NTNU
Norwegian University of
Science and Technology

# Numerical solution of non-local PDEs arising in Finance.

**Håkon Berg Johnsen**

Master of Science in Physics and Mathematics
Submission date: June 2009
Supervisor: Espen Robstad Jakobsen, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

# Problem Description

The project aims at finding efficient implementations of a new type of numerical method for non-local PDEs arising in finance. Evaluation of non-local terms using a fast Fourier transform, Crank-Nicholson time discretization and 2nd order space discretization are pursued. Numerical tests will be performed along with theoretical stability and convergence analysis.

Assignment given: 18. January 2009
Supervisor: Espen Robstad Jakobsen, MATH

# Abstract

It is a well known fact that the value of an option on an asset following a Levy jump-process, can be found by solving a Partial Integro-Differential Equation (PIDE). In this project, two new schemes are presented to solve these kinds of PIDEs when the underlying Levy process is of infinite activity. The infinite activity jump-process leads to a singular Levy measure, which has important numerical ramifications and needs to be handled with care. The schemes presented calculate the non-local integral operator via a fast Fourier transform (FFT), and an explicit/implicit operator splitting scheme of the local/global operators is performed. Both schemes will be of 2nd order on a regular Levy measure, but the singularity degrades convergence to lie in between 1st and 2nd order depending on the singularity strength. On the logarithmically transformed PIDE, the schemes are proven to be consistent, monotone and stable in $L^\infty$, hence convergent by Barles-Perthame Souganidis.

# Acknowledgments

This work has been a great learning experience for me, and I would like to thank my supervisor Espen R. Jakobsen for taking the time to aid me. His feedback and support has been greatly appreciated. I would also thank my fellow student Kjetil A. Johannessen and my brother Magnus Berg Johnsen for reading the thesis and giving viable input and TEXsupport. Last, I would like to thank my girlfriend Ingunn Jystad Postmyr for her continued support through the last 4 years of my masters degree.

# Contents

# Chapter 1

# Introduction

Options are, contrary to common perceptions, not just part of employee benefit programs. They are financial instruments traded in vast numbers every day, and in a number of different forms. Options would never have been so popular unless one were able to set the 'correct' price of such contracts. Fischer Black and Myron Scholes revolutionized this branch of finance with [3] and founded the large field of financial mathematics.

As is well known, the ordinary Black Scholes model has its flaws, one of which became all too apparent when exchanges all over the world plunged down last autumn. According to the Black Scholes model, such drastic changes in value have close to zero probability over such a short timespan. In addition a key assumption by Black Scholes is that while the stock is crashing you should continuously be able to hedge yourself in/out of your position [3], which simply is not the case.

This article will focus on an improved framework of models, where jumps are naturally incorporated, namely the rich class of Levy jump-processes. Throughout this paper only the CGMY process [4], which is a subclass of Levy jump-processes, is assumed to govern the underlying stock. The CGMY process is an infinite activity jump-process, which leads to a singular Levy measure. This singularity has important numerical ramifications, and will for instance affect the convergence of the schemes. [4] showed that the CGMY process is able to yield impressive fits to real market data, and is hence a good foundation for further algorithm development.

The algorithms developed in this article builds upon the schemes presented by Jakobsen et al. [8]. These schemes are proven to have rigorous convergence properties in $L^\infty$, which comes from the use of a monotone discretization. With a consistent, monotone and $L^\infty$-stable scheme, convergence is assured by Barles et.al. [2]. This convergence theory is very robust, and extends even to non-linear problems.

The schemes from [8] are also very generally applied to a wide range of problems. This generality and convergence does however come at a cost, which is a rather high computational complexity. A full integral operator is calculated at each time step at the cost of $O(N^2)$ with $N$ nodes in the computational grid, and the convergence rate of the schemes are only of 1st order.

The aim of this thesis is to find a new and more efficient discretization, which lowers

the computational complexity and increases the convergence to 2nd order, while maintaining the generality and monotonicity of the schemes. One obvious way of obtaining the first goal is to rewrite the integral term to a form which allows the computation via a fast Fourier transform (FFT), thus decreasing the $O(N^2)$ computation pr. timestep to $O(N \log N)$. This will be done following the idea from D'halluin et al. [6] via the introduction of an auxiliary logarithmic grid. For each timestep, the solution will be interpolated onto the auxiliary grid, where the integral is calculated via FFT. The computed integral is then interpolated back to the 'solution grid' where the local operators are approximated and a step in time is performed. The approach in [6] is however not directly applicable in the case considered in this thesis, as they only consider a non-singular Levy measure. Hence some adaptations is made, as will be explained in detail in (Chapter 3).

The particular discretization in time also follows the lines of [6], where an operator splitting approach is pursued. The local operators form a tridiagonal matrix and is hence very easy to handle implicitly, whereas the global integral operator forms a full matrix and is hence more efficient to handle explicitly. The resulting equation system is solved by a fixed point iteration proven to be convergent under a CFL condition.

Two 2nd order schemes are presented, applicable for different strengths of the singularity of the Levy measure. One of which is obtained by an approach that is new in this article, where the combination of a crude 1st order discretization of the derivative with a crude 1st order riemann discretization of the integral yields a remarkable 2nd order combined convergence. The other scheme obtains 2nd order via individual 2nd order discretization of the operators. Both schemes are however affected by the singularity of the Levy measure, which degrades convergence to lie in between 1st and 2nd order, depending on the parameter values. This will be thoroughly discussed in the theory, and will further be shown during the numerical tests.

The introduction of the auxilliary grid, is however not entirely straightforward, as even though the integral approximation is monotone on the logarithmic grid, interpolating the calculated integral back to the computational grid does in fact not yield monotonicity. There is a rather simple way out of this problem, by simply transforming the entire equation to logarithmic variables, as will be shown in the analysis chapter, although this approach leads to a loss of generality. Hence by introducing these higher order schemes with lowered computational complexity, one can make a choice. Either losing generality, by transforming everything to logarithmic variables, and thus not be able to easily adapt the scheme to a similar equation. Or one could use the auxiliary logarithmic grid, and lose the monotonicity. The second approach is what will be done throughout the numerical testing, but emphasis will still be on monotone approximations in either of the grids. In the analysis chapter, the modifications necessary to obtain monotonicity is shown. That is done by transforming the equation to logarithmic variables, and monotonicity and $L^\infty$ stability on the easier PIDE will be proved. Consistency of the schemes is of course proved in both cases, hence convergence is assured by [2] in the logarithmically transformed, easier problem.

A 2 dimensional problem is also considered and solved numerically on a simplified

model, that is no correlation in either of the two spatial directions. The point being to show whether a 2D problem is feasible to solve or not, and to outline how it could be done.

# Chapter 2

# The PIDE introduced

## 2.1 Levy Processes and the PIDE

The class of Levy jump processes is extremely rich, and presenting the theoretical framework in all of its rigour is beyond the scope of this work. For a thorough treatment see [5].

There exists both infinite and finite activity Levy jump processes, although this work is based upon an infinite activity case. An infinite activity jump model is, as the name implies, simply a model where an infinite number of jumps occur in every interval. The infinite number of jumps is of course somewhat less analytically tractable than the regular brownian motion assumed by Black-Scholes [3]. Hence closed formed solutions to the PIDE arising from the European option pricing problem, only exists on simple model problems.

One very interesting fact is that when considering infinite activity Levy process, as the CGMY process considered here, one does not need to introduce a brownian component, since the dynamics of the jumps allready is rich enough to generate nontrivial small time behaviour [4]. With the brownian term removed, the second derivative in the local operators disappears, and one could relax the CFL condition deduced in the analysis hence lowering the computational complexity. This will not be pursued in this thesis, but is noted as an alternative approach.

Following the derivation in [5], let a stock $S_t$ follow an exponential-Levy model:

$$S_t = e^{rt+X_t}, \tag{2.1.1}$$

where $X$ is a Levy process with characteristic triplet $(\sigma^2, \nu, \gamma)$ under some risk-neutral measure $\mathbb{Q}$ such that $\hat{S}_t = e^{-rt}S_t = e^{X_t}$ is a martingale. The risk neutral dynamics of $S_t$ is then given by

$$S_t = S_0 + \int_0^t rS_{u-}du + \int_0^t S_{u-}\sigma dW_u + \int_0^t \int_{-\infty}^{\infty} (e^x - 1)S_{u-}\tilde{J}_X(du, dx), \tag{2.1.2}$$

where $\tilde{J}_X$ denotes the compensated jump measure of the Levy process $X$ (Proposition 8.20 in [5]) and $\hat{S}_t$ is a square-integrable martingale:

$$\frac{d\hat{S}_t}{\hat{S}_{t-}} = \sigma dW_t + \int_{-\infty}^{\infty} (e^x - 1)\tilde{J}_X(dtdx), \quad \sup_{t\in[0,T]} E[\hat{S}_t^2] < \infty. \tag{2.1.3}$$

The value of a European option $u(s,t)$, is given as a discounted conditional expectation of its terminal payoff $H(S_T)$ under the equivalent martingale measure $\mathbb{Q}$: $u_t = E[e^{-r(T-t)}H(S_T)|\mathcal{F}_t]$. From the Markov property:

$$u(S,t) = E[e^{-r(T-t)}H(S_T)|S_t = S]. \tag{2.1.4}$$

Now applying Ito's formula plus the risk neutral dynamics of $S_t$ (2.1.2), the price of $u(s,t)$ is the solution of the following PIDE [5]:

$$u_t = -rSu_S - \frac{\sigma^2 S^2}{2}u_{SS} + ru - \int \nu(dy)[u(Se^y,t) - u(S,t) - S(e^y - 1)u_S], \tag{2.1.5}$$

with appropriate boundary conditions (BC), and final conditions, given by the option payoff, which will be introduced later. Final conditions are turned into initial conditions by simply solving the equation backward in time, ie. set $\tau = T - t$, and hence get:

$$u_\tau = rSu_S + \frac{\sigma^2 S^2}{2}u_{SS} - ru + \int \nu(y)[u(Se^y,\tau) - u(S,\tau) - S(e^y - 1)u_S]dy. \tag{2.1.6}$$

One rather interesting observation is that if the integral terms from (2.1.6) gets removed, that is $\nu(y) = 0$, the Black Scholes PDE appears [3].

In this work we use the CGMY model, where $\nu(y)$ has the form:

$$\nu(y) = \begin{cases} C\frac{e^{-M|y|}}{|y|^{1+\alpha}} & \text{if} \quad y > 0 \\ C\frac{e^{-G|y|}}{|y|^{1+\alpha}} & \text{if} \quad y < 0, \end{cases} \tag{2.1.7}$$

with $G, M > 1$, $C > 0$ and $0 < \alpha < 2$. This particular choice of measure is singular at $y = 0$, with the heuristic interpretation that there is an infinite number of jumps with size zero.

As mentioned in the introduction, a 2 dimensional model is also solved numerically. Assume a put option whose payoff is governed by (2.3.1) and two underlying stocks $x_t$ and $y_t$, whose price processes are both individually governed by (2.1.1). By assuming that the $x$ and $y$ Brownian motions are correlated with $\rho$ while the jump measure is uncorrelated, the following equation results by multidimensional Ito calculus [5]:

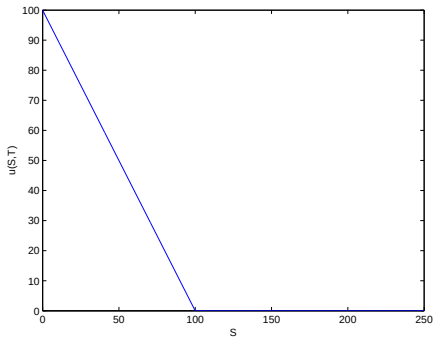


Figure 2.1: The Payoff of the Put on $S$ at $t = T$ (2.2.1). $K = 100$.

$$u_\tau = rxu_x + \frac{\sigma_x^2 x^2}{2} u_{xx} + ryu_y + \frac{\sigma_y^2 y^2}{2} u_{yy} - ru + \rho\sigma_x\sigma_y xyu_{xy} \tag{2.1.8}$$

$$+ \int \nu(dz)[u(xe^z, y, \tau) - u(x, y, \tau) - x(e^z - 1)u_x] \tag{2.1.9}$$

$$+ \int \nu(dz)[u(x, ye^z, \tau) - u(x, y, \tau) - y(e^z - 1)u_y] \tag{2.1.10}$$

$$\tag{2.1.11}$$

## 2.2 Options Defined

There are actually a vast number of different options. In this thesis however, the type will be constrained to European puts. Buying a European put on a stock at $t = 0$, gives the right to sell the stock for a given price, $K$, at a given date $t = T$. Performing this transaction is called exercising the option.

The name 'European' is in no way related to the continent Europe, which might seem a bit confusing, it is simply related to the fact that a European option can only be exercised at $t = T$. The 'opposite' being an American option which can be exercised at all times $0 < t \leq T$. Other types of options are Asian, Bermudian etc [12].

Exercising the option of course only makes sense if $K$ is greater than the price of the stock at $t = T$, $S_T$. Thus giving the following income at $t = T$:

$$H(S_T) = \max(K - S_T, 0), \tag{2.2.1}$$

also plotted in (Figure 2.1).

If $S_T \geq K$, exercising the option means one sells the stock for less than it is worth, which is rather pointless. Because, as the name implies, an option means there is a choice, and as mentioned, it gives the right but not the obligation, to sell the stock for

$K$ at $t = T$. Hence the put ends up being worthless when $S_T \geq K$. So buying a put might seem like a risky deal, although in combination with other assets it might actually reduce the overall risk. For instance holding one stock plus a put on the stock acts like an insurance on the stock. No matter what happens, your portfolio (1 stock and 1 put) ends up being worth at least $K$ at expiry. If $S_T < K$ you simply exercise the option, giving you the right to sell the stock for $K$. If $S_T \geq K$, the put expires worthless, but as the value of the stock is higher than $K$ the combined value is still higher than $K$.

The morale being the 'riskyness' of the put depends on the investors preferences, and having the opportunity to buy options adds flexibility.

## 2.3 Boundary/Initial values

Final conditions for $t = T$ are found from the option payoff, e.g. for a put they are shown in (2.2.1).

In addition, for reasons which will be clear during the discussion of the discretization, values to the left and right of the computational grid is needed during the solution process. In the case of a put, these values can be approximated arbitrarily well, depending on where the computational grid is truncated at $S = S_{max}$.

In fact with the type of option discussed in this thesis, i.e. puts on stocks, the value at $S = 0$ is known for all $t$. In that case, the company is bankrupt, and a put will certainly end up being worth $K$. Discounted back to present time, $T - t$, with a continously compunding rate of return, $r$, the value at $S = 0$ is simply $Ke^{-r(T-t)}$.

The value at $S = S_{max}$ can heuristically be determined if $S_{max} \to \infty$. In that case you are almost certain the put ends up out of the money (ie. $S > K$ when the option expires). In that case the value will simply be 0. How large $S_{max}$ in reality will be, when it comes to the numerical implementation, is explained in the Implementation chapter.

The type of option in the two dimensional case is a basket put on the combined value of two stocks $x$ and $y$. Several types of payoffs on these kinds of options exists, like e.g.

$$H(x_T, y_T) = \max(K - \min(x_T, y_T), 0),$$
$$H(x_T, y_T) = \max(K - \max(x_T, y_T), 0)$$

and perhaps the most natural payff:

$$H(x_T, y_T) = \max(K - x_T - y_T, 0), \tag{2.3.1}$$

which is used in this thesis. Where $x_T$ and $y_T$ are the stock prices of the two underlying stocks at $t = T$. (2.3.1) is displayed in (Figure 2.2).

Now when it comes to boundary conditions on the two dimensional problem it is slightly more complicated than in 1D. First consider the computational grid the equation will be solved on:

As can be seen from (Figure 2.3) BC is needed along the blue lines: $x = 0$, $x = x_{max}$, $y = 0$ and $y = y_{max}$. By the same argumentation as in the 1D-case, the value of the option at $x = x_{max}$ and $y = y_{max}$ can be set to 0. Along the lines $x = 0$ and $y = 0$ it gets

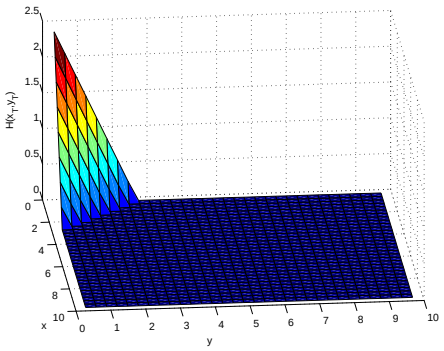


Figure 2.2: The Payoff of the Put on $x$ and $y$ from 2.3.1. $K = 3$.

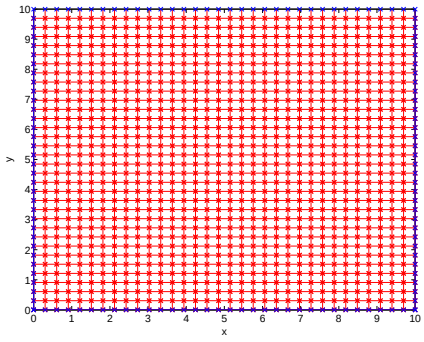


Figure 2.3: The computational grid when solving the 2D equation with 32 nodes in each spatial direction distributed equidistantly. BC is needed along the blue lines.

more difficult. If either of the underlying stocks $x$ or $y$ is equal to 0 then that company has gone bankrupt, and to obtain the value of the basket put one has to solve the 1D PIDE to find the value of the option on the other stock alone. This fact can also be seen from just setting $x$ or $y$ equal to 0 in (2.1.8) and (2.3.1).

Hence for each step in time of the 2-dimensional PIDE, two 1-dimensional problems has to be solved (one for $x = 0$ and one for $y = 0$). This might seem rather harsh, but there is a rather efficient way of solving the problem. As equidistant timesteps are used, the values of $t$ where the BC are needed, are known before the solution loop is started. So when solving for the BC when $y = 0$, the solution is simply to solve the 1D problem with the same grid as that in the $x$-direction, and with the same stepsize in $t$. Then simply store all of the values. Similarily on the line where $x = 0$. The cost of doing that is the cost of solving the 1D problem twice ie. $O(MN \log N)$, which will be completely negligable compared to the cost of solving the full 2D problem.

# Chapter 3

# Discretizing the PIDE

A key aspect in the implementation will be an operator splitting approach. This will allow the partial derivatives appearing in the IPDE, i.e. the local operators, to be treated in an implicit fashion whereas the integral term, ie. the nonlocal operator, will be handled explicitly. This to avoid forming and inverting a dense matrix operator at each timestep. The motivation of the implicit approach is to obtain a more relaxed CFL condition. By doing a little more work for each iteration, but performing fewer iterations, CPU-time could be saved. The alternative is of course a fully explicit method.

In addition, as mentioned in the introduction, there will be formed two computational grids, one on which the correlation product is computed in logarithmic variables, and one on which the local operators are discretized. The non-logarithmic grid in the 'main' grid, the logarithmic grid is only an auxiliary grid to be able to utilize FFT. It is very easy to transform everything to the logarithmic grid and solve everything there, but to keep things more general, that approach will only be pursued in the analysis chapter, where it will be showed that this transformation will in fact yield a monotone algorithm.

Due to the operator splitting approach, the integral term is only handled explicitly, hence the discrete values which are involved in the correlation product are interpolated onto the logarithmic grid where the computed is performed, before it gets interpolated back.

The next sections are dedicated to the particular discretization of the integral term.

## 3.1 $\alpha \in (0, 1)$

### 3.1.1 Transforming the integral

As mentioned, a key ingredient in my contribution is to compute

$$\int_{\mathbb{R}\setminus\{0\}} [u(Se^y) - u(S) - S(e^y - 1)\frac{\partial u}{\partial S}]\nu(dy), \tag{3.1.1}$$

via a Fast Fourier Transform (FFT), to reduce the computational complexity of evaluating N of such integrals every timestep from $O(N^2)$ to $O(NlogN)$, while still maintining

the monotonicity from [8]. To enable the use of FFT, the integral (3.1.1) is rewritten as a correlation integral via a change of variables and an integration by parts. But first the integral needs to be simplified somewhat. Consider the last term of the integral (3.1.1):

$$\int_{\mathbb{R}\setminus\{0\}} -S(e^y-1)\frac{\partial u}{\partial S}\nu(dy) = -S\frac{\partial u}{\partial S}\int_{-\infty}^{\infty}(e^y-1)\nu(dy), \tag{3.1.2}$$

by techniques as in [1]:

$$\omega_1 = \int_{\mathbb{R}\setminus\{0\}}(e^y-1)\nu(dy) = C\Gamma(-\alpha)[(M-1)^\alpha - M^\alpha + (G+1)^\alpha - G^\alpha]. \tag{3.1.3}$$

With the parameters $C$, $G$, $M$ $\alpha$ from the CGMY measure (2.1.7) and $\Gamma(-\alpha)$ being the gamma function. This is possible because $\alpha \in (0,1)$, and hence the singularity of the integrand at $y = 0$ is 'weak enough' such that this term can be integrated out. When considering $\alpha \in [1,2)$, which is done in a later section, this is not the case. The term $-Su_S\omega_1$ enters in (2.1.6) turning that equation into (3.4.1) which will be introduced later. It is easily seen from (3.1.3) that in order for the integral to converge $M, G > 1$.

Now look at the remaining parts of (3.1.1), first on the positive real axis. With an integration by parts plus the chain rule, one ends up with:

$$\int_0^\infty [u(Se^y)-u(S)]\nu(dy) = [[u(Se^y)-u(S)](-\overline{\nu}(y))]_0^\infty - \int_0^\infty \partial_y[u(Se^y)](-\overline{\nu}(y))dy, \tag{3.1.4}$$

where:

$$\overline{\nu}(y) = \begin{cases} \int_y^\infty \nu(z)dz & y > 0 \\ \int_{-\infty}^y \nu(z)dz & y < 0 \end{cases} \tag{3.1.5}$$

with $\nu(y)$ as defined in (2.1.7).

The term:

$$[[u(Se^y)-u(S)](-\overline{\nu}(y))]_0^\infty, \tag{3.1.6}$$

can be shown to be equal to zero by a taylor expansion around $y = 0$ with $u \in C^2$, and remembering that $M, G > 1$.

(3.1.5) will be rather cumbersome to compute at all the points needed to evaluate (3.1.4), even though it only needs to be computed once (as it stays fixed for all $t$), but with the same techniques which led to (3.1.3) it can be shown [1] that it is equal to:

$$\overline{\nu}(z) = \begin{cases} CM^\alpha\Gamma(-\alpha, Mz) & y > 0, \\ CG^\alpha\Gamma(-\alpha, G|z|) & y < 0, \end{cases} \tag{3.1.7}$$

with $\Gamma(a,b)$ being the incomplete gamma function, defined in [1]. Although one still needs a fast and efficient way to compute the incomplete gamma function, which is where [13] comes in. Their proposed algorithm is implemented and used throughout the paper.

Via a similar derivation of the integral on the negative half axis, the integral operator turns out to be:

$$\begin{aligned}\int_{\mathbb{R}\setminus\{0\}}[u(Se^y)-u(S)]\nu(dy) &= -\int_0^\infty \partial_y[u(Se^y)](-\overline{\nu}(y))dy - \int_{-\infty}^0 \partial_y[u(Se^y)](\overline{\nu}(y))dy \\ &= \int_0^\infty \partial_y[u(Se^y)](\overline{\nu}(y))dy - \int_{-\infty}^0 \partial_y[u(Se^y)](\overline{\nu}(y))dy.\end{aligned} \tag{3.1.8}$$

In order to be able to exploit FFT in the computation of (3.1.8), a change of variable needs to be introduced, going from $S$ to $\log S$. In the new $\log S$ variable, the integral operator takes the form of a correlation integral, which via a proper discretization is turned into a discrete correlation product, which can be utilized by FFT.

Now let $x = log(S)$. Then $S = e^x$ and with the modified function:

$$\overline{u}(x) = u(e^x), \tag{3.1.9}$$

the integral turns out to be:

$$\int_0^\infty \partial_y \overline{u}(x+y)\overline{\nu}(y)dy - \int_{-\infty}^0 \partial_y \overline{u}(x+y)\overline{\nu}(y)dy, \tag{3.1.10}$$

which is the point of departure for the discretization.

### 3.1.2 Discretizing the integral

Now as the logarithmic variable is introduced, and the integral is on the form of a correlation integral (3.1.10), all that remains is to discretize (3.1.10) to a correlation product, before the fast fourier transform can be applied. First the discrete equidistant logarithmic grid is introduced:

Let $x$ denote the variables in the logarithmic grid, such that $x = \log(S)$, and let $N_{log}$ be the number of gridpoints in the logarithmic grid. Then the discretized $\log S$-grid is a one-to-one mapping from the $S$-grid:

$$x_1 = \log S_1,$$

$$x_{N_{log}} = \log S_N,$$

further the (logarithmic) distance between two adjacent gridpoints in the $\log(S)$-grid:

$$h_{log} = \frac{log(S_N) - log(S_1)}{N_{log} - 1},$$

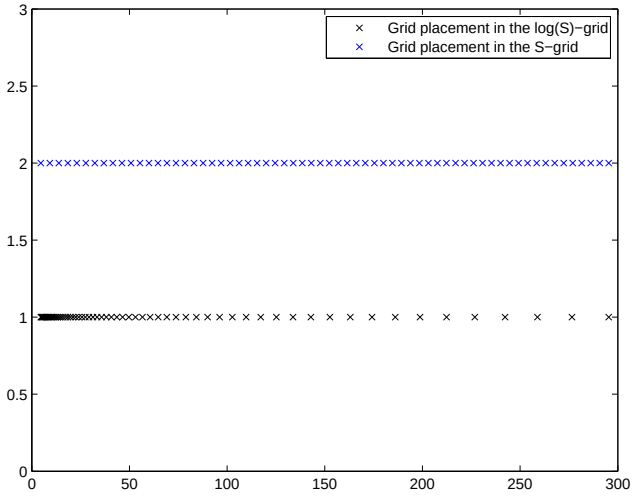


Figure 3.1: The two equidistant grids plotted in $S$-coordinates. $N = N_{log} = 64$

such that:

$$x_i = x_1 + (i-1)h_{log} \quad 1 \leq i \leq N_{log}. \tag{3.1.11}$$
$$\tag{3.1.12}$$

Now when interpolating from the $S$-grid to the $\log S$-grid it is important that the distance in $S$-coordinates (the 'physical' distance) between two gridpoints in the $\log S$-grid:

$$e^{x_{i+1}} - e_i^x = e^{x_1+ih_{log}} - e^{x_1+(i-1)h_{log}} = e^{x_i}(e^{h_{log}} - 1), \tag{3.1.13}$$

gets refined proportionally to $h_{log}$. As it is this distance squared which will be the truncation error in interpolating from the $\log S$-grid to the $S$-grid. Via a taylor expansion of $e^y$ for $y \ll 1$:

$$e^y = 1 + y + O(y^2),$$

it is easily seen that:

$$e^{x_i}(e^{h_{log}} - 1) = e^{x_i}h_{log} + O(h_{log}^2). \tag{3.1.14}$$

Ie. the 'physical' distance between gridpoints in the logarithmic grid scales like $h_{log}$, for $h_{log} \ll 1$.

With an appropriate grid, discretizing and computing (3.1.10) is within reach. But first the way to compute $\partial_y \overline{u}(x_i + y_j)$ needs to be addressed. Due to the monotonicity requirement, forward differentiation is the method of choice:

$$\begin{aligned} \delta_{y+}\overline{u}(x_i + y_j) &= \frac{\overline{u}(x_i + y_j + h_{log}) - \overline{u}(x_i + y_j)}{h_{log}} \\ &= \partial_y \overline{u}(x_i + y_j) - \frac{1}{2}\overline{u}_{xx} h_{log} + \frac{Err_{interp}}{h_{log}} + O(h_{log}^2). \end{aligned} \tag{3.1.15}$$

The second last term of the truncation error might seem unfamiliar. But remember the values at $x_i + y_j + h_{log}$ and $x_i + y_j$ are both found by interpolation from the $S$-grid. This particular error gets divided by $h_{log}$ in (3.1.15). At first glance it might seem like linear interpolation, with truncation error $O(h^2)$, should be sufficient. But, as shall soon be revealed, combining this crude first order discretization with a particular quadrature also of first order, actually turns into a 2nd order combined discretization. Hence 3rd, or higher, order interpolation needs to be employed. The particular interpolation will be introduced in the next section, taken from [9].

Now from (3.1.15), at all the discretized gridpoints $x_i + y_j$, the values of $\overline{u}(x_i + y_j + h_{log})$ is needed. An important point is however that the distance between two adjacent $y$-values, is set equal to the distance between two adjacent values of $x$, i.e.:

$$y_{j+1} - y_j = x_{j+1} - x_j = h_{log} \qquad \forall j. \tag{3.1.16}$$

This particular choice of stepsize in $y$ is important to ensure overlapping grids in $x$ and $y$, which is needed to be able to utilize the fast fourier transform. In addition, for reasons which will soon be clear: $y_0 = 0$, such that:

$$\overline{u}(x_i + y_j) = \overline{u}(x_{i+j})$$

and

$$\overline{u}(x_i + y_j + h_{log}) = \overline{u}(x_{i+j+1}),$$

hence (3.1.15) can be written as:

$$\begin{aligned} \delta_{y+}\overline{u}(x_i + y_j) &= \delta_{y+}\overline{u}(x_{i+j}) \\ &= \frac{\overline{u}(x_{i+j+1}) - \overline{u}(x_{i+j})}{h_{log}} \\ &= \partial_y \overline{u}(x_{i+j}) - \frac{1}{2} h_{log} \partial_{yy} u(x_{i+j}) + O(h_{log}^2), \end{aligned} \tag{3.1.17}$$

assuming a 3rd, or higher, order interpolation is used. In order for the algorithm to be monotone in the end, an additional approximation of the first derivative needs to be introduced, that is backward differentiation:

$$\begin{aligned}\delta_{y-}\overline{u}(x_i+y_j) &= \frac{u(x_{i+j})-u(x_{i+j-1})}{h_{log}} \\ &= \partial_y\overline{u}(x_i+y_j)+Err_{approx} \\ &= \partial_y\overline{u}(x_i+y_j)+\frac{1}{2}\overline{u}_{xx}h_{log}+O(h^2),\end{aligned} \tag{3.1.18}$$

with truncation error still assuming 3rd, or higher, order interpolation is used. (3.1.18) is needed in the approximation of the integral on the negative real axis.

One obvious alternative to (3.1.17) would be to use forward/backward differences in the $S$-grid plus the chain rule, and interpolate the calculated derivatives on the log $S$-grid, rather than interpolating and then calculating the derivates. Such an approach would however not yield a 2nd order combined convergence, neither a monotone approximation in the log $S$-grid, which is important to obtain a monotone algorithm by the full log $S$ transformation. Hence (3.1.17) and (3.1.18) are the methods of choice.

Now before the discretized integral is introduced, there is of course a need of truncating (3.1.10) before $\pm\infty$. With exponentially decaying Levy measure, the error could be chosen to be arbitrarily small, say $y_{max} = y_{N_{ext}} = N_{ext}h_{log}$, and $y_{min} = y_{-N_{ext}} = -N_{ext}h_{log}$, then as

$$\nu(y) < e^{-\min(M,G)|y|}, \quad |y| > 1,$$

and $M$, $G > 1$, it is not hard to achieve an insignificant error. A typical value of $N_{ext}$ is $\frac{N_{log}}{2}$, which is more than sufficient in the numerical experiments below.

When choosing the method to compute the integral, there are a number of issues to consider. First in order the have a monotone method, the modified quadrature weights (that is the weights multiplied with the Levy measure) have to be strictly decreasing if $y > 0$ and strictly increasing for $y < 0$.

Second, as I have mentioned, no matter how high the order of the quadrature is, the first order error of the upwinding will be the dominating factor. Unless, and this is an important point, the upwinding and quadrature schemes are combined in an intelligent manner, using the truncation error of the quadrature to cancel the first order truncation error of the upwinding scheme.

Say the allready truncated integral at $y = N_{ext}h_{log}$, on the positive half axis, is split

into $N_{ext}$ parts, each going from $y_j$ to $y_{j+1}$:

$$\begin{aligned}\int_0^{y=N_{ext}h_{log}} \partial_y \overline{u}(x_i+y)\overline{\nu}(y)dy &= \sum_{j=0}^{j=N_{ext}} \int_{y_j}^{y_{j+1}} \partial_y \overline{u}(x_i+y)\overline{\nu}(y)dy \\ &= \sum_{j=0}^{j=N_{ext}} \int_{\xi=0}^{\xi=h_{log}} \partial_y \overline{u}(x_{i+j}+\xi)\overline{\nu}(y_j+\xi)d\xi,\end{aligned}$$

with $\xi \in [0, h_{log}]$. Then consider each part of the sum from the equation over:

$$\int_0^{h_{log}} \partial_y \overline{u}(x_{i+j}+\xi)\overline{\nu}(y_j+\xi)d\xi. \tag{3.1.19}$$

Do a taylor expansion of $\partial_y \overline{u}(x_{i+j}+\xi)$ around $\xi = 0$:

$$\partial_y \overline{u}(x_{i+j}+\xi) = \partial_y \overline{u}(x_{i+j}) + \xi \partial_{yy}\overline{u}(x_{i+j}) + O(\xi^2). \tag{3.1.20}$$

Insert the approximation $\delta_{y+}\overline{u}(x_{i+j})$ instead of $\partial_y \overline{u}(x_{i+j})$ plus the truncation term of that particular approximation (3.1.17):

$$\begin{aligned}\partial_y \overline{u}(x_i+y) &= \partial_y \overline{u}(x_{i+j}) + \xi \partial_{yy}\overline{u}(x_{i+j}) + O(\xi^2) \\ &= \delta_{y+}\overline{u}(x_{i+j}) - \frac{1}{2}h_{log}\partial_{yy}\overline{u}(x_{i+j}) + \xi \partial_{yy}\overline{u}(x_{i+j}) + O(\xi^2) \\ &= \delta_{y+}\overline{u}(x_{i+j}) + (\xi - \frac{1}{2}h_{log})\partial_{yy}\overline{u}(x_{i+j}) + O(\xi^2).\end{aligned} \tag{3.1.21}$$

Now insert the resulting first order error term $(\xi - \frac{1}{2}h_{log})\partial_{yy}\overline{u}(x_{i+j})$ into the integral:

$$\int_0^{h_{log}} (\xi - \frac{1}{2}h_{log})\partial_{yy}\overline{u}(x_{i+j})\overline{\nu}(y_j+\xi)d\xi. \tag{3.1.22}$$

By taylorexpanding $\overline{\nu}(y_j+\xi)$ around $(y_j + \frac{h_{log}}{2})$ it can be shown that the first order terms vanishes, and the second order term gets integrated up to 3rd order. When taking the sum of $N_{ext}$ (which is in magnitude with $N_{log}$) such parts of the integral, the 3rd order local truncation turns into 2nd order globally. Yielding an in fact 2nd order discretization, which is quite remarkable considering each part of the discretization (a modified riemann sum) and upwinding is both individually 1st order.

$\overline{\nu}$ does however have a singularity at $y = 0$, so the first order error term does in fact not vanish on the inner part of the sum where $y_j = 0$. As $\overline{\nu}$ scales like

$$\tilde{\nu}(y) \approx \frac{1}{y^\alpha}, |y| \ll 1, \tag{3.1.23}$$

the truncation error around $y = 0$ gets integrated up to $O(h_{log}^{2-\alpha})$. Yielding global truncation error of $O(h_{log}^{2-\alpha})$. This $\alpha$-dependent convergence is verified in the numerical tests run under the analysis chapter.

On the negative half axis a completely similar procedure is done, although with backward differences instead of forward differences. This choice is necessary to keep a positive discretization, and to obtain the 2nd (or $2-\alpha$ due to the singular measure) order convergence. If forward differences are used on the negative half axis, or backward differences are used on the positive half axis, the result is a loss of a positive discretization, and a degraded convergence into 1st order. Another curiosity is that if the exact derivatives $\partial_y \overline{u}(x_{i+j})$ is used instead of $\delta_{y+}$ or $\delta_{y-}$, the order also gets degraded to first order.

So in the end the discretization turns out to be:

$$\begin{aligned} I(\overline{u}(x_i)) = I_i &= \sum_{j=0}^{N_{ext}} \delta_{y+}\overline{u}(x_{i+j}) \int_{jh_{log}}^{(j+1)h_{log}} \overline{\nu}(y)dy \\ &- \sum_{j=-N_{ext}}^{0} \delta_{y-}\overline{u}(x_{i+j}) \int_{(j-1)h_{log}}^{jh_{log}} \overline{\nu}(y)dy \\ &= \sum_{j=0}^{N_{ext}} \delta_{y+}\overline{u}(x_{i+j})\overline{k}_j^+ - \sum_{j=-N_{ext}}^{0} \delta_{y-}\overline{u}(x_{i+j})\overline{k}_j^- . \end{aligned} \tag{3.1.24}$$

where

$$\begin{aligned} \overline{k}_j^+ &= \int_{jh_{log}}^{(j+1)h_{log}} \overline{\nu}(y)dy \quad 0 \le j \le N_{ext} \\ \overline{k}_j^- &= \int_{(j-1)h_{log}}^{jh_{log}} \overline{\nu}(y)dy \quad 0 \ge j \ge -N_{ext} \end{aligned} \tag{3.1.25}$$

That is a modified riemann sum combined with forward/backward differences. This way of employing a rieman sum might seem strange, but in addition to the obvious convergence properties of the discretization (order 2-$\alpha$), there is a very good reason for not just picking a value of $\overline{\nu}$ at each interval, and rather integrating up $\overline{\nu}$ at each part of the sum. Close to zero, where $\overline{\nu}$ has a singularity, getting the contribution numerically is a challenge. Just picking a value between $y = 0$ and $y = h_{log}$, will certainly yield a high error (Consider $\overline{\nu}(y)$ plotted at two different resolutions (Figure 3.2). Calculating $2N_{ext}$ such integrals with quadrature when solving the PIDE might seem like a bottleneck, but keep in mind that $\overline{k}_j^+$ and $\overline{k}_j^-$ are independent of $t$. Hence these $2N_{ext}$ integrals, each over a distance $h_{log}$, can be precalculated and stored before the solution loop starts. This cost will be completely negligable to solving the full algorithm, which is shown in the result chapter, where exact solution times are presented for different accuracies.

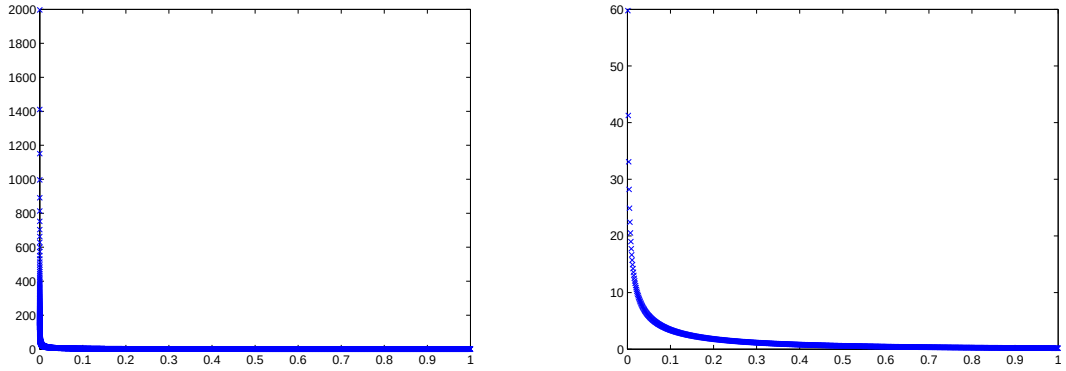


Figure 3.2: $\overline{\nu}(y)$ for $y > 0$, $M = 3$,$\alpha = 0.5$. Both plots are on an equidistant grid, although the scale differs. Left:$y_{min} = h_{log} y = 10^{-6}$. Right: $y_{min} = h_{log} y = 10^{-3}$

The observant reader might notice that when computing $I_1$ (3.1.24), the values $\overline{u}_j$ where $-N_{ext} \leq j \leq 0$ are needed in the sum. $I_2$ requires $-N_{ext} + 1 \leq j \leq 0$ etc. None of which is defined. The solution is to interpolate the values between $S_0$, where dirichlet B.C. are set, and $S_1$ which is calculated for each timestep, into an expanded grid:

$$x_i = x_1 + i h_{log} \qquad -N_{ext} \leq i \leq 0 \tag{3.1.26}$$

and just use these interpolated values in (3.1.24). Such that the first terms when computing $I_1$ on the negative half axis are:

$$I_1 = \frac{\overline{u}(x_1) - \overline{u}(x_0)}{h_{log}} \overline{k}_0^- + \frac{\overline{u}(x_0) - \overline{u}(x_{-1})}{h_{log}} \overline{k}_{-1}^-,$$

where $\overline{u}(x_0)$ etc. is found by the interpolation prosedure. Infinitely many points in the logarithmic grid can be found between $S_0$ and $S_1$ as:

$$S_0 = 0 < e^{x_1 - k h_{log}} \leq e^{x_1} = S_1 \qquad \forall k \leq -1 \tag{3.1.27}$$

Similarily $I_{N_{log}}$ requires $u_i$ where $N_{log} < i \leq Nlog + N_{ext}$ which also is outside the computational domain. The solution is the use the asymptotic B.C deduced in (Chapter 2) on these $N_{ext}$ nodal values, the same procedure is applied on $I_{N_{log}-1}$ etc. So in total there are $N_{log}$ nodes in the logarithmic grid constructed by the procedure described earlier (ie. logarithmically equidistantly placed between $\log S_1$ and $\log S_N$), and $2N_{ext}$ nodes outside of the computational domain with the same distance $h_{log}$. The number of nodes to the right and left of the grid can of course differ, but to ease the notation they are set equal to each other, such that both the integral on the positive and negative

half axis is truncated at $y_{\pm j} = \pm N_{ext} h_{log}$. Yielding a total of $2N_{ext} + N_{log}$ nodes in the logarithmic grid.

The need of involving values of the solution $u$ outside the computational domain is also intuitively clear, as the jumpterm allows the underlying to jump outside the specified grid.

Finally the integral term is on the form of a discrete correlation product(3.1.24), such that FFT can be exploited.

### 3.1.3 Speeding up the computation by FFT

The Fast Fourier Transform assumes periodic functions, which in this case is a wrong assumption, neither $\overline{k}^+$, $\overline{k}^-$ nor the derivative of $\overline{u}$ are periodic. This will however be ok if certain measures are taken. First the two zero-indexed discrete vectors $\boldsymbol{k}^+ \in \mathbb{R}^{N_{log}+2N_{ext}}$ and $\boldsymbol{k}^- \in \mathbb{R}^{N_{log}+2N_{ext}}$ is initialized. With zero-indexing the elements are:

$$\begin{aligned} \boldsymbol{k}[i]^+ &= \overline{k}_j^+ \quad 0 \leq i \leq N_{ext} \\ \boldsymbol{k}[i]^- &= \overline{k}_j^- \quad i = 0 \\ \boldsymbol{k}[Nlog + 2N_{ext} + i + 1]^- &= \overline{k}_j^- \quad -N_{ext} \leq i \leq -1. \end{aligned} \tag{3.1.28}$$

As can be seen it is simply a remapping of indices, such that element $j = -1$ of $\overline{k}^-$, gets placed in the last element of $\boldsymbol{k}^-$, $j = -2$ gets placed in the second last etc.

The values at indices which are not defined above is set to zero, such that there is 1 non-zero entry at the '0th' index in both vectors, and $N_{ext}$ non-zero values at the top of $\boldsymbol{k}^+$ and $N_{ext}$ nonzero entries at the bottom of $\boldsymbol{k}^-$. The rest are simply zero in order to avoid wrap around [11].

The discrete vector $\delta_{y+}\overline{\boldsymbol{u}} \in \mathbb{R}^{2N_{ext}+N_{log}}$ on the other hand has $N_{ext}$ interpolated derivatives in the top and bottom computed by (3.1.17), and $N_{log}$ values in the middle also computed by (3.1.17). $\delta_{y-}\overline{\boldsymbol{u}}$ is constructed in a similar way with backward differences (3.1.18) rather than forward differentiation. Now FFT can be applied. As is well known [6]:

$$FFT(I)_k = FFT(\delta_{y+}\overline{\boldsymbol{u}})_k FFT(\boldsymbol{k}^+)_k^* - FFT(\delta_{y-}\overline{\boldsymbol{u}})_k FFT(\boldsymbol{k}^-)_k^*, \tag{3.1.29}$$

with * denoting the complex conjugate. By computing the correlation product by FFT, periodicity is assumed, this is however avoided by the zero-padding of $\boldsymbol{k}^+$ and $\boldsymbol{k}^-$ and the expanded interpolation of $\delta_{y+}\overline{\boldsymbol{u}}$ and $\delta_{y-}\overline{\boldsymbol{u}}$. Thus you only get wrap around to the expanded values [11] (ie. the top and bottom $N_{ext}$ values). This is however ok, as the solutions at these points gets peeled off anyway, and only the correlation at the $N_{log}$ middle indices is needed from this computation. It is important to note that the use of FFT constraints $N_{log} + 2N_{ext}$ to be a power of 2 (well actually it does not, but the algorithm is a lot faster when the number of nodes is a power of a small prime,

even though it scales like $O(N \log N)$ anyway [7]). The citation is the particular FFT implemenation used.

The above discussion regarding $\delta_{y+}\overline{\boldsymbol{u}}$ is completely similar in the construction of $\delta_{y-}\overline{\boldsymbol{u}}$, with the only difference being backward differentiation is used.

In addition it is important to calculate all that remains constant in time before the solution process, in order to avoid calculating these every time they are needed (ie. each timestep). Such as the FFT of $\boldsymbol{k}^+$ and $\boldsymbol{k}^-$ (3.1.28) and the matrices for interpolating back and forth the logarithmic and non-logarithmic grids.

The short-version of calculating (3.1.24) by FFT follows in (Algorithm 1):

**Algorithm 1** Calculating (3.1.24) by FFT

1: Before the solution loop starts for all timesteps, compute $\overline{k}^+$ and $\overline{k}^-$ by (3.1.25).
2: Remap the indices by (3.1.28) into $\boldsymbol{k}^+$ and $\boldsymbol{k}^-$ and store the FFT of these two vectors.
3: Interpolate $2N_{ext} + N_{log}$ values of $u$ onto the logarithmic grid. $N_{log}$ values lies within $S_1 \leq e_i^x \leq S_N$, $N_{ext}$ values lies within $S_0 < e_i^x < S_1$ and the rest of the values lies for $S_N < e_i^x$.
4: Compute two different approximations of the derivative by forward(3.1.17) and backward discretization (3.1.18), on the $N_{log} + 2N_{ext}$ allready interpolated values. Insert the values in $\delta_{y+}\overline{\boldsymbol{u}}$ and $\delta_{y-}\overline{\boldsymbol{u}}$.
5: Compute the FFT of $\delta_{y+}\overline{\boldsymbol{u}}$ and $\delta_{y-}\overline{\boldsymbol{u}}$.
6: Compute the two products in Fourier space (3.1.29)
7: Take the inverse FFT of the sum from the previous step
8: Discard the the top and bottom $N_{ext}$ values, as explained previously.
9: Interpolate back to the $S$-grid

### 3.1.4 A general 2nd order discretization

As the truncation error on a general problem with riemann discretization and upwinding is limited to $O(h_{log})$, a different discretization, which is order $2-\alpha$ in general is also introduced. This particular discretization will have truncation error of order 2 on each of the individual discretizations (both the quadrature, and the derivative approximation), hence there will be no combined 'superconvergence' as in the previous case. This $2-\alpha$ discretization is only presented and will not be analysed further. The reason being that the riemann discretization (3.1.24) is in fact better on this problem, as it is monotone for all $\alpha$ and gridresolutions in the $\log S$-grid. This particular discretization will only be monotone for $\alpha > 1 - \frac{\log 2}{\log 3} \approx 0.37$, whereas the riemann sum is easily seen to be monotone $\forall \alpha$ in the results chapter. Hence this discretization is only suggested as an alternative and will not be further analyzed.

First the quadrature is introduced. Instead of the riemann discretization, a midpoint method is employed:

$$
\begin{aligned}
I(\overline{u}(x_i)) = I_i &= \sum_{j=1}^{N_{ext}} \delta_{2y+}\overline{u}(x_{i+j}) \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \overline{\nu}(y)dy + \delta_{2y+}\overline{u}(x_i) \int_0^{\frac{h_{log}}{2}} \overline{\nu}(y)dy \\
&- \sum_{j=-N_{ext}}^{-1} \delta_{2y-}\overline{u}(x_{i+j}) \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \overline{\nu}(y)dy - \delta_{2y-}\overline{u}(x_i) \int_{-\frac{h_{log}}{2}}^{0} \overline{\nu}(y)dy \\
&= \sum_{j=0}^{N_{ext}} \delta_{2y+}\overline{u}(x_{i+j})\overline{k}_j^{2+} - \sum_{j=-N_{ext}}^{0} \delta_{2y-}\overline{u}(x_{i+j})\overline{k}_j^{2-},
\end{aligned}
\tag{3.1.30}
$$

where $\delta_{2y+}$ and $\delta_{2y-}$, are second order positive discretizations of $\partial_y$ which will be introduced later, and $\overline{k}_j^{2-}$ and $\overline{k}_j^{2+}$ are defined as:

$$
\begin{aligned}
\overline{k}_j^{2+} &= \int_0^{\frac{h_{log}}{2}} \overline{\nu}(y)dy && j = 0 \\
\overline{k}_j^{2+} &= \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \overline{\nu}(y)dy && 1 \le j \le N_{ext} \\
\overline{k}_j^{2-} &= \int_{-\frac{h_{log}}{2}}^{0} \overline{\nu}(y)dy && j = 0 \\
\overline{k}_j^{2-} &= \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \overline{\nu}(y)dy && -1 \ge j \ge -N_{ext}.
\end{aligned}
\tag{3.1.31}
$$

The integrals around $y = 0$ does in fact not employ the midpoint value, but on a regular integrand, the truncation error at those two points would be $O(h_{log}^2)$. The truncation error of the sum of the other $2N_{ext}$ parts of the sum, where the midpoint value is in fact used, should be $O(h_{log}^2)$ (sum of $2N_{ext}$ terms, which is in magnitude with $N_{log}$, each with local truncation error $O(h_{log}^3)$ yielding global $O(h_{log}^2)$). Hence 2nd order convergence should arise. The truncation error of the inner terms is however again degraded by the singularity and the total scheme is only $O(h_{log}^{2-\alpha})$. This is easily seen by a similar taylorexpansion as in the riemanndiscretization, with 2nd order accurate approximation of the derivative.

Discretizing the first derivative by 2nd order, while still maintaining monotonicity is a problem. Central differences are for example not monotone. So how to achieve such a thing? The answer is actually a neat trick: Use upwinding, but instead of using steplength $h_{log}$, use $h_{log}^2$ [9]. The discrete forward difference operator $\delta_{2y+}$ with steplength

$h_{log}^2$ is hence introduced:

$$\begin{aligned}\partial_y \overline{u}(x_i) &= \delta_{2y+}\overline{u}(x_i) + O(h_{log}^2) + \frac{Err_{interp}}{h_{log}^2} \\ &= \frac{\overline{u}(x_i + h_{log}^2) - \overline{u}(x_i)}{h_{log}} + O(h_{log}^2) + \frac{Err_{interp}}{h_{log}^2},\end{aligned}$$

and backward in a similar manner of course:

$$\begin{aligned}\partial_y \overline{u}(x_i) &= \delta_{2y-}\overline{u}(x_i) + O(h_{log}^2) + \frac{Err_{interp}}{h_{log}^2} \\ &= \frac{-\overline{u}(x_i - h_{log}^2) + \overline{u}(x_i)}{h_{log}} + O(h_{log}^2) + \frac{Err_{interp}}{h_{log}^2}.\end{aligned}$$

By using linear interpolation in this case, consistency of the scheme will not even be obtained with $h_{log} = h$, as the truncation error will be $O(1)$. Clearly the order of the interpolation algorithm has to be of 4th order. Otherwise, convergence gets ruined. Still the requirement is monotonicity, thus the algorithm proposed by [9] is implemented and used. Which is of 4th order on monotone data. Consider the function $f(x)$ of one variable and the cubic Hermite interpolant on each subinterval $[x_i, x_{i+1}]$, in the following just assume constant steplength $h$ is used to ease the notation:

$$(I_h f)(x) = c_0 + c_1(x - x_i) + c_2(x - x_i)^2 + c_3(x - x_i)^3, \tag{3.1.32}$$

with parameters $c_i$ fulfilling:

$$\begin{aligned}(I_h f)(x_i) = f_i \qquad (I_h f')(x_i) = f_i' \\ (I_h f)(x_{i+1}) = f_{i+1} \quad (I_h f')(x_{i+1}) = f_{i+1},\end{aligned} \tag{3.1.33}$$

where $f_i = f(x_i)$ and $f_i'$ is some estimate of the first order derivative. Now this implies:

$$\begin{aligned}c_0 = f_i \qquad\qquad c_1 = f_i' \\ c_2 = \frac{3\Delta_i - f_{i+1}' - 2f_i'}{h} \quad c_3 = -\frac{2\Delta_i - f_{i+1}' - f_i'}{h^2},\end{aligned} \tag{3.1.34}$$

where $\Delta_i = \frac{f_{i+1} - f_i}{h}$ is just the regular upwinding procedure of finding the derivative, whereas $f'$ gets replaced by a higher order derivative approximation. In this particular algorithm [9], a 4th order accurate approximation of the first order derivative is used:

$$f_i' = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h}. \tag{3.1.35}$$

The resulting interpolation will in fact not be monotone, the parameters still need to be tweaked somewhat. Let:

$$\alpha = \frac{f_i'}{\Delta_i}, \quad \beta = \frac{f_{i+1}'}{\Delta_i}, \tag{3.1.36}$$

$$\tag{3.1.37}$$

now the interpolant is monotonic if and only if $\alpha$ and $\beta$ lies within the domain $\mathcal{M} = M_c \bigcup M_b$:

$$M_c = \{(\alpha,\beta) : (\alpha-1)^2 + (\alpha-1)(\beta-1) + (\beta-1)^2 - 3(\alpha+\beta-2) \leq 0\} \tag{3.1.38}$$

$$M_b = \{(\alpha,\beta) : 0 \leq \alpha \leq 3, 0 \leq \beta \leq 3\}. \tag{3.1.39}$$

$$\tag{3.1.40}$$

if $(\alpha, \beta)$ does not fullfill (3.1.38) the following must be done to tweak the paramters in order to achieve monotonicity (Algorithm 2).

The rest of the computation is similar to the previous riemann discretization. Ie. just follow (Algorithm 1) with $\overline{k}^{2+}$ and $\overline{k}^{2+}$ instead of $\overline{k}^{+}$ and $\overline{k}^{-}$ and of course the 2nd order upwinding.

As mentioned, this particular discretization will only be a positive approximation if:

$$\begin{aligned} \overline{k}_{j-1}^{2+} &> \overline{k}_j^{2+} \quad \forall j > 0 \\ \overline{k}_j^{2-} &> \overline{k}_{j-1}^{2-} \quad \forall j < 0, \end{aligned}$$

Which will be seen under the Analysis chapter. This criterion is easily seen to be true for $j > 1$, but for $j = 1$ this is not necessarily the case:

Remember the scaling of $\overline{\nu}(y)$ for $|y| \ll 1$ (3.1.23) and consider $\overline{k}_0^{2+} - \overline{k}_1^{2+}$:

$$\begin{aligned} \overline{k}_0^{2+} - \overline{k}_1^{2+} &= \int_0^{\frac{h_{log}}{2}} \overline{\nu}(y)dy - \int_{\frac{h_{log}}{2}}^{h_{log}+\frac{h_{log}}{2}} \overline{\nu}(y)dy \\ &\approx \int_0^{\frac{h_{log}}{2}} \frac{1}{y^\alpha}(y)dy - \int_{\frac{h_{log}}{2}}^{h_{log}+\frac{h_{log}}{2}} \frac{1}{y^\alpha}(y)dy \\ &\approx 2(\frac{h_{log}}{2})^{1-\alpha} - \frac{3h_{log}}{2})^{1-\alpha} \\ &\Downarrow \\ \alpha &> 1 - \frac{\log 2}{\log 3} \approx 0.37. \end{aligned}$$

Where the last condition on $\alpha$ is to ensure $\overline{k}_0^{2+} - \overline{k}_1^{2+} > 0$. Hence this particular discretization will asymptotically only be positive for $\alpha > 0.37$.

**Algorithm 2** Modifying $\alpha_i$ and $\beta_i$ to ensure monotonicity

Compute $f_i'$ and $f_{i+1}'$ by (3.1.35).
Compute $\Delta_i$
**if** $\Delta_i \neq 0$ **then**
  Compute $\alpha_i$ and $\beta_i$ by (3.1.36)
**else**
  Set $\alpha_i = 1$, $\beta_i = 1$.
**end if**
**if** $\Delta_i \neq 0$ **then**
  Compute $\alpha_i$ and $\beta_i$ by (3.1.36)
**else**
  Set $\alpha_i = 1$, $\beta_i = 1$.
**end if**
**if** $\alpha_i < 0$ **then**
  $\alpha_i = 0$
**end if**
**if** $\beta_i < 0$ **then**
  $\beta_i = 0$
**end if**
**if** $(\alpha, \beta) \notin \mathcal{M}$ **then**
  **if** $\alpha_i \geq 3 \&\& \beta_i \geq 3$ **then**
    set $\alpha_i = \beta_i = 3$
  **else if** $\beta_i > 3$ AND $\alpha_i + \beta_i \geq 4$ **then**
    decrease $\beta_i$ such that $(\alpha_i, \beta_i) \in \partial\mathcal{M}$.
  **else if** $\beta_i > 3$ AND $\alpha_i + \beta_i \leq 4$ **then**
    increase $\alpha_i$ such that $(\alpha_i, \beta_i) \in \partial\mathcal{M}$.
  **else if** $\alpha_i > 3$ AND $\alpha_i + \beta_i \geq 4$ **then**
    decrease $\alpha_i$ such that $(\alpha_i, \beta_i) \in \partial\mathcal{M}$.
  **else if** $\alpha_i > 3$ AND $\alpha_i + \beta_i \leq 4$ **then**
    increase $\beta_i$ such that $(\alpha_i, \beta_i) \in \partial\mathcal{M}$.
  **end if**
**end if**
$f_i' = \alpha_i \Delta_i$
$f_{i+1}' = \beta_i \Delta_i$.

## 3.2 $\alpha \in [1, 2)$

### 3.2.1 Transforming the integral

Now the singularity is stronger than the previous case, and the derivative appearing in (3.1.1) can no longer be integrated out in order for the integral to converge. Now two integrations by part needs to be performed on the integral term from (2.1.6):

$$\int \nu(y)[u(Se^y, \tau) - u(S, \tau) - S(e^y - 1)u_S]dy,$$

first one obtains:

$$\begin{aligned} &\left[[u(Se^y) - u(S) - S(e^y - 1)u_S(S)](-\int_y^\infty \nu(z)dz)\right]_{y=0}^{y=\infty} \\ &- \int_0^\infty \partial_y[u(Se^y) - u(S) - S(e^y - 1)u_S](-\int_y^\infty \nu(z)dz)dy, \\ &= 0 + \int_0^\infty \partial_y[u(Se^y) - Se^y u_S] \int_y^\infty \nu(z)dzdy, \end{aligned} \tag{3.2.1}$$

the first term is still equal to zero by a taylor expansion and $u \in C^1$ with $M, G > 1$. Now on the second term an additional integration by parts is performed to obtain:

$$\begin{aligned} &\left[\partial_y[u(Se^y) - Se^y u_S](-\int_y^\infty \int_z^\infty \nu(w)dwdz)\right]_{y=0}^{y=\infty} \\ &- \int_0^\infty \partial_{yy}[u(Se^y) - Se^y u_S](-\int_y^\infty \int_z^\infty \nu(w)dwdz)dy. \end{aligned} \tag{3.2.2}$$

where the first part still is zero by a taylor expansion with $M, G > 1$, while the second part is the point of departure for introducing (3.1.9) and applying FFT. But first the procedure is also applied on the negative half axis, via the exact same integration by parts obtaining:

$$+ \int_{-\infty}^0 \partial_{yy}[u(Se^y) - Se^y u_S] \int_{-\infty}^y \int_{-\infty}^z \nu(w)dwdzdy \tag{3.2.3}$$

First to ease the notation, define the function:

$$\hat{\nu}(y) = \begin{cases} \int_y^\infty \int_z^\infty \nu(w)dwdz & y > 0 \\ \int_{-\infty}^y \int_{-\infty}^z \nu(w)dwdz & y < 0, \end{cases} \tag{3.2.4}$$

then consider the last term of the integral:

$$-Su_S[\int_{\mathbb{R}\setminus\{0\}} e^y \hat{\nu} dy] = -SV_S[\omega_2]. \tag{3.2.5}$$

As the quantity within the clamps can be precomputed, and stored in $\omega_2$, this quantity enters in the local operators, as in the previous case when $\alpha \in (0,1)$.

The only term which could potentially yield problems in computing this correlation integral is hence the first term in (3.2.2) and (3.2.3):

$$\int_{\mathbb{R}\setminus\{0\}} \partial_{yy}[u(Se^y)]\hat{\nu}(y)dy. \tag{3.2.6}$$

Also in this case the function (3.1.9) is introduced and the change of variable $S = e^x$, still with the motivation to exploit FFT in the evaluation. With the result:

$$\int_{\mathbb{R}\setminus\{0\}} \partial_{yy}\overline{u}(x+y)\hat{\nu}(y)dy \tag{3.2.7}$$

Now a way of computing $\partial_{yy}\overline{u}(x+y)$ needs to be addressed. By the same argumentation as to why computing $\partial_y\overline{u}(x+y)$ in the $\log S$-grid was a good idea, the same discussion applies to $\partial_{yy}\overline{u}(x+y)$. That quantity is computed in the $\log S$-grid via the central difference scheme:

$$\begin{aligned}
\delta_{yy}\overline{u}(x_i) &= \frac{\overline{u}(x_{i+1}) + \overline{u}(x_{i-1}) - 2\overline{u}(x_i)}{h_{log}^2} \\
&= \partial_{yy}\overline{u}(x_i) + Err_{disc} \\
&= \partial_{yy}\overline{u}(x_i) - \frac{1}{12}h_{log}^2\partial_{yyyy}\overline{u} + \frac{Err_{interp}}{h_{log}^2}.
\end{aligned}$$

With the 4th order monotone interpolation introduced earlier, this approximation is of 2nd order, although note the fact that unless monotone data are being employed, the 4th order interpolation gets degraded to 2nd order yielding error of $O(1)$, in that case. There is however a rather simple way out, just set $h_{log} = \sqrt{h}$, obtaining a 1st order scheme when non-monotone initial data are employed. That particluar case will not be pursued further, it is only presented as an option.

In this case, there is no need for having different approximations for the positive/negative half axis, as in the calculation of the first derivative, when forward/backward differences were being employed, which will be seen during the analysis. In addition, there is an important point to make regarding the computation of $\hat{\nu}(y)$. As $\nu(y)$ now is '1 degree more singular' than when $\alpha \in (0,1)$ the computation of $\hat{\nu}(y)$ is actually problematic. The particular implementation of the incomplete gamma function is not accurate enough in this case, with the result that a full double integral needs to be calculated for each $y$, which could have an impact on the overall runtime, especially when convergence tests are run. This will be addressed in (Section 3.2.3).

### 3.2.2 Discretizing the integral

In this case, as a 2nd order discretization of the derivative is readily available with the 4th order interpolation, the midpoint method will be used in the quadrature:

$$\begin{aligned} I(\overline{u}(x_i)) = I_i &= \sum_{j=-N_{ext},j\neq 0}^{N_{ext}} \delta_{yy}\overline{u}(x_{i+j}) \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \hat{\nu}(y)dy \\ &+ \delta_{yy}\overline{u}(x_i) \int_0^{\frac{h_{log}}{2}} \hat{\nu}(y)dy + \delta_{yy}\overline{u}(x_i) \int_{-\frac{h_{log}}{2}}^0 \hat{\nu}(y)dy \\ &= \sum_{j=-N_{ext}}^{N_{ext}} \delta_{yy}\overline{u}(x_{i+j})\overline{k}_j \end{aligned} \tag{3.2.8}$$

where:

$$\begin{aligned} \overline{k}_j &= \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \hat{\nu}(y)dy \qquad j \neq 0 \\ \overline{k}_j &= \int_0^{\frac{h_{log}}{2}} \hat{\nu}(y)dy + \int_{-\frac{h_{log}}{2}}^0 \hat{\nu}(y)dy \quad j = 0 \end{aligned} \tag{3.2.9}$$

with the integral truncated at $y = \pm h_{log}N_{ext}$, as in the previous case when $\alpha \in (0,1)$. With exponentially decaying $\hat{\nu}$ the error is quite easily controlled to be negligable.

The truncation error of the sum of all terms where $j \neq 0$ is quite easily seen to be of order 2 by a taylor expansion. Now when $j = 0$, the truncation error is in general $O(3-\alpha)$ on an asymmetric CGMY measure, as the twice integrated measure $\hat{\nu}(y)$ scales like

$$\hat{\nu}(y) \approx \frac{1}{y^{\alpha-1}}, |y| \ll 1, \tag{3.2.10}$$

hence yielding an $O(h_{log}^{3-\alpha})$ algorithm. However, with a symmetric measure, that is $M = G$, consider the first order truncation term (which are easily found by a taylorexpansion of $\partial_{yy}\overline{u}(x_{i+j}+\xi)$ around $\xi = 0$:

$$\partial_{yyy}\overline{u}(x_i)\left(\int_0^{\frac{h_{log}}{2}} \xi\hat{\nu}(y)dy + \int_{-\frac{h_{log}}{2}}^0 \xi\hat{\nu}(y)dy\right) = 0.$$

As when $M = G$, $\nu(y) = \nu(-y)$, and the first order term simply vanish, yielding an in fact 2nd order algorithm. When $G$ is different from $M$ convergence gets degraded to $O(h_{log}^{3-\alpha})$.

The actual computation with FFT in this case is based upon the same as that when $\alpha \in (0,1)$, with the exception that $\boldsymbol{k}$ gets modified, and the use of central differences

(3.2.8) rather than upwinding, in particular:

$$\begin{aligned} \boldsymbol{k}[j] = & \quad \overline{k}_j \quad 0 \leq j \leq N_{ext} \\ \boldsymbol{k}[Nlog + 2N_{ext} + j + 1] = & \quad \overline{k}_j \quad -N_{ext} \leq j \leq -1, \end{aligned} \tag{3.2.11}$$

and construct a vector $\delta_{yy}\overline{\boldsymbol{u}} \in \mathbb{R}^{N_{log}+2N_{ext}}$ by the similar interpolation as (Algorithm 1), but use central difference on the interpolated values rather than the upwinding. Then simply compute the correlation product (3.2.8) by:

$$FFT(I)_k = FFT(\delta_{yy}\overline{\boldsymbol{u}})_k FFT(\boldsymbol{k})^*_k, \tag{3.2.12}$$

compute the inverse FFT and interpolate back to the $S$-grid.

### 3.2.3 Computing the Levy measure

When it comes to computing the integrals (3.2.11) close to $y = 0$ problems arises, as the singularity is stronger than when $\alpha \in (0,1)$. Actually the algorithm used in that case, ie. a fast evaluation of the incomplete gamma function (3.1.7) is no longer accurate enough when $\alpha \geq 1$. Hence an other way of computing:

$$\int_0^{\frac{h_{log}}{2}} \hat{\nu}(y)dy + \int_{-\frac{h_{log}}{2}}^0 \hat{\nu}(y)dy, \tag{3.2.13}$$

needs to be adressed. (3.2.13) is actually a bit of a problem to compute numerically with quadrature. Just refining the grid to an extreme extent and thus 'brute-forcing' the answer does not help, as numerical oscillations due to the fixed machine precision appears.

Truncating the integral at a small $y$ does not work either as a lot of 'weight' of this measure is located at $|y| \ll 1$ due to the singularity. As the taylor expansion of $e^z$ for $|z| \ll 1$ is:

$$e^z = 1 + z + O(z^2), \tag{3.2.14}$$

an asymptotic approximation of (3.2.13) for $|y| \ll 1$ yields:

$$\begin{aligned} \int_0^\delta \hat{\nu}(y)dy + \int_{-\delta}^0 \hat{\nu}(y)dy &= \int_0^\delta \int_y^\infty \int_z^\infty \frac{C}{|w|^{1+\alpha}} dwdzdy \\ &\quad + \int_{-\delta}^0 \int_y^\infty \int_z^\infty \frac{C}{|w|^{1+\alpha}} dwdzdy + O(\delta^{3-\alpha}) \\ &= \frac{2C\delta^{2-\alpha}}{\alpha(\alpha-1)(2-\alpha)} + O(\delta^{3-\alpha}), \end{aligned} \tag{3.2.15}$$

with the modified truncation term $O(\delta^{3-\alpha})$ coming from the fact that:

$$\int_y^\infty \int_z^\infty \nu(w)dwdz$$

behaves like $\frac{1}{y^{\alpha-1}}$ for $y \ll 1$, and simply integrating up the error term (for general $M$ and $G$). A typical value for $\delta$ is in the magnitude $10^{-10}$. Hence the error of approximating this term for $y \ll 1$ is negligable in the error analysis. So in the end, the integral (3.2.13) is approximated by (3.2.15) for $0 < |y| \leq \delta$ and by a quadrature procedure $\delta < |y| < \frac{h_{log}}{2}$. The particular choice of quadrature is the simpsons method. However, certain measures needs to be taken to increase the speed of the computation. Because even though (3.2.11) only needs to be computed once, calculating a triple integral with such an irregular function as the CGMY measure with $\alpha \in [1, 2)$ is expensive. One option is of course to precalculate two of the integrals at a preset list of points, and store both the calculated values and the list of points in two vectors on the Hard Drive.

When the last integral is calculated, which needs to be done for each time the solution algorithm is run, as typically $h_{log}$ and $N_{ext}$ differs (otherwise the full triple integral could of course be precomputed and saved). These two vectors are loaded, and an interpolating polynomial is created ($PP$) by linear interpolation, yielding the integrand:

$$\begin{aligned}
\overline{k}_j = &\quad \tfrac{2C\delta^{2-\alpha}}{\alpha(\alpha-1)(2-\alpha)} + \textstyle\int_\delta^{\frac{h_{log}}{2}} PP(y)dy + \int_{-\frac{h_{log}}{2}}^{-\delta} PP(y)dy \quad j = 0 \\
\overline{k}_j = &\quad \textstyle\int_{h_{log}j-\frac{h_{log}}{2}}^{h_{log}i+\frac{h_{log}}{2}} PP(y)dy \qquad\qquad\qquad\qquad j \neq 0
\end{aligned}$$

Thus reducing the runtime significantly, as opposed to calculating $N_{log}$ triple integrals. Calculating this interpolating polynomial is typically done in 40 seconds, as will be shown in the results chapter, regardless of the gridresolution.

Now the particular distribution of points at which the interpolating polynomial, $PP$, is precalculated has to be adressed. Due to the strong singularity, simply dishing the points out equidistantly between $\delta$ and $\infty$ is of course not the solution.

By looking at the CGMY measure (2.1.7) displayed in (Figure 3.3) it is clear that a logarithmically equidistant placement of points captures the singularity in a better way than just dishing the points out equidistantly between $y_{min}$ and $y_{max}$. It places more points where the variations of $\nu(y)$ are largest, which is for small $y$. By computing two of the integrals at this set of points for a prescribed set of CGMY parameters the running time can be reduced significantly.

## 3.3 Speeding up the algorithm

The computational complexity, ie. $O(N \log N)$ pr. timestep is the fastest possible, and computing the integral term faster than with FFT is simply not obtainable. There are however a few tricks available to get a rather good speedup compared to a naive MATLAB

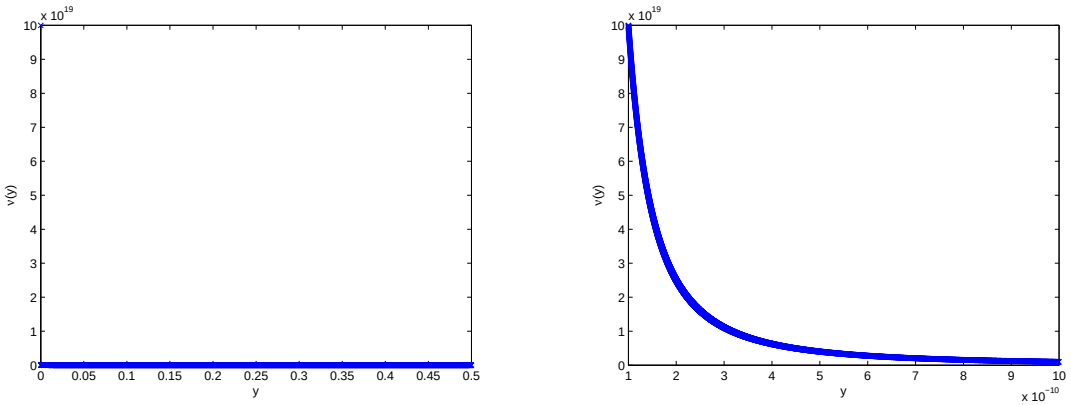


Figure 3.3: $\nu(y)$ for $y > 0$, $M = 2$, $\alpha = 1.5$. To the left 100000 gridpoints are placed equidistantly between $y_{min} = 10^{-10}$ and $y_{max} = 10$. The first 5000 nodes are displayed with a cross. To the right 100000 gridpoints are also placed between $y_{min} = 10^{-10}$ and $y_{max} = 10$ such that the points are placed logarithmically equidistant. In a similar manner as the figure to the left, only the first 5000 nodes are displayed. Clearly such a placement captures the singularity in a better manner than the approach to the left.

implementation. First of all by lowering the actual amount of floating point operations calculated, but also by lowering the time spent pr. FLOP. Perhaps the most obvious way is to precompute everything that stays constant in time throughout the solution loop. As $\theta = 0.5$ due to the Crank Nicholson discretization in time, the following is precomputed and stored in the new variables $I_1\hat{M}$, $I_2\hat{M}$, with $\hat{M}$ being the operator related to the local operators which will be introduced in the next subsection,

- $I_1\hat{M} = I + 0.5\hat{M}$
- $I_2\hat{M} = I - 0.5\hat{M}$. The FFT of $\boldsymbol{k}$ (3.1.28) or (3.2.11) is also created and stored before solving in time, plus different sets of index mapping needed in the interpolation.

The increased work pr. calculation of the integral term is limited to the above interpolation procedure, which is not that bad at all, considering the dominating term is the FFT and the inverse FFT, of order $O(N \log N)$. But the local check at each calculation point requires the use of a for-loop, which is not at all optimal when coding in MATLAB. Hence the MEX-file system in MATLAB is employed [10]. Writing a MEX wrap around function around your C++ code allows for function calls of C++ written code from within MATLAB. The easy and intuitive memory handling within MATLAB, combined with the raw speed of C++ when actually doing the calculations is a very powerful feature. Which both is fast in terms of flops, but also fast to implement.

Not all of the calculations are written in C++, as MATLAB has a huge library of compiled functions, which in most cases is hard to beat in terms of speed. However as those functions are very general and allows a wide diversity of inputs, highly customized code could reduce the runtime somewhat.

Although the C++/MEX framework really shines, compared to MATLAB, when it comes to for-loops. Hence especially the 4th order interpolation is a good candidate for a C++ implementation, as 1 such interpolation is performed each time the correlation product is calculated, the speedup obtained on that spesific part of the code were found to be around a factor 10.

## 3.4 Discretizing the local operators

First it is important to introduce the new PIDE:

$$u_\tau = (r-\omega)Su_S + \frac{\sigma^2 S^2}{2}u_{SS} - ru + \Omega(u). \tag{3.4.1}$$

with $\Omega(u)$ defined as:

$$\Omega(y) = \begin{cases} \int_0^\infty \partial_y[u(Se^y)](\overline{\nu}(y))dy - \int_{-\infty}^0 \partial_y[u(Se^y)](\overline{\nu}(y))dy & \alpha \in (0,1) \\ \int_{\mathbb{R}\setminus\{0\}} \partial_{yy}u(Se^y)\hat{\nu}(y)dy & \alpha \in [1,2) \end{cases} \tag{3.4.2}$$

and

$$\omega = \begin{cases} \omega_1 & \alpha \in (0,1) \\ \omega_2 & \alpha \in [1,2) \end{cases} \tag{3.4.3}$$

with $\omega_1$ from (3.1.3) and $\omega_2$ from (3.2.5).

When it comes to discretizing the local operators, finite differences are being used. It is important to assure global convergence which is achieved by Barles Souganidis if the scheme used is consistent, $L^\infty$-stable and monotone. Consistency follows trivially from the truncation error, which will be deduced later on, whereas stability in $L^\infty$ follows by the use of a positive approximation of both the partial derivatives, interpolation and integral plus a CFL which will be deduced later on.

As mentioned above, the use of a monotone approximation of the partial derivatives is important to ensure convergence, hence the second derivative is approximated by a central difference while the first derivative is chosen to be either forward/backward/-central difference depending on whether it locally is a monotone approximation or not. The local truncation error, at gridpoint $i$, of the central difference operator on the first derivative can be shown via taylor expansion to be, assuming constant steplength $h$:

$$u_S(S_i) = \frac{u(S_i+h) - u(S_i-h)}{2h} - u_{SSS}\frac{h^2}{6}, \tag{3.4.4}$$

whereas central difference applied to the second derivative yields the truncation error:

$$u_{SS}(S_i) = \frac{u(S_i+h) + u(S_i-h) - 2u(S_i)}{h^2} - \frac{h^2}{12}u_{SSSS}. \tag{3.4.5}$$

Forward/backward differences of the first derivative yields truncation errors:

$$u_S(S_i) = \frac{u(S_i+h) - u(S_i)}{h} - \frac{h}{2}u_{SS} \tag{3.4.6}$$

$$u_S(S_i) = \frac{-u(S_i - h) + u(S_i)}{h} + \frac{h}{2}u_{SS} \tag{3.4.7}$$

ie only first order. However, as soon shall be revealed, the limited need for this discretization does not lower the total convergence of the algorithm below second order, typically only a few nodes in the grid employ this approximation. In particular the use of first order discretization will only happen if:

$$\sigma^2 S_i \leq h|r - \omega|, \tag{3.4.8}$$

as $S_i = ih$:

$$\sigma^2 i \leq |r - \omega|, \tag{3.4.9}$$

which only might happen at a few nodes close to $S = 0$, as $r$ and $\omega$ are small in magnitude. For small $S$ the solution is in fact very well approximated by a linear function and convergence tests show the limited use of first order upwinding does in fact not degrade convergence.

The mentioned discretizations of the first and second derivatives plus a discretization in time by crank nicholson yields the following *local* equation system (3.4.10) (that is without the integral operator which will be added later on):

$$(I - \frac{\Delta t}{2}\hat{M})u^{n+1} = (I + \frac{\Delta t}{2}\hat{M})u^n \tag{3.4.10}$$

I is the identity-matrix, $\Delta t$ the timestep and $\hat{M}$ is given by:

$$[\hat{M}u^n]_i = -u_i^n(\alpha_i + \beta_i + r) + \beta_i u_{i+1}^n + \alpha_i u_{i-1}^n. \tag{3.4.11}$$

$\alpha_i$ is chosen locally to be one of $\alpha_{i,central}$, $\alpha_{i,forward}$ and $\alpha_{i,backward}$ by the positive approximation criterion, similarily with $\beta_i$. Where the $\alpha$'s and $\beta$'s are defined by:

$$\begin{aligned}
\alpha_{i,central} &= \frac{\sigma_i^2 S_i^2}{h^2} - \frac{(r-\omega)S_i}{2h} \\
\beta_{i,central} &= \frac{\sigma_i^2 S_i^2}{h^2} + \frac{(r-\omega)S_i}{2h} \\
\alpha_{i,forward} &= \frac{\sigma_i^2 S_i^2}{h^2} \\
\beta_{i,forward} &= \frac{\sigma_i^2 S_i^2}{h^2} + \frac{(r-\omega)S_i}{2h} \\
\alpha_{i,backward} &= \frac{\sigma_i^2 S_i^2}{h^2} - \frac{(r-\omega)S_i}{2h} \\
\beta_{i,backward} &= \frac{\sigma_i^2 S_i^2}{h^2}.
\end{aligned} \tag{3.4.12}$$

In all of these approximations central differences are used in the approximation of the second derivative. Whereas the name (central, forward and backward) relates to the type of discretisation of the first derivative. The choice on which approximation that are used on the node $i$ are according to (Algorithm 3).

**Algorithm 3** Modifying the approximation of local derivatives to ensure monotonicity

1: **if** $\alpha_i \geq 0$ AND $\beta_i \geq 0$ **then**
2: $\alpha_i = \alpha_{i,central}$
3: $\beta_i = \beta_{i,central}$
4: **else if** $\beta_{i,forward} \geq 0$ **then**
5: $\alpha_i = \alpha_{i,forward}$
6: $\beta_i = \beta_{i,forward}$
7: **else**
8: $\alpha_i = \alpha_{i,backward}$
9: $\beta_i = \beta_{i,backward}$
10: **end if**

This choice ensures a monotone approximation as of the former discussion.

## 3.5 Discretization in time

As mentioned Crank Nicolson is used in the integration in time, so for each timestep an implicit equation-system needs to be solved, with the discretized derivatives presented in the previous subsection,:

$$\begin{aligned} u^{n+1} &= u^n + \frac{\Delta t}{2}(\hat{M}u^n + \hat{M}u^{n+1} + \Omega(u^n) + \Omega(u^{n+1}) \\ &\Downarrow \\ u^{n+1} &= [I - \frac{\Delta t}{2}\hat{M}]^{-1}([I + \frac{\Delta t}{2}\hat{M}]u^n + \frac{\Delta t}{2}\Omega(u^{n+1}) + \frac{\Delta t}{2}\Omega(u^n)) \end{aligned} \tag{3.5.1}$$

with $\Delta t$ being the timestep. The equation system might seem odd, as $u^{n+1}$ is on the right hand side. The correlation product is however only computed explicitly, which will be shown in the next subsection how to solve (3.5.1) for each timestep.

## 3.6 Solving the system of equations

As mentioned, Crank Nicolson will be used, but still I have mentioned that the correlation product *only* is computed explicitly to avoid solving a full system each timestep. So how will this be done? The answer is to solve the system by a fixed point iteration, as in [6], where $\Omega(u^n)$ is the calculated correlation product. Note that during this fixed point iteration, the correlation is only calculated for known values $u^k$.

**Algorithm 4** The fixed point iteration

Let $(u^{n+1})^0 = u^n$, and $\hat{u}^k = (u^{n+1})^k$
**for** $k = 0, 1, 2, ...$ until conv criterion reached **do**
  $u^{k+1} = [I - \frac{\Delta t}{2}\hat{M}]^{-1}([I + \frac{\Delta t}{2}\hat{M}]u^n + \frac{\Delta t}{2}\Omega(\hat{u}^k) + \frac{\Delta t}{2}\Omega(u^n))$
  **if** $||\hat{u}^{k+1} - \hat{u}^k||_\infty < TOL$ **then**
    break.
  **end if**
**end for**

Convergence is however conditional, although by the CFL condition, which will be deduced in the analysis chapter, convergence is assured in a *low* number of iterations, essentially scaling like $O(1)$ if the CFL condition is employed. Thus giving a total work pr. iteration $O(N \log N)$ (inverting the tridiagonal matrix is done in $O(N)$ thus the calculation of the correlation product dominates).

The convergence results is however slightly worse than in [6], where unconditional convergence is achieved, as of course can be expected due to the singular CGMY measure used here, rather than the regular probability distribution assumed by [6].

## 3.7 Truncation error

As is explained in the previous section, the truncation error consists of:

1. Approximating the first/second derivatives. Discussion of this truncation error is given in the subsection regarding discretization of the local operator, however under some assumptions, the result is:

$$E_{discLocal} = u_{SS}K_1h^2 + u_{SSSS}K_2h^2 \tag{3.7.1}$$

2. Interpolating to the logarithmic grid and back again. The 4th order interpolation introduced in a previous subsection is used so the error is on the form:

$$E_{interp} = O(h^4) + O(h_{log}^2) \tag{3.7.2}$$

3. Approximating the integral (3.1.24) has error:

$$E_{integral} = K_3\overline{u}_{xxx}h_{log}^{2-\alpha} \quad \alpha \in (0,1) \tag{3.7.3}$$

Approximating the integral when $\alpha \in [1,2)$ on the other hand, with 2nd order differences and 2nd order quadrature is used:

$$E_{integral} = K_4\overline{u}_{xxxx}h_{log}^{3-\alpha} \quad \alpha \in [1,2) \tag{3.7.4}$$

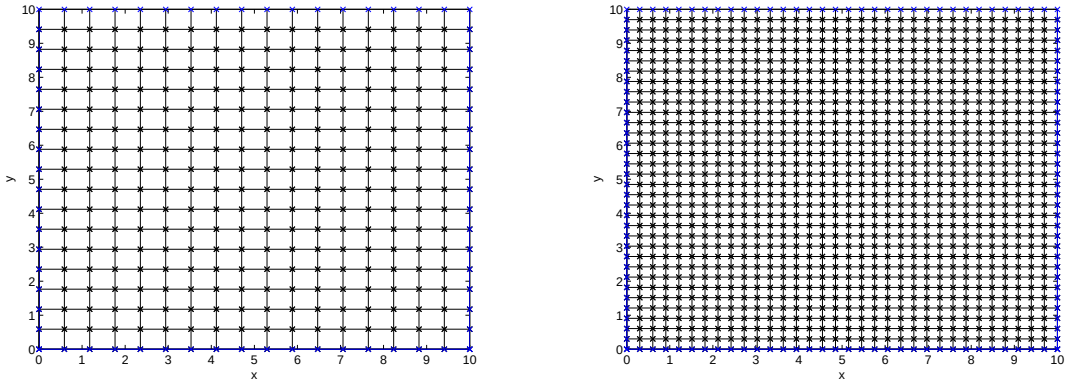


Figure 3.4: The computational Grid in the 2-Dimensional case. Blue line is where BC are set, black crosses are the unknowns. Left: 16 nodes in each spatial direction yielding a total of 256 unknowns. Right: 32 nodes in each spatial direction yielding a total of 1024 unknowns.

Although on a symmetrical measure convergence is improved to order 2.

4. The error of truncating the integral at $y = -h_{log}N_{ext}$ and $y = h_{log}N_{ext}$ can be made arbitrarily small, by simply increasing the number of expanded values, as the levy measure used decays exponentially, and as $\nu(|y|) < e^{-\min(M,G)|y|}, |y| > 1$:

5. The error of using Crank-Nicholson:

$$E_{crank} = K_5 u_{ttt}(\Delta t)^2 \tag{3.7.5}$$

with $\Delta t$ introduced earlier as the size of the timestep. Such that with bounded derivatives up until $u_{SSSS}$ in $S$ and to $u_{tt}$ in $t$ yields truncation errors of $O(h^2 + (\Delta t)^2 + h_{log}^{3-\alpha})$ for $\alpha \in [1, 2)$, and $O(h^2 + (\Delta t)^2 + h_{log}^{2-\alpha})$ for $\alpha \in (0, 1)$

## 3.8 Discretizing the 2D Option Pricing Problem

To test the efficiency of the algorithm we consider the 2D problem introduced in (Chapter 2). Throughout the article a completely equidistant grid is used on the 2D Option pricing problem (2.1.8), having different steplengths in the $x$ and $y$ direction really is not a problem, but to keep the notation to basics that will not be done. $x_{max}$ is set equal to $y_{max}$, such that the grid is turned into a perfect square (Figure 3.4).

In addition $\rho$ will be set equal to zero. Such that the 2D problem will really be like solving the 1D problem in each direction.

First the global numbering is introduced, where the numbering is such that increasing $x$ means increasing global number, $l$. So if the point $x_j = jh, y_i = ih$ is numbered $l$, then the point $x_{j+1}, y_i$ is numbered $l+1$, and $x_j, y_{i+1}$ is numbered $l+N$. Now when assembling

the matrix operator for the local derivatives, it is important to keep in mind that these derivatives, which involve closest neighbours in the $x$- and $y$-direction, are affected by the dirichlet boundary conditions for the nodes where $x_i = h$, $y_i = h$, $x_i = Nh$ and $y_i = Nh$. The two latter are where the BC are set to zero and can thus be disregarded in the further discussion, however on the two first lines, the dirichlet boundary conditions needs to be incorporated in the equation system. In particular that is for the global indices:

$$\begin{array}{rrr} l = i & 1 \leq i \leq N & y_i = h \\ l = 1 + iN & 0 \leq i \leq N-1 & x_i = h \end{array} \tag{3.8.1}$$

So going from the local node numbering, where point $(i,j)$ corresponds to $x = ih$ and $y = jh$, is illustrated when considering the global vector, where local indices are for clarity:

$$u = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ . \\ . \\ u_{N,1} \\ u_{1,2} \\ u_{2,2} \\ . \\ . \\ u_{N,N-1} \\ u_{N,N} \end{bmatrix} \tag{3.8.2}$$

The particular assembly of the matrix assiciated with the local operators is based on the kronecker tensor product, $\otimes$, where the result is a large matrix formed by taking all possible products between the two different matrices. If for example $B$ is a two by two matrix, and C is a matrix of arbitrary size:

$$B \otimes C = \begin{bmatrix} B(1,1)C & B(1,2)C \\ B(2,1)C & B(2,2)C \end{bmatrix} \tag{3.8.3}$$

So by taking the kronecker tensor product between two $N$ by $N$ matrices, the result is a $N^2$ by $N^2$ matrix.

Based upon section (3.4), the following matrices are assembled in the two spatial coordinate directions $x$ and $y$, but remember the $\alpha's$ and $\beta's$ might be slightly different for $A_x$ and $A_y$, as $\sigma_x$, $\sigma_y$ and the parameters in the two CGMY distributions might differ. Hence $\alpha_{ix}$, $\alpha_{iy}$, $\beta_{ix}$ and $\beta_{iy}$ should really be used while describing the framework. When describing $A_x$ it is however clear that $\alpha_i$ and $\beta_i$ implicitly means $\alpha_{ix}$ and $\beta_{ix}$. Likewise for $A_y$.

$$A_x = \begin{bmatrix} \alpha_1 & \beta_1 & & & & & & 0 \\ \alpha_2 & \alpha_2 & \beta_2 & & & & & \\ & \alpha_3 & \alpha_3 & \beta_3 & & & & \\ & & \alpha_4 & \alpha_4 & \beta_4 & & & \\ & & & & \ddots & & & \\ & & & & \alpha_{N-2} & \alpha_{N-2} & \beta_{N-2} & \\ & & & & & \alpha_{N-1} & \alpha_{N-1} & \beta_{N-1} \\ 0 & & & & & & \alpha_N & \alpha_N, \end{bmatrix} \tag{3.8.4}$$

$$A_y = A_x - rI_N, \tag{3.8.5}$$

with $I_N$ being the identity matrix with size $N$ by $N$, in the last equation the $rI_N$ term has to be subtracted to avoid adding the $ru$ term (2.1.8) twice.

Now with the kronecker tensor product introduced, the full operator for approximating the local terms are simply:

$$A_{2D} = A_y \otimes I_N + I_N \otimes A_x, \tag{3.8.6}$$

yielding a block tridiagonal matrix, where $\alpha_{i,j} = A_x(i,i) + A_y(j,j)$:

$$A_{2D} = \begin{bmatrix} \alpha_{11} & \beta_{1x} & & \beta_{1y} & & & & & & \\ \alpha_{2x} & \alpha_{21} & \beta_{2x} & & \beta_{1y} & & & & & \\ & & \ddots & & & \ddots & & & & \\ \alpha_{2y} & & 0 & \alpha_{12} & \beta_{1x} & & \beta_{2y} & & & \\ & \alpha_{2y} & & \alpha_{2x} & \alpha_{22} & \beta_{2x} & & \beta_{2y} & & \\ & & \ddots & & & \ddots & & & \ddots & \\ & & & \alpha_{N_{-1}y} & & \alpha_{Nx} & \alpha_{NN_{-1}} & 0 & & \beta_{N_{-1}y} \\ & & & & \alpha_{N_y} & & 0 & \alpha_{1N} & \beta_{1x} & \\ & & & & & \ddots & & & \ddots & \\ & & & & & & \alpha_{Ny} & & \alpha_{Nx} & \alpha_{NN} \end{bmatrix}. \tag{3.8.7}$$

Presenting the non-zero entries in the above matrix while still maintaining clarity is a bit of a problem. Although the two bands next to the diagonal corresponds to the derivatives in the x direction. The zeros at the entries $\alpha_{1x}$ (left to the diagonal) and $\beta_{Nx}$ (right to the diagonal) which appear with period $N$, are due to application of dirichlet boundary conditions at $x = Nh$ and $x = h$.

The other two off-diagonal bands appear with distance $N$ from the diagonal and correspond to the derivatives in the $y$-direction. As can be seen, the dirichlet boundary conditions are naturally applied at $\alpha_{1y}$ and $\beta_{Ny}$ corresponding to $y = h$ and $y = Nh$.

The boundary vector $b$ resulting from applying BC is more tricky in this case than in the 1 dimensional case. Where that vector simply had 1 non zero entry and due to its simplicity only were implicitly incorporated within $\hat{M}$, $2N - 2$ non-zero entries is the

result here, and due to the introduction of global indices needs to be explicitly addressed. First remember that along the boundary lines $x = h$ and $y = h$, the 1D problem needs to be solved and used as BC. Say $u_{x1D}$ is the solution of the 1D problem in the $x$-direction and $u_{y1D}$ the solution in the $y$-direction. These could potentially be different, as the parameters in the CGMY distribution might differ as well as the volatilities ($\sigma_x$ and $\sigma_y$). Then the $b$ turns out to be:

$$u = \begin{bmatrix} \alpha_{1x}u_{y1D}(1) + \alpha_{1y}u_{x1D}(1) \\ \alpha_{1y}u_{x1D}(2) \\ \alpha_{1y}u_{x1D}(3) \\ \vdots \\ \alpha_{1y}u_{x1D}(N) \\ \alpha_{1x}u_{y1D}(2) \\ 0 \\ \vdots \\ 0 \\ \vdots \\ \alpha_{1x}u_{y1D}(3) \\ 0 \\ \vdots \\ \alpha_{Nx}u_{y1D}(N) \end{bmatrix} \tag{3.8.8}$$

With uncorrelated jumpterms the correlation integrals simply reduces to 1-Dimensional integrals, which have allready been shown how to discretize and solve. The only problem (again) is to keep track of the indices.

The particular integration in time is also in the 2D case handled implicitly, but at a cost. Solving the band-triagonal system ($N$ by $N$ matrix with band-length $N$) has complexity $O(N^3)$ with lu-factorization. Computing $2N^2$ correlation products (two correlation products for each unknown, one in the $x$-direction and one in the $y$-direction) has complexity $O(N^2 \log N)$, hence the total complexity of the algorithm is $O(MN^3)$, assuming the fixed point iteration converges in $O(1)$, which is the case if the CFL condition is honored. The CFL condition for the total algorithm is that $\Delta t$ should be refined proportionally to $h^2$ to obtain monotonicity, hence the total complexity for the algorithm is $O(N^5)$. If an explicit method is employed, the CFL condition will still asymptotically stay the same, and there will be no need to solve the equation system for each timestep. The complexity will in that case be lowered to $O(N^2 \log N)$ pr. timestep, and $O(N^4 \log N)$ in total. Such that there seems to be much to gain by dropping the operator splitting approach and go all out explicit. The operator splitting approach is however the choice in this case, to keep the 1D and 2D case similar, but note the fact that an explicit scheme will be more efficient.

# Chapter 4

# Analysis

As hinted in the previous section, a key issue is the use of a monotone discretization which is important in the subsequent analysis. In particular, let $L_h u(x)$ be a discretized operator, a monotone approximation of that operator is:

$$L_h u(x) = \sum_n c_n(u(x+nh) - u(x)) \qquad c_n \geq 0 \tag{4.0.1}$$

A monotone approximation of an integral simply requires the weights to be greater than or equal to zero. Monotone approximation of the local operators follow by construction. Monotone interpolation requires that if the function is positive, the interpolated function also is positive, which in any case is ok for linear interpolation. In addition the 4th order monotone interpolation used to obtain 2nd order convergence, is also monotone by [9] on monotone data.

## 4.1 A monotone approximation of the integral operator, $\alpha \in (0,1)$

First consider the first order discretization of the integral when $\alpha \in (0,1)$ (3.1.24) on the $\log S$-grid.

$$I_i = \left[ \sum_{j=0}^{j=N_{ext}} \delta_{y+} \overline{u}_{i+j} \overline{k}_j^+ - \sum_{j=-N_{ext}}^{j=0} \delta_{y-} \overline{u}_{i+j} \overline{k}_j^- \right] \tag{4.1.1}$$

Consider the first term of the sum first:

$$\begin{aligned}
\sum_{j=0}^{j=N_{ext}} \delta_{y+}\overline{u}_{i+j}\overline{k}_j^+ &= \sum_{j=0}^{j=N_{ext}} \frac{\overline{u}_{i+j+1}-\overline{u}_{i+j}}{h_{log}}\overline{k}_j^+ \\
&= \sum_{j=1}^{j=N_{ext}} \frac{\overline{u}_{i+j}}{h_{log}}(\overline{k}_{j-1}^+ - \overline{k}_j^+) - \frac{\overline{u}_i}{h_{log}}\overline{k}_0^+ + u_{i+N_{ext}+1}\frac{\overline{k}_{N_{ext}}}{h_{log}} \\
&= \sum_{j=1}^{j=N_{ext}} (\overline{u}_{i+j}-\overline{u}_i)(\tilde{k}_{j-1}^+ - \tilde{k}_j^+) + (u_{i+N_{ext}+1}-u_i)\tilde{k}_{N_{ext}}
\end{aligned}$$

with $\tilde{k}_j^+ = \frac{\overline{k}_j}{h_{log}}$. Now in the following the last term will be omitted in the subsequent analysis, as $\tilde{k}_{N_{ext}} \to 0$ to keep the notation more compact. Via a similar prosedure on the negative half axis, where backward differences are being employed:

$$\begin{aligned}
-\sum_{j=-N_{ext}}^{j=0} \delta_{y-}\overline{u}_{i+j}\overline{k}_j^- &= -\sum_{j=-N_{ext}}^{j=0} \frac{\overline{u}_{i+j}-\overline{u}_{i+j-1}}{h_{log}}\overline{k}_j^- \\
&= \sum_{j=-N_{ext}}^{j=0} \frac{\overline{u}_{i+j-1}-\overline{u}_{i+j}}{h_{log}}\overline{k}_j^- \\
&= \sum_{j=-N_{ext}}^{j=-1} \frac{\overline{u}_{i+j}}{h_{log}}(\overline{k}_{j+1}^- - \overline{k}_j^-) - \frac{\overline{u}_i}{h_{log}}\overline{k}_0^- + \frac{\overline{u}_{i-N_{ext}-1}}{h_{log}}\overline{k}_{-N_{ext}}^- \\
&= \sum_{j=-N_{ext}}^{j=-1} (\overline{u}_{i+j}-\overline{u}_i)(\tilde{k}_{j+1}^- - \tilde{k}_j^-) + (\overline{u}_{i-N_{ext}-1}-\overline{u}_i)\tilde{k}_{-N_{ext}}^-,
\end{aligned}$$

with $\tilde{k}_j^- = \frac{\overline{k}_j^-}{h_{log}}$. Also in this case the last term will be omitted in the subsequent analysis, as $\overline{k}_{-N_{ext}}^- \to 0$ to keep the notation more compact, it will not affect monotonicity as $\tilde{k}_{-N_{ext}} > 0$ and $\tilde{k}_{N_{ext}} > 0$ Now in order for the discretization to be monotone:

$$\begin{aligned}
\overline{k}_{j+1}^- &\geq \overline{k}_j^- \qquad -N_{ext} \leq j \leq -1 \\
\overline{k}_{j-1}^+ &\geq \overline{k}_j^+ \qquad 1 \leq j \leq N_{ext},
\end{aligned} \tag{4.1.2}$$

which is ok by the properties of the Levy measure (on the positive half axis $\overline{\nu}$ is a decreasing function of $y$, on the negative half axis $\overline{\nu}$ is a decreasing function of $-y$, and the length of each integral $\overline{k}_j^+$ and $\overline{k}_j^-$ is the same $\forall j$ ($h_{log}$), hence (4.1.2) holds.

## 4.2 A monotone approximation of the integral operator, $\alpha \in [1,2)$

For the 2nd order discretization of the second derivative term (3.2.7), the $\overline{k}$-vector is constructed as in (3.2.11), and the calculation is like:

$$\begin{aligned}
I_i &= \sum_{j=-N_{ext}}^{j=N_{ext}} \delta_{yy}\overline{u}_{i+j}\overline{k}_j = \sum_{j=-N_{ext}}^{j=N_{ext}} \frac{-2\overline{u}_{i+j}+\overline{u}_{i+j-1}+\overline{u}_{i+j+1}}{h_{log}^2}\overline{k}_j \\
&= \sum_{j=-N_{ext},j\neq 0,1,-1}^{j=N_{ext}} \frac{-2\overline{u}_{i+j}+\overline{u}_{i+j-1}+\overline{u}_{i+j+1}}{h_{log}^2}\overline{k}_j - \frac{\overline{u}_i}{h_{log}^2}(2\overline{k}_0-\overline{k}_1-\overline{k}_{-1}) \\
&+ \frac{\overline{u}_{i+1}}{h_{log}^2}(\overline{k}_0-2\overline{k}_1)+\frac{\overline{u}_{i-1}}{h_{log}^2}(\overline{k}_0-2\overline{k}_{-1})+\frac{\overline{u}_{i-2}}{h_{log}^2}\overline{k}_{-1}+\frac{\overline{u}_{i+2}}{h_{log}^2}\overline{k}_1 \\
&\approx \sum_{j=-N_{ext}+1,j\neq 0}^{j=N_{ext}-1} \frac{\overline{u}_{i+j}}{h_{log}^2}(-2\overline{k}_j+\overline{k}_{j-1}+\overline{k}_{j+1}) - \frac{\overline{u}_i}{h_{log}^2}(2\overline{k}_0-\overline{k}_1-\overline{k}_{-1}) \\
&\approx \sum_{j=-N_{ext}+1,j\neq 0}^{j=N_{ext}-1} (\overline{u}_{i+j}-\overline{u}_i)(\tilde{k}_{j-1}-2\tilde{k}_j+\tilde{k}_{j+1})
\end{aligned}$$

with $\tilde{k}_j = \frac{\overline{k}_j}{h_{log}^2}$. The $\approx$ signs come from the fact that the terms involving $\tilde{k}_{\pm N_{ext}}$ are omitted from this analysis for abbreviation as:

$$\begin{aligned}
\overline{k}_{N_{ext}} &\to 0 \\
\overline{k}_{-N_{ext}} &\to 0.
\end{aligned}$$

If $N_{ext}$ is chosen such that $\tilde{k}_{\pm N_{ext}}$ is less than $10^{-15}$ (the accuracy of double precision integers), which easily can be done due to the exponentially decreasing measure, the above derivation is exact when computing numerically on a computer.

Now the discretization is monotone if

$$\overline{k}_{j-1}-2\overline{k}_j+\overline{k}_{j+1} \geq 0 \quad \forall j \neq 0 \tag{4.2.1}$$

which is ok for $|j|>1$, as $\overline{k}$ then is monotone for fixed $y$. First as $\hat{\nu}(y)$ is a convex function (twicely integrated positive function) consider:

$$\begin{aligned}
&\int_{(j-1)h_{log}-\frac{h_{log}}{2}}^{(j-1)h_{log}-\frac{h_{log}}{2}} \hat{\nu}(y)dy - 2\int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}-\frac{h_{log}}{2}} \hat{\nu}(y)dy + \int_{(j+1)h_{log}-\frac{h_{log}}{2}}^{(j+1)h_{log}-\frac{h_{log}}{2}} \hat{\nu}(y)dy \\
&\qquad = \int_{jh_{log}-\frac{h_{log}}{2}}^{jh_{log}+\frac{h_{log}}{2}} \hat{\nu}(y-h_{log})-2\hat{\nu}(y)+\hat{\nu}(y+h_{log})dy
\end{aligned}$$

The integrand is greater than 0 for fixed $y$ as it is convex, hence obtaning (4.2.1).

The calculation is a tad more tricky when $j = \pm 1$ for $M \neq G$, but asymptotically monotonicity is achieved also in this case for $\alpha > 1$, as regardless of the exponents $M$ and $G$, asymptotically the integrated measure at $|y| \ll 1$ will have the same scaling on either side of $y = 0$, as $\alpha$ is the same for $y > 0$ and $y < 0$.

So the integral operator is a monotone approximation on the logarithmic grid. The local operators is a monotone approximation on the $S$-grid, and a monotone interpolation is used to interpolate back and forth between the grids. The question now being; is the combined discretization a monotone approximation on the $S$-grid?

And the answer is, unfortunately, no. Because consider (4.0.1), and the fact that the discrete gridpoints in the log-S grid and the $S$-grid does in fact not coincide. By introducing the index-mapping between the two grids, following the notation by [6]:

$$S_{\Upsilon(j)} \leq e^{x_j} < S_{\Upsilon(j)+1} \tag{4.2.2}$$

Where $x_j$ were defined at (3.1.11). So the index-function $\Upsilon(j)$ denotes the value in the $S$-grid being closest (while still being smaller than or equal) to the value at index $j$ in the $\log S$-grid. And the mapping the other way around:

$$e^{x_{\Pi(i)}} \leq S_i < e^{x_{\Pi(i)+1}} \tag{4.2.3}$$

where $e^{x_{\Pi(i)}}$ is the value closest (while still being smaller than or equal) to the value at index $i$ in the $S$-grid. Hence, interpolating the function $\overline{u}$ onto $u(S_i)$ with linear interpolation can be expressed as:

$$u(S_i) = \theta \overline{u}(x_{\Pi(i)}) + (1-\theta)\overline{u}(x_{\Pi(i)+1}) \tag{4.2.4}$$

with $\theta$ the interpolation weight, and interpolating the correlation $I(\overline{u}(x_j))$ onto $I(u(S_i))$ can be expressed as (with only the integral on the positive half axis to avoid too much notation):

$$I(u(S_i)) = \theta \sum_{j=1}^{j=N_{ext}} (\overline{u}_{\Pi(i)+j} - \overline{u}_{\Pi(i)})(\tilde{k}^+_{j-1} - \tilde{k}^+_j) + (1-\theta) \sum_{j=1}^{j=N_{ext}} (\overline{u}_{\Pi(i)+j+1} - \overline{u}_{\Pi(i)+1})(\tilde{k}^+_{j-1} - \tilde{k}^+_j) \tag{4.2.5}$$

now again to express $\overline{u}_{\Pi(i)+1}$ in terms of $u$ (with linear interpolation):

$$\overline{u}_{\Pi(i)+1} = \Phi u(S_{\Upsilon(\Pi(i)+1)}) + (1-\Phi)u(S_{\Upsilon(\Pi(i)+1)+1}) \tag{4.2.6}$$

with $\Phi$ the new interpolation weights. Similarily with $\overline{u}_{\Pi(i)}$. It is quite clear that when interpolating the sum in the $\log S$-grid onto the $S$-grid, the correlation product can no longer be written as (4.0.1), hence the approximation is not monotone in the $S$-grid.

The solution, if a monotone discretization on this particular problem is wanted, is to transform the problem as explained in the introduction, transforming everything to logarithmic variables, even the local operators. Hence getting the entire problem on the same $\log S$-grid, and not needing the interpolation back and forth.

The resulting PIDE and scheme is presented below:

## 4.3 An equivalent, but easier, PIDE

To obtain an equivalent, but easier, PIDE, transform the entire PIDE with the transformation $x = log(S)$, and using the function $v(x,t) = v(log(S),t)$. First observe that:

$$\begin{aligned} v_S(x,t) &= \frac{1}{S}v_x(x,t) \\ v_{SS}(x,t) &= \frac{1}{S^2}v_{xx}(x,t) - \frac{1}{S^2}v_x(x,t) \end{aligned} \tag{4.3.1}$$

obtaining ([5] plus a partial integration of the integralterm):

$$u_\tau = (r - \omega - \frac{\sigma^2}{2})v_x + \frac{\sigma^2}{2}v_{xx} - rv + \Omega(v). \tag{4.3.2}$$

where $\Omega(v)$ is defined as:

$$\Omega(y) = \begin{cases} \int_0^\infty \partial_y[v(x+y)](\overline{\nu}(y))dy - \int_{-\infty}^0 \partial_y[v(x+y)](\overline{\nu}(y))dy & \alpha \in (0,1) \\ \int_{\mathbb{R}\setminus\{0\}} \partial_{yy} v(x+y)\hat{\nu}(y)dy & \alpha \in [1,2), \end{cases} \tag{4.3.3}$$

and $\omega$ is defined in (3.4.3). In the remaining parts of the Analysis section just let $u = v$ to ease the notation.

Equation (4.3.2) has everything on the same grid, and interpolation is hence not necessary. Proving monotonicity and stability on this particular equation is easy. First of all the local derivatives get slightly modified, but it is trivial to see what the new $\alpha_i$ and $\beta_i$ turns into, just apply the same central difference on the second derivative and the choice between forward/backward and central on the first derivative as in (Algorithm 3). In the following assume $\beta_i$ and $\alpha_i$ are modified to be consistent approximations to (4.3.2) so the total scheme on (4.3.2) turns out to be:

$$\begin{aligned} & u_i^{k+1}[1 + \frac{\Delta t}{2}(\alpha_i + \beta_i + r)] - \frac{\Delta t}{2}\beta_i u_{i+1}^{k+1} - \frac{\Delta t}{2}\alpha_i u_{i-1}^{k+1} \\ = & \frac{\Delta t}{2}\Omega_h(u^k) + \frac{\Delta t}{2}\Omega_h(u^{k+1}) + \frac{\Delta t}{2}\beta_i u_{i+1}^k + u_i^k[1 - \frac{\Delta t}{2}(\alpha_i + \beta_i + r)] + \frac{\Delta t}{2}\alpha_i u_{i-1}^k \end{aligned} \tag{4.3.4}$$

With $\Omega_h$ being the discretized integral operator (3.1.24 if $\alpha \in (0,1)$) or (3.2.8 if $\alpha \in [1,2)$). This scheme will be far easier to analyze than the interpolation back and forth. The subsequent analysis will be done on the above scheme, with $h_{log}$ being the stepsize in both the local and the global operators.

## 4.4 Proving convergence of the fixed point iteration

Following the outlines from [6], the only difference would be the singular Levy measure, which could and do actually impose restrictions on the ratio between time and space

stepping to ensure convergence. It is easily seen from algorithm (4) that the error $e^k = u^{n+1} - \hat{u}^k$, with $\theta = \frac{1}{2}$ with the new scheme (4.3.4) satisfies:

$$\begin{aligned} e_i^{k+1}[1 + \frac{\Delta t}{2}(\alpha_i + \beta_i + r)] - \frac{\Delta t}{2}\beta_i e_{i+1}^{k+1} - \frac{\Delta t}{2}\alpha_i e_{i-1}^{k+1} &= \frac{\Delta t}{2}\Omega_h(e^k) \\ &= \frac{\Delta t}{2}\sum_{j=1}^{j=N_{ext}} \frac{e_{i+j}^k - e_i^k}{h_{log}}(\overline{k}_{j-1}^+ - \overline{k}_j^+) \\ &+ \frac{\Delta t}{2}\sum_{j=-N_{ext}}^{j=-1} \frac{e_{i+j}^k - e_i^k}{h_{log}}(\overline{k}_{j+1}^- - \overline{k}_j^-) \end{aligned} \quad (4.4.1)$$

Defining $||e||_\infty^{k+1} = \max_i |e_i|^{k+1}$:

$$\begin{aligned} ||e||_\infty^{n+1}[1 + \frac{\Delta t}{2}(\alpha_i + \beta_i + r)] &\leq \frac{\Delta t}{2}\beta_i|e_{i+1}^{k+1}| + \frac{\Delta t}{2}\alpha_i|e_{i-1}^{k+1}| + |\frac{\Delta t}{2}\Omega(e_i^k)| \\ &\leq \frac{\Delta t}{2}\beta_i||e||_\infty^{n+1} + \frac{\Delta t}{2}\alpha_i||e||_\infty^{n+1} + |\frac{\Delta t}{2}\Omega_h(e_i^k)| \\ &\Downarrow \\ ||e||_\infty^{n+1}[1 + \frac{r\Delta t}{2}] &\leq |\frac{\Delta t}{2}\Omega(e_i^k)| \\ &= \frac{\Delta t}{2}\left[\sum_{j=1}^{j=\frac{N_{log}}{2}-1} \frac{|e_{i+j}^k|}{h_{log}}(\overline{k}_{j-1}^+ - \overline{k}_j^+) + \frac{|e_i^k|}{h_{log}}\overline{k}_0^+\right] \\ &+ \frac{\Delta t}{2}\left[\sum_{j=-\frac{Nlog}{2}}^{j=0} \frac{|\overline{e}_{i+j}^k|}{h_{log}}(\overline{k}_j^- - \overline{k}_{j-1}^-) + \frac{|\overline{e}_i^k|}{h_{log}}\right] \\ &\leq \frac{\Delta t}{2}\left[\frac{||e||_\infty^k}{h_{log}}\sum_{j=1}^{j=\frac{N_{log}}{2}-1}(\overline{k}_{j-1} - \overline{k}_j) + \frac{||e||_\infty^k}{h_{log}}\overline{k}_0^+\right] \\ &+ \frac{\Delta t}{2}\left[\frac{||e||_\infty^k}{h_{log}}\sum_{j=-\frac{Nlog}{2}}^{j=0}(\overline{k}_j^- - \overline{k}_{j-1}^-) + \frac{||e||_\infty^k}{h_{log}}\overline{k}_0^-\right] \\ &= \frac{\Delta t 2||e||_\infty^k}{h_{log}}(\overline{k}_0^- + \overline{k}_0^+) \\ &\Downarrow \\ ||e||_\infty^{k+1} &\leq \frac{\Delta t||e||_\infty^k}{h_{log}(1 + \frac{r\Delta t}{2})}(\overline{k}_0^+ + \overline{k}_0^-) \end{aligned}$$

Hence as $\frac{\overline{k}_0^+}{h_{log}}$ and $\frac{\overline{k}_0^-}{h_{log}}$ scales like $h_{log}^\alpha$, convergence is asymptotically assured if $\Delta t$ gets refined proportionally to $h_{log}^\alpha$.

When $\alpha \in [1,2)$ one achieves through a similar procedure as the above, asymptotic scaling of $\Delta t$ like $h_{log}^{\alpha}$ to ensure convergence of the fixed point iteration.

These conditions on $\Delta t$ with respect to $h$ might seem like a problem, but they are nevertheless honored due to the CFL condition in order to keep the scheme monotone, where $\Delta t$ has to get refined proportionally to $h_{log}^2$ which is deduced in the following subsection.

## 4.5 CFL condition

Now as every discretized operator is proven to be monotone, the full log-scheme (4.3.4) can be shown to be monotone under a CFL condition. The monotonicity of the scheme is important to ensure convergence via Barles-Perthame-Souganidis, which heuristically is the $||\cdot||_\infty$ equivalent of the Lax equivalence theorem in $||\cdot||_2$. If the scheme is consistent, monotone (of positive type) and stable in $||\cdot||_\infty$ convergence is acheived by an adaptiation of the Barles-Perthame-Souganidis procedure [2].

In order for a scheme to be monotone (of positive type) there exists coefficients $b \geq 0$ under a CFL such that:

$$b_{i,i}^{n+1} u_i^{n+1} - \sum_{j \neq i} b_{i,j}^{n+1} u_j^{n+1} - \sum_j b_{i,j}^n u_j^n = 0 \tag{4.5.1}$$

By the discretized scheme (4.3.4) we see that:

$$\begin{aligned}
b_{i,i}^{n+1} &= 1 + \tfrac{\Delta t}{2}(\alpha_i + \beta_i + r + \tilde{k}_0^+ + \tilde{k}_0^-) \\
b_{i,i}^{n} &= 1 - \tfrac{\Delta t}{2}(\alpha_i + \beta_i + r + \tilde{k}_0^+ + \overline{k}_0^-) \\
b_{i,i+1}^{n+1} &= \tfrac{\Delta t}{2}(\alpha_i + \tilde{k}_0^+ - \tilde{k}_1^-) \\
b_{i,i-1}^{n+1} &= \tfrac{\Delta t}{2}(\beta_i + \tilde{k}_0^- - \tilde{k}_1^-) \\
b_{i,i+j}^{n+1} &= \tfrac{\Delta t}{2}(\tilde{k}_{j+1}^- - \tilde{k}_j^-) \qquad j \leq -2 \\
b_{i,i+j}^{n+1} &= \tfrac{\Delta t}{2}(\tilde{k}_{j-1}^+ - \tilde{k}_j^+) \qquad 2 \leq j \\
b_{i,i-1}^{n} &= \tfrac{\Delta t}{2}(\beta_i + \tilde{k}_0^+ - \tilde{k}_{-1}^-) \\
b_{i,i+1}^{n} &= \tfrac{\Delta t}{2}(\alpha_i + \tilde{k}_0^+ - \tilde{k}_1^+) \\
b_{i,i+j}^{n} &= \tfrac{\Delta t}{2}(\tilde{k}_{j+1}^- - \tilde{k}_j^-) \qquad j \leq -2 \\
b_{i,i+j}^{n} &= \tfrac{\Delta t}{2}(\tilde{k}_{j-1}^+ - \tilde{k}_j^+) \qquad 2 \leq j
\end{aligned} \tag{4.5.2}$$

Where $\alpha_i$ and $\beta_i$ is introduced earlier as discretization of the new local operator in the log-variable.Now the only term which could possibly be less than zero is $b_{i,i}^n$. Monotonicity is achieved if:

$$\Delta t \leq \frac{2}{\alpha_i + \beta_i + r + \tilde{k}_0^+ + \tilde{k}_0^-} \tag{4.5.3}$$

With $\alpha_i$ $+\beta_i$ scaling like $O(h_{log}^{-2})$, $r$ being $O(1)$ and $\tilde{k}_0^-$ and $\tilde{k}_0^+$ scaling like $O(h_{log}^{\alpha})$. Monotonicity is asymptotically achieved if:

$$\Delta t \leq \frac{K}{h_{log}^2}, \tag{4.5.4}$$

with $K$ a constant. In a fully similar manner the CFL conditions for $\alpha \in [1,2)$ are also shown to be:

$$\Delta t \leq \frac{2}{\alpha_i + \beta_i + r + 2\overline{k}_0 - \overline{k}_1 - \overline{k}_{-1}}, \tag{4.5.5}$$

with the same asymptotic scaling as (4.5.4).

A very interesting fact is, as hinted in (Chapter 2), that the brownian term can be removed from the process since the dynamics of the jumps allready is rich enough to generate nontrivial small time behaviour. With the brownian term removed, the second derivative in the local operators will also vanish, and the CFL turns out to be that when $\alpha \in (0,1)$, $\Delta t$ should be refined proportionally to $h_{log}$ asymptotically. And when $\alpha \in [1,2)$, $\Delta t$ should be refined proportionally to $h_{log}^{\alpha}$. This approach is not pursued here, but will be discussed in the last chapter.

## 4.6 Stability in $L^\infty$

Consider the scheme when $\alpha \in (0,1)$:

First note that:

$$\begin{aligned}\sum_{j\neq i} b_{i,j}^{n+1} &= \frac{\Delta t}{2}\left[\alpha_i + \beta_i + \sum_{j=1}^{j=N_{ext}} (\tilde{k}_{j-1}^+ - \tilde{k}_j^+) + \sum_{j=-N_{ext}}^{j=-1} (\tilde{k}_{j+1}^- - \tilde{k}_j^-))\right] \\ &= \frac{\Delta t}{2}(\alpha_i + \beta_i + \tilde{k}_0^+ + \tilde{k}_0^-) = b_{i,i}^{n+1} - 1 - \frac{r\Delta t}{2}\end{aligned} \tag{4.6.1}$$

and

$$\begin{aligned}\sum_{j} b_{i,j}^{n} &= \frac{\Delta t}{2}(\alpha_i + \beta_i + \sum_{j=1}^{j=N_{ext}} (\tilde{k}_{j-1}^+ - \tilde{k}_j^+) + \sum_{j=-N_{ext}}^{j=-1} (\tilde{k}_{j+1}^- - \tilde{k}_j^-)) + b_{i,i}^{n} \\ &= \frac{\Delta t}{2}(\alpha_i + \beta_i + \tilde{k}_0^+ + \tilde{k}_0^-) + 1 - \frac{\Delta t}{2}(\alpha_i + \beta_i + \tilde{k}_0^+ + \tilde{k}_0^- + r) \\ &= 1 - \frac{r\Delta t}{2}\end{aligned} \tag{4.6.2}$$

then consider from (4.5.1):

$$\begin{aligned} b_{i,i}^{n+1}u_i^{n+1} &= \sum_{j\neq i} b_{i,j}^{n+1}u_j^{n+1} + \sum_j b_{i,j}^n u_j^n \\ &\Downarrow \\ b_{i,i}^{n+1}|u_i^{n+1}| &\leq \sum_{j\neq i} b_{i,j}^{n+1}|u_j^{n+1}| + \sum_j b_{i,j}^n |u_j^n| \end{aligned} \tag{4.6.3}$$

Defining $||e||_\infty^{k+1} = \max_i |e_i^{k+1}|$:

$$\begin{aligned} b_{i,i}^{n+1}\|u^{n+1}\|_\infty^{n+1} &\leq \sum_{j\neq i} b_{i,j}^{n+1}|u_j^{n+1}| + \sum_j b_{i,j}^n|u_j^n| \\ &\leq \|u\|_\infty^{n+1}\sum_{j\neq i} b_{i,j}^{n+1} + \|u\|_\infty^n \sum_j b_{i,j}^n \\ &= \|u\|_\infty^{n+1}(-1-\frac{r\Delta t}{2}+b_{i,i}^{n+1}) + \|u\|_\infty^n \sum_j b_{i,j}^n \\ &= \|u\|_\infty^{n+1}(-1-\frac{r\Delta t}{2}+b_{i,i}^{n+1}) + \|u\|_\infty^n(1-\frac{\Delta t r}{2}) \\ &\Downarrow \\ \|u^{n+1}\|_\infty^{n+1} &\leq \frac{1-\frac{\Delta t r}{2}}{1+\frac{\Delta t r}{2}}\|u\|_\infty^n \\ &\leq \left(\frac{1-\frac{\Delta t r}{2}}{1+\frac{\Delta t r}{2}}\right)^2 \|u\|_\infty^{n-1} \\ &\vdots \\ &\leq \left(\frac{1-\frac{\Delta t r}{2}}{1+\frac{\Delta t r}{2}}\right)^{n+1} \|u\|_\infty^0 \end{aligned} \tag{4.6.4}$$

Hence stability in $L^\infty$ is obtained, and the modified PIDE is assured to converge by a Barles-Perthame Souganidis procedure [2].

The case when $\alpha \in [1,2)$ can also be proved by (4.6.4) as (4.6.1) and (4.6.2) also holds in that case with positive coefficients under a CFL.

# Chapter 5

# Results

What is important is to verify each individual part of the code and then the algorithm as a whole to reduce the probability of having a wrong implementation, and verify the theory. First convergence tests are run on the two different algorithms for approximating the global operator, when $\alpha \in (0, 1)$ and $\alpha \in [1, 2)$. Then convergence tests are run on the full PIDE, both when $\alpha \in (0, 1)$ and $\alpha \in [1, 2)$. In the end, solutions for different values of the parameters are displayed and discussed. Throughout this chapter the parameters are set to (Table 5.1), otherwise it will explicitly be stated.

| | | |
|---|---|---|
| $C$ | = | 0.1 |
| $G$ | = | 5 |
| $M$ | = | 5 |
| $\sigma$ | = | 0.15 |
| $r$ | = | 0.05 |
| $T$ | = | 0.25 |
| $K$ | = | 3 |
| $S_{max}$ | = | 15 |
| $S_0$ | = | 0, |

Table 5.1: Parameters when approximating the equation.

As exact solutions are not available, it is not possible to verify the complete correctness of the algorithm as a whole. I can only show that the algorithm approaches a solution with the correct rate, i.e. numerical convergence.

On the algorithms for approximating the integral terms, when $\alpha \in (0, 1)$ and $\alpha \in [1, 2)$, it is possible to compute the integrals directly with a high level of accuracy with quadrature, and use these computed integrals as exact solutions. It is the integral terms that are exciting and new, and hence emphasis in the convergence analysis will be given on them. The particular function which will be tested when calculating the correlation-product is a solution 'looking like' the option price. In order to have enough derivatives available it will be the solution:

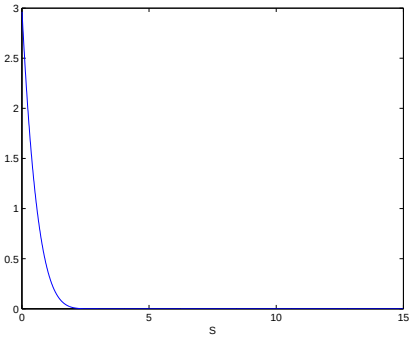


Figure 5.1: The solution used to calculate the correlationproduct.

$$u(x,t) = \max\left(\frac{(K-x)^5}{K^4}, 0\right), \tag{5.0.1}$$

plotted in (Figure 5.1).

When considering the convergence results, first consider the fact that it is anticipated that the numerical error $e = u_{exact} - u_h$, where $u_{exact}$ is the exact solution, and $u_h$ the numerically calculated solution with stepsize $h$, should behave like:

$$e = Kh^R \tag{5.0.2}$$

with $K$ just a constant and $R$ the convergence rate. Now consider having two different solutions with different stepsizes:

$$\begin{aligned} e_1 &= Kh_1^R \\ e_2 &= Kh_2^R \\ &\Downarrow \\ log(e_1) &= log(K) + Rlog(h_1) \\ log(e_2) &= log(K) + Rlog(h_2) \\ &\Downarrow \\ R &= \frac{log(e_1) - log(e_2)}{log(h_1) - log(h_2)} \end{aligned} \tag{5.0.3}$$

## 5.1 The integral term, $\alpha \in (0,1)$

The theoretical result is that the approximation of the integral term when $\alpha \in (0,1)$ should be of order $2-\alpha$. From the convergence tests run, which are plotted in (Figure 5.2)

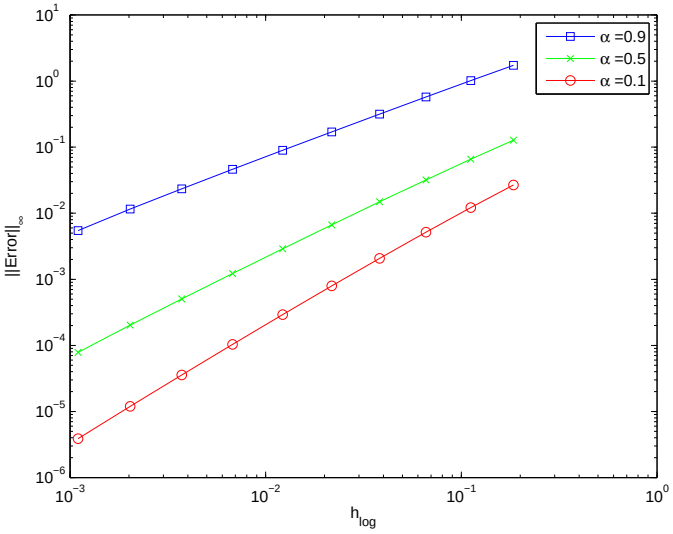


Figure 5.2: Convergence in $L^\infty$ on the correlation product alone, $\alpha = 0.9$ (blue), $\alpha = 0.5$ (green), $\alpha = 0.1$ (red). $M = G = 5$ in all three plots

and described in (Table 5.2), this $\alpha$-dependent convergence is verified by the numerical implementation. There is however a tendency of increasing rate as the grid gets refined, and it would be interesting to see if the increasing rate continues, reaching order 2 asymptotically, but the runtime of the quadrature reference solution currently limits this approach.

The calculated correlation (on the function 5.0.1) is also plotted for increasing value of $\alpha$ and for fixed $\alpha$ with asymmetric $M$ and $G$ (Figure 5.3). It is the entire integral term:

$$\int_{\mathbb{R}\setminus\{0\}} [u(Se^y) - u(S) - S(e^y - 1)\frac{\partial u}{\partial S}]\nu(y)dy,$$

that is depicted, and not just the part calculated via FFT. This is done to be able to compare the correlationproducts when $\alpha \in (0,1)$ and $\alpha \in [1,2)$, and to see what the total contribution to the Black Scholes PDE in reality turns out to be (remember with $\nu(y) = 0$, the Black Scholes PDE is obtained). As seems to be the case, there is a strictly positive contribution, which should lead to increased option values. The integral also seems to be an increasing function of $\alpha$, which heuristically amounts to higher risk.

The exponents $M$ and $G$ determines the tail behaviour, such that for small $M$, the probability of a large upwards movement is big, and for small $G$ the probability of a large downward movement is big. Hence increasing $M$ and $G$ decreases risk and vice versa. This behaviour is confirmed in (Figure 5.4), where increasing $M$ and $G$ decreases the contribution from the integral term.

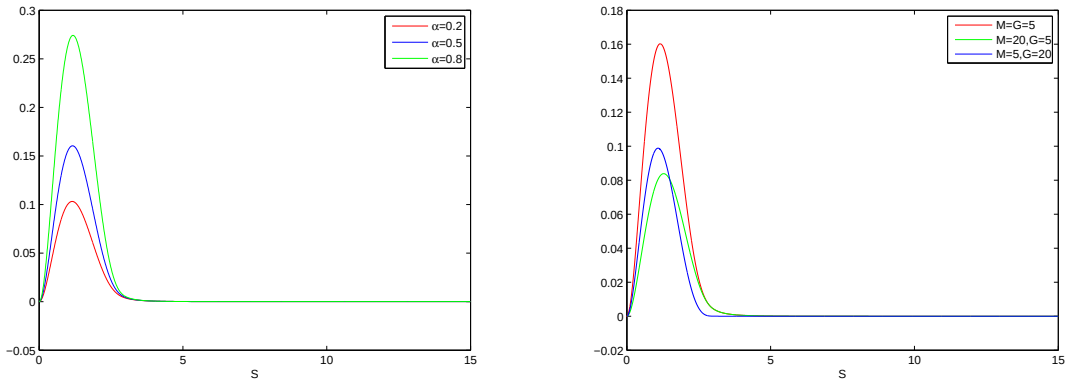


Figure 5.3: The correlationproduct calculated for different values of the parameters while $\alpha \in (0, 1)$. Left: $\alpha = 0.2$ (red), $\alpha = 0.5$ (blue), $\alpha = 0.8$ (green) $C = 1$, $G = M = 5$ in all three plots. Right: $\alpha = 0.5$ and $C = 1$ in all three plots. $M = G = 5$ (red), $M = 5$, $G = 20$ (blue) $G = 20$, $M = 5$ (green).

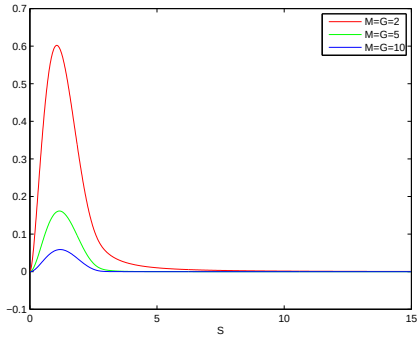


Figure 5.4: The integral term plotted for increasing exponents, $M = G = 2$ (red), $M = G = 5$ (green), $M = G = 10$ (blue). $\alpha = .5$, $C = 1$ in all three plots

| $h_{log}$ | $\|E_{\alpha=0.1}\|_\infty$ | $R_{\alpha=0.1}$ | $\|E_{\alpha=0.5}\|_\infty$ | $R_{\alpha=0.5}$ | $\|E_{\alpha=0.9}\|_\infty$ | $R_{\alpha=0.9}$ |
|---|---|---|---|---|---|---|
| 1.8483e-01 | 2.6750e-02 | N/A | 1.2743e-01 | N/A | 1.7290e+00 | N/A |
| 1.1179e-01 | 1.2153e-02 | 1.5690 | 6.5611e-02 | 1.3203 | 1.0140e+00 | 1.0614 |
| 6.6014e-02 | 5.1740e-03 | 1.6209 | 3.1988e-02 | 1.3636 | 5.7416e-01 | 1.0796 |
| 3.8204e-02 | 2.0785e-03 | 1.6675 | 1.4911e-02 | 1.3955 | 3.1675e-01 | 1.0875 |
| 2.1745e-02 | 7.9475e-04 | 1.7060 | 6.6765e-03 | 1.4259 | 1.7023e-01 | 1.1018 |
| 1.2208e-02 | 2.9176e-04 | 1.7357 | 2.8956e-03 | 1.4469 | 8.9620e-02 | 1.1113 |
| 6.7756e-03 | 1.0356e-04 | 1.7591 | 1.2231e-03 | 1.4637 | 4.6298e-02 | 1.1218 |
| 3.7247e-03 | 3.5698e-05 | 1.7801 | 5.0448e-04 | 1.4801 | 2.3439e-02 | 1.1376 |
| 2.0312e-03 | 1.1961e-05 | 1.8032 | 2.0282e-04 | 1.5026 | 1.1554e-02 | 1.1664 |
| 1.1000e-03 | 3.8689e-06 | 1.8405 | 7.8574e-05 | 1.5464 | 5.4505e-03 | 1.2252 |

Table 5.2: Convergence of the integral approximation when $\alpha = 0.1$,$\alpha = 0.5$, $\alpha = 0.9$. $C = 1$,$M = G = 5$ for all values of $\alpha$.

## 5.2 The integral term, $\alpha \in [1,2)$

Now in this case the 2nd order convergence which were anticipated on a symmetric measure is clearly verified by the numerical experiments (Figure 5.5) and (Table 5.3), although the constant in front of the error seems to be increasing for increasing $\alpha$.

The convergence gets degraded when $M \neq G$, but the experiments indicate convergence does not get fully degraded to $O(h_{log}^{3-\alpha})$ (Table 5.4) and (Figure 5.6), and also in this case the rate seems to be increasing as the grid gets refined.

The correlation is plotted for different values of $\alpha$, $M$ and $G$ (Figure 5.7), where again the entire integral is calculated, not just the part calculated via FFT. In addition a plot where $\alpha = 0.999$ is plotted vs $\alpha = 1.001$, indicating the transition from $\alpha \in (0,1)$ to $\alpha \in [1,2)$ is in fact smooth (Figure 5.8).

| $h_{log}$ | $\|E_{\alpha=1.1}\|_\infty$ | $R_{\alpha=1.1}$ | $\|E_{\alpha=1.5}\|_\infty$ | $R_{\alpha=1.5}$ | $\|E_{\alpha=1.9}\|_\infty$ | $R_{\alpha=1.9}$ |
|---|---|---|---|---|---|---|
| 6.6014e-02 | 3.0643e-03 | N/A | 1.0651e-02 | N/A | 9.5271e-02 | N/A |
| 3.8204e-02 | 8.2566e-04 | 2.3978 | 2.7267e-03 | 2.4915 | 2.2668e-02 | 2.6252 |
| 2.1745e-02 | 2.5559e-04 | 2.0807 | 7.9840e-04 | 2.1795 | 6.1547e-03 | 2.3134 |
| 1.2208e-02 | 1.2070e-04 | 1.2995 | 4.1653e-04 | 1.1270 | 3.5354e-03 | 0.9602 |
| 6.7756e-03 | 3.6883e-05 | 2.0137 | 1.2772e-04 | 2.0077 | 1.0869e-03 | 2.0032 |
| 3.7247e-03 | 1.1368e-05 | 1.9670 | 3.9694e-05 | 1.9532 | 3.3585e-04 | 1.9629 |
| 2.0312e-03 | 3.3871e-06 | 1.9968 | 1.1877e-05 | 1.9897 | 1.0045e-04 | 1.9905 |
| 1.1000e-03 | 9.8858e-07 | 2.0081 | 3.4820e-06 | 2.0009 | 2.9533e-05 | 1.9962 |

Table 5.3: Convergence of the integral approximation, $\alpha = 1.1$,$\alpha = 1.5$, $\alpha = 1.9$. $C = 1$, $M = G = 5$ for all values of $\alpha$.

| $h_{log}$ | $\|E_{\alpha=1.1}\|_\infty$ | $R_{\alpha=1.1}$ | $\|E_{\alpha=1.5}\|_\infty$ | $R_{\alpha=1.5}$ | $\|E_{\alpha=1.9}\|_\infty$ | $R_{\alpha=1.9}$ |
|---|---|---|---|---|---|---|
| 6.6014e-02 | 2.1727e-03 | N/A | 6.9380e-03 | N/A | 6.0945e-02 | N/A |
| 3.8204e-02 | 8.6229e-04 | 1.6898 | 3.0304e-03 | 1.5145 | 2.0667e-02 | 1.9773 |
| 2.1745e-02 | 3.2604e-04 | 1.7257 | 1.3227e-03 | 1.4710 | 8.2072e-03 | 1.6388 |
| 1.2208e-02 | 1.2151e-04 | 1.7095 | 5.6321e-04 | 1.4789 | 3.7193e-03 | 1.3709 |
| 6.7756e-03 | 4.1424e-05 | 1.8279 | 2.2274e-04 | 1.5755 | 1.6454e-03 | 1.3851 |
| 3.7247e-03 | 1.3468e-05 | 1.8777 | 8.2009e-05 | 1.6699 | 6.9420e-04 | 1.4423 |
| 2.0312e-03 | 4.2918e-06 | 1.8860 | 2.8193e-05 | 1.7608 | 2.7296e-04 | 1.5393 |
| 1.1000e-03 | 1.2926e-06 | 1.9569 | 9.1360e-06 | 1.8376 | 1.0009e-04 | 1.6359 |

Table 5.4: Convergence of the integral approximation, $\alpha = 1.1$,$\alpha = 1.5$, $\alpha = 1.9$. $C = 1$, $M = 5$, $G = 20$ for all values of $\alpha$

## 5.3 The full PIDE

On the full PIDE, the CFL condition deduced to obtain monotonicity on the simpler problem (4.3.2) will be used on the 'harder' problem implemented, even though it will not be monotone anyway, it will give a nice picture of the computational complexity. With the CFL that the timestep should scale like the stepping in space squared the computational complexity turns out to be $O(N^3 \log N)$, hence by increasing the number of nodes in the grid by a factor two, the runtime would increase by a factor slightly over 8. The particular runtime is displayed in (5.5) for $\alpha \in (0, 1)$ and (5.6) for $\alpha \in [1, 2)$ with $T_{pre}$ denoting the time to preprocess, ie. constructing all the vectors that remains constant in time, and the particular index mapping. The time to preprocess is clearly higher in the case when $\alpha \in [1, 2)$ due to the fact that the interpolating polynomial is created for each time the algorithm is run. This is of course not necessary in the convergence test, as the parameters stay fixed, but is nevertheless performed to highlight this fact, which in general will be the cost of running this algorithm.

$T_{runtime}$ denotes the time taken to run the full solution loop. As can be seen, the scaling slowly approaches the factor 8 in both cases, as can be expected when doubling the number of nodes in the grid for each iteration.

The full solution of the option pricing problem is also plotted vs the reference Black Scholes solution (Figure 5.10) and (Figure 5.11). As can be expected, as the correlation integral seems to be increasing with $\alpha$, the option price also seems to be an increasing function of $\alpha$, which heuristically is to be expected, as high values of $\alpha$ amounts to higher values of risk.

In the end the solution of the 2 dimensional problem is also solved and plotted in (Figure 5.12), where half of the grid is removed before the plotting, as the function will only be zero at these points anyway. The solution of the 2 dimensional problem, with computational complexity of $O(N^5)$ with lu-factorization and a CFL condition as in the 1D case, hits unfeasible problem sizes very fast. Alternative aproaches, that could lower the complexity and make 2 dimensional problems more feasible, are discussed in the last chapter.

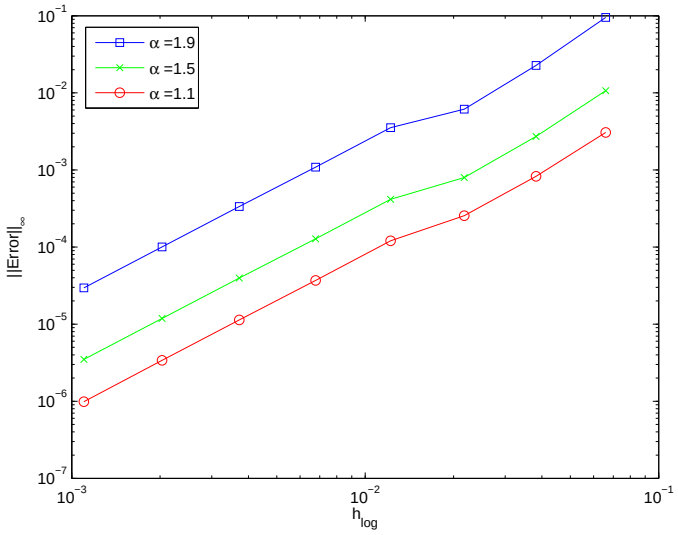


Figure 5.5: Convergence in $L^\infty$ on the correlation product alone, $\alpha = 1.9$ (blue), $\alpha = 1.5$ (green), $\alpha = 1.1$ (red). $M = G = 5$ in all three plots

| $h_{log}$ | $\|Error\|_\infty$ | $R$ | $T_{preproc}$ (s) | $T_{runtime}$ (s) |
|---|---|---|---|---|
| 1.5651e-01 | 8.6996e-03 | N/A | 0.8 | 0.9 |
| 8.8018e-02 | 5.0329e-03 | 0.9507 | 0.8 | 1.0 |
| 4.9120e-02 | 2.6598e-03 | 1.0933 | 0.9 | 1.6 |
| 2.7182e-02 | 1.2022e-03 | 1.3419 | 1.0 | 4.4 |
| 1.4920e-02 | 5.4286e-04 | 1.3256 | 1.3 | 15.3 |
| 8.1307e-03 | 2.2255e-04 | 1.4687 | 1.9 | 84.5 |
| 4.4020e-03 | 8.2319e-05 | 1.6208 | 3.2 | 560.9 |

Table 5.5: Convergence of the full algorithm, $\alpha = 0.5$

| $h_{log}$ | $\|Error\|_\infty$ | $R$ | $T_{preproc}$ (s) | $T_{runtime}$ (s) |
|---|---|---|---|---|
| 1.565171e-01 | 1.724151e-02 | N/A | 41.1 | 0.07 |
| 8.801869e-02 | 7.936607e-03 | 1.347832 | 41.2 | 0.3 |
| 4.912067e-02 | 2.641209e-03 | 1.886348 | 41.3 | 1.3 |
| 2.718224e-02 | 8.276612e-04 | 1.961056 | 41.4 | 7.5 |
| 1.492098e-02 | 2.447882e-04 | 2.031042 | 41.6 | 49.1 |
| 8.130759e-03 | 7.160766e-05 | 2.024647 | 42.3 | 332.1 |
| 4.402009e-03 | 1.996176e-05 | 2.081809 | 43.0 | 1954.4 |

Table 5.6: Convergence of the full algorithm, $\alpha = 1.5$

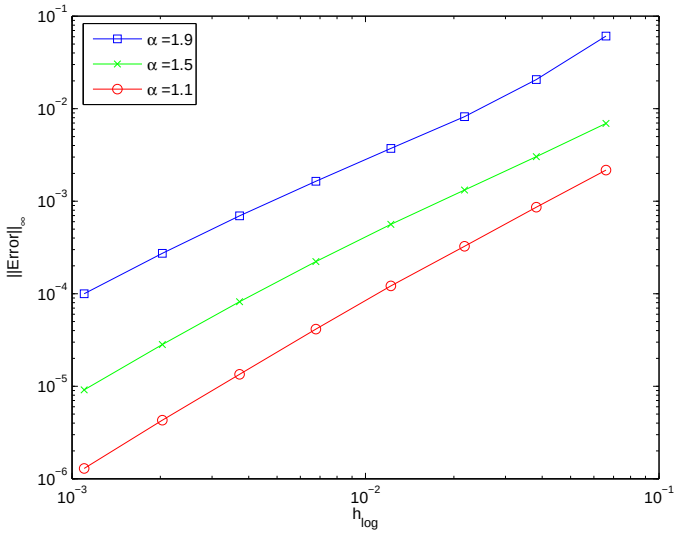


Figure 5.6: Error in $\|\cdot\|_\infty$ $\alpha \in [1,2)$ $M = 20, G = 5$

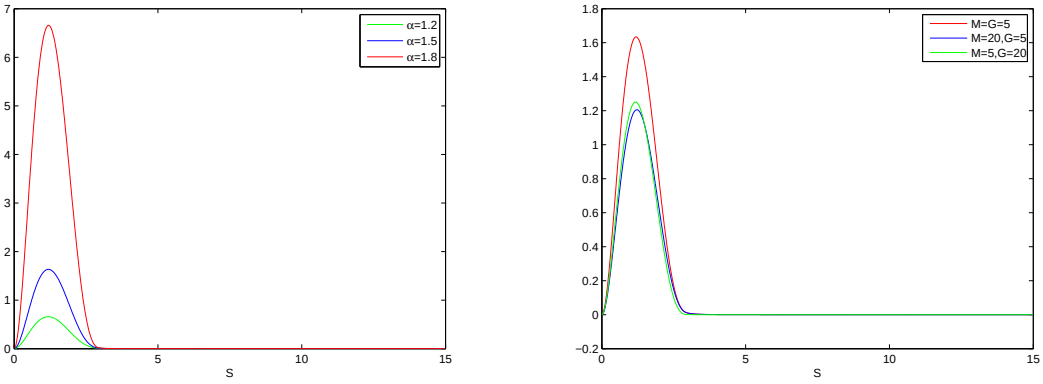


Figure 5.7: The correlationproduct calculated for different values of the parameters while $\alpha \in [1,2)$. Left: $\alpha = 1.2$ (green), $\alpha = 1.5$ (blue), $\alpha = 1.8$ (red) $C = 1$, $G = M = 5$ in all three plots. Right: $\alpha = 1.5$ and $C = 1$ in all three plots. $M = G = 5$ (red), $M = 20$, $G = 5$ (blue) $G = 5$, $M = 20$ (green).

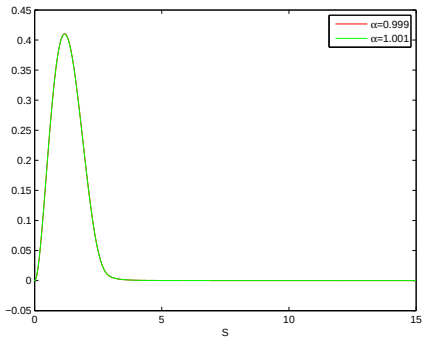


Figure 5.8: Correlation plotted for $\alpha = 0.999$ and $\alpha = 1.001$ indicating the transition from $\alpha \in (0, 1)$ to $\alpha \in [1, 2)$

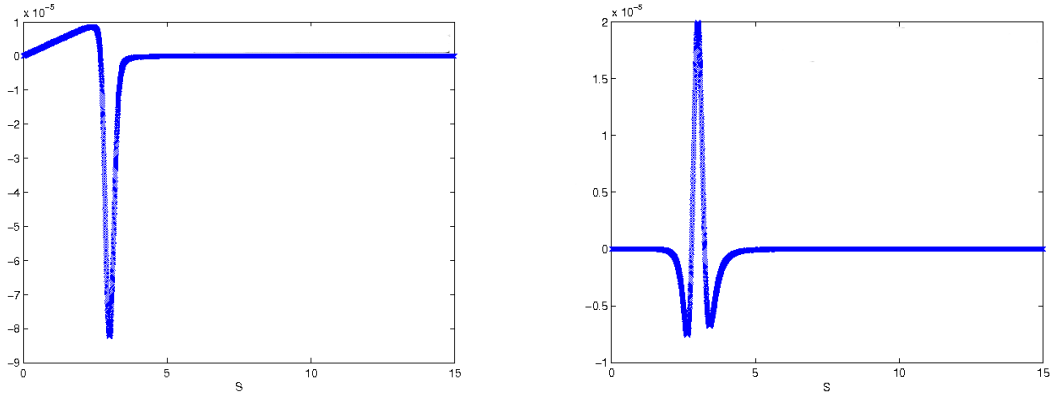


Figure 5.9: The error $(u_{exact} - u_h)$ on the full PIDE, with $h_{log} = 4.4e-03$. Left: $\alpha = 0.5$. Right: $\alpha = 1.5$.

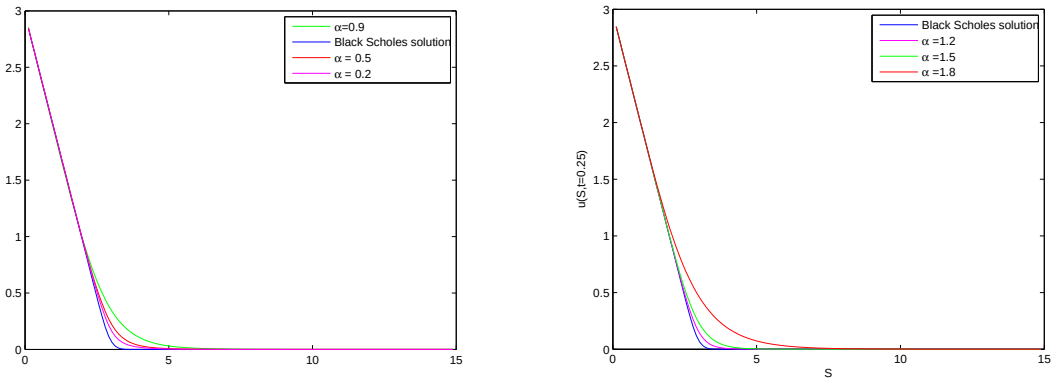


Figure 5.10: The solutions plotted with increasing $\alpha$ against the reference Black Scholes solution (blue). Left: $\alpha = 0.1$ (magenta), $\alpha = 0.5$ (red) and $\alpha = 0.9$ (green). Right: $\alpha = 1.2$ (magenta), $\alpha = 1.5$ (green) and $\alpha = 1.8$ (red). On the plot to the left $C = 1$ to emphasis the difference, otherwise all parameters are set equal to (Table 5.1).

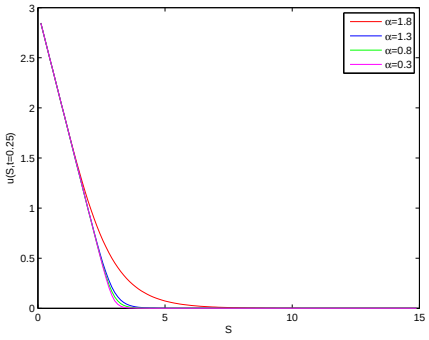


Figure 5.11: The full solution plotted with increasing $\alpha$. $\alpha = 0.2$(magenta), $\alpha = 0.8$(green), $\alpha = 1.3$ (blue), $\alpha = 1.8$ (red).

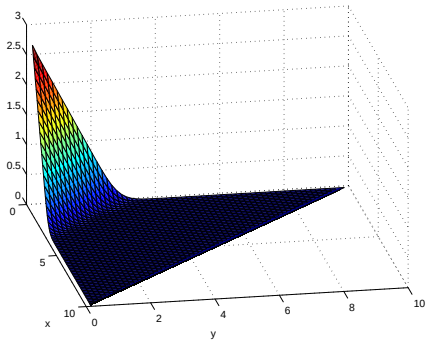


Figure 5.12: The solution of the 2-dimensional problem

# Chapter 6

# Conclusion and Future Work

To conclude, the two new schemes presented with the auxiliary logarithmic grid will not be monotone. But by a simple logarithmic transformation, obtaining an easier problem, both schemes are proven to be monotone, consistent and stable in $L^\infty$, hence convergent by Barles-Perthame Souganidis and the discussion in the Analysis. This approach does however come at a cost, as some generality is lost in the transformation. The lesson being, as also is the case in financial markets; there is no free lunch. By obtaining a lower computational complexity, some of the properties of the algorithms in [8] is lost by this approach, either the generality or the monotonicity.

The numerical results verifies the $\alpha$-dependent convergence, expected by the implementation chapter, with convergence lying in between 1st and 2nd order. The solution of the full algorithms behave according to what is to be expected, with slightly increased option prices compared to the Black-Scholes model, as the risk has been increased via the introduction of an infinite activity jump process.

Although the integration in time was handled by an operator splitting approach, there was not much to gain by this approach. Implicit methods are mainly employed to relax harsh CFL conditions that could arise from explicit schemes. By doing more work in each iteration, but performing fewer steps in time, overall CPU-time could be saved. This is not the case here. The asymptotic CFL condition is easily seen to be the same in the implicit as the explicit case, that is $\Delta t$ should be refined proportionally to $h^2$. The positive side being the increased work pr. iteration is low in the 1D case, implying that this approach yields roughly the same total work as an explicit method. In the 2 dimensional case however, solving the $N$ by $N$ matrix system which arises from the discretization of the local operators, has an impact in this implicit approach, yielding computational complexity of $O(N^5)$. A fully explicit method would lower the computational complexity to $O(N^4 \log N)$ as discussed during the introduction of the discretized scheme.

As mentioned (in Chapter 2 and 5), if the brownian term is removed, the CFL condition gets relaxed. Hence when $\alpha \in (0, 1)$, $\Delta t$ should be refined proportionally to $h$. In that case, very fast algorithms can be constructed which will still maintain monotonicity. For instance the full running time of the 1D algorithm when $\alpha \in (0, 1)$ gets lowered from

$O(N^3 \log N)$ to $O(N^2 \log N)$, which seems very appealing. When $\alpha \in [1, 2)$ the CFL condition gets relaxed to $h^\alpha$, yielding the somewhat lower complexity from $O(N^3 \log N)$ to $O(N^{\alpha+1} \log N)$. This is an approach that shows promise, and could result in very fast and efficient algorithms, while still having rigorous convergence properties.

# Bibliography


[1] Ariel Almendral and Cornelis W. Osterlee. Accurate evaluation of european and american options under the cgmy process. *SIAM J. Sci. Comput.*, 29(1), 2007.

[2] Guy Barles and Panagiotis E Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic Anal*, (3):271–283, 1991.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

[4] P.P. Carr, H Geman, D.B Madan, and M Yor. The fine structure of asset returns: An empirical investigation. *J. Of Business*, 75:305–332, 2002.

[5] Rama Cont and Peter Tankov. *Financial Modelling with Jump Processes.* Chapman and Hall, 2004.

[6] Y. D'Halluin and P. A. Forsyth. Robust numerical methods for contingent claims under jump diffusion processes. *IMA Journal of Numerical analysis*, 25(1):87–112, 2005.

[7] Matteo Frigo and Steven G. Johnson. www.fftw.org. The particular fast fourier algorithm used.

[8] Espen R. Jakobsen, Imran H. Biswas, and Kenneth H. Karlsen. Difference quadrature schemes for nonlinear degenerate parabolic integro-pde. *working paper*, 2008.

[9] Espen R. Jakobsen and Kristian Debrabant. Consistent and easy to implement monotone schemes for the bellman equation of optimal control. *working paper*, 2008.

[10] Mathworks. http://www.mathworks.com/support/technotes/1600/1605.html?bb=1. MATLAB MEX documentation.

[11] William H Press, Saul A. Teukolski, William T. Vetterling, and Flannery Brian P. *Numerical Recipes in C: The Art of scientific computing.* Cambridge University Press, 1992.

[12] Mark Yoshi. *Concept and practice of mathematical finance.* Cambridge, 2003.

[13] S Zhang and J Jin. *Computation of special functions.* Wiley, 1996.