+\frac{1}{2}N_j}$ vanish when combining (4.29) and (4.22). The combination of the two discretised integrals leads to the *multi-dimensional CONV method*:

$$V(t,\mathbf{x_m}) = \frac{e^{-r(T-t)}}{(2\pi)^d}(-1)^{\mathbf{m}} \mathcal{D}_d^{inv} \left[\varphi_{\mathbf{n}} \mathcal{D}_d \left\{\Phi_{\mathbf{k}} G_{\mathbf{k}}\right\}_{\mathbf{n}}\right]_{\mathbf{m}}. \tag{4.30}$$

with $(-1)^{\mathbf{m}} = (-1)^{m_1}(-1)^{m_2}\ldots(-1)^{m_d}$.

4.3 The fast Fourier transform

4.3.1 Divide-and-conquer

The computation of the discrete Fourier transform in equation (4.30) can be performed by means of the fast Fourier transform (FFT). Before we discuss the general technique of the FFT, we discuss the complexity of the DFT itself. The discussion in this section is done for the one-dimensional case, but it can be generalised to the multi-dimensional case as well.

Let us recall the one-dimensional DFT of a vector with $N = 2^s$ components from equation (4.23):

$$\mathcal{D}\{f\}_n = \sum_{k=0}^{N-1} f_k e^{\frac{2\pi i n k}{N}} = \sum_{k=0}^{N-1} f_k W_N^{-nk} \tag{4.31}$$

where $W_N = e^{-\frac{2\pi i}{N}}$ were introduced in Section 4.2.4. These factors are called complex roots of unity or *twiddle factors*. These twiddle factors play an important role in Fourier analysis since the following properties hold:

$$|W| = \sqrt{W_N \overline{W_N}} = 1 \tag{4.32}$$

$$W_N^{\pm n} = 1. \tag{4.33}$$

Property (4.33) refers to the terminology of complex roots of unity. Since there are n different roots of unity $0 \leqslant n \leqslant N-1$, in the literature, these roots are often called as N complex $N-$th roots of unity. In our analysis we follow the books of [38, 50].

Proposition 4.3.1. *The computation of the discrete Fourier transform (4.31) can be interpreted as the evaluation of the polynomial*

$$v(x) = \sum_{k=0}^{N-1} f_k x^k \tag{4.34}$$

in the points $x = W_N^{-n}$. The evaluation of this polynomial in N distinct points has a computational complexity of N^2 elementary computations.

Proof. By substituting:

$$v(x) = v(W_N^{-n}) = \sum_{k=o}^{N-1} f_k (W_N^{-n})^k = \mathcal{D}(f) = F_n.$$

For every n, there are N multiplications needed with the coefficients of the polynomial and the twiddle factors. Hence to evaluate N distinct values of v, this is a total complexity of N^2 elementary multiplications. □

We see that a straightforward application of the DFT requires N^2 elementary computations. We will derive that the FFT algorithm requires $N \log_2 N$ computations if $N = 2^s$.

We evaluate the polynomial v in a different way. We split the coefficients f_k into two parts. An array with the even coefficients, f_{2k} and an array with the odd coefficients f_{2k+1}. Then we have:

$$\begin{aligned} v(x) = \sum_{k=0}^{N-1} f_k x^k &= \sum_{k=0}^{N/2-1} f_{2k} x^{2k} + \sum_{k=0}^{N/2-1} f_{2k+1} x^{2k+1} \\ &= \sum_{k=0}^{N/2-1} f_{2k} x^{2k} + x \sum_{k=0}^{N/2-1} f_{2k+1} x^{2k} \\ &= v^0(x^2) + x v^1(x^2). \end{aligned}$$

The computation of $v^0(x^2)$ and $v^1(x^2)$ are evaluations of polynomials with $N/2$ coefficients. For every distinct n, there are $N + 1$ elementary multiplications and one extra for the multiplication of x and $v^1(x^2)$. If this is done for N distinct points the computational complexity should be $N^2 + N$. However, we can make use of the nice properties of the twiddle factors.

Lemma 4.3.2 (Cancellation)**.** *For any integers $N, k \geqslant 0$ and $q > 0$:*

$$W_{qN}^{-qk} = W_N^{-k}$$

Proof.

$$W_{qN}^{-qk} = \left(e^{-\frac{2\pi i}{qN}}\right)^{-qk} = e^{\frac{2\pi i qk}{qN}} = \left(e^{-\frac{2\pi i}{N}}\right)^{-k} = W_N^{-k}$$ □

Theorem 4.3.3 (Halving theorem)**.** *If N is an even positive number, then the squares of the twiddle factors, W_N, based on N, are identical to the twiddle factors $W_{N/2}$ based on $N/2$.*

Proof. By the Cancellation Lemma 4.3.2, we have $\left(W_N^{-k}\right)^2 = W_{N/2}^{-k}$. If all twiddle factors are squared, then:

$$\left(W_N^{-(k+N/2)}\right)^2 = W_N^{2k+N} = W_N^{2k} W_N^N = W_N^{2k} = \left(W_N^{-k}\right)^2$$

since $W_N^N = 1$ by property (4.33). □

By the Halving Theorem 4.3.3, the set of twiddle factors $(W_N^0)^2, \ldots, (W_N^{-N+1})^2$ consists of only $N/2$ distinct twiddle factors. In other words, to evaluate the polynomial $v(x)$ in the twiddle factors W_N we only need to evaluate the polynomials $v^0(x)$ and $v^1(x)$ in the $N/2$ twiddle factors $W_{N/2}$. Hence this *divide-and-conquer* strategy saves us computations.

The combination of the two transforms of the even and odd components is done by the addition of the two arrays of size $N/2$. The coefficient of the transform of the odd components is x and this value describes the whole range of the N distinct twiddle factors. However, we have for $n \in [0, N/2-1]$:

$$W_N^{-n-N/2} = W_N^{-n} W_N^{-N/2} = W_N^{-n} e^{\frac{2\pi i N/2}{N}} = W_N^{-n} e^{\pi i} = -W_N^{-n}.$$

Then, for the computation of the DFT we have:

$$\begin{cases} F_n & = G_n^0 + W_N^{-n} G_n^1 \\ F_{n+N/2} & = G_n^0 - W_N^{-n} G_n^1 \end{cases} \tag{4.35}$$

with $0 \leqslant n \leqslant N/2 - 1$ and where

$$G_n^0 = \sum_{k=0}^{N/2-1} f_{2k} W_{N/2}^{-nk} \tag{4.36}$$

$$G_n^1 = \sum_{k=0}^{N/2-1} f_{2k+1} W_{N/2}^{-nk} \tag{4.37}$$

$$\tag{4.38}$$

are the DFTs of size $N/2$ of the even components (G^0) and odd points (G^1). Schematically, the first step of the divide-and-conquer strategy is illustrated in Figure 4.1.

This strategy can be continued to a next level. The polynomials v^0 and v^1 can be divided into two parts. If the coefficients f_{2k} are renamed into g_k^0,

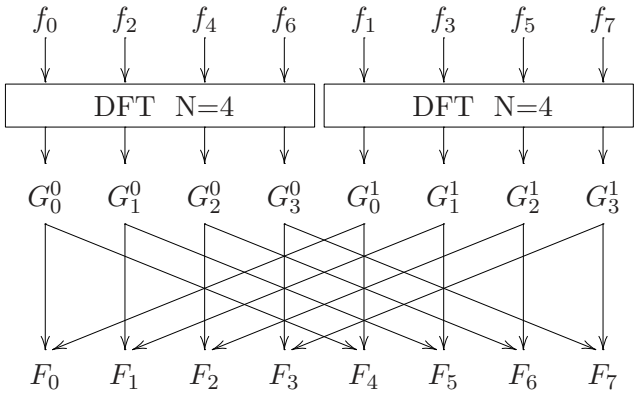


Figure 4.1: One step in the divide-and-conquer strategy for $N = 8$.

f_{2k+1} into g_k^1 and $x^2 = y$, then we have:

$$\begin{aligned}
v^0(x^2) &= \sum_{k=0}^{N/2-1} g_k^0 y^k = \sum_{k=0}^{N/4-1} g_{2k}^0 y^{2k} + \sum_{k=0}^{N/4-1} g_{2k+1}^0 y^{2k+1} \\
&= \sum_{k=0}^{N/4-1} g_{2k}^0 y^{2k} + y \sum_{k=0}^{N/4-1} g_{2k+1}^0 y^{2k} = \sum_{k=0}^{N/4-1} f_{4k} x^{4k} + x^2 \sum_{k=0}^{N/4-1} f_{4k+2} x^{4k} \\
&= v^{00}(x^4) + x^2 v^{10}(x^4). \\
v^1(x^2) &= \sum_{k=0}^{N/2-1} g_k^1 y^k = \sum_{k=0}^{N/4-1} g_{2k}^1 y^{2k} + \sum_{k=0}^{N/4-1} g_{2k+1}^1 y^{2k+1} \\
&= \sum_{k=0}^{N/4-1} g_{2k}^1 y^{2k} + y \sum_{k=0}^{N/4-1} g_{2k+1}^1 y^{2k} = \sum_{k=0}^{N/4-1} f_{4k+1} x^{4k} + x^2 \sum_{k=0}^{N/4-1} f_{4k+3} x^{4k} \\
&= v^{01}(x^4) + x^2 v^{11}(x^4).
\end{aligned}$$

The superscripts are ordered from left to right, hence v^{01} means that the first splitting is the odd elements and the second splitting takes the even elements from the new array: f_{2k+1}. Then we have:

$$\begin{aligned}
v(x) &= v^0(x^2) + x v^1(x^2) \\
&= v^{00}(x^2) + x^2 v^{10}(x^2) + x v^{01}(x^2) + x^3 v^{11}(x^2).
\end{aligned} \tag{4.39}$$

Again, by the Halving Theorem 4.3.3, the set of twiddle factors $(W_N^0)^2, \ldots, (W_{N/2}^{-N/2+1})^2$ consists of only $N/4$ distinct twiddle factors. In other words, to evaluate the polynomial $v^0(x)$ or $v^1(x)$ at the twiddle factors $W_{N/2}$ we need only to evaluate the polynomials $v^{00}(x)$ and $v^{10}(x)$ or $v^{01}(x)$ and $v^{11}(x)$ in the $N/4$ twiddle factors $W_{N/4}$. Hence again, the divide-and-conquer strategy cuts down computations.

Similarly to the previous case, the combination of the four transforms of components is done by the addition of the four arrays of size $N/4$. Let us define the DFT of the points $4k+p$ by H^p:

$$H^p = \sum_{k=0}^{N/4-1} f_{4k+p} W_N^{-4nk}. \tag{4.40}$$

According to the derivation of (4.39) and Proposition 4.3.1:, the computation of G_n^0 in equation (4.36) is now

$$G_n^0 = v^0(x^2) = v^{00}(W_N^{-4n}) + x^2 v^{10}(W_N^{-4n}) = H_n^0 + W_N^{-2n} H_n^2$$

for $0 \leqslant n \leqslant N/4 - 1$. For the points in G_n^0 with $N/4 \leqslant n \leqslant N/2 - 1$, we have:

$$H^p_{n+N/4} = \sum_{k=0}^{N/4-1} f_{4k+p} W_N^{-4(n+N/4)k} = \sum_{k=0}^{N/4-1} f_{4k+p} W_N^{-4nk} = H_n^p$$
$$W_N^{-2(n+N/4)} = W_N^{-2n} W_N^{-N/2} = -W_N^{-2n}.$$

Hence:

$$\begin{cases} G_n^0 & = H_n^0 + W_N^{-2n} H_n^2 \\ G^0_{n+N/4} & = H_n^0 - W_N^{-2n} H_n^2. \end{cases} \tag{4.41}$$

Similarly, for the computation of G_n^1 in equation (4.37), we have

$$G_n^1 = v^1(x^2) = v^{01}(W_N^{-4n}) + x^2 v^{11}(W_N^{-4n}) = H_n^1 + W_N^{-2n} H_n^3$$

and this is similar to:

$$\begin{cases} G_n^1 & = H_n^1 + W_N^{-2n} H_n^3 \\ G^1_{n+N/4} & = H_n^1 - W_N^{-2n} H_n^3. \end{cases} \tag{4.42}$$

Finally the computation of F_n is obtained by equation (4.35). The construction of the DFT of an array with $N = 8$ by using the divide-and-conquer strategy two times is presented in Figure 4.2.

Let $\mathcal{M}(s)$ be the number of elementary operations necessary to calculate a 2^s-point DFT according to the divide-and-conquer strategy with $N = 2^s$, then the number of elementary operations to calculate $v^0(x)$ and $v^1(x)$ costs in total $2\mathcal{M}(N/2) = 2\mathcal{M}(s-1)$. To calculate the combination in the points $x = W_{N/2}^0 = 1$ and $x = W_{N/2}^{N/2} = 1$, i.e. the components of the transformed vector F_0 and $F_{N/2}$ costs two elementary operations. The calculation of the components F_n and $F_{n+N/2}$ with $n = 1, 2, \ldots, N_2 - 1$ costs three elementary operations for each n, hence $3(N/2-1) = 3 \cdot 2^s - 3$. Hence the combination

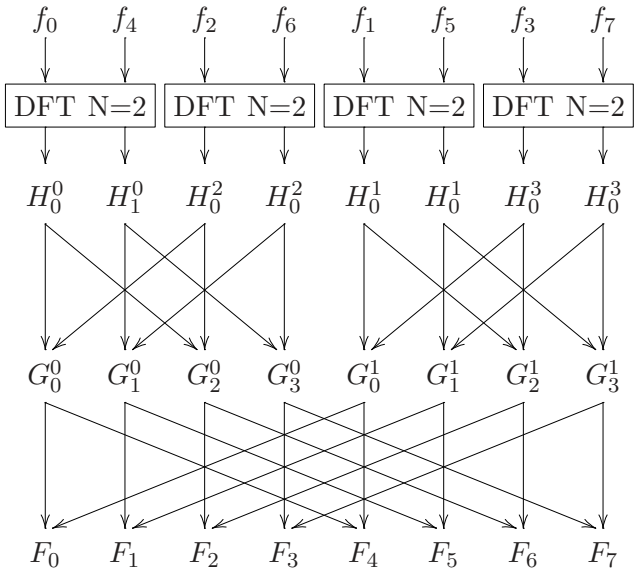


Figure 4.2: Two steps in the divide-and-conquer strategy for $N = 8$.

of v^0 and v^1 costs in total $3 \cdot 2^{s-1} - 1$. In total the number of elementary operations is a recurrence expression:

$$\mathcal{M}(s) = 2\mathcal{M}(s-1) + 3 \cdot 2^{s-1} - 1. \tag{4.43}$$

The computation of a two-point DFT (i.e. $N = 2$) is two elementary operations: $F_0 = f_0 + f_1$ and $F_1 = f_0 - f_1$, hence $\mathcal{M}(1) = 2$. Now the recurrence expression can be solved by induction.

Proposition 4.3.4. *The solution of equation (4.43) with* $\mathcal{M}(1) = 2$ *reads:*

$$\mathcal{M}(s) = (3s-2)2^{s-1} + 1. \tag{4.44}$$

Proof. $\mathcal{M}(1) = 2$ is a proper induction step, since $\mathcal{M}(0) = 0$. This follows both from equation (4.43) and (4.44). Now, we have:

$$\begin{aligned}
\mathcal{M}(s+1) &= (3(s+1)-2)2^s + 1 = 3s \cdot 2^s - 2^s + 1 \\
&= 6s2^{s-1} - 4 \cdot 2^{s-1} + 3 \cdot 2^s + 1 \\
&= 2\left(3(s-2)2^{m-1} + 1\right) + 3 \cdot 2^{s-1} - 1 \\
&= 2\mathcal{M}(s) + 3 \cdot 2^s - 1
\end{aligned}$$

□

Replacing s by $\log_2 N$, we have:

$$\mathcal{M}(N) = \mathbf{O}(N \log_2 N), \tag{4.45}$$

which is the expected computational complexity of a one-dimensional discrete Fourier transform by the application of the divide-and-conquer strategy. We will now discuss the algorithms which follow from the divide-and-conquer strategy.

4.3.2 The Cooley-Tukey algorithm

The divide-and-conquer algorithm can be implemented in many ways. For single processor operation, it does not matter if the algorithm is recursive or not. However, if the algorithm is implemented in parallel, an iterative algorithm is a better method of choice, since every part of an iteration step can be done in parallel.

From the divide-and-conquer algorithm, it follows that by successive application, the number of steps to construct the FFT is $s = \log_2 N$. We will denote every step in the algorithm by F^s. Each step, F^s, in the iterative algorithm is an update of the array with the transformed elements. However, if we investigate the flow chart of the data in Figure 4.2, we see that the input array is in a different order. This follows from the ordering via the separation of the even and odd components of the vector. The type of ordering is the so-called *bit reversal* step.

If the place k of an element in the array f is written in binary form, i.e. $f_k = f_{a_{s-1}a_{s-2}\ldots a_1 a_0}$, with

$$k = \sum_{j=0}^{s-1} a_j 2^j$$

then by bit reversal the component $f_{a_{s-1}a_{s-2}\ldots a_1 a_0}$ is moved to place $a_0 a_1 \ldots a_{s-2} a_{s-1}$ in the initial step of algorithm, i.e. F^0.

Example 4.3.5. *Let $N = 8$, then we have:*

$$\begin{aligned}
f_0 = f_{000} \to F^0_{000} = F^0_0 \quad & f_4 = f_{100} \to F^0_{001} = F^0_1 \\
f_1 = f_{001} \to F^0_{100} = F^0_4 \quad & f_5 = f_{101} \to F^0_{101} = F^0_5 \\
f_2 = f_{010} \to F^0_{010} = F^0_2 \quad & f_6 = f_{110} \to F^0_{011} = F^0_3 \\
f_3 = f_{011} \to F^0_{110} = F^0_6 \quad & f_7 = f_{111} \to F^0_{111} = F^0_7
\end{aligned}$$

and so the initialisation step leads to the array:

$$F^0 = [f_0, f_4, f_2, f_6, f_1, f_5, f_3, f_7].$$

Now, the array is well ordered to start the procedure to compute the DFT by the divide-and-conquer algorithm. The bit reversal and the combination by the divide-and-conquer strategy together are known as the FFT. If the bit-reversal is the initialisation step, then the algorithm is referred to as the *Cooley-Tukey* algorithm. This algorithm is a direct implementation of the

divide-and-conquer algorithm. In each step two components in the array are combined and stored in the new array.

Basically, each combination is done by a *butterfly pattern* as illustrated in Figure 4.3. The butterfly patterns present nicely the combination of two

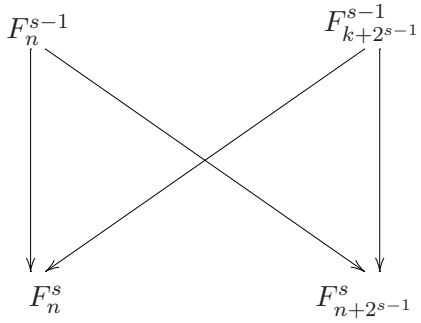


Figure 4.3: Illustration of the combination of two components of the array at level $s-1$ into level s.

elements from step $s-1$ into two new elements in level s. These butterfly patterns already occur in Figures 4.1 and 4.2. In this setting, the arrays G^0, G^1 defined in equations (4.36) and (4.37) and H^p in equation (4.40) are organised into F_n^s. For example one butterfly corresponding to equation (4.36) is presented in Figure 4.4 with the relation to F^s.

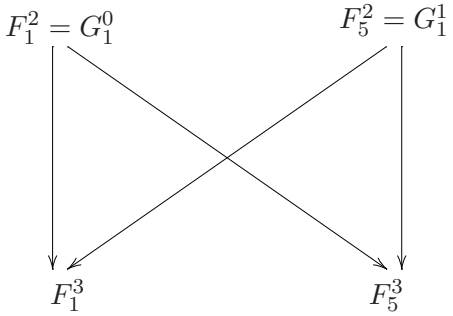


Figure 4.4: Illustration of the combination of two components of G into F from equation (4.36)

.

Remark 4.3.6. Note that in contrast to the arrays G^0, G^1 and H^p, the size of the arrays F^s is always N. At level $s = 1$ for $N = 8$, the structure of the array F^s reads:

$$F^1 = [H_0^0, H_1^0, H_0^2, H_1^2, H_0^1, H_1^1, H_0^3, H_1^3]$$

and for $s = 2$ it reads:

$$F^2 = [G_0^0, G_1^0, G_2^0, G_3^0, G_0^0, G_1^1, G_2^1, G_3^1]$$

The last step, F^3, is the final transform.

The iterative algorithm is presented in Algorithm 2. The butterfly

Algorithm 2: Cooley-Tukey algorithm

```
1  for s ← 1 to log_2 N do
2  |  W ← 1
3  |  W_s ← exp(2πi / 2^s)
4  |  for j ← 0 to 2^{s-1} − 1 do
5  |  |  for n ← j to N − 1 step 2^s do
6  |  |  |  u_1 ← F_n^{s-1}
7  |  |  |  u_2 ← W F_{n+2^{s-1}}^{s-1}
8  |  |  |  F_n^s ← u_1 + u_2
9  |  |  |  F_{n+2^{s-1}}^s ← u_1 − u_2
10 |  |  endfor
11 |  |  W ← W W_s
12 |  endfor
13 endfor
```

patterns are combined together in diagram 4.5 for $N = 8$.

Example 4.3.7. *We will now explain the flow pattern of Figure 4.5 and Algorithm 2. Consider the DFT of the vector $f = [1, 4, 2i, 3]$, so $N = 4$. Algorithm 2 for this vector implies:*

f	1	4	$2i$	3
F^0	1	$2i$	4	3
F^1	$1+2i$	$1-2i$	7	1
F^2	$8+2i$	$1-i$	$-6+2i$	$1-3i$

4.3.3 The Gentleman-Sande algorithm

A different algorithm to compute the DFT is the *Gentleman-Sande* algorithm. This algorithm is very similar to the Cooley-Tukey algorithm, but the bit-reversal is not applied at the start of the algorithm. This algorithm is also used to compute the inverse DFT which can be seen as a transpose of the original DFT. Furthermore the Gentleman-Sande algorithm is easy to derive parallel codes, because the bit-reversal procedure is omitted.

The first step is now the combination of the elements f_k and $f_{k+N/2}$ into $\widetilde{F}_k^1$ and $\widetilde{F}_{k+1}^1$. The combination is again done by butterfly patterns, but the combination equation is different. The Gentleman-Sande algorithm is

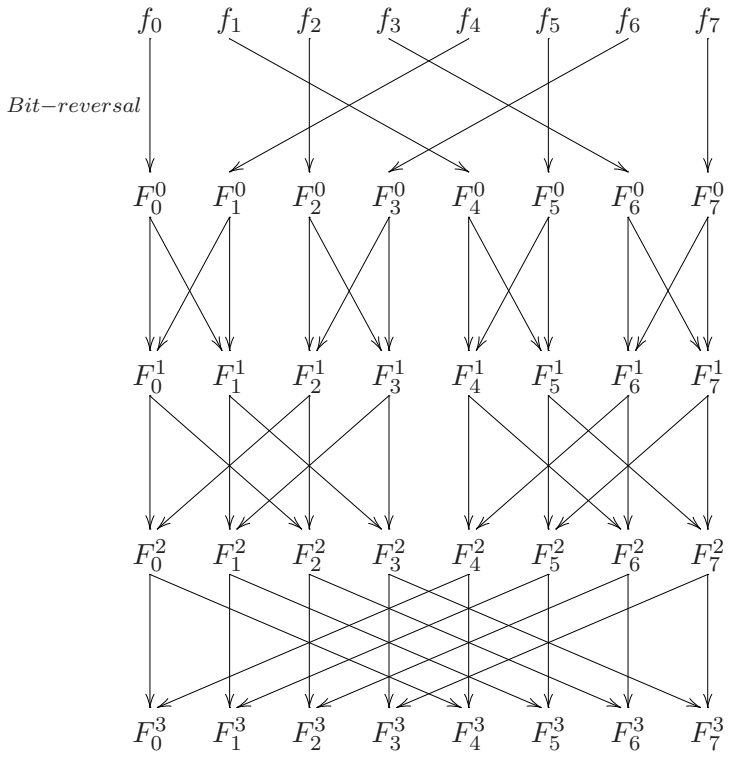


Figure 4.5: Pattern of the data-flow in the iterative divide-and-conquer algorithm for $N = 8$.

presented in Algorithm 3 and the flow chart for $N = 8$ is presented in Figure 4.6.

Algorithm 3: Gentleman-Sande algorithm

```
for s ← 1 to log_2 N do
    q ← log_2 N − s + 1
    W ← 1
    W_q ← exp(2πi / 2^q)
    for j ← 0 to 2^{q−1} − 1 do
        for n ← j to 2^q − 1 step 2^s do
            u_1 ← F_n^{s−1}
            u_2 ← F_{n+2^{q−1}}^{s−1}
            F_n^s ← u_1 + u_2
            F_{n+2^{q−1}}^s ← W(u_1 − u_2)
        endfor
        W ← W W_q
    endfor
endfor
```

If the Gentleman-Sande Algorithm 3 is compared to the Cooley-Tukey Algorithm 2 then there are similarities but there are also two major distinctions (besides the bit-reversal). First the traverse to the coefficients is in opposite direction: s runs from 1 to $\log_2 N$ and q from $\log_2 N$ to 1. Furthermore the multiplication with the twiddle factor in the Cooley-Tukey algorithm is different compared to the Gentleman-Sande algorithm.

The similarity is the result of the two algorithms. We can easily see that in the final step in the Gentleman-Sande algorithm, $\widetilde{F}_n^s$ is the bit-reversed transformed vector resulting from the Cooley-Tukey algorithm. For example with $N = 8$ and $n = 3$, we find according to Figure 4.6 and Algorithm 3:

$$\begin{aligned}
\widetilde{F}_3^3 &= W_2^{-2}(\widetilde{F}_2^2 - \widetilde{F}_3^2) \\
&= W_2^{-2}\left(W_4^0\left(\widetilde{F}_0^1 - \widetilde{F}_2^1\right) - W_4^{-1}\left(\widetilde{F}_1^1 - \widetilde{F}_3^1\right)\right) \\
&= W_2^{-2}\left(W_4^0\left(\widetilde{F}_0^0 + \widetilde{F}_4^0 - \widetilde{F}_2^0 - \widetilde{F}_6^0\right)\right) \\
&\quad - W_2^{-2}\left(W_4^{-1}\left(\widetilde{F}_1^0 + \widetilde{F}_3^0 - \widetilde{F}_3^0 - \widetilde{F}_7^0\right)\right) \\
&= f_0 - if_1 - f_2 + if_3 + f_4 - if_5 - f_6 + if_7
\end{aligned}$$

since $W_2^{-2} = W_4^0 = 1$ and $W_4^{-1} = i$. The Cooley-Tukey Algorithm 2 gives:

$$\begin{aligned} F_6^3 &= F_2^2 - W_8^{-2} F_6^2 \\ &= \left(F_0^1 - W_4^0 F_2^1\right) - W_8^{-2}\left(F_4^1 - W_4^{-4} F_6^1\right) \\ &= \left(\left(F_0^0 + W_2^0 F_1^0\right) - W_4^0\left(F_2^0 + W_2^2 F_3^0\right)\right) \\ &- W_8^{-2}\left(\left(F_4^0 + W_2^4 F_5^0\right) - W_4^{-4}\left(F_6^0 + W_2^6 F_7^0\right)\right) \\ &= f_0 - if_1 - f_2 + if_3 + f_4 - if_5 - f_6 + if_7 \end{aligned}$$

since $W_4^{-4} = 1$ and $W_8^{-2} = i$. Hence $\widetilde{F}_3 = F_6$ and vice-versa. Hence, if for some reason, a bit-reversal procedure is too expensive, the Gentleman-Sande algorithm is an option to use. It is obvious that the total complexity of both algorithms is the same. We will now continue with the parallelisation of the FFT and its multi-dimensional counterpart.

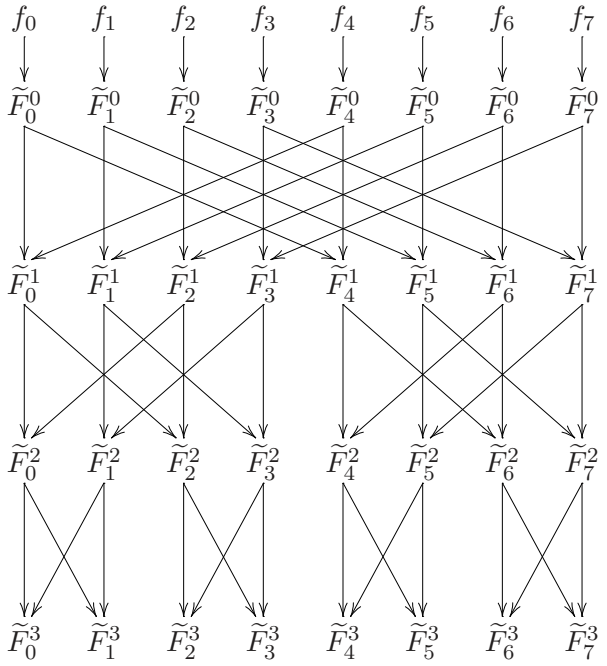


Figure 4.6: Flow chart of the Gentleman-Sande algorithm.

4.3.4 Parallelisation in general

Both the Cooley-Tukey algorithm and the Gentleman-Sande algorithm can be used in parallel. But first of all, some topics on parallelisation should be considered to make the proper choice in the parallelisation. Our aim is to solve (4.30). The first aim is to climb into the dimensions and thereafter to increase the speed.

Parallel computer architectures have been developing and changing rapidly. Since processors are much faster nowadays than a decade ago, the approach to parallelisation is different and depends on the structure of the parallel computer architecture. An algorithm for a system with many processors and a large shared memory is different from an algorithm for a so-called *cluster* of computers., i.e. a set of computers connected via a network or the Internet. The algorithms are not interchangeable in general.

Example 4.3.8. *A state-of-the-art sequential algorithm (i.e. an algorithm that can run on only one processor) is developed to store variables in the memory and to recall them later. However, a parallel computer architecture does not know where the stored values exist in the memory and other processors have to seek the variable. Therefore, many state-of-the-art sequential algorithms fail to work efficiently on a parallel computer architecture.*

There are two obvious reasons why an algorithm and/or parallel system may perform unsatisfactorily: load imbalance and communication overhead. Load imbalance means that some processors have to do much more work than most of the others. In this case, most processors have to wait for others to finish their computations before a data exchange can be performed. A purely sequential algorithm as described in Example 4.3.8 used in a parallel fashion may produce extreme load imbalance as only one processor is busy in that case.

Remark 4.3.9. For architectures with many processors (clusters), in general, it does not harm if some processors have less computational work to do than the average. However, it is crucial for the performance of the algorithm if one (or few) processors have to do much more computational work than the average. Then most of the processors will remain idle and have to wait for the overloaded ones to finish their parts of the computation.

Communication overhead means that the communication and data transfer between the processors takes too much time compared to the effective computing time. This overhead will even lead to slow-down instead of speed-up when more and more processors are used. In the high-dimensional case, the communication overhead is a reason that the performance of a parallel algorithm is bad, because the size of the discrete solution is running out of the memory. Therefore, developing an algorithm that can be divided in equivalent parts, such that each part has a minimum of data transfer, is the best option.

A grid partition is a suitable solution to reduce the demand for memory of a large problem, in particular, for the partial differential equation. If there are Q processors with a sufficient amount of memory, then a grid can be divided in Q sub-problems. Each sub-grid has different boundaries and they share the boundaries with other sub-grids. By advanced iterative methods, the problems are solved on each sub-grid and the necessary data

is transferred to other processors. The transmission is simultaneous and therefore the communication overhead is minimised.

The discrete Fourier transform, however, represents the computation of a large summation. Each element in the new vector depends on all elements from the original vector. Therefore, a straight-forward grid partition is not a proper choice, since any processor has to send the information to every other processor and a single processor has to receive the data from every other processor. It seems that the FFT can only work with communication. Hence we have to take care of the communication overhead.

4.3.5 Parallelisation of the FFT

We will now focus on the parallelisation of the Gentleman-Sande algorithm. The Cooley-Tukey algorithm is also suitable for parallelisation, but will be discussed later.

Consider again the flow chart for $N = 8$ and the case that we have two processors. We can send the original vector to the processors in pairs of two: Hence f_0, f_1, f_2, f_3 to processor 0, and f_4, f_5, f_5, f_7 to processor 1. Then the flow chart in Figure 4.7 shows that in the first level, processor 0 and processor 1 have to communicate their data. In the second and last steps, there is no communication.

We can easily generalise $N = 8$ in Figure 4.7 to N being a general power of two. If there are Q processors available (Q is a power of 2), then $\log_2 N/Q$ steps can be performed without communication. This part of the algorithm is the most efficient part of the complete parallel process.

In equation (4.45) we derived that the order of elementary computations is $N \log_2 N$. If Q processors are available then the complexity of the transform itself is $\mathcal{M} = 2N \log_2(N)/Q$. Now, we focus on the timing of the data transmission between the processors. First there is the start-up time for a single message, t_α and secondly the time needed to send a word per message, t_β. Each processor has to send one message per stage and there are $\log_2(Q)$ stages. Every message consists of N/Q words per message. This means that the start-up time is $t_\alpha \log_2(Q)$ and the transmission cost is $t_\beta N \log_2(Q)/Q$. We see that if t_β increases, the communication time will dominate the total time. In fact, in the multi-dimensional case, the time t_β is the important factor, because the size of the data is dependent of the vector size.

Another approach is to benefit from both the Gentleman-Sande algorithm and the Cooley-Tukey algorithm. We start with the distribution of the vector points according to the bit-reversal process. This is called a *cyclic* distribution. We then have components f_0 and f_4 on processor 0, f_1 and f_5 on processor 1 and so on. In fact this is the start of the Cooley-Tukey algorithm. Then after $\log_2(N/Q)$ steps we perform again a bit-reversal or transpose of the vector and we continue with the Gentleman-Sande algorithm.

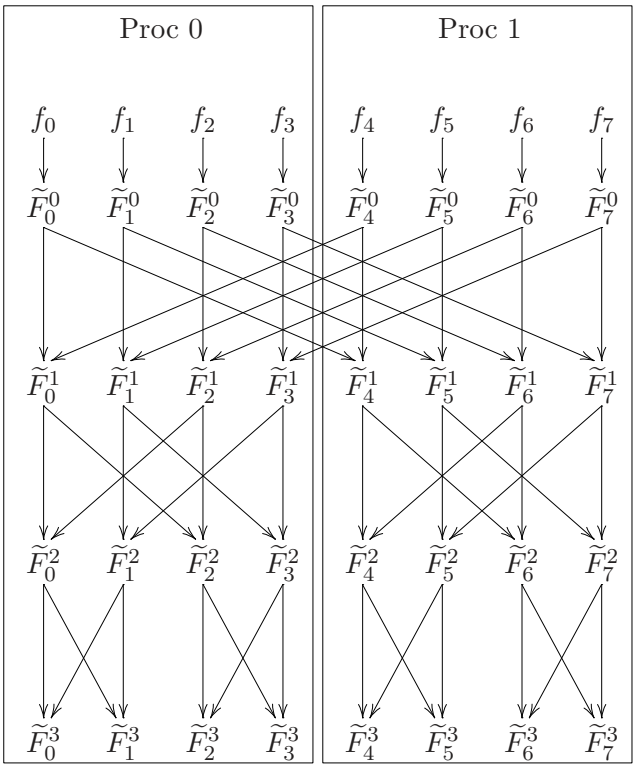


Figure 4.7: Flow chart of the Gentleman-Sande algorithm over two processors.

It is obvious that the computational part of this algorithm is the same as the standard parallelised Gentleman-Sande algorithm. However, the difference lies in the communication. Now, there is only a communication step after $\log_2(Q)$ steps. Note that for this algorithm $N > Q^2$, which is a common condition. If this condition is violated, then more reversals via communication are necessary to force all butterfly computations to stay local.

Comparing both algorithms, we see that the combined strategy sends fewer data overall, but sends $(Q-1)/\log_2 Q$ times as many messages. Thus, in the non-overlapping case, one expects the combined algorithm to do well in a low latency, high bandwidth environment, or when N is very large and Q small, and the algorithm described in Figure 4.8 does well in a high latency, low bandwidth environment, or when N is not as large but Q is large.

For a radically different way to parallelise the one-dimensional FFT, which uses the Fast Multi-pole Method to lower the communication needed when one insists on sorted inputs and outputs, see [17, 56].

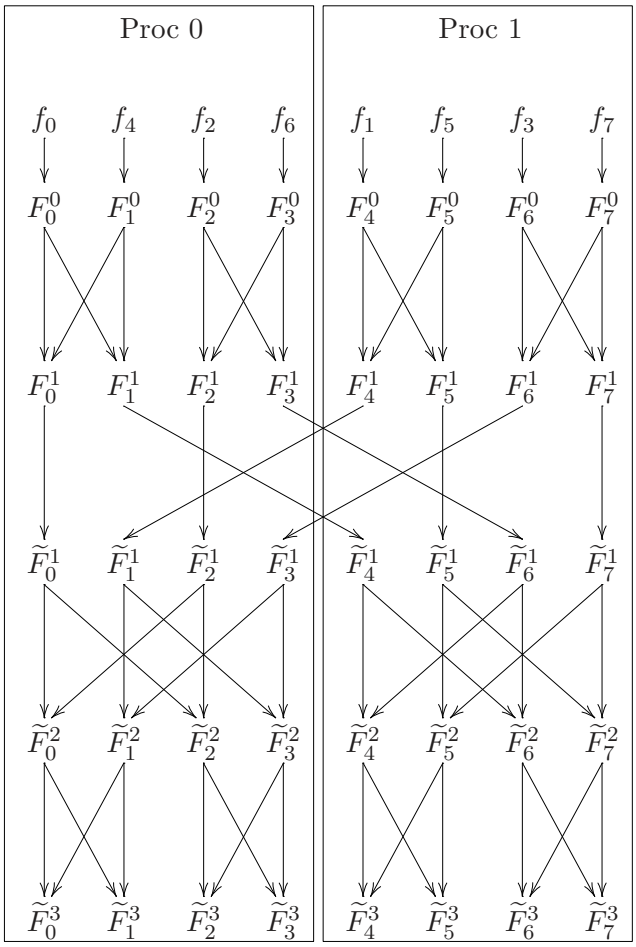


Figure 4.8: Flow chart of the combined algorithm over two processors with bit-reversal after $\log Q$ steps.

4.3.6 The FFT in more dimensions

FFTs in two or three dimensions are defined as one-dimensional FFTs for the vectors in all dimensions. We will discuss the two-dimensional FFT in some detail, and the three-dimensional case is analogous. A two-dimensional FFT requires one-dimensional FFTs on all rows and on all columns. Again, data layout is the main issue. In some applications, the data layout will be constrained by other parts of the application. For example, if the input is represented in a skewed layout, in which each processor owns $1/Q$ of every row and column, it may be better to redistribute the data before starting the FFT.

A straightforward implementation of a parallel two-dimensional FFT is based on a two-dimensional block layout. In fact, this is a special case of

the grid partitioning described in Section 4.3.4. For example, if 16 processors are available, then the processors can be organised as presented in Table 4.2. Since the elements of the input data can be organised in a ma-

Table 4.2: Processor ordering following the grid partitioning.

Proc 0	Proc 1	Proc 2	Proc 3
Proc 4	Proc 5	Proc 6	Proc 7
Proc 8	Proc 9	Proc 10	Proc 11
Proc 12	Proc 13	Proc 14	Proc 15

trix, it is easy to see that the elements $f_{0,0} \ldots, f_{0,3}$ are sent to processor 0, $f_{0,4} \ldots, f_{0,7}$ to processor 1, $f_{4,0} \ldots, f_{7,0}$ to processor 4, etc. In each processor the Gentleman-Sande algorithm can be applied. Then the data is bit-reversed and sent along the direction. For example, as soon as processor 0 is ready, this processor will sent data in $x-$bit reversed order to processor 4 and in $y-$bit reversed order to processor 1. It is easy to see that the communication overhead is larger then. The communication between the processors will increase if d and N increase and Q stays constant.

As mentioned earlier, the $d-$dimensional case is entirely analogous to the two-dimensional case. One can imagine using a $d-$dimensional blocked layout on a processor grid with $Q^{1/d}$ on each row of the grid. Note that the d dimensions need not be identical for the algorithm to work correctly.

Summarising the parallelisation for a multi-dimensional FFT, we conclude, that the parallelisation of the FFT itself requires substantial communication. Since communication is not an option for high-dimensional option pricing problems as large messages has to be sent and received, we will focus on a **redistribution of the data** by the use of the divide-and-conquer strategy.

4.3.7 Parallelisation of the CONV method

After this discussion of the parallelisation of the FFT itself, we will now focus on the underlying problem stated in equation (4.30):

$$V(t, \mathbf{x_m}) = \frac{e^{-r(T-t)}}{(2\pi)^d}(-1)^{\mathbf{m}} \mathcal{D}_d^{inv} \left[\varphi_{\mathbf{n}} \mathcal{D}_d \left\{\Phi_{\mathbf{k}} G_{\mathbf{k}}\right\}_{\mathbf{n}}\right]_{\mathbf{m}}.$$

We can replace the $\mathcal{D}$ and $\mathcal{D}^{inv}$ by the parallel versions of either the Gentleman-Sande algorithm or the combined algorithm. The curse of dimensionality forces us to parallelise the problem to make the problem manageable. This is a different issue for parallelisation than the speed-up. The amount of data communication suffers from the curse of dimensionality as

well. As shown in the previous section, a multi-dimensional FFT requires more transpose or bit-reversals operations than lower dimensional FFTs. Secondly, the data-flow to communicate will be extensively great (i.e. possibly in orders of gigabytes per communication step).

Thirdly, the multiplication with the characteristic function is a difficult operation to handle in the parallel setting. Due to the presence of correlation coefficients, the value of the characteristic function is dependent on every underlying coordinate. A straightforward choice is to compute the entire characteristic function in a $d-$ dimensional loop on every processor. This will certainly slow down the complete computation and is, therefore, not a good option.

Finally, the presence of the correlated data after multiplication with φ is also problematic, since the inverse transform has to be taken too. For European-style option contracts the final outcome of the inverse transform can be only one point on each processor. The communication a posteriori then consists of sending only Q numbers from the processors to the central processor.

These issues force us to apply the divided-and-conquer strategy directly to the entire problem. Consider first the one-dimensional version of equation (4.30):

$$H_m = \sum_{n=0}^{N-1} \varphi_n \widehat{\Phi}_n W_N^{mn}, \tag{4.46}$$

where we omit the discounting factor, for simplicity. As mentioned, we compute (4.46) as the combination of two sums of size $M = N/2$ with the even and odd points respectively:

$$\begin{aligned} H_m &= \sum_{n=0}^{M-1} \varphi_{2n} \widehat{\Phi}_{2n} W_N^{2nm} + \sum_{n=0}^{M-1} \varphi_{2n+1} \widehat{\Phi}_{2n+1} W_N^{(2n+1)m} \\ &= \sum_{n=0}^{M-1} \varphi_{2n} \widehat{\Phi}_{2n} W_M^{nm} + W_N^m \sum_{n=0}^{M-1} \varphi_{2n+1} \widehat{\Phi}_{2n+1} W_M^{nm}. \end{aligned} \tag{4.47}$$

Now the even and odd inner functions $\widehat{\Phi}_{2n}$ and $\widehat{\Phi}_{2n+1}$ are treated in the same way:

$$\begin{aligned} \widehat{\Phi}_{2n} &= \sum_{k=0}^{N-1} G_k \Phi_k W_N^{2nk}, \\ &= \sum_{k=0}^{M-1} G_k \Phi_k W_M^{nk} + \sum_{k=M}^{N-1} G_k \Phi_k W_M^{nk} \\ &= \sum_{k=0}^{M-1} G_k \Phi_k W_M^{nk} + W_M^{nM} \sum_{k=0}^{M-1} G_{k+M} \Phi_{k+M} W_M^{nk}, \end{aligned} \tag{4.48}$$

and for the odd elements:

$$\widehat{\Phi}_{2n+1} = \sum_{k=0}^{M-1} G_k \Phi_k W_N^k W_M^{nk} + W_N^M W_M^{nM} \sum_{k=0}^{M-1} G_{k+M} \Phi_{k+M} W_N^k W_M^{nk}. \quad (4.49)$$

We observe that (4.48) and (4.49) are again sums of two DFTs of size $N/2$. When the splittings (4.47), (4.48) and (4.49) are combined, we find the one-dimensional partitioned version of equation (4.46). In contrast to the divided-and-conquer strategy, we observe that the division is in four parts instead of two.

If the divide-and-conquer strategy is applied β times, with β a power of two then the size of the DFTs is $M = \beta^{-1}N$. The points used in the splitting of the inverse transform are given by $\beta n + q$, with $q \in [0, \beta - 1]$. So, the multiple partitioned version of equation (4.46) reads, using $W_N^{\beta nm} = W_M^{nm}$ by the Cancellation Lemma 4.3.2 of the twiddle factors.:

$$H_m = \sum_{q=0}^{\beta-1} \sum_{n=0}^{M-1} \varphi_{\beta n+q} \widehat{\Phi}_{\beta n+q} W_N^{m(\beta n+q)} = \sum_{q=0}^{\beta-1} W_N^{mq} \sum_{n=0}^{M-1} \varphi_{\beta n+q} \widehat{\Phi}_{\beta n+q} W_M^{mn}. \quad (4.50)$$

The partitioning into the odd and even parts can now be included:

$$\begin{aligned} \widehat{\Phi}_{\beta n+q} &= \sum_{k=0}^{N-1} \Phi_k G_k W_N^{-(\beta n+q)k} \\ &= \sum_{p=0}^{\beta-1} \sum_{k=0}^{M-1} \Phi_{k+pM} G_{k+pM} W_N^{-(\beta n+q)(k+pM)} \\ &= \sum_{p=0}^{\beta-1} W_\beta^{-pq} \sum_{k=0}^{M-1} \Phi_{k+pM} G_{k+pM} W_M^{-nk} W_N^{-qk}, \end{aligned} \quad (4.51)$$

where we used

$$W_N^{-\beta npM} = e^{2\pi i np} = 1 \quad \text{and} \quad W_N^{-pqM} = W_\beta^{-pq},$$

following from the Cancellation Lemma 4.3.2 and the Halving Theorem 4.3.3.

Combining (4.50) and (4.51), we obtain the one-dimensional multiple split version of equation (4.46):

$$\begin{aligned} H_m &= \sum_{q=0}^{\beta-1} W_N^{mq} \sum_{n=0}^{M-1} \varphi_{\beta n+q} W_M^{mn} \sum_{p=0}^{\beta-1} W_\beta^{-pq} \sum_{k=0}^{M-1} \Phi_{k+pM} G_{k+pM} W_M^{-nk} W_N^{-qk} \\ &= \sum_{p=0}^{\beta-1} \sum_{q=0}^{\beta-1} W_\beta^{-pq} W_N^{mq} \mathcal{D}^{inv} \left(\varphi_{\beta n+q} \mathcal{D} \left[\Phi_{k+pM} G_{k+pM} W_N^{-qk} \right] \right). \end{aligned} \quad (4.52)$$

This partitioning can be generalised to the multi-dimensional case. We then have a partitioning *vector* $\boldsymbol{\beta}$ containing β_j-parts for each coordinate j. The points in the outer transform (4.50) are represented by $\boldsymbol{\beta n} + \boldsymbol{q} = (\beta_1 n_1 + q_1, \ldots, \beta_d n_d + q_d)$. The points of the contract function addressed in (4.51) are represented by $\mathbf{k} + \mathbf{pM} = (k_1 + p_1 M_1, \ldots, k_d + p_d M_d)$. When using the multi-dimensional summation, the multiple partitioned version of equation (4.30) reads:

$$V(t, \mathbf{x}) = \frac{e^{-r(T-t)}}{(2\pi)^d} \sum_{\mathbf{p}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} \sum_{\mathbf{q}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} W_{\boldsymbol{\beta}}^{-\mathbf{pq}} W_{\mathbf{N}}^{\mathbf{mq}} \mathcal{D}_d^{inv} \left[\varphi_{\boldsymbol{\beta}\mathbf{n}+\mathbf{q}} \mathcal{D}_d \left\{ \Phi_{\mathbf{k}+\mathbf{pM}} G_{\mathbf{k}+\mathbf{pM}} W_{\mathbf{N}}^{-\mathbf{qk}} \right\} \right]. \tag{4.53}$$

We see that if one of the coordinates is split into two parts, i.e. $\beta_k = 2, \beta_{j\neq k} = 1$, the computation of equation (4.53) would be a combination of 4 DFTs of size $\frac{N_k}{2} \prod_{j=1, j\neq k}^{d} N_j$. If the same coordinate would be partitioned again, we would deal with 16 DFTs of size $\frac{N_k}{4} \prod_{j=1, j\neq k}^{d} N_j$. The parallel efficiency is low in this case, as the number of processors needed grows quadratically with β_j. Therefore, we rewrite equation (4.53) as,

$$V(t, \mathbf{x}) = \frac{e^{-r(T-t)}}{(2\pi)^d} \sum_{\mathbf{q}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} W_{\mathbf{N}}^{\mathbf{mq}} \mathcal{D}_d^{inv} \left[\varphi_{\boldsymbol{\beta}\mathbf{n}+\mathbf{q}} \mathcal{D}_d \left\{ \sum_{\mathbf{p}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} \Phi_{\mathbf{k}+\mathbf{pM}} G_{\mathbf{k}+\mathbf{pM}} W_{\mathbf{N}}^{-\mathbf{qk}} W_{\boldsymbol{\beta}}^{-\mathbf{pq}} \right\} \right]. \tag{4.54}$$

In this case the computations are partitioned over the $\mathbf{q}$-sum into $B = \prod_{j=1}^{d} \beta_j$ parts. Each processor now has to compute the $\mathbf{p}$ parts of the contract function. The summation over $\mathbf{p}$ could also be done in parallel with communication among the processors. As mentioned earlier, however, the drawback of allowing communication for high-dimensional problems is the need to transfer very large vectors from one processor to another. We certainly need the communication when solving *early exercise* options in parallel, but not for European options. So, early exercise options would be parallelised efficiently on a parallel machine with some form of shared memory. An alternative to this type of parallelisation could be the sparse grid method for which parallelisation is straightforward. We focus on the version of this parallel approach without any communication and solve European-style options with it.

4.3.8 Complexity analysis

We now evaluate the parallelisation technique to solve equation (4.30) by a complexity analysis, and first briefly summarise the procedure to solve (4.30) in Algorithm 4.

The construction of the contract function is a significantly faster procedure than the computation of the two FFTs and the multiplication by the characteristic function. We distinguish three portions of time consumption during the solution process of European options:

Algorithm 4: Multi-dimensional CONV-method

1 Compute the contract function on the tensor-product grid
2 Multiply the contract function by the function $G_{\mathbf{k}}$
3 Take the multi-dimensional FFT
4 Multiply the result by the characteristic function
5 Take the multi-dimensional inverse FFT
6 Multiply it by the discount factor.
7 For Bermudan options: Take the maximum of this value and the contract function at t_n. Repeat the procedure from step 2 until t_0 is reached

- T_{pay} is the time needed to construct the contract function including the multiplication with the function $G_{\mathbf{k}}$ and $W_{\mathbf{N}}^{-\mathbf{qk}}$ in (4.54).
- T_{four} is the time in steps 3 to 6 of the algorithm.
- T_{add} is the additional time needed for starting the computation, reading and writing files.

We assume here that T_{add} is negligible. The total time needed to compute equation (4.30) is then $T_{tot} \approx T_{pay} + T_{four}$. We further assume that $T_{four} = AT_{pay}$. If technique (4.54) is used and we partition the problem in B parts, the computational time per processor reads,

$$T_{tot,split} = T_{pay} + \frac{1}{B} T_{four} = \frac{A+B}{B} T_{pay}, \tag{4.55}$$

with $B = \prod_{j=1}^{d} \beta_j$. If there are Q identical processors available, then the parts B can be distributed over the Q processors. Ideally Q is a divisor of B. The number of parallel processes is therefore equal to $\lceil \frac{B}{Q} \rceil$ and the computational time reads:

$$T_{tot,split} = \left\lceil \frac{B}{Q} \right\rceil \frac{A+B}{B} T_{pay} \tag{4.56}$$

In our applications, typically, $A \in [4, 12]$ for $B = 1$.

4.3.9 Parallelisation of sparse grids

The partitioning of equation (4.30) to get (4.54) is not sufficient to deal with the curse of dimensionality. It helps to get problems of moderate size into the memory or to speed up the computation of medium-sized problems. However, the CONV method is just a quadrature technique and hence it can be combined with the sparse grid technique [45, 20, 55]. The sparse grid formulation from Section 3.3 can be used for the solution, where we

need a minimum number of four grid points per coordinate, because the FFT equals zero if $N = 2$ in one of the directions. Since the FFT itself is optimal if N is a power of two, we require $n_f \geqslant 3$. Hence, $c_i \geqslant 2$ or $b \geqslant 3$ (See Section 3.3.4). The parallelisation of the sparse grid technique is straightforward. Every problem is independent of the others and therefore the number of processors is not an issue.

If, however, a sub-problem does not fit into the memory, we additionally have to make use of the parallelisation strategy from Section 4.3.7, partitioning the multi-dimensional CONV method for all the sub-problems in a sparse grid layer. Let's consider, as an example, a seven-dimensional problem with $n_c = 10$ and $b = 2$. This problem requires 2^{28} grid points for the sub-problems in the top layer. The total number of sub-problems in that layer is 1716, but these sub-problems need to be partitioned once according to the divide-and-conquer strategy. Therefore, we deal with 3432 sub-problems of roughly 2^{27} points. The full grid problem would require 2^{70}-grid points (2^{43} GB), which is infeasible. The overall sparse grid complexity is 2^{39}-points, subdivided into 5147 sub-problems.

For a single-asset option with the asset modelled by geometric Brownian motion a second order full grid convergence was derived in [33]. We assume an error of $\mathbf{O}(\sum_{j=1}^{d} \Delta x_j^2)$ for the multi-dimensional problem. For the sparse grid integration technique, Gerstner [20] showed that the order of convergence is of $\mathbf{O}(\Delta x^2 \left(\log\left(\Delta x^{-1}\right)\right)^{d-1})$, for numerical integration problems with bounded mixed derivatives, similar to the error approximation of the PDE method (3.23). The accuracy for the sparse grid case depending on the maximum number of grid points in one direction ($N_c = 2^{n_c}$) can be related "globally" to the number of grid points $N_j = 2^{n_f}$ in the full grid case, as follows:

$$C_f 2^{-2n_f} = C_s 2^{-2n_c} n_c^{d-1}, \tag{4.57}$$

with constants C_f and C_s. The solution to this equation is given by:

$$n_c = \exp\left(-\mathcal{L}\left(-\frac{\ln 4}{d-1} e^{D}\right) - D\right),$$
$$D = \frac{-n_f \ln 4 + \ln C_f - \ln C_s}{d-1}$$

and $\mathcal{L}$ is the Lambert W function[2]. With this expression, we can compute the required number of grid points for a sparse grid computation to mimic a certain full grid problem with grid size n_f, given the desired accuracy ε, constant C_f and number n_f and the upper-bound of C_s. The constants C_f and C_s can be determined from a small-sized experiment taking into account that especially C_s is problem dependent.

[2]The Lambert W function is the solution of $\mathcal{L}(x)e^{\mathcal{L}(x)} = x$.

4.4 Numerical experiments

4.4.1 Full grid experiments

We evaluate the CPU-times of the parallel CONV method for some multi-dimensional experiments on tensor-product grids. We first evaluate the option on the geometric average for which we compare the numerical result with an analytic solution [4]. This solution can be obtained by the use of a coordinate transformation $y = \prod_{j=1}^{d} e^{\frac{x_j}{d}}$, see Section 1.4.2. In Table 4.3, the prices for the four-dimensional call option on the geometric average of the assets are presented for a different number of grid points. The first column in Table 4.3 represents the number of grid points per coordinate $N_j = 2^{n_f},\ j = 1, \ldots, 4$. The final computation, $n_f = 7$, requires 4 GB of memory and has a complexity of 2^{28}-points.

The option parameters chosen are $r = 0.06$, $\sigma_j = 0.2$, $\delta_j = 0.04$, $\rho_{jk} = 0.25$ if $j \neq k$ and $T = 1$. The strike price is €40, as is $\mathbf{S}(0)$.

The desired accuracy of errors being less than €0.01 is achieved for $n_f = 5$. We observe a second order convergence on the finer grids. The right-side part of Table 4.3 presents timings on a parallel machine, which consists of nodes with two processors each, having 8 GB of memory. Parameter A, as in (4.55) is also given.

Table 4.3: Option prices for the four-dimensional geometric average call option with the parallel timings (in sec.). Last column gives A (4.55).

$d = 4$	Call on the geometric average			CPU times with eq. (4.54)				
n_f	Price	Error	Ratio	B=1	B=2	B=4	B=16	A
3	1.962	2.0×10^{-1}	5.3	<0.1	<0.1	<0.1	<0.1	
4	2.128	3.8×10^{-2}	5.4	<0.1	<0.1	<0.1	<0.1	
5	2.156	9.3×10^{-3}	4.1	0.5	0.2	0.1	<0.1	4.5
6	2.163	2.3×10^{-3}	4.0	9.4	4.9	3.0	1.6	6.2
7	2.165	5.8×10^{-4}	4.0	164.1	85.1	45.2	25.2	7.1

Based on these results we conclude that the partitioning strategy reduces total CPU time well. Parallel efficiency would improve on finer grids and in higher dimensions.

We now consider problems that require more than the maximum available physical memory per processor. The parallel partitioning is then mandatory. In Table 4.4, we present the prices of a digital put on the geometric average of five assets. As the contract function of the digital option (it will pay an amount of €1 when the geometric average is less than the strike price in our experiment) has a discontinuity along the hyper-surface $\prod_{j=1}^{d} e^{\frac{x_j}{d}} = 1$,

it is expected that this leads to only first order error convergence. Table 4.4 indeed displays first order convergence (again the exact solution is known for the digital put on the geometric average) and we see that a grid size with $n_f = 6$ is not sufficient to reach the desired accuracy.

Table 4.5 presents the solution of a 6D standard basket put with equally weighted assets ($c_i = \frac{1}{6}$). The error convergence is irregular for this contract function, but at least of second order. The size of $n_f = 5$ is again sufficient to reach the desired accuracy. The CPU times for $B = 32$ in Tables 4.4 and 4.5 are estimated times when the number of processors Q is equal to the number of parts B.

Table 4.4: Option prices for the 5D geometric average digital put, plus parallel timing results and parameter A from (4.55).

$d = 5$	Digital put on the geom. average			CPU times		
n_f	Price	Error	Ratio	B=4	B=32	A
2	0.81	3.36×10^{-1}	1.49	<0.1	<0.1	
3	0.32	1.49×10^{-1}	2.26	<0.1	<0.1	
4	0.40	7.43×10^{-2}	2.00	0.2	0.1	4.0
5	0.43	3.71×10^{-2}	2.00	1.8	1.1	4.5
6	0.45	1.86×10^{-2}	2.00	295.6	91.1	8.7

Table 4.5: Option prices for the 6D basket put, plus parallel timing results.

$d = 6$	Basket put			CPU times	
n_f	Price	Error	Ratio	B=4	B=32
2	1.26	1.25		<0.01	<0.01
3	1.52	2.63×10^{-1}	4.7	0.09	<0.01
4	1.51	1.70×10^{-2}	15.5	5.02	1.2
5	1.50	2.62×10^{-3}	6.5	334.34	111.1

Finally, we also present the results of the hedge parameters. These equations can be discretised similarly to (4.30) into:

$$\Delta_k(t, \mathbf{x_m}) = -\frac{e^{-r(T-t)}}{(2\pi)^d S_k} \prod_{j=1}^{d} (-1)^{m_j} \mathcal{D}_d^{inv} \left[i\omega_{n_k} \phi_{\mathbf{n}} \mathcal{D}_d \left\{ V_{\mathbf{k}} G_{\mathbf{k}} \right\} \right], \quad (4.58)$$

$$\Gamma_k(t, \mathbf{x_m}) = \frac{e^{-r(T-t)}}{(2\pi)^d S_k^2} \prod_{j=1}^{d} (-1)^{m_j} \mathcal{D}_d^{inv} \left[\left(i\omega_{n_k} + \omega_{n_k}^2 \right) \phi_{\mathbf{n}} \mathcal{D}_d \left\{ V_{\mathbf{k}} G_{\mathbf{k}} \right\} \right]. \quad (4.59)$$

The hedge parameters are presented in Table 4.6.

Table 4.6: Hedge parameters of a standard 3D,4D and 5D basket call on a full grid of 2^{n_f} points per coordinate.

	3D		4D		5D	
n_f	Δ_1	error	Δ_1	error	Δ_1	error
3	0.1872		0.1369		0.1087	
4	0.1866	6.18×10^{-4}	0.1370	8.90×10^{-4}	0.1119	3.29×10^{-3}
5	0.1867	1.68×10^{-4}	0.1371	1.16×10^{-4}	0.1116	3.40×10^{-4}
6	0.1867	4.30×10^{-5}	0.1372	2.22×10^{-5}	0.1116	1.90×10^{-5}

4.4.2 Sparse grid computations

In this section, we will describe numerical experiments with the sparse grid technique to solve multi-asset options. The sparse grids technique should be chosen if the required memory or the required number of processors for the partitioned full grid version is just too large. However, as already mentioned, the efficient use of the sparse grid technique in computational finance is *seriously restricted* by the types of multi-asset option contracts in use. An acceptable accuracy with the sparse grids method can only be expected if the solution has bounded mixed derivatives. The contract functions of the examples presented in Tables 4.3, 4.4 and 4.5 do not have this property. It may be possible to transform a contract function so that the kink (or discontinuity for a digital option) is *aligned* with a grid line (see Chapter 3), but this cannot be done for every contract function. A call or put option based on the *maximum or minimum* of the underlying assets has its non-differentiability on grid lines, see Figure 1.6. It is therefore expected that these options can be handled well in the sparse grid setting.

In the sparse grid method, we use the CONV algorithm as in the full grid case. In fact, the sparse grid method can be coded as an outer loop running over all sub-problems. Within the loop, the CONV method is called with the desired grid parameters. We developed the algorithm so that if the sub-problems in the sparse grid method are additionally partitioned as described in Section 4.3.7. The number of parallel tasks increases to $U = \mathcal{N}B$, where B is the number of parts of a sub-problem. The algorithm loops over all tasks U. Every task is sent to a different processor as soon as the processor is available. The maximum number of tasks in a problem is limited to 2^{31} on a 32-bit machine and 2^{63} on a 64-bit machine. As soon as the problem is solved for a certain $\mathbf{q}$ (see equation (4.54)) and multi-index $\mathcal{I}$, the solution (an option value) is returned to the master process and summed. After this task is performed, a new task can be assigned to this processor. This kind of parallel coding is not straightforward, due to the three different types of partitioning within the algorithm, but it can be used on a heterogeneous

cluster.

We now perform numerical experiments with option contracts on the maximum or minimum of the underlying assets. The option parameters for these experiments are

- $K = 100$, $T = 1$ year, $r = 4.5\%$,.
- $\sigma_1 = 0.25, \sigma_2 = 0.35, , \sigma_3 = 0.20, , \sigma_4 = 0.25, , \sigma_5 = 0.20, , \sigma_6 = 0.21$ and $\sigma_7 = 0.27$.
- $\delta_1 = 0.05, \delta_2 = 0.07, \delta_3 = 0.04, \delta_4 = 0.06, \delta_5 = 0.04, \delta_6 = 0.03$ and $\delta_7 = 0.02$.
- $$R = \begin{pmatrix} 1.00 & -0.65 & 0.25 & 0.20 & 0.25 & -0.05 & 0.05 \\ -0.65 & 1.00 & 0.50 & 0.10 & 0.25 & 0.11 & -0.016 \\ 0.25 & 0.50 & 1.00 & 0.37 & 0.25 & 0.21 & 0.076 \\ 0.20 & 0.10 & 0.37 & 1.00 & 0.25 & 0.27 & 0.13 \\ 0.25 & 0.25 & 0.25 & 0.25 & 1.00 & 0.14 & -0.04 \\ -0.05 & 0.11 & 0.21 & 0.27 & 0.14 & 1.00 & 0.19 \\ 0.05 & -0.016 & 0.076 & 0.13 & -0.04 & 0.19 & 1.00 \end{pmatrix}.$$

with R the matrix with the correlation coefficients ρ_{jk}.

We start with the four-dimensional problem and compare the sparse and the full grid results. The parameters for this experiment are listed above, where we take the first four subscript entries. In Table 4.7, the results for the full grid experiment are presented for a European and a Bermudan style contract. The Bermudan style contract has ten exercise dates during the lifetime of the option contract. The results are presented for grids with $N_j = 2^{n_f}, \ j = 1, \ldots, d$ points. We see a smooth convergence for this type of European contract and an accuracy better than €0.01 when $n_f = 7$. For the Bermudan contract, the convergence ratio is less smooth, but the accuracy is again satisfactory.

Remark 4.4.1. In order to price American options, based on the present approach for Bermudan options, there are basically two approaches. One can compute a Bermudan option with many exercise dates, and thus very small time steps, as an approximation of an American option, or one can apply a repeated Richardson extrapolation. These two approaches have been applied to the univariate case in [33], in which it was shown that the Richardson extrapolation was superior in terms of accuracy and CPU time. The convergence of the Bermudan approach with many exercise dates was only of first order, whereas the Richardson extrapolation was significantly better than that. However, in [33] a very regular convergence of the pricing for Bermudan options with the CONV method was achieved by shifting the grids, so that the option value where the continuation and the payoff values coincide was placed at a grid point. This was the reason for the accurate

American options prices in [33]. In the multivariate case, however, it is not possible anymore to place an "early-exercise-line", or even a higher-dimensional entity, completely on a grid line. Therefore we cannot achieve a regular convergence for Bermudan options and extrapolation cannot be used to price American options. American multi-asset options are, however, still uncommon financial contracts nowadays.

Table 4.7: European and Bermudan 4D put option on the maximum of the underlying assets on a full grid.The Bermudan contract has 10 exercise dates.

$d = 4$	European			Bermudan		
n_f	price	error	Ratio	price	error	Ratio
3	0.38	7.38×10^{-1}		0.61	5.06×10^{-1}	
4	0.87	2.51×10^{-1}	2.94	0.53	9.25×10^{-1}	0.55
5	1.05	6.82×10^{-2}	3.67	1.87	3.36×10^{-1}	2.75
6	1.10	1.76×10^{-2}	3.88	1.85	2.88×10^{-2}	11.67
7	1.11	4.51×10^{-3}	3.90	1.84	5.37×10^{-3}	5.36

Although the results are satisfactory in the four-dimensional case, the Bermudan option contract, for example, cannot be computed with $n_f = 7$ in higher-dimensional cases without any form of communication among the processors. In Section 4.3.9, we derived an expression to compute the number of grid points $N_s = 2^{n_c}$ needed for mimicking the solution with sparse grid, given the accuracy and the number of grid points $N_f = 2^{n_f}$ in the full grid. The estimate of the number of grid points is very global and we assume the value of C_f as an upper bound for C_s. If the desired accuracy is $\approx 10^{-3}$, we need in the full grid $n_f = 8$ by applying the convergence ratio. From the results in Table 4.7, we have with $n_f = 7$ $C_f \approx 74$. Solving equation (4.57), the number of grid points for the sparse grid computation is $n_c = 13$, to have an accuracy of 10^{-3}. The choice of the base b in the sparse grid technique has a major influence on the complexity. The CONV method does not work if one of the coordinates is discretised in only two points, so $b \geqslant 2$. It is also reasonable for accuracy reasons to use a higher base [32], for example $b = 3$ which means at least 8 points per coordinate. This however has a significant impact on the cost of the method.

In Table 4.8, the results for the put option on the maximum of the underlying assets are presented in the four- and five-dimensional case and for European and Bermudan style based on a sparse grid technique with $b = 3$. In this table, the first column represents the mimic of the full grid. With $n_c = 13$, we conclude that the desired accuracy of 10^{-3} is reached. The sparse grid method also converges to the same value as the full grid case (see Table 4.7), although the convergence tends to be of first order and

Table 4.8: Sparse grid results of a 4D and 5D put option on the maximum of the assets

	European 4D			European 5D		
n_c	Price	Error	Ratio	Price	Error	Ratio
7	1.0488	5.25×10^{-2}		0.7407	3.48×10^{-2}	
8	1.0785	2.97×10^{-2}	1.77	0.7696	2.90×10^{-2}	1.20
9	1.0992	2.06×10^{-2}	1.44	0.7875	1.79×10^{-2}	1.62
10	1.1061	6.88×10^{-3}	3.00	0.7967	9.21×10^{-3}	1.94
11	1.1107	4.68×10^{-3}	1.47	0.8015	4.77×10^{-3}	1.93
12	1.1134	2.62×10^{-3}	1.79	0.8035	2.04×10^{-3}	2.34
13	1.1146	1.22×10^{-3}	2.15	0.8046	1.09×10^{-3}	1.34
	Bermudan 4D			Bermudan 5D		
n_c	Price	Error	Ratio	Price	Error	Ratio
10	1.830	1.34×10^{-2}	3.68	1.389	5.65×10^{-2}	0.32
11	1.838	5.09×10^{-3}	2.63	1.380	8.49×10^{-3}	6.66
12	1.840	3.18×10^{-3}	1.60	1.375	5.16×10^{-3}	1.65
13	1.841	2.56×10^{-3}	1.24	1.378	2.24×10^{-3}	2.30

irregular. For the five asset problem (right part of Table 4.8), we see the same behaviour of the four asset problem by means of accuracy and convergence. Finally, the Bermudan option contract also reaches the desired accuracy when $n_c = 13$.

Remark 4.4.2. The sparse grid convergence in Table 4.8 is somewhat irregular. Option pricing problems are typically characterised by payoff functions that do not have bounded mixed derivatives. The max option, however, has this feature only along its axes (see Figure 1.6). So, in principle we would expect the asymptotic theoretical optimal sparse grid convergence of $\mathbf{O}(h^2(\log h)^{d-1})$. In Chapter 3, it was however shown, for the multi-dimensional Poisson equation and a smooth solution that the theoretical sparse grid convergence was only achieved on relatively fine grids. Here we have the same situation. For the 4D case of Table 5 we observe significantly improved sparse grid convergence rates on finer grids. For $n_s = 14$, the error is 5.0×10^{-4} with convergence ratio 2.43; for $n_s = 15$ we have error 1.9×10^{-4} and ratio 2.79, for $n_s = 16$, the error equals 6.0×10^{-5}, and ratio 3.1, while for $n_s = 17$ the error is 1.8×10^{-5} and the ratio is 3.36.

Analogous option contracts are presented in Table 4.9. Also in this table we see the same convergence and accuracy results. Furthermore in Table 4.10, the sparse grid values of the hedge parameters for a put option on the minimum are presented. These values have the same behaviour as the option prices themselves.

Table 4.9: 4D and 5D sparse grid prices on the maximum or minimum of the assets

n_c	4D Call on maximum		4D Call on minimum		4D Put on minimum	
10	25.344	1.01×10^{-2}	0.284	4.12×10^{-3}	24.640	6.67×10^{-3}
11	25.340	4.67×10^{-3}	0.287	2.11×10^{-3}	24.637	3.24×10^{-3}
12	25.341	1.62×10^{-3}	0.288	1.74×10^{-3}	24.639	2.18×10^{-3}
13	25.340	1.07×10^{-3}	0.289	5.50×10^{-4}	24.638	1.37×10^{-3}
n_c	5D Call on maximum		5D Call on minimum		5D Put on minimum	
10	27.065	1.45×10^{-2}	0.232	4.97×10^{-3}	25.217	1.06×10^{-2}
11	27.059	5.79×10^{-3}	0.234	2.35×10^{-3}	25.214	2.61×10^{-3}
12	27.063	2.99×10^{-3}	0.236	1.84×10^{-3}	25.216	2.38×10^{-3}
13	27.061	1.92×10^{-3}	0.237	8.98×10^{-4}	25.215	1.78×10^{-3}

The interesting point is the CPU time. In Table 4.7, we have an accuracy of 4.5×10^{-3} when $n_f = 7$ for the full grid European four-dimensional option. The CPU time on 16 equivalent processors for the full grid problem (see Table 4.3) is 25 seconds. We have an accuracy of 4.68×10^{-3} with $n_c = 11$ in Table 6 for the same option, but now in sparse grid case. In Table 4.11, the parameters are presented for the four-dimensional sparse grid case with $n_c = 11$. Each row in Table 4.11 represents a layer of the combination technique with the complexity, value of ℓ and number of sub-problems. The right part gives the CPU times for each sub-problem of a specific layer, the layer's total sequential CPU time and parallel CPU time when 12 CPUs are used. The total time on a single computer is 124.2 seconds and on a heterogeneous cluster 11.1 seconds. The efficiency is 11.23, which is high as 12 is the ideal case. Also we see that the CPU time in the sparse grid on 12 CPUs is even lower than the CPU time for the full grid case on 16 CPUs (25 seconds, see Table 4.3). We conclude that the sparse grid technique is an efficient method to use in parallel on a low number of CPUs. The problem size of the partitioned full grid is still large. For example, the problem size of a problem in the top layer of the five-dimensional 2^{13} mimic in Table 4.8 has a total complexity of 2^{25} or 512 MB.

We conclude our sparse grid results with the higher-dimensional examples of the option contracts on the maximum or minimum of the assets. In Table 4.12, the results of the sparse grid computation are presented for a put option on the maximum and minimum of the six or seven underlying assets. We again see a satisfactory accuracy with $n_c = 10$ and an irregular convergence. The seven-dimensional sparse grid problem with $n_c = 10$ uses the sparse grid technique as well as the partition technique in Section 4.3.7, because the maximum available memory is 2GB on our heterogeneous

Table 4.10: Hedge parameters for the 4D put option on the minimum of the assets in full and sparse grid

	Full grid Put on the minimum			
n_f	Δ_1	error	Γ_1	error
5	-0.2232	8.99×10^{-3}	9.823×10^{-3}	4.12×10^{-4}
6	-0.2223	9.91×10^{-4}	9.765×10^{-3}	5.80×10^{-5}
7	-0.2222	9.97×10^{-5}	9.751×10^{-3}	1.47×10^{-5}
	Sparse grid Put on the minimum			
n_c	Δ_1	error	Γ_1	error
6	-0.2232	2.38×10^{-3}	9.661×10^{-3}	1.74×10^{-4}
7	-0.2211	2.18×10^{-3}	9.672×10^{-3}	1.14×10^{-4}
8	-0.2222	1.14×10^{-3}	9.774×10^{-3}	1.01×10^{-4}
9	-0.2224	2.14×10^{-4}	9.755×10^{-3}	1.86×10^{-5}
10	-0.2222	1.63×10^{-4}	9.752×10^{-3}	3.13×10^{-6}

Table 4.11: Problem parameters for sparse grid (mimic of the 4D 2^{11} full grid)

Problem parameters				CPU times		
j	ℓ	Complex.	#probl	Problem	Layer time	Q=12
1	17	2^{20}	165	0.51	82.5	7.3
2	16	2^{19}	120	0.24	28.8	2.5
3	15	2^{18}	84	0.12	10.1	1.0
4	14	2^{17}	56	0.05	2.88	0.3
Tot			425		124.2	11.1

cluster. The problem size of this experiment in the top layer is 4 GB and therefore it is partitioned with $B = 2$. Again the base of the sparse grid technique is set to $b = 3$. The hedge parameters can also be computed with the sparse grid technique.

4.5 Conclusions

The multi-dimensional CONV method is a powerful and fast method. It is able to price multi-asset options of European and early-exercise type under Lévy price dynamics, including geometric Brownian motion, and to compute the hedge parameters. The partitioning of the method enables us to distribute some multi-dimensional partitioned parts over a system of parallel computers, which speeds up the computation. Since we chose to avoid

	6D Put on minimum			6D Put on maximum		
n_c	Price	Error	Ratio	Price	Error	Ratio
7	27.093	1.43×10^{-1}		0.375	2.33×10^{-2}	
8	27.183	9.02×10^{-2}	1.58	0.396	2.13×10^{-2}	1.09
9	27.141	4.21×10^{-2}	2.14	0.412	1.50×10^{-2}	1.42
10	27.158	1.73×10^{-2}	2.43	0.420	8.89×10^{-3}	1.69
	7D Put on minimum			7D Put on maximum		
n_c	Price	Error	Ratio	Price	Error	Ratio
7	26.153	1.22×10^{-1}		0.179	1.45×10^{-2}	
8	26.217	6.31×10^{-2}	1.93	0.194	1.50×10^{-2}	0.96
9	26.189	2.72×10^{-2}	2.32	0.206	1.14×10^{-2}	1.31
10	26.203	1.34×10^{-2}	2.02	0.213	7.21×10^{-3}	1.58

Table 4.12: Sparse grid results of two types of 6D and 7D European contracts

communication in the parallel system, and thus require some additional computation, the parallel efficiency is not optimal.

With the help of the sparse grid technique, we can climb in the number of dimensions of the multi-dimensional contracts. The size of the target problem depends, of course, on the number of parallel processors that are at one's disposal. An important remark is, however, that the basic sparse grid technique, without any enhancements, can only be successfully applied for certain contract functions. We show that the min- and max-asset options exhibit a satisfactory sparse grid convergence. The parallel efficiency of the sparse grid method is excellent, as each sub-problem can be computed independently.

For *high*-dimensional problems it becomes necessary to combine the parallel sparse grid method with the parallel version of the CONV method.

A logical next step in our research would be to evaluate the resulting parallel method on a machine containing a significant amount of parallel processors.

Chapter 5

Conclusions

Multi-asset option pricing with numerical techniques is a challenging extension of the numerical pricing techniques for the options on one underlying asset. The finite difference matrices are extended to a higher dimensionality with the use of Kronecker products. The underlying discretisation is not different from the one-dimensional discretisation, provided a consistent high-dimensional grid-ordering is adopted. The mixed derivatives in the multi-dimensional Black-Scholes partial differential equation can be handled as well with Kronecker products with the standard finite differences for the first derivative with respect to two different coordinates.

The non-differentiability of the contract function is a typical property of option pricing problems. This requires careful discretisation in space and time in order to obtain satisfactory accuracy results. Fourth order accurate space and time discretisations were proposed for the single-asset option, using spatial grid stretching by means of an analytical coordinate transformation. With the proper choices of grid and stretching parameters, the fourth order accuracy could be achieved. Important for the applications is, however, a small discretisation error with only a few grid points. This was achieved by the techniques proposed with a moderate grid stretching. Furthermore, we have observed a satisfactory accuracy in the hedge parameters and in the solution of early exercise options with and without dividend payments.

The high-dimensional option pricing problem is not necessarily more difficult from a mathematical point of view than a low-dimensional problem. If some special techniques are necessary to obtain accuracy in a low-dimensional problem, the same techniques can be extended to a high-dimensional problem in a straightforward way. In particular, we think of the early exercise solution strategy and the grid stretching.

However, from a computational point of view, the complexity of the problem seriously increases when the dimensionality increases. This is sometimes called as the curse of dimensionality, i.e. the exponential growth of the

number of grid points on tensor-product grids,when d increases. On modern computer systems, the amount of data can become too large to deal with. Advanced techniques, such as domain decomposition and problem parallelisation, do not solve this feasibility problem satisfactorily. If, for example, a strategy works nicely for a four-dimensional problem, it may fail for a six-dimensional problem, due to the computer storage restrictions.

One of the possible solutions for the curse of dimensionality is the sparse grid technique. The sparse grid mimics the solution on a full grid by combining the solutions on significantly smaller problems. Because every problem is independent from all others, parallelisation is straightforward.

The sparse grid technique proposed here, is an extension of the basic sparse grid technique developed in [10]. The sparse grid technique is generalised to rectangular grids with different numbers of grid points in each direction and not necessarily powers of two. With these extensions, the sparse grid technique is used for pricing multi-asset options with the multi-dimensional PDE method and the FFT method. The restrictions to the sparse grid technique are important. The major one being that the mixed derivatives of the solution should be bounded. Since the final conditions of the option pricing contracts are typically non-differentiable, this is not guaranteed.

For pricing basket options with the multi-dimensional Black-Scholes partial differential equation with the sparse grid combination technique, a linear or a non-linear coordinate transformation can be employed in order to align the payoff function with a grid line. This alignment improves the convergence of sparse grids significantly. With the coordinate transformations it is possible to reduce the number of grid points in some coordinates, which is highly advantageous from a computational point of view. An additional coordinate stretching function concentrates points in the region around the exercise price. The effect of grid stretching is mainly significant on these non-equidistant grids if the maturity time is short (as then steep gradients in the solution may occur).

For pricing basket options with the multi-dimensional Black-Scholes FFT based method, the sparse grid technique in combination with the parallel divide-and-conquer strategy gives a powerful and fast parallel algorithm. Although the number of grid points is significantly higher than for the PDE method, the computational speed is higher too. We have shown that the min- and max-asset options exhibit a satisfactory sparse grid convergence as these contract functions give rise to solutions with bounded mixed derivatives.

The PDE and the FFT methods are completely different, but comparable. The PDE method is a flexible method. The complexity and thus the computational time does not increase significantly when applying for example early exercise and dividend payments. The computational time of the FFT method, however, increases substantially in the case of early exercise.

In one-dimensional problems, the computational speed for early exercise problems is comparable. However, in the multi-dimensional case, the FFT is a much faster approach because there is no matrix to construct. Furthermore for the European-style options, there is no time-integration necessary and a linear system need not be solved for every time step. However, for certain special "state-dependent" volatility models the PDE method can be used without a greater computational complication. In summary, still the FFT method has the greatest potential for the fast and efficient pricing of higher-dimensional option pricing problems.

Further research to the multi-dimensional option pricing problem is still necessary. The curse of dimensionality is not completely broken by the sparse grid technique, due to its serious restrictions. A promising technique is the *principal component* analysis [42]. Basically, this technique is a dimension reduction for solving problems higher than ten dimensions.

The coordinate transformations are also of serious interest for the FFT method. If the characteristic function can be transformed, then it may be possible to align the basket sum along a grid line and use the transformed characteristic functions in combination with the sparse grid method.

Local volatility models or other asset dependent volatilities and correlations can be implemented into the PDE method straightforwardly. The Kronecker products themselves are usable linear operations which can also be used on grid functions. Furthermore, the improvement of the linear solver in the PDE method is a topic of interest. New iterative solvers may speed up the computation significantly.

Bibliography

[1] G. Bakshi and Z. Chen. An alternative valuation model for contingent claims. *J. Fin. Econometrics,*, 44:123–165, 1997.

[2] G. Barone-Adesi. The valuation of American call options and the expected ex-dividend stock price decline. *J. of Fin. Econ.*, 17:91–111, 1986.

[3] R. Bellman. *Adaptive control processes; a guided tour*. Princeton university press, 1961.

[4] S.J. Berridge and J.M. Schumacher. An irregular grid method for high-dimensional free-boundary problems in finance. *Future Generation Computer Systems*, 20(3):353–362, 2004.

[5] H. bin Zubair, C.C.W. Leentvaar, and C.W. Oosterlee. Efficient d-multigrid preconditioners for sparse-grid solution of high-dimensional partial differential equations. *International Journal of Computer Mathematics.*, 84(8):1129–1147, 2007.

[6] H. bin Zubair, C.W. Oosterlee, and R. Wienands. Multigrid for high-dimensional elliptic partial differential equations on non-equidistant grids. *SIAM J. Sci. Comput.*, 29(4):1616–1636, 2007.

[7] T. Björk. *Arbitrage theory in continuous time*. Oxford university press, 1998.

[8] F. Black and M. Scholes. The pricing of options and corporate liabilities. *J. Pol. Econ.*, 81(3):637–654, 1973.

[9] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. A proof of convergence for the combination technique for the Laplace equation using tools of symbolic computation. *Math. Comp. in Simulation*, 42:595–605, 1996.

[10] H.J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, pages 147–269, May 2004.

[11] H.J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for the Laplace equation. *East-West J. Numer. Math.*, 2:21–45, 1994.

[12] P. Carr and D. Madan. Option valuation using the fast Fourier transform. *Journal of Computational Finance*, 2:61–73, 1998.

[13] N. Clarke and A. K. Parrot. Multigrid for American option pricing with stochastic volatility. *Appl. math. finance*, 6:177–179, 1999.

[14] C.W. Cryer. The solution of a quadratic programnming problem using systematic overrelaxation. *SIAM J. Control*, 9:385–392, 1971.

[15] J.Y. Datey, G. Gauthier, and J.G. Simonato. The performance of analytical approximations for the computation of Asian quanto-basket option prices. *Multinational Finance journal*, 7:55–82, 2003.

[16] M.A.H. Dempster and S.S.G. Hong. Spread option valuation and the fast fourier transform. *Techn. Rep. WP 26/2000 the Judge Inst. Manag. Studies, Univ. of Cambridge*, 2000.

[17] A. Edelman, P. McCorquodale, and S. Toledo. The future Fast Fourier transform? *SIAM Journal on Scientific Computing*, 20:1094–1114, 1999.

[18] E.G.Haug. *The complete guide to option pricing formulas.* McGraw-Hill, 1997.

[19] D. Gentle. Basket weaving. *Risk*, 6:51–52, 1993.

[20] T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18(3–4):209–232, 1998.

[21] J. Gil-Pelaez. Note on the inverse theorem. *Biometrika*, 37:481–482, 1951.

[22] M.B. Giles and R. Carter. Convergence analysis of Crank-Nicolson and Rannacher time-marching. *Report 05/16, Oxford University*, 2005.

[23] A. Graham. *Kronecker products and matrix calculus with applications.* Ellis Horwood Ltd, 1981.

[24] R.M. Gray and J.W. Goodman. *Fourier transforms.* Kluwer Dordrecht, 1995.

[25] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In *Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra*, pages 263–281. Elsevier, Amsterdam, 1992.

[26] E. Hairer, Nörsett, and K. Wanner. *Solving ordinary differential equations. Vol. 1. Non-stiff problems.* Springer Verlag, Heidelberg, 1996.

[27] E. Hairer and K. Wanner. *Solving ordinary differential equations. Vol. 2. Stiff and differential-algebraic problems.* Springer Verlag, Heidelberg, 1996.

[28] P. Hoffmann. Asymptotic expansions of the discretization error of boundary value problems in rectangular domains. *Numerische Mathematik*, 9:302–322, 1967.

[29] J.C. Hull. *Options, futures and other derivatives.* Pearson Prentice Hall, 6th edition, 2005.

[30] R. Kangro and R. Nicolaides. Far field boundary conditions for Black-Scholes equations. *SIAM J. Numer. Anal.*, 38(4):1357–1368, 2000.

[31] Y.K. Kwok. *Mathematical models of financial derivatives.* Springer Verlag, 1998.

[32] C.C.W. Leentvaar and C.W. Oosterlee. On coordinate transformation and grid stretching for sparse grid pricing of basket options. *J. Comp. Appl. Math*, to appear, 2008.

[33] R. Lord, F. Fang, F. Bervoets, and C.W. Oosterlee. A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes. *SIAM, J. Sci. Comput.*, to appear 2008.

[34] G.H. Meyer. Numerical investigation of early exercise in American puts with discrete dividends. *J. Comp. Finance*, 5:37–53, 2002.

[35] B. Øksendahl. *Stochastic differential equations, an introduction with applications.* Universitext. Springer Verlag, Heidelberg, 6th edition, 2005.

[36] C.W. Oosterlee, C.C.W. Leentvaar, and A. Almendral Vzquez. Pricing options with discrete dividends by high order finite differences and grid stretching. In *Proc. ECCOMAS 2004, P. Neittaanmäkki et al. (eds.)*, Jyväskylä, Finland, 2004.

[37] D. Pooley. *Numerical methods for nonlinear equations in option pricing.* PhD thesis, Univ. Waterloo, Waterloo (Canada), 2003.

[38] M.J. Quinn. *Parallel computing.* McGraw-Hill, 2nd edition, 1994.

[39] R. Rannacher. Finite element solution of diffusion problems with irregular data. *Numer. Math.*, 43(2):309–327, 1984.

[40] C. Reisinger. *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben*. PhD thesis, In German. Naturwissenschaftlich-Mathematischen Gesamtfakultät der Ruprecht-Karls-Universität Heidelberg, 2004.

[41] C. Reisinger and G. Wittum. On multigrid for anisotropic equations and variational inequalities: pricing multi-dimensional european and American options. *Computation and Visualisation in Science*, 7:189–197, 2004.

[42] C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM J. Sci. Comput.*, 29(1):440–458, 2007.

[43] L. Scott. Pricing stock options in a jump-diffusion model with stochastic volatility and interest rates: Application of Fourier inversion methods. *Math. Finance*, 7:413–426, 1997.

[44] R. U. Seydel. *Tools for computational finance*. Universitext. Springer, 3rd edition, 2006.

[45] P.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk. USSR*, 4:240–243, 1963.

[46] W.H. Steeb. *Kronecker product of matrix and applications*. Wissenschaftsverlag, 1991.

[47] R.M. Stulz. Options on the minimum or maximum of two risky assets. *Journal of financial markets*, 10:161–185, 1982.

[48] D. Tavella and C. Randall. *Pricing financial instruments, the finite difference method*. Wiley, New York, 2000.

[49] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2000.

[50] C. van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 2nd edition, 1992.

[51] P. Wilmott. *Paul Wilmott introduces quantitative finance*. Wiley, 2001.

[52] P. Wilmott, S. Howison, and J Dewynne. *The mathematics of financial derivatives, a student introduction*. Cambridge university press, Cambridge, 1995.

[53] H. Windcliff, P. Forsyth, and K. Vetzal. Analysis of the stability of the linear boundary condition for the Black-Scholes equation,. *J. Computational Finance*, 8(1):65–92, 2004.

[54] J. Wu, N.B. Mehta, and J. Zhang. A flexible approximation of the sum of log-normal distributed values. *Glob. Telecom. Conf. (GLOBECOM)*, 6:3413–3417, 2005.

[55] C. Zenger. Sparse grids. In *Proceedings of the 6th GAMM seminar, Notes on numerical fluid mechanics*, volume 31. Birckhauser Verlag, 1990.

[56] F. Zhao and S. Johnsson. The parallel multipole method on the connection machine. *SIAM J.Sci. Comput.*, 12(6):1420–1437, 1991.

[57] Y. Zhu, X. Wu, and I. Chen. *Derivative securities and difference methods*. Springer Verlag, New York, 2004.

Publications

- C. W. Oosterlee, C.W. Leentvaar and A. Almendral: Pricing options with discrete dividends by high order finite differences and grid stretching. In *proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2004, P. Neittaanmaki et al. (eds.)*, Jyvaskyla, 24–28 July 2004.

- C.C.W. Leentvaar and C.W. Oosterlee: American options with discrete dividends solved by highly accurate discretizations. In *di A Bucchianico, RMM Mattheij, and MA Peletier, eds, Progress in Industrial Mathematics*, pages 427-431. Springer, Berlin, 2006.

- C.C.W. Leentvaar and C.W. Oosterlee: Pricing Multi-Asset Options with Sparse Grids and Fourth Order Finite Differences. In *A Bermudez de Castro and D Gomez (Eds.), Proceedings of ENUMATH 2005* (pp. 975-986). Heidelberg: Springer. (TUD), 2006

- H. Bin Zubair, C. C.W. Leentvaar and C.W. Oosterlee: Efficient d-multigrid preconditioners for sparse-grid solution of high-dimensional partial differential equations. *Int. J. of Computer Mathematics.* 84(8): 1129-1147, 2007.

- C.C.W. Leentvaar and C.W. Oosterlee: On coordinate transformation and grid stretching for sparse grid pricing of basket options. TU Delft internal report 06-13, and *J. Comp. Appl. Math*, to appear 2008.

- H. Bin Zubair, C.C.W. Leentvaar and C.W. Oosterlee. Multigrid preconditioners for Bi-CGSTAB for the sparse grid solution of high-dimensional anisotropic diffusion equation. *The J. of Prime Research in Mathematics. Abdus Salam School of Mathmatical Sciences, GC University, Lahore, Pakistan.* 3: 186-200, 2007.

- C.C.W. Leentvaar and C.W. Oosterlee. Multi-asset option pricing using a parallel Fourier-based technique. TU Delft internal report 07-12, Submitted for publication.

Curriculum Vitae

Coen Leentvaar werd geboren op 23 september 1976, te 's Gravenhage. Na zijn middelbare school opleiding begon hij als student in 1994 aan de opleiding Technische Natuurkunde aan de Technische Universiteit in Delft. In 2000 behaalde hij daarvoor zijn ingenieurstitel op een onderwerp uit de fysische transportverschijnselen met als afstudeertitel: *Globale en lokale hydrodynamica van een air-lift-loop reactor*. Na het voltooien was hij inmiddels begonnen met een opleiding tot kerkmusicus op het gebied van orgel en koordirectie.

De wiskunde heeft altijd al een belangrijke interesse bij hem opgewekt en in 2000 startte hij met het volgen van enige vakken in de Technische Wiskunde aan de Technische Universiteit Delft. In december 2003 behaalde hij zijn tweede ingenieurstitel in de numerieke wiskunde op het afstudeerwerk getiteld *Numerical solution of the Black-Scholes equation*.

In januari 2004 startte hij met het promotiewerk aan dezelfde afdeling bij wiskunde onder begeleiding van Prof. Oosterlee op het gebied van opties op meerdere aandelen.

Dankwoord

Dit proefschrift had niet tot stand gekomen zonder de hulp van veel mensen. Een aantal wil ik graag met name noemen, waarbij ik diegenen die ik niet noem, geen tekort wil doen.

Allereerst gaat mijn dank uit naar mijn promotor Prof. Oosterlee. Kees, bedankt dat je me in de afgelopen jaren de ruimte hebt gegeven om dit onderzoek te doen. Bedankt ook, voor de opbouwende kritiek, de opmerkingen tijdens het schrijven van het proefschrift en discussies, die verhelderend waren. Mede door jouw bemiddeling heb ik ook mooie dingen tijdens mijn promotie mogen beleven, zoals Santiago de Compostela, Japan en Canada.

Mijn dank gaat ook uit naar de collegae van de afdeling numerieke wiskunde. In het bijzonder zeg ik graag *Shukria* tegen Hisham bin Zubair, die mij heeft geholpen met de multigrid code, de samenwerking bij een van de artikelen en de suggesties bij het schrijven in het Engels.

Jij, Diana Droog, was altijd de vrolijke noot op de afdeling. Moest er iets geregeld worden, dan wist Diana raad. Bedankt!

De parallelle code voor de Fourier transformatie had ik niet zo snel kunnen schrijven zonder de hulp van Kees Lemmens. Kees Lemmens, je zult bij mij bekend blijven als de grote voorstander van open source software waaronder Linux. Als je een probleempje had dan kon je het wel aan Kees Lemmens vragen.

Zonder jou, Eef Hartman, zou ik niet meer zo enthousiast zijn over schaken. De pauzes vulden wij nogal eens met een of meerdere potjes schaak, waarbij de ene blunder nog erger was dan de andere. Maar, niet alleen het schaken, ook je assistentie bij de problemen met de computers zal ik me blijven herinneren.

Jasper Anderluh, jij hebt in mij een zekere mini-belegger wakker gemaakt. Door het inzicht vanuit de directe handel, heb ik daarbij ook een andere kijk op de opties en aandelen gekregen, die ik thans in mijn werk nog steeds gebruik. Bedankt voor de fijne samenwerking en discussies.

Als laatste, maar zeker niet het minst gaat mijn dank uit naar mijn ouders, Odilia en Edwin voor jullie interesse in mijn werk en voor de steun die jullie mij hebben gegeven tijdens de uitvoering daarvan.

Coen Leentvaar, April 2008.