Numerical Analysis of Jump-Diffusion Models for Option Pricing

Arne Karsten Strauss

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mathematics

Dr. Ekkehard W. Sachs, Chair
Dr. Slimane Adjerid
Dr. Christopher Beattie

July 7, 2006
Blacksburg, Virginia

Keywords: Option pricing, Jump-diffusion processes, Finite differences, Conjugate Gradient method, Fast Fourier Transform

Abstract

Numerical Analysis of Jump-Diffusion Models
for Option Pricing

Arne Karsten Strauss

Jump-diffusion models can under certain assumptions be expressed as partial integro-differential equations (PIDE). Such a PIDE typically involves a convection term and a nonlocal integral like for the here considered models of Merton and Kou. We transform the PIDE to eliminate the convection term, discretize it implicitly using finite differences and the second order backward difference formula (BDF2) on a uniform grid. The arising dense linear system is solved by an iterative method, either a splitting technique or a circulant preconditioned conjugate gradient method. Exploiting the Fast Fourier Transform (FFT) yields the solution in only $O(n \log n)$ operations and just some vectors need to be stored. Second order accuracy is obtained on the whole computational domain for Merton's model whereas for Kou's model first order is obtained on the whole computational domain and second order locally around the strike price. The solution for the PIDE with convection term can oscillate in a neighborhood of the strike price depending on the choice of parameters, whereas the solution obtained from the transformed problem is stabilized.

To my parents

Acknowledgements

I wish to express my gratitude to Professor Dr. Ekkehard W. Sachs for being my thesis advisor and for contributing valuable ideas that formed the frame of this work. His lectures on numerical analysis in particular concerning linear systems encouraged me to tackle this task. Also I would like to thank Professor Dr. Christopher Beattie for his interesting lecture on numerical analysis which was likewise highly useful for the thesis, and my study advisor Professor Dr. Slimane Adjerid for kindly helping in all study related affairs.

Further I gratefully acknowledge the financial support of the Mathematics Department of Virginia Polytechnic Institute and State University, without which this research and my whole stay in the United States would not have been possible.

Last but not least I thank Axel Heesen, Andre Lörx and Jan Sterbutzel for pointing out typos and making sure that I said what I meant and meant what I said.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 On the background of option pricing

To set the scene we first introduce the concept of an option, describe the most important types and illuminate its history and purpose.
A *call* (*put*) *option* is the right but not the obligation to buy (sell) under specific conditions a specific amount of an *underlying asset* for a specified price, called the *strike price*. The underlying asset can be a stock or bond, for example. To illustrate how this works let us assume the underlying asset is a stock with price S_t and that we are dealing with a so-called *European option*, that means the option can only be exercised at an a-priori specified time T, also called *expiry* or *maturity* of the option. Then the corresponding payoff function in time T would depend on the strike price K and the current underlying price S_T, so if the stock price is higher than the strike this difference is our payoff, whereas in the opposite case with $S_T \leq K$, the payoff will be zero since then we do not exercise the option. Thus the payoff function for this example is described by $(S_T - K)^+$.
Besides European options there are also *American*, *Asian* and *Bermudan* options. These types categorize options with respect to their exercise conditions: European options permit exercising only at the specified expiration date, whereas American options can be exercised at any time before expiry. Bermudan options can be exercised at specific dates or time periods, that means they can be seen as a mixture between

American and European ones. For Asian options the payoff depends on the average price of a stock. Of course, these names have no geographical meaning; for example most options traded in Europe are of American style.

Options were first traded in a standardized form on April 26, 1973 with the opening of the Chicago Board Options Exchange (CBOE). In 1975 the Black-Scholes model was adopted by the CBOE and in 1977 the first put options were traded. Since then the interest in options has excessively grown from only 911 contracts on 16 underlying stocks on the first day of trade in 1973 to 468,249,301 contracts with total notional value of more than $ 12 trillion in 2005 solely at the CBOE.

Why are options so attractive? Well, basically for two reasons: They are highly useful for hedging on the one hand, and for speculation on the other. Hedging means that an investor reduces the risk of his portfolio by using appropriate financial instruments. For example, if he fears that the price of a certain stock in his portfolio might fall he can buy a put option on this stock which guarantees him that he can sell it at the specified strike price and thus limit his risk of loss.

Naturally, since an option incorporates a right rather than an obligation for its buyer, he will have to pay a price for it. And now we arrived at the fundamental question of how to find a fair price for this derivative. It is not obvious what this price should be, except at maturity where it is just the payoff. *Black and Scholes* [1973] developed a formula for finding the price for a European call which became famous in particular because of its ease with regard to practical implementation and was used since 1975, like mentioned above, by the CBOE. But despite its high acceptance it has well-known defects, for example that the price path of the underlying asset is assumed to be continuous whereas in reality jumps can occur, mostly because of the arrival of new information like an hostile takeover offer. Furthermore, empirical investigations indicate that the log-returns are not normal distributed but rather possess a higher peak and fatter tails. Accordingly, many alternative models have been proposed to capture these empirical facts, among them stochastic volatility models (*Hull and White* [1987], *Heston* [1993], *Bates* [1996]), Lévy models (*Chan* [1999]) and jump-diffusion models (*Merton* [1976], *Andersen and Andreasen* [2000], *Kou* [2002]). Models with jumps can be categorized in jump-diffusion models with finite activity and models with infinite activity. In jump-diffusion models the log-price of the underlying asset follows most of the time a diffusion process, similar to the Black-Scholes model. Only at some random time points jumps occur which can be modeled using a compound

Poisson process. This approach can be reasoned by the observation that most of the time the price only changes marginally in a small time period due to changes in the economic outlook, for example. This behavior will be mirrored by the standard geometric Brownian motion. But every now and then, the announcement of a special event, mostly concerning just one firm individually, causes the stock price to jump by a non-marginal amount. An example could be the loss or win of a court case. The kind of processes and distributions used will depend on the considered model. In this thesis, the models at stake are Merton's and Kou's model, where a compound Poisson process is responsible for the jump events. The jump sizes are assumed to follow a normal distribution for Merton's model, and a double exponential distribution for Kou's. A problem with jump diffusion models is that in general they do not yield a closed-form solution; instead one has to solve them numerically. In these two special cases, however, it is possible to express the solution in terms of infinite series. This enables us in particular to compare the numerical results to the analytical solution, in order to get an idea of the error.
There are also models with an infinite number of jumps in any interval which are therefore called infinite activity models. Here the price behavior on small time intervals is modeled by jumps and Brownian motion is no longer needed. However, in the framework of this thesis we will deal only with jump diffusion. Pure jump models with finite activity are also possible but, according to *Cont and Tankov* [2004], they do not lead to a realistic description of price dynamics.

1.2 Outline and current status of research

Jump-diffusion models are not only attractive because this mixture of Brownian motion and discrete occurrence of jumps intuitively makes sense. Moreover, empirical evidence suggests that this approach also leads to results that can explain observed features like the non-normal log-returns and the volatility smile. *Andersen and Andreasen* [2000] numerically fitted a jump-diffusion model to the S&P500 market and found that it "produces almost perfect stationary S&P500 volatility skews" with stable parameters. Therefore jump-diffusion models attracted much attention especially in recent years from several groups of researchers. Under certain assumptions the jump-diffusion models lead to a partial integro-differential equation (PIDE) involving

a non-local integral term. Some models allow analytical solutions, mostly for European vanilla options to be found, for example *Merton* [1976] and *Kou* [2002]. *Lewis* [2001] proposed a general semi-analytical solution using Fourier analysis. We will use this latter solution approach for Kou's model to provide a benchmark for measuring the error of our numerical approximations. However, note that numerical methods are still needed since they provide a solution over the whole time horizon rather than only at one point in time, are sometimes faster than the analytical ones and, in particular, are applicable to PIDEs representing more enhanced options. For example, according to *Kou* [2002] there does not seem to exist an analytical solution for barrier options for the normal jump-diffusion model, but the PIDE formulations are almost the same. Essentially, one only has to change the boundary conditions, see *Cont and Tankov* [2004].

In order to solve the PIDE problem numerically, *Andersen and Andreasen* [2000] use an unconditionally stable ADI finite difference method and accelerate it using the fast Fourier transform (FFT). In *Matache et al.* [2004], *Matache et al.* [2005a] and *Matache et al.* [2005b] wavelet methods are applied to infinite activity jump-diffusion models. Several researchers recently used implicit-explicit finite difference methods, for example *Briani* [2003], *Briani et al.* [2004] and *Cont and Voltchkova* [2005]. They avoid dense linear systems that would arise from the integral discretization by approximating it explicitly, and using implicit schemes for the second order differential term in the PIDE. *Briani* [2003] approximates the convection term explicitly as well, whereas *Cont and Voltchkova* [2005] treat the whole differential part implicitly. However, *Cont and Voltchkova* [2005] need to use a low order upwind scheme in order to avoid oscillations.

Implicit finite difference approaches in combination with FFT have been undertaken in *d'Halluin et al.* [2004] extended by a penalty method to price American options and in *d'Halluin et al.* [2005] where the correlation integral is evaluated using an FFT method. They also resort in both papers to first order forward or backward differences for the convection term depending on some coefficients. *Almendral and Oosterlee* [2005] present both an implicit discretization of the PIDE on a uniform grid using finite differences as well as a finite elements approach, where a splitting technique again combined with FFT is used to accelerate the dense matrix-vector product. Central differences are used to approximate the convection term and oscillations of the solution can appear. Finally, *Ikonen and Toivanen* [2006] use a finite difference

approach on a non-uniform grid refined around the strike price using the Rannacher time-stepping scheme and a splitting technique like in *Almendral and Oosterlee* [2005]. They obtain fast convergence but cannot use FFT due to the non-uniform grid.

In this work we improve the results of *Almendral and Oosterlee* [2005] in that a transformation of the PIDE is shown that eliminates the convection term and thus avoids possible oscillations in the solution and additionally in that we apply a circulant preconditioned conjugate gradient method that turns out to be slightly faster than their fixed point iteration for Merton's model. In particular, for Merton's model second order accuracy is obtained on the whole computational domain. They stated that second order would be lost for their approach, but in contradiction our experiments showed that second order is actually maintained. However, for Kou's model it seems indeed to be lost due to the discontinuous asymmetric jump density.

The thesis is organized as follows: In Chapter 2 we give a brief review of the most important tools being used, in particular some background on circulant and Toeplitz matrices is provided that we need throughout this work. The third chapter outlines the way from the stochastic model to the concrete PIDE formulations and presents the transformation that eliminates the convection term. Next we define the discretization schemes in Chapter 4 and discuss in the subsequent Chapters 5,6 and 7 several possibilities of how to efficiently solve the arising dense linear systems using FFT. We conclude in Chapter 8 by discussing the results of the numerical experiments that have been carried out.

Chapter 2

Toolbox

As we shall see in Chapter 3, in order to find the option price for either Merton's or Kou's jump diffusion model, we have to solve a partial integro-differential equation. After discretization with finite differences, we obtain a linear system of equations with a dense coefficient matrix that has to be solved in each time step. Direct solvers like the LU decomposition would need $O(n^3)$ operations for that, but by making use of the fast Fourier transform (FFT) algorithm exploiting the special structure of the coefficient matrices one can reduce the effort with iterative methods to only $O(n \log n)$ operations. This structure, namely the Toeplitz and the related circulant property, are defined and discussed in the subsequent, as well as the FFT algorithm itself. But first let us quickly review some facts about the conjugate gradient method which is one of the mentioned iterative methods.

2.1 Conjugate Gradient algorithm

A very efficient algorithm is the Conjugate Gradient Method (CG) which solves an $n \times n$ system $Ax = b$, where A is positive definite and symmetric. It is based on the following principle: The system $Ax = b$ can easily be solved if we know a set of n conjugate directions p_j for $j = 1, \ldots, n$ with respect to A, that means it holds that $p_i^T A p_j = 0$ for $i \neq j$. The conjugate directions are in particular linearly independent

and thus the solution x_* to the system $Ax = b$ has a decomposition $x_* = \sum_1^n \alpha_j p_j$ for $\alpha_j \in \mathbb{R}$, $j = 1, \ldots, n$. So $Ax_* = \sum_1^n \alpha_j A p_j = b$. If we multiply p_i^T for some fixed $i \in \{1, \ldots, n\}$ from the left to the last equation, we obtain due to the conjugacy $\alpha_i = p_i^T b / (p_i^T A p_i)$. Obviously we can now construct the solution x_* by computing all coefficients α_i in this way and then summing up the terms $\alpha_j p_j$ recursively by $x_{j+1} = x_j + \alpha_j p_j$, starting for example with $x_0 = \mathbf{0}$. CG uses this relation; in every iteration it computes the new coefficient α_j, adds the next term to the sum for x_* and builds a new conjugate direction p_j using an update which has the important feature that only the last conjugate direction needs to be known. Therefore we only need to save this last direction and not all of them, like for example GMRES does. The main work per iteration consists of computing one matrix-vector product. We state the algorithm as follows, where $\langle \cdot, \cdot \rangle$ denotes the ordinary dot product in $\mathbb{R}^n$:

Algorithm 2.1 (*Saad* [1996])**.** Conjugate Gradient:

1. Compute $r_0 := b - Ax_0$, $p_0 := r_0$.
2. For $j = 0, 1, \ldots,$ until convergence Do
3. $\quad \alpha_j := \langle r_j, r_j \rangle / \langle Ap_j, p_j \rangle$
4. $\quad x_{j+1} := x_j + \alpha_j p_j$
5. $\quad r_{j+1} := r_j - \alpha_j A p_j$
6. $\quad \beta_j := \langle r_{j+1}, r_{j+1} \rangle / \langle r_j, r_j \rangle$
7. $\quad p_{j+1} := r_{j+1} + \beta_j p_j$
8. End Do

It can be shown (*Saad* [1996]) that CG converges linearly in the $\| \cdot \|_A$ norm:

$$\|x_k - x_*\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x_0 - x_*\|_A,$$

with x_* being the exact solution, x_k the approximate solution obtained at the kth step of the CG algorithm and $\kappa = \lambda_{max} / \lambda_{min}$ the spectral condition number of A. Therefore we can expect very slow convergence for badly conditioned systems because in this case κ will be big and hence the fraction in parentheses close to 1. A very interesting fact that shows up in our computational results is the property of CG

that if the eigenvalues occur in k distinct clusters, CG will approximately solve the problem after k iterations. This can be seen from the following two theorems:

Theorem 2.2 (*Nocedal and Wright* [1999])**.** *If A has only k distinct eigenvalues, then the CG iteration will terminate at the solution in at most k iterations.*

Theorem 2.3 (*Luenberger* [1984])**.** *If A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, we have that*

$$\|x_{k+1} - x_*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|x_0 - x_*\|_A^2.$$

For small condition numbers CG converges fast, so we would like to improve the condition by transforming the system $Ax = b$ to an equivalent system $M^{-1}Ax = M^{-1}b$ where M should be chosen such that the coefficient matrix $M^{-1}A$ has a clustered spectrum. This so-called preconditioning is the subject of Section 6.2 where we discuss in detail several preconditioners M. An implementation of the preconditioned CG method is given in Algorithm 2.4. Note that now the main work per iteration consists of computing one matrix-vector product and additionally of solving a linear system $Mz = r$, therefore M must also be chosen such that this system is easy to solve. Section 6.3 points out how we can avoid this extra work by exploiting the special structure of the coefficient matrix arising from the discretization.

Algorithm 2.4 (*Saad* [1996])**.** Preconditioned Conjugate Gradient:

1. Compute $r_0 := b - Ax_0$, $z_0 := M^{-1}r_0$, $p_0 := z_0$
2. For $j = 0, 1, \ldots,$ until convergence Do*:*
3. $\quad \alpha_j := \langle r_j, z_j \rangle / \langle Ap_j, p_j \rangle$
4. $\quad x_{j+1} := x_j + \alpha_j p_j$
5. $\quad r_{j+1} := r_j - \alpha_j A p_j$
6. $\quad z_{j+1} := M^{-1} r_{j+1}$
7. $\quad \beta_j := \langle r_{j+1}, z_{j+1} \rangle / \langle r_j, z_j \rangle$
8. $\quad p_{j+1} := z_{j+1} + \beta_j p_j$
9. End Do

2.2 Fast Fourier Transform

The most expensive part in CG is the matrix-vector product. Under certain circumstances which we discuss in the next section it is possible to accelerate this product by making use of the *Fast Fourier Transform* algorithm (FFT), due to *Cooley and Tukey* [1965]. It enables us to compute the *discrete Fourier transform* (DFT)

$$a_j = \frac{1}{n}\sum_{k=0}^{n-1} x_k e^{-2\pi ijk/n}, \quad j = 0, \ldots, n-1,$$

in only $O(n \log n)$ operations instead of $O(n^2)$ if computed straightforward, the vectors $a = [a_0, \ldots, a_{n-1}]^T$ and $x = [x_0, \ldots, x_{n-1}]^T$ being in $\mathbb{C}^n$. The same holds for the inverse DFT:

$$x_j = \sum_{k=0}^{n-1} a_k e^{2\pi ijk/n}, \qquad j = 0, \ldots, n-1. \tag{2.1}$$

We are now going to review the principle which makes this reduction possible by following the ideas of the above cited paper applied to the inverse DFT (2.1):
First, let us suppose n can be factored as $n = r_1 r_2$. Then we can write the indices j and k in the form

$$j := j_1 r_1 + j_0, \qquad j_0 = 0, 1, \ldots, r_1 - 1, \qquad j_1 = 0, \ldots, r_2 - 1,$$

and likewise for $k := k_1 r_2 + k_0$ with $k_0 = 0, 1, \ldots, r_2 - 1$ and $k_1 = 0, 1, \ldots, r_1 - 1$. Now we split up our initial sum (2.1) by using the definition of k and j:

$$\begin{aligned} x_j = x_{j_0, j_1} &= \sum_{k=0}^{n-1} a_k e^{2\pi ikj/n} \\ &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} a_{k_0,k_1} e^{2\pi i k_1 r_2 j/n} e^{2\pi i k_0 j/n}. \end{aligned}$$

For convenience of notation let

$$R := e^{2\pi i/n}.$$

This notational simplification gives us $e^{2\pi i k_1 r_2 j/n} = R^{k_1 r_2 j}$ and $e^{2\pi i k_0 j/n} = R^{k_0 j}$. Our goal is to split up these sums into two sums with the innermost one being only dependent on j_0 and k_0. This can be achieved by using the equality

$$R^{k_1 r_2 j} = R^{j_0 k_1 r_2}, \tag{2.2}$$

which holds because

$$\begin{aligned} R^{jk_1r_2} = R^{(j_1r_1+j_0)k_1r_2} &= R^{j_1k_1r_1r_2}R^{j_0k_1r_2} \\ &= \underbrace{e^{2\pi i j_1k_1}}_{=1}R^{j_0k_1r_2} \\ &= R^{j_0k_1r_2}. \end{aligned}$$

Hence

$$x_{j_0,j_1} = \sum_{k_0=0}^{r_2-1} R^{k_0j} \sum_{k_1=0}^{r_1-1} a_{k_0,k_1} R^{j_0k_1r_2}.$$

Therefore we can define a new sum only depended on j_0 and k_0 by

$$\tilde{a}_{j_0,k_0} := \sum_{k_1=0}^{r_1-1} a_{k_0,k_1} R^{j_0k_1r_2},$$

which enables us to reformulate the expression (2.1) by

$$x_{j_0,j_1} = \sum_{k_0=0}^{r_2-1} R^{k_0(j_1r_1+j_0)} \tilde{a}_{j_0,k_0}. \tag{2.3}$$

$\tilde{a}$ has n elements since its indices are j_0 and k_0 and these are running from 0 to r_1-1 for j_0 and from 0 to r_2-1 for k_0, so in total we get $r_1r_2 = n$ elements. The sum $\tilde{a}_{j_0,k_0}$ is computed by r_1 operations. Likewise x depends on j_0 and j_1, so this array is of size $r_1r_2 = n$ as well and for each x_{j_0,j_1} the sum over k_0 takes r_2 operations.
So if we first compute the array $\tilde{a}$ by r_1n and then x by r_2n operations, the total amount will obviously be $T = (r_1 + r_2)n$, which is smaller than n^2!
This leads to the conclusion that for any integer factorization of

$$n = r_1r_2 \cdots r_m,$$

one can extend this process to m stages instead of only two and thus reduce the total number of operations to $T = n\sum_1^m r_i$. If all r_i are the same then $n = r^m$, so $m = \log_r n$ and thus $T = nr\log_r n$.

Cooley and Tukey [1965] emphasize that the factorization $n = 2^m$ will be nearly optimal efficient and has also the advantage that in this case binary computer arithmetic can be exploited. To sum up, we are able to compute the (inverse) DFT in only $T = 2n\log_2 n$ operations if n can be factorized as a power of 2. For practical implementation issues see for example *Van Loan* [1992].

Remark 2.5. We underline that if $n = 2^m$ then the cost T_{2n} of computing one (inverse) DFT of length $2n$ is given by

$$T_{2n} = 2n(m+1)2 = 4mn + 4n,$$

whereas for one of length n we need $T_n = 2mn$ operations. Thus $T_{2n} = 2T_n + 4n$, so the cost of one (inverse) DFT of length $2n$ is just a bit more than the cost of two (inverse) DFTs of length n.

2.3 Circulant matrices

In this section we highlight the key ingredients that enable us to solve even dense linear systems very fast and without need for much storage. This becomes possible exploiting two specific structures of the coefficient matrix in these systems: the Toeplitz property and the circulant property.
Let us start by formally defining a Toeplitz matrix, this means a matrix that is constant along its diagonals:

Definition 2.6. *$T \in \mathbb{C}^{n\times n}$ is called* Toeplitz *if T is determined by the $2n-1$ scalars $t_{-(n-1)}, \ldots, t_{n-1}$ with $T_{ij} = t_{j-i}$ for all i and j.*

$$T = \begin{pmatrix} t_0 & t_{-1} & \ldots & t_{2-n} & t_{1-n} \\ t_1 & t_0 & \ddots & & t_{2-n} \\ \vdots & \ddots & \ddots & & \vdots \\ t_{n-2} & & & \ddots & t_{-1} \\ t_{n-1} & t_{n-2} & \ldots & t_1 & t_0 \end{pmatrix}. \tag{2.4}$$

Observe that for a general Toeplitz matrix it is sufficient only to save the vector $[t_{n-1}, t_{n-2}, \ldots, t_{1-n}]^T \in \mathbb{C}^{2n-1}$ instead of the whole matrix. For an Hermitian Toeplitz matrix even saving only the first column is enough. A special case of Toeplitz matrices is the so-called *circulant matrix*:

Definition 2.7. *$C \in \mathbb{C}^{n\times n}$ is called* circulant *if it is a Toeplitz matrix where each column is a circular shift of its preceding column.*

So C is of the form

$$C = \begin{pmatrix} c_0 & c_{n-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & \ddots & & c_2 \\ \vdots & \ddots & \ddots & & \vdots \\ c_{n-2} & & & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{pmatrix}. \tag{2.5}$$

Circulant matrices are fully determined by their first row or column and thus we may introduce the notation $C = \operatorname{circ}(c_0, \dots, c_{n-1})$. Note that once again only little storage is needed: Saving the first column or row instead of the whole matrix is sufficient.

Example 2.8.

$$\Pi = \operatorname{circ}(0, 1, 0, \dots, 0) = \begin{pmatrix} 0 & \cdots & & & 1 \\ 1 & 0 & \ddots & & 0 \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}. \tag{2.6}$$

This permutation matrix is used to analyze circulant matrices because it corresponds to a forward shift permutation - the Π stands for "push". If we multiply it by itself, the ones are being pushed one diagonal further.

This behavior of Π has the immediate consequence that we can rewrite every circulant matrix in a polynomial form, namely

$$\operatorname{circ}(c_0, c_1, \dots, c_{n-1}) = c_0 I + c_1 \Pi + \dots + c_{n-1}\Pi^{n-1}.$$

The interesting implication is that all circulant matrices of the same order must commute since we can rewrite them as polynomials evaluated in the same matrix. In particular we have that all circulant matrices are normal, that means $C^*C = CC^*$.

Definition 2.9. *The unitary and symmetric matrix* $F_n \in \mathbb{C}^{n\times n}$ *is called* Fourier matrix *if it is of the form*

$$F_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & \omega & & \omega^{n-2} & \omega^{n-1} \\ \vdots & \vdots & & & \vdots \\ 1 & \omega^{n-2} & & & \omega^{(n-2)(n-1)} \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)(n-2)} & \omega^{(n-1)^2} \end{pmatrix}, \tag{2.7}$$

with $\omega = e^{-2\pi i/n}$.

Why could this matrix be useful? The Fourier matrix, like one can already guess from its name, has a close connection to the Fast Fourier Transform (FFT). Recall from the last section that the inverse FFT enables us to efficiently compute the sums

$$x_j = \sum_{k=0}^{n-1} a_k e^{2\pi ijk/n}, \qquad j = 0, 1, \ldots, n-1.$$

A closer look reveals that this vector $x = [x_0, \ldots, x_{n-1}]^T$ with components x_j given above is nothing else but the matrix-vector product

$$\sqrt{n} F_n^* \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = x,$$

because $F_n^* = \bar{F}_n$ due to its symmetry, where $\bar{F}_n$ denotes the complex conjugate of F_n and F_n^* the conjugate transpose of F_n. Conversely, if we intend to compute the vector $a = [a_0, \ldots, a_{n-1}]^T$ for a known x, the equation above gives directly

$$\begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \frac{1}{\sqrt{n}} F_n x,$$

which accordingly means to compute the DFT

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} x_k e^{-2\pi ijk/n}, \quad j = 0, \ldots, n-1.$$

DFT and inverse DFT can therefore be identified with an Fourier matrix times vector multiplication, that means we can compute this matrix-vector product fast by using the FFT algorithm. An in-depth treatment of FFT implementations can be found in *Van Loan* [1992].

Remark 2.10. In MATLAB the Fast Fourier Transform is defined in a slightly different way, so in order to compute the Fourier matrix times vector product matching our definition we need to apply some scaling: $F_n x = \frac{1}{\sqrt{n}}\,\texttt{fft}(x)$ and $F_n^* a = \sqrt{n}\,\texttt{ifft}(a)$. Here $\texttt{fft}(x)$ and $\texttt{ifft}(a)$ denote the corresponding MATLAB functions.

With regard to the close relationship between Fourier matrix and FFT, the following result is the backbone of all considered methods in this work:

Theorem 2.11 (*Davis* [1979])**.** *Let $C_n \in \mathbb{R}^{n\times n}$ be circulant. Then it has the decomposition*

$$C_n = F_n^* \Lambda F_n, \tag{2.8}$$

where $\Lambda = \operatorname{diag}(\lambda_0, \ldots, \lambda_{n-1})$, λ_j being the jth eigenvalue of C_n, $j = 0, \ldots, n-1$.

The theorem has in particular two very important consequences which we use throughout the thesis. First, the eigenvalues can be easily found using this decomposition: Let c denote the first column of C_n, $e_1 := [1, 0, \ldots, 0]^T$ and $\vec{e} := [1, 1, \ldots, 1]^T$. From (2.8) we obtain

$$\begin{aligned} & C_n = F_n^* \Lambda F_n \\ \Leftrightarrow \quad & F_n C_n e_1 = \Lambda F_n e_1 \\ \Leftrightarrow \quad & F_n c = \Lambda \vec{e} \frac{1}{\sqrt{n}} \\ \Leftrightarrow \quad & \sqrt{n} F_n c = \Lambda \vec{e}, \end{aligned} \tag{2.9}$$

so the eigenvalues λ_j of C_n are clearly given by

$$\lambda_j = \sum_{k=0}^{n-1} c_k e^{-2\pi i j k/n}, \qquad j = 0, \ldots, n-1.$$

Hence F_n consists of the eigenvectors of C_n; actually all circulant matrices have the same eigenvectors, namely the columns of F_n.
Second, the matrix-vector product $x = C_n a = F_n^* \Lambda F_n a$ can be computed efficiently in four steps, see for example *Golub and Van Loan* [1996]: We start by applying an FFT to a, next we compute $\tilde{c} := \Lambda \vec{e}$ by equation (2.9), multiply elementwise the so obtained vector $\tilde{c}$ containing the diagonal elements of Λ with $F_n a$ and finally apply an inverse FFT. Formally, this looks like:

$$\begin{aligned} \tilde{a} &= F_n a, \\ \tilde{c} &= \sqrt{n} F_n c, \\ z &= \tilde{c}. * \tilde{a}, \\ x &= F_n^* z. \end{aligned}$$

Here, c is again the first column of C_n and ".∗" denotes element-wise multiplication. We underline that in this way we can compute the circulant matrix times vector product $C_n a$ by three FFT's and one vector multiplication, and therefore we just need $O(n \log n)$ operations for the product. This reduction is a substantial improvement compared to straightforward computation which would cost $O(n^2)$ operations. Of course, the decomposition (2.8) also gives us immediately the inverse of any circulant matrix, $C_n^{-1} = F_n^* \Lambda_n^{-1} F_n$, because of the Fourier matrix F_n being unitary.

Circulants belong to a larger class of matrices called $\{\omega\}$-*circulant matrices*:

Definition 2.12. *Let $\omega = e^{i\theta_0}$ with $\theta_0 \in [-\pi, \pi]$. An $n \times n$ matrix W is said to be an* $\{\omega\}$-circulant matrix *if it has the spectral decomposition*

$$W_n = \Omega_n^* F_n^* \Lambda_n F_n \Omega_n. \tag{2.10}$$

Here $\Omega_n = \mathrm{diag}[1, \omega^{-1/n}, \ldots, \omega^{-(n-1)/n}]$, F_n denotes the Fourier matrix and Λ_n is a diagonal matrix containing the eigenvalues of W_n.

In the framework of this thesis we will only deal with the special cases $\omega = 1$ and $\omega = -1$, that means the circulant matrices and the so-called *skew-circulant matrices* respectively. For skew-circulant matrices we obtain likewise

$$\sqrt{n} F_n \Omega_n s = \Lambda_n \vec{e},$$

where s is the first column of the skew-circulant matrix. Note that only $O(n \log n)$ operations are needed to find Λ_n in both the circulant as well as in the skew-circulant case, and accordingly the matrix-vector product is also for skew-circulant matrices fast computable. In this sense, these types of matrices are obviously desirable especially for iterative methods that need to perform matrix-vector products in each iteration. In particular with regard to preconditioning of linear circulant-type systems we need to know in how far it is possible to work with circulant matrices without losing their structure. The following lemma equips us with a vast toolkit of possible operations:

Lemma 2.13 (*Davis* [1979]). *If $C_1 \in \mathbb{C}^{n\times n}$ and $C_2 \in \mathbb{C}^{n\times n}$ are circulant and $\{\alpha_k\}$ is a set of scalars, then C_1^T, C_1^*, $\alpha_1 C_1 + \alpha_2 C_2$, $C_1 C_2$ and $\sum_{k=0}^{r} \alpha_k C_1^k$ are circulant. Moreover, C_1 and C_2 commute. If C_1 is nonsingular, its inverse is circulant. With $C_1 = F_n^* \Lambda F_n$, its inverse is given by $C_1^{-1} = F_n^* \Lambda^{-1} F_n$.*

A similar result is valid for skew-circulant matrices:

Lemma 2.14 (*Davis* [1979]). *If $S_1 \in \mathbb{C}^{n\times n}$ and $S_2 \in \mathbb{C}^{n\times n}$ are skew circulant and $p(x)$ is a polynomial in x, then S_1^T, S_1^*, $S_1 S_2$, $p(S_1)$, S_1^{-1} (if it exists) are skew circulant. Moreover, S_1 and S_2 commute.*

The possibility to quickly compute the matrix-vector product can be extended from circulant matrices to general Toeplitz matrices because it is possible to embed a Toeplitz matrix in a circulant one. *Ng* [2004] uses the following method: For an $n \times n$ Toeplitz matrix T_n, the Toeplitz matrix-vector multiplication $T_n y$ can be computed with three FFTs by first embedding T_n into a $2n \times 2n$ circulant matrix, that means

$$\begin{pmatrix} T_n & B_n \\ B_n & T_n \end{pmatrix} \begin{pmatrix} y \\ 0 \end{pmatrix} = \begin{pmatrix} T_n y \\ * \end{pmatrix}, \tag{2.11}$$

where

$$B_n = \begin{pmatrix} 0 & t_{n-1} & \dots & t_2 & t_1 \\ t_{1-n} & 0 & t_{n-1} & & t_2 \\ \vdots & t_{1-n} & 0 & \ddots & \vdots \\ t_{-2} & & \ddots & \ddots & t_{n-1} \\ t_{-1} & t_{-2} & \cdots & t_{1-n} & 0 \end{pmatrix}.$$

The importance of this embedding lies in the fact that if we let the conjugate gradient method (CG) operate on a Toeplitz system, the crucial cost of computing the matrix-vector product in every iteration can be substantially diminished. There exists another interesting fact that we use in Chapter 6 in the preconditioned CG framework to reduce the number of FFTs needed in the computation of the matrix-vector product: We can easily partition any Toeplitz matrix into a sum of a circulant matrix and a skew-circulant matrix. Defining

$$U_n = \frac{1}{2}\begin{pmatrix} t_0 & t_{-1}+t_{n-1} & & t_{-(n-1)}+t_1 \\ t_1+t_{-(n-1)} & t_0 & \ddots & \\ & \ddots & \ddots & \\ & & \ddots & t_{-1}+t_{n-1} \\ t_{n-1}+t_{-1} & & & t_0 \end{pmatrix},$$

and

$$V_n = \frac{1}{2}\begin{pmatrix} t_0 & t_{-1}-t_{n-1} & & t_{-(n-1)}-t_1 \\ t_1-t_{-(n-1)} & t_0 & \ddots & \\ & \ddots & \ddots & \\ & & \ddots & t_{-1}-t_{n-1} \\ t_{n-1}-t_{-1} & & & t_0 \end{pmatrix},$$

so that we have

$$T_n = U_n + V_n. \tag{2.12}$$

This decomposition was first found by *Pustyl'nikov* [1980].

Chapter 3

PIDE formulations for both models of Merton and Kou

This chapter is intended to outline the way from the stochastic process to a partial integro-differential equation (PIDE) which describes the option price for general Lévy processes. In the framework of this thesis we confine ourselves to just two models, the one by *Merton* [1976] and the other more recently by *Kou* [2002]. Since these two models are rather similar, it is possible to use the same PIDE for both models, only the density and some parameters related to it have to be changed.
In the first section we restate a rather general form of a PIDE formulation, in the second we derive from it a more specific version for the two models considered and finally in the third section we derive an equivalent formulation without a convection term. For the purpose of clear distinction between the two equivalent PIDE versions in Section 2 and Section 3 we call them PIDE $\mathcal{W}$ (**W**ith convection term) and PIDE $\mathcal{N}$ (**N**o convection term), respectively.

3.1 General PIDE for Lévy processes

The Black-Scholes market is complete, which means that any financial instrument can be replicated. Therefore in this world the option price may be derived by hedging and

no-arbitrage arguments. If jumps in the price of the underlying are allowed it becomes more difficult, the market is in this case incomplete so replication is not possible. But the no-arbitrage assumption still holds and can be used to find a price.
The option price will depend on the time to maturity and the price of the underlying S_t as variables and on some parameters which are considered given. Therefore we need to describe the movement of the price of the underlying asset, for example a stock, for which we use a stochastic process $(S_t)_{t\geq 0}$ on a probability space $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ with filtration $\mathcal{F}$. We can define the price process of the underlying stock by an exponential Lévy model, that is

$$S_t = S_0 e^{rt+X_t},$$

where r is the riskfree interest rate, S_0 the stock price at time zero and X_t a Lévy process.

Definition 3.1. *A stochastic process $(X_t)_{t\geq 0}$ on $(\Omega, \mathcal{F}, \mathbb{P})$ with values in $\mathbb{R}^d$ such that $X_0 = 0$ is called a* Lévy process *if it possesses the following properties:*

(i) Independent increments: for every increasing sequence of times $t_0, \ldots, t_n$, the random variables $X_{t_0}, X_{t_1} - X_{t_0}, \ldots, X_{t_n} - X_{t_{n-1}}$ are independent.

(ii) Stationary increments: the law of $X_{t+h} - X_t$ does not depend on t.

(iii) Stochastic continuity: $\forall \epsilon > 0$, $\lim_{h\to 0} \mathbb{P}(|X_{t+h} - X_t| \geq \epsilon) = 0$.

Next we choose the Lévy process X_t to be

$$X_t := (\mu - r - \frac{\sigma^2}{2})t + \sigma W_t + \sum_{i=1}^{N_t} Y_i,$$

where we set $\mu + \lambda\eta = r$ and $\eta := \mathbf{E}(\exp(x) - 1)$ denoting the expected relative change in stock price caused by a jump. (Y_i) are independent identically distributed random variables with distribution F, (N_t) is a Poisson process with intensity λ, that is the mean number of jumps per unit time, and (W_t), (Y_i), (N_t) are mutually independent. As usual, W_t denotes the Brownian motion (also called Wiener process), σ is the volatility and μ the drift. After plugging in $\mu = r - \lambda\eta$, the price process becomes

$$S_t = S_0 \exp(L_t),$$

where $L_t := (r - \lambda\eta - \sigma^2/2)t + \sigma W_t + \sum_1^{N_t} Y_i$. For a general Lévy process X_t one can define a characteristic function by the so-called Lévy-Khintchine representation (*Sato* [1999]):

$$\mathbf{E}(e^{izX_t}) = e^{t\Psi(z)},$$

with $\Psi(t) := -az^2/2 + i\gamma z + \int_{\mathbb{R}}(\exp(izx) - 1 - izx\mathbf{1}_{|x|\leq 1})v(\mathrm{d}x)$, $z \in \mathbb{R}$, for $\sigma > 0$ and γ real constants and ν a positive measure on $\mathbb{R}$ satisfying

$$\int_{|x|\leq 1} x^2\nu(\mathrm{d}x) < \infty, \quad \int_{|x|>1} \nu(\mathrm{d}x) < \infty.$$

We only consider processes with $\int_{\mathbb{R}} |x|\mathrm{d}\nu(x) < \infty$; then according to *Cont and Tankov* [2004] the characteristic function is defined for $z \in \mathbb{R}$ by

$$\mathbf{E}(e^{izX_t}) = e^{t\Psi(z)}, \quad \Psi(z) = -\frac{a}{2}z^2 + i\gamma z + \int_{\mathbb{R}} e^{izx} - 1 - izx\,\mathrm{d}\nu(x).$$

The triple (a, γ, ν) arising in the characteristic function is called the *reduced Lévy triplet* and is important because there exists a PIDE formulation of the option pricing problem where only the interest rate r and this Lévy triplet enters. For our case, *Almendral and Oosterlee* [2005] verified that the reduced Lévy triplet for the process L_t as defined above is $(\sigma^2, r - \lambda\eta + \lambda\mathbf{E}(y) - \sigma^2/2, \lambda F)$.

The discounted price process should reflect the no-arbitrage assumption, that is it should be a martingale under some risk-neutral measure $\mathbb{Q}$. Using Esscher transforms for example it is possible for Lévy processes to find such a martingale measure $\mathbb{Q}$ equivalent to the underlying probability measure $\mathbb{P}$ such that this discounted price process is a martingale under $\mathbb{Q}$ if the prices are $\mathbb{Q}$-integrable, see *Raible* [2000]. Let us assume such a measure $\mathbb{Q}$ has been found. Then just like the discounted price process $\exp(-rt)S_t$, so is $\exp(-rt)w(t, S_t)$ as well a martingale under $\mathbb{Q}$. Following *Raible* [2000], we can therefore say that $w(t, S_t)$ is the discounted conditional expected value of the payoff function, the latter being for an European call $(S_T - K)^+$, so as a formula it becomes

$$w(t, S_t) = e^{rt}\mathbf{E}_{\mathbb{Q}}(e^{-rT}(S_T - K)^+|\mathcal{F}_t).$$

The Markov property, that means the asset price is independent from older prices, yields that only the current price is relevant, so using $S_T = S_t \exp(L_{T-t})$ we get

$$w(t, S) = e^{-r(T-t)}\mathbf{E}_{\mathbb{Q}}((S_T - K)^+|S_t = S) = e^{-r(T-t)}\mathbf{E}_{\mathbb{Q}}((Se^{L_{T-t}} - K)^+)).$$

Finally, after the change of variable $x := \ln S$ we define

$$v(t, x) := e^{-r(T-t)}\mathbf{E}_{\mathbb{Q}}((e^{x+L_{T-t}} - K)^+).$$

Theorem 3.2 (*Raible* [2000]). *Let* $v(t,x) \in C^{1,2}([0,T) \times \mathbb{R}) \cap C^0([0,T] \times \mathbb{R})$ *and assume* $\int_{\mathbb{R}} |x| \, \mathrm{d}\nu(x) < \infty$. *Then* v *satisfies the following partial integro-differential equation*

$$v_t + \frac{a}{2} v_{xx} + \gamma v_x - rv + \int_{-\infty}^{\infty} (v(t,x+y) - v(t,x) - v_x(t,x)y) \, \mathrm{d}\nu(y) = 0,$$
$$\forall (t,x) \in [0,T) \times \mathbb{R},$$

with final condition

$$v(T,x) = H(e^x), \qquad \forall \; x \in \mathbb{R}.$$

The only parameters entering are the risk-free interest rate r *and the reduced Lévy triplet* (a,γ,ν) *under the risk-neutral measure* $\mathbb{Q}$. *The function* $H(\cdot)$ *denotes the payoff at* $t = T$.

3.2 PIDE $\mathcal{W}$ with convection term

In this short section we introduce the PIDE $\mathcal{W}$ with convection term like it is stated in *Almendral and Oosterlee* [2005]. In the next chapter we solve this PIDE in the way they proposed in their paper, both to check their results and to compare the computational results of their approach with our way, which is to use the preconditioned CG method on the linear system resulting from a discretization of the transformed PIDE $\mathcal{N}$ without convection term. The latter mentioned PIDE $\mathcal{N}$ is derived in the next section.

We just saw how the PIDE looks for a general Lévy process: Now let us get more specific: In Merton's model the expected value $\mathbf{E}(\cdot)$ is taken with respect to the normal density, that means

$$f_M(x) = \frac{1}{\sqrt{2\pi}\sigma_J} e^{-(x-\mu_J)^2/(2\sigma_J^2)}, \tag{3.1}$$

with mean μ_J and standard deviation σ_J, in the option pricing context the jump volatility. The cumulative normal distribution is given by

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-z^2/2} \, \mathrm{d}z. \tag{3.2}$$

For $\eta = \mathbf{E}(e^x - 1)$ we obtain in Merton's model by using the change of variable $\tilde{x} = (x - \mu_J)/\sigma_J$:

$$\begin{aligned}
\eta_M &:= \int_{\mathbb{R}} (e^x - 1)\,\mathrm{d}F(x) = \int_{\mathbb{R}} (e^x - 1) f_M(x)\,\mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{x - ((x-\mu_J)/\sigma_J)^2/2} \frac{\mathrm{d}x}{\sigma_J} - 1 \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-\tilde{x}^2/2 + \sigma\tilde{x} + \mu_J}\,\mathrm{d}\tilde{x} - 1 \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-(\tilde{x} - \sigma_J)^2/2} e^{\mu_J + \sigma_J^2/2}\,\mathrm{d}\tilde{x} - 1 \\
&= e^{\mu_J + \sigma_J^2/2} \Phi(\infty) - 1 \\
&= e^{\mu_J + \sigma_J^2/2} - 1.
\end{aligned}$$

In Kou's model we use the double exponential density

$$f_K(x) = p\alpha_1 e^{-\alpha_1 x} \mathbf{1}_{x \geq 0} + q\alpha_2 e^{\alpha_2 x} \mathbf{1}_{x < 0}, \tag{3.3}$$

where $p,\ q \geq 0$, $p + q = 1$ are the probabilities of upward and downward jumps and $\alpha_1 > 1$, $\alpha_2 > 0$. Using this density we compute for the expected value $\eta = \mathbf{E}(e^x - 1)$:

$$\begin{aligned}
\eta_K &= \int_{\mathbb{R}} (e^x - 1) f_K(x)\,\mathrm{d}x \\
&= q\alpha_2 \int_{-\infty}^{0} e^{x(1+\alpha_2)}\,\mathrm{d}x + p\alpha_1 \int_0^{\infty} e^{x(1-\alpha_1)}\,\mathrm{d}x - q - p \\
&= \frac{q\alpha_2}{1 + \alpha_2} + \frac{p\alpha_1}{\alpha_1 - 1} - 1.
\end{aligned}$$

Now we are looking for a corresponding PIDE formulation: Using the reduced Lévy triplet $(\sigma^2, r - \sigma^2/2 - \lambda\eta + \lambda\mathbf{E}(Y), \lambda F)$ in the general PIDE stated in Theorem 3.2 we get:

$$\begin{aligned}
v_t + \frac{\sigma^2}{2} v_{xx} + (r - \frac{\sigma^2}{2} - \lambda\eta + \lambda\mathbf{E}(Y)) v_x - rv \\
+ \lambda \int_{-\infty}^{\infty} (v(t, x+y) - v(t,x) - v_x(t,x)y)\, \underbrace{\mathrm{d}F(y)}_{f(y)\,dy} = 0,
\end{aligned}$$

$$\Leftrightarrow v_t + \frac{\sigma^2}{2}v_{xx} + (r - \frac{\sigma^2}{2} - \lambda\eta)v_x + \lambda\mathbf{E}(Y)v_x - rv$$
$$- \lambda v \underbrace{\int_{-\infty}^{\infty} \mathrm{d}F(y)}_{=1} - \lambda v_x \underbrace{\int_{-\infty}^{\infty} y f(y)\,\mathrm{d}y}_{\mathbf{E}(Y)} + \lambda \int_{-\infty}^{\infty} v(t, x+y) f(y)\,\mathrm{d}y = 0.$$

So we get

$$\begin{cases} v_t + \frac{\sigma^2}{2}v_{xx} + (r - \frac{\sigma^2}{2} - \lambda\eta)v_x - (r+\lambda)v \\ \quad +\lambda \int_{-\infty}^{\infty} v(t, x+y) f(y)\,\mathrm{d}y = 0, & \forall (t,x) \in [0,T) \times \mathbb{R}, \\ v(T,x) = H(e^x), & \forall x \in \mathbb{R}. \end{cases} \tag{3.4}$$

What remains is to transform this problem to obtain a forward in time problem: We do that by defining the time to maturity $\tau = T - t$ and setting $u(\tau, \cdot) := v(T - \tau, \cdot)$. Thus $u_\tau = (v(T-\tau,\cdot))_\tau = -v_t$, and by changing variables in the integral term $z = x + y$, $y = z - x$, $\mathrm{d}x = \mathrm{d}z$ and we have now the final formulation of what we will call the PIDE $\mathcal{W}$:

$$\begin{cases} u_\tau - \frac{\sigma^2}{2}u_{xx} - (r - \frac{\sigma^2}{2} - \lambda\eta)u_x + (r+\lambda)u \\ \quad -\lambda \int_{-\infty}^{\infty} u(\tau, z) f(z-x)\,\mathrm{d}z = 0, & \forall (\tau,x) \in [0,T) \times \mathbb{R}, \\ u(0,x) = H(e^x), & \forall x \in \mathbb{R}. \end{cases} \tag{3.5}$$

3.3 PIDE $\mathcal{N}$ without convection term

We will now show how to transform our PIDE

$$u_\tau - \frac{1}{2}\sigma^2 u_{xx} - (r - \frac{1}{2}\sigma^2 - \lambda\eta)u_x + (r+\lambda)u$$
$$- \lambda \int_{-\infty}^{\infty} u(\tau, z) f(z-x)\,\mathrm{d}z = 0 \quad \forall (\tau,x) \in [0,T) \times \mathbb{R}, \tag{3.6}$$

$$u(0,x) = H(e^x), \quad \forall x \in \mathbb{R},$$

in such a way that the convection term u_x disappears. In this manner, oscillations are avoided that might otherwise appear if we discretize u_x by the central difference quotient. Furthermore, this is in particular useful for Merton's model, because we

can then discretize the PIDE such that the resulting linear equation has a symmetric Toeplitz coefficient matrix. Combined with the positive definiteness we can then implement the preconditioned CG-method with an appropriate preconditioner. The most expensive part of the CG algorithm is the matrix-vector product which has to be computed in every iteration. Since the coefficient matrix is Toeplitz we can embed it in a circulant matrix in order to exploit the previously explained virtues of the FFT in fast computing matrix-vector products if the matrix is circulant (see Chapter 2). By using a circulant preconditioner we may extend this technique to the preconditioning process and achieve an overall workload of only $O(n \log n)$ operations per iteration, even including the work necessary to solve the linear system associated with the preconditioning.
For convenience of notation, let

$$\zeta := r - \frac{1}{2}\sigma^2 - \lambda\eta.$$

We define a new variable $\xi = x + \zeta\tau$ and rewrite u as a function of τ and ξ, namely let $w(\tau, \xi(x,\tau)) := u(\tau, x)$. Then we have

$$\begin{aligned} u_\tau &= w_\tau + \zeta w_\xi, \\ u_x &= w_\xi, \\ u_{xx} &= w_{\xi\xi}. \end{aligned}$$

This idea is well-known in the Black-Scholes context. Restating (3.6) with respect to τ and ξ we obtain

$$w_\tau + \zeta w_\xi - \frac{\sigma^2}{2} w_{\xi\xi} - \zeta w_\xi + (r+\lambda)w - \lambda \int_{-\infty}^{\infty} \underbrace{w(\tau, z+\zeta\tau)}_{=u(\tau,z)} f(z - \underbrace{(\xi - \zeta\tau)}_{=x})\, \mathrm{d}z = 0,$$
$$\forall (\tau, \xi) \in [0,T) \times \mathbb{R}.$$

Like intended the first order term disappears and the integral can be simplified by the change of variable $\tilde{z} = z + \zeta\tau$, so $\mathrm{d}\tilde{z} = \mathrm{d}z$.

$$w_\tau - \frac{1}{2}\sigma^2 w_{\xi\xi} + (r+\lambda)w - \lambda \int_{-\infty}^{\infty} w(\tau, \tilde{z}) \underbrace{f(\tilde{z} - \zeta\tau - \xi + \zeta\tau)}_{f(\tilde{z}-\xi)}\, \mathrm{d}\tilde{z} = 0.$$

So finally we obtain

$$w_\tau - \frac{1}{2}\sigma^2 w_{\xi\xi} + (r+\lambda)w - \lambda \int_{-\infty}^{\infty} w(\tau, \tilde{z}) f(\tilde{z} - \xi)\, \mathrm{d}\tilde{z} = 0,$$
$$\forall (\tau, \xi) \in [0, T) \times \mathbb{R}, \quad (3.7)$$

$$w(0, \xi) = H(e^\xi), \quad \forall \xi \in \mathbb{R},$$

and we will call it the PIDE $\mathcal{N}$ (since it has **N**o convection term) to distinguish it from its equivalent form $\mathcal{W}$ (**W**ith convection term) like stated in the last section.

Chapter 4

Numerical problem approximations

Now we arrive at the task of how to approximate the analytical problems given by the PIDEs $\mathcal{W}$ and $\mathcal{N}$ using finite differences. In Section 4.1 we are walking further on the trail of *Almendral and Oosterlee* [2005], whereas in Section 4.2 we present our approach for the convection-free PIDE $\mathcal{N}$. Once the discretizations are done and the corresponding linear systems obtained, the question is how to efficiently solve them for Merton's and Kou's model, respectively. We discuss answers to this question in the remaining chapters of the thesis.

4.1 Finite Differences approach for PIDE $\mathcal{W}$ on uniform grid

4.1.1 Domain and integral truncation

In order to solve the PIDE (3.5) we first note that we have to truncate both the infinite problem domain as well as the infinite domain for the integral. We recall that the log-price of the underlying asset was denoted by $x = \ln S$ so that $x \to -\infty$ means that the price of the underlying asset $S \to 0$. Conversely, for $x \to \infty$ also $S \to \infty$. Since we need some approximation of the integral term "near infinity" one can make

use of the fact that by specifying the type of European option also the option value in the limits can easily be seen. For a put option we would have that for very large asset prices S the strike price K will be exceeded and thus

$$u(\tau, x) \to 0, \qquad (x \to +\infty),$$

and

$$u(\tau, x) \to Ke^{-r\tau}, \qquad (x \to -\infty),$$

since for S being almost zero the option will be worth the strike price, discounted over the remaining time to maturity.
We will focus our attention in what follows on an European call option, for which we have

$$\begin{aligned} u(\tau, x) &\to e^x - Ke^{-r\tau}, && (x \to +\infty), \\ u(\tau, x) &\to 0, && (x \to -\infty). \end{aligned}$$

Now we truncate the domain for $x = \ln S$ to $\Omega := (-\hat{x}, \hat{x})$ where $\hat{x}$ can be chosen in such a way that the integral term over the complement $\Omega^c = (-\infty, -\hat{x}) \cup (\hat{x}, \infty)$ is sufficiently small. As shown in *Cont and Voltchkova* [2005], the truncation error decreases exponentially with the domain size if we use the payoff function as boundary condition. Note that it is not sufficient to impose only boundary conditions at $x = \hat{x}$ and $x = -\hat{x}$ because of the nonlocal integral term. To estimate the integral on the complement of the computational domain, that means on the set Ω^c, we use the boundary conditions $u(\tau, x) \approx 0$ on $(-\infty, -\hat{x})$ and $u(\tau, x) \approx e^x - Ke^{-r\tau}$ on $(\hat{x}, +\infty)$. Thus the remainder $R(\tau, x, \hat{x})$ of the integral $\int_{\mathbb{R}} u(\tau, z) f(z - x)\, \mathrm{d}z$ outside Ω will be estimated by

$$R(\tau, x, \hat{x}) = \int_{\hat{x}}^{\infty} (e^z - Ke^{-r\tau}) f(z - x)\, \mathrm{d}z. \tag{4.1}$$

Integral estimate for Merton's model

In Merton's model $f(x)$ is assumed to be the normal density $f_M(x)$ defined in (3.1) with $\mu_J = 0$. Using the cumulative normal distribution (3.2) we can simplify the remainder R defined in (4.1) by using the change of variable $y := (z - x)/\sigma_J$ as

follows:

$$\begin{aligned}\int_{\hat{x}}^{\infty} f_M(z-x)\,\mathrm{d}z &= \frac{1}{\sqrt{2\pi}}\int_{\hat{x}}^{\infty} e^{-(z-x)^2/(2\sigma_J^2)}\,\frac{\mathrm{d}z}{\sigma_J}\\ &= \frac{1}{\sqrt{2\pi}}\int_{\frac{\hat{x}-x}{\sigma_J}}^{\infty} e^{-y^2/2}\,\mathrm{d}y\\ &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{-\frac{\hat{x}-x}{\sigma_J}} e^{-y^2/2}\,\mathrm{d}y\\ &= \Phi\left(\frac{x-\hat{x}}{\sigma_J}\right),\end{aligned}$$

and for the other term by using $y := (z-x)/\sigma_J$ and $\tilde{y} := y - \sigma_J$:

$$\begin{aligned}\int_{\hat{x}}^{\infty} e^z f_M(z-x)\,\mathrm{d}z &= \frac{1}{\sqrt{2\pi}}\int_{\hat{x}}^{\infty} e^{z-(z-x)^2/(2\sigma_J^2)}\,\frac{\mathrm{d}z}{\sigma_J}\\ &= \frac{1}{\sqrt{2\pi}}\int_{\frac{\hat{x}-x}{\sigma_J}}^{\infty} e^{\sigma_J y + x - y^2/2}\,\mathrm{d}y\\ &= \frac{1}{\sqrt{2\pi}}\int_{\frac{\hat{x}-x}{\sigma_J}}^{\infty} e^{-(y^2-2\sigma_J y+\sigma_J^2)/2} e^{x+\sigma_J^2/2}\,\mathrm{d}y\\ &= \frac{e^{x+\sigma_J^2/2}}{\sqrt{2\pi}}\int_{\frac{\hat{x}-x}{\sigma_J}-\sigma_J}^{\infty} e^{-\tilde{y}^2/2}\,\mathrm{d}\tilde{y}\\ &= \Phi\left(\frac{x-\hat{x}+\sigma_J^2}{\sigma_J}\right) e^{x+\sigma_J^2/2}.\end{aligned}$$

Overall we obtain

$$R(\tau, x, \hat{x}) = e^{x+\sigma_J^2/2}\Phi\left(\frac{x-\hat{x}+\sigma_J^2}{\sigma_J}\right) - Ke^{-r\tau}\Phi\left(\frac{x-\hat{x}}{\sigma_J}\right).$$

Integral estimate for Kou's model

For Kou's model the integral estimate looks of course a bit different since R in (4.1) is dependent on the density f. For the double exponential density f_K defined in (3.3) we have

$$\begin{aligned}R(\tau, x, \hat{x}) &= \int_{\hat{x}}^{\infty} (e^z - Ke^{-r\tau}) f_K(z-x)\,\mathrm{d}z\\ &= \int_{\hat{x}}^{\infty} e^z f_K(z-x)\,\mathrm{d}z - Ke^{-r\tau}\int_{\hat{x}}^{\infty} f_K(z-x)\,\mathrm{d}z.\end{aligned}$$

Next, keep in mind that x is in the computational domain, so x is smaller than the upper truncation point $\hat{x}$. The integration variable z, however, runs from $\hat{x}$ to infinity, so that $z - x$ is always nonnegative. Hence

$$\begin{aligned}\int_{\hat{x}}^{\infty} f_K(z-x)\,\mathrm{d}z &= p\alpha_1 e^{\alpha_1 x}\int_{\hat{x}}^{\infty} e^{-\alpha_1 z}\,\mathrm{d}z\\ &= pe^{\alpha_1(x-\hat{x})},\\ \int_{\hat{x}}^{\infty} e^z f_K(z-x)\,\mathrm{d}z &= p\alpha_1 e^{\alpha_1 x}\int_{\hat{x}}^{\infty} e^{z(1-\alpha_1)}\,\mathrm{d}z\\ &= -\frac{p\alpha_1}{1-\alpha_1}e^{\alpha_1(x-\hat{x})+\hat{x}}.\end{aligned}$$

Hence for Kou's model the integral estimate R is given by

$$R(\tau, x, \hat{x}) = -\frac{p\alpha_1}{1-\alpha_1}e^{\alpha_1(x-\hat{x})+\hat{x}} - Ke^{-r\tau}pe^{\alpha_1(x-\hat{x})}.$$

4.1.2 Discretization scheme $\mathcal{W}1$

The computational log-price domain will be the discretized set $\Omega_h := \{x_i : i = 1,\ldots,n\}$ with $x_i := -\hat{x} + (i-1)h$ for $i = 1,\ldots,n$, for a constant mesh size $h = 2\hat{x}/(n-1)$. Likewise in time $\tau_m := (m-1)k$, $m = 1,\ldots,q$, with $k = T/(q-1)$.
Now the integral term $\int_{\mathbb{R}} u(\tau_m, z)f(z-x_i)\,\mathrm{d}z$ can be approximated for each $x_i \in \Omega_h$ for example by using the composite trapezoidal rule on Ω and our remainder R on the complement of Ω like derived above:

$$\begin{aligned}\int_{\mathbb{R}} u(\tau_m, z)f(z-x_i)\,\mathrm{d}z \approx \frac{h}{2}\left[u_1^m f_{i,1} + 2\sum_{j=2}^{n-1} u_j^m f_{i,j} + u_n^m f_{i,n}\right]\\ + R(\tau_m, x_i, \hat{x}), \quad \forall i \in \{2,\ldots,n-1\},\end{aligned}$$

with $f_{ij} = f(x_j - x_i)$ and $u_j^m \approx u(\tau_m, x_j)$.
Almendral and Oosterlee [2005] now use the following finite difference scheme:

$$\begin{aligned}u_\tau(\tau_m, x_i) &\approx \begin{cases}(\frac{3}{2}u_i^m - 2u_i^{m-1} + \frac{1}{2}u_i^{m-2})/k, & \text{for } m \geq 2,\\ (u_i^m - u_i^{m-1})/k, & \text{for } m = 1,\end{cases}\\ u_{xx}(\tau_m, x_i) &\approx \frac{u_{i+1}^m - 2u_i^m + u_{i-1}^m}{h^2},\\ u_x(\tau_m, x_i) &\approx \frac{u_{i+1}^m - u_{i-1}^m}{2h}.\end{aligned}$$

So for the time derivative they use a backward difference formula of second order (BDF2) for $m \geq 2$, one of first order for the first time step (BDF1) and the ordinary second order central difference schemes for the spatial derivatives. Using this implicit scheme the linear system which we have to solve every time step looks like this:

$$\underbrace{(\omega_0 I + C + D)}_{=:A} u^m = b^m, \tag{4.2}$$

where

$$\omega_0 = \begin{cases} 1, & m = 1, \\ \frac{3}{2}, & m \geq 2. \end{cases}$$

I is the identity matrix and C, D are matrices defined by

$$\begin{aligned}
c_{ij} &= \begin{cases} -\frac{k\sigma^2}{2h^2} - \frac{k(r-\sigma^2/2-\lambda\eta)}{2h}, & i = j-1,\ 2 \leq i \leq n-1, \\ \frac{k\sigma^2}{h^2} + (r+\lambda)k, & i = j,\ 2 \leq i \leq n-1, \\ -\frac{k\sigma^2}{2h^2} + \frac{k(r-\sigma^2/2-\lambda\eta)}{2h}, & i = j+1,\ 2 \leq i \leq n-1, \\ 0, & \text{otherwise}, \end{cases} \\
d_{ij} &= \begin{cases} -\frac{1}{2}kh\lambda f_{ij}, & 2 \leq i \leq n-1,\ j \in \{1, n\}, \\ -kh\lambda f_{ij}, & 2 \leq i \leq n-1,\ 2 \leq j \leq n-1, \\ 0, & \text{otherwise}, \end{cases} \\
b_i^m &= k\lambda R(\tau_m, x_i, \hat{x}) + \omega_1 u_i^{m-1} + \omega_2 u_i^{m-2}, \quad i = 2, \ldots, n-1, \\
\omega_1 &= \begin{cases} 1, & \text{if } m = 1, \\ 2, & \text{if } m \geq 2, \end{cases} \\
\omega_2 &= \begin{cases} 0, & \text{if } m = 1, \\ -\frac{1}{2}, & \text{if } m \geq 2, \end{cases} \\
b_1 &= 0, \quad b_n = \omega_0(e^{\hat{x}} - Ke^{-r\tau_m}).
\end{aligned}$$

We denote this discretization scheme by $\mathcal{W}1$. The initial solution vector is $u^0 = [u_1^0, \ldots, u_n^0]^T = [H(e^{x_1}), \ldots, H(e^{x_n})]^T$, that means the payoff function H evaluated at n asset prices. Remember that this was originally the final condition at maturity. However, we are now going backwards in time and thus the former final condition becomes an initial one. In order to get more quickly an idea of how the matrix A looks like let us write it out for the case $m \geq 2$. Denote the diagonal element by $d := k\sigma^2/h^2 + (r+\lambda)k + 3/2 - cf_{ii}$, set $a := -k\sigma^2/(2h^2) - k(r - \sigma^2/2 - \lambda\eta)/(2h)$,

$b := -k\sigma^2/(2h^2) + k(r - \sigma^2/2 - \lambda\eta)/(2h)$ and $c := kh\lambda$. Then the $n \times n$ matrix A has the following form:

$$A = \begin{pmatrix} \frac{3}{2} & 0 & 0 & \dots & & & 0 \\ b - \frac{c}{2}f_{21} & d & a - cf_{23} & -cf_{24} & \dots & & -\frac{c}{2}f_{2n} \\ -\frac{c}{2}f_{31} & b - cf_{32} & \ddots & \ddots & \ddots & & -\frac{c}{2}f_{3n} \\ \vdots & -cf_{42} & \ddots & & & & \vdots \\ & \vdots & & & & a - cf_{n-2,n-1} & \\ -\frac{c}{2}f_{n-1,1} & -cf_{n-1,2} & \dots & & b - cf_{n-1,n-2} & d & a - \frac{c}{2}f_{n-1,n} \\ 0 & 0 & \dots & & 0 & 0 & \frac{3}{2} \end{pmatrix}. \tag{4.3}$$

Clearly, we have $u_1^m = b_1^m/\omega_0 = 0$ and $u_n^m = b_n^m/\omega_0 = (e^{\hat{x}} - Ke^{-r\tau_m})$ from the first and last row, respectively, and hence we can focus our attention on the $(n-2) \times (n-2)$ matrix $T = (A_{ij})_{2 \le i,j \le n-1}$ because we can easily reformulate our linear system $Au^m = b^m$ as $T\tilde{u}^m = \tilde{b}^m$ with $\tilde{u}^m = (u_2^m, \dots, u_{n-1}^m)^T$, $\tilde{b}_i^m = (b_i^m + cf_{in}u_n^m/2)$ for $i = 2, \dots, n-1$. The coefficient matrix T is dense but structured in that it belongs to the class of Toeplitz matrices because $f_{ij} = f(x_j - x_i) = f((j-i)h)$ and therefore f_{ij} is constant along the diagonals. Note that this property owes to the choice of a uniform grid. Overall we have $(T)_{ij} = t_{i-j}$ for all $1 \le i \le n-2$, which is exactly the Toeplitz property.

Let us take a closer look at the properties of T for the Merton case: Since the normal density f_M defined in (3.1) with $\mu_J = 0$ is even, for $f(x) := f_M(x)$ we get $f_{ij} = f_M((j-i)h) = f_M(-(j-i)h) = f_{ji}$. Hence we observe that for Merton's model T is symmetric except for the first upper and lower subdiagonal. They differ in the sign of one of the terms, originating from the central difference scheme for u_x. This observation leads us to the conclusion that getting rid of u_x would help us a lot in the Merton case because it provides us with a broader range of applicable methods of how to solve the linear system (4.2), in addition to avoiding the oscillation problem that comes with the central difference quotient for the convection term.

4.2 Finite Differences approach for PIDE $\mathcal{N}$ on uniform grid

4.2.1 Domain and integral truncation

In the last section the domain for x was restricted to $(-\hat{x}, \hat{x})$, therefore we restrict the domain for ξ to $\tilde{\Omega} = (\xi_-, \xi_+)$ with $\xi_- := -\hat{x} + \zeta T$ and $\xi_+ := \hat{x} + \zeta T$, since then in the final time T the function $u(T, x)$ on $(-\hat{x}, \hat{x})$ is equal to $w(T, \xi)$ on (ξ_-, ξ_+). Next we take care of the integral outside $\tilde{\Omega}$. Like done in the last section, for a European call we get

$$\begin{aligned} w(\tau, \xi) &\longrightarrow 0, && (\xi \to -\infty), \\ w(\tau, \xi) &\longrightarrow e^{\xi - \zeta\tau} - Ke^{-r\tau}, && (\xi \to +\infty), \end{aligned}$$

and analogously the estimation R of the integral term outside $\tilde{\Omega}$:

$$R(\tau, \xi, \xi_+) = \int_{\xi_+}^{+\infty} (e^{\tilde{z} - \zeta\tau} - Ke^{-r\tau}) f(\tilde{z} - \xi)\, \mathrm{d}\tilde{z},$$

and using the cumulative normal distribution again yields for Merton's model (see last section for a similar derivation)

$$R_M(\tau, \xi, \xi_+) = e^{-\zeta\tau} e^{\xi + \sigma_J^2/2} \Phi\left(\frac{\xi - \xi_+ + \sigma_J^2}{\sigma_J}\right) - Ke^{-r\tau} \Phi\left(\frac{\xi - \xi_+}{\sigma_J}\right).$$

For Kou's model we are using the double-exponential density (3.3) leading us to an accordingly different estimate:

$$\begin{aligned} R_K(\tau, \xi, \xi_+) &= \int_{\xi_+}^{\infty} (e^{z - \zeta\tau} - Ke^{-r\tau}) f_K(z - \xi)\, \mathrm{d}z \\ &= p\,\alpha_1 e^{-\zeta\tau} \int_{\xi_+}^{\infty} e^{(1-\alpha_1)z + \xi\alpha_1}\, \mathrm{d}z - p\,\alpha_1 K e^{-r\tau} \int_{\xi_+}^{\infty} e^{-\alpha_1(z-\xi)}\, \mathrm{d}z \\ &= -\frac{p\,\alpha_1}{1 - \alpha_1} e^{-\zeta\tau + \alpha_1(\xi - \xi_+) + \xi_+} - p\,K e^{-r\tau + \alpha_1(\xi - \xi_+)}. \end{aligned}$$

4.2.2 Discretization scheme $\mathcal{N}1$

Denote $\xi_i := \xi_- + (i-1)h$, for $i = 1, \ldots, n$ with uniform step size $h = (\xi_+ - \xi_-)/(n-1) = 2\hat{x}/(n-1)$, and as usual $\tau_m := (m-1)k$ for $m = 1, \ldots, q$. Then $w_i^m \approx w(\tau_m, \xi_i)$

and $f_{ij} := f(\xi_j - \xi_i)$. The integral will be approximated by the composite trapezoidal rule:

$$\int_{\mathbb{R}} w(\tau_m, \tilde{z}) f(\tilde{z} - \xi_i)\,\mathrm{d}\tilde{z} \approx \frac{h}{2}\left[w_1^m f_{i,1} + 2\sum_{j=2}^{n-1} w_j^m f_{i,j} + w_n^m f_{i,n}\right] + R(\tau_m, \xi_i, \xi_+), \quad \forall i \in \{2, \ldots, n-1\}.$$

We use again the finite difference schemes

$$\begin{aligned} w_\tau(\tau_m, \xi_i) &\approx \begin{cases} (\frac{3}{2}w_i^m - 2w_i^{m-1} + \frac{1}{2}w_i^{m-2})/k, & \text{for } m \geq 2, \\ (w_i^m - w_i^{m-1})/k, & \text{for } m = 1, \end{cases} \\ w_{\xi\xi}(\tau_m, \xi_i) &\approx \frac{w_{i+1}^m - 2w_i^m + w_{i-1}^m}{h^2}. \end{aligned}$$

Our linear system looks identical to the one derived previously, except for the fact that due to the absence of the first derivative term the coefficient matrix becomes symmetric.

$$\underbrace{(\gamma_0 I + C + D)}_{=:A} w^m = b^m, \tag{4.4}$$

where

$$\gamma_0 = \begin{cases} 1, & m = 1, \\ \frac{3}{2}, & m \geq 2. \end{cases}$$

I is the identity matrix and C, D are matrices defined by

$$c_{ij} = \begin{cases} -\frac{k\sigma^2}{2h^2}, & i = j-1,\ 2 \leq i \leq n-1, \\ \frac{k\sigma^2}{h^2} + (r+\lambda)k, & i = j,\ 2 \leq i \leq n-1, \\ -\frac{k\sigma^2}{2h^2}, & i = j+1,\ 2 \leq i \leq n-1, \\ 0, & \text{otherwise}, \end{cases}$$

$$d_{ij} = \begin{cases} -\frac{1}{2}kh\lambda f_{ij}, & 2 \leq i \leq n-1,\ j \in \{1, n\}, \\ -kh\lambda f_{ij}, & 2 \leq i \leq n-1,\ 2 \leq j \leq n-1, \\ 0, & \text{otherwise}, \end{cases}$$

$$
\begin{aligned}
b_i^m &= k\lambda R(\tau_m, \xi_i, \xi_+) + \gamma_1 w_i^{m-1} + \gamma_2 w_i^{m-2}, \quad i = 2, \ldots, n-1, \qquad (4.5)\\
\gamma_1 &= \begin{cases} 1, & m = 1, \\ 2, & m \geq 2, \end{cases} \\
\gamma_2 &= \begin{cases} 0, & m = 1, \\ -1/2, & m \geq 2. \end{cases}
\end{aligned}
$$

The first and last component of the right hand side b^m are given by the boundary conditions, that means $b_1^m = 0$ and $b_n^m = \gamma_0(e^{\xi_+ - \zeta\tau_m} - Ke^{-r\tau_m})$. Again reformulating the system one obtains an $(n-2) \times (n-2)$ system with the coefficient matrix

$$T = (A_{ij})_{2 \leq i,j \leq n-1}. \qquad (4.6)$$

Observe that T is symmetric; together with positive definiteness it is now possible to apply the preconditioned Conjugate Gradient (PCG) method. In order to ensure that T is indeed positive definite we analyze the eigenvalues of T in Chapter 6 for Merton's model. This can be done in an elegant way by using the concept of a generating function for a Toeplitz operator. For Kou's model with asymmetric density the matrix T is asymmetric, regardless of whether we have scheme $\mathcal{W}1$ or $\mathcal{N}1$. Possible linear solvers are for example PCG on the normal equations, GMRES or splitting iterations. The latter approach was used by several researchers, among them *Almendral and Oosterlee* [2005] and *Ikonen and Toivanen* [2006], because it turns out to be fast convergent and applicable to asymmetric systems. We take a look at this approach before proposing an alternative way using PCG.

Chapter 5

Splitting iterations using scheme $\mathcal{W}1$ for Merton's and Kou's model

Starting with this chapter we present different ways of how to efficiently solve the linear systems resulting from the discretization schemes discussed in Chapter 4. Remember that we are dealing with dense Toeplitz systems in any case that might or might not be symmetric. A straightforward approach could be to apply a splitting method. This is the way taken by *Ikonen and Toivanen* [2006] and of *Almendral and Oosterlee* [2005].
After a brief review of splitting techniques and a discussion on sufficient criteria for convergence, we go through some implementation issues that turn out to yield striking effectiveness with regard to the amount of needed operations.

5.1 Splitting iterations

First of all, we review what is understood by splitting iterations: A decomposition $T = M - N$ is called a *splitting* of the matrix T. There exist various possibilities to define the matrix M, often M is the diagonal of T or the lower triangular part of T including the diagonal. The former is known as Jacobi splitting, the latter as Gauss-Seidel splitting. Such a splitting can be used to define a fixed point iteration

that solves the linear system $Tx = b$ under certain assumptions:

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b, \qquad x_0 = \mathbf{0}. \tag{5.1}$$

As usual, the inverse of M is not computed explicitly but we rather solve the system

$$Mx_{k+1} = Nx_k + b. \tag{5.2}$$

As an immediate consequence M should be chosen such that this system is easy to solve, since we have to do that in every iteration. In addition to that, notice that we also have to compute a matrix-vector product in each iteration. What else governs the choice of M? Well, clearly we would prefer the method to converge fast. This will probably be the case if M is a good approximation of T because we can interpret M as a preconditioner of T. To see this, note that at the solution x_* the following holds:

$$\begin{aligned} x_* &= M^{-1}Nx_* + M^{-1}b \\ \Rightarrow (I - M^{-1}N)x_* &= M^{-1}b \\ \Rightarrow (I - I + M^{-1}T)x_* &= M^{-1}Tx_* = M^{-1}b. \end{aligned} \tag{5.3}$$

Consequently the fixed point iteration (5.1) can be seen as solving this left-preconditioned system. Thus we expect the splitting method to converge quickly if M is a good preconditioner of A. More precisely the speed of the convergence depends on the *spectral radius* $\rho(M^{-1}N) = \max\{|\lambda| : \lambda \text{ eigenvalue of } M^{-1}N\}$. The reason can be found by looking at the error in each splitting iteration where we use (5.3):

$$e_{k+1} = x_{k+1} - x_* = M^{-1}N(x_k - x_*) = (M^{-1}N)^{k+1}e_0. \tag{5.4}$$

If $M^{-1}N$ has n linearly independent eigenvectors we can write $e_0 = \sum \epsilon_i v_i$, where v_i is the ith eigenvector of $M^{-1}N$. Accordingly $e_{k+1} = \sum \epsilon_i \lambda_i^{k+1} v_i$ which will tend to zero if and only if all eigenvalues satisfy $|\lambda_i| < 1$ or equivalently if the spectral radius satisfies $\rho(M^{-1}N) < 1$. In order to find a bound for the convergence rate we see that the slowest possible convergence would occur if e_0 would point purely in the direction of the eigenvector belonging to the eigenvalue with maximal magnitude. In this case we would have at least a decrease in error given by

$$\|e_k\| \approx \rho(M^{-1}N)\|e_{k-1}\| \approx \rho(M^{-1}N)^k\|e_0\|.$$

Therefore the smaller the spectral radius, the faster the method will converge. Knowledge about $\rho(M^{-1}N)$ can also be used to predict the amount of iterations that one

needs at least to reduce the error to a certain percentage of the initial error. For example, aiming for an error reduction to one percent of the initial error we would solve the following inequality for k with $m = 2$:

$$\|e_k\| \approx \rho(M^{-1}N)^k \|e_0\| \leq 10^{-m}\|e_0\|$$
$$k \geq \frac{-m}{\log_{10}(\rho(M^{-1}N))} \tag{5.5}$$

We will need at least k iterations, more may be necessary.

For the linear systems arising from the discretization schemes $\mathcal{W}1$ and $\mathcal{N}1$ the main weight lies on the three main diagonals. Due to the exponential decay of the density function in both Merton's and Kou's model, the elements on the subdiagonals tend fast to zero when moving away from the main diagonal. Accordingly, it makes sense to define M as the three main diagonals of T.

Of course the question arises as to whether this fixed point method is convergent at all. Before giving a sufficient condition we first need another definition:

Definition 5.1. *Let $T = M - N$. The pair M, N is called a* regular splitting *of $T \in \mathbb{R}^{n\times n}$ if M is nonsingular and M^{-1} and N are nonnegative, where nonnegative simply means that all entries are nonnegative.*

We stated above that the splitting method converges if and only if the spectral radius $\rho(M^{-1}N) < 1$. This inequality can be characterized by two conditions on T if M and N form a regular splitting of T:

Theorem 5.2 (*Saad* [1996])**.** *Let M, N be a regular splitting of a matrix $T \in \mathbb{R}^{n\times n}$. Then*

$$\rho(M^{-1}N) < 1 \quad \Leftrightarrow \quad T \text{ is nonsingular and } T^{-1} \text{ is nonnegative.}$$

So what we need to show is that T is nonsingular, T^{-1} is nonnegative and that $M = \text{tridiag}(T)$ defines a regular splitting. As for the last condition, we take advantage of a result by *Young* [1971] stating that if T is a so-called M-matrix, then any splitting $T = M - N$ formed from T by replacing some of the off-diagonal elements by zero is a regular splitting. Thus $M = \text{tridiag}(T)$ defines a regular spliting if T has this property. M-matrices can be defined in the following way:

Definition 5.3. *$T \in \mathbb{R}^{n\times n}$ is called an* M-matrix *if*

- *its main diagonal entries are all positive.*
- *its off-diagonal entries are all nonpositive.*
- *T is nonsingular.*
- *T^{-1} is nonnegative.*

There are other equivalent ways of how to define it, for example:

Definition 5.4. *$T \in \mathbb{R}^{n\times n}$ is called an M-matrix if*

- *its off-diagonal entries are nonpositive.*
- *there exists an $v \in \mathbb{R}^n$ with nonnegative entries such that Tv is positive.*

From Definition 5.3 we see that the iteration will always converge if M, N is a regular splitting and T is an M-matrix. Let us impose some conditions that guarantee that T is an M-matrix. According to Definition 5.4 we need nonpositive off-diagonal entries. The terms from the integral discretization are negative anyway, so all we need to ensure is that the subdiagonal contributions from the non-integral terms are also nonnegative:

(a) $-\frac{k\sigma^2}{2h^2} + \frac{k(\sigma^2/2+\lambda\eta)}{2h} \leq 0.$
(b) $-\frac{k\sigma^2}{2h^2} - \frac{k(\sigma^2/2+\lambda\eta)}{2h} \leq 0.$

Next we choose $v := [1, 1, \ldots, 1]^T$ and consider the product Tv which is accordingly the vector consisting of the row sums of T. For usual choices of parameters the matrix T is diagonally dominant because the contribution from the integral discretization is very small, so the condition $Tv > 0$ is in this case satisfied. The conditions are not too restrictive since the coefficient of the integral contributions $c := kh\lambda$ is small by itself and the density f is decaying exponentially moving away from the main diagonal. *Almendral and Oosterlee* [2005] concluded that even the conditions (a) and (b) "are sufficient for an accurate stable solution."

5.2 Implementation

As for the question of how to practically implement this technique, let us start by summarizing what a straightforward implementation would cost. Like pointed out earlier the dominant operations are

- to compute the matrix-vector product Nx_k.
- to solve the system $Mx_{k+1} = Nx_k + b$.

Direct computation of the matrix-vector product Nx_k would require $O(n^2)$ operations. But remember that N is also a Toeplitz matrix. Furthermore, recall that we saw in Chapter 2 that for circulant as well as for Toeplitz matrices this product can be accelerated by means of the FFT algorithm. More precisely, we embed N in a circulant $2n \times 2n$ matrix like done in (2.11) and compute Nx_k in just three FFTs of length $2n$ and one vector multiplication. Actually, this can be even more accelerated in case that the coefficient matrix T does not change over the time because one of those three FFTs is computing the eigenvalues of the circulant embedding of N. Remember that we any circulant matrix C has the decomposition $C = F_n^* \Lambda_n F_n$, so the diagonal matrix Λ_n consists of these eigenvalues. Now if N stays the same for all time steps, as it is the case for the discretization schemes $\mathcal{W}1$ and $\mathcal{N}1$, then we simply compute Λ_n once at the very beginning and thus get along with two FFTs only in each splitting iteration. In any case, using FFTs only $O(n \log n)$ operations are needed, a considerable improvement.
Solving the linear system $Mx_{k+1} = Nx_k + b$ is even faster since we chose $M = \text{tridiag}(T)$. Direct solvers yield the solution in $O(n)$ operations, for example the so-called *Thomas algorithm*. For this method we need M to be diagonally dominant which is easily verified for the conducted experiments later on. It first eliminates the subdiagonal in our tridiagonal Toeplitz system

$$\begin{pmatrix} a & b & 0 & \cdots & 0 \\ c & \ddots & \ddots & & \vdots \\ 0 & \ddots & & & \\ & & & & b \\ 0 & \cdots & & c & a \end{pmatrix} \vec{x} = \begin{pmatrix} d_1 \\ \vdots \\ \\ \\ d_n \end{pmatrix},$$

so we obtain the system

$$\begin{pmatrix} \tilde{a}_1 & b & 0 & \cdots & 0 \\ 0 & \tilde{a}_2 & \ddots & & \vdots \\ \vdots & & \ddots & & \\ & & & & b \\ 0 & \cdots & & 0 & \tilde{a}_n \end{pmatrix} \vec{x} = \begin{pmatrix} \tilde{d}_1 \\ \vdots \\ \\ \\ \tilde{d}_n \end{pmatrix}.$$

Backward substitution yields the solution. We use the method in the following form, similar to the one stated in *Conte and de Boor* [1980]:

Algorithm 5.5. Thomas Tridiagonal Solver:

1. $\tilde{a}_1 := a$
2. $\tilde{d}_1 := d_1$
3. For $i = 2 : n$ Do
4. $\quad m := c/\tilde{a}_{i-1}$
5. $\quad \tilde{a}_i := a - m \cdot b$
6. $\quad \tilde{d}_i := d_i - m \cdot \tilde{d}_{i-1}$
7. End Do
8. $x_n := \tilde{d}_n/\tilde{a}_n$
9. For $i = n - 1 : 1$ Do
10. $\quad x_i := (\tilde{d}_i - b \cdot x_{i+1})/\tilde{a}_i$
11. End Do

Remark 5.6. Note that we only need $5n - 4$ multiplications and $3n - 3$ additions to solve the system (5.2) with this method.

Concerning the required storage all we need is

- the first column of the circulant embedding for N, that is a vector of length $2n$.
- three scalars for the the diagonal entries of M.
- the right hand side b, that is a vector of length n.

Overall we conclude that this approach is likely to produce good results as soon as the spectral radius $\rho(M^{-1}N)$ is small, that is just a few iterations are needed and each iteration is carried out in only $O(n \log n)$ operations. Indeed, in the numerical experiments we find that $\rho(M^{-1}N)$ is very small, hence the method converges rapidly after about three iterations. An in-depth discussion of the numerical performance is presented in Chapter 8.

Chapter 6

Preconditioned CG using scheme $\mathcal{N}1$ for Merton's model

In the subsequent sections we propose a more elegant way of solving the PIDE for Merton's model than the one presented before. We have seen in the fourth chapter that the implicit discretization scheme $\mathcal{N}1$ for the convection-free PIDE $\mathcal{N}$ resulted in a symmetric linear Toeplitz system that we need to solve in every time step. If this system is positive definite and if a good preconditioner can be found, it is worthwhile to consider the conjugate gradient method as an iterative solver rather than a splitting technique, because there exist many theoretical results on the application of preconditioned CG to Toeplitz systems that suggest fast convergence. In particular, attractive preconditioners have been developed for Toeplitz systems so that a very good performance can be expected.
After analyzing the coefficient matrix in particular with regard to positive definiteness and eigenvalue distribution in Section 1, we then discuss the advantages and drawbacks of several preconditioners of circulant type and conclude the chapter with some comments on efficient practical implementation.

6.1 On the eigenvalues of the coefficient matrix

Naturally, before we can implement the conjugate gradient method we need to ensure that the coefficient matrix T is both symmetric and positive definite. For Merton's model we established the symmetry already by construction of the discretization scheme $\mathcal{N}1$. Note that in Kou's model the density function is in general asymmetric and thus leads to an asymmetric coefficient matrix T. So CG cannot be applied directly to the linear systems arising in Kou's model.
However, in Merton's model the symmetry is given, but what about the location of the eigenvalues and their distribution? We would not only like to know in what interval they are but also if they are clustered, because we saw in Chapter 2 that usually few distinct eigenvalues entail few CG iterations. In order to analyze T for these features we need some theoretical results on Toeplitz operators.
Let us begin with the definition of a function g which generates a fixed singly-infinite Toeplitz matrix T_∞ with T_n being the $n \times n$ principal submatrix. This so-called *generating function* g is defined by the Fourier series

$$g(x) = \sum_{j=-\infty}^{\infty} t_j e^{-ijx}, \quad x \in [-\pi, \pi], \tag{6.1}$$

where t_j is the entry of T_∞ on the jth subdiagonal or in other words, the (i,j)th entry of both T_∞ and T_n is given by t_{i-j}. This function generates Toeplitz matrices T_n in the sense that the diagonals of T_n are given by

$$t_j = \frac{1}{2\pi} \int_{-\pi}^{\pi} g(x) e^{ijx} \, \mathrm{d}x, \qquad j = 0, \pm 1, \pm 2, \ldots,$$

and we denote the so-defined matrices by $T_n[g]$. If g is real-valued, we have

$$t_{-j} = \frac{1}{2\pi} \int_{-\pi}^{\pi} g(x) e^{-ijx} \, \mathrm{d}x = \bar{t}_j, \qquad \forall \; j,$$

and hence $T_n[g]$ must be Hermitian for all n. If g is additionally an even function, that is $g(-x) = g(x)$, using the change of variable $y = -x$ we obtain

$$\begin{aligned} t_{-j} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} g(x)e^{-ijx}\,\mathrm{d}x \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} g(-x)e^{ij(-x)}\,\mathrm{d}x \\ &= -\frac{1}{2\pi}\int_{\pi}^{-\pi} g(y)e^{ijy}\,\mathrm{d}y \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} g(y)e^{ijy}\,\mathrm{d}y \\ &= t_j = \bar{t}_j, \end{aligned} \tag{6.2}$$

so $T_n[g]$ are real and symmetric for all n in this case.
The importance of the generating function lies in the fact that there exists a close relationship between g and the spectra of T_n. Not only forms the minimum (maximum) of g a lower (upper) bound for the eigenvalues of T_n, also one can deduce information about the eigenvalue distribution from g.

Lemma 6.1 (*Widom* [1965]). *Let g be a real-valued function in $L^2[-\pi,\pi]$. If T_∞ is invertible, then $1/g$ is essentially bounded. If the essential range of $1/g$ lies entirely to one side of a line through the origin but does not intersect the line, then T_∞ is invertible. Moreover, the spectrum of T_∞ contains the essential range of g and is contained in the convex hull of the essential range of g.*

The following result enables us in particular to determine if T_n is positive definite:

Theorem 6.2 (*Grenander and Szegö* [1984]). *Let g be a real-valued function in $L^1[-\pi,\pi]$. Then the spectrum $\sigma(T_n)$ of T_n satisfies*

$$\sigma(T_n) \subseteq [g_{min}, g_{max}], \quad \forall\ n \geq 1,$$

where g_{min} and g_{max} are the essential infimum and the essential supremum of g respectively. Moreover, if $g_{max} > g_{min}$, then

$$g_{min} < \lambda_{min}(T_n) \leq \lambda_{max}(T_n) < g_{max}.$$

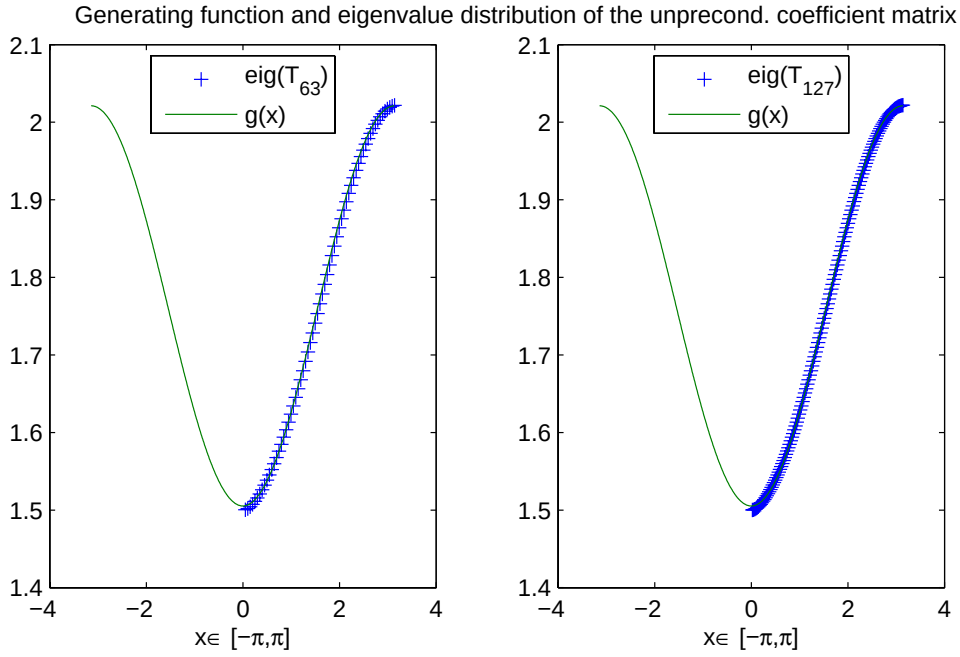

Figure 6.1. Generating function and eigenvalues of generated Toeplitz matrices for Merton's model and $m \geq 2$, $n = 63$ and $n = 127$. Parameters: $h = 0.125$, $k = 0.1$, $r = 0$, $\sigma_J = 0.5$, $\sigma = 0.2$ and $\lambda = 0.1$.

In particular, if $g_{min} > 0$, then T_n is positive definite for all n.

We also obtain information on the distribution of the eigenvalues of T_n. Before getting to that we need one more definition:

Definition 6.3. *Let g be a real-valued function in $L^1[-\pi, \pi]$. A sequence $(\alpha_k^{(n)})$ is said to be* equally distributed as $g(x)$ *if*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} F(\alpha_k^{(n)}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(g(x))\, \mathrm{d}x$$

for any continuous function F with bounded support.

Theorem 6.4 (*Grenander and Szegö* [1984])**.** *Let $g \in L^2[-\pi, \pi]$. Then the singular values of the matrices T_n generated by g are equally distributed as $|g(x)|$. In particular, for real-valued g, the eigenvalues of T_n are equally distributed as $g(x)$.*

For an intuitive understanding of the definition for "equally distributed as $g(x)$" let us take a look at Figure 6.1. Both plots contain the graph of the same generating function

$g(x)$ as well as the separately computed eigenvalues of the matrices T_{63} and T_{127} that are generated by $g(x)$. These eigenvalues are plotted against a uniform grid on the interval $[0, \pi]$ and apparently follow the curve g independent of the dimension n; they are uniformly distributed as g. The larger n, the denser are the eigenvalues along the graph of g. We will come back to this figure before long, but first we should derive $g(x)$ explicitly for our coefficient matrix T arising from the discretization scheme $\mathcal{N}1$ for Merton's model. For a given set of model parameters

$$h, k, \sigma, \sigma_J, r, \lambda, \tag{6.3}$$

let $a := k\sigma^2/h^2$, $b := kh\lambda$, and using the normal density $f := f_M$ defined in (3.1) yields

$$f_{ij} = f_M((j-i)h) = f_M(\kappa h) = \frac{1}{\sqrt{2\pi}\sigma_J} e^{-(\kappa h)^2/(2\sigma_J^2)},$$

where κ denotes $j - i$, so f_{ij} is on the κth subdiagonal. Then the diagonals t_κ of the coefficient matrix T defined in (4.6) for $m \geq 2$ are determined by

$$\begin{aligned} t_0 &= a + (r+\lambda)k + \frac{3}{2} - \frac{b}{\sqrt{2\pi}\sigma_J}, \\ t_1 = t_{-1} &= -\frac{a}{2} - \frac{b}{\sqrt{2\pi}\sigma_J} e^{-h^2/(2\sigma_J^2)}, \\ t_\kappa = t_{-\kappa} &= -\frac{b}{\sqrt{2\pi}\sigma_J} e^{-(\kappa h)^2/(2\sigma_J^2)}, \quad \kappa \geq 2. \end{aligned} \tag{6.4}$$

Now that we got formulae for the diagonal elements, we can easily derive the generating function $g(x)$ for a given set of parameters (6.3) by using the definition of a generating function given in (6.1):

$$\begin{aligned} g(x) &= \sum_{j=-\infty}^{\infty} t_j e^{-ijx} \\ &= \sum_{j \neq -1,0,1} \frac{-b}{\sqrt{2\pi}\sigma_J} e^{-(jh)^2/(2\sigma_J^2)} e^{-ijx} + t_{-1}e^{ix} + t_0 + t_1 e^{-ix} \\ &= -\frac{b}{\sqrt{2\pi}\sigma_J} \sum_{j=2}^{\infty} e^{-(jh)^2/(2\sigma_J^2)} (\cos(-jx) + i\sin(-jx)) \\ &\quad - \frac{b}{\sqrt{2\pi}\sigma_J} \sum_{j=2}^{\infty} e^{-(jh)^2/(2\sigma_J^2)} (\cos(jx) + i\sin(jx)) \\ &\quad + t_{-1}(\cos x + i\sin x) + t_0 + t_1 (\cos(-x) + i\sin(-x)) \\ &= -\frac{b}{\sqrt{2\pi}\sigma_J} \left[2\sum_{j=2}^{\infty} e^{-(jh)^2/(2\sigma_J^2)} \cos(jx) \right] + 2t_1 \cos x + t_0. \end{aligned} \tag{6.5}$$

The case $m = 1$ can of course be treated analogously. We observe that $g(x)$ is real-valued and that $g(x) = g(-x)$, so $T_n[g]$ are all real and symmetric as seen in (6.2). An important fact is that the Fourier coefficients t_j of g are absolutely summable, which we prove in the next lemma. Any generating function having this property is said to be in *Wiener class*. This harmless looking property is of great value when we are to look for an appropriate preconditioner. Actually, almost all results in the context of clustering behavior and positive definiteness of the preconditioners are subject to the assumption that the generating function of the Toeplitz matrix is positive and in Wiener class. Luckily, this is the case for our problem. First we show that it is in Wiener class:

Lemma 6.5. *The real-valued and even function g defined in (6.5) is in Wiener class, that means its Fourier coefficients t_j are absolutely summable:*

$$\sum_{j=0}^{\infty} |t_j| < \infty. \tag{6.6}$$

Furthermore, $g(x)$ is differentiable on $(-\pi, \pi)$ and its derivative is given by

$$g'(x) = \frac{b}{\sqrt{2\pi}\sigma_J} \left[2 \sum_{j=2}^{\infty} e^{-(jh)^2/(2\sigma_J^2)} \sin(jx) \cdot j \right] - 2t_1 \sin x.$$

Proof. With t_j defined in (6.4) for $j \in \{0, 1, 2, \ldots\}$ we have

$$\begin{aligned} \sum_{j=0}^{\infty} |t_j| &= |t_0| + |t_1| + \sum_{j=2}^{\infty} |t_j|, \\ \sum_{j=2}^{\infty} |t_j| &\leq \frac{b}{\sqrt{2\pi}\sigma_J} \sum_{j=2}^{\infty} \underbrace{e^{-(jh)^2/(2\sigma_J^2)}}_{=:a_j}. \end{aligned}$$

Applying the ratio test yields absolute convergence of the series:

$$\begin{aligned} \left| \frac{a_{j+1}}{a_j} \right| &= \exp\left(-\frac{1}{2} \left(\frac{(j+1)h}{\sigma_J} \right)^2 + \frac{1}{2} \left(\frac{jh}{\sigma_J} \right)^2 \right) \\ &= \exp\left(\frac{1}{2} \left(\frac{h}{\sigma_J} \right)^2 (-2j - 1) \right) \\ &\leq \exp\left(-\frac{1}{2} \left(\frac{h}{\sigma_J} \right)^2 \right) =: \Theta < 1, \qquad \forall j \geq 2. \end{aligned}$$

Hence it follows that g is in Wiener class.

In order to prove that g is differentiable we also just need some elementary real analysis. We need to show that $\gamma(x) := \sum_{j=2}^{\infty} \exp(-(jh)^2/(2\sigma_J^2)) \cos(jx)$ is differentiable and that $\gamma'(x) = -\sum_{j=2}^{\infty} j \exp(-(jh)^2/(2\sigma_J^2)) \sin(jx)$. Consider $\gamma_n(x)$, the nth partial sum of $\gamma(x)$. This sum $\gamma_n(x)$ is of course continuously differentiable on $I := (-\pi, \pi)$, and for all $x \in I$ it holds that $\gamma_n(x) \to \gamma(x)$ for $n \to \infty$. Next we show that $\gamma_n'(x)$ converges uniformly to $\gamma'(x)$. The derivative of $\gamma_n(x)$ is given by $\gamma_n'(x) = -\sum_{j=2}^{n} \underbrace{j \exp(-(jh)^2/(2\sigma_J^2)) \sin(jx)}_{=:u_j}$. The uniform convergence follows by the Weierstrass criterium: $|u_j| \leq j \exp(-(jh)^2/(2\sigma_J^2)) =: M_j$, and $\sum_2^{\infty} M_j$ converges by the ratio test:

$$\begin{aligned}\left|\frac{M_{j+1}}{M_j}\right| &= \exp(-h^2/(2\sigma_J^2)(2j+1)) + \frac{1}{j}\exp(-h^2/(2\sigma_J^2)(2j+1)) \\ &\leq 2\exp(-h^2/(2\sigma_J^2)(2N+1)) < 1 \quad \forall j \geq N,\end{aligned}$$

for N sufficiently large. Hence the statement follows. □

We remark that any function g in the Wiener class is 2π-periodic, continuous and real-valued, see for example *Chan and Yeung* [1992]. The reason is that the Fourier series associated with g has these properties and converges uniformly to g on $[-\pi, \pi]$ by the Weierstrass test.

Note that Theorem 6.2 and Theorem 6.4 can be applied to the matrix T_n, that means the spectrum of T_n is a subset of $[g_{min}, g_{max}]$. Therefore we want to know if the minimum of g is positive since in this case T_n is positive definite for all n. From now on let us denote the coefficient matrix T rising from the discretization scheme $\mathcal{N}1$ by T_n to emphasize that it is the $n \times n$ principal submatrix of the generated singly infinite matrix T_∞. Once a set of parameters (6.3) is given, the generating function g is then given as well for any dimension n. Different dimensions correspond to different choices for the size of the computational domain. Obviously, extending the domain whilst holding the parameters (6.3) including the step size h constant results in a larger space grid size n. Hence, changing ceteris paribus the domain size will only cause the eigenvalues to lie more or less dense along the generating function g, but

always in the same interval $[g_{min}, g_{max}]$. Precisely this behavior can be observed in Figure 6.1 where the graph on the right shows the eigenvalues of the coefficient matrix that was obtained by doubling the domain size of the one in the left plot. So once for a given set of parameters it is established that $g_{min} > 0$, then for every domain size T_n will be positive definite.
The next lemma provides a sufficient criterion for g being positive on its whole domain:

Lemma 6.6. *Let g be the function defined in (6.5). Then g is positive on $[-\pi, \pi]$ if the following inequality holds:*

$$-\frac{kh\lambda}{\sqrt{2\pi}\sigma_J}\left[\frac{4}{\exp(h^2/(2\sigma_J^2)) - \exp(-h^2/(2\sigma_J^2))} + 1\right] + (r+\lambda)k + \frac{3}{2} > 0, \tag{6.7}$$

where the parameters k, h, λ, σ_J and r are the same as in the definition of g.

Proof. We need to show that $g(x) > 0$ on $[-\pi, \pi]$ for g defined by

$$g(x) = -\frac{b}{\sqrt{2\pi}\sigma_J}\left[2\sum_{j=2}^{\infty} e^{-(jh)^2/(2\sigma_J^2)}\cos(jx)\right] + 2t_1\cos x + t_0, \tag{6.8}$$

where $t_0 := a + (r+\lambda)k + 3/2 - b/(\sqrt{2\pi}\sigma_J)$, $t_1 := -a/2 - b\exp(-h^2/(2\sigma_J^2))/(\sqrt{2\pi}\sigma_J)$, $a := k\sigma^2/h^2$ and $b := kh\lambda$. For the hyperbolic cosecant holds (*Gradshteyn and Ryzhik* [2000]):

$$\operatorname{cosech}(y) = 2\sum_{j=0}^{\infty} e^{-(2j+1)y} = 2\sum_{j \text{ odd} \in \mathbb{Z}_+} e^{-yj}. \tag{6.9}$$

Hence for $y > 0$:

$$\begin{aligned} 2\sum_{j=1}^{\infty} \underbrace{e^{-yj^2}}_{\leq e^{-yj}} \underbrace{\cos(jx)}_{\leq 1} &\leq 2\sum_{j=1}^{\infty} e^{-yj} \\ &= \underbrace{2\sum_{j \text{ odd} \in \mathbb{Z}_+} e^{-yj}}_{=\operatorname{cosech}(y)} + \underbrace{2\sum_{j \text{ even} \in \mathbb{Z}_+} e^{-yj}}_{\leq \operatorname{cosech}(y)} \\ &\leq 2\ \operatorname{cosech}(y) = 2\frac{1}{\sinh(y)} = \frac{4}{\exp(y) - \exp(-y)}. \end{aligned}$$

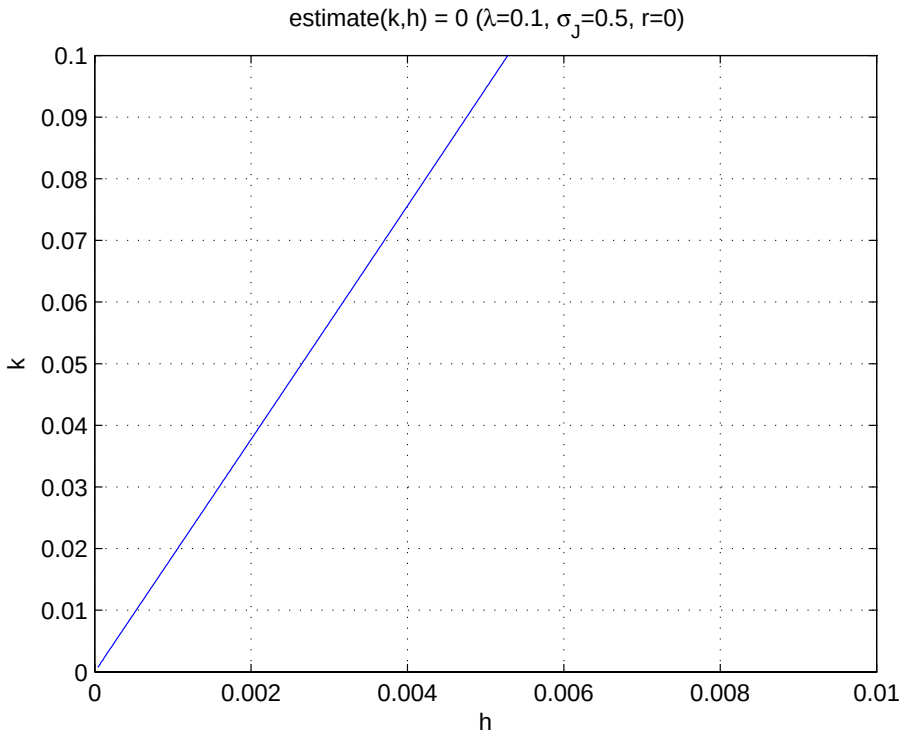

Figure 6.2. Right half space contains combinations of h and k for which the Toeplitz matrix generated by g with parameters as indicated is guaranteed to be positive definite.

Define $y := h^2/(2\sigma_J^2)$. Then for any $x \in [-\pi, \pi]$:

$$\begin{aligned} g(x) &= -\frac{b}{\sqrt{2\pi}\sigma_J}\left[2\sum_{j=1}^{\infty} e^{-yj^2}\cos(jx)\right] + a(1-\cos(x)) + (r+\lambda)k + \frac{3}{2} - \frac{b}{\sqrt{2\pi}\sigma_J} \\ &\geq -\frac{b}{\sqrt{2\pi}\sigma_J}\cdot\frac{4}{\exp(y)-\exp(-y)} + a(1-\underbrace{\cos(x)}_{\leq 1}) + (r+\lambda)k + \frac{3}{2} - \frac{b}{\sqrt{2\pi}\sigma_J} \\ &\geq -\frac{kh\lambda}{\sqrt{2\pi}\sigma_J}\left[\frac{4}{\exp(h^2/(2\sigma_J^2))-\exp(-h^2/(2\sigma_J^2))}+1\right] + (r+\lambda)k + \frac{3}{2} \\ &=: \texttt{estimate}(k,h). \end{aligned}$$

Thus a sufficient condition for the generating function g being positive for a given set of parameters λ, σ_J and r is that the function $\texttt{estimate}(k,h)$ is positive. □

So if the inequality (6.7) is satisfied the Toeplitz matrices T_n have to be positive definite for all n. Figure 6.2 depicts the "critical curve", that means the curve in

the k-h plane where the function `estimate`(k, h) as defined in the proof above is equal to zero. The right half space contains combinations of k and h for which T_n is guaranteed to be positive. However, note that this inequality (6.7) is only an estimate and therefore just a sufficient but not necessary condition.

To sum up, we have seen in this section that our coefficient matrix is guaranteed to be positive definite under the condition (6.7) and also that we know both the interval where the eigenvalues are to be found as well as their distribution. With this knowledge at hand we can continue by looking for a way of how to improve the eigenvalue distribution which will be subject of the following section.

6.2 Circulant preconditioning for Toeplitz systems

Improving the eigenvalue distribution of our Toeplitz system $T_n x = b$ can be done efficiently by preconditioning, that means we are searching for a preconditioning matrix M that we can easily invert and such that $M^{-1}T_n$ has a clustered spectrum. A good *preconditioner* should satisfy the following criteria:

(i) Any system $My = c$ should be easy to solve.
(ii) The product of the preconditioned matrix times a vector should be fast.
(iii) M should be a good approximation of T_n in some sense.
(iv) M should be symmetric positive definite (s.p.d.) since we intend to apply CG algorithm to the preconditioned system.
(v) The cost of constructing M should be inexpensive.

There are different ways of how to implement the preconditioning, namely we distinguish between

- *Left-preconditioning*: $M^{-1}T_n x = M^{-1}b$. Symmetry of M and T_n is not sufficient to guarantee symmetry of $M^{-1}T_n$, but if M is s.p.d. then $M^{-1}T_n$ is self-adjoint with regard to the M-inner product $\langle \cdot, \cdot \rangle_M$. So reformulation of the CG method with respect to this new inner product enables us to use this type of preconditioning. For an implementation see for example *Saad* [1996]. Algorithms where the stopping criterion is based on the residual are then minimizing the preconditioned residual $z = M^{-1}r$ instead of the original residual r. If this is not desired one has to adjust the algorithms accordingly.

- *Right-preconditioning*: $T_n M^{-1} y = b$, $y = Mx$. This is similar to the left-preconditioning in the sense that $T_n M^{-1}$ may not be symmetric any more, but using the $\langle \cdot, \cdot \rangle_{M^{-1}}$-inner product enables us to use the CG method again. Note that here the error is being changed so that algorithms with stopping criterion based on the error might need to be adjusted.
- *Split-preconditioning*: $M^{-1/2} T_n M^{-1/2} y = M^{-1/2} b$, where $y = M^{1/2} x$. Here $M^{-1/2} T_n M^{-1/2}$ is s.p.d. if M and T_n are. Obviously, split-preconditioning is a mixture of left- and right-preconditioning.

These different implementations will result in the same eigenvalue distributions of the preconditioned matrices, but in general the eigenvectors can be affected and this might have an influence on the convergence behavior since the start residual has different components in the eigenvector directions. However, for a symmetric positive definite matrix M exists the decomposition $M = LL^T$ and for this case it can be easily shown that the iterates of all three implementations are the same, see for example *Saad* [1996].
If we would like to obtain a method that can compete with the tridiagonal splitting approach of *Almendral and Oosterlee* [2005] we need a fast matrix-vector product. The requirement M being circulant would enable us to speed up the matrix-vector product by using FFT throughout the computations and also M is easily invertible. These were just the points (i) and (ii) in the list above that are therefore satisfied for all circulant-type preconditioners which we are going to introduce in the subsequent. Now, the question at hand now is in how far they match our other criteria for a good preconditioner.

6.2.1 Strang's preconditioner

Gilbert Strang was the first to propose a circulant preconditioner in *Strang* [1986]. His idea was to copy the central diagonals of the Toeplitz matrix T_n and to wrap them around to obtain a circulant matrix. The diagonals of the Strang preconditioner

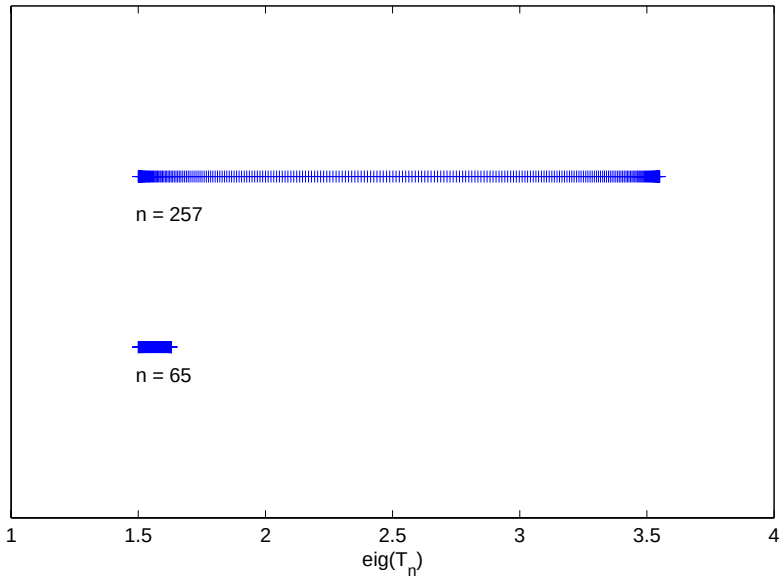

Figure 6.3. Eigenvalue distribution of the unprecond. coefficient matrix T_n for $n = 65$ and $n = 257$. Parameters: $r = 0$, $T = 1$, $\hat{x} = 4$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$, $k = T/40$ and $K = 1$

$S_n = (s_{k-l})_{0 \le k,l < n}$ are defined by:

$$s_j = \begin{cases} t_j, & 0 \le j \le floor(n/2), \\ t_{j-n}, & floor(n/2) < j < n, \\ s_{n+j}, & 0 < -j < n. \end{cases} \tag{6.10}$$

From a theoretical point of view, this preconditioner can be expected to work well because it is the optimal circulant approximation of T_n in the following sense :

Theorem 6.7 (*Chan* [1989a]). *Let T_n be an Hermitian Toeplitz matrix. The circulant matrix S_n whose entries are given by (6.10) minimizes $\|C_n - T_n\|_1 = \|C_n - T_n\|_\infty$ over all possible Hermitian circulant matrices C_n.*

So we got the good approximation property and symmetry for symmetric T_n. What about positive definiteness? Is this going to be inherited from T_n?

Theorem 6.8 (*Chan* [1989a]). *Let g be a positive function in the Wiener class, that means its Fourier coefficients are absolutely summable, see (6.6). Then for large n*

the circulant matrices S_n and S_n^{-1} are uniformly bounded in the l_2-norm. In fact, for large n, the spectrum $\sigma(S_n)$ of S_n satisfies

$$\sigma(S_n) \subseteq [g_{min}, g_{max}].$$

Theorem 6.8 guarantees us in particular that if the generating function g associated with T_∞ for which T_n is the $n \times n$ principal submatrix is positive and its Fourier coefficients are absolutely summable, then Strang's preconditioner is also positive definite. Like shown in the previous section, in our case g defined in (6.5) satisfies both conditions under appropriate conditions on the parameters and thus CG is applicable to the Strang preconditioned system.
Since our goal is to accelerate the CG method, under reference to Theorem 2.2 and Theorem 2.3 clustering the eigenvalues of T_n should provide an improvement in the convergence behavior. In this view the following result is interesting:

Theorem 6.9 (*Chan* [1989a])**.** *Let g be a real-valued function in Wiener class. Let $\{T_n\}$ be the sequence of Toeplitz matrices generated by g. Then the spectra of $S_n - T_n$ are clustered around zero for large n.*

Also can be shown that for these Toeplitz matrices PCG converges superlinearly. We remark that in several upcoming proofs the notation $\delta(\cdot)$ is used to denote the diagonal matrix with the same diagonal as the matrix in the argument.

Theorem 6.10 (*Chan* [1989a])**.** *Let g be a positive function in the Wiener class. Let $\{T_n\}$ be the sequence of Toeplitz matrices generated by g. Then for any given $\epsilon > 0$, there exists a constant $c(\epsilon) > 0$ such that the error vector e_k of the preconditioned conjugate gradient method at the kth iteration satisfies*

$$\frac{\|e_k\|}{\|e_0\|} \leq c(\epsilon)\epsilon^k,$$

where $\|v\|^2 = v^* S_n^{-1/2} T_n S_n^{-1/2} v$.

It follows in particular that the number of iterations required for convergence is independent of the size of the matrix T_n when n is large.

To sum up, checking the list of criteria reveals that S_n seems to be indeed an adequate choice. Theorem 6.7 shows that S_n is an optimal approximation of T_n and also documents the symmetry for real T_n, it will be positive definite under the assumptions of Theorem 6.8 and its construction is simple. In fact, the construction comes for free since no operations have to be performed. The drawbacks of Strang preconditioner are that the positive definiteness cannot be guaranteed for general generating functions, even if T_n is positive definite. Also, Strang's preconditioner is only applicable to Toeplitz matrices. However, in our case we happen to have a Toeplitz system, and the generating function g defined in (6.5) meets the assumptions of Theorem 6.8 and S_n is therefore positive definite for sufficiently large n.

6.2.2 T. Chan's optimal preconditioner

Tony Chan proposed in *Chan* [1988] a circulant preconditioner $c(T_n)$ that is optimal with respect to the Frobenius norm, that means it is defined as the minimizer of $\|C_n - T_n\|_F$ over the space of all $n \times n$ circulant matrices C_n. It is called the *optimal circulant preconditioner* and has the nice feature that $c(T_n)$ is s.p.d. whenever T_n is. It can be found in the following ways:

Theorem 6.11 (*Chan et al.* [1991a])**.** *Let $T_n \in \mathbb{C}^{n\times n}$ and $c(T_n)$ be the minimizer of $\|C_n - T_n\|_F$ over all circulant $n \times n$ matrices C_n. Then $c(T_n)$ is uniquely determined by T_n. Moreover,*

(i) $c(T_n)$ is given by

$$c(T_n) = \sum_{j=0}^{n-1} \left(\frac{1}{n} \sum_{p-q\equiv j \bmod n} t_{pq} \right) \Pi^j,$$

where Π is the $n \times n$ circulant "push" matrix defined in (2.6). For convenience

we restate it here:

$$\Pi = \begin{bmatrix} 0 & & & & 1 \\ 1 & 0 & & & \\ & 1 & \ddots & & \\ & & \ddots & \ddots & \\ 0 & & & 1 & 0 \end{bmatrix}.$$

(ii) $c(T_n)$ *is also given by*

$$c(T_n) = F_n^* \delta(F_n T_n F_n^*) F_n, \tag{6.11}$$

where F_n *is the Fourier matrix defined in (2.7).*

Proof. (*Chan et al. [1991a]*). A proof for (i) can be found in *Tyrtyshnikov* [1992]. For (ii), we recall that due to (2.8) we have $C_n = F_n^* \Lambda_n F_n$ with F_n being the unitary Fourier matrix. Since the Frobenius norm is unitary-invariant, we have

$$\|C_n - T_n\|_F = \|F_n^* \Lambda_n F_n - T_n\|_F = \|\Lambda_n - F_n T_n F_n^*\|_F.$$

Thus the problem of minimizing $\|C_n - T_n\|_F$ over all circulant matrices is equivalent to the problem of minimizing $\|\Lambda_n - F_n T_n F_n^*\|_F$ over all diagonal matrices. Since Λ_n can only affect the diagonal entries of $F_n T_n F_n^*$, we see that the solution for the latter problem is $\Lambda_n = \delta(F_n T_n F_n^*)$. Hence $F_n^* \delta(F_n T_n F_n^*) F$ is the minimizer of $\|C_n - T_n\|_F$. It is clear from the argument above that Λ_n and hence $c(T_n)$ are uniquely determined by T_n. □

Note that $c(T_n)$ can be defined for any square matrix, unlike the Strang preconditioner. The eigenvalues are the entries of the diagonal matrix $\delta(F_n T_n F_n^*)$. Although we shall see soon that $c(T_n)$ inherits symmetry and positive definiteness from T_n, the nonsingularity is not since nonsingularity of T_n cannot guarantee $\delta(F_n T_n F_n^*)$ to be nonsingular. For a Toeplitz matrix T_n the diagonals of $c(T_n)$ are given by

$$c_j = \begin{cases} \frac{(n-j)t_j + j t_{j-n}}{n}, & 0 \le j < n, \\ c_{n+j}, & -n < j < 0. \end{cases} \tag{6.12}$$

Therefore the preconditioner can in our case be constructed in only $O(n)$ operations. Also for Hermitian matrix T_n we have by (6.11):

$$c(T_n^*) = F_n^* \delta(F_n T_n^* F_n^*) F_n = F_n^* \delta(F_n T_n F_n^*)^* F_n = c(T_n)^*,$$

and hence $c(T_n)$ is Hermitian whenever T_n is. We have now all criteria for a good preconditioner satisfied except for the positive definiteness. This follows from

Theorem 6.12 (*Chan et al.* [1991a])**.** *If T_n is Hermitian, then $c(T_n)$ is Hermitian. Moreover, we have*

$$\lambda_{min}(T_n) \leq \lambda_{min}(c(T_n)) \leq \lambda_{max}(c(T_n)) \leq \lambda_{max}(T_n),$$

where $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ denote the largest and the smallest eigenvalues respectively. In particular, if T_n is positive definite then $c(T_n)$ is also positive definite.

Proof. (*Chan et al. [1991a]*). The inheritance of the Hermitian-property was already shown above. As for the eigenvalues of $c(T_n)$, we know that they are given by $\delta(F_n T_n F_n^*)$. Denote the values on the diagonal $\delta(F_n T_n F_n^*)$ by $\lambda_0, \ldots, \lambda_{n-1}$ with $\lambda_j := \lambda_{min}(c(T_n))$ and $\lambda_k := \lambda_{max}(c(T_n))$. Since T_n is Hermitian, we can use the Rayleigh quotient to characterize the eigenvalues:

$$\lambda_{max}(c(T_n)) = \lambda_k = \frac{e_k^* F_n T_n F_n^* e_k}{e_k^* e_k} \leq \max_{x \neq 0} \frac{x^* F_n T_n F_n^* x}{x^* x} = \max_{y \neq 0} \frac{y^* T_n y}{y^* y} = \lambda_{max}(T_n).$$

Likewise one obtains

$$\lambda_{min}(T_n) = \min_{y \neq 0} \frac{y^* T_n y}{y^* y} = \min_{x \neq 0} \frac{x^* F_n T_n F_n^* x}{x^* x} \leq \frac{e_j^* F_n T_n F_n^* e_j}{e_j^* e_j} = \lambda_j = \lambda_{min}(c(T_n)).$$

Hence the inequality is proven and with it also that $c(T_n)$ must be positive definite whenever T_n is. □

In order to prove that the spectrum of $c(T_n)^{-1} T_n$ is also clustered around one, we need the following lemma, which also provides some insight in the relation between Strang's preconditioner S_n and $c(T_n)$.

Lemma 6.13 (*Chan* [1989b]). *Let the generating function g be a positive function in the Wiener class, then the spectral radius of $S_n - c(T_n)$ has limit equal to zero, that means*

$$\lim_{n\to\infty} \rho(S_n - c(T_n)) = 0.$$

So the spectra of $c(T_n)$ and S_n are asymptotically the same and thus for large n the convergence behavior can be expected rather similar for both preconditioners. Next we got the following statement:

Lemma 6.14 (*Chan and Strang* [1987]). *Assume g is a positive function in the Wiener class. Then for all $\epsilon > 0$ there exist $M(\epsilon)$, $N(\epsilon) > 0$ such that for $n > M(\epsilon)$ at most $N(\epsilon)$ eigenvalues of $T_n - S_n$ have absolute value exceeding ϵ.*

Using the relation

$$c(T_n)^{-1}T_n = I_n + c(T_n)^{-1}(T_n - S_n) + c(T_n)^{-1}(S_n - c(T_n)),$$

in combination with the two previous lemmata leads to the conclusion that $c(T_n)^{-1}T_n - I_n$ must be clustered around zero, because $T_n - S_n$ has eigenvalues clustered around zero and $S_n - c(T_n)$ likewise. Thus

Theorem 6.15 (*Chan* [1989b]). *Let g be a positive function in the Wiener class, then for all $\epsilon > 0$ there exist $N(\epsilon), M(\epsilon) > 0$ such that for all $n > N(\epsilon)$ at most $M(\epsilon)$ eigenvalues of $c(T_n)^{-1}T_n - I_n$ have absolute value larger than ϵ.*

It has been shown in *Chan and Yeung* [1992] that a similar result holds for the broader class of functions which are continuous and 2π-periodic. The Wiener class of functions is actually a proper subset of this class. In our case the generating function is already in Wiener class so we do not follow the ideas leading to this extension.
The convergence rate of the preconditioned CG method depends on the condition number of the coefficient matrix, that means of $\kappa(M^{-1/2}T_nM^{-1/2})$ if we use split preconditioning with a preconditioner M. Therefore it makes sense to consider the preconditioner M which minimizes this condition number. M is called the *best conditioned circulant preconditioner*. Interestingly enough, it turns out that if $F_nT_nF_n^*$ has

the so-called Property A, then this M is exactly T. Chan's preconditioner $c(T_n)$. As for the definition of Property A, a matrix X is said to have *Property A* if there exists a permutation matrix P such that $PXP^T = \left(\begin{smallmatrix} D_1 & H \\ K & D_2 \end{smallmatrix}\right)$, where D_1 and D_2 are square diagonal matrices and H and K are arbitrary matrices. This can equivalently be seen as the condition that the graph of X is bipartite, that means there exist two sets of vertices (representing the unknowns) such that no edge connects two vertices in the same set. Loops are being ignored; in the matrix representation they correspond to the diagonals of D_1 and D_2.

Lemma 6.16 (*Chan and Wong* [1995])**.** *Let T_n be an $n \times n$ Hermitian positive definite matrix. If the matrix $F_n T_n F_n^*$ has Property A, then $c(T_n)$ minimizes $\kappa(C^{-1/2} T_n C^{-1/2})$ over all Hermitian positive definite circulant matrices C.*

We conclude that all desired features are available and that $c(T_n)$ is optimal in two senses under appropriate assumptions, apart from inheriting symmetry and positive definiteness from T_n.

6.2.3 Tyrtyshnikov's super-optimal preconditioner

In the previous section we examined a circulant preconditioner that was optimal in the sense that $c(T_n)$ minimized $\|C_n - T_n\|_F$ over all Hermitian circulant matrices C_n. Another preconditioner can be found by looking for the minimizer of $\|I_n - C_n^{-1} T_n\|_F$ over all nonsingular circulant matrices C_n. *Tismenetsky* [1991] and *Tyrtyshnikov* [1992] proposed this way independently and *Tyrtyshnikov* [1992] called it the *super-optimal circulant preconditioner*. First, let us take a look on how it is explicitly defined:

Theorem 6.17 (*Chan et al.* [1991a],*Chan et al.* [1991b])**.** *Let $T_n \in \mathbb{C}^{n \times n}$ be positive definite. Let $\widehat{C}_n$ be the super-optimal circulant preconditioner for T_n, defined by*

$$\|I - \widehat{C}_n T_n\|_F = \min \|I - C_n^{-1} T_n\|_F,$$

where the minimum is taken over all $n \times n$ nonsingular circulant matrices C_n. Then

$$\widehat{C}_n = c(T_n^*)(c(T_n T_n^*))^{-1}, \tag{6.13}$$

where $c(A)$ denotes T. Chan's preconditioner constructed from A.

Proof. (*Chan et al. [1991a],Chan et al. [1991b]*). $\delta(A)$ denotes again the diagonal matrix with the same diagonal as A.

Instead of minimizing $\|I_n - C_n^{-1}T_n\|_F$, we consider the problem of minimizing $\|I_n - \widehat{C}_n T_n\|_F$ over all nonsingular circulant matrices $\widehat{C}_n$. Letting $\widehat{C}_n = F_n^* \Lambda_n F_n$, we have by using the facts that the Frobenius norm is invariant for unitary matrices and that $\|A\|_F^2 = \text{trace}(AA^*)$:

$$\begin{aligned}
&\|I_n - \widehat{C}_n T_n\|_F^2 \\
=&\|I_n - F_n^* \Lambda_n F_n T_n\|_F^2 = \|I_n - \Lambda_n F_n T_n F_n^*\|_F^2 \\
=&\text{trace}(I_n - \Lambda_n F_n T_n F_n^* - F_n T_n^* F_n^* \Lambda_n^* + \Lambda_n F_n T_n T_n^* F_n^* \Lambda_n^*) \\
=&\text{trace}(I_n - \Lambda_n \delta(F_n T_n F_n^*) - \delta(F_n T_n^* F_n^*)\Lambda_n^* + \Lambda_n \delta(F_n T_n T_n^* F_n^*)\Lambda_n^*).
\end{aligned}$$

Let $\Lambda_n =: \text{diag}(\lambda_0, \ldots, \lambda_{n-1})$, $\delta(F_n T_n F_n^*) =: \text{diag}(u_0, \ldots, u_{n-1})$ and $\delta(F_n T_n T_n^* F_n^*) =: \text{diag}(w_0, \ldots, w_{n-1})$. We have

$$\begin{aligned}
&\min \|I_n - \widehat{C}_n T_n\|_F^2 \\
=&\min\{\text{trace}[I_n - \Lambda_n \delta(F_n T_n F_n^*) - \delta(F_n T_n^* F_n^*)\Lambda_n^* + \Lambda_n \delta(F_n T_n T_n^* F_n^*)\Lambda_n^*]\} \\
=&\min_{\{\lambda_0,\ldots,\lambda_{n-1}\}} \sum_{k=0}^{n-1} (1 - \lambda_k u_k - \bar{u}_k \bar{\lambda}_k + \lambda_k w_k \bar{\lambda}_k).
\end{aligned}$$

Since $c(T_n T_n^*) - c(T_n)c(T_n^*)$ is a positive semidefinite matrix (see Lemma 3 in *Chan et al.* [1991a] and by (6.11)), $w_k \geq u_k \bar{u}_k$ for all $k = 0, \ldots, n-1$. Hence for all complex scalars λ_k, for $k = 0, \ldots, n-1$, the terms $1 - \lambda_k u_k - \bar{u}_k \bar{\lambda}_k + \lambda_k w_k \bar{\lambda}_k$ are nonnegative. Differentiating them with respect to the real and imaginary parts of λ_k and setting the derivatives to zero, we get

$$\lambda_k = \frac{\bar{u}_k}{w_k}, \quad k = 0, \ldots, n-1.$$

Since T_n and $c(T_n)$ are nonsingular both w_k and u_k are nonzero. Hence λ_k are also nonzero. Thus the minimizer of $\|I_n - \widehat{C}_n T_n\|_F$ is nonsingular and is given by

$$\begin{aligned}\widehat{C}_n &= F_n^* \Lambda_n F_n = F n^* \delta(F_n T_n^* F_n^*)[\delta(F_n T_n T_n^* F_n^*)]^{-1} F_n \\ &= (F_n^* \delta(F_n T_n^* F_n^*) F_n)(F_n^* \delta(F_n T_n T_n^* F_n^*) F_n)^{-1} = c(T_n^*) c(T_n T_n^*)^{-1}. \end{aligned} \tag{6.14}$$

□

Tyrtyshnikov also showed that it inherits both symmetry and positive definiteness from T_n, just like the one from T. Chan. In this regard these two preconditioners are better than Strang's preconditioner.

Theorem 6.18 (*Tyrtyshnikov* [1992])**.** *Let $T_n \in \mathbb{R}^{n\times n}$ nonsingular. Then the superoptimal preconditioner $\widehat{C}_n$ is uniquely defined. If T_n is positive definite, then $\widehat{C}_n$ is nonsingular and, moreover, also positive definite. If $T_n = T_n^T$, then $\widehat{C}_n = \widehat{C}_n^T$.*

Like the previous two preconditioners also $\widehat{C}_n$ will cause the eigenvalues of the preconditioned matrix $\widehat{C}_n T_n$ to be clustered around one.

Theorem 6.19 (*Chan et al.* [1991b])**.** *Let the generating function g of T_n be a positive function in the Wiener class, then the spectrum of $\hat{C}T_n - I_n$ is clustered around zero. More precisely, for all $\epsilon > 0$, there exist $N(\epsilon), M(\epsilon) > 0$, such that, for all $n > N(\epsilon)$, at most $M(\epsilon)$ eigenvalues of $\hat{C}T_n - I_n$ have absolute value larger than ϵ.*

Chan et al. [1991a] provide also an efficient way of how to construct $\widehat{C}_n$ which was used in our numerical experiments. The key to their algorithm is the decomposition (2.12) of a Toeplitz matrix into the sum of a circulant and a skew-circulant matrix. The goal is to compute the first column of $\widehat{C}_n$, which is sufficient since $\widehat{C}_n$ is circulant. Using the decomposition (6.14) we obtain

$$\widehat{C}_n e_1 = F_n^* \delta(F_n T_n^* F_n^*)[\delta(F_n T_n T_n^* F_n^*)]^{-1} \vec{e} \frac{1}{\sqrt{n}}, \tag{6.15}$$

where $\vec{e} := [1, 1, \ldots, 1]^T$. Therefore we need to compute the diagonal matrices $\delta(F_n T_n^* F_n^*)$ and $\delta(F_n T_n T_n^* F_n^*)$, the latter one by using (2.12) to get $T_n = U_n + V_n =$

$F_n^*\Lambda_U F_n + V_n$. Hence

$$F_n T_n F_n^* = \Lambda_U + F_n V_n F_n^*, \tag{6.16}$$

and we obtain:

$$\begin{aligned}\delta(F_n T_n T_n^* F_n^*) &= \delta((F_n T_n F_n^*)(F_n T_n F_n^*)^*) \\ &= \Lambda_U \Lambda_U^* + \delta(F_n V_n F_n^*)\Lambda_U^* + \Lambda_U \delta(F_n V_n^* F_n^*) + \delta(F_n V_n V_n^* F_n^*).\end{aligned}$$

These four terms are now computed individually; for all applications of FFT we keep in mind that for an implementation in MATLAB the scaling factors $\sqrt{n}$ and $\frac{1}{\sqrt{n}}$ will cancel out, see Remark 2.10.

(i) $\frac{1}{\sqrt{n}}\Lambda_U \vec{e} = F_n U_n e_1$ gives us Λ_U by one FFT of length n, then $\Lambda_U \Lambda_U^*$ is done elementwise in n multiplications.

(ii) For $\delta(F_n V_n F_n^*)\Lambda_U^*$ we have by making use of the relation $c(V_n) = F_n^* \delta(F_n V_n F_n^*) F_n$ from (6.11):

$$\delta(F_n V_n F_n^*)\vec{e} = \delta(F_n V_n F_n^*) F_n e_1 \sqrt{n} = F_n c(V_n) e_1 \sqrt{n},$$

so we only need to compute the first column of T. Chan's preconditioner for the skew-circulant V_n ($3n$ multiplications and n additions), apply one FFT of length n to it and multiply it by Λ_U^* which is again n multiplications.

(iii) $\Lambda_U \delta(F_n V_n^* F_n^*) = (\delta(F_n V_n F_n^*)\Lambda_U^*)^*$, so by using the result of (ii) this term comes for free.

(iv) For the last remaining term $\delta(F_n V_n V_n^* F_n^*)$ we get similarly to (ii):

$$\delta(F_n V_n V_n^* F_n^*)\vec{e} = \delta(F_n V_n V_n^* F_n^*) F_n e_1 \sqrt{n} = F_n c(V_n V_n^*) e_1 \sqrt{n}.$$

Next we need $V_n V_n^*$. Note that V_n is skew-circulant, so by (2.10) holds $V_n = \Omega_n^* F_n^* \Lambda_V F_n \Omega_n$ and Λ_V is obtained by the relation $\frac{1}{\sqrt{n}}\Lambda_V \vec{e} = F_n \Omega_n V_n e_1$ in n multiplications for $\Omega_n V_n e_1$ and one FFT. Now the first column of $V_n V_n^*$ is easily computable by

$$V_n V_n^* e_1 = \Omega_n^* F_n^* \Lambda_V \Lambda_V^* F_n \Omega_n e_1 = \frac{1}{\sqrt{n}} \Omega_n^* F_n^* \Lambda_V \Lambda_V^* \vec{e},$$

with n multiplications for $\Lambda_V \Lambda_V^* \vec{e}$, one FFT and again n multiplications for the application of Ω_n^*. It is sufficient only to compute the first column because due to Theorem 2.14 the matrix $V_n V_n^*$ is skew-circulant. The first column of T. Chan's preconditioner $c(V_n V_n^*)e_1$ can now be build using (6.12) by n additions and $3n$ multiplications.

Finally, applying one FFT to $c(V_nV_n^*)e_1$ gives us $\delta(F_nV_nV_n^*F_n^*)\vec{e}$. By summing those four terms together we obtain $\delta(F_nA_nA_n^*F_n^*)$ in $3n$ additions, so in total we needed $5n$ additions, $11n$ multiplications and 5 FFTs.
To obtain $\delta(F_nA_n^*F_n^*)$ we use (6.16) to get

$$\delta(F_nA_n^*F_n^*) = (\delta(F_nA_nF_n^*))^* = \Lambda_U^* + (\delta(F_nV_nF_n^*))^*,$$

which means we only need n additions since everything is already computed.
Finally, (6.15) enables us now to obtain $\hat{C}e_1$ after another n multiplications to get the product $\delta(F_nA_n^*F_n^*)[\delta(F_nA_nA_n^*F_n^*)]^{-1}$, that means we perform the elementwise division

$$\delta(F_nA_n^*F_n^*)\vec{e} \,./\, \delta(F_nA_nA_n^*F_n^*)\vec{e}.$$

Applying one last FFT gives us the desired result. Thus $\hat{C}_ne_1$ has been computed in $12n$ multiplications, $6n$ additions and 6 FFTs. The work for computing the decomposition $A_n = U_n + V_n$ is not included here since we have to do this for all preconditioners anyway in order to avoid extra work in the CG method due to the preconditioning. We explain this in detail in the implementation section which concludes this chapter. The original way proposed by *Tyrtyshnikov* [1992] required 9 FFTs and $O(n)$ operations, so this one is obviously better.

Overall we conclude that all criteria are again satisfied, although in this case the cost to compute the preconditioners is much higher than for S_n and $c(T_n)$. However, in our case the coefficient matrix stays the same in all time steps and therefore we can neglect this problem since we only have to compute the preconditioner once. Actually, just computing and storing its eigenvalues at the beginning is sufficient.

6.2.4 R. Chan's preconditioner

We will briefly mention one more circulant preconditioner proposed in *Chan* [1989a]. His preconditioner is defined as the circulant matrix R_n whose diagonals are defined by

$$r_k = \begin{cases} t_{k-n} + t_k, & 0 \le k < n, \\ r_{-k}, & 0 < -k < n, \end{cases}$$

where t_{-n} is set equal to zero and T_n assumed to be real. Note that R_n is symmetric by construction. He also showed that the eigenvalue distribution of R_n is asymptotically

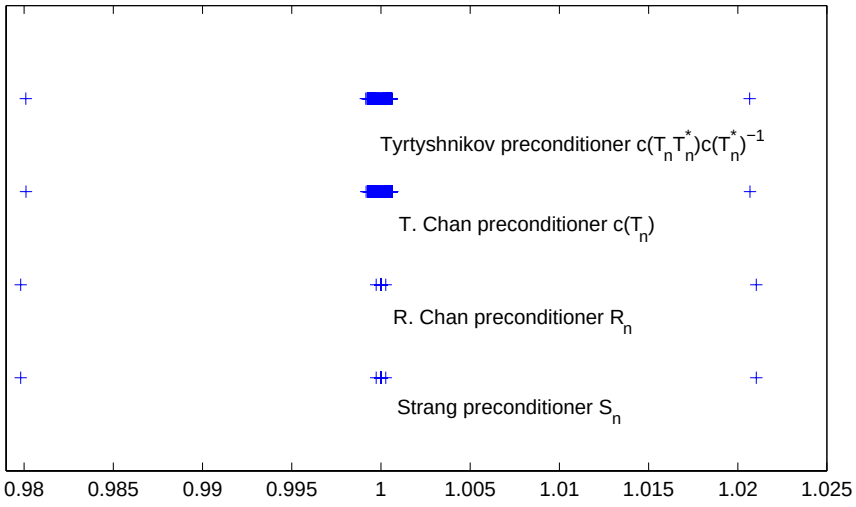

Figure 6.4. Eigenvalue distribution of the precond. coefficient matrix for several preconditioners. Parameters $r = 0$, $T = 1$, $\hat{x} = 4$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$, $n = 65$, $m = 40$ and $K = 1$.

the same as for Strang's preconditioner S_n for large n. It is possible to prove that $\lim_{n\to\infty} \|S_n - R_n\|_2 = 0$. But we do not have optimality of R_n in some norm, so that probably S_n, $c(T_n)$ and $\widehat{C}_n^{-1}$ should work better, although they should behave nearly the same for large n.

6.2.5 Comparison of the preconditioners

Strela and Tyrtyshnikov [1996] conducted numerical experiments with the preconditioner of Strang S_n and of T. Chan $c(T_n)$. They came to the conclusion that for positive generating functions no essential difference could be observed. They did not even consider Tyrtyshnikov's super-optimal preconditioner $\widehat{C}_n^{-1}$ since earlier investigations in *Strela* [1993] had shown that $\widehat{C}_n^{-1}$ did not perform as well as hoped for. *Chan et al.* [1991b] concluded from their experiments that all preconditioned systems for S_n, $c(T_n)$ and $\widehat{C}_n^{-1}$ are converging at the same rate for large n. In *Chan and Yeung* [1992] the preconditioner $c(T_n)$ was described as the best choice for Toeplitz systems whose generating function is in the Wiener class since it combines low cost of construction with guaranteed positive definiteness as long as T_n is positive definite. R. Chan's and Strang's preconditioners can be indefinite for general positive definite

Table 6.1. Number of CG iterations for different preconditioners. Parameters: $k = T/40$, $T = 1$, truncation point $\hat{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$ and strike $K = 1$.

n	Unprecond.	Strang S_n	T. Chan $c(T_n)$	Tyrtyshnikov Y_n	R. Chan R_n
17	3	3	3	3	3
33	3	3	3	3	3
65	4	3	3	3	3
129	6	4	4	4	4
257	9	4	4	4	4
513	18	4	5	5	4
1025	36	4	5	6	4
2049	73	4	5	8	4

Toeplitz matrices T_n. By *Chan et al.* [1991b] and *Chan* [1989a] all here considered preconditioned systems have asymptotically the same eigenvalues.

To test the performance of the preconditioners for our problem we applied them to our coefficient matrix using a split implementation and measured how many iterations the conjugate gradient method takes on the preconditioned system. The outcome is presented in Table 6.1. Note that without preconditioning the number of iterations increases steadily, whereas for the preconditioned systems it stays almost the same. The only exception is the super-optimal preconditioner which does not perform well for large n. It may be surprising that R_n is equally successful as S_n; this is because for our specific system R_n turns out to be nearly equal to S_n. However, recalling that for S_n the construction comes for free and noticing that S_n has the best performance together with R_n, we conclude that S_n will be the best choice for our system. For this reason we will confine our numerical tests to Strang preconditioned systems.

6.3 Implementation

Now that we have chosen a preconditioner we can proceed with the practical implementation of the Strang preconditioned conjugate gradient method (PCG). In Chapter 2 we stated with Algorithm 2.4 already a version of PCG for a general preconditioner. Note that the main computational work is needed for the matrix-vector product with the coefficient matrix and for the solution of a system $Mz_{j+1} = r_{j+1}$. Thus it seems as if this approach will need more operations than the splitting technique used by

Almendral and Oosterlee [2005] that we examined in Chapter 5. In spite of this negative first impression we recall that Strang's preconditioner is circulant and we will use this fact to show that PCG needs only about the same amount of operations per iteration as the splitting method!

Let us examine this closer: For the matrix-vector product we need two FFTs of length $2n$ by using the circulant embedding (2.11). Note that indeed two FFTs are sufficient as long as the coefficient matrix does not change over the time, because then we simply compute the eigenvalues of its circulant embedding once at the very beginning, thus saving one of the usual three necessary FFTs. Additionally we would need two FFTs of length n to solve the circulant system $Mz_{j+1} = r_{j+1}$. In fact, this overhead of the PCG versus the unpreconditioned CG where we only need two FFTs of length $2n$ can be reduced to zero by using an idea of *Freund and Huckle* [1993]. If we let PCG run in the Fourier domain, then we get the preconditioning virtually for free. The key ingredient to this implementation is the decomposition $T_n = U_n + V_n$ defined in (2.12). For left-preconditioned systems an implementation can be found in *Ng* [2004]. We will implement the same trick equivalently for a split preconditioned framework with Strang's preconditioner S_n to guarantee symmetry. Since $S_n^{-1} = S_n^{-1/2} S_n^{-1/2} = S_n^{-1/2} (S_n^{-1/2})^T$ this preconditioned system is

$$S_n^{-1/2} T_n S_n^{-1/2} y = S_n^{-1/2} b, \quad y = S_n^{1/2} x. \tag{6.17}$$

Let $S_n^{-1} = F_n^* \tilde{\Lambda} F_n$. Then $S_n^{-1/2} = F_n^* \Lambda_1 F_n$ with $\Lambda_1 = \tilde{\Lambda}^{1/2}$. The matrix T_n can be decomposed in a sum of a circulant U_n and a skew-circulant matrix V_n by (2.12) with $U_n = F_n^* \Lambda_2 F_n$ and $V_n = \Omega_n^* F_n^* \Lambda_3 F_n \Omega_n$, where Λ_1, Λ_2, Λ_3 and Ω_n are diagonal matrices. Plugging this into equation (6.17) yields

$$\begin{aligned} S_n^{-1/2} T_n S_n^{-1/2} S_n^{1/2} x = S_n^{-1/2} (U_n + V_n) S_n^{-1/2} S_n^{1/2} x &= S_n^{-1/2} b \\ F_n^* \Lambda_1 F_n (F_n^* \Lambda_2 F_n + \Omega_n^* F_n^* \Lambda_3 F_n \Omega_n)(F_n^* \Lambda_1 F_n)(F_n^* \Lambda_1^{-1} F_n) x &= F_n^* \Lambda_1 F_n b \\ \Lambda_1 (\Lambda_2 + F_n \Omega_n^* F_n^* \Lambda_3 F_n \Omega_n F_n^*) \Lambda_1 \underbrace{(\Lambda_1^{-1} F_n x)}_{=:\tilde{x}} &= \underbrace{\Lambda_1 F_n b}_{=:\tilde{b}} \end{aligned} \tag{6.18}$$

The cost for solving the preconditioned s.p.d. system (6.18) with CG is dominated by the matrix-vector product which can be done in essentially four FFTs of length n, so the cost is approximately the same like for the unpreconditioned system where we needed two FFTs of length $2n$, see Remark 2.5. Thus the overhead can indeed be approximately reduced to zero and we see that PCG solves the system at cost comparable to that of the splitting method. The cost per iteration is of order $O(n \log n)$.

Furthermore, by Theorem 6.10 PCG converges within $O(1)$ iterations, so that in each time step just $O(n \log n)$ operations are needed. As for the necessary storage, only some vectors with the diagonals of Λ_1, Λ_2, Λ_3, Ω_n and the right-hand side b are needed.

Remark 6.20. One might wonder whether direct superfast Toeplitz solvers could be preferable over PCG. These algorithms solve a Toeplitz system in $O(n \log^2 n)$ operations, even less than the older so-called fast Toeplitz solvers of *Schur* [1917], *Levinson* [1947], *Durbin* [1960] and *Trench* [1964] that require $O(n^2)$ operations. The superfast algorithms had been invented by several researchers, for instance *Bitmead and Anderson* [1980], *Brent et al.* [1980] and *Ammar and Gragg* [1987]. However, *Linzer* [1992] showed that PCG is more efficient vis-à-vis the superfast direct Toeplitz solvers if a certain decay rate, depending on the condition number, is exceeded. In our case we have exponentially decaying kernels and thus PCG can be expected to be better.

Chapter 7

Preconditioned CGNR using scheme $\mathcal{N}1$ for Kou's model

So far we discussed the application of the preconditioned conjugate gradient method (PCG) only for Merton's model, where thanks to the symmetry of the normal density the coefficient matrix resulting from the discretization scheme $\mathcal{N}1$ is symmetric. For Kou's model, however, this coefficient matrix inherits in general the asymmetry of the double exponential density. Since PCG turns out to be effective for for Merton's model we also would like to apply it here, but first we would need to transform the linear system to a symmetric one. A possibility of doing so is presented in this chapter, namely the normal equations approach.
First we recall some facts about these normal equations, then we show efficient ways to improve the condition of the hereby obtained system and conclude with some comments on implementation issues.

7.1 On normal equations

For Kou's model the real Toeplitz matrix T_n arising from the discretization scheme $\mathcal{N}1$ is in general asymmetric. So in order to apply PCG we need to transform the system $T_n x = b$ to a symmetric one. A well-known approach is to consider the system

$$T_n^* T_n x = T_n^* b. \tag{7.1}$$

System (7.1) is called the *normal equations* associated with the least squares problem

$$\min_x \|b - T_n x\|_2, \tag{7.2}$$

because $\tilde{x}$ is a minimizer of (7.2) if and only if $T_n^*(b - T_n\tilde{x}) = \mathbf{0}$. In other words, the latter condition expresses that the residual at a minimizer is orthogonal to the range of T_n. Note that the characterization of the normal equations by a least squares problem in particular has the consequence that there always exists a solution. Furthermore, if T_n is nonsingular and has full rank, then $T_n^* T_n$ is symmetric positive definite (s.p.d.) and the solution is unique. So in particular we can apply CG to the system (7.1), and we call this method CGNR for Conjugate Gradient Normal equations Residual. The R for residual is derived from the fact that the kth iterate x_k in CGNR minimizes the residual $\|b - T_n x\|_2$ over the Krylovspace $x_0 + \mathcal{K}(T_n^* T_n, r_0, k)$, see for example *Golub and Van Loan* [1996]. A typical version of CGNR for solving a system $A^T Ax = A^T b$ is the following:

Algorithm 7.1 (*Saad* [1996])**.** CGNR:

1. Compute $r_0 := b - Ax_0$, $z_0 := A^T r_0$, $p_0 := z_0$.
2. For $i = 0, \ldots,$ until convergence Do
3. $\quad w_i := Ap_i$
4. $\quad \alpha_i := \|z_i\|_2^2 / \|w_i\|_2^2$
5. $\quad x_{i+1} := x_i + \alpha_i p_i$
6. $\quad r_{i+1} := r_i - \alpha_i w_i$
7. $\quad z_{i+1} := A^T r_{i+1}$
8. $\quad \beta_i := \|z_{i+1}\|_2^2 / \|z_i\|_2^2$
9. $\quad p_{i+1} := z_{i+1} + \beta_i p_i$
10. End Do

In general CGNR suffers from the effect that the 2-norm condition number of $T_n^* T_n$ is the square of the one of T_n. But if $\mathrm{cond}_2(T_n)$ is already small, $\mathrm{cond}_2^2(T_n)$ is still

acceptable. For our problem T_n is already well-conditioned, and we can even improve the condition of $T_n^*T_n$ by appropriate circulant preconditioners. Again we confine ourselves to circulant preconditioning in order to profit of the FFT.

Remark 7.2. It is important to note that for a Toeplitz matrix T_n the product $T_n^*T_n$ is in general not Toeplitz any more. But this matrix does not need to be build explicitly, rather we just need an efficient way of computing the matrix-vector product $T_n^*T_np$. This task is accomplished by performing two Toeplitz matrix times vector products by using the circulant embedding (2.11), so essentially we need four FFTs of length $2n$.

7.2 Circulant preconditioning for normal equations of Toeplitz system

The purpose and criteria of good preconditioning have already been pointed out in the last chapter. Now we pursuit the question in how far it is possible to improve the condition of the matrix $T_n^*T_n$, preparing the way for a preconditioned version of CGNR.
We consider two circulant preconditioners: T. Chan's optimal preconditioner that we already encountered in the last chapter, and a so-called *displacement preconditioner* which is specifically designed for the normal equations of a Toeplitz system. Since both preconditioners are circulant the criteria are satisfied that $My = c$ is easy to solve, where M is the preconditioner, and that the matrix-vector product is fast. What remains is to ensure that they provide a good approximation of $T_n^*T_n$ in some sense, that they are symmetric positive definite and that their construction is inexpensive.

7.2.1 T. Chan's optimal preconditioner

Since T. Chan's preconditioner $c(\cdot)$ was already covered in the last chapter, we just underline that $c(T_n^*T_n)$ inherits symmetry and positive definiteness from $T_n^*T_n$. Note

that by Theorem 6.11 the preconditioner $c(\cdot)$ is defined for general square matrices, so the fact that $T_n^* T_n$ is not Toeplitz does not cause a problem. Only the construction will be more expensive than in the Toeplitz case because we cannot use its definition (6.12) for the Toeplitz case any more but need to apply the general definition of $c(\cdot)$ stated in Theorem 6.11.

7.2.2 Displacement preconditioner

Chan et al. [1994] have proposed a circulant preconditioner P_n for $T_n^* T_n$ which is essentially derived by making use of the fact that $T_n^* T_n$ has still a certain structure although it is in general not Toeplitz anymore. A feature of this preserved structure is the low so-called *displacement rank*, shown for example in *Kailath and Sayed* [1995]. We will define this rank in a moment, but first we need to define a few other technical terms concerning displacement. Denote by Z the square *lower shift matrix* with ones on the first lower subdiagonal and zeros everywhere else,

$$Z = \begin{pmatrix} 0 & 0 & \cdots & & 0 \\ 1 & 0 & & & \vdots \\ 0 & 1 & \ddots & & \\ \vdots & & \ddots & & \\ 0 & \cdots & & 1 & 0 \end{pmatrix}.$$

Using this lower shift matrix Z we define the displacement operator ∇ to be

$$\nabla T_n := T_n - Z T_n Z^*,$$

and ∇T_n is then called the *displacement of* T_n. The name derives from the fact that $Z T_n Z^*$ shifts T_n one step down along the main diagonal, or in other words every element is moved one index to the right and one down. The first row and first column of $Z T_n Z^*$ is the zero vector. Hence for a Toeplitz matrix T_n its displacement matrix ∇T_n is the matrix with the first column and first row equal to those of T_n and zeros elsewhere. Next we introduce a decomposition called the *displacement representation* of an arbitrary square matrix.

Lemma 7.3 (*Chan et al.* [1994])**.** *An arbitrary $n \times n$ matrix A can be written in the*

form

$$A = \sum_{i=1}^{\alpha} L(u_i)L^*(v_i),$$

where $\alpha = \operatorname{rank}(\nabla A)$, u_i *and* v_i *are vectors of length* n *and* $L(w)$ *denotes the* $n \times n$ *lower triangular Toeplitz matrix having as first column the vector* w.

The number α is called the *displacement rank* of A. *Chun et al.* [1987] showed that square Toeplitz matrices have small displacement rank, as well as those close to being Toeplitz like $T_n^* T_n$.

Example 7.4 (*Chan et al.* [1994])**.** For an Hermitian Toeplitz matrix H we have

$$H = L(u_1)L(u_1)^* - L(u_2)L(u_2)^*,$$

where

$$u_1 := [\frac{1}{2}(h_0 + 1), h_1, \ldots, h_{n-1}]^*,$$
$$u_2 := [\frac{1}{2}(h_0 - 1), h_1, \ldots, h_{n-1}]^*.$$

So it would make sense to define a preconditioner for H by replacing each of the lower triangular Toeplitz matrices by the corresponding optimal preconditioner of T. Chan. Thus the resulting circulant preconditioner M would be

$$M := c(L(u_1))c(L(u_1)^*) - c(L(u_2))c(L(u_2)^*).$$

Unfortunately, our coefficient matrix T_n is not symmetric. But there exists the following result for a decomposition of $T_n^* T_n$:

Lemma 7.5 (*Chun et al.* [1987])**.** *Let* T_n *be an* $n \times n$ *Toeplitz matrix. Then a displacement representation of* $T_n^* T_n$ *is*

$$T_n^* T_n = L(x_1)L(x_1)^* - L(x_2)L(x_2)^* + L(y_1)L(y_1)^* - L(y_2)L(y_2)^*,$$

where $x_1 = T_n^* T_n e_1 / \|T_n e_1\|_2$, $x_2 = ZZ^* x_1$, $y_1 = [0, t_{-1}, t_{-2}, \ldots, t_{1-n}]^*$ *and* $y_2 = [0, t_{n-1}, t_{n-2}, \ldots, t_1]^*$.

We are in a position to simplify this displacement representation. Note that by the transformation ZZ^*x_1 the first component of x_1 is replaced by zero, the other components remain the same. Hence $L(x_1)$ is the same matrix like $L(x_2)$ except for the fact that the main diagonal of $L(x_2)$ consists only of zeros. Thus $L(x_1) = L(x_2) + \tilde{t}_{11}/\|T_n e_1\|_2 I$, where $\tilde{t}_{11}$ denotes the element $(T_n^* T_n)_{11}$. Since $\tilde{t}_{11} = (T_n e_1)^*(T_n e_1) = \|T_n e_1\|_2^2$, we obtain $L(x_1) = L(x_2) + \|T_n e_1\|_2 I$. Plugging this into $L(x_1)L(x_1)^*$ yields

$$L(x_1)L(x_1)^* - L(x_2)L(x_2)^* = \|T_n e_1\|_2 L(x_2) + \|T_n e_1\|_2 L(x_2)^* + \|T_n e_1\|_2^2 I =: \tilde{H}.$$

The matrix $\tilde{H}$ is symmetric, Toeplitz and its first column is equal to $T_n^* T_n e_1$. So we end up with

$$T_n^* T_n = \tilde{H} + L(y_1)L(y_1)^* - L(y_2)L(y_2)^*.$$

Chan et al. [1994] showed that the last term $L(y_2)L(y_2)^*$ has little impact on the whole expression and that for an Hermitian Toeplitz matrix H the circulant preconditioner defined in Example 7.4 is equal to T. Chan's optimal preconditioner $c(H)$. Putting all together we define the *displacement preconditioner* P_n by

$$P_n := c(\tilde{H}) + c(L(y_1))c(L(y_1))^*.$$

The symmetry and positive definiteness of P_n follow from its construction. We finish this section with an result concerning the ability of P_n to approximate $T_n^* T_n$ sufficiently good. In *Chan et al.* [1994] was proven that $P_n^{-1} T_n^* T_n$ will have eigenvalues clustered around 1 if the generating function g of T_n is in the Wiener class. In the following Lemma we show that this is the case for T_n arising for Kou's model with the discretization scheme $\mathcal{N}1$. This means in the case $m \geq 2$ we consider the singly-infinite Toeplitz matrix T_∞ with diagonals t_κ defined by

$$\begin{aligned} t_0 &= a + (r+\lambda)k + \frac{3}{2} - bp\alpha_1, \\ t_1 &= -\frac{a}{2} - bp\alpha_1 e^{-\alpha_1}, \quad t_{-1} = -\frac{a}{2} - bq\alpha_2 e^{-\alpha_2}, \\ t_\kappa &= \begin{cases} -bp\alpha_1 e^{-\alpha_1 \kappa}, & \text{for } \kappa \geq 2, \\ -bq\alpha_2 e^{\alpha_2 \kappa}, & \text{for } \kappa \leq -2, \end{cases} \end{aligned} \tag{7.3}$$

where we set $a := k\sigma^2/h^2$, $b := kh\lambda$ and use the double exponential density function.

Lemma 7.6. *Let g be the generating function of the Toeplitz matrices with the diag-*

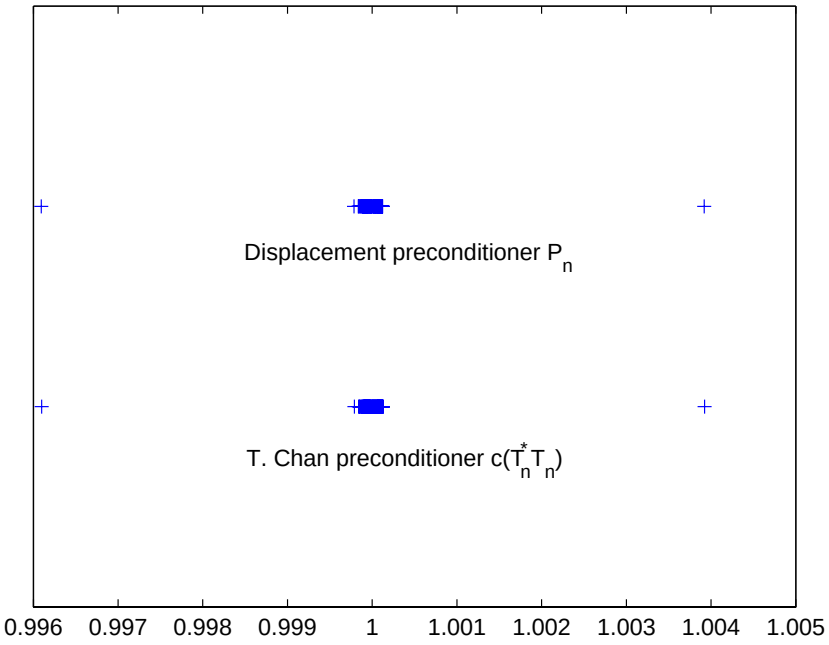

Figure 7.1. Eigenvalue distribution of $M^{-1}T_n^*T_n$ for different preconditioners. Parameters: $r = 0$, $T = 0.2$, $\hat{x} = 6$, $\sigma = 0.2$, $\lambda = 0.2$, $n = 65$, $k = T/40$, $K = 1$, $\alpha_1 = 3$ and $\alpha_2 = 2$.

onal elements defined in (7.3). Then g is in the Wiener class, that means

$$\sum_{j=-\infty}^{\infty} |t_j| < \infty. \tag{7.4}$$

Proof. In order to prove (7.4) it is sufficient to show that

$$\sum_{j=2}^{\infty} |t_j| = \sum_{j=2}^{\infty} bp\alpha_1 e^{-\alpha_1 j} < \infty,$$

and

$$\sum_{j=-\infty}^{-2} |t_j| = \sum_{j=2}^{\infty} bq\alpha_2 e^{-\alpha_2 j} < \infty.$$

An easy application of the ratio test yields directly that both inequalities are satisfied and thus the lemma follows. □

The case $m = 1$ can be shown similarly. Thus we can apply the following theorem concerning the clustering behavior of the displacement preconditioner P_n:

Table 7.1. Number of P-CGNR iterations for different preconditioners. Parameters: $r = 0$, $T = 0.2$, $\hat{x} = 6$, $\sigma = 0.2$, $\lambda = 0.2$, $K = 1$, $\alpha_1 = 3$, $\alpha_2 = 2$ and $p = 0.5$. Grid sizes: Time $m = 40$, Space n.

n	No precond.	T. Chan precond. $c(T_n^*T_n)$	Displ. precond. P_n
65	3	3	3
129	4	3	3
257	6	4	4
513	10	4	4
1025	24	5	5

Theorem 7.7 (*Chan et al.* [1994])**.** *Let the generating function g of the $n \times n$ Toeplitz matrix T_n be in the Wiener class with no zeros on $[-\pi, \pi]$. Then for all $\epsilon > 0$, there exist $N(\epsilon) > 0$ and $M(\epsilon) > 0$, such that for all $n > N(\epsilon)$ at most $M(\epsilon)$ eigenvalues of the matrix*

$$P_n^{-1}(T_n^*T_n) - I$$

have absolute values larger than ϵ.

As for the construction cost, P_n can be computed in $O(n \log n)$ operations.

7.2.3 Comparison of the preconditioners

Both preconditioners can be expected to work well since on the one hand $c(T_n^*T_n)$ is the optimal circulant approximation of $T_n^*T_n$ in the Frobenius norm, and on the other hand P_n was constructed also using T. Chan's preconditioner, so it should be a good approximation in this sense. Additionally, P_n exploits the special structure of $T_n^*T_n$ which could result in some improvement compared to $c(T_n^*T_n)$. The higher cost of P_n is unimportant since the coefficient matrix T_n stays the same over all time steps. Indeed, in Figure 7.1 we observe for both approaches similar eigenvalue distributions. Especially the desired clustering effect is attained. Similar to the last chapter we compare the performance of the preconditioners in terms of required CGNR iterations, see Table 7.1. We keep the time step constant and vary the amount of grid points in space. Both preconditioners work equally well; we will choose P_n for future computations in P-CGNR.

In the next section we explain a way of how to implement P-CGNR.

7.3 Implementation

Regarding practical implementation, we emphasize that the product $T_n^* T_n$ is never build explicitly. In Algorithm 7.8 we state a left-preconditioned version of CG on the normal equations of T_n. The passage from system $T_n x = b$ to $P_n^{-1} T_n^* T_n x = P_n^{-1} T_n^* b$ can be interpreted as two times left-preconditioning, first with "preconditioner" T_n^* and then with the displacement preconditioner P_n. In Section 6.2 we saw that left-preconditioning changes the residual of the system and algorithms with stopping criterion based on the residual might need to be adjusted. In the given version of P-CGNR in Algorithm 7.8 this adjustment has already been incorporated since it breaks the updating process down to the computation of the original residual r which, of course, can then be used by a stopping criterion.

Algorithm 7.8 (*Saad* [1996])**.** Left-Preconditioned CGNR:

1. Compute $r_0 := b - T_n x_0$, $\tilde{r}_0 := T_n^* r_0$, $z_0 := P_n^{-1} \tilde{r}_0$, $p_0 := z_0$.
2. For $i = 0, \ldots,$ until convergence Do
3. $\quad w_i := T_n p_i$
4. $\quad \alpha_i := \langle z_i, \tilde{r}_i \rangle / \|w_i\|_2^2$
5. $\quad x_{i+1} := x_i + \alpha_i p_i$
6. $\quad r_{i+1} := r_i - \alpha_i w_i$
7. $\quad \tilde{r}_{i+1} := T_n^* r_{i+1}$
8. $\quad z_{i+1} := P_n^{-1} \tilde{r}_{i+1}$
9. $\quad \beta_i := \langle z_{i+1}, \tilde{r}_{i+1} \rangle / \langle z_i, \tilde{r}_i \rangle$
10. $\quad p_{i+1} := z_{i+1} + \beta_i p_i$
11. End Do

The main computational work arises in lines 3, 7 and 8 for the matrix vector products. Since T_n is Toeplitz and constant over the time after the first time step, for both lines 3 and 7 we get along with each two FFTs of length $2n$ and one vector multiplication. Additionally, there is some work needed for the solution of the system in line 8. For Merton's model we saw in the last chapter that it is possible to reduce this extra work

for the preconditioning virtually to zero by taking advantage of the decomposition of a Toeplitz matrix T_n into the sum of a circulant and a skew-circulant matrix. However, this is not possible any more in our case since $T_n^* T_n$ loses its Toeplitz property. So we have to perform the matrix vector product in line 8 which is possible in two FFTs of length n and one vector multiplication since the preconditioner P_n is circulant and constant over the time after the first time step.
The dominating cost per P-CGNR iteration is therefore approximately ten FFTs. Due to the predicted eigenvalue clustering we expect the method to converge fast, so that the work per time step is $O(n \log n)$.
As for the required storage, we still only need to save some vectors of length n.

Overall we conclude that this approach should also work well, but is most likely to be noticeably slower than the splitting technique since the work per iterations is higher.

Chapter 8

Numerical results

This chapter addresses the question as to what extent one method might be preferable to the other, depending on the underlying discretization scheme. First we focus on Merton's model and discuss the outcome of several experiments and its implications before we examine Kou's model.
All CPU times in the tables of this chapter are based on an Intel Centrino M 1.60 GHz and computed using MATLAB.

8.1 Merton's model

In order to obtain information on the accuracy of the numerical solution w will need some kind of benchmark. There exists an analytical solution for a European call option in form of an infinite series found by *Merton* [1976]. We compute the numerical error by using a version of this infinite series given in *Wilmott* [1998]. For $\mu_J = 0$ the analytical solution is then

$$w(t,s) = \sum_{m=0}^{\infty} \frac{e^{-\lambda(1+\eta)\tau}(\lambda(1+\eta)\tau)^m}{m!} V_{BS}(\tau, s, K, r_m, \sigma_m),$$

where $\tau := T - t$ is the time to maturity, $\eta = e^{\sigma_J^2} - 1$ denotes the expected percentage change in the stock price originating from a jump, $\sigma_m^2 := \sigma^2 + m\sigma_J^2/\tau$ the volatility, $r_m := r - \lambda\eta + m\log(1+\eta)/\tau$ and V_{BS} the Black-Scholes price of a call (*Black and*

Scholes [1973]) computed by

$$V_{BS}(\tau, S, K, r, \sigma) = S\Phi(d_1) - Ke^{-r\tau}\Phi(d_2),$$

with

$$d_1 = \frac{\log(\frac{S}{K}) + (r + \frac{1}{2}\sigma^2)\tau}{\sigma\sqrt{\tau}}, \quad d_2 = d_1 - \sigma\sqrt{\tau}.$$

The cumulative normal distribution $\Phi(x)$ is defined in (3.2).
We conduct experiments for the following approaches:

(i) Tridiagonal splitting iterations for scheme $\mathcal{W}1$.
(ii) PCG for scheme $\mathcal{N}1$.
(iii) Tridiagonal splitting iterations for scheme $\mathcal{N}1$.

Let us start with the results for (i). As stopping criterion we use

$$\|x_{k+1} - x_k\|_\infty < 10^{-8}, \tag{8.1}$$

where x_k denotes the kth iterate. The initial guess x_0 is chosen to be the solution from the last time step, which is a somewhat better starting point than the zero vector. The splitting method works very well because the spectral radius $\rho(M^{-1}N)$ is small, see Table 8.1. The way how the spectral radius influences the convergence speed can be seen best by an example:
We derive an estimate for the number of iterations which we will at least need using the stopping criterion (8.1). This estimate is based on the spectral radius and was described in (5.5) which we restate now for convenience:

$$\begin{aligned} & (\rho(M^{-1}N))^k \leq 10^{-m} \\ \Leftrightarrow \qquad & k \geq \frac{-m}{\log_{10}(\rho(M^{-1}N))} =: \tilde{m}. \end{aligned} \tag{8.2}$$

This estimate shows us how many iterations we will need at least in order to reduce the error to a certain percentage of its initial value, see Chapter 5. In combination with the stopping criterion (8.1) we can use knowledge on the spectral radius to predict how many iterations will be necessary. For this estimate we need some basic results for the $\|\cdot\|_\infty$ and $\|\cdot\|_2$ norm and some estimates on terms arising in this context. For the $\|\cdot\|_\infty$ we have for a matrix $A \in \mathbb{R}^{n\times n}$ that $\|A\|_2 = \sqrt{\lambda_{max}(A^TA)}$, so $\|M^{-1}N - I\|_2 \approx 1$ if the eigenvalues of $M^{-1}N$ are small. This is of course the case if the spectral radius is small. Furthermore, $\|A\|_\infty \leq \sqrt{n}\|A\|_2$, and we note that for

Table 8.1. Amount of iterations for tridiagonal splitting method using scheme $\mathcal{W}1$ with stopping criterion (8.1) and predictions ceil($\tilde{m}$) according to (8.2). $T = 1$, truncation point $\hat{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$ and strike $K = 1$.

n	65	129	257	513	1025
k	T/5	T/10	T/20	T/40	T/80
$\rho(M^{-1}N)$	0.0091	0.0055	0.003	0.0016	0.0008
ceil($\tilde{m}$)	4	4	3	3	3
iter	4	4	3	3	3

the five scenarios for n and k in Table 8.1 the term $\sqrt{n}$ will have order of magnitude of about 10^1. The initial error $\|e_0\|_\infty$ has in all cases order of magnitude 10^{-2} if we choose the solution at the previous time step as initial guess. Using the expression for the error e_k given in (5.4) we have

$$\begin{aligned}
\|x_{k+1} - x_k\|_\infty &= \|(x_{k+1} - x_*) - (x_k - x_*)\|_\infty \\
&= \|e_{k+1} - e_k\|_\infty \\
&= \|(M^{-1}N)^{k+1}e_0 - (M^{-1}N)^k e_0\|_\infty \\
&\leq \underbrace{\|(M^{-1}N)^k e_0\|_\infty}_{\approx \rho(M^{-1}N)^k \|e_0\|_\infty} \underbrace{\|M^{-1}N - I\|_\infty}_{\leq \sqrt{n}\|M^{-1}N - I\|_2} \\
&\leq \underbrace{\rho(M^{-1}N)^k}_{\leq 10^{-m}} \underbrace{\|e_0\|_\infty}_{\leq 10^{-2}} \underbrace{\|M^{-1}N - I\|_2}_{\approx 1} \underbrace{\sqrt{n}}_{\approx 10^1}.
\end{aligned}$$

So for $m = 7$ we are satisfying the stopping criterion. In Table 8.1 the spectral radius of $M^{-1}N$ for each scenario is given and ceil($\tilde{m}$) denotes our prediction of iterations for the splitting method with $\tilde{m}$ defined in (8.2), whereas *iter* is the number of observed iterations. The spectral radius is very small and therefore the reason for the rapid convergence. Additionally, it is being halved from scenario to scenario so that the number of required iterations is accordingly getting smaller. Obviously, the predictions coincide with the observed iterations in the experiment. The predictions and the actual number of iterations coincide, and we see that indeed the small spectral radius enables us to obtain a solution in only about three or four iterations.

Next we check for accuracy. Observe in Table 8.2 the second order accuracy both at the strike price as well as on the whole interval, the latter being mirrored by the l^∞ error. By l^∞ error we mean the infinity norm of the difference between the numerical solution vector and the analytical solution evaluated at the grid points in the final time $\tau = T$. As for possible improvement of accuracy, we observe in Table 8.4 and

Table 8.2. FD results with tridiagonal splitting method (using pre-compiled Thomas algorithm) for Merton's model using discretization scheme $\mathcal{W}1$. Error at the strike price $x_K = \log(K)$ and in $\|\cdot\|_\infty$ norm over the whole interval for $T = 1$. Truncation point $\hat{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$ and strike $K = 1$.

n	k	Error at x_K	l^∞ error	(sec)	Time ratio
65	T/5	0.0044	0.0051	0.06	
129	T/10	0.0010	0.0013	0.14	2.3
257	T/20	2.5365e-004	3.1665e-004	0.49	3.5
513	T/40	6.3270e-005	7.9228e-005	1.88	3.8
1025	T/80	1.5812e-005	1.9808e-005	7.50	4.0
2049	T/160	3.9530e-006	4.9517e-006	30.2	4.0

Table 8.3. FD results with split preconditioned CG method (using Strang's preconditioner) for Merton's model using discretization scheme $\mathcal{N}1$. Error at the strike price $x_K = \log(K)$ and in $\|\cdot\|_\infty$ norm over the whole interval for $T = 1$. Truncation point $\hat{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$ and strike $K = 1$.

n	k	Error at x_K	l^∞ error	(sec)	Time ratio
65	T/5	9.7235e-004	0.0024	0.05	
129	T/10	8.1181e-006	6.0094e-004	0.12	2.4
257	T/20	1.7232e-004	1.7850e-004	0.42	3.5
513	T/40	3.0994e-005	3.8414e-005	1.62	3.9
1025	T/80	3.2405e-006	9.6034e-006	6.65	4.1
2049	T/160	3.7181e-008	2.4007e-006	27.2	4.1

Table 8.5 that refinement of the space grid provides better results than refinement of the time grid, so good error rates can be expected for even an relatively rough time grid.

Furthermore, concerning the required workload the *Time ratio* in Table 8.2 is of interest. It is the fraction of the run time in the current row divided by the run time of the previous one. Note that although all computations were done in MATLAB, the Thomas solver was implemented and pre-compiled in C to obtain better comparability since PCG makes heavily use of the `fft`$(\cdot)$ function in MATLAB, which in turn uses already compiled code. The run time increases by a factor of about 4 which shows that the approach only needs $O(n \log n)$ operations since we double both space and time grid points from row to row.

Problems arise if the interest rate $r > 0$ and the volatility σ the underlying asset is small, because in this case the convection term gains more influence and oscillations

Table 8.4. FD results for tridiagonal splitting using $\mathcal{W}1$ and PCG method using $\mathcal{N}1$ for Merton's model with constant space step. Error at the strike price $x_K = \log(K)$ and in $\|\cdot\|_\infty$ norm over the whole interval for $T = 1$. Truncation point $\hat{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$ and strike $K = 1$.

n	m	Tridiag. Split. using $\mathcal{W}1$		PCG using $\mathcal{N}1$	
		Error at x_K	l_∞ error	Error at x_K	l_∞ error
65	10	4.2106e-003	5.0797e-003	7.8348e-004	2.3278e-003
65	20	4.1720e-003	5.0833e-003	7.4888e-004	2.4886e-003
65	40	4.1616e-003	5.0842e-003	7.4012e-004	2.5288e-003
65	80	4.1589e-003	5.0844e-003	7.3790e-004	2.5389e-003
65	160	4.1582e-003	5.0845e-003	7.3734e-004	2.5414e-003
65	320	4.1581e-003	5.0845e-003	7.3702e-004	2.5400e-003

Table 8.5. FD results for tridiagonal splitting using $\mathcal{W}1$ and PCG method $\mathcal{N}1$ for Merton's model with constant time step k. Error at the strike price $x_K = \log(K)$ and in $\|\cdot\|_\infty$ norm over the whole interval for $T = 1$. Truncation point $\hat{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$ and strike $K = 1$.

n	m	Tridiag. Split. using $\mathcal{W}1$		PCG using $\mathcal{N}1$	
		Error at x_K	l_∞ error	Error at x_K	l_∞ error
65	20	4.1720e-003	5.0833e-003	7.4888e-004	2.4886e-003
129	20	9.7796e-004	1.2674e-003	2.7866e-005	5.8798e-004
257	20	2.5365e-004	3.1665e-004	1.7232e-004	1.7850e-004
513	20	7.5715e-005	7.9214e-005	4.0066e-005	7.8481e-005
1025	20	3.1415e-005	3.2147e-005	1.4611e-005	6.8448e-005
2049	20	2.0351e-005	2.2252e-005	1.1986e-005	6.8725e-005

arise particularly in a neighborhood of the strike price, see Figure 8.1. This picture contains two plots (a) and (b), each showing functions $\delta(x)$ that depict the difference between numerical and analytical solution for the approaches (i) and (ii). The left plot represents an extreme case with very high interest rate r and zero volatility of the underlying, whereas (b) displays the corresponding curves for a more moderate scenario with interest rate $r = 7\%$ and volatility $\sigma = 1\%$. Still, the latter choice of parameters is not describing a typical stock since the annualized volatility is mostly around 10% to 20%. For such volatilities the oscillations disappear, but nevertheless one should prefer PIDE $\mathcal{N}$ as we see for example in Figure 8.2 for $r = 8\%$ and $\sigma = 10\%$. Historical stock volatility data can be found on the web site of the Chicago Board Options Exchange. We emphasize again that the oscillations appear in a neighborhood

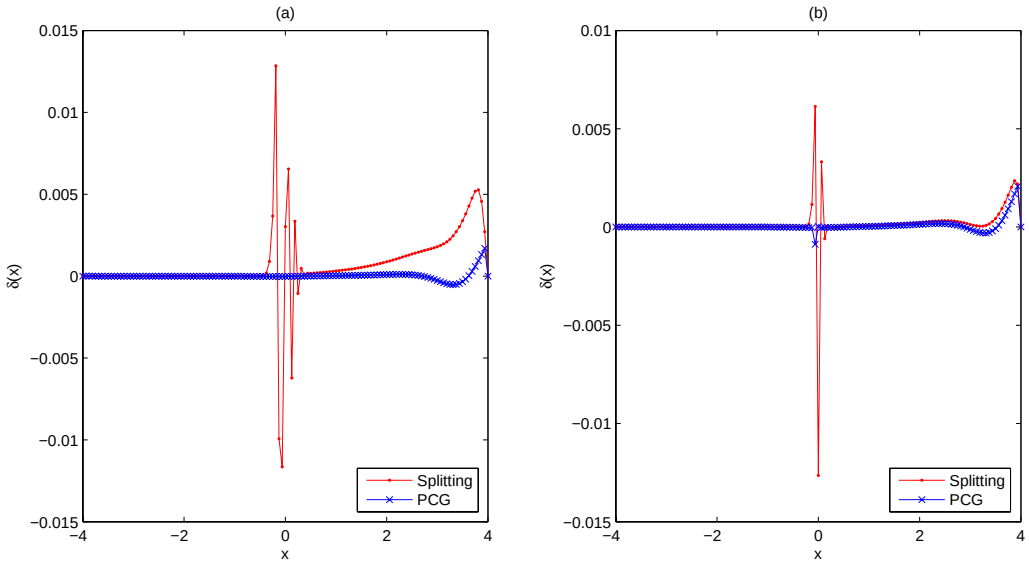

Figure 8.1. $\delta(x)$ denotes the difference between the analytical and the numerical solution for Merton's model. The splitting method is used with the discretization scheme $\mathcal{W}1$, PCG is used with scheme $\mathcal{N}1$. The variable x denotes the log-asset price. Figure (a): $r = 0.2$, $T = 1$, $\bar{x} = 4$, $\sigma = 0$, $\sigma_J = 0.5$, $\lambda = 0.1$, $n = 129$, $m = 40$ and $K = 1$. Figure (b): $r = 0.07$, $T = 1$, $\bar{x} = 4$, $\sigma = 0.01$, $\sigma_J = 0.5$, $\lambda = 0.1$, $n = 129$, $m = 10$ and $K = 1$.

of the strike price, that means just in the most important area where the current price of the underlying asset is most likely to be.

This effect can be avoided by using the transformed PIDE $\mathcal{N}$, leading us to the approach (ii). Here we use the conjugate gradient method with stopping criterion

$$\|z_k\|_\infty < 10^{-8},$$

where z_k denotes the kth residual. As initial guess for CG we use the solution at the previous time step. We already saw in Section 6.2.5 that CG will just take a few iterations per time step if we use an appropriate preconditioner such as the one from Strang which we use in all computations in this chapter. This is because the spectrum of the preconditioned coefficient matrix is clustered around 1 and only about two eigenvalues are located outside, also seen in Chapter 6. According to the convergence results of CG sketched in Chapter 2 we would therefore expect three to four iterations until convergence. If we regard the cluster around 1 as approximately one eigenvalue,

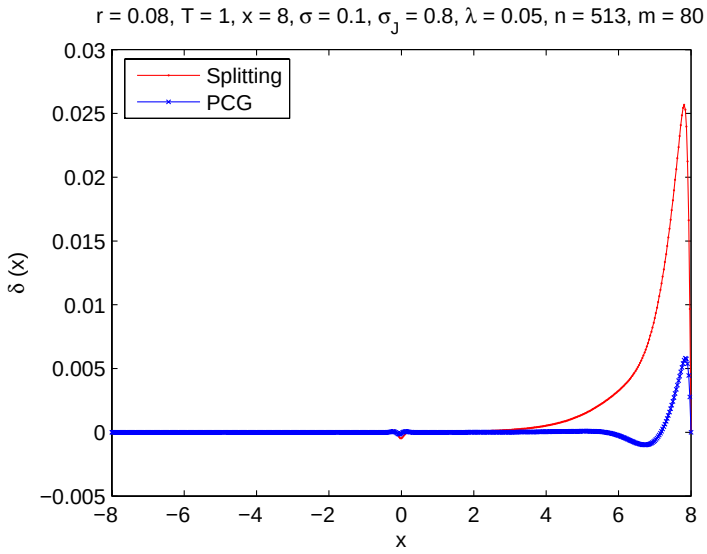

Figure 8.2. $\delta(x)$ denotes the difference between the analytical and the numerical solution for Merton's model. Splitting iterations using scheme $\mathcal{W}1$, PCG using scheme $\mathcal{N}1$. Parameters as indicated.

in total we would have three distinct ones. Hence, by Theorem 2.2, we expect convergence after three iterations. However, since the cluster is only approximately at 1 but rather sharp, about four iterations should do, which is exactly what we observed in Table 6.1.

The accuracy is of second order on the whole interval as we see from Table 8.3, and especially around the strike price x_K is the error mostly about one order of magnitude better than for the splitting approach (i) with convection term.

By keeping either the number of space grid points n or of time grid points m constant, we obtain a similar behavior like for (i) as seen in Table 8.4 and Table 8.5, so m does not need to be large. With regard to the needed amount of work again the *Time ratio* in Table 8.3 shows that the work is of order $O(n \log n)$ since the run time only increases by a factor of 4 despite doubling both the number of space and time grid points.

The stability is improved owing to the scheme $\mathcal{N}1$, see Figure 8.1. Without convection term the oscillations disappear leaving us with a far better approximation around the strike price. Comparing the run times in Table 8.2 and Table 8.3 yields that the PCG approach (ii) is slightly faster than the splitting approach (i), in spite of the fact that the tridiagonal solver was implemented in a pre-compiled version using C.

Table 8.6. FD results for splitting method with (pre-compiled) Thomas solver using $\mathcal{N}1$ for Merton's model. Error at the strike price $x_K = \log(K)$ and in $\|\cdot\|_\infty$ norm over the whole interval for $T = 1$. Parameters: Truncation point $\bar{x} = 4$, $r = 0$, $\sigma = 0.2$, $\sigma_J = 0.5$, $\lambda = 0.1$, and $K = 1$.

n	k	Error at x_K	l^∞ error	(sec)	Time ratio
65	T/5	9.7234e-004	2.3656e-003	0.05	
129	T/10	8.1124e-006	6.0090e-004	0.14	2.8
257	T/20	1.7232e-004	1.7850e-004	0.49	3.5
513	T/40	3.0991e-005	3.8396e-005	1.88	3.8
1025	T/80	3.2393e-006	9.5959e-006	7.4	3.9
2049	T/160	3.6739e-008	2.3981e-006	29.7	4.0

Finally, one might wonder whether the splitting method combined with discretization scheme $\mathcal{N}$ is better than the CG approach using the same scheme. For this purpose we have some results in Table 8.6 which we can compare with the ones in Table 8.3. The error behavior is nearly exactly the same, thus improvements in the error of the PCG approach (ii) vis-à-vis the splitting approach (i) using $\mathcal{W}1$ are due to the transformed PIDE. But from Table 8.3 and Table 8.6 we see that the PCG approach still is a bit faster than the splitting technique.

Overall we summarize that the passage from PIDE $\mathcal{W}$ to $\mathcal{N}$ yields improvements of the error behavior insofar as oscillations of the solution around the strike price are avoided, and that using a preconditioned CG method instead of splitting iterations can shorten the run time by roughly ten percent.

8.2 Kou's model

The discretization scheme $\mathcal{N}1$ of the PIDE $\mathcal{N}$ for Merton's model led us to a symmetric Toeplitz system we had to solve every time step. For Kou's model, however, this is unfortunately no longer the case if the double exponential density (3.3) is chosen to be asymmetric. This lost symmetry seems to play an important role as we shall see in the subsequent. In particular, note that the asymmetric double exponential density has a jump at zero, see Figure 8.4. An exception is the case $p := q := 1/2$ and $\alpha_1 := \alpha_2$, where the density is both symmetric and continuous. The density is then called "*the first law of Laplace*". We conduct experiments with both the asymmetric

Table 8.7. Amount of iterations for tridiagonal splitting method for Kou's model using discretization scheme $\mathcal{W}1$ with (un-)symmetric density. $r = 0$, $T = 0.2$, truncation point $\hat{x} = 6$, $\sigma = 0.2$, $\lambda = 0.2$, $\alpha_2 = 2$ and strike $K = 1$. Grid sizes: Space n, Time m.

n		65	129	257	513	1025
m		10	20	40	80	160
For $\alpha_1 = 2$:	$\rho(M^{-1}N)$	0.0015	0.0010	0.0006	0.0003	0.0002
	iter	3	3	3	3	2-3
For $\alpha_1 = 3$:	$\rho(M^{-1}N)$	0.0013	0.0009	0.0005	0.0003	0.0002
	iter	4	4	3	3	3

and symmetric double exponential density and compare the corresponding results, where we use $\alpha_1 = 3$, $\alpha_2 = 2$ and $\alpha_1 = \alpha_2 = 2$, respectively. But first, we need again to state an analytical solution for an European call in Kou's model. Kou provided such a formula in his paper *Kou* [2002] in form of an infinite series, but by using Fourier analysis it is also possible to obtain a solution in integral form (*Lewis* [2001]). We use the form given in *Almendral and Oosterlee* [2005]:

$$u(\tau, x) = \mathrm{Re}\left[-\frac{1}{\pi}\int_0^\infty \frac{\exp(-zxi + \tau\Psi(-z))}{z^2 - zi}\,\mathrm{d}y\right],$$

where $z = y + (3/2)i$ and

$$\Psi(z) = -\frac{1}{2}\sigma^2 z^2 - \left(\lambda\eta + \frac{\sigma^2}{2}\right)zi + \lambda\left(\frac{p\alpha_1}{\alpha_1 - zi} + \frac{q\alpha_2}{\alpha_2 + zi} - 1\right).$$

In the numerical experiments, the approximate solution was found by using Simpson's rule on the interval $[0, 120]$ with 2048 supporting points.
Now, the following experiments are awaiting us:

(i) Tridiagonal splitting method on $\mathcal{W}$.
(ii) P-CGNR on $\mathcal{N}$.
(iii) Tridiagonal splitting method on $\mathcal{N}$.

Let us start again with (i). The tridiagonal splitting converges again very fast because the spectral radius is small as we see from Table 8.7, independent from symmetry, and just as for Merton's model it decreases for decreasing space/time step h and k, respectively. Again we use (8.1) as stopping criterion and the solution vector of the last time step as initial guess. Turning to accuracy Table 8.8 shows that for the symmetric first law of Laplace as density we obtain globally second order accuracy, which is essentially coming from the space discretization as seen in Table 8.10. In the

Table 8.8. Tridiagonal splitting using $\mathcal{W}1$ and P-CGNR using $\mathcal{N}1$ results for Kou's model with symmetric density. Point-wise errors at maturity time $T = 0.2$. The truncation point $\hat{x} = 6$ and the parameters of the process are: volatility $\sigma = 0.2$, $r = 0$, $\alpha_1 = 2$, $\alpha_2 = 2$, $p = 0.5$, $\lambda = 0.2$, $K = 1$ and $x_K = \log(K)$. Grid sizes: Space n, Time m.

n	m	Tridiag. Split.			P-CGNR		
		Error at x_K	l_∞ error	(sec)	Error at x_K	l_∞ error	(sec)
65	10	1.6647e-002	8.8478e-002	0.04	9.5871e-003	1.1797e-001	0.05
129	20	6.1053e-003	2.2693e-002	0.09	2.3585e-003	3.0707e-002	0.12
257	40	1.3042e-003	5.7378e-003	0.30	6.5749e-005	7.7767e-003	0.35
513	80	3.1039e-004	1.4441e-003	1.14	5.9478e-005	1.9621e-003	1.34
1025	160	7.6829e-005	3.6164e-004	4.49	1.5583e-006	4.9122e-004	5.75
2049	320	1.9161e-005	9.0440e-005	17.8	1.4859e-005	1.2286e-004	25.6

Table 8.9. Tridiagonal splitting using $\mathcal{W}1$ and P-CGNR using $\mathcal{N}1$ results for Kou's model with asymmetric density. Point-wise errors at maturity time $T = 0.2$. The truncation point $\hat{x} = 6$ and the parameters of the process are: volatility $\sigma = 0.2$, $r = 0$, $\alpha_1 = 3$, $\alpha_2 = 2$, $p = 0.5$, $\lambda = 0.2$, $K = 1$ and $x_K = \log(K)$. Grid sizes: Space n, Time m.

n	m	Tridiag. Split.			P-CGNR		
		Error at x_K	l_∞ error	(sec)	Error at x_K	l_∞ error	(sec)
65	10	1.6171e-002	7.2058e-001	0.04	1.3071e-002	7.2563e-001	0.05
129	20	5.9348e-003	3.3680e-001	0.09	4.2061e-003	3.3865e-001	0.11
257	40	1.2551e-003	1.6289e-001	0.30	5.7500e-004	1.6267e-001	0.35
513	80	2.9334e-004	7.9824e-002	1.14	2.9810e-005	7.9616e-002	1.36
1025	160	7.0073e-005	3.9455e-002	4.76	9.9052e-007	3.9344e-002	5.75
2049	320	1.6222e-005	1.9607e-002	17.9	1.9138e-006	1.9532e-002	24.0

latter table the run time is approximately doubled with each doubling of the number of space grid points n, thus indicating $O(n \log n)$.

For the asymmetric double exponential density with $\alpha_1 = 3$, $\alpha_2 = 2$, the global quadratic convergence is lost, see Table 8.9. Despite this, we still find quadratic convergence around the strike price. Figure 8.3 gives us an impression of what is happening: For both the symmetric and asymmetric case the error grows at the right end of the domain. But note that for the symmetric density the scale is $\times 10^{-4}$, so the behavior near the right boundary is not as pronounced as in the asymmetric case. As for the stability, we have of course the same problems with the convection term

Table 8.10. Tridiagonal Splitting using $\mathcal{W}1$ and P-CGNR using $\mathcal{N}1$ results for Kou's model and symmetric double-exponential density ($\alpha_1 = \alpha_2 = 2$, $p = 0.5$) with constant time grid m. Parameters: $r = 0$, $T = 0.2$, $\hat{x} = 6$, $\sigma = 0.2$, $\lambda = 0.2$, $K = 1$. Grid sizes: Space n, Time m.

n	m	Tridiag. Split.			P-CGNR		
		Error at x_K	l_∞ error	(sec)	Error at x_K	l_∞ error	(sec)
65	40	1.6620e-002	8.8531e-002	0.10	9.5659e-003	1.1745e-001	0.11
129	40	6.0959e-003	2.2697e-002	0.17	2.3515e-003	3.0583e-002	0.22
257	40	1.3042e-003	5.7378e-003	0.30	6.5749e-005	7.7767e-003	0.35
513	40	3.1169e-004	1.4440e-003	0.58	6.0600e-005	1.9945e-003	0.70
1025	40	7.8471e-005	3.6162e-004	1.18	1.5621e-007	5.3188e-004	1.58
2049	40	2.0890e-005	9.0433e-005	2.34	1.6330e-005	1.6571e-004	3.69

like for Merton's model, so it would be advisable to use discretization scheme $\mathcal{N}1$ rather than $\mathcal{M}1$.

The approach (ii), that is using P-CGNR with discretization scheme $\mathcal{N}1$ unfortunately is not able to solve these problems, except for the oscillation issue. We still just need a few iterations if we use for example the displacement preconditioner as seen in Table 7.1. But due to the fact that P-CGNR needs four FFTs of length $2n$ and two FFTs of length n per iteration, whereas the splitting method only needs two FFTs of length $2n$ and the solution of a tridiagonal system, we already expect that P-CGNR will be noticeable slower. From Table 8.8 and Table 8.9 we see that P-CGNR is of order $O(n \log n)$ and shows similar convergence behavior like approach (i), but is indeed slower than the splitting approach (i). As stopping criterion for P-CGNR we use $\|r_k\|_\infty < 10^{-8}$. The error plot in Figure 8.3 leads us to the conclusion that the asymmetric discontinuous density combined with the uniform discretization grid is most likely the reason for the loss of global second order accuracy, since both methods on either $\mathcal{N}$ or $\mathcal{W}$ seem to suffer from the same problem. Only the stability will be improved, but this is of course due to scheme $\mathcal{N}1$ and not a merit of P-CGNR.

Approach (iii) improves the speed vis-à-vis approach (ii) with P-CGNR, but still Table 8.11 shows that second order is lost over the whole interval.

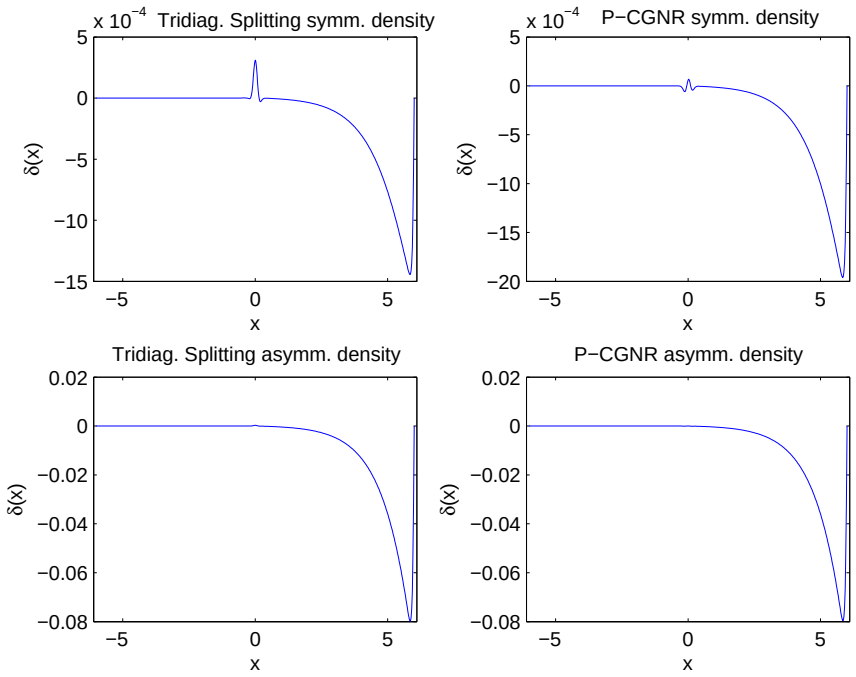

Figure 8.3. $\delta(x)$ denotes the difference between the analytical and the numerical solution for Kou's model. Splitting iterations using scheme $\mathcal{W}1$, P-CGNR using scheme $\mathcal{N}1$. Grid sizes $n = 513$, $m = 80$. Parameters same as in Table 8.10

8.3 Conclusion and future research

For Merton's model we observed global quadratic convergence for the splitting approach using discretization scheme $\mathcal{W}1$ as well as for the preconditioned CG (PCG) approach using $\mathcal{N}1$. We remark that these results oppose those of *Almendral and Oosterlee* [2005] who claimed that the quadratic convergence would be lost over the whole interval for this splitting approach with $\mathcal{W}1$, although our implementation followed exactly the way they described it. We proposed a new approach to efficiently solve the PIDE describing Merton's model by first transforming it in order to eliminate the convection term, and then using a preconditioned CG method to solve the symmetric Toeplitz systems arising in every time step due to the implicit discretization. The transformation stabilizes the solution which might otherwise contain oscillations in a neighborhood of the strike price. Furthermore, the PCG method uses FFTs throughout and the solution is therefore obtained in only $O(n \log n)$ operations.

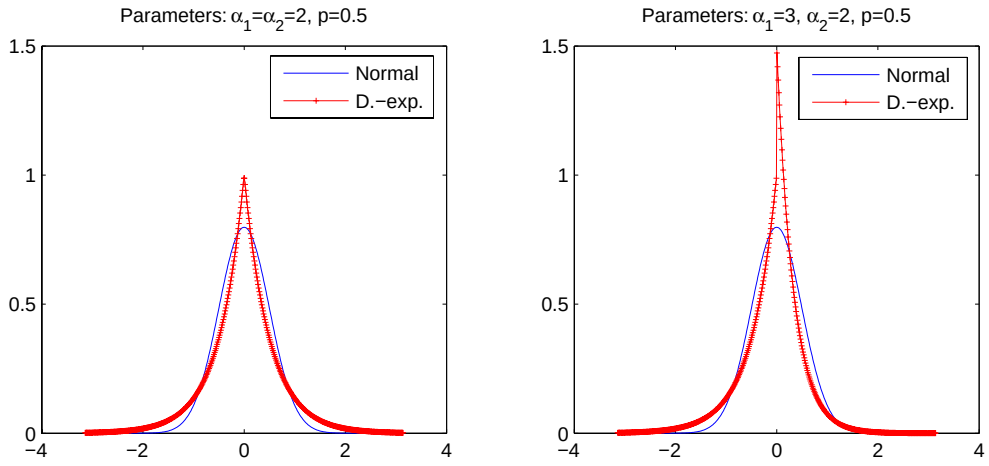

Figure 8.4. Double exponential and normal density function. Figure (a) depicts the symmetric special case and Figure (b) an asymmetric example. The normal distributions have both mean zero and volatility $\sigma_J = 0.5$.

Table 8.11. Tridiagonal splitting results using $\mathcal{N}1$ for Kou's model with asymmetric density. Point-wise errors at maturity $T = 0.2$. The truncation point $\hat{x} = 6$ and the parameters of the process are: $r = 0$, volatility $\sigma = 0.2$, $\alpha_1 = 3$, $\alpha_2 = 2$, $p = 0.5$, $\lambda = 0.2$, $K = 1$ and $x_K = \log(K)$. Grid sizes: Space n, Time m.

n	k	Error at x_K	l^∞ error	Time (sec)	Time ratio
65	10	1.3072e-002	7.2564e-001	0.04	
129	20	4.2062e-003	3.3865e-001	0.09	2.3
257	40	5.7500e-004	1.6267e-001	0.30	3.3
513	80	2.9811e-005	7.9617e-002	1.12	3.7
1025	160	9.9054e-007	3.9344e-002	4.65	4.2
2049	320	1.9135e-006	1.9533e-002	18.8	4.0

PCG is equally effective as the tridiagonal splitting technique and turns out to be even slightly faster if we use an appropriate circulant preconditioner for the Toeplitz systems.

For Kou's model the global second order accuracy does not carry over, only if the double exponential density in Kou's model is chosen to be symmetric, then global second order accuracy can be maintained. Otherwise it is only linear, and quadratic around the strike price. The asymmetry of the density causes the linear systems that we need to solve in every time step to be likewise asymmetric, so that PCG is not directly applicable. But we can use a tridiagonal splitting or PCG on the normal equations (P-CGNR) instead. The two methods yield similar results if used on the

convection-free scheme $\mathcal{N}1$, except for the fact that P-CGNR is somewhat slower. Note, however, that also the approach using P-CGNR provides the solution in only $O(n \log n)$ operations, and just a few vectors need to be stored.

Future research can be undertaken by improving the performance for Kou's model. Instead of the splitting technique one could use a version of GMRES, with the matrix-vector product being carried out by FFTs. Since P-CGNR converged very quickly we probably would only need to store a few Krylov-vectors. However, the main problem seems to be the discretization itself for Kou's model. Using a nonuniform grid makes global second order accuracy possible, see for example the recent work by *Ikonen and Toivanen* [2006]. A problem in using a nonuniform grid is that the linear systems lose the Toeplitz structure, so that we cannot apply FFT any more to accelerate the matrix-vector product.

Bibliography

Almendral, A., and C. Oosterlee, Numerical valuation of options with jumps in the underlying, *Applied Numerical Mathematics*, *53*, 1–18, 2005.

Ammar, G., and W. Gragg, The generalized Schur algorithm for the superfast solution of Toeplitz systems, in *Lecture Notes in Mathematics*, vol. 1237, edited by J. Gilewicz, M. Pindor, and W. Siemaszko, pp. 315–330, Springer-Verlag, 1987.

Andersen, L., and J. Andreasen, Jump-Diffusion Processes: Volatility Smile Fitting and Numerical Methods for Option Pricing, *Rev. of Derivatives Res.*, *4*, 231–262, 2000.

Bates, D., Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche Mark options, *Rev. Financial Studies*, *9*, 69–107, 1996.

Bitmead, R., and B. Anderson, Asymptotically fast solution of Toeplitz and related systems of linear equations, *Linear Algebra Appl.*, *34*, 103–116, 1980.

Black, F., and M. Scholes, The Pricing of Options and Corporate Liabilities, *Journal of Political Economy*, *81* (3), 637–654, 1973.

Brent, R., F. Gustavson, and D. Yun, Fast solution of Toeplitz systems of equations and computation of Padé approximants, *J. Algo.*, *1*, 259–295, 1980.

Briani, M., Numerical methods for option pricing in jump-diffusion markets, Ph.D. thesis, Università degli Studi di Roma "La Sapienza", 2003.

Briani, M., R. Natalini, and G. Russo, Implicit-explicit numerical schemes for jump-diffusion processes, *38 (4/2004)*, IAC Report, 2004.

Chan, R., Circulant preconditioners for Hermitian Toeplitz systems, *SIAM J. Matrix Anal. Appl.*, *10*, 542–550, 1989a.

Chan, R., The spectrum of a family of circulant preconditioned Toeplitz systems, *SIAM J. Numer. Anal.*, *26*, 503–506, 1989b.

Chan, R., and G. Strang, The asymptotic Toeplitz-circulant eigenvalue problem, *87-5*, MIT App. Math. Dept., 1987.

Chan, R., and C. Wong, Best-conditioned circulant preconditioners, *Linear Algebra Appl.*, *218*, 205–212, 1995.

Chan, R., and M. Yeung, Circulant preconditioners for Toeplitz matrices with positive continuous generating functions, *Math. Comp.*, *58*, 233–240, 1992.

Chan, R., X. Jin, and M. Yeung, The circulant operator in the Banach algebra of matrices, *Linear Algebra Appl.*, *149*, 41–53, 1991a.

Chan, R., X. Jin, and M. Yeung, The spectra of super-optimal circulant preconditioned Toeplitz systems, *SIAM J. Numer. Anal.*, *28*, 871–879, 1991b.

Chan, R., J. Nagy, and R. Plemmons, Displacement preconditioner for Toeplitz least squares iterations, *Elec. Trans. Numer. Anal.*, *2*, 44–56, 1994.

Chan, T., An optimal circulant preconditioner for Toeplitz systems, *SIAM J. Sci. Stat. Comput.*, *9*, 766–771, 1988.

Chan, T., Pricing contingent claims on stocks driven by Lévy processes, *Ann. Appl. Probab.*, *9*(2), 504–528, 1999.

Chun, J., T. Kailath, and H. Lev-Ari, Fast parallel algorithms for QR and triangular factorization, *SIAM J. Sci. Stat. Comp.*, *8*, 899–913, 1987.

Cont, R., and P. Tankov, *Financial Modelling with Jump Processes*, Chapman & Hall/CRC, Boca Raton, FL, 2004.

Cont, R., and E. Voltchkova, A finite difference scheme for option pricing in jump diffusion and exponential Lévy models, *SIAM J. Numer. Anal.*, *43*(4), 1596–1626, 2005.

Conte, S., and C. de Boor, *Elementary numerical analysis*, McGraw-Hill, New York, 1980.

Cooley, J., and J. Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, *Mathematics of Computation*, *19*(90), 297–301, 1965.

Davis, P., *Circulant matrices*, John Wiley & Sons, New York, 1979.

d'Halluin, Y., P. Forsyth, and G. Labahn, A penalty method for American options with jump diffusion processes, *Numerische Mathematik*, *97*, 321–352, 2004.

d'Halluin, Y., P. Forsyth, and K. Vetzal, Robust numerical methods for contingent claims under jump diffusion processes, *IMA J. of Numerical Analysis*, *25*, 87–112, 2005.

Durbin, J., The fitting of time series models, *Rev. Int. Stat.*, *28*, 233–244, 1960.

Freund, R., and T. Huckle, Iterative solution of linear systems with low displacement rank by conjugate gradient-type algorithms, in *XII Householder Symp.*, Lake Arrowhead, CA, 1993.

Golub, G., and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, 1996.

Gradshteyn, I., and I. Ryzhik, *Table of Integrals, Series, and Products*, Academic Press, San Diego, 2000.

Grenander, U., and G. Szegö, *Toeplitz forms and their applications*, 2nd ed., Chelsea Publishing, New York, 1984.

Heston, S., A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options, *The Review of Financial Studies*, *6*(2), 327–343, 1993.

Hull, J., and A. White, The Pricing of Options on Assets with Stochastic Volatilites, *The Journal of Finance*, *42*(2), 281–300, 1987.

Ikonen, S., and J. Toivanen, Numerical valuation of European and American options with Kou's jump-diffusion model, Amamef Conference on Numerical Methods in Finance, Inria-Rocquencourt, France, 2006.

Kailath, T., and A. Sayed, Displacement structure: Theory and Applications, *SIAM Review*, *37*, 297–386, 1995.

Kou, S., A Jump-Diffusion Model for Option Pricing, *Management Science*, *48*(8), 1086–1101, 2002.

Levinson, N., The Wiener rms (root mean square) error criterion in filter design and prediction, *J. Math. Phys.*, *25*, 261–278, 1947.

Lewis, A., A simple option formula for general jump-diffusion and other exponential Lévy processes, Center for Applied Probablility Columbia University, 8th Annual CAP Workshop on Derivative Securities and Risk Management, 2001.

Linzer, E., Extended circulant conditioning of Toeplitz systems, *Technical Report*, IBM Research, Yorktown Heights, NY, 1992.

Luenberger, D., *Introduction to Linear and Nonlinear Programming*, 2nd ed., Addison Wesley, 1984.

Matache, A.-M., T. von Petersdorff, and C. Schwab, Fast deterministic pricing of options on Lévy driven assets, *Mathematical Modelling and Numerical Analysis*, *38*(1), 37–71, 2004.

Matache, A.-M., P.-A. Nitsche, and C. Schwab, Wavelet Galerkin pricing of American options on Lévy driven assets, *Quantitative Finance*, *5*(4), 403–424, 2005a.

Matache, A.-M., C. Schwab, and T. Wihler, Fast numerical solution of parabolic integrodifferential equations with applications in finance, *SIAM J. Sci. Comput.*, *27*, 369–393, 2005b.

Merton, R., Option pricing when underlying stock returns are discontinuous, *Journal of Financial Economics*, *3*, 125–144, 1976.

Ng, M., *Iterative Methods for Toeplitz Systems*, Oxford University Press, New York, 2004.

Nocedal, J., and S. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

Pustyl'nikov, L., On the algebraic structure of the spaces of Toeplitz and Hankel matrices, *Soviet Math. Dokl.*, *21*, 141–144, 1980.

Raible, S., Lévy Processes in Finance: Theory, Numerics, and Empirical Facts, Ph.D. thesis, Albert-Ludwigs-Universität Freiburg i. Br., 2000.

Saad, Y., *Iterative Methods for Sparse Linear Systems*, 1st ed., PWS, Boston, 1996.

Sato, K., *Lévy Processes and Infinitely Divisible Distributions*, Cambridge University Press, Cambridge, UK, 1999.

Schur, I., Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind, *J. Reine Angew. Math.*, *147*, 205–232, 1917.

Strang, G., A proposal for Toeplitz matrix calculations, *Stud. Appl. Math.*, (74), 171–176, 1986.

Strela, V., Exploration of circulant preconditioning properties, *Matrix Methods and Algorithms*, pp. 9–46, 1993.

Strela, V., and E. Tyrtyshnikov, Which circulant preconditioner is better?, *Math. Comp.*, *65*(213), 137–150, 1996.

Tismenetsky, M., A decomposition of Toeplitz matrices and optimal circulant preconditioner, *Linear Algebra Appl.*, *156*, 105–121, 1991.

Trench, W., An algorithm for the inversion of finite Toeplitz matrices, *SIAM J. Appl. Math.*, *12*, 515–522, 1964.

Tyrtyshnikov, E., Optimal and superoptimal circulant preconditioners, *SIAM J. Matrix Anal. Appl.*, *13*(2), 459–473, 1992.

Van Loan, C., *Computational frameworks for the fast Fourier transform*, SIAM, Philadelphia, 1992.

Widom, H., Toeplitz matrices, *Studies in Real and Complex Analysis*, I. Hirshman Jr, MAA, 1965.

Wilmott, P., *Derivatives - The theory and practice of financial engineering*, John Wiley & Sons, 1998.

Young, D., *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.