POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea in Ingegneria Matematica

Structural default model with mutual liabilities and jump risk

Relatore: **Prof. Daniele Marazzina**

Candidato:

Giulio Ciofini

Matr. 852953

Anno Accademico 2016-2017

"Ai miei genitori, a mia sorella e alla mia famiglia tutta, grazie ai quali sono la persona che sono e ai quali posso dire solo un sincero grazie"

Acknowledgements

I would like to extend my gratitude to my supervisor Daniele Marazzina for his guidance and advice throughout the preparation of this master thesis. Finally and most importantly I would like to thank my family for their continued emotional and financial support, without which I would not be in this position today.

Contents

Chapter 1

Introduction

Credit risk measurement and credit derivatives pricing is today one of the most intensely studied areas in quantitative finance. The fame of credit derivatives arises from the fact that they allow investors to buy protection against opposed credit event.
Correctly modelling multiple defaults became crucially important after 2008 crisis. In 2008 we saw that even without defaults the deterioration of credit quality is enough to cause potential fatal losses. During the crisis of 2008, the downward trend of financial markets was accompanied by extensive credit deterioration of financial institutions. Following the bankruptcy of Lehman Brothers, massive financial institutions all came close to bankruptcy and had to be rescued.
With the 2008 crisis we lost the "too big to fail" mentality in favour to the idea that even large banks can default. A key concern around the default of a large institution is the systematic risk arising from a cascade of events that could lead to a major crisis. This is strictly connect with the interbank network connection.
In modern financial system the main role of banks is to hold deposit as can be seen in their balance sheet, where deposit are predominant voice. In this situation depositors become unsecured creditors of the banks. Regulators required a liquid capital cushions to the bank in order to avoid unpaid debt in case of default. These requirements lead to a bank network closely linked. To show that we make a simple example of how banks system works. Let's assume that there are bank A, bank B and a borrower. Suppose now that the borrower asks a loan to, let's say bank A. In order to issue the loan, bank A gives part of his cash to the borrower. Suppose now that the borrower deposits the loan to the bank B. Now bank B has more cash than required, while bank A is seeking for cash. Bank B borrows his excess to bank A. In this way the two bank are now linked through the loan. A default of bank A can lead a default or a credit deterioration of bank B. This is a far too simple example of how bank system works, but although its simplicity can

describe the situation very accurately.
In [8] the author showed that the financial network has changed from an enough simple and diversified network to a complex interconnected and no diversified network. This kind of network is robust to typical shocks but very fragile when the shocks hit a dominant entities. Since the crisis a lot a literature is now available on these subjects, with special focus given to the estimation on counterparty risk.

1.1 General Setting

We use a structural approach inspired to [2]. This family of models has indeed much details and can be used in order to better modelling the aspects of corporate defaults rather than reduce-form models. We map the capital structure of a bank in a stochastic process for equity and debt, we then model default as hitting time problem. We must be careful in the choice of the stochastic process: it's well known that pure diffusion models are incapable of a good description of credit events in case of short time-scales.

1.2 Brief description of the work

Starting from [13] we model the asset dynamics using a Lévy jump-diffusion process. We first analyse the simpler case of one dimensional case in order to well understand the model characteristic. Then we analyse the two dimensional case which allow us to introduce interesting feature like mutual liabilities and correlation between assets. Since there is not an available closed formula we focus on the numerical computation of survival probabilities and credit derivative products. Following [14] we develop a finite difference method for the partial integro-differential equation. In order to efficiently compute the integral part we developed a recursive algorithm exploiting some features of the jump process. Finally we propose several methods to improve the convergence and efficiency of the numerical methods.

1.3 Thesis structure

This thesis is organized as follows.
In Chapter 2 we recall some of the basic notions and results of stochastic calculus with jumps.
In Chapter 3 we present the one dimensional case. We introduce the simplified model in case of a single bank. Then, we discuss how to compute the principal credit products in our framework. We present a numerical scheme for solving the PIDE with explicit treatment of jump operator, and finally we conclude with the numerical results.

In Chapter 4 we follow the construction of Chapter 3 to describe the two dimensional case that is a simplification of the model presented in [16]. We identify and model the difference with the one dimensional case. Furthermore we present the general pricing problem through partial integro-differential equation, and similar to the one dimensional case we present how to compute the main credit derivatives products.
In Chapter 5, following [14], we first propose an alternative time discretization that guarantees second order convergence in both space and time, and we prove its stability and consistency. We also propose an averaging method to smooth the discontinuous solution.

Chapter 2

Preliminary Results

In this chapter we briefly describe the mathematical concepts which are necessary for our analysis. We recall the main results of stochastic calculus in the jump diffusion framework following [4]. We introduce a simple jump process and then we consider a more interesting extension of this simple process. Then we introduce the Itô formula in jump diffusion framework.

2.1 Stopping time

One important concept in finance is the notion of stopping time. The stopping time is a specific type of "random time ". Then we can model it as random variable whose value is interpreted as the time at which a given stochastic process exhibits a certain behaviour.

Definition 1. *Let $(\Omega, \mathcal{F}_t, \mathbb{P})$ be a probability space. A random variable $\tau : \Omega \to [0, +\infty]$ is a stopping time if for any t*

$$(\tau < t) \in \mathcal{F}_t.$$

One important feature of stopping time is that the minimum of two stopping time is still a stopping time, i.e.

Proposition 1. *Suppose the filtration $\mathcal{F}_t$ satisfied standard condition and let τ and σ be two stopping times. Then also $\lambda = \min(\tau, \sigma)$ is a stopping time.*

In our setting the stopping time describe the random time in which the asset process crosses the default boundaries. So we are interested in the so called "exit time"

Proposition 2. *Let X_t be a cadlag process such that $X_t = 0$ and $a < 0$. The exit time from the interval $(a, +\infty)$ given by*

$$\tau = \inf\{s \geq t | X_s < a\},$$

is a stopping time.

Empirically this is simple to understand because for any time t it is enough to know the past value of X_t in order to establish whether the boundaries are crossed or not.

2.2 Counting process

The fundamental example of stochastic process with discontinuous trajectories is given by the counting process. The counting process $X_t \in \mathbb{R}^+$ is a stochastic process that counts the number of random times $\{T_n, n \geq 1\}$ occurring until time t. Therefore one possible definition is given by

$$X_t = \sum_{n \geq 1} \mathbf{1}_{\{t \geq T_n\}}.$$

where in order to guarantees that the process X_t is well defined we assume that $\mathbb{P}(T_n \to +\infty) = 1$. The counting process $(X_t)_{t \geq 0}$ is a cadlag process because at each time T_n the paths of the process experience a jump of size $+1$, as shown in Figure 2.1.

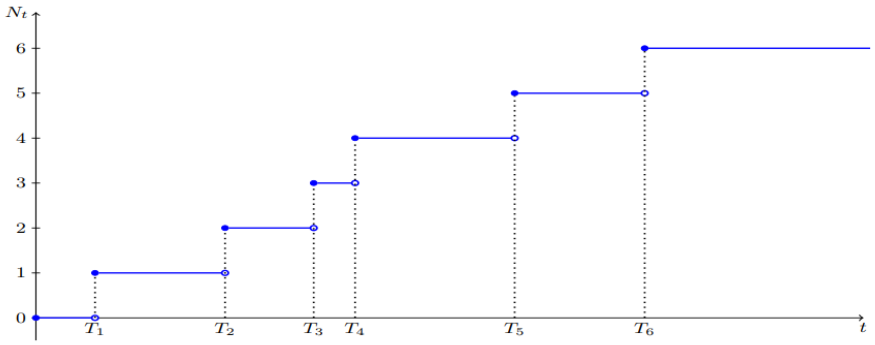

Figure 2.1: Sample of a counting process

Poisson process as counting process

Definition 2 ([4] Definition 2.17)**.** *Let $(\tau_i)_{i \geq 1}$ be a sequence of independent exponential random variables with parameter λ and $T_n = \sum_{i=1}^{n} \tau_i$. The process $(N_t)_{t \geq 0}$ defined by*

$$N_t = \sum_{n \geq 1} \mathbf{1}_{\{t \geq T_n\}},$$

is called Poisson process with intensity λ.

Therefore the Poisson process $(N_t)_t \in \mathbb{R}^+$ is a counting process with special distribution of random times, i.e. taking the interarrival random times τ_i

distributed as exponential random variables. In our application we consider the random times as times in which jumps occur. Doing so the Poisson process counts the number of jumps until t. The choice of exponential distribution gives to the Poisson process the following important proprieties:
1. Independence of increments: for all $0 \leq t_0 < t_1 < \cdots < t_n$ and $n \geq 1$ the increments

$$N_{t_1} - N_{t_0}, \cdots, N_{t_n} - N_{t_{n-1}},$$

are independent random variables.
2. Stationarity of increments : $N_{t+h} - N_{s+h}$ has the same distribution as $N_t - N_s$ for all $h > 0$ and $0 \leq s \leq t$.
The stationary condition means that the distribution of $N_{t+h} - N_{s+h}$ does not depend of the increment h.
3. For any $t > 0$, N_t follows a Poisson distribution with parameter λt, i.e.

$$\forall n \in \mathbb{N},\ \mathbb{P}(N_t = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}.$$

From the propriety 3 follows at least other two important proprieties. First of all knowing the distribution let us to give and expression of the moments of that distribution. We have that if $N_t \sim Poisson(\lambda t)$ then

$$\mathbb{E}[N_t] = \lambda t, \quad Var[N_t] = \lambda t.$$

Second, the following two propositions hold true.

Proposition 3. *If N_t^1 and N_t^2 are independent Poisson processes with intensity λ_1 and λ_2, then the process $(N_t^1 + N_t^2)_{t\geq 0}$ is a Poisson process with intensity $\lambda_1 + \lambda_2$.*

Proposition 4. *Define the filtration $\mathcal{F}_t$ generated by the Poisson process $(N_t)_{t\geq 0}$, i.e.*

$$\mathcal{F}_t := \sigma(N_s : s \in [0, t]).$$

The compensated Poisson process

$$(N_t - \lambda t)_{t\geq 0},$$

is a martingale with respect to $(\mathcal{F}_t)_{t\geq 0}$.

2.3 Compound Poisson Processes

The Poisson process appears to be too limited to developed realistic price model as its jumps are of constant size. Therefore we add a random variable describing the jump size. Let $(Z_k)_{k\geq 1}$ be an independent and identically distributed (i.i.d) sequence of square integrable random variables distributed with probability function ν.

Definition 3. *The process*

$$Y_t = \sum_{k=1}^{N_t} Z_k,$$

is called compound Poisson process.

We show a sample path of the compound Poisson process in Figure 2.2.

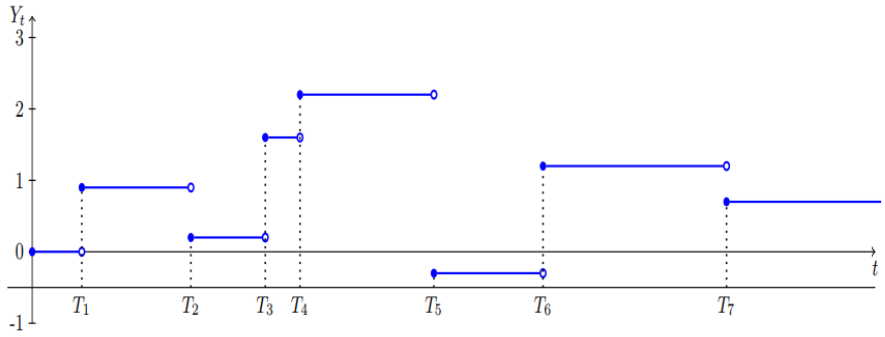

Figure 2.2: Sample of a compound Poisson process

Proposition 5. *For any $t \in [0, T]$ we have*

$$\mathbb{E}[exp(\alpha(Y_T - Y_t)))] = exp(\lambda(T-t)\int_{-\infty}^{+\infty}(e^{\alpha y} - 1)\nu(dy)).$$

From this proposition follows that the compound Poisson process at time t is a random variable with expected value and variance given by

$$\mathbb{E}[Y_t] = \lambda t \mathbb{E}[Z] \qquad Var[Y_t] = \lambda t \mathbb{E}[|Z|^2]$$

Another important result states the following:

Proposition 6. *The compound Poisson process*

$$Y_t = \sum_{k=1}^{N_t} Z_k,$$

has independent increments, i.e. for any finite sequence of times $t_0 < t_1 < \cdots < t_n$ the increments

$$Y_{t_1} - Y_{t_0}, Y_{t_2} - Y_{t_1}, \cdots, Y_{t_n} - Y_{t_{n-1}}$$

are mutually independent variables.

Finally as in the case of Poisson process we have:

Proposition 7. *The compensated compound Poisson process*

$$M_t = Y_t - \lambda t \mathbb{E}[Z],$$

is a martingale.

2.4 Itô formula

Theorem 1 (Itô formula with jumps)**.** *Let X be a jump diffusion process with evolution given by $X_t = X_0 + \int_0^t u_s dW_s + \int_0^t v_s ds + \sum_{i=1}^{N_t} \Delta X_i$, where u_s is the volatility term, v_s is the drift term and ΔX_i corresponds to jump i in the process. Define $Y_t = f(X_t)$ where $f(x)$ is a regular enough function. Then it holds*

$$dY_t = v_t f'(X_t)dt + u_t f'(X_t)dW_t + \frac{1}{2} f''(X_t)u_t^2 dt + f(X_{s^-} + \Delta X_t) - f(X_{s^-}).$$

2.5 Optional stopping theorem

This is one of the most important result presented here since it is a fundamental tool of mathematical finance in the asset pricing theorem.

Theorem 2. *Let $\sigma \leq \tau$ be bounded stopping times. For any cadlag martingale X, the random variables X_σ, X_τ are integrable an the following is satisfied*

$$X_\sigma = \mathbb{E}[X_\tau | \mathcal{F}_\sigma].$$

In our application we use this theorem with simpler setting taking $\sigma = 0$. Since all our process start from the value 0 this theorem stated that the stopped stochastic integral is martingale and therefore the expected value of the integral is 0.

Chapter 3

One Dimensional Model

In this chapter we propose a structural default model following [16]. Structural default model was first introduced by Merton in [20]. It was a simple model with strong and not practical assumptions. Black and Cox later extend it to more general case in [2].
The family of structural default models better described the dynamics of banks assets rather than the reduced models. Indeed structural models map the banks assets in stochastic processes. Therefore these kind of models are rich of information and can be easily manipulate in order to model different custom features like correlation or external influences. In these models default occurs when the banks assets touch a barrier. Starting from [2] the authors of [16] extend the model in case of jump diffusion process. The reason why the model needs to add jump processes derives from the inadequacy of diffusive models in the pricing of credit products for relatively short time-scales. In this chapter we propose the same model of [16], but we limit our analysis to the one dimensional case. In our framework we only have one bank so there are not mutual liabilities.

3.1 Dynamics of asset and liabilities

Assume that the bank has external assets and liabilities, A and L. We assume that the asset's bank dynamic is given by

$$\frac{dA}{A} = (\mu - \kappa\lambda)dt + \sigma dW(t) + (e^J - 1)dN(t), \tag{3.1}$$

where μ is the deterministic growth rate, not necessary risk neutral, σ is the lognormal volatility, $W(t)$ is a Wiener process, N is Poisson process independent of W, λ is the intensity of jump arrivals and J is the random negative exponentially distributed jump sizes with probability density function

$$\tilde{\omega}(s) = \begin{cases} 0, & s > 0, \\ \theta e^{\theta s}, & s \leq 0, \end{cases}$$

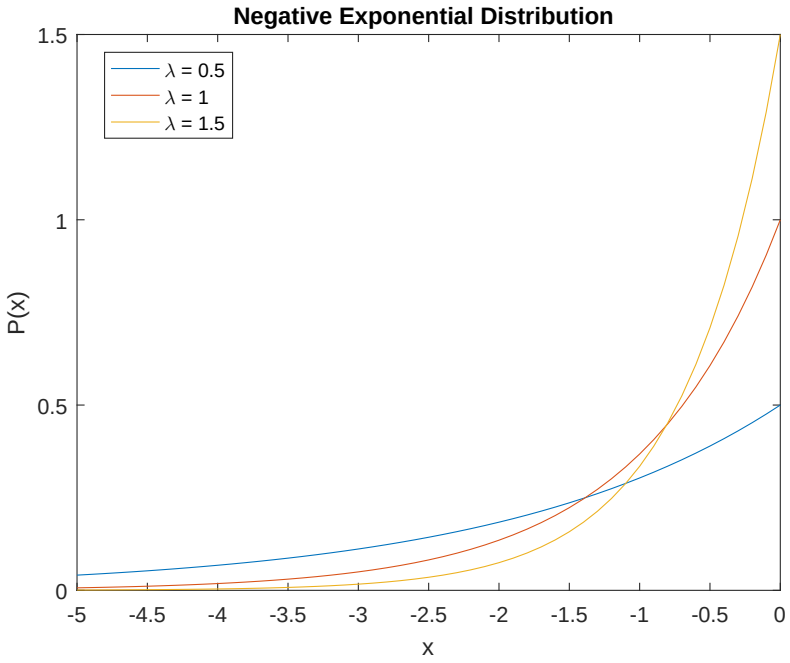

Figure 3.1: Negative exponential distribution for different intensity λ

with parameters $\theta > 0$, and κ is the jump compensator

$$\kappa = \mathbb{E}[e^J - 1] = -\frac{1}{\theta + 1}.$$

An example of negative exponential distribution is given in Figure 3.1. For liabilities we assume deterministic behaviour through the following dynamics

$$\frac{dL}{L} = \mu dt,$$

where μ is the same of (3.1). For pricing purposes, under the risk-neutral measure, we consider μ as a risk-free short rate. For better focus on the impact of jumps we take $\mu = 0$ in the following. The following analysis holds true also for $\mu \neq 0$ and the extension can be obtained easily.

3.2 Default boundaries

Following [16] we consider time dependent default boundaries $\Lambda(t)$. Bank defaults if its asset value process crosses the default boundaries. Then it is natural to introduce a stopping time to describe the time in which the bank defaults.

$$\tau = \inf\{t | A(t) \leq \Lambda(t)\},$$

i.e. τ is the first time in which the bank assets lie below the default boundary $\Lambda(t)$. We assume that before maturity T creditors believe that the bank has a chance to recover from its debt, i.e. creditors believe that although at time t the bank is not able to fully pay his debt, the bank has a chance to recover before maturity. This believe is quantified considering the default boundaries as a fraction R of total debt. Obviously once the bank asset falls below this value the creditors don't believe anymore that the bank could recover from its debt and they consider it defaulted.
At maturity there is no more time to recover and the default boundaries are considered as the full debt. From modelling point of view we assume that R is constant. With this assumption in T default boundaries experience a jump, i.e. we consider that in last day the recovery rate rapidly tends to 1. The default boundaries can therefore be modelled as

$$\Lambda = \begin{cases} RL \equiv \Lambda^{<}, & t < T, \\ L \equiv \Lambda^{=}, & t = T, \end{cases} \tag{3.2}$$

where $0 \leq R \leq 1$ is the constant recovery rate.

3.3 Formulation of backward Kolmogorov equation

For convenience, we introduce lognormal normalized variable

$$X_t = \ln\left(\frac{A_t}{\Lambda^{<}}\right),$$

and denote

$$\xi = -\left(\frac{\sigma^2}{2} + \kappa\lambda\right).$$

Applying Itô's formula to X_t, we find its dynamics (see Appendix B for details)

$$dX_t = \xi dt + \sigma dW(t) + J dN(t).$$

Substituting from (3.2) we obtain the default boundaries for the new variable X_t

$$\mu = \begin{cases} \mu^{<} = 0, & t < T, \\ \mu^{=} = \ln\left(\frac{\Lambda^{=}(t)}{\Lambda^{<}(t)}\right), & t = T. \end{cases}$$

With this change of variable we restrict the problem domain to the first quadrant. Given the lognormal asset dynamics we are able to price credit products. From non arbitrage theory we have that the price $V(x,t)$ of contract is given by the expected value under risk neutral measure of the future cash flows. Consider a contract with terminal payoff $\psi(X_T)$, payment $\chi(s,x)$ at intermediate time $t \leq s \leq T$(for example, coupon payment), and

payoff in case of intermediate default of the bank $\phi_0(t)$. Then, the value function is given by

$$V(x,t) = \mathbb{E}\Big[\psi(X_T)\cdot \mathbf{1}_{\{\tau \geq T\}} - \int_t^T \chi(s,X_s)\cdot \mathbf{1}_{\{\tau > s\}} ds + \phi_0(\tau)\cdot \mathbf{1}_{\{\tau < T\}} | X(t) = x\Big],$$

According to the Feynman-Kac formula, the corresponding pricing equation is the Kolmogorov backward equation:

$$\frac{\partial V}{\partial t} + \mathcal{L}V = \chi(t,x), \tag{3.3}$$

$$V(t,0) = \phi_0(t), \qquad V(t,x_1) \xrightarrow[x_1 \to +\infty]{} \phi_\infty(t), \tag{3.4}$$

$$V(T,x) = \psi(x), \tag{3.5}$$

where

$$\mathcal{L}f = \frac{\sigma^2}{2} f_{xx} + \xi f_x + \lambda \mathcal{J}f - \lambda f = \mathcal{D}f + \lambda \mathcal{J}f - \lambda f, \tag{3.6}$$

$$\mathcal{J}f(x) = \theta \int_0^x f(x-u) e^{-\theta u} du, \tag{3.7}$$

and ϕ_0 is a given function depending on the kind of contract. The derivation of the jump operator is considered in Appendix A.

3.4 Pricing Problem

Through the equations (3.3)-(3.5) we rewrite a probabilistic problem as a deterministic partial-integro differential equation. In this section we formulate the Kolmogorov backward equation for specific quantities.

3.4.1 Survival Probability

We formulate the problem with general domain because it will be useful in the two dimensional case. Let $\mu^<$ be the default boundaries, then the PIDE domain is given by $D = [\mu^<, \infty] \times [t,T]$ and the default time is given by $\tau = \inf\{t | X(t) \leq \mu^<\}$. Defining $\vartheta(t,x)$ the survival probability of the bank, it's easy to see that $\vartheta(t,x) = \mathbb{E}\big[\mathbf{1}_{\{\tau \geq T, X(T) > \mu^=\}} \big| X(t) = x\big]$. It's important to notice that

- $\mathbf{1}_{\{\tau \geq T\}}$ controls that the bank doesn't default before maturity,
- $\mathbf{1}_{\{X(T) > \mu^=\}}$ controls that the bank does not default at maturity T, where the boundary changes from $\mu^<$ to $\mu^=$,

From equations (3.3)-(3.5), taking $\psi(x) = \mathbf{1}_{\{x > \mu^=\}}$ and $\chi(x,t) = 0$, we obtain

$$\frac{\partial}{\partial t}\vartheta(t,x) + \mathcal{L}\vartheta(t,x) = 0,$$

with $\mathcal{L}f$ defined in (3.6).
The final condition and the boundary condition at $x = \mu^<$ become

$$\begin{aligned}
&\vartheta(T, x) = \mathbf{1}_{\{x \geq \mu^=\}}, \\
&\vartheta(t, \mu^<) = 0.
\end{aligned}$$

Since we are dealing with a semi infinite domain, the problem does not define any border condition. In order to solve the PIDE we must give an artificial border condition. Since for large value of the bank asset x the probability of default vanishes, it seems natural to impose that at $x \to +\infty$ the solution tends to 1. Then the problem becomes

$$\begin{aligned}
&\frac{\partial}{\partial t}\vartheta(t, x) + \mathcal{L}\vartheta(t, x) = 0, \\
&\vartheta(t, \mu^<) = 0, \qquad \vartheta(t, x) \xrightarrow[x \to +\infty]{} 1, \\
&\vartheta(t, x) = \mathbf{1}_{\{x > \mu^<\}}.
\end{aligned}$$

3.4.2 Credit Default Swap

A credit default swap is a contract designed to eliminate credit risk over a Reference Name (RN). Protection Buyer (PB) buys a CDS over RN to a Protection Seller (PS). PB pays periodic coupon payments c to PS until default of RN or maturity T. In exchange PS pays to PB the loss given RN's default. Consider $C(t, x)$ the value of a CDS written on the bank (RN) with coupon c. Assume that the coupon is paid continuously. According to [1] the value becomes

$$C(t, x) = \mathbb{E}\left[(1-R)\mathbf{1}_{\{\mu^< \leq X(T) \leq \mu^=, \tau \geq T\}} - \int_t^T c\mathbf{1}_{\{\tau > s\}} ds + (1-R)\mathbf{1}_{\{\tau < T\}} \middle| X(t) = x\right].$$

At maturity T the default boundaries moves from $\mu^<$ to $\mu^=$, so the CDS value is 0 if the bank survives and $1 - R$ otherwise because, in this case, the RN defaults and the loss given default is paid.
The integral term accounts for the payments of coupon c until maturity T or default.
Finally if the bank defaults before maturity, i.e. $\tau < T$ the value of the CDS becomes $1-R$. The border condition when $x \to +\infty$ is obtained considering the CDS as a risk-free annuity. Then the problem can be formulate as

$$\begin{aligned}
&\frac{\partial}{\partial t}C(t, x) + \mathcal{L}C(t, x) = c, \\
&C(t, \mu^<) = 1 - R, \qquad C(t, x) \xrightarrow[x \to +\infty]{} -c(T - t), \\
&C(T, x) = (1 - R)\mathbf{1}_{\{\mu^< \leq x \leq \mu^=\}}.
\end{aligned}$$

3.5 Numerical scheme

We formulate all the pricing problem as the solution of a PIDE. Unfortunately, unlike in the case without jumps, no analytical solution is available so far. Finite difference methods have firmly established themselves as the most versatile and powerful methods for solving such problems. However the finite difference schemes suffer the dimensionality of the problem, and their efficiency rapidly deteriorate. Since we are dealing with only one dimension in this Section we propose a way to solve (3.3)-(3.5) using finite difference scheme. First of all the PIDE domain needs to be approximate by a finite grid so that the solution can be approximate in each node of the grid. For simplicity we use the same grid for both jump and differential operator.

3.5.1 Grid discretization

It's well known that the choice of the grid in the finite difference scheme takes a central role in the accuracy of the solution. It's also be shown that non uniform grid can lead to better approximations rather than uniform grid. Indeed this kind of grid allows to have more points lying near the critical zones, and less points in area with minor financial interest. There are two main methods to create a non uniform grid:

- Refine an uniform grid,
- Redistribution of uniform grid.

We choice the latter one since it keeps constant the number of nodes and so it's easy to analyse the algorithm's complexity since the number of points doesn't change. In credit risk framework the relevant areas are the one close to the default boundary μ where the credit events may happen. For the construction of non uniform grid through grid transformation we follow [11] and [23].

The grid

From equation (3.3) follows that the computational domain $D = [0, \infty)$ is semi infinite, so it's approximated by $\tilde{D} = [0, x_{max}]$. Let $N + 1$ be the number of points used to approximate the domain. Then the grid G is given by

$$G = \{0 = x^0 < x^1 < \cdots < x^N = x_{max}\},$$

where the values of x_i are given transforming an uniform grid ξ through S, i.e.

$$x_i = S(\xi), \qquad i = 0, \ldots, N.$$

Transformation

Given a uniformly distributed grid $\xi \in [0,1]$ we must find the right grid transformation $S = S(\xi)$, indeed wrong transformation can add an error as show in [23]. A lot of transformations are available. In the following we present a simple transformation based on the studies of the Jacobian.

Single Critical Point

In case of single barrier we have explicit formula for the transformation. Given a transformation $S = S(\xi)$ we know that the variation of the grid is given by $\Delta S(\xi) = J(\xi)\Delta\xi$, where J is the Jacobian of the transformation. A good transformation can be obtained with $J(\xi) = (\alpha^2 + (S(\xi) - B)^2)^{\frac{1}{2}}$, where α is a custom parameter and B is the barrier. In this way if $|S - B| >> \alpha$ the grid step is proportional to S, then grid is almost exponential , while if $|S - B| << \alpha$ the grid step is proportional to α, then the grid is almost uniform. With this choice the problem reduces to solve an ODE

$$\frac{dS(\xi)}{d\xi} = (\alpha^2 + (S(\xi) - B)^2)^{\frac{1}{2}}.$$

It's simple to verify that the solution of the ODE is

$$S(\xi) = B + \alpha \sinh(c_2\xi + c_1(1 - \xi)).$$

The constant c_1 and c_2 can be found imposing that $S(0) = S_{min}$ and $S(1) = S_{max}$. Thus:

- $c_1 = \frac{sinh^{-1}(S_{min} - B)}{\alpha}$,
- $c_2 = \frac{sinh^{-1}(S_{max} - B)}{\alpha}$.

Such grid maps the interval $0 \leq \xi \leq 1$ into the interval $S_{min} \leq S \leq S_{max}$. It's also possible to compute the value ξ_B relative to the barrier B. This value is given by $\xi_B = \frac{-c_1}{c_2 - c_1}$. The parameter α controls levels of uniformity of the grid. Taking $\alpha \ll S_{max} - S_{min}$ yields very non uniform grid, while $\alpha \gg S_{max} - S_{min}$ yields almost uniform grid.

3.5.2 Differential Operator

Since we use a non uniform grid, we don't perform the standard finite difference discretization but we follow [15] in order to obtain the best discretization possible. For completeness we present backward, central and forward schemes:

$$\frac{\partial V}{\partial x}(x^i) \approx \alpha_{i,-2}V(x^{i-2}) + \alpha_{i,-1}V(x^{i-1}) + \alpha_{i,0}V(x^i), \tag{3.8}$$

$$\frac{\partial V}{\partial x}(x^i) \approx \beta_{i,-1}V(x^{i-1}) + \beta_{i,0}V(x^i) + \beta_{i,1}V(x^{i+1}), \tag{3.9}$$

$$\frac{\partial V}{\partial x}(x^i) \approx \gamma_{i,0}V(x^i) + \gamma_{i,1}V(x^{i+1}) + \gamma_{i,2}V(x^{i+2}), \tag{3.10}$$

with coefficients

$$\alpha_{i,-2} = \frac{\Delta x^i}{\Delta x^{i-1}(\Delta x^{i-1} + \Delta x^i)}, \quad \alpha_{i,-1} = \frac{-\Delta x^{i-1} - \Delta x^i}{\Delta x^i \Delta x^{i-1}}, \quad \alpha_{i,0} = \frac{\Delta x^{i-1} + 2\Delta x^i}{\Delta x^i(\Delta x^i + \Delta x^{i-1})},$$
$$\beta_{i,-1} = \frac{-\Delta x^{i+1}}{\Delta x^i(\Delta x^i + \Delta x^{i+1})}, \quad \beta_{i,0} = \frac{\Delta x^{i+1} - \Delta x^i}{\Delta x^i \Delta x^{i+1}}, \quad \beta_{i,1} = \frac{\Delta x^i}{\Delta x^{i+1}(\Delta x^i + \Delta x^{i+1})},$$
$$\gamma_{i,0} = \frac{\Delta x^{i+2} - 2\Delta x^{i+1}}{\Delta x^{i+1}(\Delta x^{i+1} + \Delta x^{i+2})}, \quad \gamma_{i,1} = \frac{-\Delta x^{i+1} - \Delta x^{i+2}}{\Delta x^{i+1} \Delta x^{i+2}}, \quad \gamma_{i,2} = \frac{-\Delta x^{i+1}}{\Delta x^{i+1}(\Delta x^{i+1} + \Delta x^{i+2})}.$$

where $\Delta x^i = x^i - x^{i-1}$.
To approximate the second derivative we use the central scheme

$$\frac{\partial^2 V}{\partial x^2}(x^i) \approx \delta_{i,-1} V(x^{i-1}) + \delta_{i,0} V(x^i) + \delta_{i,1} V(x^{i+1}), \tag{3.11}$$

with coefficients

$$\delta_{i,-1} = \frac{2}{\Delta x^i(\Delta x^i + \Delta x^{i+1})}, \quad \delta_{i,0} = \frac{-2}{\Delta x^i \Delta x^{i+1}}, \quad \delta_{i,1} = \frac{2}{\Delta x^{i+1}(\Delta x^i + \Delta x^{i+1})}.$$

Equations (3.8)-(3.10) and (3.11) are classical approximations of derivative in case of non uniform mesh. In the case of uniform mesh they become standard finite difference scheme. The scheme in (3.8)-(3.10) and (3.11) has a second order truncation error for meshes that are either uniform or a smooth transformation of such meshes. With the above scheme we can approximate the diffusion operator $\mathcal{D}$ using the matrix $D \in \mathbb{R}^{(N+1)\times(N+1)}$. Thus, D is a tridiagonal matrix given by

$$D = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ \Delta_{1,-1} & \Delta_{1,0} & \Delta_{1,1} & 0 & \dots & 0 \\ 0 & \Delta_{2,-1} & \Delta_{2,0} & \Delta_{2,1} & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \Delta^1_{N-1,-1} & \Delta_{N-1,0} & \Delta_{N-1,1} \\ 0 & \dots & 0 & 0 & 0 & 0 \end{pmatrix}$$

where :

- $\Delta_{i,j} = \xi\beta_{i,j} + \frac{\sigma^2}{2}\delta_{i,j}$.

3.5.3 Jump Operator

We developed two different discretizations of the jump operator. In the first discretization we approach the integral directly.

First method

Introducing auxiliary function $\varphi_l(x) = \mathcal{J}V(t_l, x)$ and recalling the definition of the jump operator, we have

$$\varphi_l(x) = \theta \int_{-x}^{0} V(t, x+j)e^{\theta j} dj,$$
$$\varphi_l(x+h) = \theta \int_{-x-h}^{0} V(t, x+j+h)e^{\theta j} dj, \tag{3.12}$$

by changing variable in equation (3.12), with $z = h + j$, we get

$$\varphi_l(x+h) = \theta e^{-\theta h}\bigg(\int_{-x}^{0} V(t_l, x+z)e^{\theta z} dz + \int_{0}^{h} V(t_l, x+z)e^{\theta z} dz \bigg).$$

The first term is by definition $\varphi_l(x)$, the second term can be solved using first a Taylor expansion to the second order and then approximating the derivative term with central difference. Finally we obtain

$$\begin{aligned} \varphi_l(x+h) &= e^{-\theta h}\varphi_l(x) + \omega_0(\theta, h)V(t_l, x) + \omega_1(\theta, h)V(t_l, x+h) + \mathcal{O}(h^3), \\ \varphi_l(0) &= 0, \end{aligned}$$

where

$$\omega_0(\theta, h) = \frac{1 - (1+\theta h)e^{-\theta h}}{\theta h}, \quad \omega_1(\theta, h) = \frac{-1 + \theta h + e^{-\theta h}}{\theta h}. \tag{3.13}$$

Then rewriting the recursive formula for the jump operator we get

$$\mathcal{J}V(t, x+h) = e^{-\theta h}\mathcal{J}V(t, x) + \omega_0(\theta, h)V(t, x) + \omega_1(\theta, h)V(t, x+h) + \mathcal{O}(h^3).$$

where the coefficient ω_0, ω_1 are defined in (3.13).
Given the grid in Section 3.5.1 we denote with J^i the approximation of $\mathcal{J}V(t, x^i)$ on the grid. Then we have :

$$\begin{aligned} J^{i+1} &= e^{-\theta h^{i+1}} J^i + \omega_0(\theta, h_{i+1})V(t, x^i) + \omega_1(\theta, h_{i+1})V(t, x^{i+1}), \\ J^0 &= 0, \end{aligned} \tag{3.14}$$

where increments are given by $h_{i+1} = x^{i+1} - x^i$.

Second method

For an alternative discretization we rewrite the jump operator in (3.7) as differential equation

$$\frac{\partial}{\partial x}\bigg(\mathcal{J}V(x)e^{\theta x} \bigg) = \theta V(x)e^{\theta x}.$$

To solve these differential equation we apply Adam-Moulton method of second order which gives us third order of accuracy locally as shown in [3]. Then

$$J^{i+1} = e^{-\theta h_{i+1}} J^i + \frac{1}{2}\theta h_{i+1} e^{-\theta h_{i+1}} V(x^i) + \frac{1}{2}\theta h_{i+1} V(x^{i+1}).$$

We obtain a scheme that is equivalent to the trapezoidal rule. In order to have coherent notation by defining

$$\omega_0(\theta, h) = \frac{1}{2}\theta h e^{-\theta h}, \quad \omega_1(\theta, h) = \frac{1}{2}\theta h.$$

we obtain the same formula of (3.14). Both approximation give the same results with the same efficiency as shown in Figures 3.2-3.3. In Figure 3.4 we plot the difference of the jump operator in the two methods. The difference is coherent with the behaviour of the ω_0 and ω_1 in function of the increment h shown in Figures 3.5-3.6. With this explicit recursive formula the jump operator $\mathcal{J}V$ can be computed on each node of the grid with a complexity of $\mathcal{O}(N)$. From a computation perspective in the case of $N = 100$ the recursive formula allows to compute the J contribution in 0.041 seconds respect to the 0.617 seconds needed using numerical integration. Notice that the efficiency of the algorithm lies on the kind of jump size distribution. Only for exponential jump size the jump operator can be expressed in such a way.

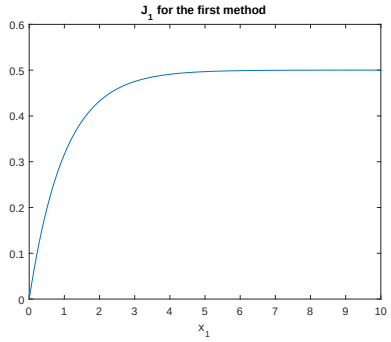

Figure 3.2: Jump contribution using first method using $\mathbf{1}$ *test vector*

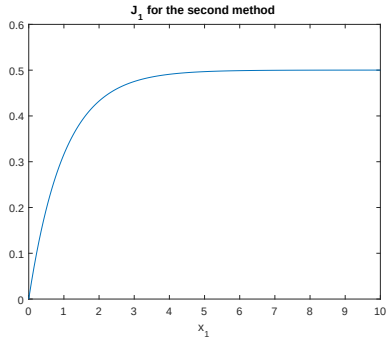

Figure 3.3: Jump contribution using second method using $\mathbf{1}$ *test vector*

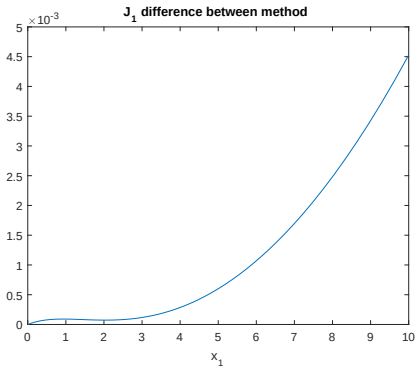

Figure 3.4: Difference of jump contribution between method

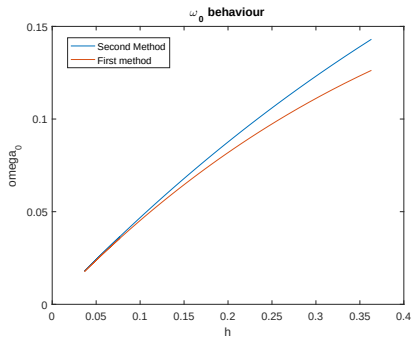

Figure 3.5: Behaviour of coefficient ω_0 *in function of the increments*

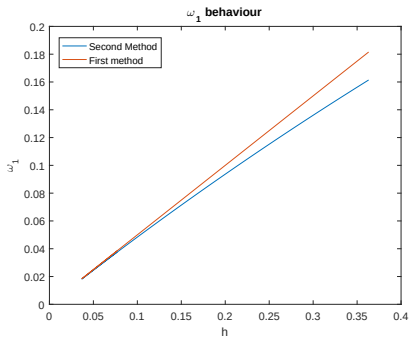

Figure 3.6: Behaviour of coefficient ω_1 *in function of the increments*

3.5.4 Time discretization

We make a change of time variable in order to deal with forward-in-time equation. Define $\tau = T - t$, the problem becomes

$$\frac{\partial V}{\partial \tau} = \mathcal{L}V - \chi(\tau, x), \tag{3.15}$$

$$V(\tau, 0) = \phi_{1,0}(\tau), \qquad V(\tau, x_1) \xrightarrow[x_1 \to +\infty]{} \phi_{1,\infty}(\tau), \tag{3.16}$$

$$V(0, x) = \psi(x). \tag{3.17}$$

With the result in Section 3.5.2 and 3.5.3, we can approximated the operator $\mathcal{L}V$ in each point of the grid using the matrix $LV = DV + \lambda JV - \lambda V$. After the discretization the PIDE (3.15) reduces to solve a system of ordinary linear differential equations. In order to solve the system of ODE we introduce the vector $U(t) \in \mathbb{R}^{N+1\times 1}$ whose i-th component represents the solution $V(t, x_i)$. Then, we get

$$\begin{aligned} U'(t) &= AU(t) + \lambda J(U^n) + b(t), \\ U(0) &= U_0. \end{aligned}$$

where $A = D - \lambda I$, and $b(t)$ takes into account for both boundary conditions and non homogeneous term of the PIDE. To solve this problem we can use well known Crank-Nicolson scheme with $\theta = \frac{1}{2}$ which, under mild assumption, guarantees second order convergence in time. Treating the jump part explicitly we get

$$\frac{U^{n+1} - U^n}{\Delta t} = \frac{1}{2}\left[AU^{n+1} + AU^n\right] + \lambda J(U^n).$$

Solving respect to U^{n+1} we get the linear system

$$\left(I - \frac{1}{2}\Delta t A\right)U^{n+1} = \left(I + \frac{1}{2}\Delta t A\right)U^n + \Delta t \lambda J(U^n) + b(t_{n+1}).$$

We then correct the first and last row of A, the first and last element of both U^n and $J(U^n)$ to zero in order to strongly impose the border condition. The matrix A is tridiagonal so there are a lot of efficiently algorithms to solve the linear system.

3.6 Results

In this section we analyse the model characteristic computing different credit products. We consider the parameters in Table 3.2 for the model without jump.

$L_{1,0}$	$L_{2,0}$	R_1	R_2	T	σ_1	σ_2
60	70	0.4	0.45	1	0.4	0.3

Table 3.1: Model parameters.

In case of model with jump we consider the parameters in both Tables 3.2-3.1.

θ_1	θ_2	λ_1	λ_2
1	1	0.095	0.055

Table 3.2: Jump parameters.

All test are made using a grid with 100 nodes and using 100 time steps. The grid is built concentrating points in $\tilde{\mu}$ and taking parameter α proportional to $x^{max} - \tilde{\mu}$ by a factor of 1/10. From a computation perspective, using such grid the computation time of both model (with and without jumps) is of the order of 0.06 seconds for each payoff.

Survival Probability

We consider the survival probability of the first bank. In Figures 3.8-3.7 we plot the survival probability in function of both time and bank asset for model without and with jump respectively. In these plots we consider the default boundaries at $\tilde{\mu}^{<}$. As expected we find a decreasing function of time. In Figure 3.9 we plot the difference between model with and without jump and we notice that the jumps mostly impact the area near default boundaries. Here the probability is quite lower than the case without jump. This is what we expected since our model considers only negative jumps.

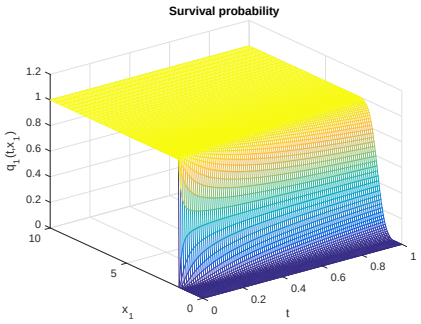

Figure 3.7: Survival probability for model without jump

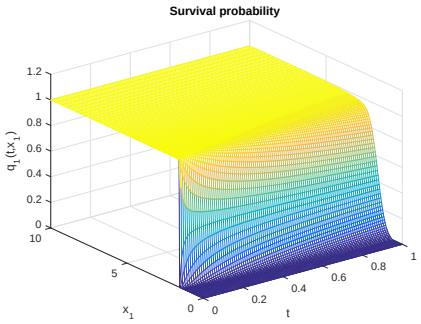

Figure 3.8: Survival probability for model with jump

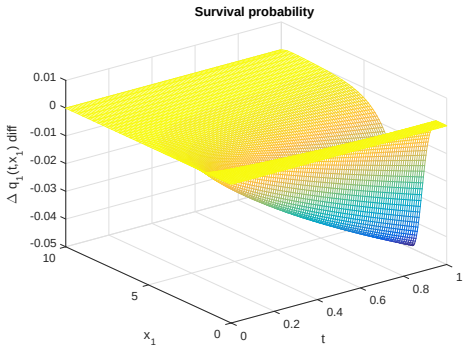

Figure 3.9: Difference of survival probability between model with and without jump

Credit default swap

We consider the value of a credit default swap on the first bank (RN). In Figures 3.10-3.11 we plot the credit default swap price in function of time and bank's assets for both models. In Figure 3.12 we plot the difference between this models. We notice that again the jump strongly impacts the area near default boundaries. This time the difference is positive because when considering jumps the survival probability of the bank decreases, then the value of a CDS can only increase.

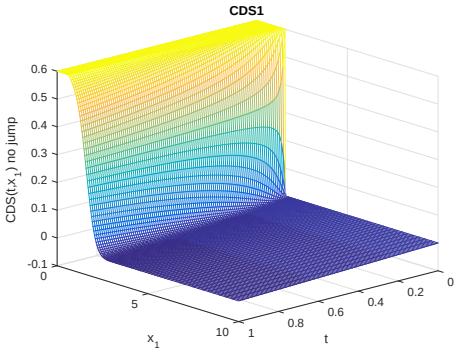

Figure 3.10: Credit default swap spread for model without jump

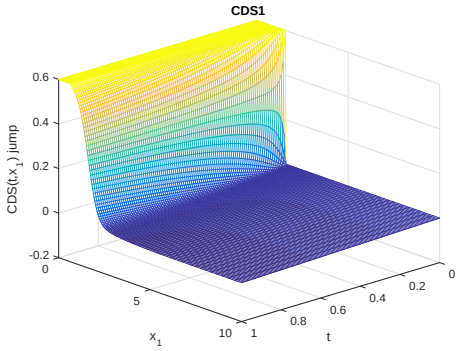

Figure 3.11: Credit default swap spread for model with jump

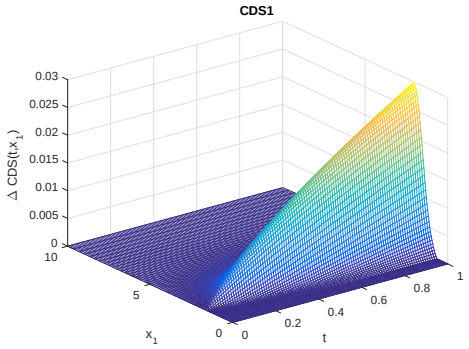

Figure 3.12: Difference of Credit default swap spread between model with and without jump

As mentioned in Section 5.2 Crank-Nicholson method has first order convergence in case of discontinuous payoffs. Following the smoothing approach of Section 5.2 we are able to restore second order convergence even with discontinuous payoff.

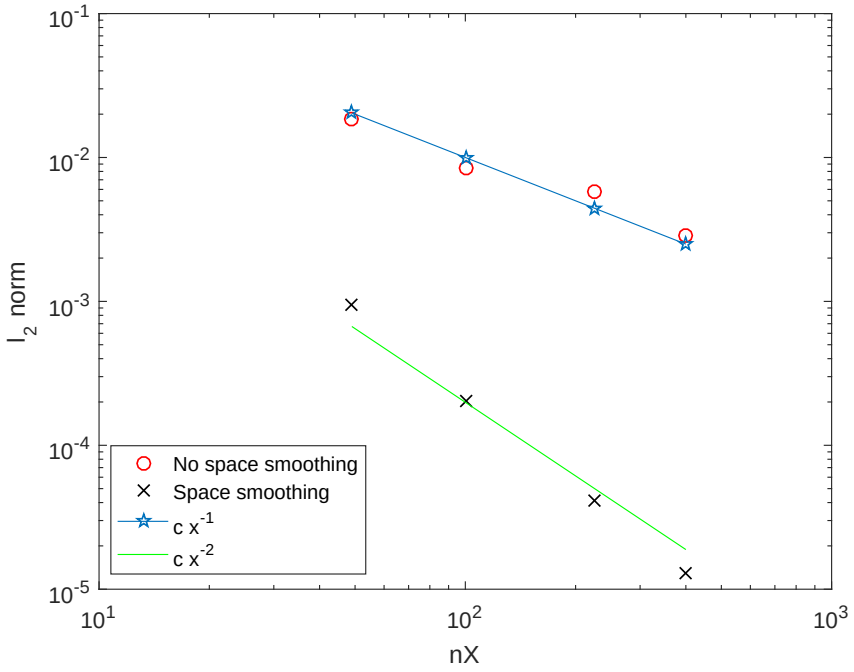

Figure 3.13: Rate of convergence

The observed convergence is shown in Figure 3.13. As expected, the convergence is of order one without smoothing and of order two in case of smoothing.

Chapter 4

Two Dimensional Model

In this chapter we follow the structure of the one dimensional model with some important different features. The main important feature is the presence of mutual liabilities. This is a key point in order to well describe the banking network as a whole. In such a way adverse shock movements to a bank can be rapidly transmitted to the whole banking system in case of large mutual liabilities. In the presence of large mutual liabilities, a bank which, if considered singularly appears to be stable and apparently without imminent default risk, may actually shows itself in a bad shape if considered in a system of banks.
In 2014, the authors of [5] show that the interbank loans in the Europe area was the 12% of the total liabilities. So the mutual liabilities have a significant contribution in the behaviour of a bank. Therefore, following [16] we assume that the banks have external assets and liabilities, A_i and L_i respectively, for $i = 1, 2$. In order to model interlinked bank's network we introduce mutual liabilities L_{12} and L_{21} between each bank, where L_{ij} is the amount that the i-th bank owes to the j-th bank. So we can write the total asset and liabilities as

$$\tilde{A}_1 = A_1 + L_{21}, \qquad \tilde{L}_1 = L_1 + L_{12},$$
$$\tilde{A}_2 = A_2 + L_{12}, \qquad \tilde{L}_2 = L_2 + L_{21}.$$

4.1 Dynamics of asset and liabilities

As in Chapter 3 we assume that the assets banks dynamics are given by

$$\frac{dA_i}{A_i} = (\mu_i - \kappa_i\lambda_i)dt + \sigma_i dW_i(t) + (e^{J_i} - 1)dN_i(t), \quad i = 1, 2, \tag{4.1}$$

where μ_i, σ_i, λ_i, κ_i and $\tilde{\omega}_i(s)$ are equal to the parameters of one dimensional dynamics. The jump size are therefore distributed with probability

distribution given by

$$\tilde{\omega}_i(s) = \begin{cases} 0, & s > 0, \\ \theta_i e^{\theta_i s}, & s \leq 0, \end{cases} \tag{4.2}$$

with parameters $\theta_i > 0$, and κ_i are jump compensator

$$\kappa_i = \mathbb{E}[e^{J_i} - 1] = -\frac{1}{\theta_i + 1}.$$

To deal with interbank network risk we need to model the correlated dynamics of banks. To do so we take W_i ρ-correlated Wiener process, i.e.

$$dW_1(t)dW_2(t) = \rho dt,$$

We also take correlated jump processes. Correlation of jump processes is a way to model the systematic source of jumps. Indeed dealing with jump terms positive correlated guarantees that if one asset jumps, also the other asset jump with probability proportional to the correlation. The correlation is make as in [19].
Consider independent Poisson processes $N_{\{1\}}(t)$, $N_{\{2\}}(t)$ and $N_{\{12\}}(t)$ with the corresponding intensities $\lambda_{\{1\}}$, $\lambda_{\{2\}}$, $\lambda_{\{12\}}$. The correlated Poisson processes $N_1(t)$ and $N_2(t)$ are obtain as

$$\begin{aligned} N_i(t) &= N_{\{i\}}(t) + N_{\{12\}}(t), \quad i = 1, 2, \\ \lambda_i &= \lambda_{\{i\}} + \lambda_{\{12\}}. \end{aligned}$$

i.e., we consider both systematic and idiosyncratic sources of jumps. The Poisson process $N_i(t)$ is capable of producing independent jumps through $\lambda_{\{i\}}$, and simultaneous jump through $\lambda_{\{12\}}$. This is not the only way to correlated jump processes. The advantage of this method is that the marginal distributions do not change. It's well known fact that the summation of independent Poisson distributions of intensity λ_i is a Poisson distribution with intensity $\lambda = \sum_i \lambda_i$.
For liabilities we assume deterministic behaviour through the following dynamics

$$\frac{dL_i}{L_i} = \mu_i dt, \qquad \frac{dL_{ij}}{L_{ij}} = \mu_i dt,$$

where μ_i is the same of Equation (4.1). For pricing purposes, under the risk-neutral measure, we consider μ_i as a risk-free short rate. For simplicity we take $\mu_i = 0$ for $i = 1, 2$ in the following.

4.2 Default boundaries

Following [16] we consider time dependent default boundaries $\Lambda_i(t)$. We assume that the default event of bank i happens when its asset value process

crosses the default boundaries. We assume that the bank assets are monitored at all times $0 < t \leq T$, and that the default event can occur at any time between the current time t and the maturity T. Then it is natural to introduce a stopping time to describe the time in which bank i-th defaults. Since we are dealing with non continuous processes we cannot model the default time as a hitting time, but we model it as the time of crossing of a barrier, i.e.,

$$\tau_i = \inf\{t | A_i(t) \leq \Lambda_i(t)\}, \qquad i = 1, 2,$$

Therefore τ_i is the time in which bank i defaults. Then we define $\tau = \min(\tau_1, \tau_2)$ the time of first default.
Considering a system of several banks, in this case 2, the default boundaries are no longer constant but depend on whether the other bank defaults or not. Before any of the banks $i = 1, 2$ default, i.e $t < \tau$ both banks are able to pay their debts. The default boundaries are

$$\Lambda_i = \begin{cases} R_i(L_i + L_{i\bar{i}}) - L_{\bar{i}i} \equiv \Lambda_i^{<}, & t < T, \\ L_i + L_{i\bar{i}} - L_{\bar{i}i} \equiv \Lambda_i^{=}, & t = T, \end{cases} \tag{4.3}$$

where $0 \leq R_i \leq 1$ is the recovery rate and $\bar{i} = 3 - i$. Before maturity creditors believe that bank i is still able to recover its debt, therefore they consider the default boundaries as a fraction R_i of total liabilities $(L_i + L_{i\bar{i}})$. Bank i also cashes in the liabilities of the other bank $L_{\bar{i}i}$. Instead at maturity T there is no more time to recover part of liabilities, so the debts must be fully paid.
If the i-th bank defaults at intermediate time t, then for the surviving bank $\bar{i} = 3 - i$ the default boundary changes to $\Lambda_{\bar{i}}(t_+) = \tilde{\Lambda}_{\bar{i}(t)}$, where

$$\tilde{\Lambda}_{\bar{i}} = \begin{cases} R_{\bar{i}}(L_{\bar{i}} + L_{\bar{i}i} - R_i L_{i\bar{i}}) \equiv \tilde{\Lambda}_{\bar{i}}^{<}, & t < T, \\ L_{\bar{i}} + L_{\bar{i}i} - R_i L_{i\bar{i}} \equiv \tilde{\Lambda}_{\bar{i}}^{=}, & t = T. \end{cases}$$

Again if, for example, bank i defaults before maturity, the bank $\bar{i}$ cashes in only a fraction of the debt that bank i owns to bank $\bar{i}$, i.e $R_i L_{\bar{i}i}$, that is considered like a "liabilities ". It is clear that for $\Delta\Lambda_{\bar{i}}(t) = \Lambda_{\bar{i}}(t+) - \Lambda_{\bar{i}}(t)$ we have

$$\Delta\Lambda_{\bar{i}} = \tilde{\Lambda}_{\bar{i}} - \Lambda_{\bar{i}} = \begin{cases} (1 - R_{\bar{i}} R_i) L_{i\bar{i}}, & t < T, \\ (1 - R_i) L_{i\bar{i}}, & t = T. \end{cases}$$

Thus, $\Delta\Lambda_{\bar{k}} > 0$ and the corresponding default boundaries move to the right. This is an important feature of the model. We empathize this with an example. If the bank i defualts in t the default boundaries of bank $\bar{i}$ moves to the right. In doing so, it may happen that in t the bank $\bar{i}$ is able to repay

its debts but in t_+, with the new boundaries, defaults. This mechanism can therefore trigger cascades of defaults as shown in the Figure 4.1.

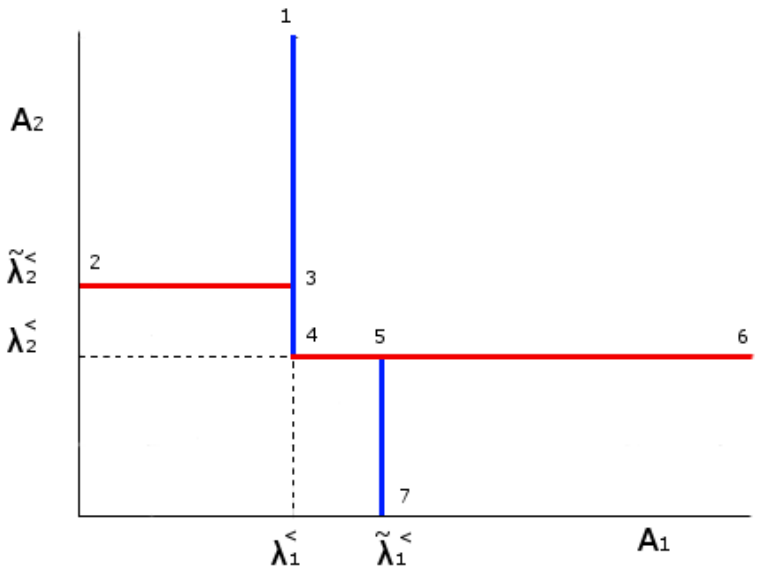

Figure 4.1: Default boundaries for $t < T$.

If there are no defaults the domain is given by the area which lies above the line 1-4-6. If bank 1 defaults the domain becomes the area above the line 2-3-4-6. Finally if bank 2 defaults the domain is given by the area to the right of line 1-4-5-7.

4.3 Terminal condition

We need to specify the settlement process at time $t = T$. We shall do this in the spirit of [6]. As said before at time T banks don't have time to recover. Therefore they can pay to their obligors the current amount of money in hand. When bank i defaults in T, the total bank asset is a fraction ω_i of its total liabilities. Then, trivially, $\omega_i = 1$ when bank i pays all liabilities (both external and interbank) and survives. On the other hand, if $0 < \omega_i < 1$, bank i defaults, and pays only a fraction of its liabilities. Thus, we can describe the terminal condition as a system of equations

$$\min\{A_i(T) + \omega_{\bar{i}} L_{\bar{i}i}, L_i + L_{i\bar{i}}\} = \omega_i (L_i + L_{i\bar{i}}). \tag{4.4}$$

There is a unique vector $\omega = (\omega_1, \omega_2)^T$ such that the condition (4.4) is satisfied. See [6] for details. Equation (4.4) well described the linked bank's net-

work. In order to see it we manipulate the equation. Using the non dimensional variables $\tilde{L}_i(T) = (L_i + L_{i\bar{i}})$, $a_i = A_i(T)/\tilde{L}_i(T)$ and $l_{\bar{i}i} = L_{\bar{i}i}/\tilde{L}_i(T)$, the problem in (4.4) becomes

$$\min\{a_i + \omega_{\bar{i}} l_{\bar{i}i}, 1\} = \omega_i.$$

Then it is clear that $\omega_i = 1$ when $a_i + \omega_{\bar{i}} l_{\bar{i}i} > 1$. This suggests that bank's default can happen through owns debts when

$$A_i(T) < \lambda_i^{=}(T),$$

but also through contagion when,

$$L_i + L_{i\bar{i}} - \omega_{\bar{i}} L_{\bar{i}i} > A_i(T) \geq \lambda_i^{=}(T).$$

This suggests that we need to change our definition of the boundaries in T to $\tilde{\lambda}_{i,T} = L_i + L_{i\bar{i}} - \tilde{R}_{\bar{i}}(\omega_i) L_{\bar{i}i}$, where

$$\tilde{R}_{\bar{i}} = \min\left(1, \frac{A_{\bar{i}} + \omega_i L_{i\bar{i}}}{L_{\bar{i}} + \omega_i L_{\bar{i}i}}\right), \quad \bar{i} = 1, 2.$$

Then the default boundary of bank i in T depends on the asset of the other bank. When the asset of bank $\bar{i}$ grows from $\lambda_{\bar{i}}^{<}$ to $\lambda_{\bar{i}}^{=}$, the default boundary $\tilde{\lambda}_{i,T}$ moves from $\tilde{\lambda}_i^{=}$ to $\lambda_i^{=}$.
Furthermore we define the domain in which banks survive. Let D_1 be the domain in which only bank 1 survives. Similarly we define D_2 the domain in which only bank 2 survives. Finally we define with D_{12} the domain in which both banks survive. Defining the whole domain as D, we eventually use the notation $\hat{D} = D \backslash (D_1 \cup D_2 \cup D_{12})$ for the domain in which all banks default. For better understanding we show the domain in Figure 4.2.

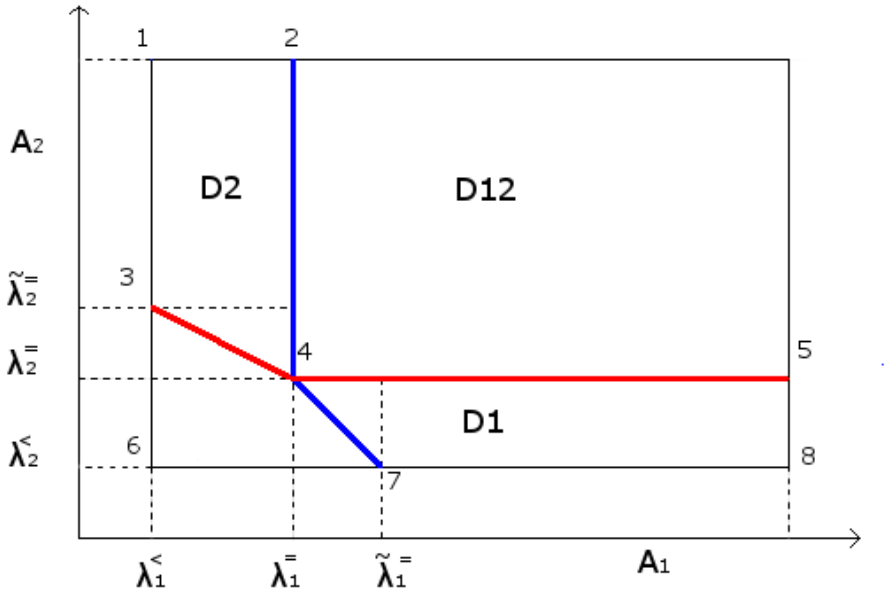

Figure 4.2: Default boundaries for $t = T$.

So the D_{12} is the area which lies above the line 2-4-5, D_1 is the area at the bottom-right of the line 5-4-7-8 and D_2 is the area top-left of the line 1-3-4-2.
In case of two banks the problem (4.4) reduces to solving a linear system for each domain.
In D_1 only bank 1 survives, i.e. it is able to pay his liabilities and therefore $A_1 + \omega_2 L_{21} > (L_1 + L_{12})$ while for the second bank $A_2 + \omega_1 L_{12} \leq A_2 + L_{12} < \omega_2(L_2 + L_{21})$. The linear system (4.4) becomes

$$\begin{cases} (L_1 + L_{12}) & = \omega_1(L_1 + L_{12}), \\ A_2 + \omega_1 L_{12} & = \omega_2(L_2 + L_{21}). \end{cases}$$

From which the solution is given by

$$\begin{aligned} \omega_1 &= 1, \\ \omega_2 &= \frac{A_2 + L_{12}}{L_2 + L_{12}}. \end{aligned}$$

Similar consideration holds for D2 and the solution is given by $\omega_1 = \frac{A_1 + L_{12}}{L_2 + L_{12}}$
In D_{12} the solution is trivially given by $\omega_1 = 1$ and $\omega_2 = 1$ because neither bank 1 nor bank 2 default.
More interesting is the case of $\hat{D}$. Here both bank 1 and bank 2 default, so

the system becomes

$$\begin{cases} A_1 + \omega_2 L_{21} = \omega_1 (L_1 + L_{12}), \\ A_2 + \omega_1 L_{12} = \omega_2 (L_2 + L_{21}). \end{cases}$$

Thus, after simple algebra, the solution is :

$$\omega_i = \frac{L_{\bar{i}} A_i + L_{\bar{i}\bar{i}} (A_i + A_{\bar{i}})}{L_1 L_2 + L_{12} L_2 + L_1 L_{21}} \quad i = 1, 2. \tag{4.5}$$

4.4 Formulation of backward Kolmogorov equation

For convenience, we introduce lognormal variables

$$X_i = \ln \Big(\frac{A_i}{\Lambda_i^<} \Big).$$

Let us denote also

$$\xi_i = -\bigg(\frac{\sigma_i^2}{2} + \kappa_i \lambda_i \bigg).$$

Applying Itô's formula to X_i, we find its dynamics

$$dX_i = \xi_i dt + \sigma_i dW_i(t) + J_i dN_i(t).$$

For details see Appendix B.
Substituting from equation (4.3) we obtain the default boundaries for the new variable X_i

$$\mu_i = \begin{cases} \mu_i^< = 0, & t < T, \\ \mu_i^= = \ln \left(\frac{\Lambda_i^=(t)}{\Lambda_i^<(t)} \right), & t = T. \end{cases} \tag{4.6}$$

Once we obtain the new default boundaries we can focus in pricing credit products.
Consider now a contract with terminal payoff $\psi(X_T)$, payment $\chi(s, x)$ at intermediate time $t \leq s \leq T$. Consider also payoffs $\phi_{1,0}(t, X_2(t))$ and $\phi_{2,0}(t, X_1(t))$ in case of prior default of bank 1 and bank 2 respectively. Then we can formulate the problem as

$$\begin{aligned} V(x,t) = \mathbb{E} \Big[& \psi(X_T) \cdot \mathbf{1}_{\{\tau \geq T\}} - \int_t^T \chi(s, X_s) \cdot \mathbf{1}_{\{\tau > s\}} ds \\ & + \phi_{1,0}(\tau_1, X_2(\tau_1)) \cdot \mathbf{1}_{\{\tau_1 < T\}} + \phi_{2,0}(\tau_2, X_1(\tau_2)) \cdot \mathbf{1}_{\{\tau_2 < T\}} | X(t) = x \Big], \end{aligned}$$

Thanks to Feynman-Kac formula we can move from a stochastic formulation of the problem to a deterministic formulation through a partial-integro

differential equation. The corresponding pricing equation is given by the following Kolmogorov backward equation: (for details see appendix C)

$$\frac{\partial V}{\partial t} + \mathcal{L}V = \chi(t,x), \qquad x := (x_1, x_2) \in \mathbb{R}^+ \times \mathbb{R}^+, \tag{4.7}$$

$$V(t,0,x_2) = \phi_{2,0}(t,x_2), \qquad V(t,x_1,x_2) \xrightarrow[x_1 \to +\infty]{} \phi_{2,\infty}(t,x_2), \tag{4.8}$$

$$V(t,x_1,0) = \phi_{1,0}(t,x_1), \qquad V(t,x_1,x_2) \xrightarrow[x_2 \to +\infty]{} \phi_{1,\infty}(t,x_1), \tag{4.9}$$

$$V(T,x) = \psi(x). \tag{4.10}$$

where

$$\begin{aligned}\mathcal{L}f = &\frac{\sigma_1^2}{2} f_{x_1x_1} + \rho\sigma_1\sigma_2 f_{x_1x_2} + \frac{\sigma_2^2}{2} f_{x_2x_2} + \xi_1 f_{x_1} + \xi_2 f_{x_2} \\ &+ \lambda_1 \mathcal{J}_1 f + \lambda_2 \mathcal{J}_2 f + \lambda_{12} \mathcal{J}_{12} f - (\lambda_1 + \lambda_2 + \lambda_{12}) f,\end{aligned} \tag{4.11}$$

$$\mathcal{J}_1 f(x) = \theta_1 \int_0^{x_1} f(x_1 - u, x_2) e^{-\theta_1 u} du, \tag{4.12}$$

$$\mathcal{J}_2 f(x) = \theta_2 \int_0^{x_2} f(x_1, x_2 - u) e^{-\theta_2 u} du, \tag{4.13}$$

$$\mathcal{J}_{12} f(x) = \theta_1\theta_2 \int_0^{x_1}\int_0^{x_2} f(x_1 - u, x_2 - \upsilon) e^{-\theta_1 u - \theta_2 \upsilon} d\upsilon, \tag{4.14}$$

and $\phi_{i,0}, \phi_{i,\infty}$ are given functions. The derivation of equations (4.12)-(4.14) is given in Appendix A. In the following, we formulate the Kolmogorov backward equation for specific quantities.

4.5 Pricing problem

In this section we formulate the problem for multiple credit products. Unlike the one dimensional case, adding an additional bank allows us to investigate the impact of the behaviour of the counterparty making the pricing problem much more interesting.

4.5.1 Marginal Survival Probability

In this section we compute the survival probability of bank i. Let's refer to the other bank with $j = 3 - i$. Although we are talking about marginal survival probability it's important to understand why it's a problem in both x_i and x_j. The two banks are indeed linked by the mutual liabilities. The survival probability of the i-th bank is directly linked to the j-th bank through the default boundaries. If bank j defaults then the default boundaries of bank i move to the right, and its probability of default increases.
Let's $q_i(t,x) = q_i(t,x_i,x_j)$ be the survival probability of the i-th bank.
The bank i survives when:

- There are no defaults before T.
- In T bank i does not default.
- The bank i survives also when bank j defaults.

The first event is described by $\mathbf{1}_{\{\tau \geq T\}}$.
As said before, in T the boundaries change. Therefore the stopping time τ is not able to describe any events in T. So we introduce $\mathbf{1}_{\{X_T \in D_i \cup D_{12}\}}$ to correctly describe the second event using the boundaries in T. Notice that here we are taking into account both the cases of no default and default of bank j. Indeed D_i is the region where bank i survives, but bank j defaults. For the third event, consider the case where bank j defaults before T, then bank i deals with the modified boundaries $\tilde{\mu}_1$. This new probability of survival is given by $\Xi(\tau_j, X_i(\tau_2))\mathbf{1}_{\{\tau_j < T\}}$. It's important to notice that like in the one dimensional case here we are not describing what happen in T. Infact it would be an error since the event "bank i survives in T, while bank j does not" is already considered in the second case. Finally the problem becomes

$$q_i(t,x) = \mathbb{E}\big[\mathbf{1}_{\{\tau \geq T, X_T \in D_i \cup D_{12}\}} + \Xi(\tau_j, X_i(\tau_j))\mathbf{1}_{\{\tau_j < T\}} \big| X(t) = x\big].$$

From equations (4.7)-(4.10) taking $\varphi(x) = \mathbf{1}_{\{x \in D_i \cup D_{12}\}}$ and $\chi(x,t) = 0$ we obtain

$$\begin{aligned}
&\frac{\partial}{\partial t} q_i(t,x) + \mathcal{L} q_i(t,x) = 0, \\
&q_i(t,0,x_j) = 0, \\
&q_i(t,x_i,0) = \Xi(t,x_i), \\
&V(T,x) = \mathbf{1}_{\{X_T \in D_i \cup D_{12}\}}.
\end{aligned}$$

with $\mathcal{L}$ defined in (4.11). The domain is once again semi infinite so it has no border conditions for $x_{i,j} \to +\infty$. In order to solve the PIDE we must formulate an artificial conditions. It seems natural to impose that for $x_i \to +\infty$ the solution tends to 1. For $x_j \to +\infty$, bank j can be considered not defaultable, then it's natural to consider that the solution tends to the survival marginal probability of bank i.
Therefore the problem becomes

$$\begin{aligned}
&\frac{\partial}{\partial t} q_i(t,x) + \mathcal{L} q_i(t,x) = 0, && x := (x_i, x_j) \in \mathbb{R}^+ \times \mathbb{R}^+, \\
&q_i(t,0,x_j) = 0, && q_i(t,x_i,x_j) \xrightarrow[x_i \to +\infty]{} 1, \\
&q_i(t,x_i,0) = \Xi(t,x_i), && q_i(t,x_i,x_j) \xrightarrow[x_j \to +\infty]{} \chi_{i,\infty}(t,x_i), \\
&q_i(T,x) = \mathbf{1}_{\{x \in D_i \cup D_{12}\}}. &&
\end{aligned}$$

As said before $\Xi(t, x_i)$ is the probability of survival of bank i when bank j defaults, i.e consider the modified boundaries $\tilde{\mu}$

$$\Xi(t, x_i) = \begin{cases} \chi_{i,0}(t, x_i), & x_i \geq \tilde{\mu}_i^<, \\ 0, & x_i < \tilde{\mu}_i^<. \end{cases}$$

It's easy to see that $\chi_{i,0}(t, x_i)$ solves the problem described in Section 3.4.1 with modified boundaries $\tilde{\mu}$. While $\chi_{i,\infty}(t, x_i)$, since is the probability when bank j can be considered not defaultable, solves the problem described in Section 3.4.1 with unmodified boundaries μ.

4.5.2 Joint Survival Probability

The joint survival probability $Q(t, x)$ is the probability that both banks survive. Then the problem is straightforward

$$Q(t, x) = \mathbb{E}\big[\mathbf{1}_{\{\tau \geq T, X_1(T) \geq \mu_1^=, X_2(T) \geq \mu_2^=\}}, |X(t) = x\big].$$

From equations (4.7)-(4.10), taking $\varphi(x) = \mathbf{1}_{\{x_1 \geq \mu_1^=, x_2 \geq \mu_2^=\}}$ and $\chi(x, t) = 0$ the problem becomes

$$\frac{\partial}{\partial t} Q(t, x) + \mathcal{L} Q(t, x) = 0,$$

with $\mathcal{L}$ defined in equation (4.11). The boundary conditions are given by

$$\begin{aligned} &Q(t, 0, x_2) = 0, \\ &Q(t, x_1, 0) = 0, \\ &Q(T, x) = \mathbf{1}_{\{x_1 \geq \mu_1^=, x_2 \geq \mu_2^=\}}. \end{aligned}$$

Once again we don't have border conditions for $x_1, x_2 \to +\infty$. For $x_1 \to +\infty$, it seems natural to consider the first bank not defaultable so the joint survival probability reduces to the marginal probability of bank 2. Similar consideration holds for $x_2 \to +\infty$.
Then the problem becomes

$$\begin{aligned} &\frac{\partial}{\partial t} Q(t, x) + \mathcal{L} Q(t, x) = 0, && x := (x_1, x_2) \in \mathbb{R}^+ \times \mathbb{R}^+, \\ &Q(t, 0, x_2) = 0, && Q(t, x_1, x_2) \xrightarrow[x_1 \to +\infty]{} \chi_{2,\infty}(t, x_2), \\ &Q(t, x_1, 0) = 0, && Q(t, x_1, x_2) \xrightarrow[x_2 \to +\infty]{} \chi_{1,\infty}(t, x_1), \\ &Q(T, x) = \mathbf{1}_{\{x_1 \geq \mu_1^=, x_2 \geq \mu_2^=\}}. \end{aligned}$$

where $\chi_{k,\infty}(t, x_k)$ with $k = 1, 2$ is the one dimensional survival probability of bank k considering the other bank not defaultable. Both solve the problem described in Section 3.4.1 with unmodified default boundaries μ_k.

4.5.3 Credit Default Swap

A Credit Default Swap (CDS) is a contract designed to eliminate credit risk over a Reference Name (RN). Protection Buyer (PB) buys a CDS over RN to a Protection Seller (PS). PB pays periodic coupon payments c until default of RN or maturity T. In exchange PS pays to PB the loss given RN's default. Consider $C_1(t,x)$ the value, from the PB perspective, of a CDS written on the first bank (RN) with coupon c. Assume that the coupon is paid continuously. According to [1] the value becomes

$$C_1(t,x) = \mathbb{E}\Big[\phi(X_T)\mathbf{1}_{\{\tau \geq T\}} - \int_t^T c\mathbf{1}_{\{\tau > s\}} ds + \Xi(\tau_2, X_1(\tau_2))\mathbf{1}_{\{\tau_2 < T\}} | X_t = x\Big],$$

where:

$$\phi(x) = \begin{cases} 1 - \min(R_1, \tilde{R}_1(1)), & (x_1,x_2) \in D_2, \\ 1 - \min(R_1, \tilde{R}_1(\omega_2)), & (x_1,x_2) \in D \backslash (D_1 \cup D_2 \cup D_{12}). \end{cases}$$

$$\tilde{R}_1(\omega_2) = \min\left[1, \frac{A_1(T) + \omega_2 L_{21}(T)}{L_1(T) + \omega_2 L_{12}(T)}\right].$$

$$\Xi(t,x_1) = \begin{cases} c_{1,0}(t,x_1), & x_1 \geq \tilde{\mu}_1^{<}, \\ 1 - R1, & x_1 < \tilde{\mu}_1^{<}. \end{cases}$$

$\tilde{R}_1(\omega_2)$ represents the percentage of asset respect to the total liabilities of the first bank in function of the fraction ω_2 of debt that bank 2 is able to pay at maturity. If bank 1 defaults then we have $\tilde{R}_1(\omega_2) < 1$ otherwise $\tilde{R}_1(\omega_2) = 1$.
Let's better analyse each term. At maturity T we have three main scenarios described by $\phi(x)$:

- **Bank 1 survives**: the value of CDS is 0 since no default occurs.
- **Bank 1 defaults, bank 2 survives**: In this case bank 2 pays all its debt and so $\omega_2 = 1$. Then since bank 1 defaults we have that $\tilde{R}_1(1) < 1$. This quantity represents the percentage that bank 1 is able to pay back to creditors. It may happen that the bank can recover even less than R_1. Then the value of CDS becomes $1 - \min(R_1, \tilde{R}_1(1))$.
- **Both banks default**: In this case bank 2 pays only a fraction of his debt and so $\omega_2 < 1$. Then since bank 1 defaults we have that $\tilde{R}_1(\omega_2) < 1$. This quantity represents the percentage that bank 1 is able to pay back to creditors given the fraction of debt that bank 2 is able to pay. Then the value of CDS becomes $1 - \min(R_1, \tilde{R}_1(\omega_2))$.

The term $-\int_t^T c\mathbf{1}_{\{\tau>s\}}ds$ takes into account the payments of coupon c until default or maturity T.
Finally $\Xi(\tau_2, X_1(\tau_2))\mathbf{1}_{\{\tau_2<T\}}$ describes what is the value of the CDS in case of bank 2 default. Since there are mutual liabilities the default of bank 2 can lead to bank 1 default and therefore activate the insurance of the CDS. The function $\Xi(\tau_2, X_1(\tau_2))\mathbf{1}_{\{\tau_2<T\}}$ accounts for both the following events:

- **Bank 1 survives despite bank 2 defaults**: due to the default of bank 2 the default boundaries of bank 1 move from μ to $\tilde{\mu}$. Despite the new boundaries bank 1 survives, so the insurance is not activated and the CDS value is then a function of only the bank 1 assets. Therefore the CDS value is given by the 1-dimensional CDS $c_{1,0}(t, x_1)$ with modified boundaries $\tilde{\mu}$.
- **Bank 1 defaults due to default of bank 2**: the RN defaults so the insurance is activated and the value of the CDS becomes $1 - R_1$.

From equations (4.7)-(4.10), taking $\varphi(x) = \phi(x)$ and $\chi(x,t) = c$ the problem becomes

$$\begin{aligned}
&\frac{\partial}{\partial t}C_1(t,x) + \mathcal{L}C_1(t,x) = c,\\
&C_1(t,0,x_2) = 1 - R_1,\\
&C_1(t,x_1,0) = \Xi(t,x_1) = \begin{cases} c_{1,0}(t,x_1), & x_1 \geq \tilde{\mu}_1^<,\\ 1-R1, & x_1 < \tilde{\mu}_1^<, \end{cases}\\
&C_1(T,x) = \phi(x).
\end{aligned}$$

For the condition at $x_i \to +\infty$ we formulate suitable condition.
If $x_1 \to +\infty$ the CDS can be considered as a risk-free annuity because the probability that RN defaults tends to zero. If $x_2 \to +\infty$ the CDS is given by $c_{1,\infty}(t, x_1)$. Both $c_{1,0}(t, x_1)$, $c_{1,\infty}(t, x_1)$ are the solutions of the one dimensional problem. Since $c_{1,0}(t, x_1)$ is the value of the CDS when bank 2 defaults, it solves the problem in Section 3.4.2 with modified default boundaries $\tilde{\mu}$. Similarly, $c_{1,\infty}(t, x_1)$ solves the problem with unmodified default boundaries μ.
Putting all together the final problem becomes

$$\begin{aligned}
&\frac{\partial}{\partial t}C_1(t,x) + \mathcal{L}C_1(t,x) = c, && x := (x_1,x_2) \in \mathbb{R}^+ \times \mathbb{R}^+,\\
&C_1(t,0,x_2) = 1 - R_1, && C_1(t,x_1,x_2) \xrightarrow[x_1\to+\infty]{} -c(T-t),\\
&C_1(t,x_1,0) = \Xi(t,x_1) = \begin{cases} c_{1,0}(t,x_1), & x_1 \geq \tilde{\mu}_1^<,\\ 1-R1, & x_1 < \tilde{\mu}_1^<, \end{cases} && C_1(t,x_1,x_2) \xrightarrow[x_2\to+\infty]{} c_{1,\infty}(t,x_1),\\
&C_1(T,x) = \phi(x).
\end{aligned}$$

4.5.4 First-To-Default swap

A First-To-Default swap (FTD) is a contract designed to eliminate credit risk of first default over a basket of Reference Names(RNs). Protection

Buyer(PB) buys a FTD over RNs to a Protection Seller(PS). PB pays periodic coupon payments c to PS up to the first default of any of the RNs in the basket or maturity T. In exchange PS pays to PB the loss given the first default. Consider $F(t,x)$ the price of a FTD written on the basket of the two banks (RNs) with coupon c. Assume that the coupon is paid continuously. According to [13] the value becomes

$$F(t,x) = \mathbb{E}\Big[\phi(X_T)\mathbf{1}_{\{\tau>T\}} - \int_t^T c\mathbf{1}_{\{\tau>s\}}ds + (1-R_1)\mathbf{1}_{\{\tau_1<T\}} + (1-R_2)\mathbf{1}_{\{\tau_2<T\}} | X_t = x\Big], \tag{4.15}$$

where

$$\phi(x) = \beta_0\mathbf{1}_{\{x\in D\setminus(D_1\cup D_2\cup D_{12})\}} + \beta_1\mathbf{1}_{\{x\in D_1\}} + \beta_2\mathbf{1}_{\{x\in D_2\}},$$

$$\beta_0 = 1 - \min\big[\min(R_1, \tilde{R}_1(\omega_2)), \min(R_2, \tilde{R}_2(\omega_1))\big],$$
$$\beta_1 = 1 - \min(R_2, \tilde{R}_2(1)) \qquad \beta_2 = 1 - \min(R_1, \tilde{R}_1(1)),$$

and

$$\tilde{R}_1(\omega_2) = \min\left[1, \frac{A_1(T) + \omega_2 L_{21}(T)}{L_1(T) + \omega_2 L_{12}(T)}\right], \quad \tilde{R}_2(\omega_1) = \min\left[1, \frac{A_2(T) + \omega_1 L_{12}(T)}{L_2(T) + \omega_1 L_{21}(T)}\right].$$

Let's better analyse each term. At maturity T we have three main scenarios described by $\phi(x)$:

- **Bank 1 survives, bank 2 defaults**: similar to the CDS, the value of FTD is $1 - \min(R_2, \tilde{R}_2(1))$ since bank 1 is able to fully pays its debts.
- **Bank 1 defaults, bank 2 survives**: the value of FTD is given by $1 - \min(R_1, \tilde{R}_1(1))$ since bank 2 is able to fully pays its debts.
- **Both banks default**: Both banks default simultaneously therefore the definition of "first default "is ambiguous. For convention we assume that in case of simultaneous defaults the first bank that defaults is the one with less percentage of recovery. Since both banks default, both pay only a fraction ω_1 and ω_2 of their debts, therefore $\omega_1 < 1$ and $\omega_2 < 1$. Similar to the case of CDS it may happen that the banks can even recover less than the recovery rate, then the value of FTD becomes $1 - \min(\min(R_1, \tilde{R}_1(\omega_2)), \min(R_2, \tilde{R}_2(\omega_1)))$.

The term $-\int_t^T c\mathbf{1}_{\{\tau>s\}}ds$ takes into account the payments of coupon c until first default or maturity T.
The last two term of equation (4.15) represent the value of FTD in case of prior defaults. If bank 1 (bank 2) defaults before maturity the FTD value would be $1 - R_1$ $(1 - R_2)$.

From equations (4.7)-(4.10), taking $\varphi(x) = \phi(x)$ and $\chi(x,t) = c$ the problem becomes

$$\begin{aligned}
&\frac{\partial}{\partial t}F(t,x) + \mathcal{L}F(t,x) = c,\\
&F(t,0,x_2) = 1 - R_1,\\
&F(t,x_1,0) = 1 - R_2,\\
&F(T,x) = \beta_0 \mathbf{1}_{\{x \in D \setminus (D_1 \cup D_2 \cup D_{12})\}} + \beta_1 \mathbf{1}_{\{x \in D_1\}} + \beta_2 \mathbf{1}_{\{x \in D_2\}}.
\end{aligned}$$

For the condition at $x_i \to +\infty$ we assume suitable condition. If $x_1 \to +\infty$ the FTD can be considered as a CDS written on the second bank with value $f_{2,\infty}(t,x_2)$. If $x_2 \to +\infty$ the FTD can be considered as a CDS written on the first bank with value $f_{1,\infty}(t,x_1)$. So $fi,\infty(t,x_i)$ is the solution of the one dimensional problem in Section 3.4.2 where, since bank $3-i$ does not default, we consider the unmodified default boundaries μ.
Putting all together we obtain that the price of FTD solves

$$\begin{aligned}
&\frac{\partial}{\partial t}F(t,x) + \mathcal{L}F(t,x) = c, \quad x := (x_1,x_2) \in \mathbb{R}^+ \times \mathbb{R}^+,\\
&F(t,0,x_2) = 1 - R_1, \qquad F(t,x_1,x_2) \xrightarrow[x_1 \to +\infty]{} f_{2,\infty}(t,x_2),\\
&F(t,x_1,0) = 1 - R_2, \qquad F(t,x_1,x_2) \xrightarrow[x_2 \to +\infty]{} f_{1,\infty}(t,x_1),\\
&F(T,x) = \beta_0 \mathbf{1}_{\{x \in D \setminus (D_1 \cup D_2 \cup D_{12})\}} + \beta_1 \mathbf{1}_{\{x \in D_1\}} + \beta_2 \mathbf{1}_{\{x \in D_2\}}.
\end{aligned}$$

4.5.5 Credit and Debt Value Adjustments for CDS

Credit Value Adjustment (CVA) and Debt Value Adjustment (DVA) are measure that take into account the probability of default of a counterparty. There exists both unilateral and bilateral adjustments. We only focus on the unilateral case.
Unilateral CVA and DVA Consider the case of a CDS written on one bank RN. Consider also that the Protection Seller is the other bank. In this way both RN and PS can default, while the Protection Buyer is considered not defaultable. Then we need to take into account the risk that PB can have in case of default of PS. Following [13], the CVA can be defined as

$$V^{CVA} = (1 - R_{PS})\mathbb{E}\big[\mathbf{1}_{\{\tau^{PS} < \min(T,\tau^{RN})\}}(V^{CDS}_{\tau^{PS}})^+ | \mathcal{F}_t\big],$$

where R_{PS} is the recovery rate of PS, τ^{PS} and τ^{RN} are the default time of PS and RN, respectively. V^{CDS}_t is the price of a CDS without counterparty credit risk. From the definition is clear that the CVA is an adjustment to the CDS value. Let's analyse each scenario.

- $\tau_{RN} < \tau_{PS}$: in this case the RN entity defaults before the PS, so at time τ_{RN} the PS is able to honour the payment, so the correction is 0.

- $\tau_{PS} < \tau_{RN}$: in this case the PS defaults before RN , therefore is not able to pay the PB, but PB can recover part of the CDS value, if any. So the PB recovers $R_{PS}(V^{CDS}_{\tau^{PS}})^+$. Therefore in case of default of PS, we must consider that PB has a loss of $(1 - R_{PS})(V^{CDS}_{\tau^{PS}})^+$.

We associate x_1 with RN and x_2 with PS. Then the CVA can be given by the solution of equations (4.7)-(4.10) with $\chi(t, x) = 0$ and $\phi(x) = 0$. Thus,

$$\begin{aligned}
&\frac{\partial}{\partial t}V^{CVA} + \mathcal{L}V^{CVA} = 0, && x := (x_1, x_2) \in \mathbb{R}^+ \times \mathbb{R}^+,\\
&V^{CVA}(t, 0, x_2) = 0, && V^{CVA}(t, x_1, x_2) \xrightarrow[x_1 \to +\infty]{} 0,\\
&V^{CVA}(t, x_1, 0) = (1 - R_{PS})V^{CDS}(t, x_1)^+, && V^{CVA}(t, x_1, x_2) \xrightarrow[x_2 \to +\infty]{} 0,\\
&V^{CVA}(T, x) = 0.
\end{aligned}$$

The condition at $x_i \to +\infty$ is obtained considering the following. In case of $x_1 \to +\infty$ the RN entity can be considered not defaultable and therefore no correction is needed. Similarly in the case of $x_2 \to +\infty$ the PB can be considered not defaultable, therefore it always be able to honour the payments. Then, once again, the correction is 0. The DVA is specular to the CVA. Now we consider the case of a CDS written on one bank RN and we consider the other bank as Protection Buyer(PB). We also consider both banks defaultable, while the Protection Seller is considered not defaultable. Then we need to take into account the additional risk that PS can have in case of default of PB. Then the DVA can be defined as

$$V^{DVA} = (1 - R_{PB})\mathbb{E}\big[\mathbf{1}_{\{\tau^{PB} < \min(T, \tau^{RN})\}}(V^{CDS}_{\tau^{PB}})^- | \mathcal{F}_t\big],$$

where R_{PB} is the recovery rate of PB, τ^{PB} and τ^{RN} are the default time of PB and RN, respectively. We associate x_1 with PB and x_2 with RN. Then the DVA can be given by the solution of equations (4.7)-(4.10) with $\chi(t, x) = 0$ and $\phi(x) = 0$. Thus,

$$\begin{aligned}
&\frac{\partial}{\partial t}V^{DVA} + \mathcal{L}V^{DVA} = 0, && x := (x_1, x_2) \in \mathbb{R}^+ \times \mathbb{R}^+,\\
&V^{CVA}(t, 0, x_2) = (1 - R_{PB})V^{CDS}(t, x_2)^-, && V^{DVA}(t, x_1, x_2) \xrightarrow[x_1 \to +\infty]{} 0,\\
&V^{DVA}(t, x_1, 0) = 0, && V^{DVA}(t, x_1, x_2) \xrightarrow[x_2 \to +\infty]{} 0,\\
&V^{DVA}(T, x) = 0.
\end{aligned}$$

.

4.6 Numerical scheme

In this section following the step of Section 3.5 we provide numerical method for solving the problem stated in equations (4.7)-(4.10).

4.6.1 Grid discretization

Following the one dimensional case the grid is created by redistribution of uniform grid. Since the two space variables are independent we can use the same transformation for both dimensions x_1, x_2.

The grid

The domain of the PIDE is given by $D = [0, \infty) \times [0, \infty)$. First of all we approximate the unbounded domain D with finite domain $\tilde{D} = [0, x_1^{max}] \times [0, x_2^{max}]$. We use $N_1 + 1$ points for x_1 and $N_2 + 1$ points for x_2. Then the grid becomes

$$0 = x_1^0 < x_1^1 < \cdots < x_1^{N_1},$$
$$0 = x_2^0 < x_2^1 < \cdots < x_2^{N_2},$$

From the numerical perspective is better to deal with vectors rather than matrices. So we enumerates the points like in Figures 4.3.

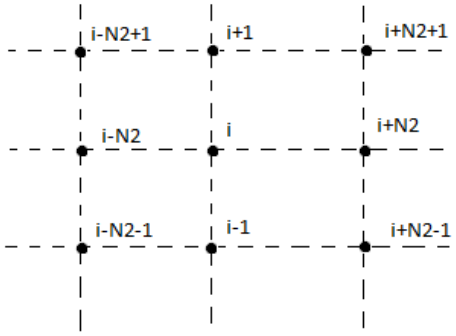

Figure 4.3: Grid enumeration

Then we consider a vector of points

$$(x_1^0, x_2^0), (x_1^0, x_2^1), \ldots, (x_1^1, x_2^{N_2}), (x_1^1, x_2^0), \ldots, (x_1^1, x_2^{N_2}), \ldots (x_1^{N_1}, x_2^{N_2}).$$

In such a way the solution is a vector $U \in \mathbb{R}^{(N_1+1)(N_2+1)\times 1}$ whose component i is the solution at the point $(x_1 = x_1^{\lfloor (i-1)/(N_2+1) \rfloor}, x_2 = x_2^{i(mod(N_2+1)}$).
In two dimensional case we have two interesting default boundaries, i.e. μ and $\tilde{\mu}$, so one can be interested in a transformation with multiple critical barriers.

Multiple Critical Points

In the case of multiple critical points there is not explicit formula for the transformation. Like the case of single critical point we focus on the Jacobian. We define a global Jacobian that combines individual Jacobians for

each critical point. We choose to combine each individual Jacobian with harmonic squared average that guarantees smooth transformation

$$J_k(\xi) = (\alpha_k^2 + (S(\xi) - B_k)^2)^{\frac{1}{2}},$$

and

$$J(\xi) = \left[\sum_{k=1}^{k=n} J_k(\xi)^{-2}\right]^{-\frac{1}{2}}.$$

Near each critical point the global Jacobian $J(\xi)$ is dominated by the behaviour of the local $J_k(\xi)$, but the influence of nearby critical points ensures that the transition between them are smooth. As said before there is not closed formula and the global Jacobian must be integrated numerically. The integration can be computed with standard ODE integrator like Runge-Kutta.

4.6.2 Differential Operator

For completeness we report the discretization of differential operator also in the two dimensional case. This is a trivial extension of the one dimensional case, except for the case of mixed derivatives. We use the following scheme for derivative over x_1:

$$\frac{\partial V}{\partial x_1}(x_1^i, x_2^j) \approx \alpha_{i,-2}^1 V(x_1^{i-2}, x_2^j) + \alpha_{i,-1}^1 V(x_1^{i-1}, x_2^j) + \alpha_{i,0}^1 V(x_1^i, x_2^j), \quad (4.16)$$

$$\frac{\partial V}{\partial x_1}(x_1^i, x_2^j) \approx \beta_{i,-1}^1 V(x_1^{i-1}, x_2^j) + \beta_{i,0}^1 V(x_1^i, x_2^j) + \beta_{i,1}^1 V(x_1^{i+1}, x_2^j), \quad (4.17)$$

$$\frac{\partial V}{\partial x_1}(x_1^i, x_2^j) \approx \gamma_{i,0}^1 V(x_1^i, x_2^j) + \gamma_{i,1}^1 V(x_1^{i+1}, x_2^j) + \gamma_{i,2}^1 V(x_1^{i+2}, x_2^j), \quad (4.18)$$

while for derivative over x_2 we consider

$$\frac{\partial V}{\partial x_2}(x_1^i, x_2^j) \approx \alpha_{j,-2}^2 V(x_1^i, x_2^{j-2}) + \alpha_{j,-1}^2 V(x_1^i, x_2^{j-1}) + \alpha_{j,0}^2 V(x_1^i, x_2^j), \quad (4.19)$$

$$\frac{\partial V}{\partial x_2}(x_1^i, x_2^j) \approx \beta_{j,-1}^2 V(x_1^i, x_2^{j-1}) + \beta_{j,0}^2 V(x_1^i, x_2^j) + \beta_{j,1}^2 V(x_1^i, x_2^{j+1}), \quad (4.20)$$

$$\frac{\partial V}{\partial x_2}(x_1^i, x_2^j) \approx \gamma_{j,0}^2 V(x_1^i, x_2^j) + \gamma_{j,1}^2 V(x_1^i, x_2^{j+1}) + \gamma_{j,2}^2 V(x_1^i, x_2^{j+2}), \quad (4.21)$$

with coefficients

$$\alpha_{i,-2}^k = \frac{\Delta x_i^k}{\Delta x_k^{i-1}(\Delta x_k^{i-1} + \Delta x_k^i)}, \quad \alpha_{i,-1}^k = \frac{-\Delta x_{i-1}^k - \Delta x_k^i}{\Delta x_k^i \Delta x_k^{i-1}}, \quad \alpha_{i,0}^k = \frac{\Delta x_k^{i-1} + 2\Delta x_k^i}{\Delta x_k^i(\Delta x_k^i + \Delta x_k^{i-1})},$$

$$\beta_{i,-1}^k = \frac{-\Delta x_k^{i+1}}{\Delta x_k^i(\Delta x_k^i + \Delta x_k^{i+1})}, \quad \beta_{i,0}^k = \frac{\Delta x_k^{i+1} - \Delta x_k^i}{\Delta x_k^i \Delta x_k^{i+1}}, \quad \beta_{i,1}^k = \frac{\Delta x_k^i}{\Delta x_k^{i+1}(\Delta x_k^i + \Delta x_k^{i+1})},$$

$$\gamma_{i,0}^k = \frac{\Delta x_k^{i+2} - 2\Delta x_k^{i+1}}{\Delta x_k^{i+1}(\Delta x_k^{i+1} + \Delta x_k^{i+2})}, \quad \gamma_{i,1}^k = \frac{-\Delta x_k^{i+1} - \Delta x_k^{i+2}}{\Delta x_k^{i+1} \Delta x_k^{i+2}}, \quad \gamma_{i,2}^k = \frac{-\Delta x_k^{i+1}}{\Delta x_k^{i+1}(\Delta x_k^{i+1} + \Delta x_k^{i+2})}.$$

where $\Delta x_k^i = x_k^i - x_k^{i-1} \quad k = 1, 2$.
To approximate the second derivative we use the central scheme

$$\frac{\partial^2 V}{\partial x_1^2}(x_1^i, x_2^j) \approx \delta_{i,-1}^1 V(x_1^{i-1}, x_2^j) + \delta_{i,0}^1 V(x_1^i, x_2^j) + \delta_{i,1}^1 V(x_1^{i+1}, x_2^j), \quad (4.22)$$

$$\frac{\partial^2 V}{\partial x_2^2}(x_1^i, x_2^j) \approx \delta_{j,-1}^2 V(x_1^i, x_2^{j-1}) + \delta_{j,0}^2 V(x_1^i, x_2^j) + \delta_{j,1}^2 V(x_1^i, x_2^{j+1}), \quad (4.23)$$

with coefficients

$$\delta_{i,-1}^k = \frac{2}{\Delta x_k^i (\Delta x_k^i + \Delta x_k^{i+1})}, \quad \delta_{i,0}^k = \frac{-2}{\Delta x_k^i \Delta x_k^{i+1}}, \quad \delta_{i,1}^k = \frac{2}{\Delta x_k^{i+1} (\Delta x_k^i + \Delta x_k^{i+1})}.$$

Respect to the one dimensional case we must deal with mixed derivative term. We will use standard 9-point stencil scheme for non uniform grid, i.e.

$$\frac{\partial^2 V}{\partial x_1 \partial x_2}(x_1^i, x_2^j) \approx \sum_{k,l=-1}^{1} \beta_{i,k}^1 \beta_{j,l}^2 V(x_1^{i+k}, x_2^{j+l}).$$

As a result the differential operator $\mathcal{D}V$ can be approximated by

$$DV = D_1 V + D_2 V + D_{12} V.$$

where $D_1 V$ contains the discretization of the derivatives over x_1, $D_2 V$ contains the discretization of the derivatives over x_2 and $D_{12} V$ contains the discretization of the mixed derivative. We emphasize this dimensional splitting of the operator because it is needed in the time discretization of Chapter 5. Similar to the one dimensional case the scheme have a second order truncation error for meshes that are either uniform or a smooth transformation of such meshes.

Numerical

We consider a grid with $N_1 + 1$ points for x_1 and $N_2 + 1$ points for x_2. Then the matrix $\boldsymbol{D2} \in \mathbb{R}^{(N_1+1)(N_2+1)\times(N_1+1)(N_2+1)}$ is a block diagonal matrix

$$\boldsymbol{D2} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & D2 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & D2 & & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where $\mathbf{0} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$ is the matrix full of zeros and $D2 \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$ is given by

$$D2 = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ \Delta^2_{1,-1} & \Delta^2_{1,0} & \Delta^2_{1,1} & 0 & \dots & 0 \\ 0 & \Delta^2_{2,-1} & \Delta^2_{2,0} & \Delta^2_{2,1} & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \Delta^2_{N_2-1,-1} & \Delta^2_{N_2-1,0} & \Delta^2_{N_2-1,1} \\ 0 & \dots & 0 & 0 & 0 & 0 \end{pmatrix},$$

where :

- $\Delta^2_{i,j} = \xi_2 \beta^2_{i,j} + \frac{\sigma_2^2}{2} \delta^2_{i,j}$.

The matrix $\boldsymbol{D1} \in \mathbb{R}^{(N_1+1)(N_2+1)\times(N_1+1)(N_2+1)}$ is a tridiagonal matrix

$$\boldsymbol{D1} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & & \dots & \mathbf{0} \\ L_1 & C_1 & U_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & L_2 & C_2 & U_2 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & L_{N_1-1} & C_{N_1-1} & U_{N_1-1} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where :

- $\mathbf{0} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$ is the zero matrix.
- $L_i = \begin{pmatrix} 0 & 0 & \dots & & 0 \\ 0 & \Delta^1_{i,-1} & \dots & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & \Delta^1_{i,-1} & 0 \\ 0 & \dots & & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$.
- $C_i = \begin{pmatrix} 0 & 0 & \dots & & 0 \\ 0 & \Delta^1_{i,0} & \dots & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & \Delta^1_{i,0} & 0 \\ 0 & \dots & & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$.
- $U_i = \begin{pmatrix} 0 & 0 & \dots & & 0 \\ 0 & \Delta^1_{i,1} & \dots & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & \Delta^1_{i,1} & 0 \\ 0 & \dots & & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$.

and $\Delta^1_{i,j} = \xi_1 \beta_1^{i,j} + \frac{\sigma_1^2}{2} \delta^1_{i,j}$.
The matrix $\boldsymbol{D12} \in \mathbb{R}^{(N_1+1)(N_2+1)\times(N_1+1)(N_2+1)}$ is a nine-diagonal matrix

$$\boldsymbol{D12} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & & \dots & \mathbf{0} \\ L_1 & C_1 & U_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & L_2 & C_2 & U_2 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & L_{N_1-1} & C_{N_1-1} & U_{N_1-1} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where :

- $\mathbf{0} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}$ is the zero matrix.
- $C_i = \begin{pmatrix} 0 & 0 & 0 & & \dots & 0 \\ B^{i,0}_{1,-1} & B^{i,0}_{1,0} & B^{i,0}_{1,1} & & \dots & 0 \\ 0 & B^{i,0}_{2,-1} & B^{i,0}_{2,0} & B^{i,0}_{2,1} & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & B^{i,0}_{N_2-1,-1} & B^{i,0}_{N_2-1,0} & B^{i,0}_{N_2-1,1} \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}.$
- $U_i = \begin{pmatrix} 0 & 0 & 0 & & \dots & 0 \\ B^{i,1}_{1,-1} & B^{i,1}_{1,0} & B^{i,1}_{1,1} & & \dots & 0 \\ 0 & B^{i,1}_{2,-1} & B^{i,1}_{2,0} & B^{i,1}_{2,1} & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & B^{i,1}_{N_2-1,-1} & B^{i,1}_{N_2-1,0} & B^{i,1}_{N_2-1,1} \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}.$
- $L_i = \begin{pmatrix} 0 & 0 & 0 & & \dots & 0 \\ B^{i,-1}_{1,-1} & B^{i,-1}_{1,0} & B^{i,-1}_{1,1} & & \dots & 0 \\ 0 & B^{i,-1}_{2,-1} & B^{i,-1}_{2,0} & B^{i,-1}_{2,1} & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & B^{i,-1}_{N_2-1,-1} & B^{i,-1}_{N_2-1,0} & B^{i,-1}_{N_2-1,1} \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(N_2+1)\times(N_2+1)}.$

and $B^{i,k}_{j,l} = \rho\sigma_1\sigma_2\beta^1_{i,k}\beta^2_{j,l}$.
Notice that the zeros of the matrices in the rows relative to boundaries need to deal for Dirichlet boundary condition when discretizing over time. We analyse this aspect in Section 4.6.4 and Section 5.1.

4.6.3 Jump Discretization

The jump discretization is the same as in the one dimensional case for the jump over x_1 and x_2. We now also consider contribution of systemic jump.

First method

As shown in Section 3.5.3 using the first approach the integral can be approximated as

$$\mathcal{J}_1 V(x_1+h,x_2) = e^{-\theta_1 h}\mathcal{J}_1 V(x_1,x_2) + \omega_0(\theta_1,h)V(x_1,x_2) + \omega_1(\theta_1,h)V(x_1+h,x_2) + \mathcal{O}(h^3),$$
$$\mathcal{J}_2 V(x_1,x_2+h) = e^{-\theta_2 h}\mathcal{J}_2 V(x_1,x_2) + \omega_0(\theta_2,h)V(x_1,x_2) + \omega_1(\theta_2,h)V(x_1,x_2+h) + \mathcal{O}(h^3),$$

where

$$\omega_0(\theta_i,h_i) = \frac{1-(1+\theta_i h_i)e^{-\theta_i h_i}}{\theta_i h_i}, \quad \omega_1(\theta_i,h_i) = \frac{-1+\theta h_i + e^{-\theta_i h_i}}{\theta_i h_i}, \quad i=1,2,$$

with the increments $h_{i+1}^1 = x_1^{i+1} - x_1^i$, $h_{j+1}^2 = x_2^{j+1} - x_2^j$.
Given the grid as in Section 4.6.1 we denote with $J_1^{i,j}$, $J_2^{i,j}$, $J_{12}^{i,j}$ the approximation of $\mathcal{J}_1 V(x_1^i, x_2^j)$, $\mathcal{J}_2 V(x_1^i, x_2^j)$, $\mathcal{J}_{12} V(x_1^i, x_2^j)$ on the grid. Then

$$J_1^{i+1,j} = e^{-\theta_1 h_{i+1}^1} J_1^{i,j} + \omega_0(\theta_1, h_{i+1}^1)V(x_1^i,x_2^j) + \omega_1(\theta_1,h_{i+1}^1)V(x_1^{i+1},x_2^j),$$
$$J_2^{i,j+1} = e^{-\theta_2 h_{j+1}^2} J_2^{i,j} + \omega_0(\theta_2, h_{j+1}^2)V(x_1^i,x_2^j) + \omega_1(\theta_2,h_{j+1}^2)V(x_1^i,x_2^{j+1}),$$

where $J_1^{0,j} = 0 \quad \forall j$, $J_2^{i,0} = 0 \quad \forall i$.
In order to approximate $\mathcal{J}_{12} = \mathcal{J}_1\mathcal{J}_2 V$ we apply above approximations for $\mathcal{J}_1$ and $\mathcal{J}_2$ consecutively. Then the two-step procedure becomes

$$I_{12}^{i+1,j} = e^{-\theta_1 h_{i+1}^1} I_{12}^{i,j} + \frac{1}{2}\theta_1 h_{i+1}^1 e^{-\theta_1 h_{i+1}^1} V(x_1^i,x_2^j) + \frac{1}{2}\theta_1 h_{i+1}^1 V(x_1^{i+1},x_2^j),$$

and

$$J_{12}^{i,j+1} = e^{-\theta_2 h_{j+1}^2} J_{12}^{i,j} + \frac{1}{2}\theta_2 h_{j+1}^2 e^{-\theta_2 h_{j+1}^2} I_{12}^{i,j} + \frac{1}{2}\theta_2 h_{j+1}^2 I_{12}^{i,j+1}.$$

Second Method

For the second method we rewrite the jump operator in equations (4.12)-(4.14) as differential equations

$$\frac{\partial}{\partial x_1}\left(\mathcal{J}_1 V(x_1,x_2)e^{\theta_1 x_1}\right) = \theta_1 V(x_1,x_2)e^{\theta_1 x_1},$$
$$\frac{\partial}{\partial x_2}\left(\mathcal{J}_2 V(x_1,x_2)e^{\theta_2 x_2}\right) = \theta_2 V(x_1,x_2)e^{\theta_2 x_2},$$
$$\frac{\partial^2}{\partial x_1 \partial x_2}\left(\mathcal{J}_{12} V(x_1,x_2)e^{\theta_1 x_1+\theta_2 x_2}\right) = \theta_1\theta_2 V(x_1,x_2)e^{\theta_1 x_1+\theta_2 x_2}.$$

To solve these differential equation we apply Adam-Moulton method of second order which gives us third order of accuracy locally as shown in [3].

Then

$$J_1^{i+1,j} = e^{-\theta_1 h_{i+1}^1} J_1^{i,j} + \frac{1}{2}\theta_1 h_{i+1}^1 e^{-\theta_1 h_{i+1}^1} V(x_1^i, x_2^j) + \frac{1}{2}\theta_1 h_{i+1}^1 V(x_1^{i+1}, x_2^j),$$
$$J_2^{i,j+1} = e^{-\theta_2 h_{j+1}^2} J_2^{i,j} + \frac{1}{2}\theta_2 h_{j+1}^2 e^{-\theta_2 h_{j+1}^2} V(x_1^i, x_2^j) + \frac{1}{2}\theta_2 h_{j+1}^2 V(x_1^i, x_2^{j+1}),$$
$$I_{12}^{i+1,j} = e^{-\theta_1 h_{i+1}^1} I_{12}^{i,j} + \frac{1}{2}\theta_1 h_{i+1}^1 e^{-\theta_1 h_{i+1}^1} V(x_1^i, x_2^j) + \frac{1}{2}\theta_1 h_{i+1}^1 V(x_1^{i+1}, x_2^j),$$
$$J_{12}^{i,j+1} = e^{-\theta_2 h_{j+1}^2} J_{12}^{i,j} + \frac{1}{2}\theta_2 h_{j+1}^2 e^{-\theta_2 h_{j+1}^2} I_{12}^{i,j} + \frac{1}{2}\theta_2 h_{j+1}^2 I_{12}^{i,j+1}.$$

This scheme is equivalent to the trapezoidal rule.
Defining

$$\omega_0(\theta_i, h_i) = \frac{1}{2}\theta_i h_i e^{-\theta_i h_i}, \quad \omega_1(\theta_i, h_i) = \frac{1}{2}\theta_i h_i, \quad i = 1, 2,$$

we find the same formulas of the first method.
With these recursive formulas the jump operator can be discretized in all the node of the grid with a complexity of $\mathcal{O}(N_1 N_2)$. Once again notice that the efficiency of these recursive formulas are given by the special feature of the exponential distribution.
Finally the total jump operator can be approximated as $J = \lambda_1 J_1 + \lambda_2 J_2 + \lambda_{12} J_{12}$.

4.6.4 Time discretization

Like in the one dimensional case we compute a change of time variable to deal with forward-in-time problem.
Defining $\tau = T - t$, the problem becomes

$$\frac{\partial V}{\partial \tau} = \mathcal{L}V - \chi(\tau, x), \tag{4.24}$$
$$V(\tau, 0, x_2) = \phi_{2,0}(\tau, x_2), \qquad V(\tau, x_1, x_2) \xrightarrow[x_1 \to +\infty]{} \phi_{2,\infty}(\tau, x_2), \tag{4.25}$$
$$V(\tau, x_1, 0) = \phi_{1,0}(\tau, x_1), \qquad V(\tau, x_1, x_2) \xrightarrow[x_2 \to +\infty]{} \phi_{1,\infty}(\tau, x_1), \tag{4.26}$$
$$V(0, x) = \psi(x). \tag{4.27}$$

From the result of previous Section we can approximate $\mathcal{L}V$ with $LV = DV + \lambda JV - \lambda V$. With this discretization the problem becomes once again a system of ordinary linear differential equations. Define the vector $U(t) \in \mathbb{R}^{(N_1+1)(N_2+1)\times 1}$ whose i-th component represents the solution $V(t, x_i, x_j)$ with the convention of Section 4.6.1 for the enumeration of nodes. Furthermore, from Section 4.6.3 we can recursive compute the jump operator and so we treat it explicitly. Then the problem reduces to

$$U'(t) = AU(t) + J(U(t)) + b(t),$$
$$U(0) = U_0.$$

where $A = D - \lambda I$, $J(U(t))$ is the jump operator computed with the current value of the solution and $b(t)$ takes into account for both boundary conditions and non homogeneous term of the PIDE. As in the one dimensional case our intention is to solve this problem with Crank-Nicolson scheme. Therefore

$$\frac{U^{n+1} - U^n}{\Delta t} = \frac{1}{2}\left[AU^{n+1} + AU^n\right] + J(U(n)).$$

Solving respect to U^{n+1} we get the linear system

$$\left(I - \frac{1}{2}\Delta t A\right)U^{n+1} = \left(I + \frac{1}{2}\Delta t A\right)U^n + \Delta t J(U^n) + b(t_{n+1}). \tag{4.28}$$

In order to strongly impose the border condition we take the value of matrix A for the rows relative to border nodes. Let denote with K the set of indices relative to border nodes. For each index $k \in K$, the k-th row of matrix A is replaced by a row of zeros. Moreover at the beginning of each time iteration $n+1$, for all $k \in K$, we also correct to 0 the values of the k-th component of the previous solution U_n. Same correction is made to the jump vector. In such a way for boundary nodes we are imposing the boundary condition. For example, let $k \in K$ be the index of a border node. After the correction the k-th equation of the system given by (4.28) becomes

$$U_k^{n+1} = U_k^n + J(U^n)_k + b(t_{n+1})_k = b(t_{n+1})_k,$$

where the subscript k indicates the k-th component of the vector.

4.7 Results

In this section we analyse the impact of both jumps and mutual liabilities. In order to properly compare the result with the one dimensional case we take the same marginal parameters. We therefore consider the parameters in Table 4.1.

$L_{1,0}$	$L_{2,0}$	$L_{12,0}$	$L_{21,0}$	R_1	R_2	T	σ_1	σ_2	ρ
60	70	10	15	0.4	0.45	1	0.4	0.3	0.51

Table 4.1: Model parameters

For the model with jump we also consider the parameters in Table 4.2.

θ_1	θ_2	λ_1	λ_2	λ_{12}
1	1	0.095	0.055	0.012

Table 4.2: Jump parameters

All the test are computed using a spacial grid with 100 points for both x_1 and x_2 and 100 time steps grid. For the spacial domain we truncate it at almost one order of magnitude more than the default boundaries. With parameters as in Table 4.1 the default boundaries are given by $\tilde{\mu}_1^{<} = 0.6659$, $\tilde{\mu}_2^{<} = 0.2548$, $\mu_1^{=} = 1.4424$, $\mu_2^{=} = 0.9764$, $\tilde{\mu}_1^{=} = 1.5821$, $\tilde{\mu}_2^{=} = 1.0534$. Since the default boundaries are all close to one we fix the spacial domain limit to $X_1^{Max} = X_2^{Max} = 10$. Like in the one dimensional case the grid is built concentrating points in $\tilde{\mu}$ and taking parameter α proportional to $x^{max} - \tilde{\mu}$ by a factor of $1/10$ for both dimensions. The grid is shown in Figure 4.4.

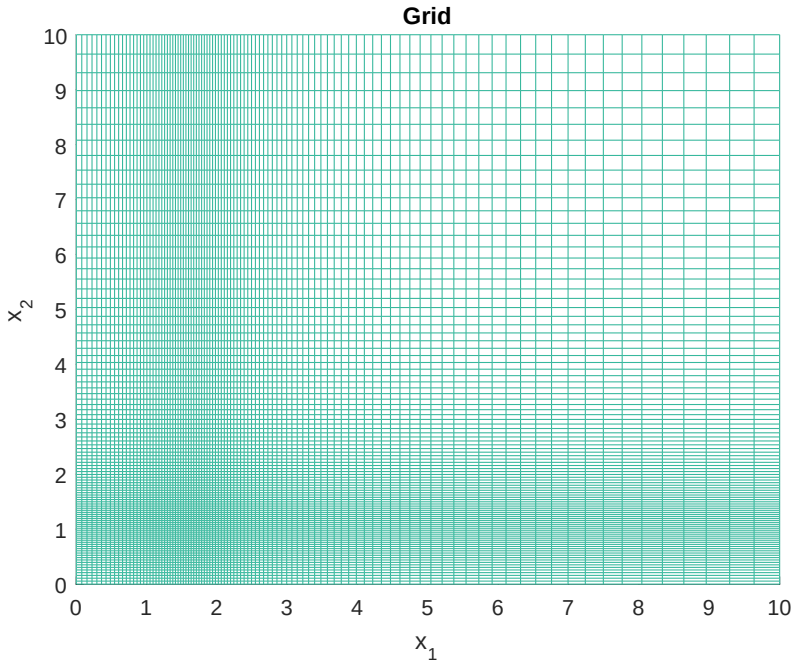

Figure 4.4: Two dimensional grid

In the following we show the results for different payoffs. From a computation the computation time is of order of 6 seconds for the model without jumps and almost 20 for the model with jumps. The main difference between the two times is due to the many numerical integration necessary to compute the correction of the jump operator.

Marginal Survival Probability

We consider the survival probability of the first bank. As mention is Section 4.5.1 the marginal survival probability is a function of both x_1 and x_2 as we can see in Figures 4.6-4.5. In Figure 4.7 we plot the difference between the model with and without jumps at $t = 0$. The jump mainly impacts the area

near default boundaries. As expected, here the probability is quite lower than the case without jump since our model considers only negative jumps.

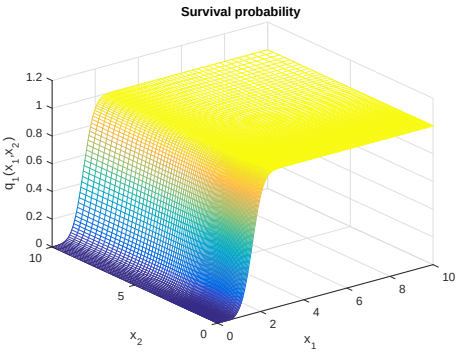

Figure 4.5: Survival probability for model without jump

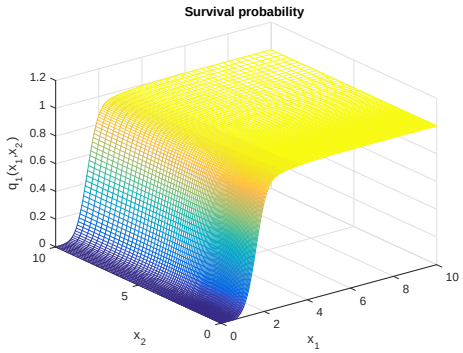

Figure 4.6: Survival probability for model with jump

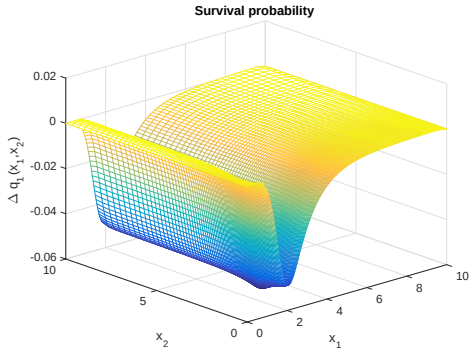

Figure 4.7: Difference of survival probability between model with and without jump

Joint Survival Probability

In Figures 4.8-4.9 we plot the value of joint survival probability at time $t = 0$ for the model without jump and with jump respectively. In Figure 4.10 we plot the difference between the two models. We can observe that the jump impacts the area near default boundaries for both banks. This is reasonable as adding negative shocks primarily influences close to the default barriers where a default cascade is most likely to trigger.

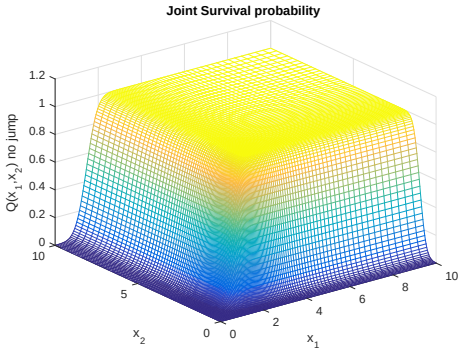

Figure 4.8: Joint survival probability for model without jump

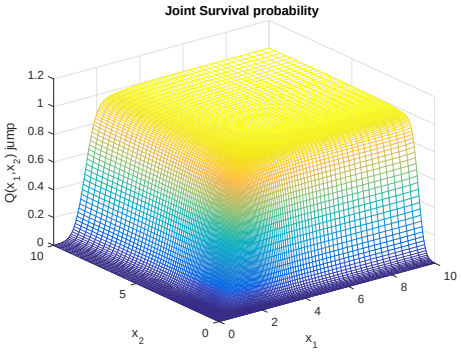

Figure 4.9: Joint survival probability for model with jump

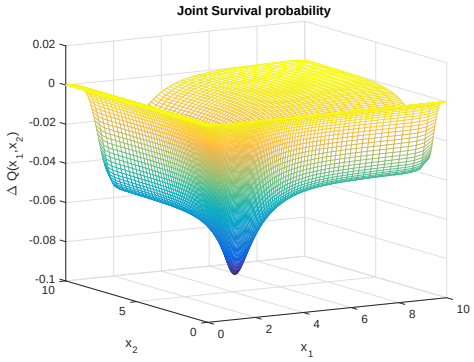

Figure 4.10: Difference of joint survival probability between model with and without jump

Credit Default Swap

Here we plot the value of a Credit Default Swap written on the first bank(RN). As shown in Section 4.5.3 also the second bank influences the CDS value. In Figures 4.11-4.12 we plot the value of CDS at time $t = 0$ for the model without jump and with jump respectively. In Figure 4.13 we can observe that the jumps impact the area near default boundaries of the RN entities. This time the difference is positive because the jumps decrease the survival probability of the banks and so the CDS value increases.

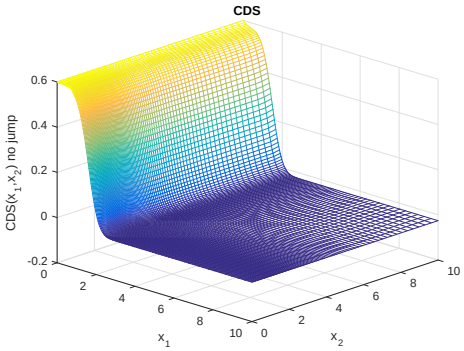

Figure 4.11: Credit default swap price for model without jump

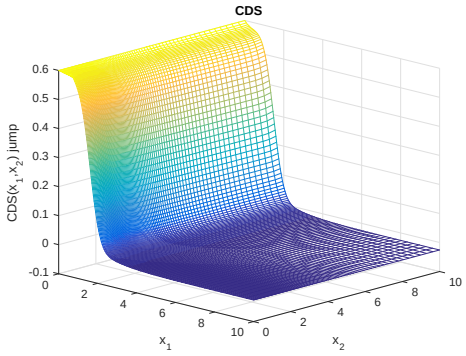

Figure 4.12: Credit default swap price for model with jump

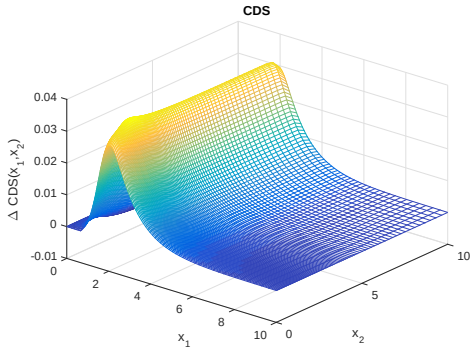

Figure 4.13: Difference of credit default swap price between model with and without jump

Credit Value Adjustments of CDS

As in Section 4.5.5, consider a CDS written on the first bank(RN) and assume the second bank as a Protection Seller(PS). In Figures 4.14-4.15 we plot the adjustments in case of a defaultable counterparty at time $t = 0$ for the model without jump and with jump respectively. We notice that the biggest adjustments are when the PS defaults, accordingly with the definition of CVA. In Figure 4.16 we can observe that the jumps mainly impact the area near default boundaries of the RN entities. The jumps also impact the area near the PS default boundaries. Once again the difference is positive because the jumps increase the default probability of the banks and so the adjustments can only increase.

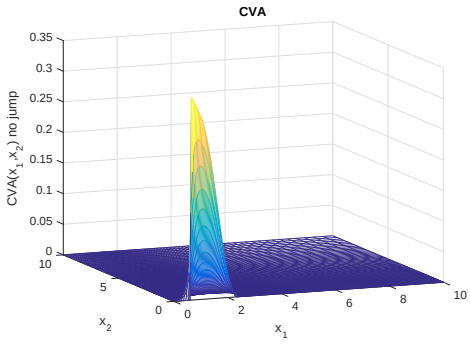

Figure 4.14: CVA of CDS for model without jump

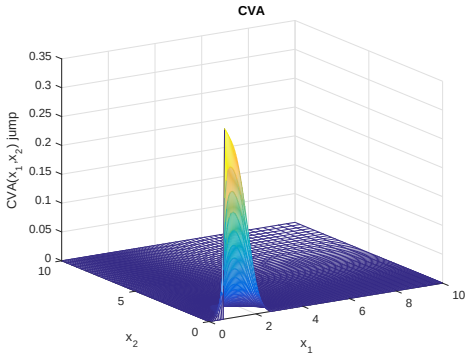

Figure 4.15: CVA of CDS for model with jump

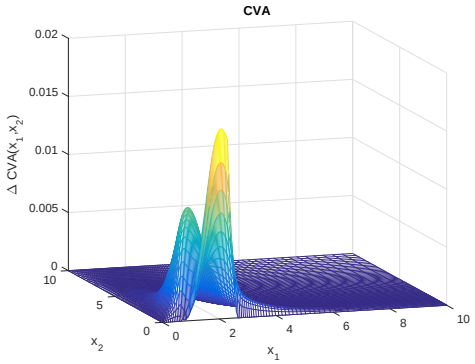

Figure 4.16: Difference of credit value adjustments between model with and without jump

First To Default

As in Section 4.5.4, consider a basket of entities given by first and second bank. In Figures 4.17-4.18 we plot the FTD price at time $t = 0$ for the model without jump and with jump respectively. In Figure 4.19 we can observe that the jump mainly impacts the area near default boundaries of both banks. The difference is positive because the jumps increase the default probability of the banks and so the FTD value can only increase.

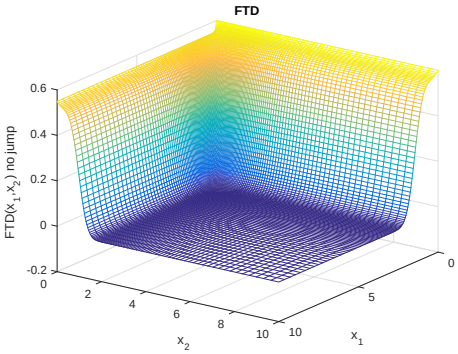

Figure 4.17: First to default price for model without jump

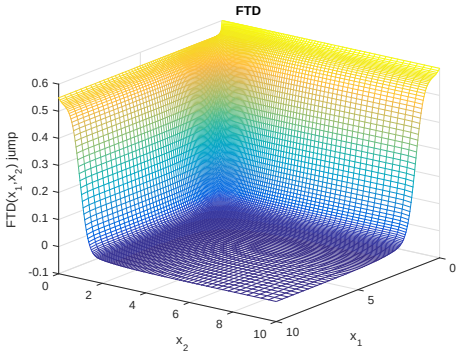

Figure 4.18: First to default price for model with jump

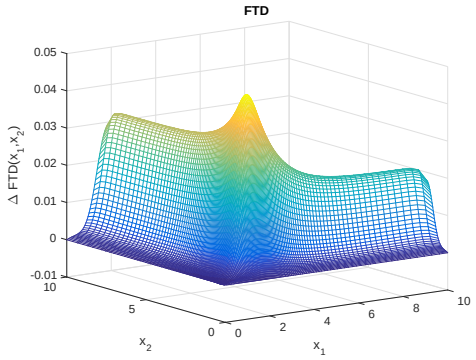

Figure 4.19: Difference of first to default price between model with and without jump

Chapter 5

Improvements

In this section we propose different technical measures in order to improve both efficiency and convergence of the scheme.

5.1 Time discretization

It's well known fact that Crank-Nicholson scheme suffers the dimensionality of the problem (see [7] for details). The reason is that in the one dimensional case the diffusion matrix is typically tridiagonal. When the dimensionality of the problem grows, the diffusion matrix loses its tridiagonal shape and assumes a bad bandwidth structure. In order to solve this problem one idea is to split the problem by dimension. In order to better understand the methodology we first analyse the simpler case with zero boundaries. After spacial discretization of the equations (4.7), the problem is given by

$$\begin{aligned} U'(t) &= AU(t) + b(t), \\ U(0) &= U_0. \end{aligned} \tag{5.1}$$

where A is the approximation of the $\mathcal{L}$ defined in (4.11), and $b(t)$ is 0 for simplicity. In order to split the problem by dimension we consider $A = A_0 + A_1 + A_2$, with

$$\begin{aligned} A_0 &= D_{12} + \lambda_1 J_1 + \lambda_2 J_2 + \lambda_{12} J_{12}, \\ A_1 &= D_1 - \left(\lambda_1 + \frac{\lambda_{12}}{2}\right) I, \\ A_2 &= D_2 - \left(\lambda_2 + \frac{\lambda_{12}}{2}\right) I, \end{aligned}$$

where the matrix D_{12}, D_1, D_2 are defined in Section 4.6.2. Then we apply the Crank-Nicolson scheme. Since both the jump operator and mixed derivative term have bad bandwidth shape we treat them explicitly. Then we get

$$(I - \theta\Delta t A_1 - \theta\Delta t A_2)U^{n+1} = (I + (1-\theta)\Delta t A_1 + (1-\theta)\Delta t A_2 + \Delta t A_0)U^n.$$

Noticing that $A_1A_2 \sim o(\Delta t^2)$ and $U^{n+1} - U^n \sim o(\Delta t)$, we can add $\theta A_1 A_2 U^{n+1}$ to both terms. Thus after some simplification, we get

$$\begin{aligned}(I - \theta\Delta t A_1)(I - \theta\Delta t A_2)U^{n+1} = (I + \Delta t A_0 + (1-\theta)\Delta t A_1 + \Delta t A_2 + \Delta t A_0)U^n, \\ - (I - \theta A_1)\theta A_2 U^n. \end{aligned} \tag{5.2}$$

We start splitting by dimension by firstly taking care of the explicit term. Introducing

$$Y_0 = U^n + \Delta t(A_0 + A_1 + A_2)U^n,$$

we can rewrite the equation (5.2) as

$$(I - \theta\Delta t A_1)(I - \theta\Delta t A_2)U^{n+1} = Y_0 - \theta\Delta t A_1 U^n - (I - \theta A_1)\theta A_2 U^n,$$

The we can finally split the problem over both dimensions introducing $Y_1 = (I - \theta A_2)U^{n+1} - \theta A_2 u^n$, $Y_2 = U^{n+1}$.

$$\begin{aligned}(I - \theta\Delta t A_1)Y_1 &= Y_0 - \theta A_1 U^n, \\ (I - \theta\Delta t A_2)Y_2 &= Y_1 - \theta A_2 U^n.\end{aligned}$$

Summing up the ADI scheme solves the problem in (5.1) solving at each time step n the system

$$\begin{aligned}Y_0 &= U^n + \Delta t(A_0 + A_1 + A_2)U^n, \\ (I - \theta\Delta t A_1)Y_1 &= Y_0 - \theta A_1 U^n, \\ (I - \theta\Delta t A_2)Y_2 &= Y_1 - \theta A_2 U^n, \\ U^{n+1} &= Y_2.\end{aligned}$$

With this method we need to invert only matrices A_1 and A_2 that are tridiagonal. In order to solve our original problem we propose a modification of the previous scheme. More precisely we will use a Hundsdorfer-Verwer scheme :

$$Y_0 = U_{n-1} + \Delta t\big((A_0 + A_1 + A_2)U_{n-1} + b_0(t_{n-1}) + b_1(t_{n-1}) + b_2(t_{n-1})\big), \tag{5.3}$$

$$Y_1 = Y_0 + \theta\Delta t(A_1 Y_1 + b_1(t_n) - A_1 U_{n-1} - b_1(t_{n-1})), \tag{5.4}$$

$$Y_2 = Y_1 + \theta\Delta t(A_2 Y_2 + b_2(t_n) - A_2 U_{n-1} - b_2(t_{n-1})), \tag{5.5}$$

$$\tilde{Y}_0 = Y_0 + \sigma\Delta t\big((A_0 + A_1 + A_2)Y_2 - (A_0 + A_1 + A_2)U_{n-1} + B\big), \tag{5.6}$$

$$\tilde{Y}_1 = \tilde{Y}_0 + \theta\Delta t(A_1\tilde{Y}_1 - A_1 Y_2), \tag{5.7}$$

$$\tilde{Y}_2 = \tilde{Y}_1 + \theta\Delta t(A_2\tilde{Y}_2 - A_2 Y_2), \tag{5.8}$$

$$U_n = \tilde{Y}_2, \tag{5.9}$$

where $B = b_0(t_n) + b_1(t_n) + b_2(t_n) - b_0(t_{n-1}) - b_1(t_{n-1}) - b_2(t_{n-1})$.
The main idea of this scheme is the same of the basic ADI scheme. In the

first step both jumps and mixed derivatives are treated explicitly. In second step the problem is solved in x_1 direction considering the second variable x_2 as a constant. In this way the problem can be efficiently solved thanks to the band width shape of matrix A_1. In the third step the problem is solved in x_2 direction. Finally the procedure is repeated in order to stabilize the scheme.

Border Condition

In order to impose the border condition we solve the step backward in order to find the value of Y_0 such that the solution on the boundaries matches the border condition. Imposing border condition we get:

$$\tilde{Y}_2^b = U_n^b.$$

So from (5.8) we obtain

$$\tilde{Y}_1^b = \tilde{Y}_2^b - \theta\Delta t(A_2\tilde{Y}_2^b - A_2 Y_2^b),$$

but from (5.7) we have

$$\tilde{Y}_1^b = \tilde{Y}_0^b + \theta\Delta t(A_1\tilde{Y}_1^b - A_1 Y_2^b).$$

Solving respect to $\tilde{Y}_0^b$ we find $\tilde{Y}_0^b = f(Y_2^b)$ then we can impose (5.5) on the border and find Y_2^b. Repeating this procedure we are able to find the Y_0^b such that the border conditions are satisfied.

5.1.1 Stability analysis

In this section, following [14] we prove that the Hundsdorfer-Verwer scheme is von Neumann stable. For simplicity we consider (4.24) without default boundaries. We also consider a uniform grid. So taking $F_j(t,x) = \tilde{A}_j x$ the HV scheme becomes

$$\begin{cases} Y_0 = U_{n-1} + \Delta t\tilde{A}U_{n-1}, \\ Y_j = Y_{j-1} + \theta\Delta t(\tilde{A}_j Y_j - \tilde{A}_j U_{n-1}), \quad j = 1,2, \\ \tilde{Y}_0 = Y_0 + \sigma\Delta t\big(\tilde{A}Y_2 - \tilde{A}U_{n-1}\big), \\ \tilde{Y}_j = \tilde{Y}_{j-1} + \theta\Delta t(\tilde{A}_j\tilde{Y}_j - \tilde{A}_j Y_2), \qquad j = 1,2, \\ U_n = \tilde{Y}_2. \end{cases} \tag{5.10}$$

For convenience we denote the above scheme as an application of F, in this way we have $U_n = FU_{n-1}$. We also assume that diffusion and jump operators are discretized on an infinite uniform mesh $\{(j_1h_1, j_2h_2), (j_1,j_2) \in \mathbb{Z}^2\}$ and so described by infinite matrix. In order to prove stability we will prove that all the eigenvalues of the operator F have module bounded by 1 plus an $\mathcal{O}(\Delta t)$ where the corresponding eigenfunctions are given by $e^{i\phi_1 j_1}e^{i\phi_2 j_2}$ with

ϕ_1,ϕ_2 the wave numbers and j_1,j_2 the grid coordinates. We start from [10] and we extend the theory to the case with jumps. Following the notation of [10] we start defining variables for diffusive case $A = A_0 + A_1 + A_2$ with $A_0 = D_{12}$, $A_1 = D_1$, $A_2 = D_2$ and μ_0, μ_1, μ_2 the corresponding eigenvalues of the corresponding matrices. Then we define the scaled eigenvalues $z_0 = \mu_0\Delta t$, $z_1 = \mu_1\Delta t$, $z_2 = \mu_2\Delta t$. For the jump case we introduce the $\sim$ -variables. We define

$$\begin{aligned}
\tilde{A}_0 &= D_{12} + \lambda_1 J_1 + \lambda_2 J_2 + \lambda_{12} J_{12}, \\
\tilde{A}_1 &= D_1 - \left(\lambda_1 + \frac{\lambda_{12}}{2}\right) I, \\
\tilde{A}_2 &= D_2 - \left(\lambda_2 + \frac{\lambda_{12}}{2}\right) I.
\end{aligned}$$

Then the eigenvalues $\tilde{\mu}_j$ of $\tilde{A}_j$ are given by

$$\tilde{\mu}_0 = \mu_0 + \lambda_1\omega_1 + \lambda_2\omega_2 + \lambda_{12}\omega_{12}, \tag{5.11}$$

$$\tilde{\mu}_1 = \mu_1 - \left(\lambda_1 + \frac{\lambda_{12}}{2}\right), \tag{5.12}$$

$$\tilde{\mu}_2 = \mu_2 - \left(\lambda_2 + \frac{\lambda_{12}}{2}\right), \tag{5.13}$$

where $\omega_1 = \omega_1(\varsigma_1, h_1)$, $\omega_2 = \omega_2(\varsigma_2, h_2)$, $\omega_{12} = \omega_1\omega_2$ are the eigenvalues of J_1, J_2, J_{12}. We then define scaled eigenvalues for the jump part $s_1 = \omega_1\Delta t$, $s_2 = \omega_2\Delta t$, $s_{12} = \omega_{12}\Delta t$ and $s_0 = \lambda_1 s_1 + \lambda_2 s_2 + \lambda_{12} s_{12}$. Multiplying (5.11)-(5.13) by Δt we obtain

$$\begin{aligned}
\tilde{z}_0 &= z_0 + s_0, \\
\tilde{z}_1 &= z_1 - \left(\lambda_1 + \frac{\lambda_{12}}{2}\right)\Delta t, \\
\tilde{z}_2 &= z_2 - \left(\lambda_2 + \frac{\lambda_{12}}{2}\right)\Delta t,
\end{aligned}$$

We are finally ready to show some results.

Theorem 3 ([10], Theorem 3.2)**.** . *Assume* $\Re(z_1) \leq 0$, $\Re(z_2) \leq 0$, $|z_0| \leq 2\sqrt{\Re(z_1)\Re(z_2)}$, *where* z_0,z_1 *and* z_2 *are the eigenvalues of* A_0, A_1 *and* A_2, *and*

$$\frac{1}{2} \leq \sigma \leq \left(1 + \frac{\sqrt{2}}{2}\right)\theta.$$

Then, for the eigenvalues $T(z_0, z_1, z_2)$ *of* F *holds true*

$$|T(z_0, z_1, z_2)| \leq 1,$$

and the Hundsdorfer-Verwer scheme 5.10 is stable.

Lemma 4 ([14], Lemma 1)**.** . *The scaled eigenvalues of A_0, A_1, A_2, J_1, J_2, J_{12} can be expressed as*

$$\begin{aligned}
z_0 &= -\rho b(\sin(\phi_1)\sin(\phi_2)),\\
z_1 &= -a_1(1-\cos(\phi_1)) + i\xi_1 q_1 \sin(\phi_1),\\
z_2 &= -a_2(1-\cos(\phi_2)) + i\xi_2 q_2 \sin(\phi_2),\\
s_1 &= \Delta t\theta_1 h_1\left(\frac{1}{2} + \frac{\exp -h_1\theta_1 + i\phi_1}{1-\exp -h_1\theta_1 + i\phi_1}\right),\\
s_2 &= \Delta t\theta_2 h_2\left(\frac{1}{2} + \frac{\exp -h_2\theta_2 + i\phi_2}{1-\exp -h_2\theta_2 + i\phi_2}\right),\\
s_{12} &= \frac{s_1 s_2}{\Delta t},
\end{aligned}$$

where

$$q_1 = \frac{\Delta t}{h_1}, \quad q_2 = \frac{\Delta t}{h_2}, \quad a_1 = \frac{\Delta t}{h_1^2}, \quad a_2 = \frac{\Delta t}{h_2^2}, \quad b = \frac{\Delta t}{h_1 h_2},$$

and $\phi_j \in [0, 2\pi]$ for $j = 1, 2$. Moreover,

$$|z_0| \le 2\sqrt{\Re(z_1)\Re(z_2)}.$$

Theorem 5 ([14], Theorem 2)**.** *Consider $\frac{1}{2} \le \sigma \le \left(1 + \frac{\sqrt{2}}{2}\theta\right)$. Then exist $c > 0$, independent of $\Delta t \le 1$, h_1 and h_2, such that*
1.

$$|T(\tilde{z}_0, \tilde{z}_1, \tilde{z}_2)| \le 1 + c\Delta t, \qquad \forall \phi_1, \phi_2 \in [0, 2\pi],$$

i.e., the scheme is von Neumann stable;
2.

$$|U_n|_2 \le e^{cn\Delta t}|U_0|_2, \qquad \forall n \ge 0,$$

for $|U_n|_2 = h_1 h_2\left(\sum_{j_1,j_2=-\infty}^{\infty} |U_n(j_1, j_2)|^2\right)^{\frac{1}{2}}$, i.e., the scheme is l_2 stable.

Proof. From [10] we have that in the diffusive case

$$|T(z_0, \tilde{z}_1, \tilde{z}_2)| = \left|1 + 2\frac{z_0 + \tilde{z}}{p} - \frac{z_0 + \tilde{z}}{p^2} + \sigma\frac{(z_0 + \tilde{z})^2}{p^2}\right| \le 1,$$

where $p = (1-\theta\tilde{z}_1)(1-\theta\tilde{z}_2)$ and $\tilde{z} = \tilde{z}_1 + \tilde{z}_2$. Since λ_1, λ_2, λ_{12} are positive we have that $\Re(\tilde{z}_1) \le \Re(z_1) \le 0$ and the same holds for z_2. Then the theorem still holds since

$$|z_0| \le 2\sqrt{\Re(z_1)\Re(z_2)} \le 2\sqrt{\Re(\tilde{z}_1)\Re(\tilde{z}_2)}.$$

Notice that for $\tilde{z}_0$ such estimation is not longer available. Then with simple substitution we obtain that

$$T(\tilde{z}_0, \tilde{z}_1, \tilde{z}_2) = T(z_0, \tilde{z}_1, \tilde{z}_2) + 2\frac{s_0}{p} - \frac{s_0^2}{p^2} + \sigma\frac{2s_0(z_0 + \tilde{z})^2 + s_0^2}{p^2}.$$

Observe that $|s_0| \leq c_0\Delta t$ for a constant c_0 independent of Δt, h_1, h_2, ϕ_1, ϕ_2, for example $c_0 = 2\lambda_1+2\lambda_2+4\lambda_{12}$ works. Notice also that since $|p| \geq 1$, $\frac{|z_0+\tilde{z}|}{|p|} \leq c_1$, for c_1 constant. Finally we have the following estimation

$$\left| 2\frac{s_0}{p} - \frac{s_0^2}{p^2} + \sigma\frac{2s_0(z_0+\tilde{z})^2 + s_0^2}{p^2} \right| \leq c\Delta t,$$

for any $c \geq (3 + 2\sigma c_1 + c_0\sigma)c_0$. The first statement is proved. In order to prove the second statement we define the Fourier transform $\hat{U}$ of U. Then by Parceval identity

$$\begin{aligned}
|U_n|_2^2 &= \frac{1}{4\pi}|\hat{U}_n|_2^2 \\
&= \frac{1}{4\pi}\frac{1}{h_1^2h_2^2}\int_{-\pi}^{\pi}|\hat{U}_n(\phi_1,\phi_2)|^2 d\phi_1 d\phi_2 \\
&\leq \frac{1}{4\pi}\frac{1}{h_1^2h_2^2}\int_{-\pi}^{\pi}(1+c\Delta t)^{2n}|\hat{U}_0(\phi_1,\phi_2)|^2 d\phi_1 d\phi_2 \\
&\leq e^{2cn\Delta t}\frac{1}{4\pi}\frac{1}{h_1^2h_2^2}\int_{-\pi}^{\pi}|\hat{U}_0(\phi_1,\phi_2)|^2 d\phi_1 d\phi_2 \\
&= e^{2cn\Delta t}|U_0|^2.
\end{aligned}$$

□

This theorem proves that the scheme is von Neumann stable and l_2 stable. l_2 stability and second order consistency implies l_2-convergence of second order for all solution which are sufficiently smooth that the truncation error is defined and bounded. In our problem we have discontinuous initial condition so the results are no true anymore. Since our discontinuous function is the step function who lies in l_2-closure of smooth functions, convergence is guaranteed, but not of second order. We will show how to restore second order convergence in the next section.

5.2 Convergence Analysis

Authors of [21] show that using central finite difference scheme with discontinuous initial condition reduces the order to converge to one. In order to restore the order of converge we then smooth the initial condition by method of local averaging. This method, instead of directly taking the solution, take the approximation

$$\phi(x_1^i, x_2^j) \approx \frac{1}{h_1h_2}\int_{x_2^j-h_2/2}^{x_2^j+h_2/2}\int_{x_1^i-h_1/2}^{x_1^i+h_1/2}\phi(\xi_1,\xi_2)d\xi_1 d\xi_2.$$

This method uses as value in (x_1^i, x_2^j) the average solution in the symmetric neighbour $I_{i,j}$ of size $h_1 \times h_2$ centred in (x_1^i, x_2^j). In our case, since we have only step functions, this method attaches to each node the fraction of area where the payoff is 1 in the neighbour $I_{i,j}$. In order to compute the rate of convergence of a method of order $p \geq 1$ we estimate the error as

$$|q_1^{nX}(x_1, x_2) - q_1(x_1, x_2)|_2 \approx \frac{1}{2^p - 1} |q_1^{nX}(x_1, x_2) - q_1^{nX/2}(x_1, x - 2)|_2,$$

where q_1 is the exact solution and q_1^{nX} is the solution with nX spacial points. All test are made with the same parameters of Tables 4.1-4.2. We also consider 1000 time steps. We show the improvements of convergence in the case of survival marginal probability on the first bank in Figure 5.1. Clearly without smoothing procedure the convergence is of order one, while smoothing the data we restore convergence of order two.

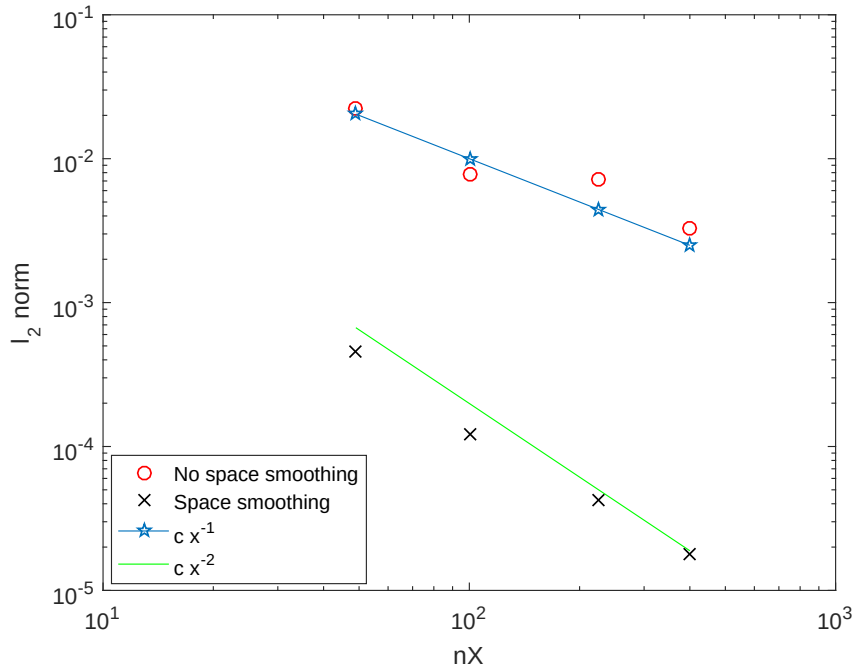

Figure 5.1: Rate of convergence

From a computation perspective the time discretization of Section 5.1 allows us to solve the problem in about for 3 seconds for each payoff respect to the 11 seconds in the case of Crank-Nicholson.

Chapter 6

Conclusion

In this thesis we present a jump diffusion model which combines both for mutual liabilities and correlated processes.
We discuss how these liabilities affect joint and marginal survival probability and by inheritance all the credit products. This allows us to model the propagation of default by contagion. To make the analysis tractable we use a dynamics driven by Lévy processes in which each driving Poisson process has a jump variable associated with it. This allows greater freedom to model correlation as we can vary both the jump intensity for the common jump and the jump mean and variance.
However this model does not allow an explicit option pricing formula. Therefore we develop a efficient numerical scheme by splitting into dimensions and jumps simultaneously. Then we price the main credit products and study the effect of both jumps and mutual liabilities on the credit products.
Pricing of the credit option requires inputting five extra jump parameters respect to the model without jumps. These parameters would have to be estimated in practice before the model could be used to price any options and there may be a number of choices of parameters that could arguably model the assets equally well. This may be a downside to the model and the reason why the Black-Scholes model is favoured by practitioners due the small number of input parameters needed to gain an accurate option price. However to correctly modelling the effects of mutual liabilities Black-Scholes model are not enough and therefore we need to consider more complex model like this one.

References

[1] T. R. Bielecki and M. Rutkowski. *Credit risk: modeling, valuation and hedging.* Science & Business Media., 2013.

[2] F. Black and J.C. Cox. Valuing corporate securities: some effect of bond indenture provisions. *Journal of Finance*, 31(2):351–367, 1976.

[3] J. C. Butcher. Numerical methods for ordinary differential equations. *John Wiley & Sons*, 2008.

[4] R. Cont and P. Tankov. *Financial Modelling With Jump Process.* Chapman & Hall, 2003.

[5] A David and A. Lehar. Why are banks highly interconnected? 2014.

[6] L. Eisenberg and T. H. Noe. Systemic risk in financial systems. *Management Science*, 47(2):236–249, 2001.

[7] W. Hundsdorfer and J. G. Verwer. *Numerical solution of time dependent advection-diffusion-reaction equations*, volume 33. Springer Science & Business Media, 2013.

[8] T. R. Hurd. *Contagion!Systemic Risk in Financial Networks.* Springer, 2016.

[9] K. In't Hout and S. Foulon. Adi finite difference schemes for option pricing in the heston model with correlation. *International Journal of Numerical Analysis and Modeling*, 7(2):303–320, 2010.

[10] K. In't Hout and B. Welfert. Stability of adi scheme applied to convection-diffusion equations with mixed derivative terms. *Applied Numerical Mathematics*, 57(1):19–35, 2007.

[11] A. Itkin and P. Carr. Jump without tears. a new splitting technology for barrier options. *International Journal of Numerical Analysis and Modeling*, 8(4):667–704, 2011.

[12] A. Itkin and A. Lipton. Efficient solution of structural default models with correlated jumps and mutual obligations. *International Journal of Numerical Analysis and Modeling*, 92(12):2380–2405, 2015.

[13] A. Itkin and A. Lipton. Structural default models with mutual obligations. *Review of Derivatives Research*, 2016.

[14] V. Kaushansky, A. Lipton, and C. Reisingers. Numerical analysis of an extended structural default model with mutual liabilities and jump risk. *Journal of Computational Science*, 2016.

[15] T. Klunge. Pricing derivatives in stochastic volatility models using the difference method. Master's thesis, TU Chemnitz, 2002.

[16] A. Lipton. Modern monetary circuit theory, stability of interconnected banking network, and balance sheet optimization for individual banks. *International Journal of Theoretical and Applied Finance*, 19(06), 2016.

[17] A. Lipton and I. Savescu. Pricing credit default swaps with bilateral value adjustments. *Quantitative Finance*, 14(1):171–188, 2014.

[18] A. Lipton and A. Sepp. *Credit value adjustment in the extended structural default model In Lipton, A. and Rennie, A., editors, The Oxford Handbook of Credit Derivatives,*. Oxford University Press, 2013.

[19] A. W. Marshall and I. Olkin. A multivariate exponential distribution. *Journal of the American Statistical Association*, 62(317):30–44, 1967.

[20] R.C. Merton. On the pricing of corporate debt: the risk structure of interest rates. *Journal of Finance*, 29:449–470, 1974.

[21] D. M. Pooley, K. R. Vetzal, and P. A. Forsyth. Convergence remedies for non-smooth payoffs in option pricing. *Journal of Computational Finance*, 6(4):25–40, 2003.

[22] C. Resinger and A. Whitley. The impact of natural time change on the convergence of the crank-nicolson scheme. *IMA J.Numer.Anal.*, 2013.

[23] D. Tavella and C. Randall. Pricing financial instruments. the finite-difference method. *John Wiley & Sons, Inc.*, 2000.

Appendix A

Jump Operator

In this appendix we will show that the integral operator can be computed as in (4.12)-(4.14).
Our problem is stated to the domain $\mathbb{R}^2_+ \times [t,T]$, i.e,

$$\begin{cases} V_t(t,x) + \mathcal{L}V(t,x) = \chi(t,x), & x \in \mathbb{R}^2_+, \\ V(t,x) = z(t,x), & x \notin \mathbb{R}^2_+. \end{cases}$$

where $\mathcal{L} = \mathcal{D} + \mathcal{J}$ and the jump operator is defined as $\mathcal{J} = \lambda_1\mathcal{J}_1 + \lambda_2\mathcal{J}_2 + \lambda_{12}\mathcal{J}_{12}$ with

$$\begin{aligned} \mathcal{J}_1 V(x_1,x_2) &= \int_{-\infty}^{0} V(x_1+j,x_2)\tilde{\omega}(j)dj, \\ \mathcal{J}_2 V(x_1,x_2) &= \int_{-\infty}^{0} V(x_1,x_2+j)\tilde{\omega}(j)dj, \\ \mathcal{J}_{12} V(x_1,x_2) &= \int_{-\infty}^{0}\int_{-\infty}^{0} V(x_1+j,x_2+k)\tilde{\omega}(j)\tilde{\omega}(k)djdk, \end{aligned}$$

and $\tilde{\omega}(j)$ is the probability density given by (4.2).
We focus only on the operator $\mathcal{J}_1$, the others can be obtain in similar way.
Substituting (4.2) we get

$$\begin{aligned} \int_{-\infty}^{0} V(x_1+j,x_2)\tilde{\omega}(j)dj &= \int_{-x_1}^{0} V(x_1+j,x_2)\tilde{\omega}(j)dj + \int_{-\infty}^{-x_1} z_{-,+}(x_1+j,x_2)\tilde{\omega}(j)dj \\ &= \int_{-x_1}^{0} V(x_1+j,x_2)\theta_1 e^{\theta_1 j}dj + \int_{-\infty}^{-x_1} z_{-,+}(x_1+j,x_2)\theta_1 e^{\theta_1 j}dj \\ &= \int_{0}^{x_1} V(x_1-u,x_2)\theta_1 e^{-\theta_1 u}du + \int_{x_1}^{+\infty} z_{-,+}(x_1-u,x_2)\theta_1 e^{-\theta_1 u}du. \end{aligned}$$

Thus the jump operator can be seen as

$$\mathcal{J}_1 V(x_1,x_2) = \hat{\mathcal{J}}_1 V(x_1,x_2) + Z_{-,+}(x_1,x_2).$$

Similar consideration hold for both $\mathcal{J}_2$ and $\mathcal{J}_{12}$. With the same computations we obtain

$$\begin{aligned}\mathcal{J}_2 V(x_1,x_2) &= \int_0^{x_2} V(x_1,x_2-u)\theta_2 e^{-\theta_2 u}du + \int_{x_2}^{+\infty} z_{+,-}(x_1,x_2-u)\theta_2 e^{-\theta_2 u}du \\ &= \hat{\mathcal{J}}_2 V(x_1,x_2) + Z_{+,-}(x_1,x_2),\end{aligned}$$

and, paying attention to handling the double integral of $\mathcal{J}_{12}$, we get

$$\begin{aligned}\mathcal{J}_{12} V(x_1,x_2) &= \int_0^{x_1}\int_0^{x_2} V(x_1-v,x_2-u)\theta_1 e^{-\theta_1 v}\theta_2 e^{-\theta_2 u}dudv \\ &+ \int_{x_1}^{+\infty}\int_0^{x_2} z_{-,+}(x_1-v,x_2-u)\theta_1 e^{-\theta_1 v}\theta_2 e^{-\theta_2 u}dudv \\ &+ \int_{x_1}^{+\infty}\int_{x_2}^{+\infty} z_{-,-}(x_1-v,x_2-u)\theta_1 e^{-\theta_1 v}\theta_2 e^{-\theta_2 u}dudv \\ &+ \int_0^{x_1}\int_{x_2}^{\infty} z_{+,-}(x_1-v,x_2-u)\theta_1 e^{-\theta_1 v}\theta_2 e^{-\theta_2 u}dudv \\ &= \hat{\mathcal{J}}_{12} V(x_1,x_2) + Z_{-,+}(x_1,x_2) + Z_{-,-}(x_1,x_2) + Z_{+,-}(x_1,x_2).\end{aligned}$$

The value $z_{-,+}$ is the value of the solution when the first bank log asset value goes below under 0(-) and the second bank log asset stays above 0(+). Thus in our framework this value is given the boundary condition $Q(t,0,x_2)$. Similar arguments hold for $z_{+,-}$, $z_{+,+}$.
The $Z_{-,+}$, $Z_{-,+}$, $Z_{-,+}$ values are a deterministic corrections, therefore in the resolution of the PIDE we raise them from the jump operator and we incorporate them into the known term, i.e. with an abuse of notation we consider

$$\begin{aligned}\mathcal{J}_1 V(x_1,x_2) &= \hat{\mathcal{J}}_1 V(x_1,x_2), \\ \mathcal{J}_2 V(x_1,x_2) &= \hat{\mathcal{J}}_2 V(x_1,x_2), \\ \mathcal{J}_{12} V(x_1,x_2) &= \hat{\mathcal{J}}_{12} V(x_1,x_2), \\ \hat{\chi}(t,x) &= \chi(t,x) - \lambda_1 Z_{-,+}(x_1,x_2) - \lambda_2 Z_{+,-}(x_1,x_2) \\ &\quad - \lambda_{12}(Z_{-,+}(x_1,x_2) + Z_{-,-}(x_1,x_2) + Z_{+,-}(x_1,x_2)).\end{aligned}$$

In order to use simpler notation in the thesis we omit the $\hat{\cdot}$ notation.

Appendix B

Asset Dynamics

We apply the Itô formula given in Theorem 1. Given the following asset's dynamics

$$\frac{dA_i}{A_i} = (\mu - \kappa_i\lambda_i)dt + \sigma_i dW_i(t) + (e^{J_i} - 1)dN_i(t).$$

Let $X_t = f(A_i(t)) = ln\left(\frac{A_i(t)}{\Lambda_i^<}\right)$. Then the first and second derivative of f are given by :

$$f_A = \frac{1}{\Lambda_i^<}\frac{1}{\frac{A_i}{\Lambda_i^<}} = \frac{1}{A_i},$$
$$f_{AA} = -\frac{1}{A_i^2}.$$

Appling Itô we have, taking for simplicity $\mu = 0$,

$$\begin{aligned} dX_i(t) &= A_i(-\kappa_i\lambda_i)f_A dt + A_i\sigma_i f_A dW_i(t) + \frac{1}{2}A_i^2\sigma_i^2 f_{AA}dt + \big(f(X_t) - f(X_{t^-})\big)dN_i(t) \\ &= -\big(\kappa_i\lambda_i + \frac{1}{2}\sigma_i^2\big)dt + \sigma_i dW_i(t) + \left(ln\big(\frac{A_i + A_i(e^{J_i} - 1)}{\Lambda_i}\big) - ln\big(\frac{A_i}{\Lambda_i}\big)\right)dN_i(t) \\ &= -\big(\kappa_i\lambda_i + \frac{1}{2}\sigma_i^2\big)dt + \sigma_i dW_i(t) + J_i dN_i(t) \\ &= \xi_i dt + \sigma_i dW_i(t) + J_i dN_i(t), \end{aligned}$$

where $\xi_i = -\big(\kappa_i\lambda_i + \frac{1}{2}\sigma_i^2\big)$.

Appendix C

Feynman-Kac formula

Theorem 6. *Given the following PIDE with boundary conditions :*

$$\begin{aligned}
&\frac{\partial V}{\partial t} + \mathcal{L}V = \chi(t,x), \\
&V(t,0,x_2) = \phi_{2,0}(t,x_2) \qquad V(t,x_1,x_2), \xrightarrow[x_1\to+\infty]{} \phi_{2,\infty}(t,x_2), \\
&V(t,x_1,0) = \phi_{1,0}(t,x_1) \qquad V(t,x_1,x_2), \xrightarrow[x_2\to+\infty]{} \phi_{1,\infty}(t,x_1), \\
&V(T,x) = \psi(x).
\end{aligned} \tag{C.1}$$

where

$$\begin{aligned}
\mathcal{L}f =& \frac{1}{2}\sigma_1^2 f_{x_1x_1} + \rho\sigma_1\sigma_2 f_{x_1x_2} + \frac{1}{2}\sigma_2^2 f_{x_2x_2} + \xi_1 f_{x_1} + \xi_2 f_{x_2} \\
&+ \lambda_1\mathcal{J}_1 f + \lambda_2\mathcal{J}_2 f + \lambda_{12}\mathcal{J}_{12} f - \upsilon f \\
&= \Delta_\rho f + \xi\cdot\nabla f + \mathcal{J} f - \upsilon f,
\end{aligned}$$

with $\upsilon = \lambda_1 + \lambda_2 + \lambda_{12}$ *and*

$$\mathcal{J}_1 f(x) = \theta_1 \int_0^{x_1} f(x_1 - u, x_2) e^{-\theta_1 u} du, \tag{C.2}$$

$$\mathcal{J}_2 f(x) = \theta_2 \int_0^{x_2} f(x_1, x_2 - u) e^{-\theta_2 u} du, \tag{C.3}$$

$$\mathcal{J}_{12} f(x) = \theta_1\theta_2 \int_0^{x_1}\int_0^{x_2} f(x_1 - u, x_2 - \upsilon) e^{-\theta_1 u} e^{-\theta_2 \upsilon} d\upsilon, \tag{C.4}$$

and $\phi_{i,0}, \phi_{i,\infty}$ *are given function.*
Then, the solution $V(x,t)$ *is given by*

$$\begin{aligned}
V(x,t) = \mathbb{E}\Bigg[&\psi(X_T)\cdot\mathbf{1}_{\{\tau\geq T\}} - \int_t^T \chi(s,X_s)\cdot\mathbf{1}_{\{\tau>s\}} ds + \\
&+ \phi_{1,0}(\tau_1, X_2(\tau_1))\cdot\mathbf{1}_{\{\tau_1<T\}} + \phi_{2,0}(\tau_2, X_1(\tau_2))\cdot\mathbf{1}_{\{\tau_2<T\}} | X(t) = x\Bigg].
\end{aligned}$$

where $dX_i(t) = \xi_i dt + \sigma_i dW_i(t) + J_i dN_i(t)$ *with* $i = 1, 2$, *and* $\tau_i = \inf\{t | X_i(t) < 0\}$, $\tau = \min(\tau_1, \tau_2)$.

Proof. We are in case of PIDE with boundary condition. Our domain is given by $D = [0, \infty]^2 \times [0, T]$.
Inside D the following holds true.
Let's define $Y(X_s, s) = V(X_s, s)$, in the following we omit the arguments when there is no misunderstanding. Using Itô lemma we have

$$\begin{aligned} dY_s = &(V_t + \frac{1}{2}\Delta_\rho V + \xi \cdot \nabla V)ds + \nabla V dW_s \\ &+ (V(s, X^1_s, X^2_{s^-}) - V(s, X^1_{s^-}, X^2_{s^-}))dN_{\{1\}}(s) \\ &+ (V(s, X^1_{s^-}, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))dN_{\{2\}}(s) \\ &+ (V(s, X^1_s, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))dN_{\{12\}}(s), \end{aligned} \tag{C.5}$$

where we use the fact that $dN_1(s) = dN_{\{1\}}(s) + dN_{\{12\}}(s)$, $dN_2(s) = dN_{\{2\}}(s) + dN_{\{12\}}(s)$. Now since we want to use martingale arguments we re-write under the compensated Poisson process. Our Poisson process are correlated as in [19], so we use $N_{\{1\}}$, $N_{\{2\}}$ and $N_{\{12\}}$ independent Poisson processes of intensities λ_1, λ_2 and λ_{12} respectively. Then the compensated Poisson process are

$$\tilde{N}_{\{1\}}(t) = N_{\{1\}}(t) - \lambda_1 t, \quad \tilde{N}_{\{2\}}(t) = N_{\{2\}}(t) - \lambda_2 t, \quad \tilde{N}_{\{12\}}(t) = N_{\{12\}}(t) - \lambda_{12} t.$$

Substituting in C.5 we obtain

$$\begin{aligned} dY_s = &(V_t + \Delta_\rho V + \xi \cdot \nabla V)ds + \nabla V dW_s \\ &+ (V(s, X^1_s, X^2_{s^-}) - V(s, X^1_{s^-}, X^2_{s^-}))(d\tilde{N}_{\{1\}}(s) + \lambda_1 ds) \\ &+ (V(s, X^1_{s^-}, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))(d\tilde{N}_{\{2\}}(s) + \lambda_2 ds) \\ &+ (V(s, X^1_s, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))(d\tilde{N}_{\{12\}}(s) + \lambda_{12} ds) \\ = &\Big[V_t + \Delta_\rho V + \xi \cdot \nabla V + \lambda_1(V(s, X^1_s, X^2_{s^-}) - V(s, X^1_{s^-}, X^2_{s^-})) \\ &+ \lambda_2(V(s, X^1_{s^-}, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-})) \\ &+ \lambda_{12}(V(s, X^1_s, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))\Big] ds + \nabla V dW_s \\ &+ (V(s, X^1_s, X^2_{s^-}) - V(s, X^1_{s^-}, X^2_{s^-}))d\tilde{N}_{\{1\}}(s) \\ &+ (V(s, X^1_{s^-}, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))d\tilde{N}_{\{2\}}(s) \\ &+ (V(s, X^1_s, X^2_s) - V(s, X^1_{s^-}, X^2_{s^-}))d\tilde{N}_{\{12\}}(s), \end{aligned}$$

where $V(s, X^1_{s^-}, X^2_{s^-})$ is the solution before jumps occur. Then from the definition of $\mathcal{J}_1 f(x)$ in (C.2) we have $\mathcal{J}_1 V = V(s, X^1_s, X^2_{s^-}) = V(s, X^1_{s^-} + J_1, X^2_{s^-})$. Similar definitions hold for jumps over x_2 and simultaneous jump

over x_1 and x_2. Finally

$$\begin{aligned} dY_s = &\Big[V_t + \Delta_\rho V + \xi \cdot \nabla V - (\lambda_1 + \lambda_2 + \lambda_{12})V \\ &+ \lambda_1 \mathcal{J}_1 V + \lambda_2 \mathcal{J}_2 V + \lambda_{12} \mathcal{J}_{12} V\Big] dt + \nabla V dW \\ &+ (V(s, X_s^1, X_{s^-}^2) - V(s, X_{s^-}^1, X_{s^-}^2)) d\tilde{N}_{\{1\}}(s) \\ &+ (V(s, X_{s^-}^1, X_s^2) - V(s, X_{s^-}^1, X_{s^-}^2)) d\tilde{N}_{\{2\}}(s) \\ &+ (V(s, X_s^1, X_s^2) - V(s, X_{s^-}^1, X_{s^-}^2)) d\tilde{N}_{\{12\}}(s). \end{aligned}$$

From the PIDE (C.1) we have $V_t + \mathcal{L}V = \chi$. Integrating from t to τ we obtain

$$\begin{aligned} Y(\tau) = V(X_\tau, \tau) = &V(x,t) + \int_t^\tau \chi(X_s, s) ds + \int_t^\tau \nabla V dW_s \\ &+ \int_t^\tau (V(s, X_s^1, X_{s^-}^2) - V(s, X_{s^-}^1, X_{s^-}^2)) d\tilde{N}_{\{1\}}(s) \\ &+ \int_t^\tau (V(s, X_{s^-}^1, X_s^2) - V(s, X_{s^-}^1, X_{s^-}^2)) d\tilde{N}_{\{2\}}(s) \\ &+ \int_t^\tau (V(s, X_s^1, X_s^2) - V(s, X_{s^-}^1, X_{s^-}^2)) d\tilde{N}_{\{12\}}(s). \end{aligned}$$

Taking the expectation and, considering that since $\mathbb{E}[\tau \wedge T] < \infty$ by Theorem 2 all the stochastic integrals vanish, we obtain

$$\begin{aligned} V(t,x) &= \mathbb{E}\left[V(X_\tau, \tau) - \int_t^\tau \chi(X_s, s) ds\right] \\ &= \mathbb{E}\left[V(X_T, T)\mathbf{1}_{\{\tau = T\}} - \int_t^T \chi(X_s, s)\mathbf{1}_{\{\tau > s\}} ds\right] \\ &= \mathbb{E}\left[\psi(X_T)\mathbf{1}_{\{\tau = T\}} - \int_t^T \chi(X_s, s)\mathbf{1}_{\{\tau > s\}} ds\right]. \end{aligned}$$

For the boundary, instead, it easy to see that when computing $V(t, 0, x_2)$ we have $\tau_1 = t$ since $x_1 = 0$ and so $\tau = t$. Then

$$\begin{aligned} V(t, 0, x_2) &= \mathbb{E}\left[\phi_{1,0}(\tau_1, X_2(\tau_1)) \cdot \mathbf{1}_{\{\tau_1 < T\}} | X_1(t) = 0, X_2(t) = x_2\right] \\ &= \mathbb{E}\left[\phi_{1,0}(t, X_2(t)) \cdot \mathbf{1}_{\{t < T\}} | X_1(t) = 0, X_2(t) = x_2\right] \\ &= \mathbb{E}\left[\phi_{1,0}(t, x_2) \cdot \mathbf{1}_{\{t < T\}} | X_1(t) = 0, X_2(t) = x_2\right] \\ &= \phi_{1,0}(t, x_2). \end{aligned}$$

Similar results are obtain for the other border condition at $x_2 = 0$. Since our domain is upper unbounded we have no theoretical border conditions

for $x_i \to \infty$, but in order to solve the PIDE these conditions must be given. So we add a suitable conditions depending on the pricing problem.

□

Appendix D

Implementation

```
function [x1,x2] = Grid2D(xMin,xMax,yMin,yMax,Nx,Ny,xBarrier,
    yBarrier)
%creates a heterogeneous 2D-grid considering the two variables
    independent.
% x-grid
Bx=xBarrier;
% custom parameters
alphax =(xMax-Bx)/10;
c1= asinh((xMin-Bx)/alphax);
c2= asinh((xMax-Bx)/alphax);
psi = linspace(0,1,Nx+1);
x1 = Bx+alphax*sinh(c2.*psi+c1.*(1-psi));
% y-grid
By=yBarrier;
% custom parameters
alphay = (yMax-By)/10;
d1= asinh((yMin-By)/alphay);
d2= asinh((yMax-By)/alphay);
nu = linspace(0,1,Ny+1);
x2 = By+alphay*sinh(d2.*nu+d1.*(1-nu));
end
```

```
function solution = SmoothFunction2D( x1,x2,fun)
% Smoothing the function using local averaging method.

N1= length(x1);
dx1 = x1(2:end)-x1(1:end-1);
N2= length(x2);
dx2 = x2(2:end)-x2(1:end-1);
solution = zeros(N1,N2);
for i = 1 : N1
    for j = 1: N2
        Different = FindingDifference(x1,x2,i,j,N1,N2,fun);
        if(Different)
            if(i == 1)
                lower1 = x1(i);
                upper1 = x1(i)+dx1(i)/2;
```

```
16              elseif (i == N1)
17                  lower1 = x1(i)-dx1(i-1)/2;
18                  upper1 = x1(i);
19              else
20                  lower1 = x1(i)-dx1(i-1)/2;
21                  upper1 = x1(i)+dx1(i)/2;
22              end
23              if(j == 1)
24                  lower2 = x2(j);
25                  upper2 = x2(j)+dx2(j)/2;
26              elseif(j == N2)
27                  lower2 = x2(j)-dx2(j-1)/2;
28                  upper2 = x2(j);
29              else
30                  lower2 = x2(j)-dx2(j-1)/2;
31                  upper2 = x2(j)+dx2(j)/2;
32              end
33              solution(i,j) = 1/ ((upper1-lower1)*(upper2-lower2))
     *integral2(fun,lower1,upper1,lower2,upper2,'Method','
    iterated');
34          else
35              solution(i,j) = fun(x1(i),x2(j));
36          end
37      end
38  end
39  solution= solution';
40  solution = solution(:);
41  end
```

```
1  function Res = Lipton2D( par, N1,N2, M, x1, x2, BC,
       TimeCondition,InterPayments,NeumannBorder)
2  % Discretization of the PIDE.
3
4  % Crank Nicholson scheme
5  %Differential discretization
6  A = DifferentialDiscretization(x1,x2,par);
7  % Time discretization
8  Res = TimeDiscretization2D(N1,N2,M,par.T,A,TimeCondition,BC,par.
       lambda1,par.lambda2,par.lambda12,par.psi1,par.psi2,x1,x2,
       InterPayments,NeumannBorder);
9
10 % ADI scheme
11 [A1, A2, D12] = DifferentialDiscretizationADI(x1,x2,par);
12 Res =HV_scheme(par,par.HV,M,par.T,D12,A1,A2,TimeCondition,
       InterPayments,BC,x1,x2);
13
14 end
```

```
1  function A = DifferentialDiscretization(x1,x2,par)
2  % Computes the discretization of the spacial differental part of
       the PIDE
3
4  N1 = length(x1);
5  N2 = length(x2);
```

```
%Boundary indices
numpoints=N1*N2;
node_W=1:N2;
node_E=N2*(N1-1)+(1:N2);
node_S=1:N2:numpoints;
node_N=N2:N2:numpoints;
node_bound=sort(unique([ node_W node_E node_S node_N ]));
% Coefficents
dx1 = x1(2:end)-x1(1:end-1);
dx2 = x2(2:end)-x2(1:end-1);
beta_1_under1 = (-dx1(2:end)./(dx1(1:end-1).*(dx1(1:end-1)+dx1(2:end))));
beta_1_0 = ((dx1(2:end)-dx1(1:end-1))./(dx1(1:end-1).*dx1(2:end)));
beta_1_over1 = (dx1(1:end-1)./(dx1(2:end).*(dx1(1:end-1)+dx1(2:end))));
delta_1_0 = (-2./(dx1(1:end-1).*dx1(2:end)));
delta_1_under1 = (2./(dx1(1:end-1).*(dx1(1:end-1)+dx1(2:end))));
delta_1_over1 = (2./(dx1(2:end).*(dx1(1:end-1)+dx1(2:end))));
beta_2_under1 = (-dx2(2:end)./(dx2(1:end-1).*(dx2(1:end-1)+dx2(2:end))));
beta_2_0 = ((dx2(2:end)-dx2(1:end-1))./(dx2(1:end-1).*dx2(2:end)));
beta_2_over1 = (dx2(1:end-1)./(dx2(2:end).*(dx2(1:end-1)+dx2(2:end))));
delta_2_0 =(-2./(dx2(1:end-1).*dx2(2:end)));
delta_2_under1 = (2./(dx2(1:end-1).*(dx2(1:end-1)+dx2(2:end))));
delta_2_over1 = (2./(dx2(2:end).*(dx2(1:end-1)+dx2(2:end))));

%% Matrix
A=sparse(numpoints,numpoints);
for i=1:numpoints
    index1 = fix((i-1)/N2);
    index2 = mod(i,N2)-1;
    if min( abs(i-node_bound) )>0 % interior node
        % No derivative
        A(i,i)=-(par.lambda1+par.lambda2+par.lambda12);
        %A(i,i)=0;
        % First derivative (x2)
        coeff=par.epsilon2;
        A(i,i) = A(i,i) + coeff*beta_2_0(index2);
        A(i,i+1)=coeff*beta_2_over1(index2);
        A(i,i-1)=coeff*beta_2_under1(index2);
        % First derivative (x1)
        coeff=par.epsilon1;
        A(i,i) = A(i,i) + coeff*beta_1_0(index1);
        A(i,i+N2)=coeff*beta_1_over1(index1);
        A(i,i-N2)=coeff*beta_1_under1(index1);
        % Second derivative (x2)
        coeff=0.5*(par.sigma2)^2;
        A(i,i+1)=A(i,i+1)+coeff*delta_2_over1(index2);
        A(i,i)=A(i,i)+coeff*delta_2_0(index2);
        A(i,i-1)=A(i,i-1)+coeff*delta_2_under1(index2);
        % Second derivative (x1)
```

```
54          coeff=0.5*(par.sigma1)^2;
55          A(i,i+N2)=A(i,i+N2)+coeff*delta_1_over1(index1);
56          A(i,i)=A(i,i)+coeff*delta_1_0(index1);
57          A(i,i-N2)=A(i,i-N2)+coeff*delta_1_under1(index1);
58          % Mixed derivative
59          coeff=par.rho*par.sigma1*par.sigma2;
60          A(i,i)      = A(i,i)    + coeff*beta_1_0(index1)*beta_2_0(
        index2);
61          A(i,i+1)    = A(i,i+1) + coeff*beta_1_0(index1)*
        beta_2_over1(index2);
62          A(i,i-1)    = A(i,i-1) + coeff*beta_1_0(index1)*
        beta_2_under1(index2);
63          A(i,i+N2)   = A(i,i+N2)+ coeff*beta_1_over1(index1)*
        beta_2_0(index2);
64          A(i,i+N2+1)=              coeff*beta_1_over1(index1)*
        beta_2_over1(index2);
65          A(i,i+N2-1)=              coeff*beta_1_over1(index1)*
        beta_2_under1(index2);
66          A(i,i-N2)   = A(i,i-N2)+ coeff*beta_1_under1(index1)*
        beta_2_0(index2);
67          A(i,i-N2+1)=              coeff*beta_1_under1(index1)*
        beta_2_over1(index2);
68          A(i,i-N2-1)=              coeff*beta_1_under1(index1)*
        beta_2_under1(index2);
69      end
70  end
71  end
```

```
1   function J  = JumpVector(N1,N2,x1,zeta1,x2,zeta2,lambda1,lambda2
        ,lambda12,U)
2   % Computes the jump contribution
3   J1 = zeros(N1*N2,1);
4   J2 = zeros(N1*N2,1);
5   I12 = zeros(N1*N2,1);
6   J12 = zeros(N1*N2,1);
7
8   h1 = x1(2:end)-x1(1:end-1);
9   h2 = x2(2:end)-x2(1:end-1);
10
11  %First choose of parameters
12  % J1omega0 = 0.5*h1*zeta1.*exp(-zeta1*h1);
13  % J1omega1 = 0.5*h1*zeta1;
14  % J2omega0 = 0.5*h2*zeta2.*exp(-zeta2*h2);
15  % J2omega1 = 0.5*h2*zeta2;
16  % %Second choose of parameters
17  %omega0(zeta1,h1)
18  J1omega0 = (1-(1+zeta1*h1).*exp(-zeta1*h1))./(zeta1*h1);
19  %omega1(zeta1,h1)
20  J1omega1 = (-1+zeta1*h1+exp(-zeta1*h1))./(zeta1*h1);
21  %omega0(zeta2,h2)
22  J2omega0 = (1-(1+zeta2*h2).*exp(-zeta2*h2))./(zeta2*h2);
23  %omega1(zeta2,h2)
24  J2omega1 = (-1+zeta2*h2+exp(-zeta2*h2))./(zeta2*h2);
25
```

```
for i = 1 : N1*N2
    index1 = fix((i-1)/(N2));
    index2 = mod((i-1),N2);
    %J1 & I12
    if(i<=N2)
        J1(i)=0;
        I12(i) = 0;
    else
        J1(i) = exp(-zeta1*h1(index1))*J1(i-N2)+J1omega0(index1)*U(i-N2)+J1omega1(index1)*U(i);
        I12(i) = exp(-zeta1*h1(index1))*I12(i-N2)+J1omega0(index1)*U(i-N2)+J1omega1(index1)*U(i);
    end
    %J2 & J12
    if(mod(i,N2)==1)
        J2(i) = 0;
        J12(i) = 0;
    else
        J2(i) = exp(-zeta2*h2(index2))*J2(i-1)+J2omega0(index2)*U(i-1)+J2omega1(index2)*U(i);
        J12(i) = exp(-zeta2*h2(index2))*J12(i-1)+J2omega0(index2)*I12(i-1)+J2omega1(index2)*I12(i);
    end
end
J = lambda1*J1+lambda2*J2+lambda12*J12;
end
```

```
function Correction = IntegralCorrection2D(x1,x2,BC,theta1,theta2,lambda1,lambda2,lambda12)
% Correction of jump operator
N1 = length(x1);
N2 = length(x2);
M = size(BC.N,2);
J1Correction= zeros(N1*N2,M);
J2Correction= zeros(N1*N2,M);
First= zeros(N1*N2,M);
Second= zeros(N1*N2,M);
Third= zeros(N1*N2,M);
IsConstantW = CheckForConstant(BC.W(:,1));
IsConstantS = CheckForConstant(BC.S(:,1));
for t = 1: M
    %J1 correction
    for i = 1:N1
        J1Correction((i-1)*N2+1:i*N2,t) = BC.W(:,t).*exp(-theta1*x1(i));
    end
    %J2 correction
    for i = 1 :N2
        J2Correction(i:N2:N1*N2,t) = BC.S(:,t).*exp(-theta2*x2(i));
    end
    %J12 correction
    for i = 1:N1
        for j = 1:N2
```

```
25              u1 = x1(1:i)';
26              u2 = x2(1:j)';
27              if(length(u1) == 1)
28                  Second(j+N2*(i-1),t) = 0;
29                  if(length(u2) == 1)
30                      First(j+N2*(i-1),t)  = 0;
31                  else
32                      if(IsConstantW)
33                          First(j+N2*(i-1),t)  = BC.W(1,t)*exp(-
    theta1*x1(i))*(1-exp(-theta2*x2(j)));
34                      else
35                          First(j+N2*(i-1),t)  = exp(-theta1*x1(i)
    )* theta2*trapz(u2,BC.W(j:-1:1,t).*exp(-theta2*u2));
36                      end
37                  end
38                  Third(j+N2*(i-1),t)  = BC.W(1,t)*exp(-theta2*x2(
    j))*exp(-theta1*x1(i));
39              else
40                  if(IsConstantS)
41                      Second(j+N2*(i-1),t) = BC.S(1,t)*exp(-theta2
    *x2(j))*(1-exp(-theta1*x1(i)));
42                      if(length(u2) == 1)
43                          First(j+N2*(i-1),t)  = 0;
44                      else
45                          if(IsConstantW)
46                              First(j+N2*(i-1),t)  = BC.W(1,t)*exp
    (-theta1*x1(i))*(1-exp(-theta2*x2(j)));
47                          else
48                              First(j+N2*(i-1),t)  = exp(-theta1*
    x1(i))* theta2*trapz(u2,BC.W(j:-1:1,t).*exp(-theta2*u2));
49                          end
50                      end
51                      Third(j+N2*(i-1),t)  = BC.W(1,t)*exp(-theta2
    *x2(j))*exp(-theta1*x1(i));
52                  else
53                      Second(j+N2*(i-1),t) = exp(-theta2*x2(j))*
    theta1*trapz(u1,BC.S(i:-1:1,t).*exp(-theta1*u1));
54                      if(length(u2) == 1)
55                          First(j+N2*(i-1),t)  = 0;
56                      else
57                          if(IsConstantW)
58                              First(j+N2*(i-1),t)  = BC.W(1,t)*exp
    (-theta1*x1(i))*(1-exp(-theta2*x2(j)));
59                          else
60                              First(j+N2*(i-1),t)  = exp(-theta1*
    x1(i))* theta2*trapz(u2,BC.W(j:-1:1,t).*exp(-theta2*u2));
61                          end
62                      end
63                      Third(j+N2*(i-1),t)  = BC.W(1,t)*exp(-theta2
    *x2(j))*exp(-theta1*x1(i));
64                  end
65              end
66          end
67
```

```
68      end
69  end
70  J12Correction = First+Second+Third;
71  Correction = lambda1*J1Correction+lambda2*J2Correction+lambda12*
        J12Correction;
```

```
1   function Res = TimeDiscretization2D(N1,N2,M,T,A,InitialCondition
        ,BC,lambda1,lambda2,lambda12,zeta1,zeta2,x1,x2,InterPayments,
        NeumannBorder)
2
3   dt = T/M;
4   Upre = InitialCondition ;
5   Res = zeros(length(Upre),M+1);
6   Res(:,1) = Upre;
7   C = speye(size(Upre,1));
8   %% Border Pre setting & Matrix Correction
9   %Finding border index
10  [NeuIndex, DirIndex,A,boundary] = FindingNeuman(N1,N2,A,BC,dt,
        NeumannBorder);
11  boundIndex = sort(unique([NeuIndex DirIndex]));
12  B = zeros(N1*N2,M+1);
13  % Computing integral correction
14
15  if(lambda1 == 0 && lambda2 == 0 && lambda12 == 0)
16      %No jump
17      Correction = zeros(N1*N2,M+1);
18  else
19      % Jump
20      Correction = IntegralCorrection2D(x1,x2,BC,zeta1,zeta2,
        lambda1,lambda2,lambda12);
21  end
22  % Creating and correction for boundary conditions
23  for i = 1 : length(DirIndex)
24      % Vector of BC
25      B(DirIndex(i),:)=boundary(i,:);
26      Correction(DirIndex(i),:)=zeros(1,size(Correction,2));
27  end
28  B =B -dt*InterPayments+dt*Correction;
29  % Computing solution
30  [LA, UA] = lu((speye(N1*N2)-dt*A));
31  h = waitbar(0,'Please wait...');
32  for i = 1 : M
33      %Computing Jump contribution
34      J   = JumpVector(N1,N2,x1,zeta1,x2,zeta2,lambda1,lambda2,
        lambda12,Upre);
35      %Correcting in order to impose border condition
36      Upre(boundIndex)    = zeros(length(boundIndex),1);
37      J(boundIndex)= zeros(length(boundIndex),1);
38      % Solving the problem
39      U = UA\(LA\((C*Upre+B(:,i+1)+dt*J)));
40      Upre = U;
41      Res(:,i+1) = U ;
42      waitbar(i/M,h)
43  end
```

```
44 close(h)
45 end
```

```
1  function [D1, D2, D12]= DifferentialDiscretizationADI(x1,x2,par)
2  % Computes the discretization of the spacial differental part of
       the PIDE
3
4  N1 = length(x1);
5  N2 = length(x2);
6  % Index boundary
7  numpoints=N1*N2;
8  node_W=1:N2;
9  node_E=N2*(N1-1)+(1:N2);
10 node_S=1:N2:numpoints;
11 node_N=N2:N2:numpoints;
12 node_bound=sort(unique([ node_W node_E node_S node_N ]));
13 % Coefficents
14 dx1 = x1(2:end)-x1(1:end-1);
15 dx2 = x2(2:end)-x2(1:end-1);
16 beta_1_under1 = (-dx1(2:end)./(dx1(1:end-1).*(dx1(1:end-1)+dx1
       (2:end))));
17 beta_1_0 = ((dx1(2:end)-dx1(1:end-1))./(dx1(1:end-1).*dx1(2:end)
       ));
18 beta_1_over1 = (dx1(1:end-1)./(dx1(2:end).*(dx1(1:end-1)+dx1(2:
       end))));
19 delta_1_0 = (-2./(dx1(1:end-1).*dx1(2:end)));
20 delta_1_under1 = (2./(dx1(1:end-1).*(dx1(1:end-1)+dx1(2:end))));
21 delta_1_over1 = (2./(dx1(2:end).*(dx1(1:end-1)+dx1(2:end))));
22 beta_2_under1 = (-dx2(2:end)./(dx2(1:end-1).*(dx2(1:end-1)+dx2
       (2:end))));
23 beta_2_0 = ((dx2(2:end)-dx2(1:end-1))./(dx2(1:end-1).*dx2(2:end)
       ));
24 beta_2_over1 = (dx2(1:end-1)./(dx2(2:end).*(dx2(1:end-1)+dx2(2:
       end))));
25 delta_2_0 =(-2./(dx2(1:end-1).*dx2(2:end)));
26 delta_2_under1 = (2./(dx2(1:end-1).*(dx2(1:end-1)+dx2(2:end))));
27 delta_2_over1 = (2./(dx2(2:end).*(dx2(1:end-1)+dx2(2:end))));
28 % Matrix
29 D1=sparse(numpoints,numpoints);
30 D2=sparse(numpoints,numpoints);
31 D12=sparse(numpoints,numpoints);
32 for i=1:numpoints
33     index1 = fix((i-1)/N2);
34     index2 = mod(i,N2)-1;
35     if min( abs(i-node_bound) )>0
36         % interior node
37         % No derivative
38         D1(i,i)=-(par.lambda1+par.lambda12/2);
39         D2(i,i)=-(par.lambda2+par.lambda12/2);
40         % First derivative (x2)
41         coeff=par.epsilon2;
42         D2(i,i) = D2(i,i) + coeff*beta_2_0(index2);
43         D2(i,i+1)=coeff*beta_2_over1(index2);
44         D2(i,i-1)=coeff*beta_2_under1(index2);
```

```
45          % First derivative (x1)
46          coeff=par.epsilon1;
47          D1(i,i) = D1(i,i) + coeff*beta_1_0(index1);
48          D1(i,i+N2)=coeff*beta_1_over1(index1);
49          D1(i,i-N2)=coeff*beta_1_under1(index1);
50          % Second derivative (x2)
51          coeff=0.5*(par.sigma2)^2;
52          D2(i,i+1)=D2(i,i+1)+coeff*delta_2_over1(index2);
53          D2(i,i)=D2(i,i)+coeff*delta_2_0(index2);
54          D2(i,i-1)=D2(i,i-1)+coeff*delta_2_under1(index2);
55          % Second derivative (x1)
56          coeff=0.5*(par.sigma1)^2;
57          D1(i,i+N2)=D1(i,i+N2)+coeff*delta_1_over1(index1);
58          D1(i,i)=D1(i,i)+coeff*delta_1_0(index1);
59          D1(i,i-N2)=D1(i,i-N2)+coeff*delta_1_under1(index1);
60          % Mixed derivative
61          coeff=par.rho*par.sigma1*par.sigma2;
62          D12(i,i)      = D12(i,i)    + coeff*beta_1_0(index1)*
        beta_2_0(index2);
63          D12(i,i+1)    = D12(i,i+1) + coeff*beta_1_0(index1)*
        beta_2_over1(index2);
64          D12(i,i-1)    = D12(i,i-1) + coeff*beta_1_0(index1)*
        beta_2_under1(index2);
65          D12(i,i+N2)   = D12(i,i+N2)+ coeff*beta_1_over1(index1)*
        beta_2_0(index2);
66          D12(i,i+N2+1)=              coeff*beta_1_over1(index1)*
        beta_2_over1(index2);
67          D12(i,i+N2-1)=              coeff*beta_1_over1(index1)*
        beta_2_under1(index2);
68          D12(i,i-N2)   = D12(i,i-N2)+ coeff*beta_1_under1(index1)*
        beta_2_0(index2);
69          D12(i,i-N2+1)=              coeff*beta_1_under1(index1)*
        beta_2_over1(index2);
70          D12(i,i-N2-1)=              coeff*beta_1_under1(index1)*
        beta_2_under1(index2);
71      else
72          %Boundary
73          D1(i,i) = 1;
74          D2(i,i) = 1;
75          D12(i,i) = 1;
76      end
77  end
```

```
1  function Res = HV_scheme(par,HV,M,T,D12,D1,D2,InitialCondition,
       InterPayments,BC,x1,x2)
2  % Hundsdorfer-Verwer scheme
3
4  N1 = length(x1);
5  N2 = length(x2);
6  dt = T/M;
7  Upre = InitialCondition ;
8  Res = zeros(length(Upre),M+1);
9  Res(:,1) = Upre;
10 % Boundary index
```

```
west  = 1 : N2;
south = 1 : N2 : N1*N2 ;
east = N2*(N1-1)+1 : N1*N2;
north = N2: N2 : N1*N2;
DirIndex = sort(unique([west south east north]));

%Correction jump operator
Correction = IntegralCorrection2D(x1,x2,BC,par.psi1,par.psi2,par
    .lambda1,par.lambda2,par.lambda12);
for i = 1 : length(DirIndex)
    Correction(DirIndex(i),:)=zeros(1,size(Correction,2));
end
b0 = -InterPayments+ Correction;
b1 = zeros(size(b0));
b2 = zeros(size(b0));
% Computing solution
[L1,U1] = lu(speye(size(D1))-HV.theta*dt*D1);
[L2,U2] = lu(speye(size(D2))-HV.theta*dt*D2);
h = waitbar(0,'Please wait...');
for i = 1 : M
    J = JumpVector(N1,N2,x1,par.psi1,x2,par.psi2,par.lambda1,par
    .lambda2,par.lambda12,Upre);
    Y0 = Upre + dt * ((D12+D1+D2)*Upre + J + b0(:,i)+b1(:,i)+b2
    (:,i));
    Y0(DirIndex) = boundary(D12,D1,D2,J,b0,b1,b2,BC,DirIndex,HV,
    dt,i);
    Y1 = U1\(L1\(Y0 + HV.theta*dt*(-D1*Upre + b1(:,i+1) - b1(:,i
    ))));
    Y2 = U2\(L2\(Y1 + HV.theta*dt*(-D2*Upre + b2(:,i+1) - b2(:,i
    ))));
    J_over_Y2 = JumpVector(N1,N2,x1,par.psi1,x2,par.psi2,par.
    lambda1,par.lambda2,par.lambda12,Y2);
    Y0tilde = Y0 + HV.sigma*dt*((D12+D1+D2)*Y2+ J_over_Y2+ b0(:,
    i+1)+b1(:,i+1)+b2(:,i+1) - (D12+D1+D2)*Upre -J - b0(:,i)-b1
    (:,i)-b2(:,i));
    Y1tilde = U1\(L1\(Y0tilde-HV.theta*dt*D1*Y2));
    U = U2\(L2\(Y1tilde-HV.theta*dt*D2*Y2));
    Upre = U;
    Res(:,i+1) = U ;
    waitbar(i/M,h)
end
close(h)
end
```