Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA CIVILE, EDILE E AMBIENTALE

Corso di Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Iterative methods for option pricing in Merton's Jump diffusion model

Relatore:
Prof. Luca Bergamaschi

Laureando:
Andrea Baldina
1211477

Anno Accademico 2021/2022

Contents

Introduction

In financial markets, options are definitely among the most traded financial derivatives. Essentially, an option is a contract which offers the holder the right to buy or sell an asset at a mutually agreed-upon strike price, while the other counterpart of the contract is obliged to buy or sell the underlying asset once the option holder decides to exercise the option. Therefore, the option contract at first turns out to be an asymmetric opportunity; for such reason, the holder must pay a premium to the other counterpart in order to keep the risk fair, which is the so called option price. This raises a question: how can we price an option?
A first approach was developed in 1900 due to L. Bachelier in his thesis [1], whom proposed the application of a discrete version of a Brownian motion to model the asset price. Then, in 1973 the most famous and used model was proposed by Black and Scholes in [3], whose contribution was identifying a partial differential equation that could satisfy the fair price of an European option contract, taking into account the time and risk factors inherent in the fluctuation of the underlying asset. Moreover, the PDE for this model admits an analytic solution which is called Black and Scholes formula, but despite its widely usage even today, this model has some well-known limits. As a consequence, many other alternative models have been developed as stochastic volatility models (Heston [12], Bates [2]), local volatility models (Dupire [9]) and jump-diffusion models (Merton [18] and Kou [15]). Some of these methods can be solved using analytical solutions, more commonly for vanilla-style options. Nevertheless, numerical methods are still used, since they might perform faster than the closed formula and also provide a solution over the entire time horizon rather than just at one point in time.
The main theme of this thesis framework will focus on addressing the problem of option pricing in a market modelled through jump-diffusion processes, and in particular we will focus on the Merton jump-diffusion model. In recent years several numerical methods studies has been carried out to further explore this model, many of those proposing numerical solutions for PDEs and in particular applying implicit-explicit finite difference methods as we can see in the framework [4] and [7]. The aim of this thesis is to approach implicit finite difference schemes and implement an efficient linear system solver in order to compare the accuracy and the computational cost between the implicit schemes and the implicit-explicit one. For each scheme implemented, we will be required to create as many linear systems as the number of time steps in which the time domain is decomposed; the matrices associated with such linear systems will be constant, while the other components will vary over time with reference to the solution of the previous step.
As a means to obtain an efficient solver, it will be given an overview of some of

the most popular iterative methods and, once the characteristics and properties of the schemes are determined, we will exploit the most adapted methods, namely the GMRES method in two different versions, without and with the integration of a scalable preconditioner to speed up its convergence; finally, a further attempt will be made implementing a Multigrid V-cycle solver.
The main goals that we would like to achieve are the following:

- Develop implicit schemes for Merton jump-diffusion model, using the finite differences method to discretize the model PDE; the matrix associated with the obtained linear system will be dense, hence it will be necessary to develop iterative methods that exploit preconditioners.
- Experiment the efficiency of different possible preconditioners in order to accelerate the convergence of the mentioned iterative methods.
- Develop an implicit-explicit for Merton jump-diffusion model using finite differences method to discretize the model PDE; direct methods will therefore be applied to tridiagonal linear systems.

The thesis is organized by chapters as follows:

1. In the first chapter, we briefly introduce the Black and Scholes model and present the Merton jump-diffusion model and its partial differential equation.
2. In the second chapter, we introduce the finite differences method and the matrix-vector multiplication algorithm with Fast Fourier Transform.
3. In the third chapter, we present the numerical schemes used to discretize the PDE and the algebraic properties for the linear systems introduced.
4. In the fourth chapter, we study the chosen linear system solvers, which are GMRES method, Multigrid V-cycle and the direct method for tridiagonal linear systems.
5. In the last chapter, we show the numerical experiments carried out for all the analyzed schemes and for each of them the linear system solvers used is presented and compared with the analytic solution.

Chapter 1

Option pricing models

In this chapter, we will introduce the Black and Scholes model and the main topic of this thesis that will be the Merton's jump diffusion model, then we will discuss their principal features and finally present the equation problem that we are going to analyze in the next chapters.

1.1 Financial aspects

In order to have a more comprehensive view of the problem at hand, we first introduce the concept of option. Options are indeed one of the most popular and widely used financial derivatives. It is a contract between two or more parties that gives one side the right, but not the obligation, to buy or sell the underlying asset to the other counterparts within a specific time-frame; in particular, there are no restrictions on the type of underlying asset, it can be represented by a stock, a bond, an interest rate or even commodities and so on. Intrinsically, the price of an option would be the cost of neutralizing the risk to which the owner of the right is exposed; in fact, whoever holds that right pays a premium to exercise the right or to not exercise it. Besides the price, other fundamental elements of this contract are: the underlying asset value defined as S_t, the strike price K which is the price at which the underlying asset will be sold or bought and the expiration date T which defines the date of maturity of the option. The most famous options styles are European options, which means that the derivative can be exercised only at the maturity time specified in the contract as T, which is the expiration date of the option as well. Besides European options, there are many other types of options, such as: American options, Asian options, Barrier options, Bermudan options and so on. American options, differently from the European ones, can be exercised at any time before the maturity time. Instead, Asian options are characterized by having a payoff which depends on the average price of the underlying. Alternatively, a Barrier option is an option whose payoff strictly depends on the fact that the underlying reaches or not a certain fixed barrier value. Finally, Bermudan options can be exercised at specific date or time intervals, which means they can be formalized as a hybrid between American and European ones. For the purposes of this analysis we will focus on European style option, which can be mainly divided in two categories according to the type of right it is given to the owner as outlined below:

- Call: allows the holder to buy the asset at the strike price within a specific time-frame
- Put: allows the holder to sell the asset at the strike price within a specific time-frame

Let us define the option's payoff at the maturity date as:

$$\phi(S_T, K) = \begin{cases} (S_T - K)^+, & \text{(call option)} \\ (K - S_T)^+, & \text{(put option)} \end{cases} \tag{1.1}$$

In order to price options, among the set of pricing techniques available in quantitative finance numerical methods have become a relevant area of development and are now used for broad applications; therefore, this thesis will hence be focused on investigating their use for this purpose.
Why are options so used?
Basically, there are two main reasons: they are used as a hedging device, since they can provide investors risk-reduction strategies for their own portfolio, and secondly they can also be used for speculative reasons.
Why do they need to be priced?
By its definition as a contract, an option gives to its holder the right, rather than an obligation, to "buy/sell" the underlying asset; as we already mentioned, in order to obtain this entitlement the buyer of the contract must pay a certain amount, namely the option's fair value, since otherwise it would be intuitively potentially unfair for the other counterpart and risk would be unbalanced. Therefore, option pricing theory consists in evaluating the option by assigning a price which is based on the calculated probability to create a positive income at the end of the contract, known as the case where the option will end up to be In-The-Money (ITM). Besides the terminal case at the maturity, where indeed the price is directly given by the payoff, during the period that elapses between the date of signature of the contract and the maturity there is no clue about the price of such instrument. In 1973 the economists Fischer Black and Myron Scholes developed a fundamental formula in option pricing theory, namely the Black and Scholes formula [3], which permits the calculation of the price for European call and put options.

1.2 Toolbox

To deeper analyze the model that will be subsequently presented, it is worth recalling some useful notions of finance and probability, which can be further explored on [16]. We consider a financial market where the underlying is traded continuously up to a fixed maturity time T. The market is defined by a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ endowed with a right continuous filtration $\mathbb{F} = \{\mathcal{F}_t \mid t \in [0, T]\}$ where $\mathcal{F}_T = \mathcal{F}$. Let us start with the properties of a Stochastic process:

Definition 1 (Stochastic Process)**.** *Let $T \subseteq [0, \infty)$, a family of random variables $\{X_t\}_{t \in T}$ (indexed by T) is called a stochastic process. When $T = \mathbb{N}$,$\{X_t\}_{t \in T}$ is said to be a discrete-time process, while if $T = [0, \infty)$ the family of random variable is called continuous-time process.*

In this thesis we will deal mostly with random variables that will be distributed following the probability density of a normal distribution; where a random variable X is said to be normally distributed with mean μ and variance σ^2 if its probability density function is the following:

$$X \sim \mathcal{N}(\mu, \sigma^2) \implies f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad x \in (-\infty, +\infty)$$

Another important probability distribution used in this framework is the Poisson distribution which is different from the Gaussian distribution due to the fact it is a discrete probability distribution defined as:

$$\mathbb{P}(X = n) = \frac{\lambda^n e^{-\lambda}}{n!}$$

where:

- n is the number of occurrences of an event
- λ is the expected value of event in a interval

Definition 2 (Gaussian process). *Let a stochastic process $\{X_t\}, t \in [0, T]$, it is called Gaussian process such that the random vectors $(X_{t_0}, X_{t_1}, \ldots, X_{t_n})$ have a joint Gaussian distribution for all n-tuples $(t_1, t_2, \ldots, t_n)$ where $t_i \in T$.*

Another important stochastic process is the Brownian motion, whose definition is the following:

Definition 3 (Brownian motion). *A Brownian motion is a stochastic process $B_t, t \in [0, \infty)$ such that fulfills the following conditions:*

1. *$B_0 = 0$*
2. *With probability 1, $t \to B_t$ is continuous on $[0, \infty)$*
3. *$\{B_t\}_{t \geqslant 0}$ has stationary, independent increment*
4. *If $0 < s < t$ then $(B_t - B_s) \sim N(0, t - s)$*

As by definition the Brownian motion has a stochastic behavior, therefore we are going to simulate it using Montecarlo simulation. Defining the law of the Brownian motion we have $\mathcal{L}(B_t) = \mathcal{N}(0, t)$, let $\Delta t > 0$ be a constant time increment. Now discretizing the time we have $T_j = j\Delta t$ the value of Brownian motion B_t can be written as a series of increments:

$$B_{j\Delta t} = \sum_{k=1}^{j} (B_{k\Delta t} - B_{(k-1)\Delta t})$$

The increments can be calculated from a random variable x distributing with normal distribution, $x \sim \mathcal{N}(0, 1)$, so we have the discrete model for the Brownian process which will be:

$$\Delta B_k = x\sqrt{\Delta t} \qquad \forall k$$

Indeed to simulate the value of B_t we firstly generate a Normal random variable X and then we evaluate B_t as multiplication of the random variable and the standard deviation of the Brownian motion which is by the definition $\sqrt{t}$. To simulate the process B_t we discretize the time domain $\mathcal{T} = [0, T]$ in k steps so we have $\mathcal{T}_k = t_0 = 0, \ldots, t_N = T$, then using the Forward simulation we get:

$$B(0) = 0 \qquad \text{by the first feature of Brownian motion}$$

and for the rest of element we have:

$$B_{t_k} = B_{t_{k-1}} + \sqrt{t_k - t_{k-1}} x_k$$

Here below we are going to show a simulation for the Brownian motion using MATLAB:

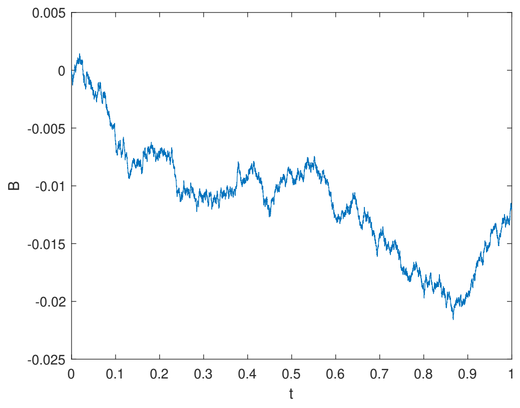


Figure 1.1: Simulation Brownian motion.
Number of steps $n = 10000$.

In the famous Black and Scholes the behavior of asset price (denoted by S) is shaped under the risk-neutral probability measure $\tilde{\mathbb{P}}$ by the following stochastic differential equation:

$$dS_t = rS_t dt + \sigma S_t dB_t$$

The stochastic process S_t is sad Geometric Brownian Motion if it satisfies the previous SDE. Since the asset price in the Merton model is described by Levy process, let us define this kind of process.

Definition 4 (Levy process). *A stochastic process $\{X_t\}_{t \geq 0}$ on $(\Omega, \mathcal{F}, \mathbb{P})$ such that $X_0 = 0$ is called Levy process if it fulfills the following properties:*

1. *Stationary increments: $\forall t \in (t_1, t_n)$ the random variables $X_{t_{i+1}} - X_{t_i}$ does not depend on variable t*

2. *Independent increments: $\forall t \in (t_1, t_n)$ the random variables $X_{t_{i+1}} - X_{t_i}$ are independent*

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

The most important stochastic formula in quantitative finance is Ito's formula, as defined in [16]:

Theorem 1. *Let $(X_t)_{t\in[0,T]}$ be an Ito process:*

$$X_t = X_0 + \int_0^t K_s \, ds + \int_0^t H_s \, dB_s$$

where:

$$H_t(\omega) = \sum_{i=1}^{n} \phi_i(\omega) \boldsymbol{1}_{(t_{i-1},t_i]}(t)$$

and let $f \in \mathcal{C}^{1,2}$. Then

$$f(X_t) = f(X_0) + \int_0^t f^{'}(X_s)dX_s + \frac{1}{2}\int_0^t f^{''}(X_s)H_s dB_s$$

where, by definition,

$$\langle X, X\rangle = \int_0^t H_s^2 ds$$

Likewise, if $(t,x) \to f(t,x) \in \mathcal{C}^{1,2}(t,x)$, the Ito formula becomes:

$$f(t,X_t) = f(0,X_0) + \int_0^t f_s(s,X_s)ds + \int_0^t f_x(s,X_s)dX_s + \frac{1}{2}\int_0^t f_{xx}(s,X_s)d\langle X,X\rangle_s$$

1.3 Black and Scholes model

Let us start from the most famous option pricing formula: Black and Scholes. As [18] states, to define the option pricing formula of Black and Scholes we assume the following ideal conditions in the market of stocks and derivatives:

- Frictional markets: no transactions or differential taxes costs, trading is continuously in time and borrowing and short-selling are allowed without restriction. Moreover same rate of borrowing and lending. The assets are perfectly divisible.

- The short-time interest rate is known and it is constant on time

- The stock does not pay any dividends during the option life

- Only European option

- The stock price behavior is defined by geometric Brownian motion on time, thus the stock price has a log-normal distribution on time

Let us consider a financial market defined by the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and given the standard Brownian motion adapted to the filtration $W = (W_t)_{t\in[0,T]}$. We define two assets: risk-less asset and risky asset. The risk-less asset is a risk-free bond with price B_t while the other risky asset has S_t as price value. The two asset fulfill the following SDE as follows:

$$\begin{cases} dB_t = rB_t dt \\ dS_t = \mu S_t dt + \sigma S_t dB_{t_0} \end{cases} \tag{1.2}$$

The interest rate r, the mean rate return of underlying value μ and the volatility of underlying σ are assumed to be constants. The model is valid on the time domain $[0, T]$, the equations on (1.2) thanks Ito's formula has the following closed-form solution:

$$\begin{cases} B_t = e^{rt} \\ S_t = S_0 e^{\mu t - \frac{\sigma^2}{2} t + \sigma B_t} \end{cases} \tag{1.3}$$

s where S_0 is the price got at the initial time in our case $t = 0$ (spot price), one can see the result S_t has a log-normal behavior. Given this formulation for the asset value S_t, the main goal is to find the price of a option contract. Let $V(S_t, t)$ the price of an option contract, by (1.1) the payoff at expiry date $V(S_T, T)$ is known, to find the value $V(S_t, t)$ s.t. $t \in [0, T)$, we need a fundamental result in stochastic calculus: Ito's lemma from theorem (1). Through that, given a scalar function $f(x, t)$ and a process X_t called Ito drift-diffusion process giving by:

$$dX_t = \mu dt + \sigma dB_t \tag{1.4}$$

and given a scalar function $f(x, t)$, the Ito's lemma states in differential terms states:

$$df(X_t, t) = \Big(\frac{\partial f}{\partial x}\mu + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\sigma^2\Big)dt + \sigma\frac{\partial f}{\partial x}dB_t \tag{1.5}$$

Now, apply this formula to $V(S_t, t)$ we obtain the following equation:

$$dV(S_t, t) = \Big(\frac{\partial V}{\partial S_t}\mu S_t + \frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S_t^2}\sigma^2 S_t^2\Big)dt + \sigma S_t\frac{\partial V}{\partial S_t}dB_t \tag{1.6}$$

As defined by [13], the definition of the portfolio is:

$$\Pi = -V + \frac{\partial V}{\partial S}S \tag{1.7}$$

Thus, the change of portfolio in the time interval is given:

$$\partial\Pi = -\partial V + \frac{\partial V}{\partial S}\partial S \tag{1.8}$$

Substituting equation (1.5) into (1.8) we get:

$$\partial\Pi = \Big(-\frac{\partial V}{\partial t} - \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\Big)\partial t \tag{1.9}$$

Since this equation does not involve ∂S, this leads the portfolio has to be risk-less on ∂t. Therefore due to the assumptions, the portfolio have to earn the same rate of return as other short-term risk-less asset instantaneously. It follows:

$$\partial \Pi = r\Pi\partial t \tag{1.10}$$

Substituting the equation (1.7) and (1.9) into (1.10), we get

$$\Big(\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\Big)\partial t = r\Big(V - \frac{\partial V}{\partial S}S\Big)\partial t$$

Therefore that:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 = rV \tag{1.11}$$

This equation above is known as Black-Scholes PDE, with initial condition $V(S_t, t) = \phi(S_t, t)$. The option price in Black and Scholes framework $V(S_t, t)$ can be expressed by the following theorem in [16, Chapter 4.3.2]:

Theorem 2. *In the Black and Scholes model, any option defined by a non-negative, $\mathcal{F}_{\mathcal{T}}$-measurable random variable $\phi(S_t, t)$, which is square-integrable under the probability $\mathbb{P}$, is replicable and the value at time t of any replicating portfolio is given by:*

$$V(S_t, t) = e^{-r(T-t)}\mathbb{E}^{\mathbb{P}}\Big[\phi(S_t, t)|\mathcal{F}_t\Big]$$

where the function $V \in C^{1,2}([0, T] \times \mathbb{R})$ solve the Black and Scholes PDE (1.10).

We recall that the space of square-integrable functions is:

$$L^2(\Omega) = \{f : \Omega \mapsto \mathbb{R} \mid \int_{\Omega} |f(x)|^2 d\Omega < +\infty\}$$

As stated in [13, Chapter 15.8], we can write the closed-formulas for European call and put, as following

$$V(S_t, t) = \begin{cases} V(S_t, t) = S_0\,\mathcal{N}(d_1) - Ke^{-r(T-t)}\mathcal{N}(d_2) & \text{(Call option)} \\ V(S_t, t) = Ke^{-r(T-t)}\mathcal{N}(d_2) - S_0\,\mathcal{N}(d_1) & \text{(Put option)} \end{cases} \tag{1.12}$$

where:

$$d_1 = \frac{\ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}} \qquad d_2 = d_1 - \sigma\sqrt{T-t} \tag{1.13}$$

1.3.1 Simulation of asset price in Black and Scholes model

The price of the underlying asset S_t is defined by the geometric Brownian motion in (1.2), to simulate it we have to discretize the time domain $\mathcal{T} = [0, T]$ in n steps in order to have $\mathcal{T}_k = t_0 = 0, \ldots, t_N = T$. Then we discretize the variable S_{t_k} and we have:

$$S_{t_k} = S_{t_{k-1}}e^{(r-\frac{\sigma^2}{2})(t_k - t_{k-1}) + \sigma B_{t_k - t_{k-1}}}$$

Then we discretize with evenly size intervals $\Delta t = t_k - t_{k-1}$. Next, we use the definition of thee simulation for the Brownian motion above, we have for the time steps k the following value for the underlying:

$$S_{t_k} = S_{t_{k-1}} e^{(r-\frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta} x_k}$$

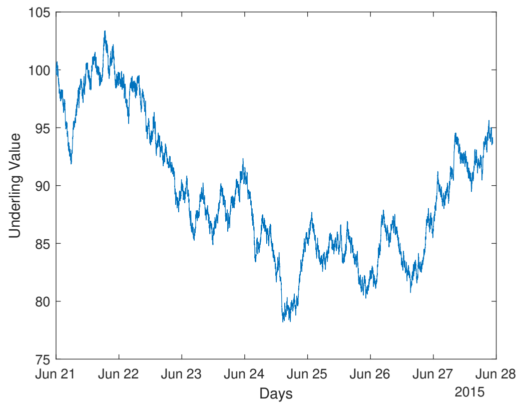


Figure 1.2: Simulation of underlying under Black and Scholes model (GMB). Given the data: $S_0 = 100$, $r = 0.02$, $\sigma = 0.2$ with number of steps equals to 10000.

1.4 Merton model in jump-diffusion market

Most of the literature in quantitative finance mostly for pricing and hedging of asset is based on the assumption that the price of the underlying assets follow the process described by Black and Scholes model. As was pointed out in [18] the critical assumption in Black and Schoels is the trading of the assets is continuously in time and its dynamics have a continuous sample with probability one. That is not real behavior for the assets' price, in fact, some events can lead to instantaneous variations in the price. Moreover, these assets' price changes produce a log-normal distribution for the price in the next price values, some studies of stock price series show many outliers for simple log-normal distribution. Additionally, as we can see from the stock price analysis there appear to be 'jumps'. Another fair alternative is to overlay continuous stock price changes with these sorts of 'jumps'. Moreover, this model despite the Black and Scholes model has the following properties:

- it is able to reproduce the leptokuric properties of the return distribution and the volatility smile [15]
- it can produce closed-form solutions for standard European options, and moreover it has closed formulation for path-dependent options: barriers , America, look-back options [15]
- one of its motivations comes from behavioral finance, in fact some studies say that markets tend to react to various news (see, for example, Fama, 1998 and Barberis et al., 1998). The jump part of the model is the market's response

to external news, while in the absence of external news the asset price follows a GB motion.

In this section, we are going to study the simplest one, the Merton's jump-diffusion model.

1.4.1 Asset dynamic

In this subsection, we will describe the dynamic for an asset in Merton model in jump-diffusion market. In Merton's study [18], it's stated that the stock price changes are the composition of two types of variations:

1. Default change: it is called the normal variation of the price, it happens when there is a gap between supply and demand, a change in rates or new information creating marginal variation in the asset's price. This type of variation is modeled by the GMB in Black and Scholes model.

2. Atypical change: it is due to new information on the stock which has more impact to the asset's price. This component is modeled by the jump process reflecting the non-marginal variation.

Therefore it follows that for the first component it can be used the Wiener process while for the jump component a suitable prototype is the Poison process. Following Merton's construction, let the standard Brownian motion $(W_t)_t \in [0,T]$, and let $\eta - 1$ be the change from S to ηS when the asset value undergoes a jump, while dq_t is Poisson compounded process and assumed to be independent of the Brownian motion. For dq_t we have:

$$dq_t = \begin{cases} 0 & \text{with probability} \quad 1 - \lambda dt \\ 1 & \text{with probability} \quad \lambda dt \end{cases}$$

Where λ is the average number of events per time interval for Poisson distribution. The underlying asset price S is given by a process in the form of:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dB_t + (\eta - 1)dq_t \tag{1.14}$$

Here, the probability distribution of the jump width η is:

$$J(\eta) = \frac{1}{\sqrt{2\pi}\gamma_j\eta_j} e^{\frac{-(\ln\eta_j - \mu_j)^2}{2\gamma_j^2}} \tag{1.15}$$

Therefore for $\mathbb{E}[\eta - 1] = \kappa$, where κ using the change of variable $\tilde{x} = \frac{(x-\mu_j)}{\gamma_j^2}$ has the following value:

$$\begin{aligned} \kappa = \mathbb{E}[\eta] &= \int_{\mathbb{R}} e^x - 1 dF(x) \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{x - \left(\frac{x-\mu_j}{\gamma_j}\right)^2 /2} \frac{dx}{\gamma_j} - 1 \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-\frac{\tilde{x}^2}{2} + \sigma\tilde{x} + \mu_j} d\tilde{x} - 1 \\ &= e^{\mu_j + \frac{\gamma_j^2}{2}} - 1 \end{aligned}$$

Assuming that the jump activity is finite we define the linear integral-differential operator:

$$\frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 V}{\partial S_t^2} + (r - \lambda\kappa) S_t \frac{\partial V}{\partial S_t} - (r+\lambda)V + \lambda \int_{\mathbb{R}} J(\eta_t) V(t, \eta_t S_t) d\eta_t = \mathcal{L}V \tag{1.16}$$

Therefore, the evolution of the asset's price for an European option under this jump-diffusion model can be defined through the following partial integral-differential equation:

$$\frac{\partial V}{\partial t} + \mathcal{L}V = 0 \tag{1.17}$$

The $J(y)$ is the probability density function of the jump κ. The previous PIDE can be derived by standard no-arbitrage arguments and Ito's lemma generalization, see [16] and [18]. With $\mathcal{L}V$ we defined the linear integral-differential operator associated with the jump-diffusion process S_t, for future convenience. We are going to look for a solution $C^{1,2}$ for the Cauchy problem:

$$\begin{cases} \dfrac{\partial V(S_t,t)}{\partial t} + \mathcal{L}V(S_t,t) = 0 & \forall (S_t,t) \in \times[0,T), \\ V(S_T,T) = \Phi(S_t) & \forall S_T \in \mathbb{R} \end{cases} \tag{1.18}$$

The boundary conditions are defined as following:

$$\Phi(S_T,T) = \begin{cases} (S_T - K)^+ & \text{(Call option)} \\ (K - S_T)^+ & \text{(Put option)} \end{cases}$$

Before delving into the numerical analysis for this problem, we are going to show the closed formulation for the price of an option developed by Merton in [18], this formula will be compared to the numerical methods we will have implemented in the next sections. We try to give an explicit expression for the price of a European call option or a European put option with the strike price K. The following formulation found details in the original paper of Merton [18], which is written as:

$$V(S,t) = \sum_{n=0}^{\infty} \frac{(\lambda' t)^n}{n!} e^{-\lambda'\tau} V_{BS}(S,\tau,K,r_n,\sigma_n) \tag{1.19}$$

where we have:

$$\lambda' = \lambda(1+\kappa) \qquad \sigma_n^2 = \sigma^2 + \frac{n\gamma^2}{t} \qquad r_n = r - \lambda\kappa + \frac{n}{t}\Big(\mu + \frac{1}{2}\gamma^2\Big)$$

And where $V_{BS}(S,\tau,K,r_n,\sigma_n)$ is the Black and Scholes formula evaluated with no jumps which is defined in (1.12).

1.4.2 Simulation underlying price of jump-diffusion in Merton model

For the simulation of the underlying asset price we follows the idea in [16], we first defined the price S_t jumps at random times $t_1, \ldots, t_n, \ldots$ and the relative change of its value with $U_1, \ldots, U_n, \ldots$. We assume that between two jumps times, the price S_t follows the Black and Scholes model; then at time t_n is the time jumps of a Poisson process N_t with magnitude λ_t. Then, on the intervals $[t_n, t_{n+1})$ be the time interval where there is a jump, so by the equation

$$dS_t = \mu S_t dt + \sigma S_t dB_t$$

which is the process with out jumps. Instead, at the time $t = t_n$ the jump is given by:

$$\Delta_n = S_{t_n} - S_{t_n-} = U_n S_{t_n-}$$

So we have:

$$S_{t_n} = (1 + U_n) S_{t_n-}$$

We have for $t \in [0, t_1)$:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma B_t}$$

for $t \to t_1^-$

$$S_{t_1^-} = S_0 e^{(\mu - \frac{\sigma^2}{2})t_1 + \sigma B_{t_1}}$$
$$S_{t_1} = S_0 (U_1) e^{(\mu - \frac{\sigma^2}{2})t_1 + \sigma B_{t_1}}$$

Moreover, with $t \in [t_1, t_2]$, we have

$$S_t = S_0 (1 + U_n) e^{(\mu - \frac{\sigma^2}{2})t + \sigma B_t}$$

For the jump part, $U_n = e^{\gamma + \delta x_n}$ where $x_n \sim \mathcal{N}(0, 1)$. While for the simulation of no-jumps part we follows the previous statements in (1.3.1). For this underlying dynamic we implement a MATLAB simulation in order to show the behavior with these assumptions in this model:

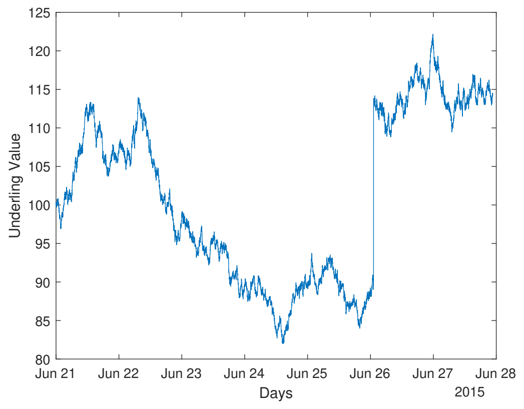


Figure 1.3: Simulation of underlying under Black and Scholes model (GMB). Given the data: $S_0 = 100$, $r = 0.02$, $\sigma = 0.2$, $\lambda = 0.7$, $\delta = 0.3$, $\gamma = 0$ with number of steps equal to 10000.

In this simulation (1.3), the underlying's price follows the behavior of (1.2) except on 26 June where we can see a vertical change in the price. This is due to a jump, in fact at this step the Poisson distribution probability assumed value equals 1, so in that, we have the jump contribution.

1.5 Merton Jump-Diffusion vs Black and Scholes

In this section, we are going to show a comparison between the Black and Scholes pricing option and Merton jump diffusion We below we will plot two graphs for European calls using different λ for the Merton model. We will call "MDJ-model" Merton jump diffusion model while "BS-model" the Black and Scholes model. Now we define the benchmark parameter in common for the models, and the options we are going to evaluate:

- Strike price $K = 1$
- $\sigma = 0.35$
- $r = 0.5$
- Time to maturity $T = 1$

For the European call option we have:

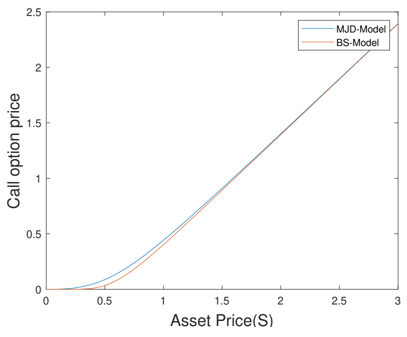


Figure 1.4: European call option.For Merton model: $\lambda = 1$, $\gamma = 0.5$

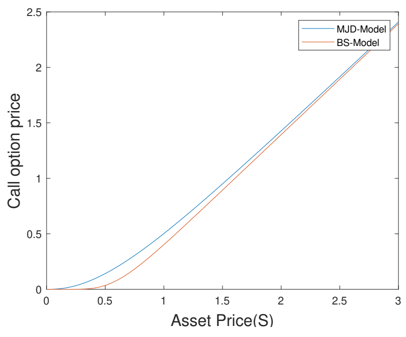


Figure 1.5: European call option.For Merton model: $\lambda = 2.5$, $\gamma = 0.5$

For the pricing of put option we have:

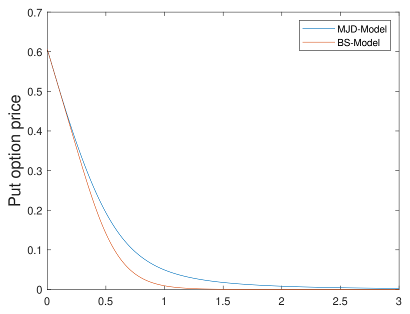


Figure 1.6: European put option.For Merton model: $\lambda = 1$, $\gamma = 0.5$

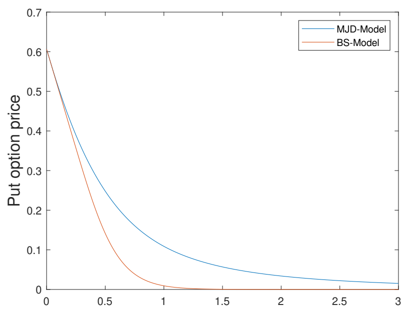


Figure 1.7: European put option.For Merton model: $\lambda = 2.5$, $\gamma = 0.5$

As we can see from the graphs the option prices in Merton's model are greater than Black and Scholes model according to the theoretical results in [18]. Moreover, the larger the λ, the larger the price; this fact can easily be seen in the closed formula for Merton model in (1.19). Furthermore from [17], the standard deviation for Black and Scholes is:

$$Std_{BS}(S_t) = \sigma_{BS}\sqrt{t}$$

While for Merton, from the equation (1.14) at t-period it is:

$$Std_{MJD}(S_t) = \sqrt{(\sigma_{MDJ}^2 + \lambda\kappa^2 + \lambda\mu^2)t}$$

This means that if we set $\sigma_{BS} = \sigma_{MDJ}$, the Merton price is always greater than Black and Scholes, by the assumption of the parameters.

Chapter 2

Numerical backgrounds

As we will see in Chapter 3, to find the option price for Merton's jump diffusion model, we will have to solve the partial integro-differential equation (1.16), which has been presented in Chapter 1. In this Chapter, we introduce the tools which will be used: the finite differences schemes we will use and the matrix-vector algorithm using Fast Fourier Transform (FFT).

2.1 Finite Differences Toolbox

In this section, we present some numerical tools for finite difference schemes. The main references for the tools used in this chapter is in (see, Chap. 3 and Chap. 8 in [10] and [5]), from which most of information has been taken about these tools. Since we are going to discretize the PDE using the finite difference method, we introduce the scheme for ordinary differential equation (ODE) which definition is:

$$F\left(x, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \dots, \frac{d^ny}{dx^n}\right) = 0$$

For purposes of example we focus on first order differential equation, whose Cauchy problem is the following:

$$\begin{cases} \dfrac{dy}{dx} = f(x,y) \\ y(x_0) = y_0 \end{cases} \tag{2.1}$$

For this kind of equation, we are going to describe three famous schemes: Forward Euler, Backward Euler and Crank-Nicolson. First of starting with scheme, we will defined an important property for a scheme that is stability which is defined as:

Definition 5 (Stability)**.** *Given a scheme (S) it is called stable if and only if a for a given bounded initial condition, the solution does exist and is bounded along its variable steps. Considering a domain defined on $[0,T] \times \mathbb{R}$, it has to be bounded independently in Δt, Δx:*

$$\exists \quad C > 0, \quad \forall \Delta t > 0, \quad \Delta x > 0, i \in \mathbb{Z}, n \in \{0, \dots, M\} : \quad |u_i^n| \leqslant C$$

2.1.1 Forward Euler

For the formulation of this scheme for the Cauchy problem (2.1), we discretize the x domain in order to have: $x_0 < x < x_n$ where $x_{i+1} = x_i + h$, where $h = (x_n - x_0)/n$ and n are the spatial point we chose in order to divide our x domain. Therefore, we have in correspondence of the node x_n the following formulation:

$$y(x_{i+1}) = y(x_i) + hf(x_n, y(x_i))$$

In fact one might view this method as a series of first order Taylor expansion of the function y, this follows that for the node x_{i+1} the Taylor series around x_i is:

$$y(x_{i+1}) \approx y(x_i) + y'(x_i)(x_{i+1} - x_i) = y(x_i) + hf(x_n, y(x_i))$$

The solution is an approximation with a polygonal of segment whose slope is defined by the function. For the convergence of the forward Euler method, we have the following result from [5]:

Theorem 3. *Assuming that for each value of n the initial error $|y(x_0) - \hat{y}_n(x_0)| \leqslant K_n$ where K_n is a constant and the greatest stepsize is bounded by a constant H_n. Moreover, if $n \to \infty$, $H_n \to 0$ and $K_n \to 0$. Under these condition:*

$$||y_(x) - \hat{y_n}(x)|| \to 0, \qquad as \to 0$$

In this method the local error is bounded by a constant which behavior follows $\mathcal{O}(h^2)$, while for the global error we have the following theorem:

Lemma 1. *For the forward Euler method the global error can be bounded by:*

$$|e_n| \leq Ch$$

where $|e_n| = |x(t_n) - y_n|$, for all $n = 0, \dots, N$; where $C > 0$ is constant that does not depend on h.

Thus, the global error has a behavior that follows $\mathcal{O}(n)$. As last property, this scheme can be conditional stable, which means that it is stable if and only if the time step h fulfills a certain condition, which depends on the ODE considered.

2.1.2 Backward Euler

Discretizing the x domain as above, in correspondence of the node x_i the formulation is:

$$y(x_{i+1}) = y(x_i) + hf(x_{i+1}, y(x_{i+1})) \tag{2.2}$$

This can be derive considering the equation (2.1) and integrating it from x_i to $x_{i+1} = x_i + h$ which yields:

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y(x))dx$$

Approximating the integral through rectangle method it becomes:

$$y(x_{i+1}) - y(x_i) \approx hf(x_{i+1}, y(x_{i+1}))$$

For the convergence of the backward Euler method, we have the following in:

Theorem 4. *Let $y_h(t)$ be the piece-wise linear interpolant of the backward Euler method, in which h is the time step. Then, let $x(t)$ be the actual solution for the Cauchy problem (2.1). Then there exist a constant $C > 0$ such that:*

$$||y_h - x||_\infty \leq Ch$$

Therefore, the method converges to the actual solution as $h \to 0$.

Then, the error estimates and the proof of convergence of the implicit Euler are quite similar to the forward scheme. For the local truncation error in this method, it can also be bounded by a constant whose behavior follows $\mathcal{O}(h^2)$. Furthermore, as the forward Euler's global error, the backward Euler global error can be bounded by:

$$|e_n| \leq Ch$$

where e_n is the error between the actual solution and the method solution at the n time step, in addiction C does not depend on the time step h. By this inequality the global error of this scheme behaves as follows $\mathcal{O}(n)$.
Differently, from the explicit method, this method is more expensive in terms of computation since we have to solve the non-linear equation, but it is an unconditionally stable method and therefore we can choose h larger.

2.1.3 Crank-Nicolson method

Making generalizations, we can computed the arithmetic average of the two method above in order to have a second-order method: Crank-Nicolson method. It indeed is based on the trapezoidal rule and it is defined as follows:

$$y(x_{n+1}) = y(x_n) + \frac{h}{2}\Big[f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))\Big] \tag{2.3}$$

In this scheme, the local error behaves as $\mathcal{O}(h^3)$, while the global error is $\mathcal{O}(h^2)$, which means that Crank-Nicolson is more accurate than the Forward and the Backward Euler methods.
Likewise the implicit Euler, it is an implicit method to get the y_{n+1} value. Therefore, it needs an iterative method to get the value. Nevertheless, it is unconditionally stable, thus it converges for each choice of the spatial interval h.

2.1.4 Finite Difference Schemes

The basic idea behind the finite difference scheme is to define the derivatives of the equation by finite differences. The derivatives are approximated by Taylor expansion. Certainly, the finite differences scheme can be used for every order of derivatives, for our scope we are going to present the first and the second derivative approximations. Starting from the first derivative of a function $u(x)$ evaluated at node x_i, there are three main approximations: forward approximation, backward approximation and central approximation. The following formulation is for forward approximation:

$$\frac{\partial u}{\partial x} = \frac{u(x+\Delta x) - u(x)}{\Delta x} - \mathcal{O}(\Delta x)$$

In the case we switch $-\Delta x$ to Δx it becomes a backward difference approximation for the first derivative:

$$\frac{\partial u}{\partial x} = \frac{u(x) - u(x-\Delta x)}{\Delta x} - \mathcal{O}(\Delta x)$$

For the central approximation we take difference between two x steps, it becomes:

$$\frac{\partial u}{\partial x} = \frac{u(x+\Delta x) - u(x-\Delta x)}{2\Delta x} - \mathcal{O}(\Delta x)^2$$

Then for the second derivative, we have:

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &\approx \frac{u'(x+\Delta x) - u'(x)}{\Delta x} \\
&\approx \frac{\frac{u(x+\Delta x)-u(x)}{h} - \frac{u(x)-u(x-\Delta x)}{h}}{\Delta x} \\
&= \frac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{\Delta x^2} + \mathcal{O}(\Delta x^2)
\end{aligned}$$

All the previous formulation can be easily derived from Taylor's expansion.

2.1.5 Integral approximation

To resolve an integral numerically we define a formulation belonging to the quadrature class the trapezoidal rule which is defined as:

Proposition 1. *Let $f(x)$ be a continuous function on the interval $[a, b]$ we can split it into n equal sub-interval for each we have the width $\Delta x = \frac{b-a}{n}$ such that $a = x_0 < x_1 < \cdots < x_n = b$. The approximation of the integral of the function over the interval is given by:*

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}\left(\sum_{k=1}^{N-1} f(x_k) + \frac{f_N + f(x_0)}{2}\right) \tag{2.4}$$

Then if $f \in C^2([a, b])$ the computed error is:

$$E(f) = -\frac{(b-a)^3}{12} f''(\xi) \qquad \textit{with} \quad \xi \in (a, b) \tag{2.5}$$

2.2 Matrix-vector multiplication using the FFT

One of the most expensive operation dealing with matrices and vectors is the matrix-vector product. The algorithm to accelerate this product using the Fast Fourier

Transform algorithm (FFT), thanks to [8]. Using it we can compute the discrete Fourier transform in just $\mathcal{O}(n \log n)$ operator instead of $\mathcal{O}(n^2)$, the same property are for the inverse discrete Fourier transform. The definitions for the two transformations are as follows: Discrete Fourier transform (DFT):

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

And for the inverse DFT:

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

Now, we will understand the principle which makes this computational cost reduction possible. First of all, we can write the inverse DFT, using factorization of the coefficients as:

$$x_j = x_{j0,j_1} = \sum_{k=0}^{n-1} a_k e^{2\pi ikj/n} = \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} a_{k_0,k_1} e^{2\pi ik_1r_2j/n} e^{2\pi ik_0j/n}$$

For the sake of simplicity we call $R = e^{2\pi i/n}$ By this we can write the previous sum as:

$$x_{j0,j_1} = \sum_{k_0=0}^{r_2-1} R^{k_0j} \sum_{k_1=0}^{r_1-1} a_{k_0,k_1} R^{j_0k_1r_2}$$

Then, we can reformulate it as:

$$x_{j_0,j_1} = \sum_{k_0=0}^{r_2-1} R^{k_0(j_1r_1+j_0)} \tilde{a}_{j_0,k_0} \tag{2.6}$$

The defined the indexes $j_0 = 0, 1, \dots, r_1 - 1$ and $j_1 = 0, \dots, r_2 - 1$ in the same way we defined k_0 and k_1, where $r_1r_2 = n$. By this construction the $\tilde{a}$ has n elements, the we compute sum $\tilde{a}_{j_0,k_0}$ in r_1 operations, likewise x has size n. Thus we compute the array $\tilde{a}$ in r_1n and x by r_2n operations, eventually the amount be $T = (r_1 + r_2)n$. The conclusion is that for any integer factorization of n to m steps, we can reduce the total number of operation to $T = n\sum_1^m r_i$ and have $n = r^m$, so $m = log_r n$ which implies to $T = nr \log_r n$. A good choice for the factorization is $n = 2^m$ as stated by the paper we followed.

Now we can introduce the usage of the FFT to the matrix-vector multiplication. Firstly, two important structures are the Toeplitz matrix and the Circulant matrix. Let us start with the formal definition of a Toeplitz matrix:

Definition 6. *Let T be a matrix, it is called Toeplitz if T is determined by the $2n-1$ scalars $t_{-(n-1)}, \dots, t_{n-1}$ with $T_{ij} = t_{j-i}$ for all i and j.*

So we have the matrix T with this form:

$$A = \begin{pmatrix} t_0 & t_{-1} & t_2 & \cdots & \cdots & \cdots & \cdots & t_{n-1} \\ t_1 & t_0 & t_{-1} & t_2 & & & & \vdots \\ a_2 & t_1 & t_0 & t_{-1} & \ddots & & & \vdots \\ \vdots & t_2 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & t_2 & \vdots \\ \vdots & & & \ddots & t_1 & t_0 & t_{-1} & t_2 \\ \vdots & & & & t_2 & t_1 & t_0 & t_{-1} \\ t_{n-1} & \cdots & \cdots & \cdots & \cdots & t_2 & t_1 & t_0 \end{pmatrix}$$

One can see that for a generic Toeplitz matrix, it is sufficient just to storage the vector $[t_{n-1}, t_{n-2}, \ldots, t_{1-n}]^T \in \mathbb{R}^{2n-1}$ instead to storage the whole matrix. Note that for a Hermitian Toeplitz matrix we can save just the first columns. Then we can move to the Circulant matrix which is a special case of the Toeplitz matrix that is defined as:

Definition 7. *Let C be a matrix, it is called circulant if it is a Toeplitz matrix where each column is a circulant shift of its preceding column.*

Therefore for a Circulant matrix we have the form:

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

Additionally , we call the unitary and symmetric matrix F_n the Fourier matrix which has the form:

$$F_N = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \ldots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \ldots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \ldots & \omega_N^{(N-1)^2} \end{pmatrix} \tag{2.7}$$

where $\omega = e^{-2\pi i/n}$. An important relationship between FFT and circulant matrix is given by the following theorem:

Theorem 5. *Let $C_n \in \mathbb{R}^{n\times n}$ be a circulant matrix. It can be decomposed as:*

$$C_n = F_n^* \Lambda F_n$$

where $\Lambda = diag(\lambda_0, \ldots, \lambda_{n-1})$, indicating λ_j the eigenvalue of C_n.

Following [11], and using the above theorem, we can efficiently compute the matrix-product $x = C_n a = F_n^* \Lambda F_n a$. It can be done in four step:

1. $\hat{a} = F_n a$

2. $\hat{c} = \sqrt{n} F_n c$

3. $b = \hat{c} \odot \hat{a}$

4. $x = F_n^* b$

Where the symbol $\odot$ is the element-wise multiplication. Using these kinds of operations, we compute the matrix-vector product computing three FFT's and just one vector multiplication, which leads to $\mathcal{O}(n \log n)$ operations. Moreover, the matrix-vector product mo can be extended to Toeplitz matrices, since we can embed a Toeplitz matrix in a circulant one. Let T be a $n \times n$ matrix, for which we have to compute the matrix-vector multiplication Ty. Firstly, we embed the matrix into $2n \times 2n$ circulant matrix, which is:

$$Ty = \begin{pmatrix} I & 0 \end{pmatrix} A \begin{pmatrix} y \\ 0 \end{pmatrix}$$

Where the matrix A is defined as:

$$A = \begin{pmatrix} T & B \\ B & T \end{pmatrix}$$

where the matrix B has the following structure:

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

So using the form, we can compute the matrix-vector product with A, that is a circulant matrix, which can be performed using the FFT following the four steps described above.

Chapter 3

Numerical Approximations

This chapter is dedicated to find an approximation of the analytical problem for the PDE (1.17), this task will be carried out using finite differences and the schemes described in the previous chapter. First of all, we will present the three kinds of finite schemes for the Merton model in jump-diffusion PDE, for each type of scheme we will first use the discretization of the spatial derivatives in such a way we will obtain a punctual ODE in the time. Then we will use a finite difference scheme for the time domain; the schema will be named after the type of schema used for differentiation on the time. The scheme will be the following:

- Implicit Euler
- Crank-Nicolson
- Implicit-Explicit

3.1 Discretization Merton's PDE

In this section, we present two numerical procedures for solving the PDE of Merton jump-diffusion model (1.18). Notice that in the case we have $S_t = 0$, it leads to a degeneration at that point for the SDE defined in (1.14), according to [4] one of the condition for the linear integral-differential problem for the jump-diffusion model is that $\sigma(S,t) > 0 \quad \text{for all} \quad (S,t) \in \mathbb{R} \times [0,T]$. This leads that a good way to express the option price is to evaluate it in term of logarithmic value of the asset value S_t. By making the change of variables, we define:

- $x = \ln(S_t)$
- $y = \ln(\eta)$
- $\tau = T - t$
- $u(\tau, x) = V(T-t, e^x)$

Using these change of variables in the linear differential-integral operator defined in (1.16) we obtain:

$$\mathcal{L}V(\tau,x) = -\frac{1}{2}\sigma^2\frac{\partial^2 u}{\partial x^2} - \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\frac{\partial u}{\partial x} + (r+\lambda)u - \lambda\int_{-\infty}^{+\infty} u(\tau, x+y)J(y)dy \quad (3.1)$$

Where the probability density function of the jumps (1.15) is re-defined as:

$$J(y) = \frac{1}{\sqrt{2\pi}\gamma} e^{\frac{-(y-\mu)^2}{2\gamma^2}}$$

Therefore the Cauchy problem defined in (1.18) can be defined as:

$$\begin{cases} \dfrac{\partial u(\tau,x)}{\partial \tau} + \mathcal{L}V(\tau,x) = 0 & \forall (x,t) \in \mathbb{R} \times [0,T), \\ u(\tau,x) = \Phi(x) & \forall x \in \mathbb{R} \end{cases} \tag{3.2}$$

First of all, we have to define the boundary condition for the PDE (1.18), in fact for this Cauchy problem we have boundary condition for: European call option and European put option. Starting from European call option, we get:

$$\Phi(\tau,x) = \begin{cases} u(\tau,x) \to 0, & \text{when} \quad x \to -\infty \\ u(\tau,x) \to e^x - Ke^{-r\tau} & \text{when} \quad x \to +\infty \end{cases} \tag{3.3}$$

While for a put option the boundary condition are:

$$\Phi(\tau,x) = \begin{cases} u(\tau,x) \to Ke^{-r\tau}, & \text{when} \quad x \to -\infty \\ u(\tau,x) \to 0 & \text{when} \quad x \to +\infty \end{cases} \tag{3.4}$$

Then given the terminal conditions, the initial condition are evaluated as follows:

$$u(0,x) = \Phi(x) = \begin{cases} (e^x - K)^+, & \text{in the European call option} \\ (K - e^x)^+, & \text{in the European put option} \end{cases} \tag{3.5}$$

As one can see the x domain extends throughout $\mathbb{R}$, thus in order to evaluate numerically the solution of this PDE we have to delimit the relative C domain; thus we consider a finite domain region.

3.1.1 Definition of bounded spatial domain

For convenience the x variable will be called "spatial variable". First of all we have to truncate the space domain to a bounded domain $x \in (-x^*, x^*)$, where the vale x^* is has a finite value. As shown in Cont and Voltchkova's study (see, [7]) an extension the solution beyond the domain can be given by the payoff function, which is asymptotically close to the solution at the extremes of $\mathbb{R}$. Moreover, we can see the truncation error goes down exponentially with the dimension of the domain from the following proposition with the domain size $2x^*$. So we can define the solution u_{x^*} as the solution of the truncated problem which is localize in the restricted domain of $(-x^*, x^*)$. Therefore, imposing the payoff function as boundary conditions we can define the Cauchy problem on the truncated domain:

$$\begin{cases} \dfrac{\partial u_{x^*}}{\partial \tau} + \mathcal{L}u_{x^*} = 0 & (0,T] \times (-x^*, x^*) \\ u_{x^*}(0,x) = \Phi(x) & x \in (-x^*, x^*) \\ u_{x^*}(\tau,x) = h(\tau,x) & x \notin (-x^*, x^*) \end{cases} \tag{3.6}$$

where $h(\tau, x)$ is defined as:

$$h(\tau, x) = \begin{cases} 0 \\ \Phi(\tau, x) \end{cases}$$

Where $g(\tau, x)$ is the boundary conditions for call option as defined in (3.3) and put option in (3.4). Then as in [7], we can provide a proposition stating that the localization error goes down exponentially in the domain of size $2x^*$:

Proposition 2. *Let Φ be bounded, so $||\Phi||_\infty < \infty$ and $\exists \quad \alpha > 0$ such that*

$$\int_{|x|>1} e^{\alpha|x|} \nu dx < \infty \tag{3.7}$$

Let $u(\tau, x)$ and $u_{x^*}(\tau, x)$ be the solution for the Cauchy problems with boundary condition defined for $h(\tau, x)$, then:

$$|u(\tau, x) - u_{x^*}| \leqslant 2C_{\tau,\alpha} ||\Phi||_\infty e^{-\alpha(A-|x|)} \qquad \forall \in (-x^*, x^*)$$

where $C_{\tau,\alpha}$ is a constant not depending on x^*.

Proof. This proof is based on the probabilistic representation of equation's solution we are searching for. Let X_t be the Levy process, we define $M_\tau^x = sup_{t \in 0,\tau]} |X_t + x|$, then the probabilistic representation of the solutions are the following:

$$\begin{cases} u(\tau, x) &= \mathbb{E}[\Phi(X_\tau + x)] \\ u_{x^*}(\tau, x) &= \begin{cases} \mathbb{E}[\Phi(X_\tau + x)\mathbb{I}_{\{M_\tau^x < x^*\}}], & \text{if} \quad h(, x) = 0 \\ \mathbb{E}[\Phi(X_t + x)\mathbb{I}_{\{M_\tau^x < x^*\}} + \Phi(X_{\theta_x} + x)\mathbb{I}_{\{M_\tau^x \geq x^*\}}], & \text{if} \quad h(\tau, x) = \Phi(\tau, x) \end{cases} \end{cases}$$

where $\theta(x) = \inf\{\tau \geq 0, |X_t + x| \geq x^*|\}$ is the hitting time of the process $X_t + x$ from the domain $[-x^*, x^*]$. Subtracting the two solution we have for $h(\tau, x) = 0$:

$$\begin{aligned} |u(\tau, x) - u_{x^*}(\tau, x)| &= |\mathbb{E}[\Phi(X_\tau - x)\mathbb{I}_{\{M_\tau^x \geq x^*\}}]| \\ &\leqslant ||\Phi||_\infty \mathbb{Q}(M_\tau^x \geq x^*) \end{aligned}$$

while in the case of $h(\tau, x) = \Phi(\tau, x)$ we get:

$$\begin{aligned} |u(\tau, x) - u_{x^*}(\tau, x)| &= \mathbb{E}[\Phi(X_t + x)\mathbb{I}_{\{M_\tau^x < x^*\}} + \Phi(X_{\theta_x} + x)\mathbb{I}_{\{M_\tau^x \geq x^*\}}]| \\ &\leqslant \mathbb{E}|\Phi(X_t + x)\mathbb{I}_{\{M_\tau^x \geq x^*\}}| + \mathbb{E}|\Phi(X_{\theta(x)} + x)\mathbb{I}_{\{M_\tau^x \geq x^*\}}| \\ &\leqslant 2||\Phi||_\infty \mathbb{Q}(M_\tau^x \geq x^*) \end{aligned}$$

We can note that in both cases:

$$|u(\tau, x) - u_{x^*}(\tau, x)| \leqslant 2||\Phi||_\infty \mathbb{Q}(M_\tau^x \geq x^*) \tag{3.8}$$

From theorem 25.18 in [14] and with the definition of (3.7) we get:

$$C_{\tau,\alpha} = \mathbb{E}e^{\alpha M_\tau^0} < \infty$$

Then by Chebyshev's inequality we have:

$$\mathbb{Q}(M_\tau^0 \geq x^*) \leqslant C_{\tau,\alpha} e^{-\alpha x^*}$$

Then, to get from M_τ^0 to M_τ^x we follow the implication below:

$$\begin{aligned} \sup|X_t + x| &\leqslant \sup|X_t| + |x| \\ \mathbb{Q}(M_\tau^x \geq x^*) &\leqslant \mathbb{Q}(M_\tau^0 \geq x^* - |x|) \\ &\leqslant C_{\tau,\alpha} e^{-\alpha(x^*-|x|)} \end{aligned}$$

Then using this on (3.8) we have the desired result. □

The above proposition implies that for any point not too much close to the boundary such that $|x| \leq (1-\delta)x^*$ where $0 \leq \delta \leq 1$ the localization error decreases with the domain size:

$$|u(\tau, x) - u_{x^*}(\tau, x)| \leqslant C e^{-\alpha\delta x^*}$$

3.1.2 Approximation of the integral term

Next step is to approximate the integral operator in the linear operator (3.1) which is defined as:

$$I(u(\tau, x)) = \lambda \int_{\mathbb{R}} J(y) u(\tau, x+y) dy \tag{3.9}$$

In order to compute it numerically we have to truncate the integral domain. Firstly, we assume and consider finite activity which means that given an interval almost all paths of the Levy process defining the jump have a finite number of jumps. We have defined the spatial domain as a truncation for $\mathbb{R}$, in fact now we have $\Omega = (-x^*, x^*)$; note that it is not sufficient to impose the payoff function as boundary condition since the integral operator is a non-local operator, therefore to compute the integral we need to extend the function beyond the domain on set $\Omega^c = (-\infty, -x^*) \cup (x^*, \infty)$. To estimate the integral on Ω^c we use the extend the boundary condition of the Cauchy problem, so we have for the European call option:

$$\begin{cases} u(\tau, x) \to 0, & \text{when} \quad x \to (-\infty, -x^*) \\ u(\tau, x) \to e^x - Ke^{-r\tau} & \text{when} \quad x \to (x^*, \infty) \end{cases} \tag{3.10}$$

While for a put option the boundary conditions become:

$$\begin{cases} u(\tau, x) \to Ke^{-r\tau}, & \text{when} \quad x \to (-\infty, -x^*) \\ u(\tau, x) \to 0 & \text{when} \quad x \to (x^*, \infty) \end{cases} \tag{3.11}$$

After the definition of the integral operator's domain and its boundary condition we can start to approximate it. By making the change of variable $y = z - x$ we obtain:

$$I(u(\tau, x)) = \lambda \int_{\mathbb{R}} J(y) u(\tau, x+y) dy = \int_{\mathbb{R}} u(\tau, z) J(z-x) dz$$

Under the continuity assumptions of the function $u(\tau, x)$ we can use the addictivity property of integral in order to split the integral domain so we have:

$$\begin{aligned} \int_{\mathbb{R}} u(\tau, z) J(z-x) dz = &\int_{-\infty}^{x^*} u(\tau, z) J(z-x) dz \\ &+ \int_{-x^*}^{x^*} u(\tau, z) J(z-x) dz + \int_{x^*}^{+\infty} u(\tau, z) f(z-x) dz \end{aligned}$$

Now to approximate the integral lying on the interval $\Omega* = (-x^*, x^*)$, we are going to discretize the interval which becomes $-x^* = x_0 < x_1 < \cdots < x_{n-1} < x_n = x^*$ with spatial interval $\Delta x = (x_n - x_0)/n$. Then we are going to use the composite trapezoidal rule (2.4) on that interval considering the spatial point $x \in (-x^*, x^*)$, so we obtain the approximation for the integral:

$$\int_{-x^*}^{x^*} u_{\tau,z} J(z-x)dz \approx \frac{\Delta x}{2}\biggl(J(x_0 - x)u_{\tau,x_0} \\ + J(x_N - x)u_{\tau,x_N} + 2\sum_{k=1}^{N-1} J(x-x_k)u_{\tau,x_k}\biggr) \tag{3.12}$$

For the other two intervals, since they depends on the boundary conditions, we are going to consider the European call option boundary conditions. Therefore, using the (3.10) we can approximate the integral over the set $\mathbb{R}\backslash\Omega^*$ as follows:

$$\begin{aligned}\mathcal{I}_{call}(\tau, x, x^*) &= \int_{-\infty}^{x^*} u(\tau,z)J(z-x)dz + \int_{x^*}^{+\infty} u(\tau,z)J(z-x)dz \\ &\approx \int_{x^*}^{+\infty} (e^z - Ke^{-r\tau})J(z-x)dz \\ &= \int_{x^*}^{+\infty} (e^z - Ke^{-r\tau})\frac{1}{\sqrt{2\pi}\gamma} e^{-\frac{(z-x-\mu_j)^2}{2\gamma^2}} dz \\ &= \frac{1}{\sqrt{2\pi}\gamma}\biggl(\int_{x*}^{+\infty} e^{z-\frac{(z-x-\mu_j)^2}{2\gamma^2}} dz - Ke^{-r\tau}\int_{x^*}^{+\infty} e^{-\frac{(z-x-\mu_j)^2}{2\gamma^2}} dz\biggr)\end{aligned}$$

Then since for the Merton's model $\mu_j = 0$ we obtain:

$$\mathcal{I}_{call}(\tau, x, x^*) = \frac{1}{\sqrt{2\pi}\gamma}\biggl(\int_{x*}^{+\infty} e^{z-\frac{(z-x)^2}{2\gamma^2}} dz - Ke^{-r\tau}\int_{x^*}^{+\infty} e^{-\frac{(z-x-)^2}{2\gamma^2}} dz\biggr)$$

By making a change of variable $\xi = \frac{x-z+\gamma^2}{\gamma}$, so $-\gamma d\xi = dz$, we have:

$$\begin{aligned}\mathcal{I}_{call}(\tau, x, x^*) &= \frac{1}{\sqrt{2\pi}\gamma}\biggl(e^{x+\frac{\gamma^2}{2}}\int_{x^*}^{\infty} e^{-\frac{(x-z+\gamma^2)^2}{2\gamma^2}} dz - Ke^{-r\tau}\int_{x^*}^{+\infty} e^{-\frac{(z-x-)^2}{2\gamma^2}} dz\biggr) \\ &= \frac{1}{\sqrt{2\pi}\gamma}\biggl(e^{x+\frac{\gamma^2}{2}}\int_{-\infty}^{\frac{x-x^*+\gamma^2}{\gamma}} e^{-\frac{\xi^2}{2}} d\xi + Ke^{r\tau}\int_{-\infty}^{\frac{x-x^*}{\gamma}} e^{-\frac{\xi^2}{2}} d\xi\biggr)\end{aligned} \tag{3.13}$$

One might see that the integrated functions are equal and moreover they are the normal probability distribution, so using the cumulative normal distribution function:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{\xi^2}{2}} d\xi \tag{3.14}$$

So the integral in (3.13) becomes:

$$\mathcal{I}_{call}(\tau, x, x^*) = e^{x+\frac{\gamma^2}{2}}\Phi\Bigl(\frac{x-x^*+\gamma^2}{\gamma}\Bigr) - Ke^{-r\tau}\Phi\Bigl(\frac{x-x^*}{\gamma}\Bigr) \tag{3.15}$$

Adding up the equations (3.12) and (3.15) we obtain the approximation of the integral over $\mathbb{R}$ getting the formulation for the European call option:

$$I(u(\tau,x)) = \lambda\mathcal{I}_{call}(\tau,x,x^*) - \frac{\Delta x}{2}\left(J(x_0-x)u_{\tau,x_0} + J(x_N-x)u_{\tau,x_N} + 2\sum_{k=1}^{N-1} J(x-x_k)u_{\tau,x_k}\right)$$

In the same way we can get the approximation of the integral in the case of European put option. In fact in the case of European put option, the integral in the set Ω^* will be the same as the equation (3.12), while for the integral in the complementary set Ω^c, as the previous case of European call, we need to extend the boundary condition in (3.11) becoming:

$$\begin{aligned}
\mathcal{I}_{put}(\tau,x,x^*) &= \int_{-\infty}^{x^*} u(\tau,z)J(z-x)dz + \int_{x^*}^{+\infty} u(\tau,z)J(z-x)dz \\
&\approx \int_{-\infty}^{-x^*} (Ke^{-r\tau} - e^z)J(z-x)dz \\
&= \int_{-\infty}^{-x^*} (Ke^{-r\tau} - e^z)\frac{1}{\sqrt{2\pi}\gamma}e^{-\frac{(z-x-\mu_j)^2}{2\gamma^2}}dz \\
&= \frac{1}{\sqrt{2\pi}\gamma}\left(\int_{-\infty}^{-x^*} Ke^{-r\tau}dz - e^{z-\frac{(z-x-\mu_j)^2}{2\gamma^2}}\int_{-\infty}^{-x^*} e^{-\frac{(z-x-\mu_j)^2}{2\gamma^2}}\right)dz
\end{aligned}$$

As we did in the European call option $\mathcal{T}_{call}(\tau,x,x^*)$ we assume for the Merton's model $\mu_j = 0$, then by making the change of variable $\xi = (x - z + \gamma^2)/\gamma$, so $-\gamma d\xi = dz$, and using the cumulative norm distribution function we obtain the following definition:

$$\mathcal{I}_{put}(\tau,x,x^*) = Ke^{-r\tau}\Phi\left(\frac{-x^*-x}{\gamma}\right)$$

For ease of writing we define $J(x_j - x_i) = J_{j,i}$ and eventually we have the numerically approximation of the integral term which can be generally written at the spatial point x_i as:

$$I(u(\tau,x_i)) = \lambda\mathcal{I}(\tau,x_i,x^*) - \frac{\lambda\Delta x}{2}\left(J_{i,0}u_0 + J_{i,N} + 2\sum_{k=1}^{N-1} J_{i,k}u_k\right) \tag{3.16}$$

where:

$$\mathcal{I}(\tau,x_i,x^*) = \begin{cases} \mathcal{I}_{call}(\tau,x_i,x^*), & \text{in the case of European call option} \\ \mathcal{I}_{put}(\tau,x_i,x^*), & \text{in the case of European put option} \end{cases}$$

3.2 Finite differences: Implicit Euler

In this section we are going to perform a efficient space integration tool and then we will be looking for a time integration tool. We start with finite difference discretization of the spatial derivatives.

By the construction of Merton's model we can observe the linear differential-integral operator in (3.1) as a composition of:

- linear differential operator
- integral operator

Starting from the linear differential operator it has the following formulation:

$$L^{diff}u = -\frac{1}{2}\sigma^2\frac{\partial^2 u}{\partial x^2} - \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\frac{\partial u}{\partial x} + (r+\lambda)u \tag{3.17}$$

Here below we are going to describe the discretization on the spatial domain of that linear operator, to do that we are going to truncate the domain of $\mathbb{R}$, which defines the spatial domain x thank to the section above about the definition of bounded spatial domain. In fact, the domain we are considering is $\Omega^* = (-x^*, x^*)$; on this domain we will consider an uniform domain of x, hence $-x^* = x_0 < x_1 < \cdots < x_{N-1} < x_N = x^*$. The spatial derivatives are going to be approximated with the formulation on section (2.1.4), using the central finite differences we can write derivatives into play, considering them on the time point τ, as: First order derivative:

$$\frac{\partial u}{\partial x}(\tau, x_i) \approx \frac{u_{i+1}(\tau) - u_{i-1}(\tau)}{2\Delta x}$$

Second order derivative:

$$\frac{\partial^2 u}{\partial x^2}(\tau, x_i) \approx \frac{u_{i+1}(\tau) - 2u_i + u_{i-1}(\tau)}{\Delta x^2}$$

Given $\mathbf{u}(\tau)$ defined as:

$$\mathbf{u}(\tau) = \begin{pmatrix} u_1(\tau) \\ u_2(\tau) \\ \vdots \\ u_{N-2}(\tau) \\ u_{N-1}(\tau) \end{pmatrix}$$

defining $\Delta x = x_{i+1} - x_i$, $u_i(\tau) = u(\tau, x_i)$, then follows that:

$$\begin{aligned} L^{\text{diff}}u = &-\frac{1}{2}\sigma^2\Big(\frac{u_{i+1}(\tau) - 2u_i + u_{i-1}(\tau)}{\Delta^2 x}\Big) \\ &- \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Big(\frac{u_{i+1}(\tau) - u_{i-1}(\tau)}{2\Delta x}\Big) + (r+\lambda)u_i(\tau) \end{aligned}$$

Summing up all element of $i = 1, \dots, N-1$ we get the matrix formulation, which will bring to a tridiagonal matrix where we can denote it as T. We will denote the matrix as T whose of-diagonal elements are defined as:

$$T_{i,i-1} = \frac{-\sigma^2 + \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Delta x}{2\Delta x^2} \qquad T_{i,i+1} = \frac{-\sigma^2 - \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Delta x}{2\Delta x^2}$$

While for the principal diagonal:

$$T_{i,i} = r + \lambda - T_{i,i-1} - T_{i,i+1}$$

By the fact that with $i = 1$ and $i = N - 1$ there are the presence of the boundary elements respectively u_0 and u_N, in these point we have the known value add to the matrix; thus the matrix form of the linear differential operator will become:

$$L^{\text{diff}}\mathbf{u}(\tau) = T\mathbf{u}(\tau) + \mathbf{k} \tag{3.18}$$

where as stated above we have:

$$\mathbf{k}_1 = \Big(-\frac{\sigma^2}{2\Delta x^2} + \frac{1}{2\Delta x}\Big(r - \frac{1}{\sigma^2} - \lambda\kappa\Big)\Delta x\Big)u_0$$
$$\mathbf{k}_N = -\Big(\frac{\sigma^2}{2\Delta x^2} + \frac{1}{2\Delta x}\Big(r - \frac{1}{\sigma^2} - \lambda\kappa\Big)\Delta x\Big)u_N$$

Elsewhere, for $i = 2, 3, \ldots, N-2$ it will assume values $\mathbf{k}_i = 0$. From section (2.1.4) we can conclude that the finite difference discretization for the linear differential operator $\mathcal{L}^{\text{diff}}\mathbf{u}$ is an approximation of second order with accuracy of $\mathcal{O}(\Delta x^2)$. Let $r \geq 0$, for a sufficiently small spatial interval Δx the off-diagonal elements of T are non-positive so the matrix will be a M-matrix, but if the spatial interval fulfills $\Delta x > \sigma^2/|r - \frac{1}{2}\sigma^2 - \lambda\kappa|$, the matrix may have positive off-diagonal elements so it can be a M-matrix. M-matrices can be defined in the following way:

Definition 8. *Let $A \in \mathbb{R}^{n\times n}$ is called an M-matrix if it fulfills the following properties:*

- *the main diagonal values are all positive*
- *the off-diagonal values are all non-positive*
- *A is non-singular*
- *A^{-1} is non-negative*

To avoid this we can use an artificial diffusion into the model as was done in [22] in order to keep the M-matrix properties of the matrix T. Of course this evaluation often causes an order reduction of accuracy since including artificial volatility is equivalent to use a combination of central finite difference and an one-sided finite difference for the first-order spatial derivative. Now, we are going to add that artificial volatility such that the volatility of the model becomes:

$$\hat{\sigma}^2 = \max\Big\{\sigma^2, \Big(r - \frac{1}{2}\sigma^2 - -\lambda\kappa\Big)\Delta x, -\Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Delta x\Big\} \tag{3.19}$$

Hence the off-diagonal elements of T will be written with this form:

$$\hat{T}_{i,i-1} = \frac{-\hat{\sigma^2} + \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Delta x}{2\Delta x^2} \quad \hat{T}_{i,i+1} = \frac{-\hat{\sigma^2} - \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Delta x}{2\Delta x^2} \tag{3.20}$$

and then the diagonal element assumes this formulation:

$$\hat{T}_{i,i} = r + \lambda - \hat{T}_{i,i-1} - \hat{T}_{i,i+1} \tag{3.21}$$

Adding this artificial volatility to the model we define the matrix T with specific features as we can understand from the theorem from [23] which states:

Theorem 6. *Let r be the interest rate such that $r > 0$ and let T be the tridiagonal matrix whose elements are defined in (3.20) and (3.21). Then the matrix $T - \lambda\mathbf{I}$ is a strictly diagonally dominant M-matrix such that:*

$$\sum_k (T - \lambda\mathbf{I})_{i,k} > 0 \quad \textit{and} \qquad T_{i,k} \leqslant 0 \quad \forall k \neq i \quad \textit{with} \quad i = 1, \dots, n-1$$

Proof. Firstly we are going to show that the off-diagonal elements $\hat{T}_{i,i-1}$ defined in (3.20) and (3.21) are not positive. By the definition of the elements $\hat{T}_{i,i-1}$ and the definition of the artificial volatility in (3.19) we obtain:

$$\begin{aligned}\hat{T}_{i,i-1} &= \frac{-\max\left\{\sigma^2, \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x, -\left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x\right\} + \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2} \\ &= \frac{-\max\left\{\sigma^2 + \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x, 2\left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x, 0\right\}}{2\Delta x^2}\end{aligned}$$

This formulation leads to the fact that for any value of $\sigma \geq 0$, $\lambda \geq 0$ and $\kappa \geq 0$ the element will be:

$$T_{i,i-1} = \frac{-\max\left\{\sigma^2 + \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x, 2\left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x, 0\right\}}{2\Delta x^2} \leqslant 0$$

In the same way we can prove that the off-diagonal element $T_{i,i+1}$ is non-positive, consequently from the definition of $T_{i,i}$ it is easy to see that the matrix $T - \lambda\mathbf{I}$ has a positive diagonal. Moreover, it is strictly diagonal dominant matrix because of:

$$\sum_k (T - \lambda\mathbf{I})_{k,j} \geq r > 0$$

□

Moving on the integral operator, we can extend the formulation on the equation (3.12) on every spatial points, hence using the matrix representation we have the following form for the integral operator:

$$L^{\text{int}}\mathbf{u} = J\mathbf{u} + \mathbf{f} \tag{3.22}$$

where:

$$\mathbf{f}(\tau) = \begin{pmatrix} -\lambda\mathcal{I}(\tau, x_1, x^*) - \frac{\lambda\Delta x}{2}\left(J_{1,0}u_0 + J_{1,N}u_N\right) \\ -\lambda\mathcal{I}(\tau, x_2, x^*) - \frac{\lambda\Delta x}{2}\left(J_{2,0}u_0 + J_{2,N}u_N\right) \\ \vdots \\ -\lambda\mathcal{I}(\tau, x_{N-2}, x^*) - \frac{\lambda\Delta x}{2}\left(J_{N-2,0}u_0 + J_{N-2,N}u_N\right) \\ -\lambda\mathcal{I}(\tau, x_{N-1}, x^*) - \frac{\lambda\Delta x}{2}\left(J_{N-1,0}u_0 + J_{N-1,N}u_N\right) \end{pmatrix}$$

and the J is a matrix defined as:

$$J = -h\lambda \begin{pmatrix} J_{1,1} & J_{1,2} & \cdots & J_{1,N-2} & J_{1,N-1} \\ J_{2,1} & J_{2,2} & \cdots & J_{2,N-2} & J_{2,N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ J_{N-2,1} & J_{N-2,2} & \cdots & J_{N-2,N-2} & J_{N-2,N-1} \\ J_{N-1,1} & J_{N-1,2} & \cdots & J_{N-1,N-2} & J_{N-1,N-1} \end{pmatrix}$$

As we can see J is assembled with the probability density function of the jump, thus the more the two points considered are distant the smaller is the value of the probability this means that the matrix values decay from the principal diagonal. As the previous matrix T, J has special features as defined in the theorem below stated in [23]:

Theorem 7. *Let Δx satisfies the condition $\Delta x \leq 1$, then the matrix $(J + \lambda\mathbf{I})$ is a diagonally dominant Z-matrix with a non-negative diagonal hence:*

$$\sum_k (J + \lambda\mathbf{I})_{i,k} \geq 0, \qquad \textit{and} \quad J_{i,k} \leqslant 0 \quad \forall k \neq i \quad \textit{for} i = 1, \ldots, N-1$$

Proof. By the fact that $\lambda \geq 0$ and given two spatial point x_i and x_k the probability density function $J(x_i - x_k) \geq 0$ in fact it has defined as Z-matrix.Then to prove $\sum_k (J + \lambda\mathbf{I})_{i,k} \geq 0$ for $i = 1, \ldots, N-1$ we note that:

$$\sum_k (J + \lambda\mathbf{I})_{i,k} = \lambda - \frac{\lambda\Delta x}{\sqrt{2\pi}\gamma} \sum_k e^{-\frac{(x_k - x_i - \mu)^2}{2\gamma^2}}$$

Let $(x_j - x_i) = \Delta x \to 0$, we can write the summation as an integral, thus we obtain:

$$\begin{aligned} \lambda - \frac{\lambda\Delta x}{\sqrt{2\pi}\gamma} \sum_k e^{-\frac{(x_k - x_i - \mu)^2}{2\gamma^2}} &= \lambda - \frac{\lambda\Delta x}{\sqrt{2\pi}\gamma} \int_{-x^*}^{x^*} e^{-\frac{(x_k - x_i - \mu)^2}{2\gamma^2}} dx \\ &= \lambda - \frac{\lambda\Delta x}{\sqrt{2\pi}\gamma} \sqrt{2\pi}\gamma \geq 0 \end{aligned}$$

whenever $\Delta x \leqslant 1$. □

Adding up the two operator that we have defined as: linear differential operator in (3.18) and the integral operator (3.22) we obtain in the matrix form:

$$\begin{aligned} L\mathbf{u}(\tau) &= L^{diff}\mathbf{u} + L^{int}\mathbf{u} \\ &= (L^{diff} + L^{int})\mathbf{u} \\ &= (\hat{T} + J)\mathbf{u} + \mathbf{b} \end{aligned}$$

Where $\mathbf{b} = \mathbf{k} + \mathbf{f}$. It comes up that we obtain a semi-discretize system of the PDE, which in matrix form is:

$$\frac{\partial \mathbf{u}}{\partial \tau} + L\mathbf{u}(\tau) = \mathbf{b}(\tau) \qquad \tau \in (0, T] \tag{3.23}$$

As a consequence of the two theorems (Theorem 6) and (Theorem 7) we obtain the following corollary :

Corollary 1. *Given $\Delta x \leq 1$ and given the interest rate $r > 0$, the matrix $\hat{T} + J$ is a strictly diagonally dominant M-matrix.*

Proof. It is a consequence of theorems: (6) and (7). □

After the discretization on the spatial domain of the spatial derivatives term and then the approximation of the integral term, we have a semi-discrete linear problem (3.24) which could be handled as an ODE. To discretize it we use the implicit Euler's method as defined above in (2.2) so we obtain:

$$\frac{\mathbf{u}(\tau) - \mathbf{u}(\tau - \Delta\tau)}{\Delta\tau} + L\mathbf{u}(\tau) = \mathbf{b}(\tau)$$

For simplicity we write $u_j^k = u_j(\tau - \Delta\tau)$ where $j = 1, \ldots, N-1$ defining the number of spatial point, while L_j represents the relative matrix row as well as for $\mathbf{b}_j^k$ is the relative element of $\mathbf{b}$, we get:

$$\mathbf{u}_j^{k+1} + \Delta\tau L_j \mathbf{u}_j^{k+1} = u_j^k + \Delta\tau \mathbf{b}_j^k$$

Using some easy math, the previous equation will be rewritten for the relative row j as:

$$(1 + \Delta\tau L(j))\mathbf{u}_j^{k+1} = \mathbf{u}_j^k + \Delta\tau \mathbf{b}(\tau) \tag{3.24}$$

Therefore in matrix form we will have:

$$(\mathbb{I} + \Delta\tau L)\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\tau \mathbf{b}(\tau) \tag{3.25}$$

The last linear system is the linear system evaluated at the time τ, hence starting from $\tau = 0$ we resolve m linear system until $\tau = T$ since $\Delta\tau = \tau/m$. The linear system can be defined in the form

$$A\mathbf{u}^{k+1} = \hat{\mathbf{b}} \tag{3.26}$$

where

$$A = \mathbb{I} + \Delta\tau L \qquad \text{and} \qquad \hat{\mathbf{b}} = \mathbf{u}^k + \Delta\tau \mathbf{b}(\tau)$$

moreover the model has a full matrix. The linear system's matrix A has the following properties:

- Toeplitz matrix
- Strictly diagonally dominant
- M-matrix

Let us prove these properties:

Toeplitz property: Starting from the fact that the matrix T is tridiagonal and whose elements are defined by (3.20) and (3.21), it is defined Toeplitz by construction. The other matrix J is also toeplitz by the fact:

$$J_{i,j} = \frac{1}{\sqrt{2\pi}\gamma} e^{-\frac{(x_i - x_j)^2}{2\gamma^2}}$$

Moving along the diagonal the element becomes:

$$\begin{aligned} J_{i+1,j+1} &= \frac{1}{\sqrt{2\pi}\gamma} e^{-\frac{(x_{i+1}-x_{j+1})^2}{2\gamma^2}} \\ &= \frac{1}{\sqrt{2\pi}\gamma} e^{-\frac{(x_i+\Delta x-x_j-\Delta x)^2}{2\gamma^2}} \\ &= \frac{1}{\sqrt{2\pi}\gamma} e^{-\frac{(x_i-x_j)^2}{2\gamma^2}} \end{aligned} \tag{3.27}$$

Therefore the two elements along the diagonal have the same value, hence $J_{i,j} = J_{i+1,j+1}$ which is the definition of Toeplitz matrix. Adding up the two matrices (T) and (J) the L matrix is still a Toeplitz matrix. Then it is easy to prove that the final matrix $A = \mathbb{I} + \Delta\tau L$ keeps the Toeplitz property of the two matrices $\mathbb{I}$ and $\Delta\tau L$.

Strictly diagonally dominant property: This matrix's feature is easily proven by the fact that the matrix $\mathbb{I}$ is by definition strictly diagonally dominant and the matrix L is a sum of strictly diagonal matrix.

M-matrix property: As the previous property, since the matrix $\mathbb{I}$ is by definition a M-matrix and by corollary (1), it follows that A is also an M-matrix.

3.2.1 Consistency

In this subsection we to study the consistency property of the implicit Euler scheme presented above.

Proposition 3 (Consistency). *The finite difference scheme (3.24) is locally consistent with equation (3.2), moreover:* $\forall\ u \in C_0^\infty([0,T]\times\mathbb{R})$ *and* $\forall(\tau_n; x_i) \in [0,T]\times\mathbb{R}$ *we have:*

$$\left|\frac{u_j^{n+1}-u_j^n}{\Delta\tau} + (Lu)_j^{n+1} - \left(\frac{\partial u}{\partial\tau} + \mathcal{L}u\right)(\tau_n, x_j)\right| = r_j^n(\Delta\tau, \Delta x^2) \to 0,$$

when $(\Delta\tau, \Delta x) \to 0$. *Moreover:*

$$\exists \quad c > 0, \qquad |r_j^n(\Delta\tau,\Delta x)| \leqslant c(\Delta\tau, \Delta x^2)$$

Proof. Considering the first term and the first derivative of the solution in the time, in the absolute value, we are going to use a second order Taylor series to obtain:

$$\begin{aligned} \Big|\frac{u_j^{n+1}-u_j^n}{\Delta\tau} - \frac{\partial u}{\partial\tau}(\tau_j, x_j)\Big| &= \Big|\frac{u_j^{n+1}-u_j^n}{\Delta\tau} - \frac{u_j^{n+1}-u_j^n}{\Delta\tau} - \frac{\Delta\tau}{2}\frac{\partial^2 u}{\partial\tau^2}(\tau_n, x_j)\Big| \\ &= \Big|\frac{\Delta\tau}{2}\frac{\partial^2 u}{\partial\tau^2}(\tau_n, x_j)\Big| \\ &\leqslant \frac{\Delta\tau}{2}\Big|\Big|\frac{\partial^2 u}{\partial\tau^2}\Big|\Big|_\infty \end{aligned}$$

Now considering the integral term in $(Lu)_j^{n+1}$ called $(Ju)_j^{n+1}$, while in the linear operator $\mathcal{L}u(\tau_n, x_j)$ the integral term we call it $(\mathcal{J}u)_j^n$, thus we have:

$$\begin{aligned}
\Big|(Ju)_j^{n+1} - (\mathcal{J}u)(\tau_n, x_j)\Big| &= \Big|(Ju)_j^{n+1} - (\mathcal{J}u)(\tau_{n+1}, x_j) + \Delta t(\mathcal{J}u)(\hat{\tau}, x_j)\Big| \\
&= \Big|\frac{x_{j+1} - x_j}{2}\Big(u(\tau_n, x_j)J(x_j - x_i) \\
&\quad + u(\tau_n, x_{j+1})J(x_{j+1} - x_i)\Big) \\
&\quad - \int_{x_j}^{x_{j+1}} u(\tau_{n+1}, z)J(z - x_i)dz \\
&\quad + \Delta t(\mathcal{J}u)(\hat{\tau}, x_j)\Big|
\end{aligned}$$

We already know the local truncation error of the trapezoidal rule from(2.5), so we can write the equation above as follows:

$$\Big| - \frac{\Delta x^3}{12}\frac{\partial u^2}{\partial x}(\xi) + \Delta t(\mathcal{J}u)(\hat{\tau}, x_j)\Big| \leqslant C_1\Delta t + C_2\Delta x^3$$

Then for the spatial derivatives part, we have $(Du)_j^{n+1}$ for the finite difference part, while for the derivative part we have the differential part of linear operator $\mathcal{L}u$, which is can be written by $(\mathcal{D}u)_j^{n+1}$ we have:

$$\begin{aligned}
|(Du)_j^{n+1} - (\mathcal{D}u)(\tau_n, x_j)| &= |(Du)_j^{n+1} - (\mathcal{D}u)(\tau_{n+1}, x_j) + \Delta\tau(\mathcal{D}u)(\hat{\tau}, x_j)| \\
&= \Big|\frac{\sigma^2}{2}\Big(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} - \frac{\Delta x^2}{12}\frac{\partial^4 u}{\partial x^4}(\tau_{n+1}, x_j)\Big) \\
&\quad + (r - \frac{1}{2}\sigma^2 - \lambda\kappa)\Big(\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} - \frac{\Delta x^2}{6}\frac{\partial^2 u}{\partial x^3}(\tau_{n+1}, x_j)\Big) \\
&\quad - (\mathcal{D}u)(\tau_{n+1}, x_j) + \Delta\tau(\mathcal{D}u)(\hat{\tau}, x_j)\Big| \\
&\leqslant C_3\Delta\tau + C_4\Delta x^2
\end{aligned}$$

Therefore these inequality lead to:

$$r_j^n(\Delta, \Delta x) \leqslant C_i\Delta\tau + C_m\Delta x^2 \to 0$$

□

Since backward Euler stability is a well-known result from the literature (see, [19] for more details), we can state that the scheme shown is unconditionally stable for any choice of $\Delta\tau$ and Δx, which means the stability zone for the equation in (3.24) is always included in the stability zone of the numerical method, regardless of the time step Δt and also Δx. Now the main difficulty of solving the linear systems as (3.25) it that the linear system matrix is full.

3.3 Finite differences: Crank-Nicolson scheme

Since the scheme being used in the previous section is a first order-convergence in time, in this section we are going to perform a second order-convergence scheme. To do this, we are going to use Crank-Nicolson's method over time domain. Starting from equation (3.24), we apply the Crank-Nicolson scheme defined in (2.6) in order to get the equation:

$$\frac{\mathbf{u}(\tau+\Delta\tau)-\mathbf{u}(\tau)}{\Delta t} = -\frac{1}{2}\Big[L\mathbf{u}(\tau+\Delta\tau)-\mathbf{b}(\tau+\Delta\tau)+L\mathbf{u}(\tau)-\mathbf{b}(\tau)\Big]$$

For simplicity we define:

$$\mathbf{u}^{n+1}=\mathbf{u}(\tau+\Delta\tau) \qquad \mathbf{b}^{n+1}=\mathbf{b}(\tau+\Delta\tau)$$

using these annotations we can write the linear system in order to get the unknown $\mathbf{u}^{n+1}$ as follows:

$$\mathbf{u}^{n+1}+\frac{\Delta\tau}{2}L\mathbf{u}^{n+1}=\mathbf{u}^{n}-\frac{\Delta\tau}{2}L\mathbf{u}^{n}+\frac{\Delta\tau}{2}\mathbf{b}^{n+1}+\frac{\Delta\tau}{2}\mathbf{b}^{n}$$

The we will get the following matrix form:

$$\Big(\mathbb{I}+\frac{\Delta\tau}{2}L\Big)\mathbf{u}^{n+1}=\Big(\mathbb{I}-\frac{\Delta\tau}{2}L\Big)\mathbf{u}^{n}+\frac{\Delta\tau}{2}\mathbf{b}^{n+1}+\frac{\Delta\tau}{2}\mathbf{b}^{n}$$

Then we can define:

$$A\mathbf{u}^{n+1}=\hat{\mathbf{b}} \tag{3.28}$$

where:

$$A=\mathbb{I}+\frac{\Delta\tau}{2}L$$

and

$$\hat{\mathbf{b}}=\Big(\mathbb{I}-\frac{\Delta\tau}{2}L\Big)\mathbf{u}^{n}+\frac{\Delta\tau}{2}\mathbf{b}^{n+1}+\frac{\Delta\tau}{2}\mathbf{b}^{n}$$

The linear system's matrix holds all the properties of the matrix in the linear system (3.26). In fact, it is:

- Toeplitz matrix
- Strictly diagonally dominant
- M-matrix

These properties, for the matrix A, can be easily proven by the fact that the matrix L in (3.28) is divided by the coefficient 2, which holds the properties of the previous L; therefore the matrix A in (3.28) has the same properties of A of the linear system (3.26). Since the vector $\hat{\mathbf{b}}$ has the first element defined as multiplication of a Toeplitz matrix and a vector, this operation can be sped up using the algorithm described in (2.2). The consistency of the Crank-Nicolson method is known in the literature, indeed it has a second-order accuracy in time by the construction of the scheme itself. While for accuracy order in the spatial variable, since we used the finite difference in the same way we used them in the previous method (Implicit Euler), it has a second order accuracy. Lastly, the scheme is unconditionally stable as stated in [10].

3.4 Finite differences: Implicit-Explicit scheme

As we can see from the previous methods, due to the integral operator, they define a full matrix A; which has to be inverted at each time step. The simple idea in this chapter is to define a method in order to avoid the inversion of the full matrix. The idea behind the next numerical method is based on splitting the $\mathcal{L}$ linear differential-integral operator in (3.1) into two elements employing an IMEX-Euler scheme, in the following form:

$$\frac{\partial u}{\partial \tau} + \mathcal{L}u = 0 \implies \frac{\partial u}{\partial \tau} + \mathcal{T}u + \mathcal{J}u = 0$$

where the new two operators are respectively:

1. $\mathcal{T}$ differential operator
2. $\mathcal{J}$ integral operator

Then we are going to approximate $\mathcal{T}u$ using finite difference approximation, while $\mathcal{J}u$ using the approximation defined in (3.16). In this method, we are going to treat the integral operator in an explicit time stepping order in that way it leads to avoid the inversion of the full-dense matrix J. For this model, we are going to assume that the number of activity case is finite.
The two operators are defined as follows:

$$\mathcal{T}u = -\frac{1}{2}\sigma^2\frac{\partial^2 u}{\partial x^2} - \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\frac{\partial u}{\partial x}$$
$$\mathcal{J}u = (r+\lambda)u - \lambda\int_{-\infty}^{+\infty} u(\tau, x+y)J(y)dy$$

Now, we will introduce an uniform grid on the domain $[0,T]\times\Omega^*$, where for the time domain we divide it in M time steps, and we define $\tau_m = m\Delta\tau$, where $m = 0,\dots,M$. While for the spatial domain we will consider the truncated domain of Ω^* we split it in N interval of size $\Delta x = 2x^*/N$ considering the two extremes point of the domain; then the solution vector is defined as $\mathbf{u}(\tau) = (u_1(\tau),\dots,u_{N-1}(\tau))^T$. The space derivatives are approximated by finite differences defined in the section (2.1.4); using the central finite differences, we can write derivatives considering them on the time point τ, as :
First order derivative:

$$\frac{\partial u}{\partial x}(\tau, x_i) \approx \frac{u_{i+1}(\tau) - u_{i-1}(\tau)}{2\Delta x}$$

Second order derivative:

$$\frac{\partial^2 u}{\partial x^2}(\tau, x_i) \approx \frac{u_{i+1}(\tau) - 2u_i + u_{i-1}(\tau)}{\Delta x^2}$$

Then we can consider the first operator $\mathcal{T}$ on the spatial point j for this model being defined as follows: Linear differential operator explicit considered at time point $n+1$

$$\mathcal{T}_j u_j^{n+1} = -\frac{1}{2}\sigma^2\Big(\frac{u_{i+1}^{n+1}(\tau) - 2u_i^{n+1} + u_{i-1}^{n+1}(\tau)}{\Delta^2 x}\Big) - \Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Big(\frac{u_{i+1}^{n+1}(\tau) - u_{i-1}^{n+1}(\tau)}{2\Delta x}\Big)$$

Now we are going to define the matrix form for the linear operator defined above. Therefore, summing up all elements of $i = 1, \dots, N-1$ we get the matrix formulation, which will bring to a tridiagonal matrix where we can denote it as D. We will denote the matrix as D whose off-diagonal elements are defined as:

$$D_{i,i-1} = \frac{-\sigma^2 + \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2} \qquad D_{i,i+1} = \frac{-\sigma^2 - \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2}$$

While for the principal diagonal:

$$D_{i,i} = -D_{i,i-1} - D_{i,i+1}$$

By the fact that with $i = 1$ and $i = N-1$ there are also the boundary elements respectively u_0 and u_N, in these points we have the known value added to the matrix; thus the matrix form of the linear differential operator will become:

$$\mathcal{T}\mathbf{u}^{n+1} = D\mathbf{u}^{n+1} + \mathbf{k} \tag{3.29}$$

where as stated above we have:

$$\begin{aligned}\mathbf{k}_1 &= \Big(-\frac{\sigma^2}{2\Delta x^2} + \frac{1}{2\Delta x}\Big(r - \frac{1}{\sigma^2} - \lambda\kappa\Big)\Delta x\Big)u_0 \\ \mathbf{k}_N &= -\Big(\frac{\sigma^2}{2\Delta x^2} + \frac{1}{2\Delta x}\Big(r - \frac{1}{\sigma^2} - \lambda\kappa\Big)\Delta x\Big)u_N\end{aligned}$$

As the previous method defined in section (2.1.4), for $i = 2, 3, \dots, N-2$ it will assume values $\mathbf{k}_i = 0$. Moreover, we can conclude that the finite difference discretization for the linear differential operator T is an approximation of second order with accuracy of $\mathcal{O}(\Delta x^2)$. Let $r \geq 0$, for a sufficiently small spatial interval Δx the off-diagonal elements of T are non-positive so the matrix will be a M-matrix, but if the spatial interval fulfills $\Delta x > \sigma^2/|r - \frac{1}{2}\sigma^2 - \lambda\kappa|$, the matrix may have positive off-diagonal elements so it can be a M-matrix. Now, we are going to add that artificial volatility such that the volatility of the model becomes:

$$\hat{\sigma}^2 = \max\Big\{\sigma^2, \Big(r - \frac{1}{2}\sigma^2 - -\lambda\kappa\Big)\Delta x, -\Big(r - \frac{1}{2}\sigma^2 - \lambda\kappa\Big)\Delta x\Big\}$$

Hence, the off-diagonal elements of T will be written with this form:

$$\hat{D}_{i,i-1} = \frac{-\hat{\sigma^2} + \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2} \quad \hat{D}_{i,i+1} = \frac{-\hat{\sigma^2} - \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2} \tag{3.30}$$

and then the diagonal element assumes this formulation:

$$\hat{D}_{i,i} = -\hat{D}_{i,i-1} - \hat{D}_{i,i+1} \tag{3.31}$$

Adding this artificial volatility to the model we define the matrix D with specific features as we can understand from the theorem defined in (1).
Now we are moving on to the integral operator, we can extend the formulation on

the equation (3.16) on every spatial point, hence using the matrix representation we have the following form for the integral operator:

$$\mathcal{J}\mathbf{u}(\tau) = J\mathbf{u}(\tau) + \mathbf{f}(\tau) + (r+\lambda)\mathbf{u}(\tau) \tag{3.32}$$

where:

$$\mathbf{f}(\tau) = \begin{pmatrix} -\lambda\mathcal{I}(\tau, x_1, x^*) - \frac{\lambda\Delta x}{2}\left(J_{1,0}u_0 + J_{1,N}u_N\right) \\ -\lambda\mathcal{I}(\tau, x_2, x^*) - \frac{\lambda\Delta x}{2}\left(J_{2,0}u_0 + J_{2,N}u_N\right) \\ \vdots \\ -\lambda\mathcal{I}(\tau, x_{N-2}, x^*) - \frac{\lambda\Delta x}{2}\left(J_{N-2,0}u_0 + J_{N-2,N}u_N\right) \\ -\lambda\mathcal{I}(\tau, x_{N-1}, x^*) - \frac{\lambda\Delta x}{2}\left(J_{N-1,0}u_0 + J_{N-1,N}u_N\right) \end{pmatrix}$$

and the J is a matrix defined as:

$$J = -h\lambda \begin{pmatrix} J_{1,1} & J_{1,2} & \cdots & J_{1,N-2} & J_{1,N-1} \\ J_{2,1} & J_{2,2} & \cdots & J_{2,N-2} & J_{2,N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ J_{N-2,1} & J_{N-2,2} & \cdots & J_{N-2,N-2} & J_{N-2,N-1} \\ J_{N-1,1} & J_{N-1,2} & \cdots & J_{N-1,N-2} & J_{N-1,N-1} \end{pmatrix}$$

As we can see J is assembled with the probability density function of the jump, so the more the two points considered are distant the smaller the value of the probability, this means that the matrix values decay from the principal diagonal. The integral operator $\mathcal{J}$ will be:

$$\mathcal{J}\mathbf{u}^n = J\mathbf{u}^n + \mathbf{f}(\tau) + (r+\lambda)\mathbf{u}^n = (J + (r+\lambda)\mathbb{I})\mathbf{u}^n + \mathbf{f}(\tau) = \hat{J}\mathbf{u}^n + \mathbf{f}(\tau)$$

As the previous matrix T, J has special features as defined in the theorem (7). Using (3.29) and (3.32) we obtain the spatial discretization of the equation we have been looking for with form:

$$\frac{\partial \mathbf{u}}{\partial \tau} + \mathcal{T}\mathbf{u}^{n+1} + \mathcal{J}\mathbf{u}^n = 0$$

Now we have to approximate the previous model in time using the implicit Euler's method, which becomes:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta\tau} + \mathcal{T}\mathbf{u}^{n+1} + \mathcal{J}\mathbf{u}^n = 0$$

Using some easy math we isolate variable in time, picking up the two vector $\mathbf{k}$ and $\mathbf{f}$ as $\mathbf{b} = -\mathbf{k} - \mathbf{f}$ we get:

$$(\mathbb{I} + \Delta\tau D)\mathbf{u}^{n+1} = (\mathbb{I} - \Delta\tau\hat{J})\mathbf{u}^n + \Delta\tau\mathbf{b}$$

which can be written as:

$$A\mathbf{u}^{n+1} = S\mathbf{u}^n + \Delta\tau\mathbf{b} \tag{3.33}$$

In this matrix form, the matrix A is a tridiagonal matrix and also a M-matrix and also strictly dominant matrix as a consequence of Theorem 6. As already proven in (3.27) the matrix J is a Toeplitz matrix, consequently the matrix S is also a Toeplitz matrix, and by that the multiplication between $S\mathbf{u}^n$ can be accelerated by FFT algorithm.

3.4.1 Consistency

In this subsection, we are going to study the consistency property of the IMEX-Euler scheme presented above.

Proposition 4 (Consistency)**.** *The finite difference scheme (3.32) is locally consistent with equation (3.15): $\forall \quad u \in C_0^\infty([0,T]\times\mathbb{R})$ and $\forall(\tau_n, x_i) \in [0,T]\times\mathbb{R}$ we have:*

$$\Big|\frac{u_j^{n+1}-u_j^n}{\Delta\tau} + (\mathcal{T}u)_j^{n+1} + (\mathcal{J}u)_j^n - \Big(\frac{\partial u}{\partial\tau} + \mathcal{L}u\Big)(\tau_n, x_j)\Big| = r_j^n(\Delta\tau, \Delta x^2) \to 0,$$

when $(\Delta\tau, \Delta x) \to 0$. Moreover:

$$\exists \quad c > 0, \qquad |r_j^n(\Delta\tau, \Delta x)| \leqslant c(\Delta\tau, \Delta x^2)$$

Proof. Considering the first term and the first derivative of solution in the time, in the absolute value, we are going to use a second order Taylor series to obtain:

$$\begin{aligned}\Big|\frac{u_j^{n+1}-u_j^n}{\Delta\tau} - \frac{\partial u}{\partial\tau}(\tau_j, x_j)\Big| &= \Big|\frac{u_j^{n+1}-u_j^n}{\Delta\tau} - \frac{u_j^{n+1}-u_j^n}{\Delta\tau} - \frac{\Delta\tau}{2}\frac{\partial^2 u}{\partial\tau^2}(\tau_n, x_j)\Big| \\ &= \Big|\frac{\Delta\tau}{2}\frac{\partial^2 u}{\partial\tau^2}(\tau_n, x_j)\Big| \\ &\leqslant \frac{\Delta\tau}{2}\Big|\Big|\frac{\partial^2 u}{\partial\tau^2}\Big|\Big|_\infty\end{aligned}$$

Now considering the term in $\mathcal{J}u$, by the fact we used the trapezoidal rule we can easily write the trapezoid inequality:

$$\begin{aligned}&\Big|\int_{x_j}^{x_{j+1}} u(\tau,z)J(z-x_i)dz - \frac{x_{j+1}-x_j}{2}\Big(u(\tau,x_j)J(x_j-x_i) + u(\tau,x_{j+1})J(x_{j+1}-x_i)\Big)\Big| \\ &\qquad\qquad\qquad\qquad \leqslant \frac{\Delta x^3}{12}||(u(\tau,z)J(z-x_i))''||_\infty\end{aligned}$$

Then for the $(Du)_j^{n+1}$ we have:

$$\begin{aligned}|(Du)_j^{n+1} - (\mathcal{D}u)(\tau_n, x_j)| &= |(Du)_j^{n+1} - (\mathcal{D}u)(\tau_{n+1}, x_j) + \Delta\tau(\mathcal{D}u)(\hat\tau, x_j)| \\ &= \Big|\frac{\sigma^2}{2}\Big(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} - \frac{\Delta x^2}{12}\frac{\partial^4 u}{\partial x^4}(\tau_{n+1}, x_j)\Big) \\ &\quad + (r - \frac{1}{2}\sigma^2 - \lambda\kappa)\Big(\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} - \frac{\Delta x^2}{6}\frac{\partial^2 u}{\partial x^3}(\tau_{n+1}, x_j)\Big) \\ &\quad - (\mathcal{D}u)(\tau_{n+1}, x_j) + \Delta\tau(\mathcal{D}u)(\hat\tau, x_j)\Big| \\ &\leqslant C_3\Delta\tau + C_4\Delta x^2\end{aligned}$$

Therefore these inequality lead to:

$$r_j^n(\Delta, \Delta x) \leqslant C_1\Delta\tau + C_2\Delta x^2 \to 0$$

□

3.4.2 Monotonicity and Stability

In this section we are going to show other two important properties for the implicit-explicit scheme the monotonicity and the stability of it, following the idea in the work of Rama Cont and Ekaterina Voltchkova in [6, Chapter 4.2].

Proposition 5. *If $\Delta\tau \leq 1/\hat{\lambda}$, the scheme "Implicit-Explicit" used is stable and verifies the discrete comparison principle: if u^0 and v^0 are two bounded initial condition then:*

$$u^0 \geq v^0 \Longrightarrow \qquad [\forall n \geq 1, u^n \geq v^n]$$

Proof. We start with rewriting the equation (3.32) in the following form:

$$\begin{aligned} c\Delta\tau u_{i-1}^{n+1} + (1+a\Delta\tau)u_i^{n+1} + b\Delta\tau u_{i+1}^{n+1} =& (1-\Delta\tau(r+\lambda))u_i^n \\ &- \lambda\Delta\tau\frac{\Delta x}{2}\Big[J_{i,0}u_0^n + J_{i,N}u_N^n \\ &+ 2\sum_{j=1}^{N-1} f_{i,j}u_j^n\Big] \end{aligned} \tag{3.34}$$

where we denote:

$$\begin{aligned} b &= \frac{-\hat{\sigma^2} - \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2} \\ c &= \frac{-\hat{\sigma^2} + \left(r - \frac{1}{2}\sigma^2 - \lambda\kappa\right)\Delta x}{2\Delta x^2} \\ a &= -b - c \end{aligned}$$

Note that $(1 - \Delta\tau(r+\lambda)) \geq 0$ by hypothesis.
Stability We have a tri-diagonal linear system on $\mathbf{u}^{n+1} = (u_1^{n+1}, \ldots, u_{N-1}^{n+1})^T$. It has a unique solution since the main diagonal is dominant in fact: $1+\Delta\tau a \geq c\Delta\tau + b\Delta\tau$. We are going to show that, if Φ is bounded: $||\Phi||_\infty < \infty$, then $\forall n$,

$$||\mathbf{u}^n||_\infty \leq ||\Phi||_\infty.$$

Let us proceed by induction, thus by the definition of $\mathbf{u}^0$ we have $||\mathbf{u}^0||_\infty \leq |\Phi||_\infty$. Let us suppose $||\mathbf{u}^{n+1}||_\infty > ||\Phi||_\infty$, it means that $\exists i_0 \in \{0, \ldots, N\}$, such that $|u_{i_0}^{n+1}| = ||\mathbf{u}^{n+1}||_\infty$, and $\forall i \in \mathbb{Z}$, $|u_i^{n+1}| \leqslant |u_{i_0}^{n+1}|$ therefore we have:

$$\begin{aligned} ||\mathbf{u}^{n+1}||_\infty = |u_{i_0}^{n+1}| &= -c\Delta\tau|u_{i_0}^{n+1}| + (1+a\Delta\tau)|u_{i_0}^{n+1}| - b\Delta\tau|u_{i_0}^{n+1}| \\ &\leq -c\Delta\tau|u_{i_0-1}| + (1+a\Delta\tau)|u_{i_0}^{n+1}| - b\Delta\tau_{i_0+1}^{n+1}| \\ &\leqslant |-c\Delta\tau u_{i_0-1}^{n+1} + (1+a\Delta\tau)u_{i_0}^{n+1} - b\Delta\tau u_{i_0+1}^{n+1}| \end{aligned}$$

With (3.34) this gets:

$$\begin{aligned}
||\mathbf{u}^{n+1}||_\infty &\leqslant (1-\Delta\tau(r+\lambda))||\mathbf{u}^n||_\infty - \lambda\Delta\tau\frac{\Delta x}{2}\Big[J_{i,0}|u_0^n| + J_{i,N}|u_N^n| + 2J||\mathbf{u}^n||_\infty\Big] \\
&\leq ||\mathbf{u}||_\infty \\
&\leq ||\Phi||_\infty
\end{aligned}$$

Which it is a contraction over our assumption. Therefore $||u^{n+1}||_\infty \leq ||\Phi||_\infty$. **Monotonicity**. Let $\mathbf{u}^n$ and $\mathbf{v}^n$ be two solution of the scheme which has initial condition $h(x)$ and $g(x)$ in order to have $h(x) \geq g(x), \forall x \in \mathbb{R}$. Let us define $\mathbf{w}^n = \mathbf{u}^n - \mathbf{v}^n$, we have to prove $\mathbf{w}^n \geq 0, \forall n \geq 0$. By definition we have $\mathbf{w}_i^0 = h(x_i) - g(x_i) \geq 0, \forall i \in \mathbb{Z}$. Let $\mathbf{w}^n \geq 0$ and suppose that $\inf_{i\in\mathbb{Z}} w_i^{n+1} < 0$. By the fact that $\forall i \in \mathbb{Z}\backslash\{0,\ldots,N\}, w_i^{n+1} = h(x_i) - g(x_i) \geq 0$, we have that $\exists i_0 \in \{0,\ldots,N\}$, s.t $w_{i0}^{n+1} = \inf_{i\in\mathbb{Z}} w_i^{n+1}$. Then using the previous definition for the scheme in (3.34) we get:

$$\begin{aligned}
\inf_{i\in\mathbb{Z}} w_i^{n+1} = w_{i_0}^{n+1} &= -c\Delta\tau w_{i_0}^{n+1} + (1+a\Delta\tau)w_{i_0}^{n+1} - b\Delta\tau w_{i_0}^{n+1} \\
&\geq -c\Delta\tau w_{i_0-1}^{n+1} + (1+a\Delta\tau)w_{i_0}^{n+1} - b\Delta\tau w_{i_0+1}^{n+1} \\
&= (1-\Delta\tau(r+\lambda))||\mathbf{w}_0^n||_\infty - \lambda\Delta\tau\frac{\Delta x}{2}\Big[J_{i,0}w_0^n + J_{i,N}w_N^n + 2J||\mathbf{w}^n||\Big] \\
&\geq 0
\end{aligned}$$

it is a contradiction of the assumption, so $\inf_{i\in\mathbb{Z}} w_i^{n+1} \geq 0$ and consequently $\mathbf{w}^{n+1} \geq 0$. □

Chapter 4

Methods for solving linear systems

In this chapter, we will present the solving methods that will be used to solve linear systems. First of all, we will develop the algorithm and understand the main properties of the speed of convergence of the main method we will use: GMRES. Explaining this method we briefly present the projection method and Krylov subspace method, since they are fundamental concepts in theGMRES method. After that, we will introduce the Multigrid method being used. Lastly, we will introduce the tridiagonal linear system algorithm known also as Thomas' algorithm.

4.1 Projection methods

A projection method is to find an approximation to the solution of the linear system. The problem of solving the linear system $A\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. Now we have to find an approximation of the solution $x \subseteq \mathbb{R}^n$ being called K. An easy way to find it let us suppose the dimension of K is $m < n$, m conditions are needed to determinate the approximation inside the subspace. These constraints impose the residual given as $\mathbf{b} - A\mathbf{x}$ is orthogonal to another subspace L with dimension m which is called Petrov-Galerkin condition. Starting from the initial approximation x_0 the problem is:

$$\hat{\mathbf{x}} \in \mathbf{x}_0 + K \mid \mathbf{b} - A\hat{\mathbf{x}} \perp L$$

Then, supposing the bases for the sub-spaces K and L being called:

- $B \in \mathbb{R}^{n\times m}$ containing the m vectors of the K basis
- $F \in \mathbb{R}^{n\times m}$ containing the m vectors of the L basis

The approximation can be expressed by:

$$\hat{\mathbf{x}} = \mathbf{x}_0 + B\mathbf{y}$$

with $y \in \mathbb{R}^m$. Imposing the orthogonality condition, it leads to the residual being able to be written as:

$$\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}} = \mathbf{b} - A\mathbf{x}_0 - AB\mathbf{y}$$

while the for the orthogonality to L we have:

$$F^T\mathbf{r} = F^T\mathbf{b} - F^TA\mathbf{x}_0 - F^TAB\mathbf{y} = F^T\mathbf{r}_0 - F^TAB\mathbf{y} = 0$$

Hence, we have the expression of $\mathbf{y}$ and we can write the approximation as:

$$\hat{\mathbf{x}} = \mathbf{x}_0 + B(F^TAB)^{-1}F^T\mathbf{r}_0$$

Then for the given the equation above, we have the projection method that can be presented by the algorithm in [21]:

Algorithm 1 Projection Method

```
1: repeat
2:     Select subspace L and K;
3:     Choose bases $B = [b_1, b_2, \dots, b_m]$ and $F = [f_1, f_2, \dots, f_m]$ for $L$ and $K$;
4:     $\mathbf{r} := \mathbf{b} - A\mathbf{x}$;
5:     $\mathbf{y} := (F^TAB)^{-1}F^T\mathbf{r}$;
6:     $\mathbf{x} := \mathbf{x} + B\mathbf{y}$;
7: until Convergence
```

The condition for this algorithm is that the matrix F^TAB has to be nonsingular. The projection method minimizes the 2-norm of the residual, so we have the result formalized in [21] as follows:

Proposition 6. *If A is an arbitrary square matrix and $L = AK$, then $\hat{\boldsymbol{x}}$ is the result of a projection method onto K, orthogonally to L, if and only if it minimized the 2-norm of the residual over $\boldsymbol{x}_0 + K$, i.e.*

$$||\boldsymbol{b} - A\hat{\boldsymbol{x}}||_2 = \min_{x \in x_0 + K} ||\boldsymbol{b} - A\boldsymbol{x}||_2 \tag{4.1}$$

4.1.1 Krylov subspaces

Starting from the definition for these subspaces, we have:

Definition 9. *Given a nonsingular $A \in \mathbb{R}^{n \times n}$ and $\boldsymbol{y} \in \mathbb{R}^n$ and an integer $m \leq n$, a Krylov subspace is:*

$$\mathcal{K}_m = \mathcal{K}_m(A, \boldsymbol{y}) = span(\boldsymbol{y}, A\boldsymbol{y}, A^2\boldsymbol{y}, \dots, A^{m-1}\boldsymbol{y})$$

This means, $\mathcal{K}_m$ is the subspace of all the vectors of $\mathbf{z}\mathbb{R}$ which can be written in the form:

$$\mathbf{z} = p(A)\mathbf{y}$$

where $p(A)$ belongs to the set of the polynomial of degree at most $m-1$. Of course, $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \dots$, the dimension increases at most by one dimension in each step. It is clear to choose the approximate solution $\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n(A, \mathbf{r}_0$. We have to define the minimal polynomial of $\mathbf{y}$ as the nonzero monic polynomial p such that $p(A)\mathbf{y} = 0$ with the lowest degree and where the degree of the $\mathbf{y}$ as the degree of p. Therefore, let us define the following propositions:

Proposition 7. *Let ν be a positive integer defined as $\nu := \nu(\boldsymbol{y}, A)$ which is called grade of $\boldsymbol{y}$ w.r.t matrix A, such that:*

$$dim\,\mathcal{K}_m(A, \boldsymbol{y}) = \min\{m, \nu\}$$

Proof. Given the subspace $\mathcal{K}_m$, the constraint to be m-dimension subspace is that vectors $\mathbf{y}, A\mathbf{y}, \dots, A^{m-1}\mathbf{y}$ have to be linear independent. This means that can not exist any nonzero polynomial $\deg(q) \leq m-1$ such that $q(A)\mathbf{y} = 0$ and so we have that degree of $(\mathbf{y})$ is at least m. Calling $\nu = \deg(\mathbf{y})$, we have $\nu \geq m$, then we consider the definition of grade of $\mathbf{y}$, so there exists coefficients c_i such that:

$$\sum_{i=0}^{\nu} c_i A^i \mathbf{y} = 0$$

This means, using the linear independence:

$$\sum_{i=0}^{\nu-1} c_i A^i \mathbf{y} + c_\nu A^\nu \mathbf{y} = 0 \implies A^\nu \mathbf{y} = -\sum_{i=0}^{\nu-1} \frac{c_i}{c_\nu} A^i \mathbf{y}$$

As one can see $A^\nu \mathbf{y}$ is linear combination of vectors in the subspace $\mathcal{K}_\nu$. Then, let us consider a vector $\mathbf{z} \in \mathcal{K}_{\nu+1}$, it can be expressed by:

$$\mathbf{z} = \sum_{i=0}^{\nu} \beta_i A^i \mathbf{y} = \beta_\nu A^\nu \mathbf{y} + \sum_{i=0}^{\nu-1} \beta_i A^i \mathbf{y}$$

The two terms belongs to the subspace $\mathcal{K}_\nu$. This implies that $\mathbf{z} \in \mathcal{K}_\nu$, and that also implies that the two subspaces $\mathcal{K}_\nu$ and $\mathcal{K}_{\nu+1}$ are the same subspace. Eventually, it $m \leq \nu$, the dimension is m, while if $m \geq \nu$ the dimension is ν which can be defined as $\min\{m, \nu\}$. □

Corollary 2. *$\mathcal{K}_\nu(A, \boldsymbol{y})$ is the smallest A-invariant subspace that contains $\boldsymbol{y}$.*

One can see that starting with the vector $\mathbf{y}$ which is an eigenvector of A, the on the subspace $\mathcal{K}_m$ any other vector will be written as $\mathbf{z}_i = A^i\mathbf{y}$ which leads to any $\mathbf{z}_i$ being parallel to $\mathbf{y}$. Therefore, the dimension $m = 1$ of the subspace. Moreover, if $\mathbf{y}$ is a linear combination of some eigenvectors this leads to the maximum dimension of any Krylov subspace using $\mathbf{y}$ will be the number of eigenvectors making $\mathbf{y}$ up. We now introduce the Krylov subspace methods, which are defined as a projection method where the subspaces K and L are Krylov subspaces. Now, we need to build a good basis for the space $\mathcal{K}_m$. We can note that for a large value of m, most of the vectors in the Krylov basis are parallel that leads to an ill-conditioned basis. A great choice from the standpoint of numerical stability could be the Arnoldi process, which produces an orthonormal basis, which is simply the modified Gram-Schmidt iteration.

Let us introduce the Arnoldi's process algorithm:

Algorithm 2 Arnoldi process

```
1: Arbitrary $y_1$, s.t $||y|| = 1$;
2: for $i = 1, 2, \dots, m$ do;
3:     for $j = 1, 2, \dots, i$ do
4:         $h_{j,i} = (A_i)^T y_j$;
5:     end for
6:     $w_i = Ay_i - \sum_{a=1}^{i} h_{ai} y_a$;
7:     $h_{i+1}, i = ||w_i||_2$;
8:     if $h_{i+1,i} = 0$ then
9:         Break;
10:    else
11:        $y_{i+1} = w_i / h_{i+1,i}$;
12:    end if
13: end for
```

So for this algorithm, there is an important proposition that states:

Proposition 8. *Algorithm 2 stops at step i, so that $h_{i+1,i} = 0$, if and only if the deg($\boldsymbol{y}_1$) = i*

From the proposition (8) the Arnoldi succeeds to find a full orthonormal basis for $\mathcal{K}_m$; in fact, it either computes all the m vectors or it stops reaching the dimension of the subspace. Therefore, in the case m increases we have two possible ways. In the first one, the dimension of the space grows and the Arnoldi process is able to compute the vector for the basis. This means that the next approximation found is for sure more accurate than the previous one and since the residual is minimized on a subspace with a larger dimension it will be smaller. For the second case, we have that the maximum dimension is reached and the algorithm stops. In this case, the solution of any projection method is exact and this leads to an iterative procedure in which for each iteration m increases in order to find a good approximation. Practical speaking the method has to be stopped regarding to a threshold on the residual since the exact solution will be never reached. For this algorithm there are some notable notes. The mostly memory space is for the $(m + 1)n$ for storing Arnoldi vectors $\mathbf{y}_i$. Then the matrix A is referenced through the matrix-vector multiplication, so it is ideal for large scale matrices; furthermore, these kinds of operations are between $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$ while it takes $\mathcal{O}(mn)$ operations plus. Moreover, this method has a straight property which in fact is that the computational cost of a single iteration increase as the iteration count proceeds, by the fact that each time there is an extra vector with respect to which the new vectors have to be orthonormalized. Despite that, there are also some special cases which reduce the vector number to which the new vector is orthogonalized, this leads to a short recurrence for the properties of the linear system matrix. A notable example of this is the Conjugate Gradient. For a generic non-symmetric matrix we can not benefit from a short recurrence, so the cost of the Arnoldi process is raw every time. This conducts two categories for iterative methods. The first one performs fully Arnoldi method, the second type is a variation for the Arnoldi method where there is a fixed maximum number of vectors against which to orthogonalize. The last one is a non-optimal computation for the solution but since the number is fixed the computational cost does not increase and an example of this kind of method is the GMRES. Lastly, for this algorithm, the outputs are two matrices:

- $V_m \in \mathbb{R}^{n\times m}$ containing the orthonormal vectors for the basis of $\mathcal{K}_m$
- $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ containing the coefficients h_n, j

We defined in addiction the matrix $H_m \in \mathbb{R}^{m\times m}$ which is defined taking $\bar{H}$ and removing its last row. We obtain a relationship for those matrices:

$$\begin{aligned} AV_m &= V_m + \bar{H}_m \\ V_m^T AV_m &= H_m \end{aligned} \tag{4.2}$$

4.1.2 GMRES

One of the most known projection methods is the Generalized Minimum Residual Method (GMRES) developed by Yousef Saad and Martin H. Schultz in 1986, it found more details on [20]. This iterative method has $K = \mathcal{K}_m$ and $L = A\mathcal{K}_m$. It is a method of solving linear systems working with general systems of linear equations. The method involved starts with an initial guess $\mathbf{x}_0$, where the first vector for the Krylov space is taken to be the normalized initial residual $\mathbf{y}_1 = \mathbf{r}_0/||\mathbf{r}_0||$. As we already saw in the Krylov subspace method, an approximation for the solution $\mathbf{x}$ at the iteration m has to belong to the space $\mathbf{x}_0 + \mathcal{K}_m$. After that, knowing a basis for the space we stored the vectors of the basis in the columns of the matrix V_m, so we can define the solution as:

$$\mathbf{x} = \mathbf{x}_0 + V_m\mathbf{v}$$

Now we have to find the vector $\mathbf{y} \in \mathbb{R}^m$ which minimizes the residual among all the vectors in that space, where the residual can be evaluated as follows:

$$\begin{aligned} \mathbf{r} &= \mathbf{b} - A\mathbf{x} \\ &= \mathbf{b} - A\mathbf{x}_0 - AV_m\mathbf{v} \\ &= \mathbf{r}_0 - V_{m+1}\bar{H}_m\mathbf{v} \end{aligned}$$

This derives from the relationships on (4.2). Now, we define the norm of the initial residual β and calling $\mathbf{e}_1$ the first vector of the canonical basis we have:

$$\mathbf{y}_1 = V_{m+1}\mathbf{e}_1$$

Using the equation above we have:

$$\begin{aligned} \mathbf{b} - A\mathbf{x} &= \beta\mathbf{y}_1 - V_{m+1}\bar{H}_m\mathbf{v} \\ &= V_{m+1}(\beta e_1 - \bar{H}_m\mathbf{v}) \end{aligned}$$

After that, in order to optimize the problem, we have to minimize the norm of the residual and by the fact that the columns of the matrix with the vector stored V_{m+1} are orthonormal by construction we obtain:

$$||\mathbf{b} - A\mathbf{x}|| = ||\beta\mathbf{e}_1 - \bar{H}_m\mathbf{v}||$$

Hence, the approximation $\mathbf{x}_m$ can be found with the equation below:

$$\mathbf{x}_m = \mathbf{x}_0 + V_m \mathbf{v}_m$$

where $\mathbf{v}_m$ can be defined as:

$$\mathbf{v}_m = \min ||\beta \mathbf{e}_1 + \bar{H}_m \mathbf{v}||$$

In order to evaluate the vector $\mathbf{v}$, let us consider the QR decomposition of the matrix $\bar{H}_m$, which means that any square or rectangular matrix can be decomposed into a multiplication of orthogonal matrix Q and another upper triangular matrix R. In the problem considered we have $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ whose last row is a zero-row. Hence, we have:

$$\begin{aligned}\beta \mathbf{e}_1 - \bar{H}_m \mathbf{v} &= \beta \mathbf{e}_1 - QR\mathbf{v} \\ &= Q(^T \mathbf{e}_1 + R\mathbf{v})\end{aligned}$$

Let us define $g_k = \beta Q^T \mathbf{e}_1$ and R_k as the $m \times m$ sub-matrix of R which can be obtained removing the last row full of zeros. Then defining g_1 the first m elements of g and g_{m+1} the last one we have:

$$\beta Q^T \mathbf{e}_1 - R\mathbf{y} = \begin{bmatrix} g_1 \\ g_{m+1} \end{bmatrix} - \begin{bmatrix} R_1 \\ \mathbf{0} \end{bmatrix} \mathbf{v}$$

Using these kinds of annotation, we have to minimize $||g - R\mathbf{v}$, moreover the matrix Q is orthogonal. In this case the minimum is reached when $R_1 \mathbf{v} = g_1$, in such a way that the argument of the norm has m components that are zero and only the last one different from zero, this leads to the fact that $\mathbf{v} = R_1^{-1} g_1$. For all these steps we have that: QR decomposition has a computational cost asymptotic to $\mathcal{O}(m^3)$ but despite that, the value of m is usually way too smaller than the dimension of the linear system matrix; the solution of the linear system y actually is really cheap since the matrix R_1 is triangular. The most expensive operations are the ones to determine the basis of the Krylov space. In fact, to do that we will use Arnoldi process which requires matrix-vector products involving a large matrix size, even though the procedure need not be repeated each time since for each iteration we can just upgrade the matrices V and H.
The GMRES method can be implemented in an alternative way by imposing the orthogonality condition $L_m = A\mathcal{K}_m \perp r_m$, which is defined as:

$$V_m^T A^T (\mathbf{r}_0 - AV_m \mathbf{v}) = 0$$

Hence, the solution can be written as:

$$\mathbf{x} = \mathbf{x}_0 + V_m (V_m^T A^T A V_m)^{-1} V_m^T A^T \mathbf{r}_0$$

It leads to the vector $\mathbf{v}$ to be written as:

$$\begin{aligned}\mathbf{v} &= (V_m^T A^T A V_m)^{-1} V_m^T A^T \mathbf{r}_0 \\ &= (\bar{H}_{m+1,m}^T V_{m+1}^T V_{m+1} \bar{H}_{m+1,m})^{-1} \bar{H}_{m+1,m}^T V_{m+1}^T \mathbf{r}_0 \\ &= (\bar{H}_{m+1,m}^T V_{m+1}^T V_{m+1} \bar{H}_{m+1,m})^{-1} \bar{H}_{m+1,m} V_{m+1}^T \mathbf{r}_0 \\ &= (\bar{H}_{m+1,m}^T \bar{H}_{m+1,m})^{-1} \bar{H}_{m+1,m}^T V_{m+1} \mathbf{r}_0\end{aligned}$$

the columns of the matrix V_{m+1} are orthonormal, so the matrix $V_{m+1}^T V_{m+1}$ is equal to the identity matrix with dimension $m+1$. Then, the residual $\mathbf{r}_0$ can be expressed by $\mathbf{r}_0 = \beta y_1$

$$\mathbf{v} = (\bar{H}_{m+1,m}^T \bar{H}_{m+1,m})^{-1} \bar{H}_{m+1,m} \beta \mathbf{y}_1$$

Then the product $V_{m+1}^T \mathbf{y}_1 = \beta \mathbf{e}_1$

$$\mathbf{v} = (\bar{H}_{m+1,m}^T \bar{H}_{m+1,m})^{-1} H_{m+1,m} \beta \mathbf{e}_1$$

Therefore, we have a system of normal equation for which finds the least square solution for $\mathbf{v}$ of the function:

$$\bar{H}_{m+1,m} \mathbf{v} = \beta \mathbf{e}_1$$

In fact if we multiply the previous system by the matrix $\bar{H}_{m+1,m}^T$ we can get:

$$\bar{H}_{m+1,m}^T \bar{H}_{m+1,m} \mathbf{v} = \bar{H}_{m+1,m}^T \beta \mathbf{e}_1$$

whose solution is the previous system. Using that we have proven using the projection methods and we obtain the same result. Since using a non-symmetric matrix at each iteration the method require to orthonormalize the new vector with the previous ones, to improve that we can use the restarted GMRES for which we can fix the maximum number of vector that can be memorized. Applying this method of course we lost the accuracy but we can improve the computational cost it needs to compute. There is also another version for GMRES which is the Truncated GMRES which may save computations costs but not storage costs. Regarding the GMRES method's breakdown, it can happen if and only if the Arnoldi process stops. In fact it stops when $w_j = 0$ so, when $h_{j+1,j} = 0$ at a given step j. This might leads to a not exact solution, instead, the residual vector is zero and this leads to the exact solution. This is stated by the following proposition:

Proposition 9. *Given the nonsingular matrix A, the GMRES algorithm breaks down at step j, i.e. $h_{j+1,j} = 0$ if and only if the approximate solution x_j is exact.*

Here below, we present the algorithm for the GMRES:

Algorithm 3 GMRES

```
    Input: $A$,$b$,$x_0$, $k_{max}$ tol;
 1: $r_0 = b - Ax_0$, $k = 0$, $\rho_0 = ||r_0||_2$, $\beta = \rho$, $v_1 = \frac{r_0}{\beta}$;
 2: while $\rho_k > tol||b||_2$ AND $k < k_{max}$ do
 3:     $v_{k+1} = Av_k$;
 4:     for $j = 1, 2, 3 \ldots, k$ do
 5:         $h_{j,k} = v_{k+1}^T v_j$;
 6:         $v_{k+1} = v_{k+1} - h_{j,k} v_j$;
 7:     end for
 8:     $h_{k+1,k} = ||v_{k+1}||_2$;
 9:     $v_{k+1} = v_{k+1}/h_{k+1,k}$;
10:     $QR(H_{k+1,k}) \implies \bar{H}_{k+1} = QR$;
11:     $\rho_k = |\beta q_{1,k+1}|$;
12:     $k = k + 1$;
13: end while
14: $y_k = \text{argmin}||\beta e_1 - QR(\bar{H}_{k+1,k})y||$;
15: $x_k = x_0 + V_k y_k$;
```

4.1.3 GMRES: Theoretical features

Following Y. Saad and M.H. Schultz [21], we are going to show some theoretical fundamental features of the GMRES. So now we will study the convergence of the GMRES method:

Theorem 8. *Let $A \in R^{n\times n}$ be the matrix of nonsingular system; the GMRES method fins the solution in at most n steps.*

Proof. Considering the charatetistic polynomial of A defining as$p(x) = det(A - xI)$; it has degree n, $p(0) \neq 0$ and $p(A) = 0$ by the Cayley-Hamilton theorem. Let the polynomial $\bar{p}(x)$ be defined as $\bar{p}(x) = \frac{p(x)}{p(0)}$, therefore $\bar{p}(0) = 1$. Since the generic approximation of $\mathbf{x}$ belongs to the space defined by $\mathbf{x}_0 + \mathcal{K}_m$ it can be evaluated following the definition of Krylov subspace as:

$$\mathbf{x} = \mathbf{x}_0 + \sum_{i=0}^{m-1} \alpha_i A^i \mathbf{r}_0$$

Having above definition, we can also express the generic residual as:

$$\begin{aligned}\mathbf{r} &= \mathbf{b} - A\mathbf{x} \\ &= \mathbf{b} - A\mathbf{x}_0 - \sum_{i=0}^{m-1} \alpha_i A^{i+1} \mathbf{r}_0\end{aligned}$$

The GMRES method by its properties of minimization generates a residual at the generic step m and by the previous polynomial of degree m with $p(0) = 1$, we obtain:

$$||\mathbf{r}_m|| = \min_{\substack{p_m \in \Pi_m \\ p_m(0)=1}} ||p_m(A)\mathbf{r}_0||$$

From that, we have:

$$\frac{||\mathbf{r}_n||}{||\mathbf{r}_0||} \leq ||q(A)|| \quad \forall\, q \in P_n \mid q(0) = 1$$

From that, we choose $q = \hat{p}$ in order to get $||r_n|| \leq 0 \implies \mathbf{r}_n = \mathbf{0}$ and so we get the exact approximation. □

By the construction of the generic non-symmetric matrix in the GMRES method, there can be found a generic and simple result that describe the relationship between the residual and the condition number as in the case of CG, example in (Theorem 6.29 in [21]). This kind of result can be achieved in GMRES if the matrix A is diagonalizable. In fact we have this fundamental result about speed of convergence of the GMRES method:

Theorem 9. *Let A be a nonsingular diagonalizable matrix, i.e. $A = VDV^{-1}$. where $V = diag\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ is the diagonal matrix of eigenvalues. Then, the residual reached on the $i - th$ steps of GMRES fulfills the following:*

$$\frac{||r_i||}{||r_0||} \leq \kappa(V) \min_{\substack{p\in\Pi \\ p(0)=1}} \max_{j=1,\dots,n} |p(\lambda_j)|$$

Proof. Since, we already know that $\forall p \in \Pi_i$ s.t. $p(0) = 1$ we have:

$$\frac{||r_i||}{||r_0||} \leq ||p(A)||$$

The norm of $p(A)$ can be estimated as:

$$||p(A)|| \leq ||V||\ ||V^{-1}||\ ||p(D)||$$

One can note that the term $||V||$ is the condition number of the matrix V and since D is diagonal with values equals to the eigenvalues of the initial matrix A, we observe that:

$$||p(D)|| = \max_{j=1,\dots,n} |p(\lambda_j)|$$

Since with the definition of matrix norm for D the combination of the $n - th$ power of the eigenvalues of A which is a convex combination, thus we have that:

$$||p(D)|| \leq \max_{j=1,\dots,n} |p(\lambda_j)|$$

So we have:

$$\begin{aligned}\frac{||r_i||}{||r_0||} &\leq ||p(A)|| \\ &\leq ||V||\ ||V^{-1}|| \max_{j=1,\dots,n} |p(\lambda_j)|\end{aligned}$$

Since the solution minimizes the residual, the polynomial p which minimizes the second part of inequality can be used, this leads to:

$$\frac{||r_i||}{||r_0||} \leq \kappa(V) \min_{\substack{p\in\Pi \\ p(0)=1}} \max_{j=1,\dots,n} |p(\lambda_j)|$$

Which is the result required and the thesis is proven. □

Unfortunately, we can not simply relate the speed of convergence of GMRES to the eigenvalue distribution of the matrix A. Moreover, the matrix V might be ill-conditioned.

Preconditioned GMRES

Sometimes, we have to face really ill-conditioned systems, thus in these cases the GMRES method could be really slow to converge or in other case it can not even converge. For those kinds of problem, we might use a technique called Preconditioner. In fact, we can precondition the matrix of linear system in order to improve the performance of the GMRES method. Given the linear system in the classic form:

$$A\mathbf{x} = \mathbf{b}$$

Then, suppose it is an ill-conditioned problem, a preconditioner is a matrix M which is applied to the linear system and somehow we have to obtain an equivalent system for which the iterative method converges faster and reduce the condition number. Moreover, a significant result is given by the more the preconditioner gets close to A, the more the eigenvalues cluster around the value 1; and by the Theorem ([**?**]) faster is the convergence of the GMRES method. The intuition behind the preconditioner is to find a matrix M which is similar to A, but it is cheap to compute and invert as well. There are three options to precondition a linear system: left preconditioning, right preconditioning and split preconditioning.
Starting from the left preconditioning, it is equivalent to apply the GMRES to the system and we obtain this construction:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}$$

Therefore in this case, the left preconditioned GMRES has to minimizes the following residual:

$$||M^{-1}b - M^{-1}A\mathbf{x}_m|| = ||M^{-1}\mathbf{r}_m||$$

The left preconditioned GMRES is quite different from the GMRES algorithm defined in (3), in fact it is different in the following two steps:

- $\mathbf{r}_0 = M^{-1}(\mathbf{b} - A\mathbf{x}_0)$, so the initial residual
- in the third step which $\mathbf{v}_{k+1} = M^{-1}A\mathbf{v}_k$

The Arnoldi process building the orthogonal basis of the left preconditioned Krylov subspace has this form:

$$\text{span}\{\mathbf{r}_0, M^{-1}A\mathbf{r}_0, \dots, (M^{-1}A)^{m-1}\mathbf{r}_0\}$$

For the right preconditioning, we have to solve the following linear system:

$$AM^{-1}\mathbf{y} = b$$

where y is the new variable that need not to be computed explicitly, it is defined as:

$$\mathbf{x} = M^{-1}y$$

In this kind of problem, the residual is defined as

$$||\mathbf{r}_m|| = ||\mathbf{b} - AM^{-1}\mathbf{y}_m||$$

Then it has to be minimized among all the vectors:

$$\mathbf{y}_m = \mathbf{y}_0 + \text{span}\{\mathbf{r}_0, AM^{-1}\mathbf{r}_0, \dots, (AM^{-1})^{m-1}\mathbf{r}_0\}$$

For the algorithm of the right preconditioned GMRES is slightly different from the algorithm (3) in the following steps:

- $\mathbf{v}_{k+1} = AM^{-1}\mathbf{v}_k$
- In this case the initial residual for solution $\mathbf{y}_0$ and $\mathbf{x}_0$ are computed as $\mathbf{r}'_0 = \mathbf{b} - AM^{-1}\mathbf{y}_0$ and $\mathbf{r}_0 = \mathbf{b} - AM^{-1}\mathbf{x}_0$
- The final approximation is obtained with the equation $\mathbf{x}_m = \mathbf{x}_0 + M^{-1}V_m\mathbf{v}_m$

The split preconditioning, in many cases M is the result of a decomposition of the form:

$$M = LU$$

Therefore, we can use the GMRES on the split-preconditioned problem defined in the following structure:

$$\begin{aligned} &L^{-1}AU^{-1}y = L^{-1}\mathbf{b} \\ &x = U^{-1}y \end{aligned}$$

In this case, we need to compute the initial residual by L^{-1} when we start the algorithm and by U^{-1} on the linear combination $V_m\mathbf{v}_m$ in forming the approximate result. In this situation, the differences with the GMRES algorithm in (3) are the following:

- the initial residual defined as $\mathbf{r}_0 = L^{-1}(\mathbf{b} - A\mathbf{x}_0)$
- $v_{m+1} = L^{-1}AU^{-1}\mathbf{v}_k$ at the step 3
- The approximate that we obtain at the last step as $\mathbf{x}_m = \mathbf{x}_0 + U^{-1}V_m\mathbf{v}_m$

In this case, there are different types of residuals and this fact might lead to a negative effect on the stopping criterion for the algorithm, and furthermore, it can be stopped too soon or later. In addiction, a bad choice of M, thus ill-conditioned, can corrupt the process' performance. Between the left and right preconditioned problem, the choice could be not particularly significant, the only exception is when M is ill-conditioned. As stated in [21] in the case of A nearly symmetric a split preconditioned may be much more performing. One of the simplest and most used preconditioner techniques is to factorize the matrix. Pointing out that an exact factorization for a large sparse matrix would not produce another sparse matrix, but it could produce large dense matrix and it leads to an increase in the computational and storage cost. Another way is to use incomplete factorization which can preserve sparsity. There are: incomplete LU factorization and incomplete Cholesky factorization. Even though they are not equal to the previous matrix, the inverses of these decompositions are easy to compute, since the matrices that we obtain are triangular. The most used ILU variant in Y. Saad. is based on two parameters:

- p, the maximum number of non-zero elements allowed in rows of L and U
- τ (drop tolerance), threshold below which elements in matrix L are discarded

Preconditioner: matrix splitting

A preconditioning technique we are going to use in our linear systems will be the splitting matrix. Let us define the splitting process in a matrix. First of all, we define the linear system:

$$A\mathbf{x} = \mathbf{b}$$

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. By matrix splitting we refer to the definition of the matrix A as:

$$A = M - N$$

with M nonsingular. A large number of iterative methods to solve the linear system above can be composed through the splitting and it can be rearranged in order to approximate the solution $\mathbf{x}^{n+1}$, as follows:

$$M\mathbf{x}^{n+1} = N\mathbf{x}^n + \mathbf{b} \tag{4.3}$$

where n are the iteration steps to compute the approximation. The starting vector is $x^{(0)}$; as such the iterative method is convergent to the unique solution:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

for every initial guess $x^{(0)}$ if and only if $\rho(M^{-1}N) < 1$, this means that the splitting we used for A is convergent, by definition:

Definition 10. *Let A, M, N be matrix belong $\mathbb{R}^{n\times n}$, where M is nonsingular. The decomposition for $A = M - N$ is a convergent splitting of A if:*

$$\rho(M^{-1}N) < 1 \quad \textit{or, equivalently}\; \rho(NM^{-1}) < 1$$

For a large value of n the solution error decreases in magnitude by approximately a factor of $\rho(M^{-1}N)$ at every iteration. Thus, the smaller the $\rho(M^{-1}N)$ the quicker convergence of the method. A general property for the splitting of A is the following theorem:

Theorem 10. *Let $A = M - N$ be a splitting of A. If A and M are nonsingular then:*

$$M^{-1}NA^{-1} = A^{-1}NM^{-1}$$

By that, the matrices $M^{-1}N$ and $A^{-1}N$ commute and also the matrices NM^{-1} and NA^{-1} commute.

Proof. Using the definition of splitting of A, we obtain this:

$$\begin{aligned} M^{-1} &= (A+N)^{-1} \\ &= A^{-1}(I + NA^{-1})^{-1} \\ &= (I + A^{-1}N)^{-1}A^{-1} \end{aligned}$$

and we have also:

$$\begin{aligned} A^{-1} &= M^{-1} + M^{-1}NA^{-1} \\ &= M^{-1} + A^{-1}NM^{-1} \end{aligned}$$

which implies the thesis and so:

$$M^{-1}NA^{-1}N = A^{-1}NM^{-1}N$$

and

$$NM^{-1}NA^{-1} = NA^{-1}NM^{-1}$$

$\square$

By the above theorem, we can deduce the following result:

Corollary 3. *Let $A = M - N$ be a splitting of A, with A and M nonsingular matrices. Then by the theorem (10), both matrices $A^{-1}N$ and $M^{-1}N$ have the same eigenvectors; and in addiction both matrices NM^{-1} and NA^{-1} have the same eigenvectors.*

Having in mind the above process described, by the equation 4.5 and the initial guess $\mathbf{x}_0 = \mathbf{0}$, we have to solve this system that is in the form:

$$\begin{aligned} M\mathbf{u}_{k+1}^n &= N\mathbf{u}_k^n + M^{-1}\mathbf{b} \\ &= N\mathbf{u}_k^n + \hat{\mathbf{b}} \end{aligned}$$

To solve it we use the GMRES method. A fundamental insight is to note that a suitable choice of M can lead to a good approximation of A, because we can see M as a preconditioner of A and we can use the matrix M as preconditioner for the linear system. Aiming to solve the preconditioned problem, let us denote the solution as $\mathbf{x}_{Sol}$, then we can rewrite the equation (4.5) as follows:

$$\begin{aligned} \mathbf{u}_{Sol} &= M^{-1}N\mathbf{u}_{Sol} + M^{-1}\hat{\mathbf{b}} \\ (I - M^{-1}N)\mathbf{u}_{Sol} &= M^{-1}\hat{\mathbf{b}} \\ (I - I + M^{-1})A\mathbf{u}_{Sol} &= M^{-1}\hat{\mathbf{b}} \\ M^{-1}A\mathbf{u}_{Sol} &= M^{-1}\hat{\mathbf{b}} \qquad (4.4) \end{aligned}$$

By this formulation, the fixed point iteration in (4.5) can be seen as a left-preconditioned system, alternatively using the Theorem (10) the formulation can be expressed as a right-preconditioned system. Thereby, we are expecting to the more M is a good preconditioner of A the that faster the iterative method used converges. Since the convergence speed relies on the spectral radius $\rho(M^{-1}N)$, and also for the splitting method converges if and only if the spectral radius $\rho(M^{-1}N) < 1$.

4.2 Multigrid

In this section, we present a method which does not belong to Krylov space methods, indeed it is a Multigrid method; where one can find more details in [24]. The Multigrid methods are a class of methods that are used for solving or preconditioning. The description of the Multigrid method will concern the one-dimensional problem, furthermore, we will implement the Multigrid V-cycle. Under careful choice of the parameters involved, these methods can produce a convergence rate which does not depend on the problem's dimension. In fact, another kinds of solvers or preconditioners usually have an increment of the number of iterations as the interval of the grid considered becomes smaller; this leads to a computational cost which does have a super-linear behavior. Instead using the Multigrid method we can reach a computational cost that follows $\mathcal{O}(n)$. To understand the idea behind the Multigrid method we consider the two-level method, in which we have two grids, a fine one and a coarse one, and then move from one another using some operators. These operators are:

- restriction going to the finer grid to coarser one
- prolongation or interpolation going from the coarser grid to the finer one

Using these operators we can use the coarser grid in order to compute a better initial guess for the finer grid. This is an advantage since the solution computed in the coarser grid is much cheaper due to the reduction of the problem size. The two-level method has the following steps:

1. Resolve on the finer grid the linear system $A^h\mathbf{u}^h = \mathbf{b}^h$ on Ω^h. This operation is called smoothing or relaxation. This step given just an rough approximation of $\mathbf{u}^h$ equals to $\mathbf{v}^h$
2. Compute the residual $\mathbf{r}^h = \mathbf{b}^h - A^h\mathbf{v}^h$
3. Restrict the residual to Ω^h, the result will be denoted by $R(\mathbf{r}^h)$
4. Solve directly $A^{2h}\mathbf{e}^{2h} = R(\mathbf{r}^h)$, to have an approximation of the error $\mathbf{e}^{2h}$
5. Interpolate the error $\mathbf{e}^2h$ to Ω^h, the result will be called $P(e^{2h})$
6. Update the approximation of the solution on Ω^h through $\mathbf{v}^h = \mathbf{v}^h + P(\mathbf{e}^{2h})$

Now, we defined the space S_{2h} as the space of function defined in the coarser grid, which is the space of linear combinations of basis functions defined on the coarser grid; while with S_h we defined the space of functions belonging to the fine grid. Defining the space of function on the nodes that are just in the finer grid with B_h, by its construction the space $S_h = S_{2h} + B_h$ and so $S_{2h} \subseteq S_h$. As such, before we can pass from the finer to the coarse grid, we have to apply a sort of reduction to the coarser grid elements called. In the next few pages, we will discuss the definition of the system on the coarse grid, how to restrict the residual to the coarse grid and how to interpolate the correction to the fine grid.

4.2.1 Restriction

The simplest restriction we will describe is the restriction by injection. It is defined by a restriction operator denoted by I_h^{2h} which:

$$I_h^{2h} : \Omega^h \to \Omega^{2h}$$

The operator I_h^{2h} apply on the residual v^h, we obtain:

$$\mathbf{v}^{2h} = I_h^{2h}\mathbf{v}^h \implies v_j^{2h} = v_{2j}^h, \qquad j = 1, \dots, \frac{N}{2} - 1$$

Where in fact we have that $\Omega^h \subseteq \mathbb{R}^{N-1}$ and $\Omega^{2h} \subseteq \mathbb{R}^{N/2-1}$. Basically, this form of restriction takes for each node on the coarser grid the value of the grid function at the relative node on the finer grid. As such, using the restriction operator R by injection, it turns out that we will ignore the values of the residual in the nodes which belong to B_h, which leads to an inefficient method.
A solution for that can be the weighted restriction that uses all nodes on the finer grid. Now we will define the weight restriction operator that is defined as follows:

$$\mathbf{v}^{2h} = I_h^{2h} \implies v_j^{2h} = \frac{1}{4}(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h) \qquad j = 1, \dots, \frac{N}{2} - 1$$

In the case the space Ω^h and Ω^{2h} have standard bases, the matrix representation for the weighted restriction operator has the following structure:

$$I_h^{2h} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & & \\ & 1 & 2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 2 & 1 \end{pmatrix}$$

where $I_h^{2h} \in \Omega^{(N/2-1)\times(N-1)}$.

4.2.2 Prolongation or Interpolation

The other operator we will introduce is the interpolation operator, which transfer the correction from the coarser grid to the finer one. For our Multigrid implementation we will just consider the linear interpolation, which is defined as a local averaging. Let the residual value solved in the coarser grid v^{2h} belongs to Ω^{2h}, then the interpolation operator is defined as:

$$I_{2h}^h : \Omega^{2h} \to \Omega^h$$

Having this operator, we apply it to the vector $\mathbf{v}^{2h}$. The definition of the linear operator is given by:

$$v_{2j}^h = v_j^{2h}, \qquad j = 1, \ddots, N/2 - 1$$
$$v_{2j+1}^h = \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \qquad j = 0, \ddots, N/2 - 1$$

The operator can be expressed in matrix form, for which we use the standard basis for $\mathbb{R}^{N/2-1}$ and $\mathbb{R}^{N-1}$ to obtain:

$$I_h^{2h} = \frac{1}{2}\begin{pmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & \ddots & \\ & & \ddots & \\ & & \ddots & 1 \\ & & & 2 \\ & & & 1 \end{pmatrix}$$

where $I_{2h}^h \in \Omega^{(N-1)\times(N/2-1)}$. Moreover this operator is a linear operator. For these representations for the interpolation and the restriction operators, one might note a relationship between the weighted restriction and the interpolation:

$$I_{2h}^h = 2(I_h^{2h})^T$$

In fact, which relationship comes out from the relationship between the coarser grid and the finer one. The interpolation operator will be denoted as P.

4.2.3 Smoothing

The other fundamental key of Multigrid is a smoothing operator, a classical choice for this operator is a stationary iterative method: Jacobi or Gauss-Seidel. These kinds of methods perform splitting of the linear matrix. The Jacobi method takes a M matrix as the diagonal of the matrix A. Besides, the Gauss-Siedel method takes M equals the lower triangular of A. To these smoother operators there are other alternatives, in our implementation we chose Gauss-Seidel. Applying the smoother, we call the residual $\mathbf{r}^k = \mathbf{b} - Ax^k = A\mathbf{e}^k$ the residual at the k-th step. Then after just a few iterations the component of the error vector $\mathbf{e}^k$ is almost zero. It follows the residual $\mathbf{r}^k$ can be restricted to a coarser grid, so we have:

$$\mathbf{r}^h = R\mathbf{r}^{2h} = RA\mathbf{e}^{2h}$$

Now we have all the operator we can summarize the algorithm for the two-grid cycle:

Algorithm 4 Two-grid

```
   Input: Choose $\mathbf{u}_0$
1: repeat
2:     Smooth $A^h\mathbf{u}_i^h = \mathbf{b}^h$:
3:     Restrict residual $\mathbf{r}^{2h} = R(\mathbf{b} - A\mathbf{u}_i^h)$;
4:     Solve the coarser grid correction $\mathbf{r}^{2h} = A^{2h}\mathbf{e}^{2h}$
5:     Interpolate the error and update $\mathbf{u}_i^h + P\mathbf{e}^{2h} \to u_{i+1}$
6: until Convergence
```

4.2.4 Multigrid V-cycles

Now, a great improvement for this algorithm to do is to generalize the two-grid cycle to a generic number grids. First of all, we can improve the two-cycle adding at the end of every iteration another step for the smoother, in order to improve the solution. The easiest implementation of this idea is the V-cycle: we suppose we have $l+1$ grids, starting from the finest grid, the residual is restricted from one grid to the next and the coarse grid correction is solved only when the coarsest grid. The finest grid step is h and going down in coarser grids the grid spacing increase by the coefficient 2. We summarize the algorithm for the V-cycle iteration here below:

Algorithm 5 Multigrid V-cycle

```
    Input: $A^i i = 1, ..., l+1; b^1$
    Output: $u^1$
 1: Initialize $u^i$ to zero vectors;
 2: Initialize $R^i$ to zero vectors;
 3: for $i = 1, \dots, l$ do
 4:     Relax $A^i u^i = b^i$, set solution equals to variable $v^i$
 5:     Reduction $b^{(i+1)} = I_h^{2h} r^h = I_h(b^i - A^i v^i)$
 6: end for
 7: Solve $A^{(l+1)} u^{(l+1)} = b^{(l+1)}$
 8: for $i = l, \dots, 1$ do
 9:     Interpolation $v^i = v^i + I^i_{(i+1)} v^{(i+1)}$
10:     Relax $A^i u^i = b^i$
11: end for
```

The correct choice of the parameters in the relaxation operation can lead the method as a solver of a linear system that always converges in the same number of iteration that does not depend on the dimension of the problem. the algorithm in (5) is for a V-cycle iteration, then setting a tolerance, we will solve the linear system using several iterations of V-cycles one after the other until the error of the solution satisfies the chosen tolerance.

4.3 Tridiagonal Solver

Despite the other method presented in this chapter, the next method is a direct solver, which means that it finds the exact solution, which can be performed only with a tridiagonal linear system and it is also called Thomas algorithm. Furthermore, it is a simplified method of the more famous Gaussian Elimination method. The linear system, in this case, is in this form:

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

In this case the matrix of linear system if tri-diagonal, keeping in mind the notation before, we define the matrix A in which $a_{ii} = b_i,\ i = 1, 2, \ldots, n$; $a_{i,i-1} = a_i\ i = 2, 3, \ldots, n$ and $a_{i+1,i} = c_i\ i = 1, 2, \ldots, n-1$. At the k-row we have:

$$x_k = A_k x_{k+1} + Y_k, \qquad k = 1, 2, \ldots, n-1,$$

While for the last row we have:

$$x_n = Y_n$$

Where the coefficient A_k and Y_k are coefficients found in the forward step, for which we define $A_n = 0$ and we then get that:

$$x_k = A_k x_{k+1} + Y_k, \qquad k = 1, 2, \ldots, n \tag{4.5}$$

Now, for the first equation of the linear system, the coefficients in the equation above become:

$$A_1 = -\frac{c_1}{b_1} \qquad Y_1 = \frac{y_1}{b_1} \tag{4.6}$$

Then, computing all the other coefficient A_k and Y_k until to fixed $1 \leq k \leq n-1$; we can substitute in the equation $x_k = A_k x_{k+1} F_k$ the equation number $k+1$ of the linear system, which becomes:

$$x_{k+1} = A_{k+1} x_{k+2} + Y_k \tag{4.7}$$

where the two coefficients A_k and Y_k for $k = 1, 2, \ldots, n-1$ are:

$$A_{k+1} = -\frac{c_{k+1}}{b_{k+1} + a_{k+1} A_k} \qquad Y_{k+1} = \frac{y_{k+1} - a_{k+1} Y_k}{b_{k+1} + a_{k+1} A_k}$$

As one can see, this algorithm gets split into two stages: the first one we compute the coefficients A_k and Y_k for the steps $k = 1, 2, \ldots, n$ using the equation (4.6) and (4.7), while for the second we solve backward elimination for the actual unknowns $x_n, x_{n-1}, \ldots, x_1$ using the equation (4.5). Here below, we are show the pseudo-code for the tridiagonal-solver:

Algorithm 6 Tridiagonal matrix Algorithm

```
    Input: $A \in \mathbb{R}^{n \times n}$, $y \in \mathbb{R}^n$;
 1: $b_1 = A_{1,1}$
 2: for $i = 2, 3, \ddots, n$ do
 3:     $a_{i-1} = A_{i,i-1}$;
 4:     $b_i = A_{i,i}$;
 5:     $c_{i-1} = A_{i,i-1}$;
 6:     $w = \frac{a_{i-1}}{b_{i-1}}$;
 7:     $b_i = b_i - w c_{i-1}$;
 8:     $y_i = y_i - w d_{i-1}$;
 9: end for
10: $x_n = \frac{y_n}{b_n}$;
11: for $i = n-1, \ldots, 1$ do
12:     $x_i = \frac{y_i - c_i x_{i+1}}{b_i}$;
13: end for
```

The algorithm can be unstable, the condition to be stable is the diagonal dominance of the linear system matrix:

$$||b_i|| \geq ||a_i|| + ||c_i||$$

For all i which are the row of the matrix. If the algorithm is numerically unstable then we have to rearrange the equation that is known as pivoting. We can also estimate the computational complexity of this algorithm: for the forward step the elimination according the equation in (4.6) and (4.7) are $\mathcal{O}(n)$ arithmetic operation; for the backward step the equation in (4.5) also requires $\mathcal{O}(n)$ arithmetic operation. Therefore, the complexity of the above algorithm is $\mathcal{O}(n)$. Comparing it with the classical Gaussian elimination that is $\mathcal{O}(n^3)$, for large matrix the tridiagonal solver is way faster than that one.

Chapter 5

Numerical results

In this last chapter, we are going to illustrate the results of the numerical experiments carried out. The properties of the matrices will be described and used in order to find numerical results, which will be subsequently compared with the actual solutions found using the closed formula in (1.19) for Merton model in jump-diffusion. For our implementation, we used the programming language MATLAB. We also compared the implemented solving methods with MATLAB's function "\". The financial option problems usually comprehend two main categories: European call option and European put option. From the definition of the underlying PDE in (1.18), the difference between the two options only relies on the definition of the boundary conditions. Due to this fact, for our purpose, we are going to consider the European call option. In the following presented numerical examples, the results are considered at $x = \ln(K)$ and $\tau = T$; furthermore, in order to obtain information on the accuracy of the scheme used to find the numerical solution, we define the "Error between iterative method and closed formula", which can be expressed as:

$$E_{I,C}(x,\tau) = |u_I(x,\tau) - u_C(x,\tau)| \tag{5.1}$$

where we have labeled: $u_I(x,\tau)$ as the solution found using the iterative method, while with $u_C(x,\tau)$ the analytic solution using the analytic formula defined in (1.19). To check the accuracy of the finite difference schemes used, iterative methods, and their implementation, we will compare the iterative method with the MATLAB function "\". Hence, we define the error between them as:

$$E_{I,B}(x,\tau) = |u_I(x,\tau) - u_B(x,\tau)| \tag{5.2}$$

With $x \in (-\ln(S_{max}), \ln(S_{max}))$ and $\tau = T$.

In the equation above, we have labeled $u_B(x, \tau)$ the price function found using the MATLAB "Backslash" method. The numerical experiments we will conduct are the following:

Option	**Schema**	**Methods**
Call	Implicit Euler	Multigrid No-Preconditioned GMRES Preconditioned GMRES
	Crank-Nicolson	No-Preconditioned GMRES Preconditioned GMRES
	Implicit-Explicit	Thomas tridiagonal solver Backslash solver

Table 5.1: Discretization schemes and methods

Now, we define the numerical example for a European call option. To this aim, we have chosen the following benchmark values:

$$\sigma = 0.25, \quad r = 0.025, \quad T = 1, \quad K = 1, \quad \lambda = 0.2, \quad \gamma = 0.5.$$

In order to compare the different exposed schemes with one other, we define some default space-time grids, namely:

Grid	**Spatial points**	**Time points**
G65	65	9
G129	129	17
G257	257	33
G513	513	65
G1025	1025	129
G2049	2049	257
G4097	4097	513
G8193	8193	1025
G16385	16385	2049

Table 5.2: Properties of the grids

We have taken into account the computed price and the pricing error at the strike price ($S = K = 1 \rightarrow x = 0$, in our grid setting) for each implemented method. To perform our schemes we have to truncate the underlying's domain to set $S \in (S_{\min}, S_{\max})$, where for the next implementations $S_{\max} = 5$ and $S_{\min} = 0.2$ are chosen. Then using the change of variable in the equation (1.18), we obtain $x \in [\ln(S_{\min}), \ln(S_{\max})]$. In the considered grids, we have not taken into account the two extremes nodes in the spatial domain, which represent the boundary conditions of our problem; the reason is given by the fact that for those nodes we already know the resulting price value obtained from the boundary conditions themselves on (3.3) and (3.5).

5.1 Implicit Euler scheme

As first discretization scheme, we perform the implicit Euler scheme following the framework in (3.2), using benchmark values. Based on the implicit Euler scheme approximations for time integration, we get a full discrete linear system as follows:

$$A\mathbf{u}^n = \mathbf{b} \tag{5.3}$$

which is a recall for the linear system in (3.26). The linear system involved is represented by the unknown term $\mathbf{u}^n$ and the matrix A. Through the boundary conditions for $\tau = 0$, we start solving the first linear system at this point. We lead back the algorithm to a N linear system, each one depending on the previous step as stated in (5.3). In the next table, we are going to recall two important properties of the linear system matrix, which are critical for the resolution of the linear system using iterative methods. The two properties are the following: condition number and spectral radius.

Spatial points	**Time points**	**Spectral radius**	**Condition number**
65	9	7.592	7.524
129	17	13.754	13.692
257	33	26.101	26.041
513	65	50.805	50.747
1025	129	100.218	100.161
2049	257	199.048	198.991
4097	513	396.708	396.652
8193	1025	792.031	791.974

Table 5.3: For each spatial point and time point, the table provides the corresponding spectral radius $\rho(A)$ and the condition number related to the linear system matrix. The condition number has been computed with the MATLAB function cond(A).

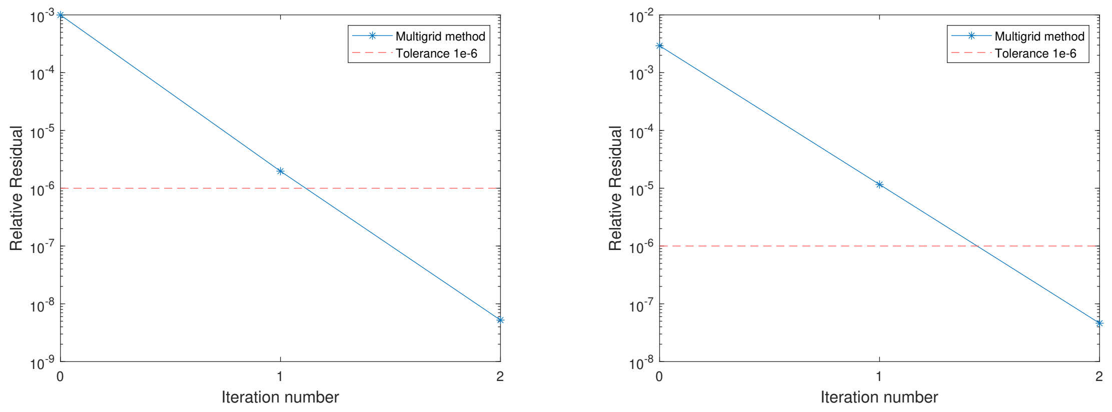
5.1.1 Multigrid V-cycle

The first iterative method we will show is the Multigrid method. For the Multigrid has been used the Multigrid V-cycle. We followed the algorithm in (5). For this algorithm we used as relaxation the Jacobi-Seidel algorithm. Since the Multigrid travels down the various grid, using the V-cycle Multigrid the grid interval doubles itself, the value L becomes the number of points to travel down the coarsest grid which depends on the number of points, it is defined as $L = 1 \dots 2^{n-1}$. For this method, we have tuned the parameters related to: the numbers of relaxation, numbers of V-cycle, and the level for the method; this has been done in order to get computational cost behaved following $\mathcal{O}(n)$. By doing this, the computational cost does not depends on the problem size. The tolerance chosen is $1e-06$, to satisfy the Multigrid exit condition. Here below a table will show the results using the Multigrid method:

Spatial points	**Time points**	**Multigrid error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
65	9	2.569e-03	2.233e-10	24	0.020
129	17	1.178e-03	1.795e-10	48	0.084
257	33	5.626e-04	4.850e-10	96	0.295
513	65	2.747e-04	8.775e-10	256	1.644
1025	129	1.357e-04	7.866e-10	640	12.677
2049	257	6.749e-05	7.816e-09	1280	145.289

Table 5.4: Implicit-scheme Multigrid V-cycles. The tolerance value of Multigrid is $1e - 06$. "Iterative error" is the error defined in (5.1), which is computed at $K = 1$ at $T = 1$, while "Iterative vs. Direct" is given by 5.2 with K = 1 at T = 1. "Total iterations" is the number of the V-cycles of Multigrid for all time steps. "Iterative time" is the required time to solve all the linear systems.

Here below, we will show the profile convergence of this Multigrid method:



(a) Multigrid with grid of: spatial points 129 and time points 17.

(b) Multigrid with grid of: spatial points 257 and time points 33.

Figure 5.1: Implicit Euler scheme with Multigrid method V-cycles with different grids.

As we can see, for the first three grids G65, G129 and G257, the number of V-cycles and so the number of operations have a behavior following $\mathcal{O}(n)$. In fact, the parameters related to the Jacobi relaxation have been tuned to keep the number of operations the same despite the increase in problem size. This kind of behavior has not held for the grid G513, while it has held in the last two grids considered.

5.1.2 GMRES no-Preconditioned

According to the discretization scheme, the linear system matrix is a non-symmetric full matrix. Hence, an optimal iterative method to solve this kind of matrix would be GMRES method for these types of linear system, which is described in (4.1.2). Moreover, we are looking for a scalable preconditioner that allows GMRES to converge in a low number of iterations that does not get worse whereas the grid becomes finer. First of all, we will solve those linear systems without any preconditioners. In the following table we show the results for the GMRES with no preconditioner:

Spatial points	**Time points**	**Iterative error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
65	9	2.568e-03	2.095e-10	248	0.022
129	17	1.178e-03	1.158e-09	688	0.053
257	33	5.626e-04	2.436e-09	1888	0.231
513	65	2.747e-04	3.632e-09	5248	0.644
1025	129	1.356e-04	6.575e-09	14464	9.049
2049	257	6.740e-05	1.030e-08	40192	75.161

Table 5.5: Implicit-scheme solved with GMRES with no-preconditioner, the tolerance is $1e-10$. "Iterative error" is the error defined in (5.1), which is computed at $K = 1$ and $T = 1$, while "Iterative vs. Direct" is given by (5.2) with $K = 1$ at maturity time $T = 1$. "Total iterations" are the iterations of GMRES for all time steps. "Iterative time" is the required time to solve all the linear systems.

5.1.3 GMRES preconditioned

Before all else, we define the matrix of our linear systems' problem: A which has the following properties:

- Toeplitz matrix
- Non-Symmetric matrix
- Diagonally dominant
- M-matrix
- Values off the three-main diagonal going to zero

The matrix A, having these properties coming from section (3.2), has the following structure:

$$A = \begin{pmatrix} a_0 & a_{-1} & a_2 & \cdots & \cdots & \cdots & \cdots & a_{n-1} \\ a_1 & a_0 & a_{-1} & a_2 & & & & \vdots \\ a_2 & a_1 & a_0 & a_{-1} & \ddots & & & \vdots \\ \vdots & a_2 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & a_2 & \vdots \\ \vdots & & & \ddots & a_1 & a_0 & a_{-1} & a_2 \\ \vdots & & & & a_2 & a_1 & a_0 & a_{-1} \\ a_{n-1} & \cdots & \cdots & \cdots & \cdots & a_2 & a_1 & a_0 \end{pmatrix}$$

For the given problem, a solution for the GMRES is to find a preconditioner M, thereby we are expecting the more M is a good preconditioner of A the that faster the iterative method used converges. As we have seen in Chapter 3 in the section (5.1.3), an option for preconditioner is to use the matrix splitting. Therefore, using that, we will find a perconditioner for the given problem. According to the implicit Euler scheme, the linear system matrix off three main diagonal values are close to zero. This is proven by the fact that the elements out the main three diagonals are the values of the matrix $\hat{T}$ in (3.19). Subsequently, we have chosen two preconditioners M for the splitting problem as follows: We implemented two left preconditioners M for this splitting problems:

- M is a n-diagonal matrix, whose diagonals are taken by a tolerance from A
- M is a tridiagonal matrix, whose diagonals are the main three diagonal of A

Since the spectral radius and the condition number are particularly important when we are dealing with an iterative method for linear system, and also for the splitting method converges if and only if the spectral radius $\rho(M^{-1}N) < 1$; thereby we show below these matrix properties for the left-preconditioned problem, for the grid, we have considered:

Spatial	**Time**	**Spectral Radius**		**Condition number**	
points	**points**	n-Diagonal	Tridiagonal	n-Diagonal	Tridiagonal
257	33	1.000	1.000	1.000	1.024
513	65	1.000	1.000	1.000	1.012
1025	129	1.000	1.000	1.000	1.006
2049	257	1.000	1.000	1.000	1.003
4097	513	1.000	1.000	1.000	1.001
8193	1025	1.000	1.000	1.000	1.000

Table 5.6: Spectral number and condition number consider the two splitting problems as preconditioners. We computed the spectral radius of $\rho(M^{-1}A)$ and the condition number as $\text{cond}(M^{-1}A)$; the values are approximated at the sixth digits with a tolerance of $\pm 1e - 07$.

As can be seen from the table above the two preconditioners both significantly reduced the spectral radius and the condition number compared to the results in the no-preconditioned problem (5.3). This leads to the fact that the two preconditioners are good preconditioners for the linear system matrix A. For the n-diagonals preconditioner we chose $tol = 1e - 06$ for the grids G257, G513, G1015 and G2049; for G4097 and G8193 $tol = 1e - 07$, while for the last grid G16385 $tol = 1e - 08$.

Spatial points	**Time points**	**Iterative error**	**Iterative vs. direct**	**Total iterations**	**Tolerance diagonals**	**Iterative time**
257	33	5.626e-04	2.546e-13	64	1e-06	0.163
513	65	2.747e-04	1.081e-11	128	1e-06	0.670
1025	129	1.356e-04	4.949e-11	256	1e-06	2.720
2049	257	6.739e-05	5.979e-12	632	1e-06	9.952
4097	513	6.739e-05	8.286e-12	1536	1e-07	56.410
8193	1025	1.672e-05	6.272e-11	3072	1e-07	306.004
16385	2049	8.321e-06	1.511e-11	6144	1e-08	1839.300

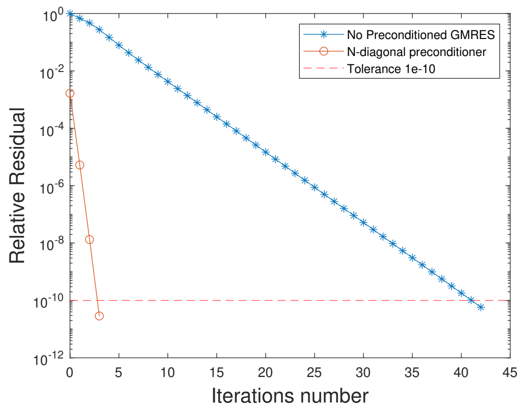
Table 5.7: Implicit-scheme with preconditioned GMRES, the preconditioner is the n-diagonals, taken from the linear system matrix greater than a tolerance, which is factorized with ILU. GMRES tolerance $1e - 10$. "Iterative error" is the error in the formula (5.1), which is computed at $K = 1$ and $T = 1$."Iterative vs. Direct" is given by (5.2) at $K = 1$ and $T = 1$. "Total iterations" are the iterations of GMRES for all time steps. "Tolerance diagonals" is the tolerance chosen to draw the matrix diagonal elements for the n-diagonals preconditioner. "Iterative time" is the required time to solve all the linear systems.

In the next table, we show the results of the second preconditioner M, which is composed of the three diagonals of the linear system matrix.

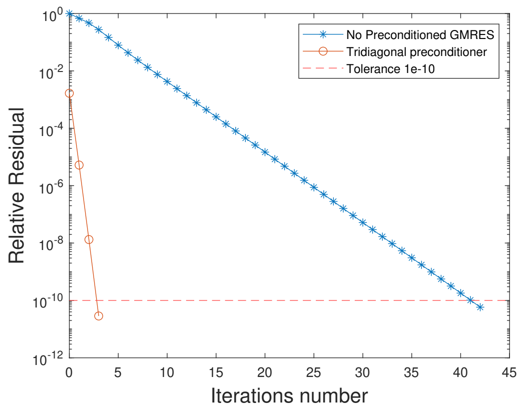
Spatial points	**Time points**	**Iterative error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
257	33	5.626e-04	2.306e-10	128	0.185
513	65	2.747e-04	2.877e-11	256	0.593
1025	129	1.356e-04	5.869e-09	384	2.456
2049	257	6.739e-05	1.374e-09	768	9.963
4097	513	3.356e-05	3.195e-10	1536	46.603
8193	1025	1.672e-05	6.272e-11	3072	213.041
16385	2049	8.321e-06	1.354e-10	6144	1216.538

Table 5.8: Implicit-scheme GMRES preconditioned, the preconditioner is the tridiagonal matrix of the linear system matrix and factorized with ILU. GMRES is the iterative method used with tolerance tol= $1e - 10$. "Iterative error" is the error formula in (5.1), which is computed as $K = 1$ at $T = 1$, while "Iterative vs. Direct" is given by (5.2) with $K = 1$ at $T = 1$. "Total iterations" are the iterations for all time steps. "Iterative time" is the required time to solve all the linear systems.

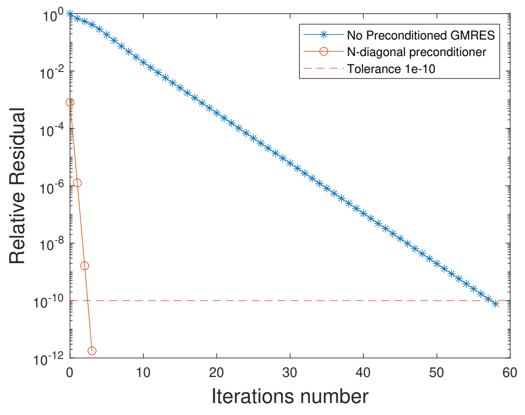
We are also going to report the convergence profile of implicit-scheme of GMRES comparing the no-preconditioned GMRES to the two preconditioners we have applied using the splitting-problem. For sake of visualization we took two grids with not many iterations: G129 and G257.



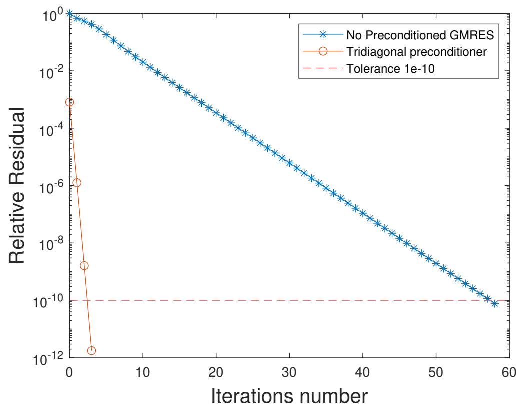
(a) Grid G129. GMRES with no-preconditioner compared with N-diagonal preconditioner with ILU facotrization.



(b) Grid G129. GMRES with no-preconditioner compared with tridiagonal preconditioner with ILU facotrization.



(c) Grid G257. GMRES with no-preconditioner compared with N-diagonal preconditioner with ILU facotrization.



(d) Grid G257. GMRES with no-preconditioner compared with tridiagonal preconditioner with ILU facotrization.

Figure 5.2: Comparison of iteration numbers for: GMRES with no preconditioner, preconditioned GMRES splitting problem. The preconditioners are: N-diagonal (a),(c) and Tridiagonal (b),(d). The preconditioners are both factorized using ILU factorization. The computations are at maturity time $T = 1$.

Comparing the results on the table (5.6) and the others from (5.7) and (5.8), we notice different performances. Indeed, using the preconditioner we have reduced the condition number of the matrix. Therefore, it has led to a relevant reduction of the number of total iterations for the two preconditioned GMRES in tables (5.7) and (5.8). As the last result, there has been a reduction in the computation time in both of them. Analyzing the two preconditioned problems, we can notice that the average number of iterations for each time step using the n-diagonal preconditioner and the three-diagonal preconditioner tend to be the same fining the grid. Despite the number of iterations for the finest grids are equals for the two types of preconditioned problem, the error with the closed formula is slightly higher for the tridiagonal preconditioner since it approximates slightly less accurate the linear system matrix by the fact that in this preconditioner we have just the tri-main diagonals. Moreover from (5.6), the condition number for tridiagonal preconditioner is quite bigger than the n-diagonal preconditioner and as we can see it leads to a less accurate tridiagonal-preconditioned problem. In addition, the more element has the matrix, the more the iterative time increase for the n-diagonal preconditioned problem; in fact the memory cost and the multiplication cost is greater for the n-diagonal preconditioned matrix, by the construction of the preconditioned, than the tridiagonal preconditioned problem. From the results, we deduce that the tridiagonals preconditioner is better than n-diagonals preconditioner for memory cost and the errors are almost the same. Lastly, the two preconditioners are both good preconditioners since the results with the MATLAB function "\" are almost the same, but as we can guess the n-diagonal preconditioned problem is more accurate than the tridiagonal preconditioned.

5.2 Crank-Nicolson scheme

The next approximation method we are going to perform is the Crank-Nicolson scheme, whose implementation is related to section (3.3) of Chapter 2. The linear system's matrix has the formulation in (3.33), which has the same features of the Euler implicit scheme, as we have proven in the relative theoretical section. In the right-hand vector $\hat{\mathbf{b}}$ is defined as:

$$\hat{\mathbf{b}} = \Big(\mathbb{I} - \frac{\Delta\tau}{2}L\Big)\mathbf{u}^n + \frac{\Delta\tau}{2}\mathbf{b}^{n+1} + \frac{\Delta\tau}{2}\mathbf{b}^n$$

The multiplication between the matrix $\Big(\mathbb{I} - \frac{\Delta\tau}{2}L\Big)$ and the vector $\mathbf{u}^n$ can be sped up using the Fast-Fourier Transform since the matrix has a Toeplitz structure. Using FFT algorithm, the product can be computed in $\mathcal{O}(n\log n)$ time.

In the next table, we are going to show, as in the previous section, two key features: condition number and spectral radius.

Spatial points	**Time points**	**Spectral radius**	**Condition number**
65	9	4.296	4.277
129	17	7.377	7.360
257	33	13.550	13.535
513	65	25.902	25.887
1025	129	50.609	50.594
2049	257	100.024	100.009
4097	513	198.854	198.840
8193	1025	396.515	396.501
16385	2049	791.838	791.823

Table 5.9: For each spatial point and time point spectral radius $\rho(A)$ and condition number related to the linear system matrix, the condition number has been computed with MATLAB function cond(A) for Crank-Nicolson scheme.

As we can find in these results, comparing to the results in the table (5.3), the condition number and as well as the spectral radius are halved. This fact leads to more accuracy and well-conditioned problem.

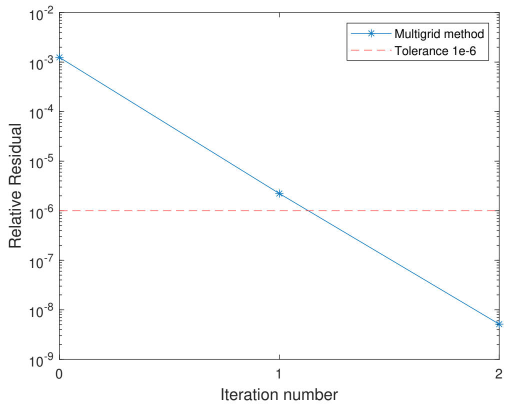
5.2.1 Multigrid V-cycle

As in the previous scheme, the first iterative method we show is the Multigrid method which has the same algorithm as the Multigrid shown in the (5.1.1), in fact we used the V-cycle Multigrid. The Multigrid's exit condition is the tolerance of $1e-06$. Here below a table will show out the results using the Multigrid method.

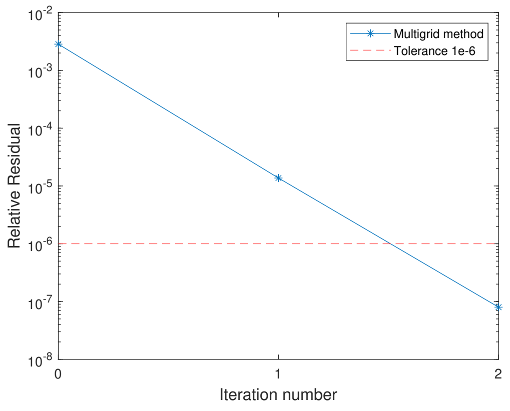
Spatial points	**Time points**	**Multigrid error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
65	9	3.854e-04	1.911e-10	24	0.021
129	17	1.041e-04	6.042e-10	48	0.078
257	33	2.872e-05	7.373e-11	96	0.290
513	65	8.362e-06	2.335e-11	192	1.370
1025	129	2.673e-06	1.978e-09	384	16.579
2049	257	9.673e-07	2.462e-09	1536	137.568

Table 5.10: Implicit-scheme Multigrid V-cycles.
The tolerance value of Multigrid method is $1e-06$. "Iterative error" is the error between iterative method and closed formula defined in 5.1, which is computed at $K=1$ and $T=1$, while "Iterative vs. Direct" is given by 5.2 with $K=1$ at $T=1$. "Total iterations" are the iterations of Multigrid for all time step. "Iterative time" the time for all linear systems.

Here below we will show the profile convergence of this Multigrid method:



(a) Multigrid with grid of: spatial points 129 and time points 17.



(b) Multigrid with grid of: spatial points 257 and time points 33.

Figure 5.3: Implicit Euler scheme with Multigrid method V-cycles in different grids.

As we can see, in this scheme, for the first five grids the number of V-cycles and so the number of operations have a behavior following $\mathcal{O}(n)$. In fact, the parameters related to the Jacobi relaxation have been tuned to keep the number of operations the same despite the increase in problem size. This kind of behavior has not held for the last grid, for which the number of operations has increased drastically.

5.2.2 GMRES no-Preconditioned

According to scheme used, in this linear system, the matrix is a non-symmetric full matrix as well as the Implicit Euler method (3.2). Therefore, we will use the GMRES to compute the solutions for this scheme. Firstly, we show the results for the GMRES without a preconditioner:

Spatial points	**Time points**	**Iterative error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
65	9	3.853e-04	4.193e-10	184	0.069
129	17	1.040e-04	7.330e-10	496	0.043
257	33	2.862e-05	1.415e-09	1344	0.192
513	65	8.261e-06	1.421e-09	3776	0.802
1025	129	2.572e-06	2.063e-09	10496	6.957
2049	257	8.715e-07	3.928e-09	28928	52.632

Table 5.11: Crank-Nicolson scheme computed using GMRES with no-preconditioner. GMRES is the iterative method used with tolerance $tol = 1e - 10$. "Iterative error" is the error defined in (5.1), which is computed at $K = 1$ and $T = 1$, while "Iterative vs. Direct" is given by (5.2) with $K = 1$ at $T = 1$. "Total iterations" are the iteration of GMRES for all time step. "Iterative time" is the time for all linear systems.

5.2.3 GMRES Preconditioned

By the construction of the matrix in (3.28), it has the properties stated in Section (3.3) and furthermore, it contains the larger modules values in the three main diagonal as well as the implicit Euler scheme. By that, following the splitting iteration in section (5.1.3), we will use the three main diagonals of A as preconditioner. Here is the calculus of the spectral radius and condition number for the left-preconditioned problem, for the grids we have considered:

Spatial points	**Time points**	**Spectral radius**	**Condition number**
257	33	1.000	1.012
513	65	1.000	1.010
1025	129	1.000	1.008
2049	257	1.000	1.006
4097	513	1.000	1.005
8193	1025	1.000	1.003
16385	2049	1.000	1.002

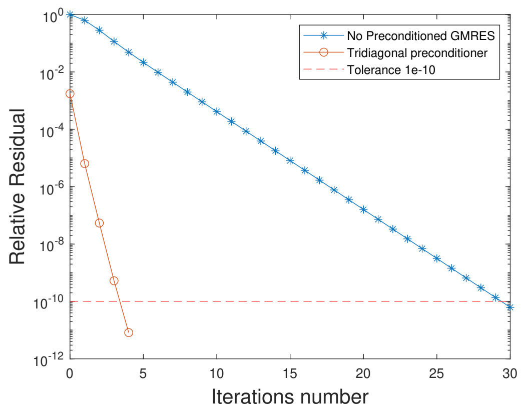
Table 5.12: Spectral number and condition number for the left-preconditioned for the Implicit scheme. We computed the spectral radius of $\rho(M^{-1}A)$ and the condition number as $cond(M^{-1}A)$.

As can be seen from the table above the preconditioner significantly reduced the spectral radius and the condition number compared to the results in the no-preconditioned problem (5.3). This leads to the fact that the preconditioner is a good preconditioner for the linear system matrix A. In the next table we will show the result for the Crank-Nicolson scheme using GMRES preconditioned:

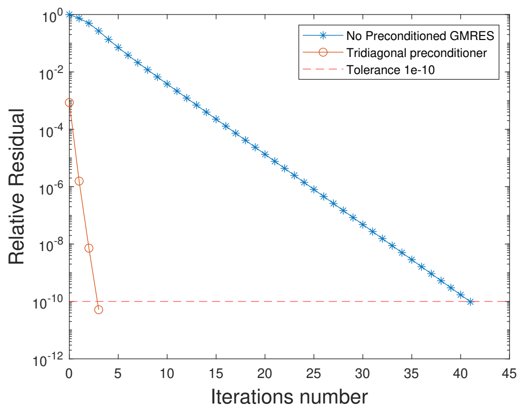
Spatial points	**Time points**	**Iterative error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
257	33	2.862e-05	2.251e-10	144	0.159
513	65	8.362e-06	5.740e-11	262	0.335
1025	129	2.674e-06	1.635e-09	514	2.449
2049	257	9.703e-07	5.135e-10	814	10.328
4097	513	4.064e-07	1.562e-10	1552	48.204
8193	1025	1.967e-07	4.001e-11	3077	253.147
16385	2049	1.110e-07	3.908e-10	6145	1250.747

Table 5.13: Crank-Nicolson scheme computed using GMRES with preconditioner, the preconditioner is the tridiagonal matrix of the linear system matrix with ILU decomposition. GMRES is the iterative method used with tolerance tol= $1e-10$. "Iterative error" is the error defined in (5.1), which is computed at $K = 1$ and $T = 1$, while "Iterative vs. Direct" is given by (5.2) with $K = 1$ and $T = 1$. "Total iterations" are the iteration of GMRES for all time steps. "Iterative time" is the time for all linear systems.

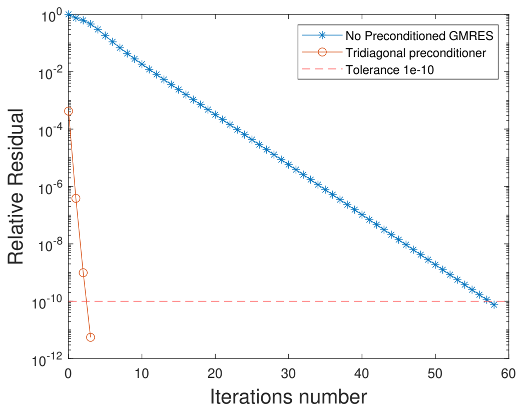
The results in the table (5.13) show that the preconditioner used has improved the problem and so the number of iterations has significantly decreased. Consequently, the computation times of the iterative method have decreased. The iterative error along the grids is reduced by a larger factor than in the first scheme used, it is due to the fact that the second-order accuracy over the time. We show below, the convergence profiles of two grids considered with a comparison between the no-preconditioned and the preconditioned GMRES.



(a) Crank-Nicolson with grid G129. GMRES with no-preconditioner compared with tridiagonal preconditioner with ILU factorization



(b) Crank-Nicolson with grid G257. GMRES with no-preconditioner compared with tridiagonal preconditioner with ILU factorization



(c) Crank-Nicolson with grid G513. GMRES with no-preconditioner compared with tridiagonal preconditioner with ILU factorization

Figure 5.4: Crank-Nicolson scheme, comparison with different grids.

5.2.4 GMRES preconditioned: halved time steps

By the construction of the Crank-Nicolson scheme, we can halve the time steps keeping a good level of accuracy for that and moreover the iterative time will be halved. Here below, we will halve the time steps and use the GMRES preconditioned:

Spatial points	**Time points**	**Iterative error**	**Iterative vs. direct**	**Total iterations**	**Iterative time**
257	17	4.472e-04	3.335e-11	82	0.078
513	33	2.187e-04	2.719e-10	142	0.300
1025	65	1.080e-04	6.024e-11	261	1.259
2049	129	5.372e-05	9.334e-12	513	5.139
4097	257	2.679e-05	1.793e-09	814	22.404
8193	513	1.339e-05	1.012e-09	1552	110.055
16385	1025	6.710e-06	7.941e-10	3077	608.353

Table 5.14: Crank-Nicolson scheme, time steps halved, computed using GMRES with preconditioner, the preconditioner is the tridiagonal matrix of the linear system matrix with ILU decomposition. GMRES is the iterative method used with tolerance tol= $1e-10$. "Iterative error" is the error between iterative method and closed formula defined in (5.1), which is computed as $K = 1$ at maturity time $T = 1$, while "Iterative vs. Direct" is given by (5.2) with $K = 1$ at maturity time $T = 1$. "Total iterations" are the iterations of GMRES for all time steps. "Iterative time" is the required time to solve all the linear systems.

Comparing the two tables (5.13) and (5.14), we can see that halving the time steps has decreased the total iterations for the grids in table with half time steps. From the point of view of accuracy, in this case, the iterative error has increased as expected. Nevertheless, the grade of accuracy can be compared to the Implicit scheme one, however, keeping a great level of error.
Important fact is that the iterative time for this case has halved as well, leading to a good trade-off between accuracy and computational cost.

5.3 Implicit-Explicit scheme

In this sections, we are going to put into practice the implicit-explicit discretization schema, which has been introduced in the Chapter 2 in (3.4). This type of discretization avoids dense linear system that arises from the integral discretization by approximating it explicitly, and it uses implicit schemes for the second order differential term in the equation. The linear system for this scheme is defined in (3.33), in this equation the right-hand vector label with $\mathbf{r}^{n+1}$ has the following form:

$$\mathbf{r}^{n+1} = S\mathbf{u}^n + \Delta\tau\mathbf{b}$$

By the construction of the S matrix, it is a Toeplitz matrix. Therefore, the product between S and $\mathbf{u}^n$ can be sped up using FFT algorithm.

In the next table, we are going to show the condition number and the spectral radius.

Spatial points	**Time points**	**Spectral radius**	**Condition number**
65	9	7.564	7.569
129	17	13.740	13.742
257	33	26.094	26.095
513	65	50.801	50.802
1025	129	100.216	100.217
2049	257	199.047	199.047
4097	513	396.708	396.708
8193	1025	792.031	792.031
16385	2049	1582.675	1582.675

Table 5.15: For each spatial point and time point spectral radius $\rho(A)$ and condition number related to the linear system matrix, the condition number has been computed with MATLAB function cond(A), for the IM-EX scheme.

The linear system, in this case, has a non-symmetric tridiagonal Toeplitz matrix, for this kind of matrix we have chosen to use two different types of linear system solver:

- Tridiagonal matrix algorithm (also known as Thomas algorithm)
- MATLAB function backslash (defined as ”\”)

5.3.1 Tridiagonal Solver

As first one solver, we have implemented the Tridiagonal algorithm described in (5). As mentioned in Chapter 4 in section (4.3), the convergence condition for this algorithm is that the matrix A of the linear system has to be diagonal dominant. This condition is fulfilled for the numerical approximation method we used. In fact, by the equation (3.33), we can understand that the matrix A is a diagonally dominant matrix. Thus, the tridiagonal solver converges to the solution.

Here below, we present the results for Tridiagonal solver in Im-Ex scheme:

5.3.2 MATLAB: Backslash Solver

As second method we will use MATLAB function "\\". Since the documentation it has multiple checks and calculation. To speed up this function and avoid these kinds of checks, we use LU factorization to define two matrices (L is the lower-tridiagonal of A while U is the upper-tridiagonal of A) reducing the linear systems in two linear system as follows:

$$Ax = b \begin{cases} Ly = b \\ Ux = y \end{cases} \tag{5.4}$$

These kinds of linear systems (Triangular linear system) lead to the MATLAB function "\\ a cheaper computation cost, since it checks for the triangularity of the matrix and therefore it solves the linear system using triangular solver, avoiding permuted triangular and other expensive operations in the case the matrices would be non-triangular. Moreover, the LU decomposition has been computed only once and uses L and U at every time step. Here below, the results using the "\\" of MATLAB:

Spatial Steps	**Time Steps**	**Back-Slash error**	**Iterative Time**
257	33	3.079e-04	0.129
513	65	1.472e-04	0.504
1025	129	7.195e-05	2.025
2049	257	3.557e-05	8.103
4097	513	1.769e-05	32.210
8197	1025	8.840e-06	130.251
16385	2049	4.431e-06	511.499

Table 5.16: Implicit-explicit scheme computed using MATLAB function backslash. "Back-Slash error" is the error defined in (5.1), computed at $K = 1$ and $T = 1$. "Iterative time" is the required time to solve all the linear systems.

Now, we will show below the price function an a three-dimensional reference system, where we have the corresponding definition: $(x, y, z) \rightarrow (\ln(S), t, price)$, computed using Im-Ex scheme.

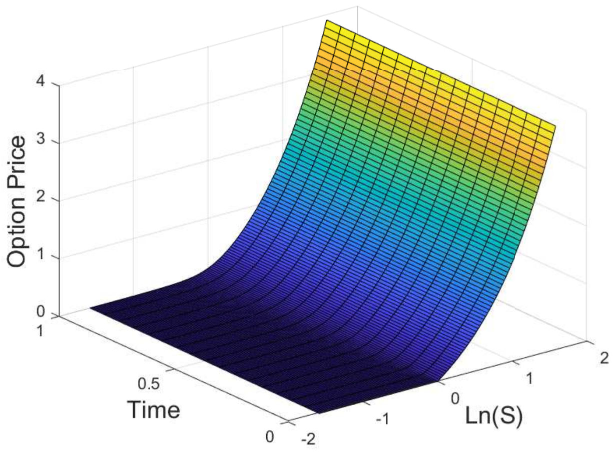
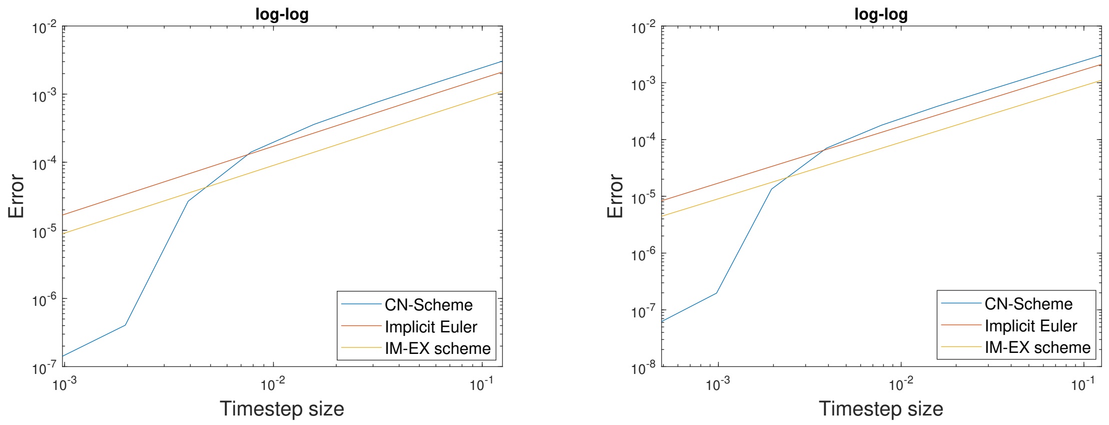


Figure 5.5: The price of European call option under Merton's jump diffusion using the Im-Ex scheme with grid G129

5.4 Schemes comparison

After the implementation of the various scheme, we will compare the distributions of schemes' errors depending on the number of spatial points and time points. Firstly in the next two charts, we will fix the time steps and increase the spatial node in order to display the performance of each discretization over the space domain.



(a) Spatial points: 4097. Evaluation at time points: 9, 17, 33, 65, 129, 257, 513, 1025, 2049.

(b) Spatial points: 8193. Evaluation at time points: 17, 33, 65, 129, 257, 513, 1025, 2049, 4097.

Figure 5.6: Comparison of error distribution with fixed spatial nodes and various time steps. The errors are computed at $S = K$ at $T = 1$, using the formula in (5.1). The charts are log-log-plot.

As expected, the implicit error distribution follows a straight line since it is a first order accuracy in time by the definition of the implicit Euler scheme. Likewise, the implicit-explicit method has a first order accuracy in time as proven in the section (3.4). Moreover, the IM-EX is more accurate than the implicit as we can see from the results in the table (5.13) and (5.14) than the implicit. Besides, the Crank-Nicolson scheme since has a second order accuracy in time so it converges faster. Moreover, for middle spatial steps, the Crank-Nicolson has a more accurate solution than the other schemes used. As concerns the evaluation of the schemes' errors depending on the spatial points, we display two other plots where for those we will fix the numbers of time nodes and range the spatial nodes.

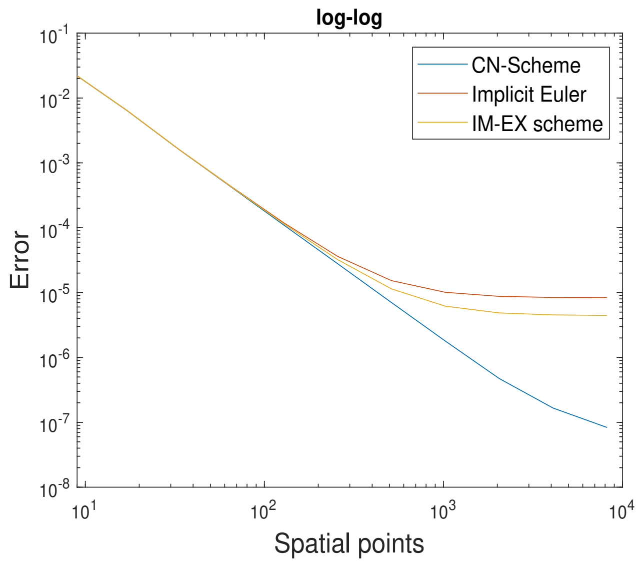


Figure 5.7: Comparison of errors' distributions with fixed time nodes and various spatial steps. The errors are computed at $S = K$ and $T = 1$, using the value of (5.1). Time points: 8193. Evaluation at spatial points: 9,17, 33, 65, 129, 257, 513, 1025, 2049, 4097. The chart is a log-log-plot.

Visualizing the chart above (5.6), we can see the Crank-Nicolson scheme performs better in terms of accuracy than the other two schemes. It is due to the second-order accuracy of the Crank-Nicolson scheme over time. In fact it as well as having slightly better results for the first half of the x-axis. Moreover, it reaches the limit of accuracy, for this spatial point, with larger time points than the other two schemes.

Conclusion

The main purpose of this thesis was to study the Merton jump-diffusion model, in particular, it focused on the implementation of a numerical solution for evaluating the fair price of a European call option deriving from the PDE that describes the model studied.
We developed an implicit scheme using Backward Euler that led to a full linear system, for which we applied a Multigrid V-cycle method. Unfortunately, we did not perform a Multigrid scheme that could maintain scalability in terms of computational cost since, especially for large problems, this metric increased considerably when rising the grid size. On the other hand, using the Preconditioned GMRES we were able to obtain restrained computational costs for very large dimension problems; between the two studied preconditioners, the one that resulted in the most satisfactory outcome was the tridiagonal preconditioner; the reason lies in the fact that it was able to correctly approximate the underlying matrix and bring low storage costs, as one can see for instance for grid G8193 and for the largest one G16385.
The second scheme we implemented was the Crank-Nicolson scheme, which is a method that leads to a great level of accuracy both in time and spatial domain. Following the same approach adopted in the previous implicit scheme, we performed the Multigrid V-cycle method that again could not preserve scalability given the high resulting computational costs. Nevertheless, we managed to perform the analysis even for very large problem sizes using the preconditioned GMRES method and also performed the matrix-vector multiplication thanks to the Fast Fourier Transform (FFT) algorithm given the matrix properties.
The last scheme implemented was the Implicit-Explicit one, which accomplished a great level of accuracy and low computational time, due to the tridiagonal linear matrix, using both the tridiagonal and the MATLAB "\" solver. Moreover, we were able to perform the Crank-Nicolson scheme halving the time nodes on the grid in order to obtain comparable results with the Implicit-Explicit scheme from the point of view of accuracy and computational cost.

Bibliography

[1] Louis Bachelier. Théorie de la spéculation. *Annales scientifiques de l'École Normale Supérieure*, 3e série, 17:21–86, 1900.

[2] David S Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9(1):69–107, 1996.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. In *World Scientific Reference on Contingent Claims Analysis in Corporate Finance: Volume 1: Foundations of CCA and Equity Valuation*, pages 3–21. World Scientific, 2019.

[4] Maya Briani. *Numerical methods for option pricing in jump-diffusion markets.* PhD thesis, Universita degli Studi di Roma La Sapienza, 2003.

[5] John Charles Butcher. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

[6] Rama Cont and Ekaterina Voltchkova. A finite difference scheme for option pricing in jump diffusion and exponential levy models. *Internal Report CMAP, No, 513*, 2003.

[7] Rama Cont and Ekaterina Voltchkova. A finite difference scheme for option pricing in jump diffusion and exponential lévy models. *SIAM Journal on Numerical Analysis*, 43(4):1596–1626, 2005.

[8] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[9] Bruno Dupire et al. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

[10] Massimiliano Ferronato Giuseppe Gambolati. *Lezioni di metodi numerici per l'ingegneria.* Progetto Libreria, 2015.

[11] Gene H Golub and Charles F Van Loan. *Matrix computations.* JHU press, 2013.

[12] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6(2):327–343, 1993.

[13] John C. Hull. *Options, futures, and other Derivatives.*

[14] Sato Ken-Iti. *Lévy processes and infinitely divisible distributions.* Cambridge university press, 1999.

[15] Steven G Kou. A jump-diffusion model for option pricing. *Management science*, 48(8):1086–1101, 2002.

[16] Damien Lamberton and Bernard Lapeyre. *Introduction to stochastic calculus applied to finance.* CRC press, 2011.

[17] Kazuhisa Matsuda. Introduction to merton jump diffusion model. *Department of Economics. The Graduate Center, The City University of New York*, 2004.

[18] Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144, 1976.

[19] Alfio Quarteroni and Silvia Quarteroni. *Numerical models for differential problems*, volume 2. Springer, 2009.

[20] Yousef Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of computation*, 37(155):105–126, 1981.

[21] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[22] Santtu Salmi. Numerical methods for pricing options under jump-diffusion processes. *Jyväskylä studies in computing*, (180), 2013.

[23] Jari Toivanen. Numerical valuation of european and american options under kou's jump-diffusion model. *SIAM Journal on Scientific Computing*, 30(4):1949–1970, 2008.

[24] Pieter Wesseling. Introduction to multigrid methods. Technical report, 1995.